Shimmer User Manual

Revision 2R.d

Legal Notices and Disclaimers

Disclaimers

CAUTION

RISK OF EXPLOSION IF BATTERY IS REPLACED

BY AN INCORRECT TYPE.

DISPOSE OF USED BATTERIES ACCORDING

TO THE INSTRUCTIONS.

Informed Consent Statement

As is the nature of a research prototype, Shimmer is an experimental device and consequently not fully tested to commercial product nor medical class regulatory approved standards. Consequently, although great care was taken in the design of this device, there is some inherent risk both with the design and manufacturing that you assume when the device is in close proximity to your body or the body of your test subjects:

These Risks Include:

There is a risk of electrical shock due to manufacturing defects or improper use (see usage guidelines and warnings).

There is also a risk of sustaining a burn due to a catastrophic failure of the device which could result from overheating of components.

There is a risk of radio interference with the operation of other electronic devices and we make no claims to the consequences of this.

There is a risk of some minor skin irritation from electrode pads over prolonged periods of time which may cause discomfort.

The device is not designed with proper safeguards for proper defibrillation. As such, electrodes must be removed before defibrillation is attempted.

Data privacy limitations: It should be understood from the outset and you should communicate to test subjects that the physiological data that is streamed, stored, and analyzed through use of the device is not anonymized or privacy-protected in any way and you should take appropriate precautions in the protection and handling of such data in your research activities. Shimmer itself may buffer raw physiological data unencrypted on the integrated flash memory device. RF data streaming from the Shimmer may not be encrypted and could be intercepted by others. RF data downstream from an aggregation device, such as a cell phone may not be encrypted and is likewise susceptible to access.

Physiological data generated through use of Shimmer may indicate conditions that your test subject was previously unaware of prior to participation in research using the device.

There may be a risk of exposure to minute amounts of chemicals from the manufacturing process or the components themselves (such as latex, lead etc.).

There may be an increased risk of physical injury by the physical presence of the device on a test subject's body and you fully assume this responsibility.

Realtime Technologies Ltd is not liable for damage or loss of data when using the MicroSD card feature

Some Shimmer peripherals rely on 3rd party driver support. While Realtime/Shimmer Research have tested features using a typical system, in some cases the end user will need to contact the peripheral vendor for resolution of installation, compatibility, or operational issues.

By your use of Shimmer you acknowledge these and other risks inherent in the use of an experimental device and you assume full responsibility for testing this device with human subjects.

This device cannot be marketed or put into service within the EU until it has been made to comply with the Medical Devices Directive 93/42/EEC. In the United States, Shimmer is an investigational device, limited by US law to investigational uses.

Glossary

ACLK	Auxiliary Clock
ADC	Analogue-to-Digital Converter
AnEx	Analogue External Connector Breakout board
BSL	Boot Strap Loader
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access
CVS	Concurrent Versioning System
DCO	Digitally Controlled Oscillator
DMA	Direct Memory Access
ECG	Electrocardiogram
ECG	Electrocardiogram
EEG	Electroencephalogram
EMG	Electromyogram
EOG	Electrooculography
FHSS	Frequency Hopping Spread Spectrum
GPIO	General Purpose Input/Output
GSR	Galvanic Skin Response
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers
ISM	Industrial, Scientific and Medical Band
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MAB	Memory Address Bus
MAC	Media Access Control
MCLK	Master Clock
MDB	Memory Data Bus
BioMOBIUS	Multimodal Open-architecture Bio-sensor Interface User-interaction System.
MPR	Microprocessor and Radio Boards

MTS	Mote Sensor
nesC	network embedded system C
PAN	Personal Area Network
PWM	Pulse Width Modulation
RAM	Random Access Memory
RF	Radio Frequency
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RPM	RedHat Package Manager
SD	Secure Digital
SDK	Software Development Kit
SIG	Special Interest Group
SIMO	Slave In, Master Out
SIP	Session Initiation Protocol
SMCLK	Sub-Main Clock
SOMI	Slave Out, Master In
SPI	Serial Peripheral Interface
SPP	Serial Port Profile
STE	Slave Transmit Enable
TCP/IP	Transmission Control Protocol / Internet Protocol
TinyOS	Tiny Operating System
UART	Universal Asynchronous Receiver/Transmitter
UCLK	USART Clock
USART	Universal Serial Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WBSN	Wireless Based Sensor Network
WSN	Wireless Sensor Network

Agency Compliance

Contains FCC ID: X2W-SR7-1
Contains FCC ID: T9-RN42

The FCC ID marking label is to remain affixed at all times to comply with FCC requirements for Modular approval:

```
Model: Shimmer2R
 Contains: FCC ID: X2W-SR7-1;
                   T9-RN42
              IC : 8838A-SR71;
                   6514A-RN42
 Blue-Tooth ID: 
Made in Ireland
```

This device complies with Part 15 of the FCC Rules.
Operation is subject to the following two conditions:
(1) This device may not cause harmful interference, and
(2) This device must accept any interference received,
including interference that may cause undesired operation.

RADIO AND TELEVISION INTERFERENCE

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.

- Increase the separation between the equipment and the receiver.

- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

- Consult the dealer or an experienced radio/TV technician for help.

Changes and Modifications not expressly approved by Realtime Technologies Ltd. can void your authority to operate this equipment under Federal Communications Commission rules.

ICES-003 Label

This Class (*) digital apparatus complies with Canadian ICES-003
Cet appareil numérique de la classe (*) est conforme à la norme
NMB-003 du Canada

(*) Insert either “A” or “B” but not both as appropriate for the equipment requirements

Welcome message

Thank you for your purchase of Shimmer! This guide will help you to understand the capabilities of the Shimmer platform and provide essential operating instructions. Shimmer ships with a print or electronic *Readme* document. The latest version of this manual is also in the downloads area at **http://www.shimmer-research.com**

If you are new to Shimmer, there is quick start information in Section 2. The remaining tutorials and technical content are best browsed based on personal interest and development goals.

If you have questions or need support you can always contact us, or use the Shimmer-users mailing list.

info@shimmer-research.com	(non-technical questions)
support@shimmer-research.com	(technical questions)
labview@shimmer-research.com	(labview questions)

http://www.shimmer-research.com

https://www.eecs.harvard.edu/mailman/listinfo/shimmer-users

Best wishes,

The Shimmer Research Team

Table of Contents

1 Introduction and Platform Overview

1.1 Introduction

Shimmer is a small sensor platform well suited for wearable applications. The integrated 3-axis accelerometer, large storage, and low-power standards based communication capabilities enable emerging applications in motion capture, long-term data acquisition, and real-time monitoring.

Shimmer2R, the latest revision of the Shimmer platform adds improvements based on years of field trials and deployments. There is a direct-to-host data transfer path from Shimmer's MicroSD card to hasten data offload, minimize user training, and provide fail-safe data access. Power control features have been added including soft-power switching, power monitoring and low-battery shutdown. A passive vibration sensor efficiently signals when the device is in use. Finally, Shimmer2R uses radio modules that have FCC, ETSI, and Industry Canada certification.

Shimmer, Shimmer2 and Shimmer2R are used interchangeably in this manual. Much of the information applies generically to Shimmer Research Products. For details on legacy devices see Section 9.

1.2 Shimmer Platform Key Features

Feature	Purpose	Component/Capabilities
I/O	Capture of sensor and user data	**Integrated** • 3 Axis Accelerometer using Freescale MMA7361 1.5/6g MEMs Accelerometer (MMA7331 4g/12 is a factory option) • 3 Colored Status LEDs • Soft-power button (short press for Reset, hold for 10 seconds to power-down board) • SignalQuest SQ-SEN200 Passive MEMs Omni directional tilt and vibration sensor **Expansion** • Hirose ST60 series 18 position rugged mobile style external Header for charging, programming, and tethered sensor extensions (12 multi-purpose I/O connections). • Hirose DF12 series 20 position internal Expansion Header for internal sensor daughter boards (14 multi-purpose I/O connections) • Unregulated battery/dock voltage on expansion connectors for improved efficiency and operation of high powered peripherals • JTAG test pads **Expansion Boards** • Kinematics: Gyro and/or Magnetic sensing • 3 Lead ECG • EMG • Infrared Motion Detection • "AnEx" Prototyping breakout board • Galvanic Skin Response (GSR) • Strain Gauge Bridge (2011)
Processing	Control operating state Provide best Signal quality Operational alerts and messages	**MSP430F1611 CPU** ▪ 10Kbyte RAM, 48Kbyte Flash ▪ Up to 8MHz ▪ 2 DAC outputs ▪ 8 12bit A/D inputs ▪ Extremely low power during periods of inactivity ▪ Proven solution in medical sensing applications
Storage	No loss of data while mobile, during network outages or while changing batteries	**MicroSD slot** ▪ Up to 2GByte currently available ▪ Full-speed host transfer when docked (requires use of Shimmer Dock). ▪ Soft-power control

Communication	Hi-reliability Standards Based Mobility	**802.15.4 Radio** ▪ Shimmer Research SR7 (CC2420) **Class 2 Bluetooth Radio** ▪ Roving Networks RN-42 ▪ Soft-power control
Form factor	Wearable	▪ Size in enclosure: 53mmx32mmx 15mm ▪ Modular 3-piece enclosure
Operating Life and Power	Long operating life Easy maintenance and deployment	▪ 450mAh battery ▪ Integrated Li-ion battery charger ▪ Scaled battery voltage measurement ▪ Instantaneous current measurement. ▪ Soft power button with automatic low-battery system shutdown **Accessories** ▪ 6-Shimmer stand alone multi-charger ▪ USB Reader dock ▪ Span USB 802.15.4 Radio Adapter ▪ Straps ▪ Uses standard cables, leads, electrodes

1.3 Connections and Controls

1.3.1 Operating the Reset/Power switch

Shimmer arrives from the factory with power switched off. To power-on or reset Shimmer, press the reset button briefly. To power-off Shimmer, depress reset button for 10 seconds. Shimmer applications and board peripherals are all restarted after either a reset or power-on. Depending on which type of case you specified you may need to use a paper-clip:

Case Type	Reset/Power Method
top with pinhole	Use a paper-clip to depress the button under the pinhole
Sealed top	Depress the reset button on the Shimmer USB dock

The power-of feature of Shimmer is intended for storage or transportation.

<u>A Shimmer that has been powered off will still charge</u>.

If you wish to leave a Shimmer in a low-power state without turning off the device program the Shimmer with the Sleep application or activate the Sleep control from a software application. Shimmer uses re-chargeable Lithium batteries-- these batteries will self-discharge at a faster rate than conventional batteries. Some Shimmer applications include a low-battery indicator using an LED indicator (located near the reset button). There is also hardware low-battery protection which will prevent damage to the battery or Shimmer device. If your application fails to start or terminates abruptly after initiating streaming data over a radio or writing to flash it is probably a symptom of a battery in need of recharging.

To meet the connector ratings and thermal restrictions of a small design, Shimmer charges at 125mA/hr and conditions at 12.5mA/hr. Using these figures you can calculate charge times for a larger or smaller battery.

Likewise, there are cases where an application may draw more than the 12.5mA conditioning current (for example by running the a radio). If you are having trouble charging a Shimmer with a fully discharged battery try programming sleep or blink.

The charger has a 7hr time-out. Users using >800mAh capacity rechargeable batteries need to increase the charge rate on the board to ensure a full charge.

1.3.2 Charging Shimmer

Make sure your dock or multi-charger is plugged in before inserting Shimmer!

The External "Dock" Connector is used to recharge Shimmer with either a USB dock or multi-charger accessory. The connector is keyed and does not require force to insert-- forcing the connector will cause permanent damage to your Shimmer. Insert the Shimmer with the Bluetooth ID label facing away from the USB cable. Once docked, the green LED on the Power/Reset button on the dock will light up indicating that the Shimmer has been correctly docked and is powered. If this LED is not lit press this button briefly to power on the Shimmer. The charging LED indicator operates as follows:

- When this LED is unlit it indicates that the Shimmer is either charged, or in a battery conditioning cycle (completely discharged-- rare).
- The led will turn on (yellow) during the normal charging cycle.

Normal charge time is approximately 3.5 hours, but for an extremely discharged battery there could be a lengthy pre-charge battery conditioning period. If the indicator briefly switches erratically from on to off, there is no cause for concern-- it is just transitioning between conditioning and charging state. Likewise, if the indicator switches from on to off in a more deliberate pattern, it is likely a side effect of a higher powered application (for example running Bluetooth, powering gyros, and writing to the microSD card). For the most efficient recharging after a deep discharge, it is recommended to program the sleep application.

The dock connector is also used to reprogram, and exchange data with a host computer. Some Shimmer Expansions also plug into this connector.

1.3.3 LED Indicators

Three software-controlled LED indicators are available (Green, Yellow, and Red). Developers are encouraged to follow international standards for indicator lights:

- Red is used exclusively to indicate a warning or danger and the requisite need for urgent action.
- Yellow is used to indicate caution or attention required.

In some configurations of Shimmer, the LEDs may be blocked by the case or an expansion board.

1.3.4 Installing Internal Expansion boards and a MicroSD card

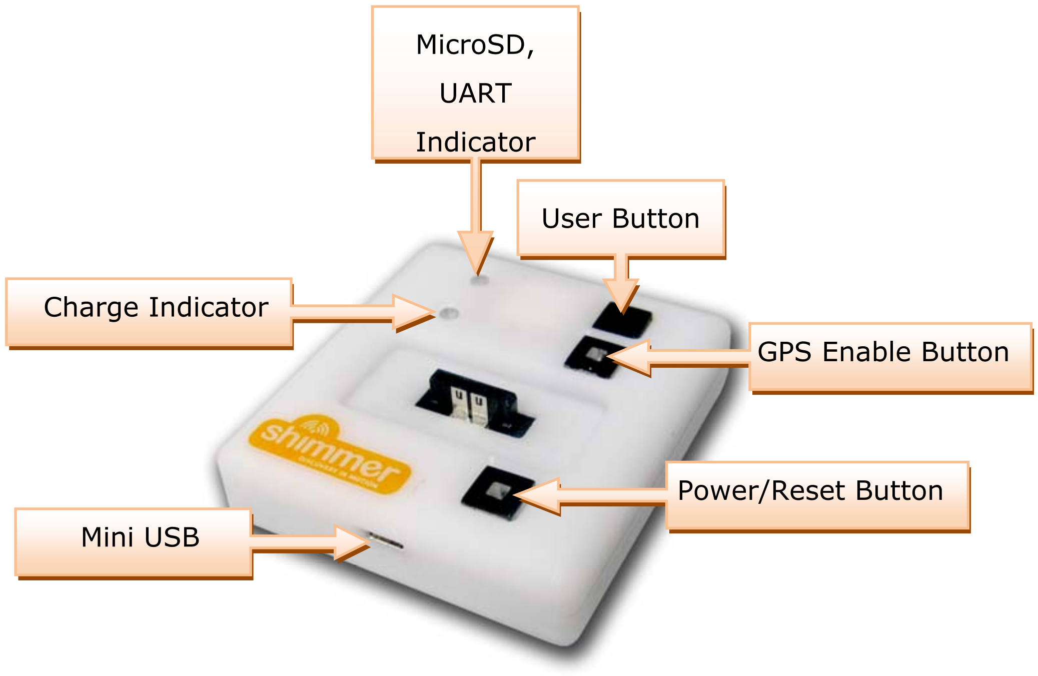
To open the case, carefully loosen the two screws with a small screwdriver and open the case carefully noting the position of the PCB, wires, reset actuator (if present), and battery for re-assembly. Between the PCB and the battery you will find the MicroSD card socket. It is recommended to install the flash card-- as Shimmer has a power switch on this component, there is no drawback to leaving a card installed at all times. The card socket is spring-loaded and you can install the card from the side without complete disassembly.

The white internal expansion connector is on the opposite side of the Shimmer (under the logo sticker). Depending on which expansion you are installing, you may need to install a new top. The internal expansion connector is not keyed. Boards must be installed in the correct orientation. Refer to documentation received with the expansion board, online images or more generally, orient the expansion board so that it doesn't extend beyond the Shimmer PCB outline or sides of the case. Please take care to ensure that the board is parallel to the Shimmer PCB and that no conductive components are touching before closing up the case.

1.3.5 Using the USB Dock Accessories

While there have been several Dock design iterations, the current dock is the "Shimmer Dock 2" which is best identified by a square white enclosure, with 3 black buttons. The docks use class-compliant USB peripherals that are supported by their respective vendors (FTDI, SMSC, etc).

In addition to the charge-status LED, docks contain a HOST data activity indicator, power/reset, user button and GPS enable button.



Item	Function Overview
Power/Reset Button	Power-on, Reset, or Power-off Shimmer
Power/Reset Button Indicator	**Solid Green** when Shimmer is powered on
GPS Enable Button	Toggle between GPS and Host PC UART connection to Shimmer
GPS Enable Button Indicator	**Solid Green**: GPS mode with position fix. **Blinking Green** : GPS mode w/o position fix **Off**: UART mode
User Button	Application specific signal to Shimmer
Charge Indicator	**Solid Yellow** during primary charging state
MicroSD Indicator	**Blinks Blue** with host PC access to Shimmer microSD
UART Indicator	**Blinks Orange** during programming (BSL) or UART activity

When a Shimmer with MicroSD card is inserted into a USB Reader Dock that is connected to a PC, the host system can mount the flash storage card in Shimmer as though it were a USB flash key. Depending on the specifics of your system, a few things may happen:

- A window may open to display the contents of the Shimmer's flash
- A prompt window may pop-up and ask you what you want to do
- Nothing may happen, but when you click on the drive letter or volume name associated with the USB port of the USB Dock, the contents of the Shimmer flash may be browsed.

There are a few requirements for this interaction to work:

1) The Shimmer must be powered!

2) Your Shimmer card must have a compatible format (FAT file system is the only currently supported method for Shimmer Applications). If this isn't the case, you may be prompted to reformat the card. Be careful-- irreversible data loss will occur! Otherwise a binary/raw disk utility (such as DD) must be used to copy or access data. For more information, please see the *Shimmer MicroSD Media Guide*

3) Some Shimmer applications may block this ability or limit mounting to specific times during operation due to legacy compatibility issues, or pending disk operations on Shimmer. This will result in a blinking blue LED followed by a time-out from the HOST, or a blue LED that turns off (media is detected but fails OS poll). You can try to re-insert the Shimmer, inspect the code, or contact the developer.

This version of the Shimmer dock has GPS capability. The GPS Enable button is used to switch between GPS and host computer UART functionality. In GPS mode, the indicator on the button will blink green while GPS location is being acquired. The indicator will turn solid after a lock is reached. This requires a clear view of the sky or placement on a windowsill or ledge.

The GPS module will transmit serial GPS NMEA sentences to the docked Shimmer's external connector UART lines. This data can be used for application synchronization or localization. Configuration commands use the MTK NMEA format. For more information on the commands, data format, and operational parameters of the GPS module see the Shimmer GPS/PRES User Guide available at www.shimmer-research.com in downloads.

Toggling GPS Enable so that the indicator is off will select UART functionality. The UART lines will now pass thru to the USB UART on the host computer. In this mode, operation matches the previous Shimmer USB dock accessories. A common UART indicator is used for both the host UART, and the BSL programming interface UART. Beyond the indicator difference, the UART mode behaves like legacy USBREADER or USB DUART docks.

Application developers should consider the enhanced feature set of the Shimmer Dock 2. Additional user training may be required. Some example applications may operate differently or unpredictably when the GPS mode is selected.

Troubleshooting

Occasionally the USB UART drivers will fail to load. This condition typically happens when the Shimmer Dock has been unplugged and is often indicated by a solid UART activity LED. The simple fix is to unplug the USB cable and retry. If this process fails to remedy the situation, please contact Shimmer Research support for assistance.

While the Shimmer Dock has a memory retention feature for GPS almanac data, after a period of inactivity, or geographic movement, initialization of GPS almanac data may take up to an hour. Without a clear sky-view, data output from the GPS module will be incomplete.   This condition is indicated by a persistent blinking GPS indicator or blank fields in output data. When confronted with scenario try a different location. In some indoor cases, insufficient sky-view, RF blocking building materials, or shadows from adjacent structures will limit GPS functionality.

1.4 Maintenance

1.4.1 Cleaning

The Shimmer case is not waterproof and should never be submerged or saturated with fluids during operation or cleaning.

Periodic wipe-down of the case with an antiseptic wipe or according to the standard operating procedure used on any piece of equipment in the place of operation. The external connector should be swabbed with a fine brush.

If you are using Shimmer in the presence of biohazards, treatment with a disposable wrap or cover is required according to best practices. Biohazard contamination will void warranty and devices returned to Realtime Technologies or any Shimmer Research address will be disposed of in accordance with applicable laws.

Replacement cases are available.

1.4.2 Inspection

The battery should be periodically inspected at least weekly. Due to the unpredictable usage patterns of a research device, premature aging or failure may occur. If the battery seems "puffy" to the extent that it is impacting the fit of the enclosure (>1mm) or the integrity of the battery pack has been compromised by a puncture or abrasion you should contact customer support immediately for service.

1.4.3 Battery Replacement

The Battery should only be replaced by qualified personnel.

The device uses a diode wired-OR to prevent device damage from reversed battery leads and allow operation from external power while charging.

The Texas Instruments BQ24080 Smart Li Charger is used for battery management. The BQ24080 implements a multi-phase charge profile including battery conditioning and overcharge protection currently. The default Rset value of 6.49k provides conservative 125mA charge limiting. The included 450mAh battery pack includes secondary failsafe protection against over/under voltage and over-discharge-- all user selected or installed batteries must provide secondary failsafe protection. The maximum discharge current on the supplied battery is 3.360A, a rate far exceeding expected conditions for sensing applications.

1.4.4 Disposal

Never expose Shimmer devices to excessive heat or an open flame

Shimmer devices should be disposed of like other rechargeable devices. They should never be thrown away in the trash without first removing the battery. Lithium-Ion Polymer batteries are classified as hazardous substances in most municipalities should be disposed of according to local law or practice.

1.5 Quick Start

1.5.1 Shimmer system concepts

After power-up or reset, compiled application code resident in the Shimmer's program memory is executed. This program is often called the *Shimmer Application*, *Shimmer firmware*, *firmware image* or *bootstrap image* to differentiate from other code involved in a data gathering system. In order to change the Shimmer application, a new firmware program must be loaded via the USB dock (often called "BSL-ing"). There are a few popular methods-- pick the approach that best matches your needs:

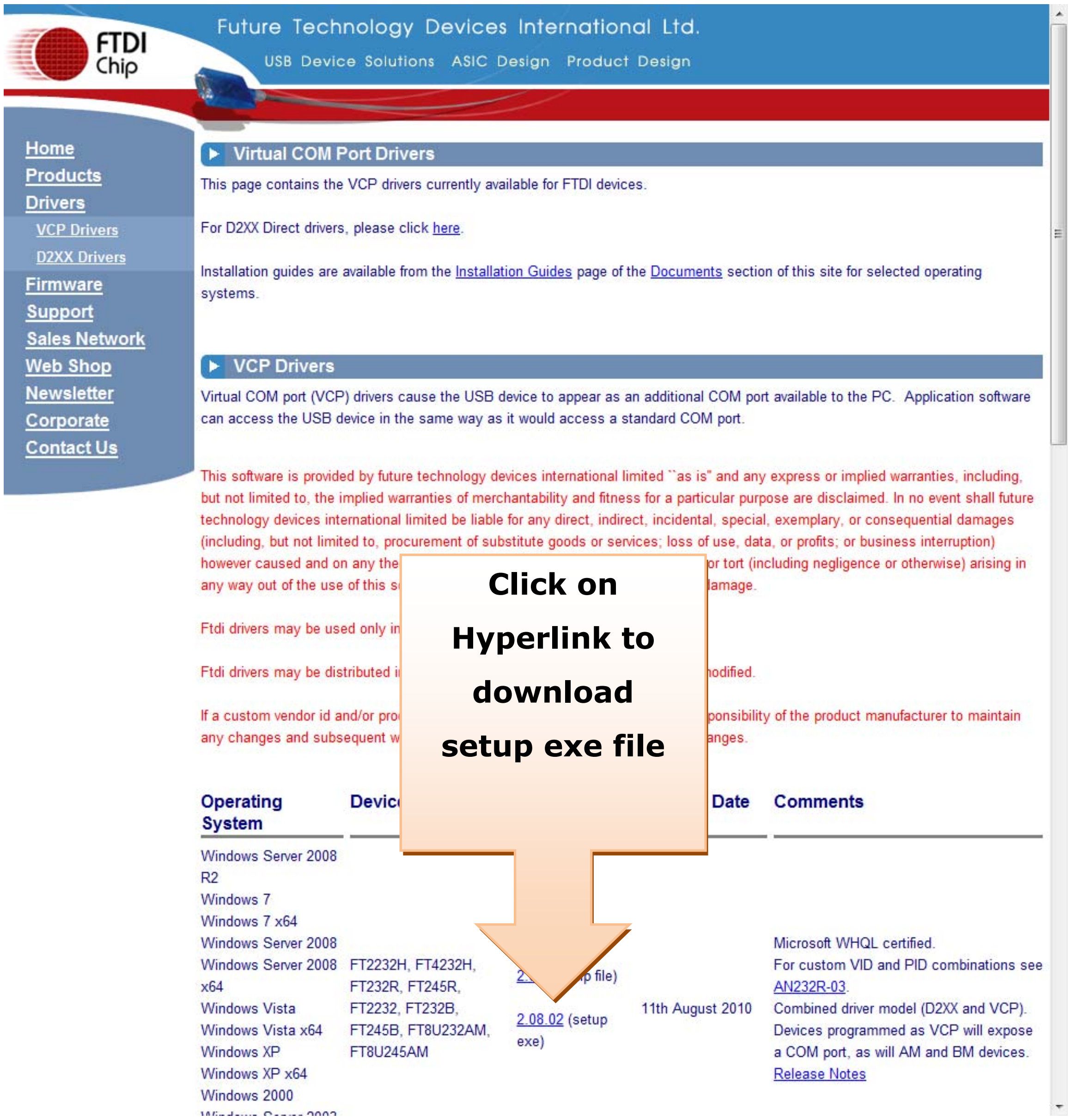
- On Windows, use the supplied BSL430.exe program to select and program the Shimmer with pre-compiled images using a graphical application.
- Use the free VMware Player to execute the Linux Development environment image on the Live Distribution. Your environment will allow you to compile and program the Shimmer with any application in the Shimmer software repository with a single command. The virtual machine runs within your regular PC session without restarting or stopping applications.
- Boot the supplied Live Distribution. Your environment will allow you to compile and program the Shimmer with any application in the Shimmer software repository with a single command. You can boot from most PCs without changing pre-existing software.
- Create a native development system. Your environment will allow you to compile and program the Shimmer with any application in the Shimmer software repository with a single command.

The remainder of this section will describe the first three approaches and walk a user thru programming a Shimmer. Configuring a native development environment is described later in this manual. In many cases, the Shimmer will interact with another computing device or mobile phone. The software running on those devices is usually referred to as the *host application* or just *application*. The Windows Quickstart will also describe a Bluetooth streaming data display application which is a simple way to get started with Shimmer. Many of the concepts presented are applicable to other host devices.

1.5.2 Quickstart for Windows

Do not allow windows to search or install the driver automatically

If this is the first time you've inserted the USB dock or USB reader into your system first execute the setup executable driver link on the right most column of the Future Technology Devices International Ltd. VCP drivers page. The URL is: **http://www.ftdichip.com/Drivers/VCP.htm**


FTDI Chip
Future Technology Devices International Ltd.
USB Device Solutions ASIC Design Product Design
Home
Products
Drivers
VCP Drivers
D2XX Drivers
Firmware
Support
Sales Network
Web Shop
Newsletter
Corporate
Contact Us
Click on Hyperlink to download setup exe file

Virtual COM Port Drivers

This page contains the VCP drivers currently available for FTDI devices.

For D2XX Direct drivers, please click here.

Installation guides are available from the Installation Guides page of the Documents section of this site for selected operating systems.

VCP Drivers

Virtual COM port (VCP) drivers cause the USB device to appear as an additional COM port available to the PC. Application software can access the USB device in the same way as it would access a standard COM port.

This software is provided by future technology devices international limited ``as is'' and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall future technology devices international limited be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any the or tort (including negligence or otherwise) arising in any way out of the use of this s amage.

Ftdi drivers may be used only in

Ftdi drivers may be distributed i nodified.

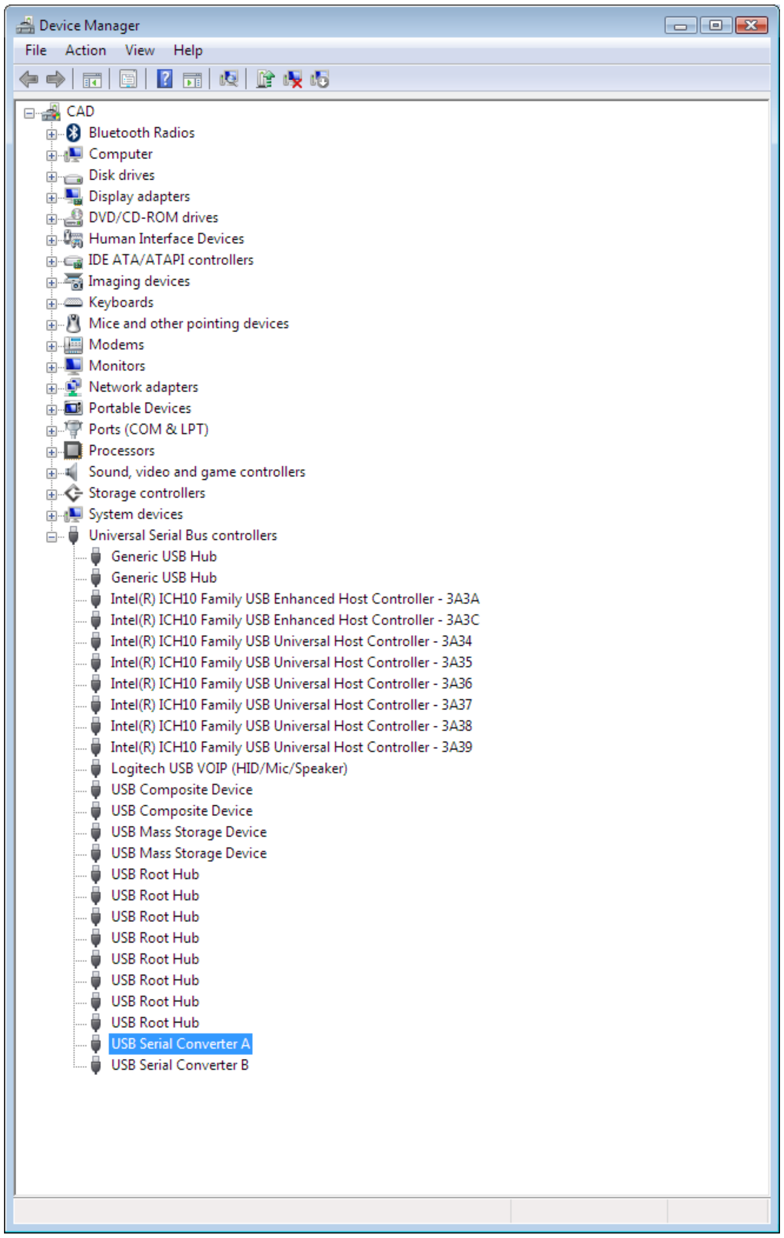
If a custom vendor id and/or pro ponsibility of the product manufacturer to maintain any changes and subsequent w anges.

Operating System	Devic		Date	Comments
Windows Server 2008 R2 Windows 7 Windows 7 x64 Windows Server 2008 Windows Server 2008 x64 Windows Vista Windows Vista x64 Windows XP Windows XP x64 Windows 2000	FT2232H, FT4232H, FT232R, FT245R, FT2232, FT232B, FT245B, FT8U232AM, FT8U245AM	2. p file) 2.08.02 (setup exe)	11th August 2010	Microsoft WHQL certified. For custom VID and PID combinations see AN232R-03. Combined driver model (D2XX and VCP). Devices programmed as VCP will expose a COM port, as will AM and BM devices. Release Notes

Next, plug in your Shimmer USB Dock or USB reader. If the reader hasn't be powered recently, you may get a driver error. Simply unplug and try again.

Next verify your system settings:

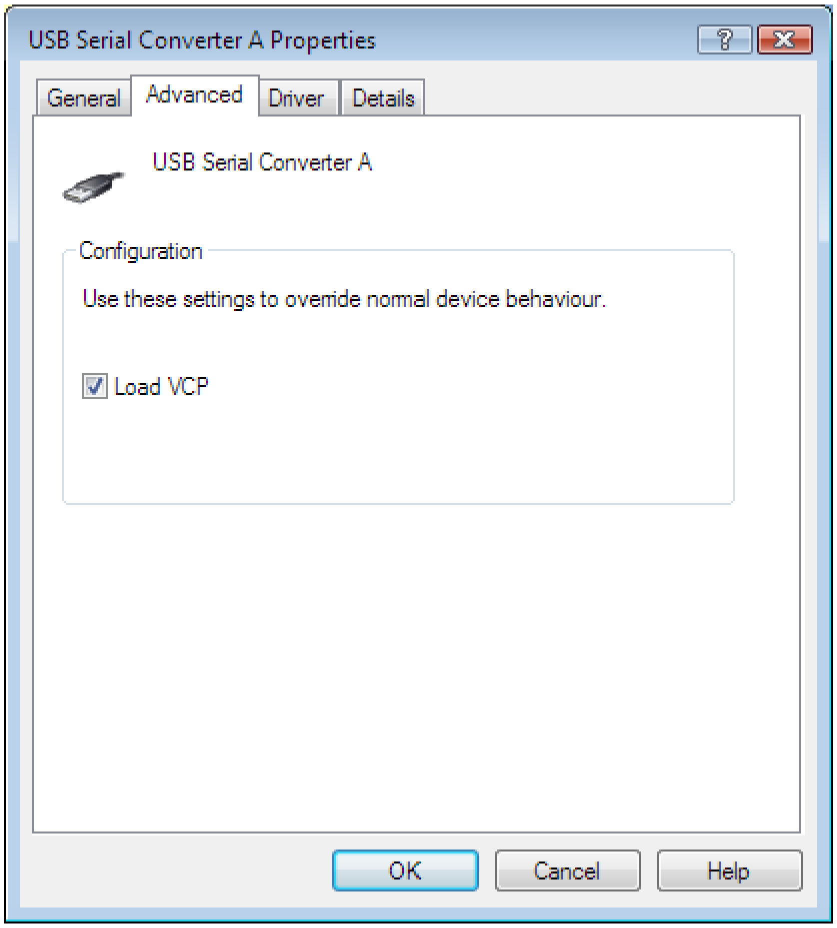
1) From the Control Panel, select Device Manager (control panel->system->hardware->device manger on Windows) and expand the USB or Universal Serial Bus entry.



```
CAD
  Bluetooth Radios
  Computer
  Disk drives
  Display adapters
  DVD/CD-ROM drives
  Human Interface Devices
  IDE ATA/ATAPI controllers
  Imaging devices
  Keyboards
  Mice and other pointing devices
  Modems
  Monitors
  Network adapters
  Portable Devices
  Ports (COM & LPT)
  Processors
  Sound, video and game controllers
  Storage controllers
  System devices
  Universal Serial Bus controllers
    Generic USB Hub
    Generic USB Hub
    Intel(R) ICH10 Family USB Enhanced Host Controller - 3A3A
    Intel(R) ICH10 Family USB Enhanced Host Controller - 3A3C
    Intel(R) ICH10 Family USB Universal Host Controller - 3A34
    Intel(R) ICH10 Family USB Universal Host Controller - 3A35
    Intel(R) ICH10 Family USB Universal Host Controller - 3A36
    Intel(R) ICH10 Family USB Universal Host Controller - 3A37
    Intel(R) ICH10 Family USB Universal Host Controller - 3A38
    Intel(R) ICH10 Family USB Universal Host Controller - 3A39
    Logitech USB VOIP (HID/Mic/Speaker)
    USB Composite Device
    USB Composite Device
    USB Mass Storage Device
    USB Mass Storage Device
    USB Root Hub
    USB Root Hub
    USB Root Hub
    USB Root Hub
    USB Root Hub
    USB Root Hub
    USB Root Hub
    USB Root Hub
    USB Serial Converter A
    USB Serial Converter B
```

2) You should see at least two entries for USB Serial Converter (the other entries will vary with system configuration and may not match the image above exactly).

3) Click on each one and under the General tab, confirm that the manufacturer is FTDI and under the advanced tab, make sure the "Load VCP" checkbox is checked. If the checkbox isn't checked, you will need to check for each Serial converter and the unplug and re-insert the dock.



USB Serial Converter A Properties

USB Serial Converter A

Configuration

Use these settings to override normal device behaviour.

[✓] Load VCP

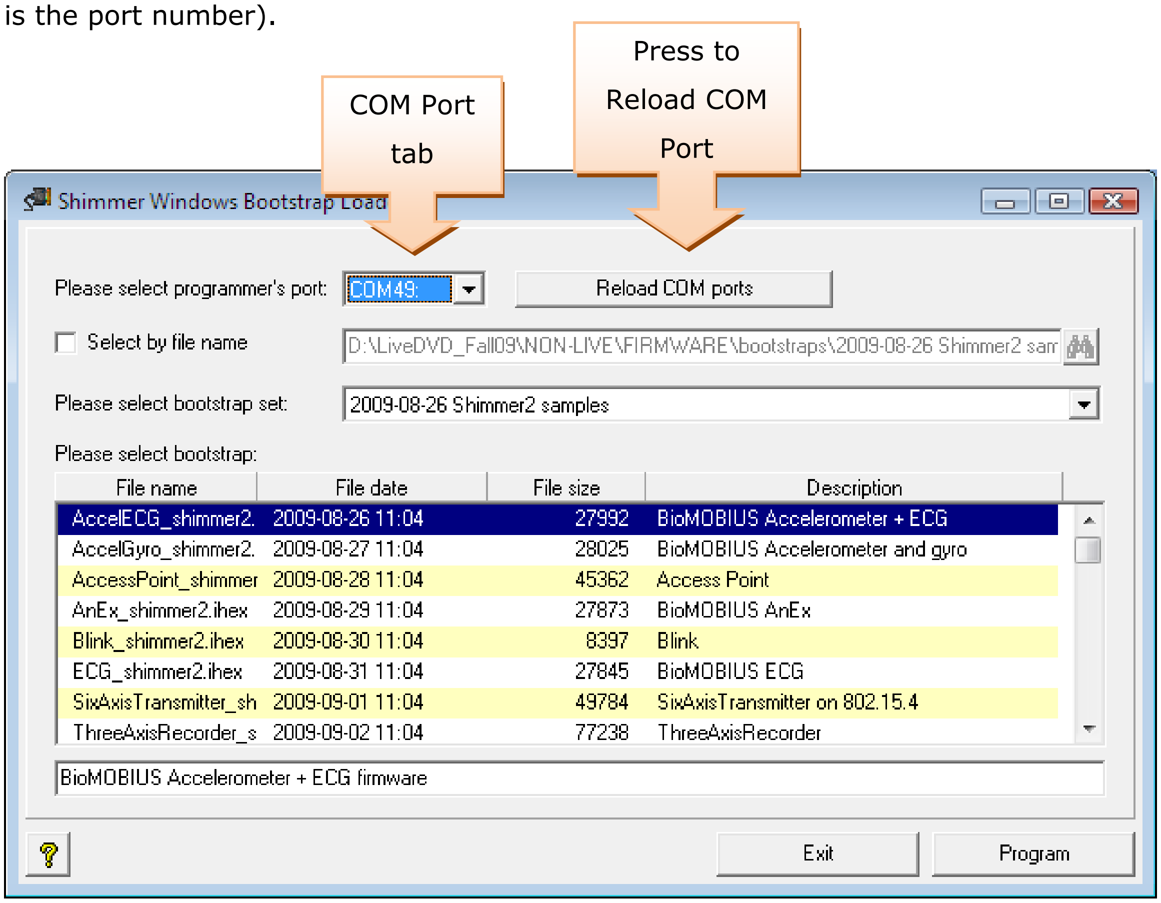
If you are experiencing problems with the USB Dock, try unplugging the USB cable and reinserting. Often, this process will resolve problems.

1.5.2.1 BSL430 Application

The BSL430.exe program is available via download from http://www.shimmer-research.com in the user area or in the /NON-LIVE/FIRMWARE directory of the Live Distribution.

You can copy the entire Firmware Directory to a more convenient place or create a shortcut to the BSL430.exe program on your desktop.

When you execute BSL430, the program will attempt to automatically detect the programmer at start-up as indicated by an entry in the port tab of "COMxx" where xx is the port number).



Please select programmer's port: COM49:

☐ Select by file name D:\LiveDVD_Fall09\NON-LIVE\FIRMWARE\bootstraps\2009-08-26 Shimmer2 sam

Please select bootstrap set: 2009-08-26 Shimmer2 samples

Please select bootstrap:

File name	File date	File size	Description
AccelECG_shimmer2.	2009-08-26 11:04	27992	BioMOBIUS Accelerometer + ECG
AccelGyro_shimmer2.	2009-08-27 11:04	28025	BioMOBIUS Accelerometer and gyro
AccessPoint_shimmer	2009-08-28 11:04	45362	Access Point
AnEx_shimmer2.ihex	2009-08-29 11:04	27873	BioMOBIUS AnEx
Blink_shimmer2.ihex	2009-08-30 11:04	8397	Blink
ECG_shimmer2.ihex	2009-08-31 11:04	27845	BioMOBIUS ECG
SixAxisTransmitter_sh	2009-09-01 11:04	49784	SixAxisTransmitter on 802.15.4
ThreeAxisRecorder_s	2009-09-02 11:04	77238	ThreeAxisRecorder

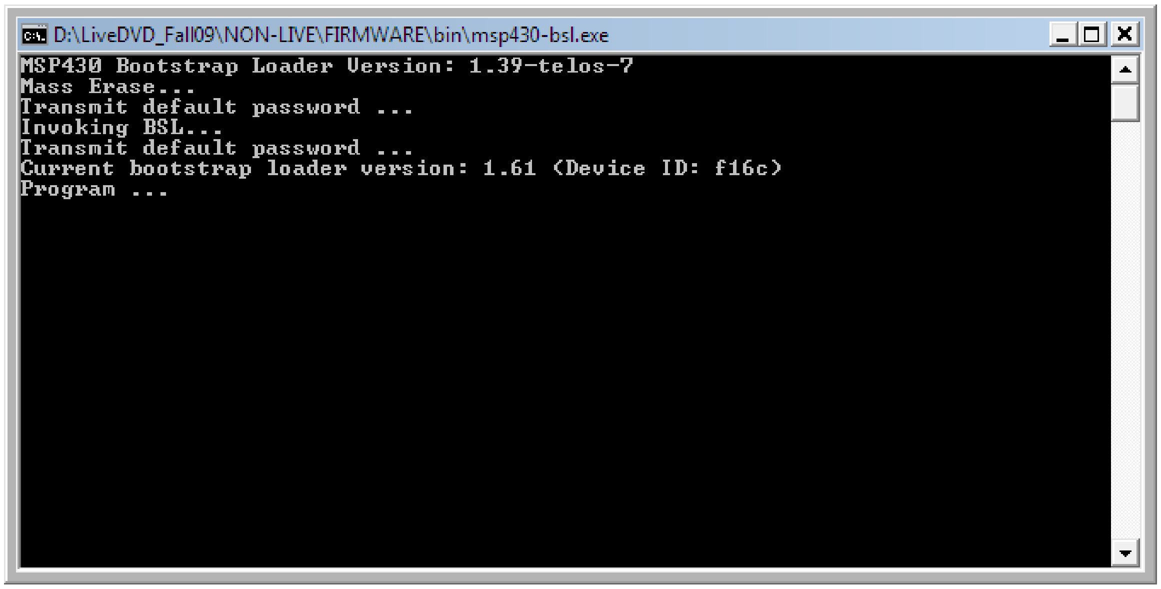
BioMOBIUS Accelerometer + ECG firmware

If no port is displayed, try pressing the "reload COM ports" button. If that fails, first check the cable, then revisit the USB configuration steps above.

You can use the scroll bar, and selection tabs to browse and select a precompiled bootstrap set-- usually a file with an .ihex extension.   When using the supplied bootstrap sets, the selection must match the target platform or unpredictable behaviour will occur.

Select the "Blink_shimmer2r.ihex" file (from the Shimmer2R samples bootstrap set) and press the Program Button.

A Windows should pop up that looks like this:



```
MSP430 Bootstrap Loader Version: 1.39-telos-7
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Program ...
```

In addition, the program light near on the USB Dock will flash. Depending on the program size, it may take up to five minutes to program a Shimmer, but blink is quite small and the firmware update should complete quickly.

The LEDs on your Shimmer should be blinking-- congratulations you have programmed your Shimmer!

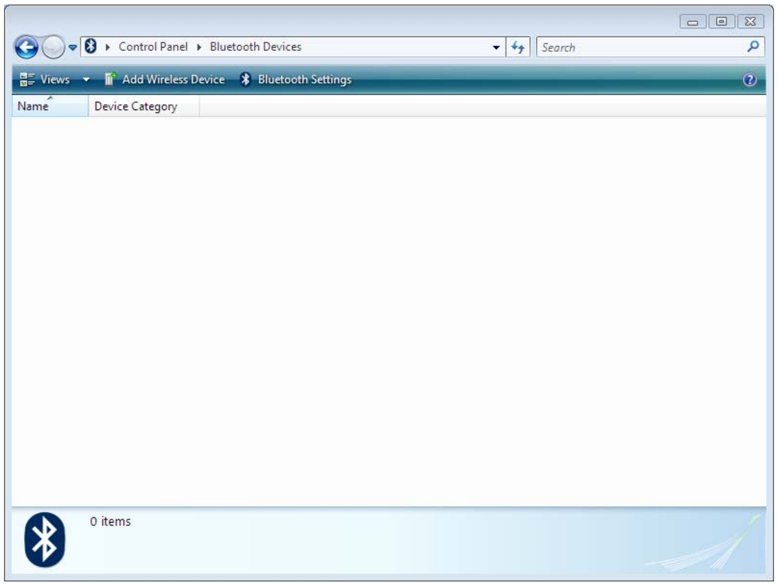
If you experience difficulty, try again and then verify that your Shimmer is powered on, and the correct COM port is selected.

1.5.2.2 Shimmer Bluetooth Pairing for Windows

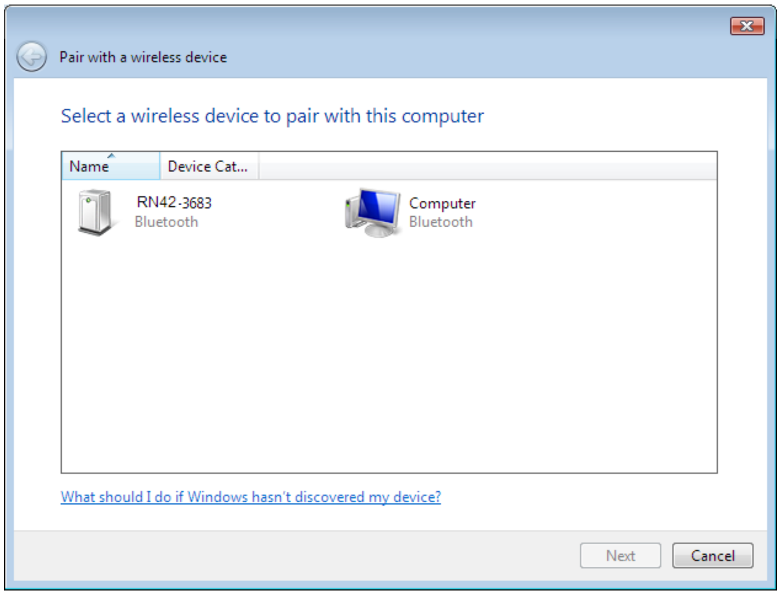
ShimmerConnect, the focus of the next section is an application that lets you quickly visualize, forward, or capture data from Shimmer's sensor using a Bluetooth connection. Before we get to that we must setup our Bluetooth connection, this is called "pairing". The Windows process is very involved, but applicable to mobile phones and other systems (usually with far fewer steps involved!). You must have a plug-in or built in Bluetooth radio installed and activated to continue. If you are unsure if you have Bluetooth-- check your control panel for the Bluetooth Devices icon-- if it isn't present, or all of the panels within or blank, you need to contact the appropriate party for support.

Recalling the previous section, program your Shimmer with the BoilerPlate image. This time, programming will take a minute or two.

Next open the Bluetooth Devices icon on your status bar, or via the control panel:

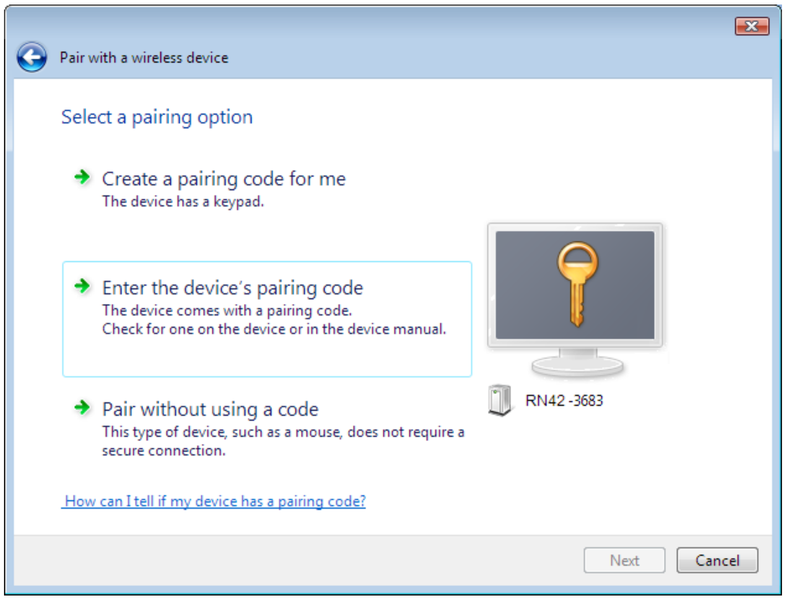


To add your Shimmer, click on the "add wireless device" button. After a few minutes, a list of devices will be presented:

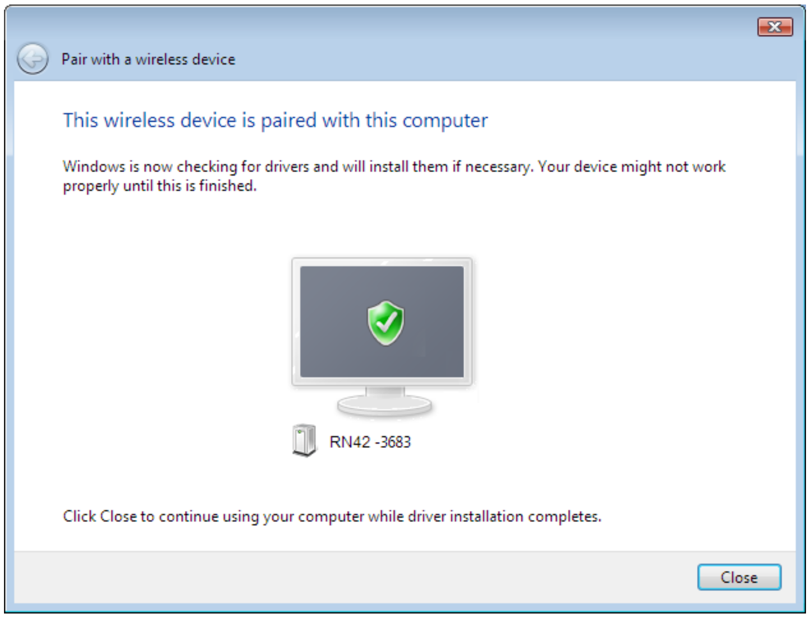

Pair with a wireless device
Select a wireless device to pair with this computer
Name
Device Cat...
RN42-3683
Bluetooth
Computer
Bluetooth
What should I do if Windows hasn't discovered my device?
Next
Cancel

Shimmer2R default device name is "RN42-xxxx", where xxxx is a unique number. In the case above, RN42-3683 is the Shimmer. Developers can update this name and as well as many additional parameters using the Bluetooth driver in the TinyOS repository. Click on the Shimmer device to continue.

Next you will be presented with a Pairing menu:


Pair with a wireless device
Select a pairing option
Create a pairing code for me
The device has a keypad.
Enter the device's pairing code
The device comes with a pairing code.
Check for one on the device or in the device manual.
Pair without using a code
This type of device, such as a mouse, does not require a secure connection.
RN42-3683
How can I tell if my device has a pairing code?
Next
Cancel

Select the middle option, "enter the device's pairing code" (you must do so within a few minutes or pairing will fail). The default code is **1234**. Press Next.



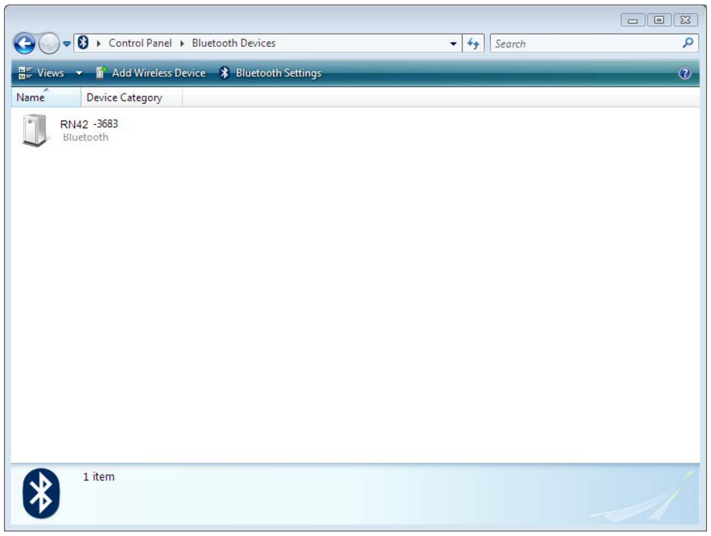
This wireless device is paired with this computer

Windows is now checking for drivers and will install them if necessary. Your device might not work properly until this is finished.

RN42 -3683

Click Close to continue using your computer while driver installation completes.

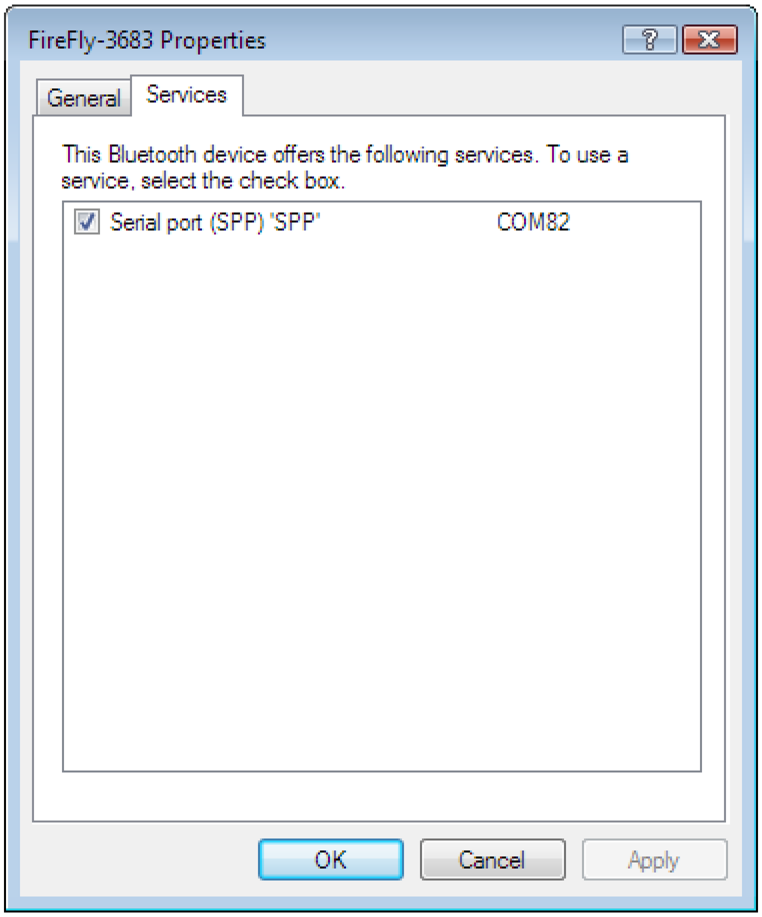
After successful pairing, your device will be listed in the Bluetooth devices list.



RN42 -3683
Bluetooth

1 item

Now double-click on the device to open its properties. Under the Services tab you will see a specific Serial Port assignment which is used by all Bluetooth applications:


FireFly-3683 Properties

This Bluetooth device offers the following services. To use a service, select the check box.

☑ Serial port (SPP) 'SPP' COM82

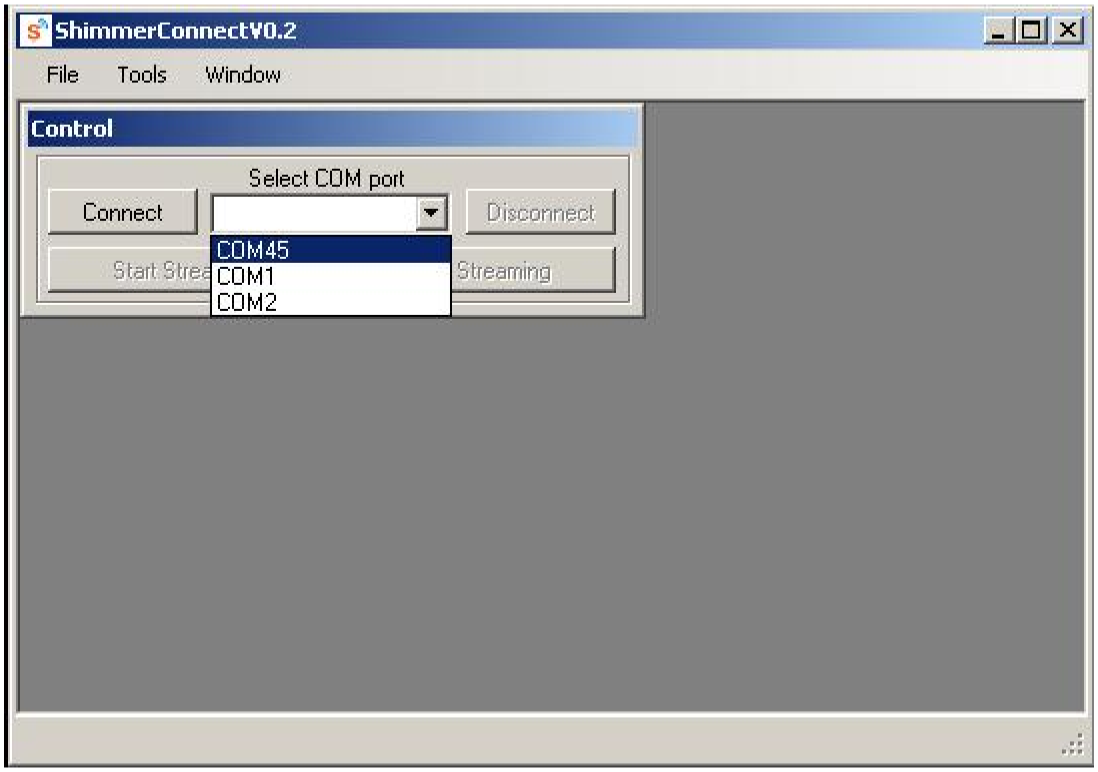
If you have multiple Shimmers, each will have its own serial port number. That number varies from system to system and represents the Serial "cable" that the Bluetooth Serial Port Profile is replacing with a wireless connection. Normally, you will not have to re-pair your Shimmer again on the system. This may be a good time to physically label your Shimmers with the 4-digit firefly suffix as well-- with multiple devices, it is easy to get confused!

1.5.2.3 ShimmerConnect Application

The ShimmerConnect application can run on either Windows or Linux. This section will describe getting started with ShimmerConnect in Windows. See the ShimmerConnect user manual for more information on running it in Linux.

You can download the ShimmerConnect application from the shimmer-research website:

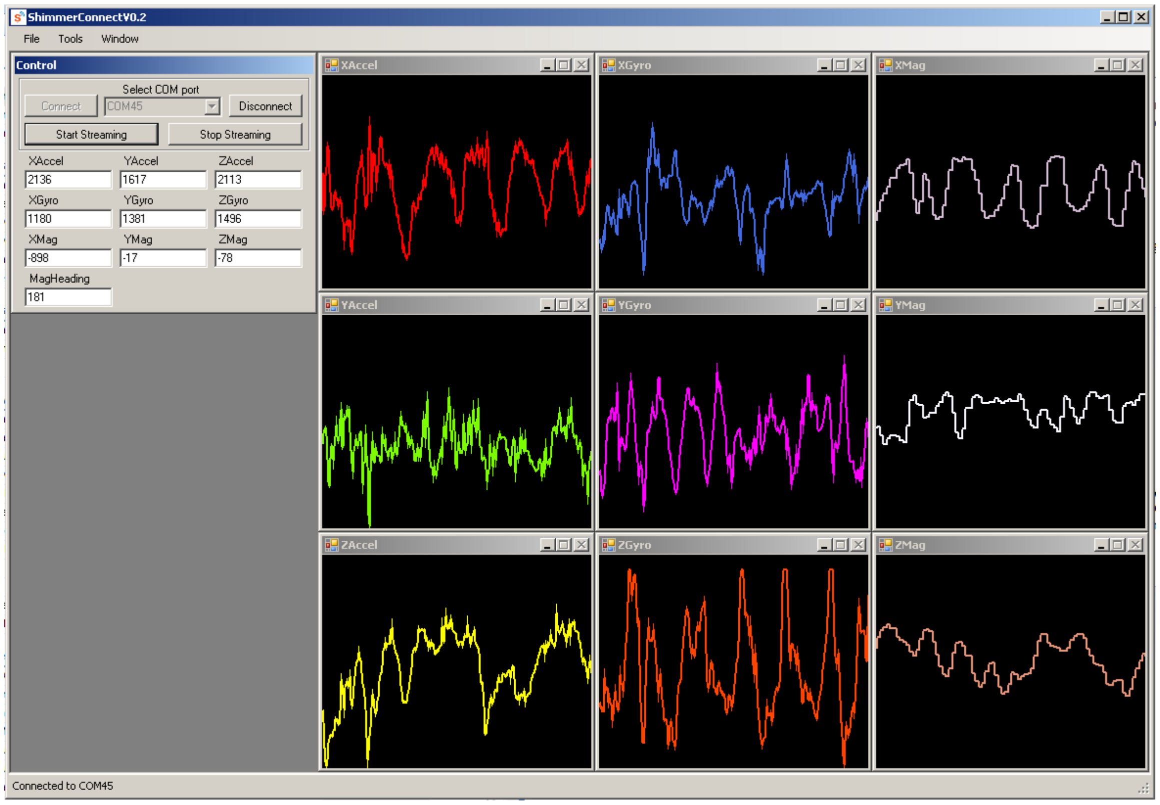
Save the file in a convenient place and execute. The first step is to configure the Bluetooth transmission channel.



From the “Select COM Port” drop down menu select the correct COM port number for your Shimmer.

In our example, we've selected COM45:

To get started, press the Connect button. When you press Start, your Shimmer will start streaming data to the host application. The data is displayed on the screen and can be stored to a CSV file. When you are done, stop transmission. If you press the Disconnect button, it will release the connection to the host application and remain idle. Here is an example of an Accel, Gyro and Magnetometer session:



For more information on the functionality of ShimmerConnect please see the ShimmerConnect user manual.

1.5.3 Virtual Machine Developer Quick Start

The VMware pre-made virtual machine is located on the Live Distribution USB Drive under /NON-LIVE/VMware - you can use it from either the **FREE** VMware Player or if you already have a license, you can also use VMware Workstation (where more of the parameters of the virtual machines will be editable). In order to use the pre-made virtual machine you need a computer with 2-4 GB RAM and at least 8-10 GB disk space.

The Live Distribution includes a compressed image of a fully configured Linux development environment. To use it you must first uncompress the image with 7-Zip, we supply a windows installer for this application in the /NON-LIVE/VMware directory. Once 7-Zip installed, browse to the ShimmerLive_xxx.7z image and extract it by right clicking its icon and selecting 7zip->Extract to and completing the dialog box with the location of your choice.

The resulting VMware .vmx file is then executed by the VMware Player.

IMPORTANT: In order to access the programming dock as described in the next section, first expand the Virtual Machine menu on the VMware titlebar and open Removable Devices. Select a peripheral to connect/disconnect from host. When a peripheral is connected to VMware it is no longer visible to the host OS and vice versa. The future devices shimmer usb reader is required for programming and serial communication. If you wish to access the contents of the Shimmer MicroSD from VMware you should connect the Standard Shimmer Reader.

At this point you can open a terminal window and continue with the Live Distribution Quickstart Tinyos-1.x or Tinyos-2.x directions below.

1.5.4 Live Distribution Overview and Developer Quick start

Any modern x86 computer with at least 512 MB RAM that can read and boot from a USB Flash Drive should be able to use the Live Distribution. Some computers will require you to enter the 'boot menu' to change the device priority so that a USB device- sometimes "removable media" - appears before the hard drive.

Initially a boot prompt will be displayed, and if <Enter> is pressed or no action is taken for about 15-30 seconds the boot will continue with the default configuration - the full boot sequence might take up to 1-5 minutes.

The username for logon is **tiny1** and the password is also **tiny1** - all the TinyOS build tools are already installed/preconfigured and in the home directory for that user the full CVS image of the TinyOS 1.x project is found in the **tinyos-1.x** folder - you can check it by opening a terminal window and entering:

```
cd ~/tinyos-1.x/ contrib/handhelds/apps/Blink

make shimmer2r
```

You will see a compilation message on the terminal. If you have a Shimmer (powered-on) in a programming dock connected to your computer via a mini-USB cable and no other 'serial-over-USB' port plugged-in you can also try:

```
make shimmer2r install bsl,/dev/ttyUSB0
```

The simplest way to confirm that you are using the correct device is to remove the dock and type at a prompt:

ls -l /dev/ttyUSB*

You might see nothing; that's OK. Now, plug the dock back in and the dock's two ports will also appear. When programming, choose the first of the two.

Congratulations, you just programmed your Shimmer!

Now you can start using TinyOS - but please remember that **ALL THAT YOU DO IN A SESSION WILL BE LOST WHEN YOU RESTART** (unless you use the 'persistent' mode or you copy your work on some disk - either a local disk - including Windows NTFS disks - or a network share).

TinyOS-2.x is also available; a second login of tiny2/tiny2 accesses this environment. Switch users by logging out and logging back in as tiny2. Enter tiny2 as the password.

As before, you can check the build entering:

```
cd /home/tiny2/tinyos-2.x/apps/Blink
```

```
make shimmer2r
```

and if you have a Shimmer in the dock and no other 'serial-over-USB' port plugged-in you can also try:

```
make shimmer2r install bsl,/dev/ttyUSB0
```

(the Blink application itself might be a little different in TinyOS v2 than in TinyOS v1).

2 Hardware

2.1 Introduction

This section provides an overview of the Shimmer hardware architecture and discusses the hardware sub-systems contained within. Consideration will also be given to the internal and external expansions available.

2.2 Shimmer Hardware Overview

Figure 2.1 illustrates a block diagram of the Shimmer baseboard interconnections and integrated devices.

The core element of the platform is the low-power MSP430F1611 microprocessor which controls the operation of the device. Nearly every feature of the CPU is exercised in the Shimmer implementation! The CPU configures and controls various integrated peripherals through I/O pins, some of which are available on the internal/external-expansion connectors. The CPU has an integrated 8-channel 12bit analog-to-digital converter (ADC) which is used to capture sensor data from the accelerometer, battery, or sensor expansions such as ECG, kinematics, GSR, and EMG. The external expansion allows communication to and from the baseboard using the docking station. The Shimmer board has a built in MicroSD Flash socket for additional storage and has three light-emitting diodes (LED) for display purposes. For wireless data streaming the platform is equipped with both Bluetooth and 802.15.4 radio modules.

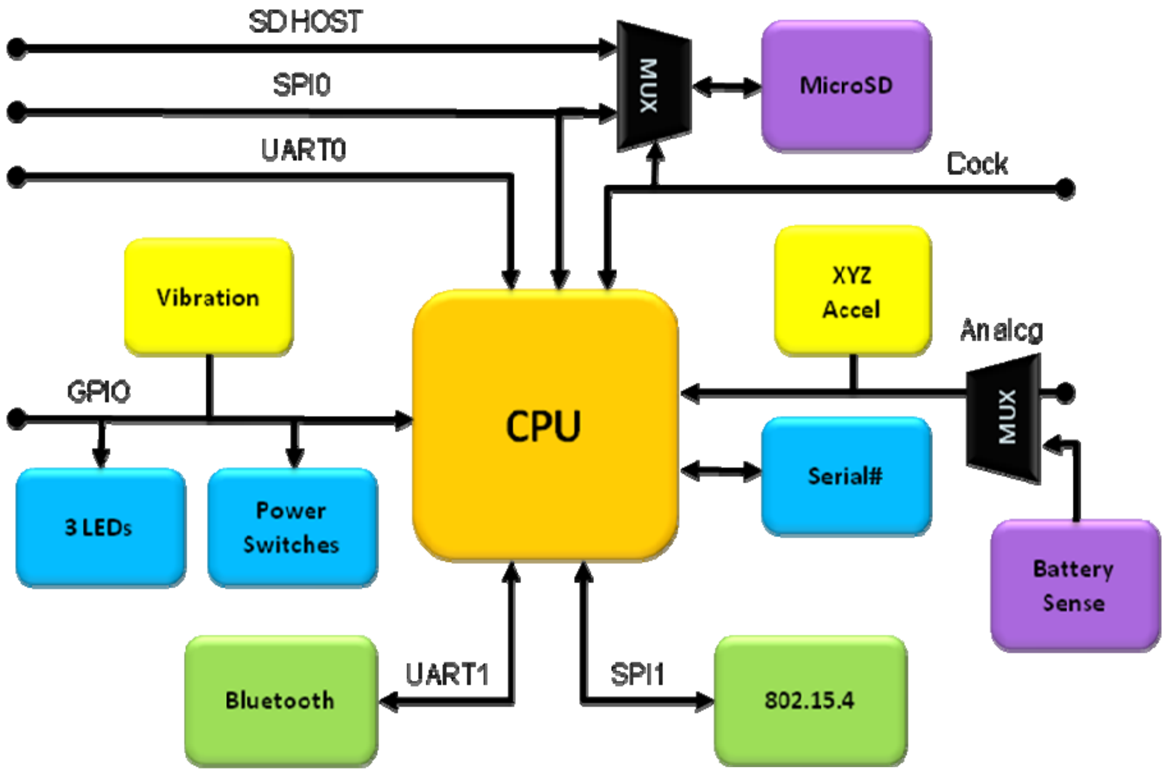
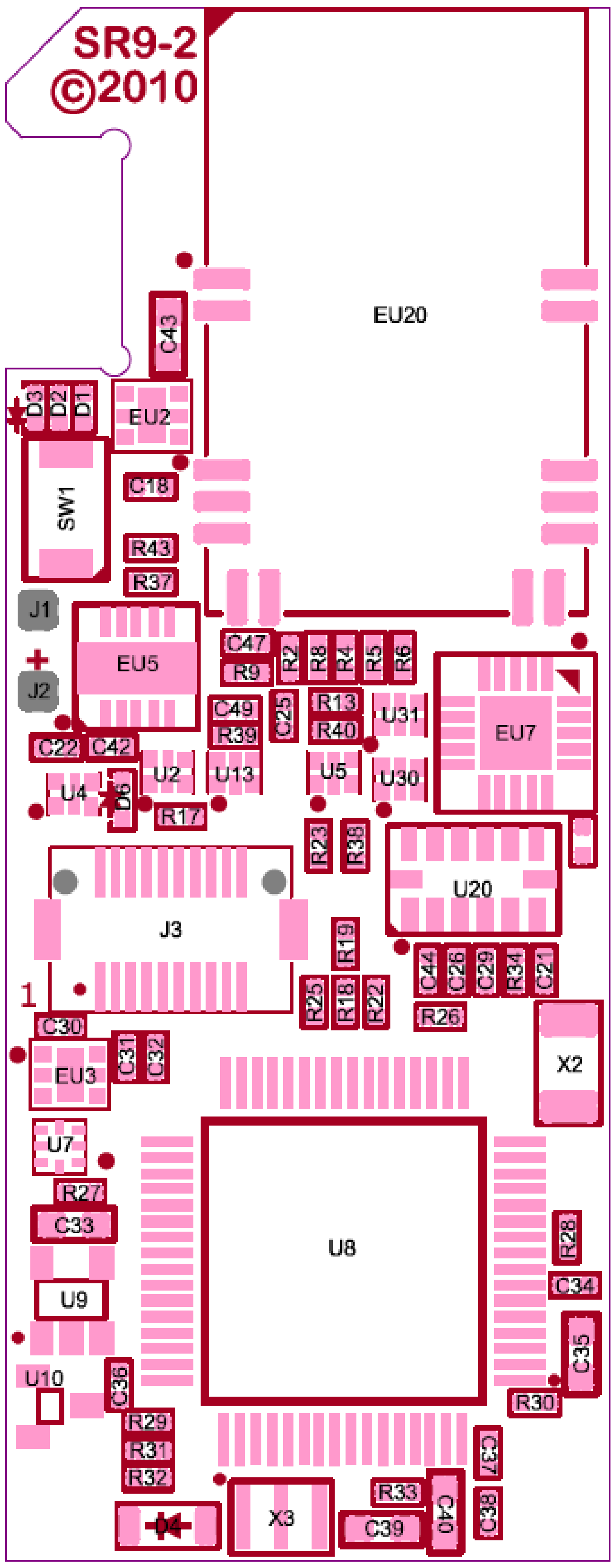


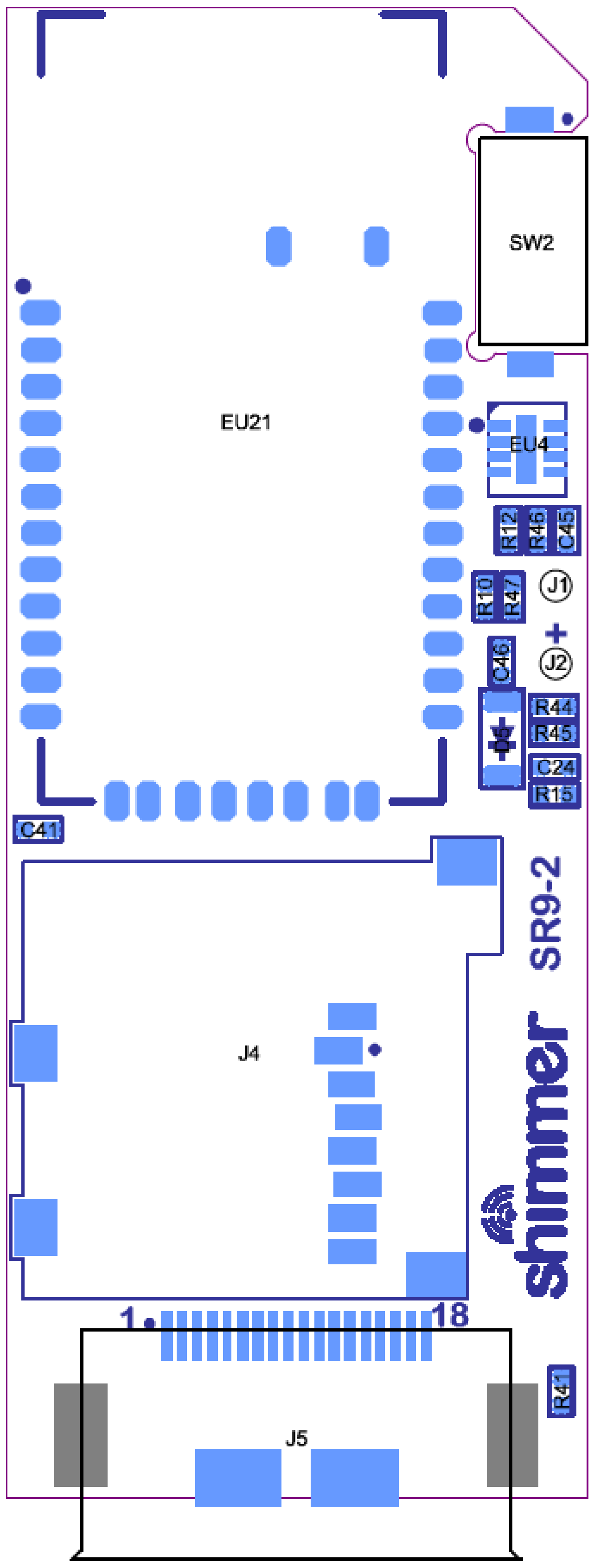
Figure 2.1, Shimmer2R System Diagram

2.3 Board detail for debug and testing

2.3.1 Important Components

These following components may be removed or replaced to support specific user applications, configure the board, or perform power-measurement testing. For more information contact Shimmer Research.


SR9-2
©2010
EU20
C43
D3 D2 D1
EU2
C18
SW1
R43
R37
J1
J2
EU5
C47
R9
R2 R8 R4 R5 R6
EU7
C49
R39
C25
R13
R40
U31
C22 C42
U2
U13
U5
U30
U4
D6
R17
R23
R38
U20
J3
R19
C44 C26 C29 R34 C21
1
R25 R18 R22
R26
C30
X2
EU3
C31 C32
U7
R27
C33
U8
R28
C34
U9
C35
U10
C36
R30
R29
R31
R32
C37
R33
C40
C38
X3
C39
D4


EU21
SW2
EU4
R12
R46
C45
R10
R47
J1
+
J2
C46
D5
R44
R45
C24
R15
C41
SR9-2
shimmer
J4
1
18
J5
R41

J1	Negative battery terminal
J2	Positive battery terminal
U9	Primary Regulator (3.0V LDO)
EU5	Battery charger (BQ24080)
R37	Battery charger RSET
EU4	Power-button and Reset controller (LTC2954-2)
C45	Power-off time capacitor
X2	32.768k crystal
X3	XT2 clock source, 8MHz resonator
R30	CPU power jumper (use for power measurements/pi filter)
U8	CPU (MSP430F1611) [12]
EU2	Bluetooth soft power switch (FPF1005)
EU21	RN-42 Bluetooth Radio Module
EU20	SR7 802.15.4 Radio Module [15]
U10	Unique Silicon Serial ID# (DS2411)
R34	Accelerometer power jumper (use for power measurements/pi filter)
EU6	3-Axis Accelerometer (MMA7260Q) [13]
C44	Accelerometer x-filter
C26	Accelerometer y-filter
C29	Accelerometer z-filter
SW1	Reset/Power button
D1	Green LED
D2	Yellow LED
D3	Red LED
J3	Internal Expansion (DF128-20DS-0.5V) [18]

J5	External Expansion (ST80-18P)
J4	MicroSD socket
EU3	MicroSD socket soft power switch (FPF1005)
EU7	MicroSD to-host-bypass Mux (ADG774)
U2,4,5,13	SPI0 Interface Buffers (NC7SP126)
U30,31	Power measurement Muxes (FSA4157A)
D5	Battery Isolation Diode (SBR130S3)
SW2	Tilt/Vibration sensor (SQ-SEN-200)
R46	Regulator voltage divider resistor (top)
R47	Regulator voltage divider resistor (bottom)
R44	Battery voltage divider resistor (top)
R45	Battery voltage divider resistor (bottom)

2.3.2 Connector Pin Assignments

External Expansion Connector:

Conn Pin	Net Name	hardware.h Pin Label	Interruptible?	Function
1	PV_CHG	N/A	N	Battery Charging Power 5-6.5VDC. Do not exceed 300ma connector current rating
2	SER0_RXD	URXD0	N	USART0: Serial
3	SER0_TXD	UTXD0	N	USART0: Serial
4	SPI0_SCLK_EXT	UCLK0	N	USART0: SPI
5	MUX_SPI0_SOMI	SOMI0	N	USART0: SPI
6	MUX_SPI0_SIMO	SIMO0	N	USART0: SPI
7	JTAG_MSP_TCK	N/A	N	BSL programming
8	MSP_RESET_N	N/A	N	BSL programming and Global Reset

9	PV_REG	N/A	N	Unregulated board power (2.7-5V)
10	GPIO_EXTERNAL	GIO0	Y	GPIO, Pulled low on Shimmer with 6.49k Resistor. Recommended use is User Button or SW attention signal
11	BSL_RX	PROG_IN	Y	BSL programming
12	MUX_VSENSE_ADC0	ADC_0	N	ADC input or GPO
13	MUX_VSENSE_ADC7	ADC_7	N	ADC input, GPO or DAC output
14	BSL_TX	PROG_OUT	Y	BSL Programming, Recommend use is expansion power control Signal
15	SER0_RTS	SER0_RTS	Y	USART0: Serial, GPIO
16	SER0_CTS	SER0_CTS	Y	USART0: Serial, 1-Wire
17	PV	N/A	N	Regulated Board power output: 3.0VDC @ 50ma*
18	GND	N/A	N	Board Ground

* Exact current capacity will vary based on operating state of integrated peripherals and processor.

Internal Expansion Connector:

(For function descriptions see previous table)

Net name	hardware.h Pin Label	Conn Pin Number	Conn Pin Number	hardware.h Pin Label	Net Name
PV	N/A	1	20	N/A	PV_REG
GND	N/A	2	19	N/A	GND
SER0_RTS	SER0_RTS	3	18	SIMO0	SPI0_SIM0
VSENSE_ADC6	ADC_6	4	17	SOMI0	SPI0_SOMI
VSENSE_ADC2	ADC_2	5	16	N/A	MSP_RESET_N
VSENSE_ADC1	ADC_1	6	15	N/A	JTAG_MSP_TCK
BSL_TX	PROG_OUT	7	14	UCLK0	SPI0_SCLK_INT
SER0_CTS	SER0_CTS	8	13	UTXD0	SER0_TXD
GPIO_Internal	GIO1	9	12	URXD0	SER0_RXD
BSL_RX	PROG_IN	10	11	GIO2	GPIO_INTERNAL1 (100k pull-up resistor)

Notes:

- BSL_TX/RX can be used as GPIO
- BSL_TX/RX and Serial CTS/RTS can be used as ext. interrupt lines
- I2C is multiplexed onto SPI bus 0 lines.
- BT_PROG_CS is a reserved pin for manufacturing use only.
- VSENSE lines can be used for GPO

2.3.3 Recommended pin usage

To ensure compatibility with existing SW and in cases where both internal and external expansion devices are used, the following conventions are recommended. Where applicable a description is provided for expansions that use the pin:

External Expansion connector:

Pin 10 (GPIO_EXTERNAL) is pulled low on Shimmer.

- Pulse high (button) as USER Attention signal to Shimmer

Pin 14 (BSL_TX) reserved, power-enable/control for internal expansion

- MUX Address Select Bit0 on GSR (LSB)

Pin 15 (SER0_RTS) power-enable or control for external expansion

- AnEx Expansion – High enables +/-5VDC regulator (recommended SW default: Low)
- Used as SD card data when docked

Pin 16 (SER0_CTS)

- MUX Address Select Bit1 on GSR (MSB)
- Used as Bluetooth CS when docked

Pin 12 (MUX_VSENSE_ADC0)

- Used for SD card data when docked

Pin 13 (MUX_VSENSE_ADC7)

- Used for SD card data when docked

Internal Expansion connector:

Pin 3 (SER0_RTS) reserved, power-enable/control for external expansion

Pin 8 (SER0_CTS)

- Used as Bluetooth CS when docked

Pin 15 (SER0_RTS) power-enable or control for external expansion

- AnEx Expansion – High enables +/-5VDC regulator (recommended SW default: Low)
- MUX Address Select Bit1 on GSR (MSB)
- Used as SDHOST CS when docked

Pin 7 (BSL_TX) power-enable/control for internal expansion

- Kinematics Expansion – Low enables Gyroscopes (recommended SW default: High)
- Mux Address Select Bit0 on GSR (LSB)

2.3.4 CPU Pin Assignment

The CPU pin connections for Shimmer developers are listed below:

Pin Name	**Pin #**	**Board Name**	**Hardware.h pin name**	**Use**	**Type**
DVCC	1	PV_MSP	N/A	Power	Analog
AVCC	64	PV_MSP	N/A	Power	Analog
VREF+	7	PV_VREF_MSP	N/A	A/D Ref	Analog
VeREF+	10	GND	N/A	A/D Ref	Analog
DVSS	63	GND	N/A	Power	Analog
AVSS	62	GND	N/A	Power	Analog
VREF-	11	GND	N/A	A/D Ref	Analog
TCK	57	JTAG_TCK	N/A	BSL,JTAG	Input
TMS	56	TP_JTAG_TMS	N/A	JTAG	
TDI	55	TP_JTAG_TDI	N/A	JTAG	
TDO	54	TP_JTAG_TDO	N/A	JTAG	
RST_N	58	MSP_RESET_N	N/A	BSL, Button	Open Drain
XIN	8	CLK_MSP_XIN	N/A	Primary XTAL	Clocking
XOUT	9	CLK_MSP_XOUT	N/A	Primary XTAL	Clocking
XT2IN	53	CLK_MSP_HF_XIN	N/A	8MHz Resonator	Clocking
XT2OUT	52	CLK_MSP_HF_XOUT	N/A	8MHz Resonator	Clocking
P1.0	12	RADIO_FIFP	RADIO_FIFOP	802.15.4	Input (IRQ)
P1.1	13	BSL_TX	PROG_OUT	BSL/GPIO	Output
P1.2	14	RADIO_SFD	RADIO_SFD	802.15.4	Input (IRQ)
P1.3	15	SER0_RTS	SER0_RTS	USART0: Serial	Output
P1.4	16	SER0_CTS	SER0_CTS	USART0: Serial	Input (IRQ)
P1.5	17	RADIO_FIFO	RADIO_FIFO	802.15.4	Input (IRQ)
P1.6	18	BT_RTS	BT_RTS	Bluetooth	Input (IRQ)
P1.7	19	BT_CTS	BT_CTS	Bluetooth	Output

P2.0	20	GPIO_EXTERNAL	GIO0	User button, Ext. GPIO	Unassigned IRQ)
P2.1	21	GPIO_INTERNAL	GIO1	Int. GPIO or CS (SPI0)	Unassigned IRQ)
P2.2	22	BSL_RX	PROG_IN	BSL	Input (IRQ)
P2.3	23	DOCK_N	DOCK_N	CPU/Flash	Input (IRQ)
P2.4	24	TILT	TILT	TILT	Input (IRQ)
P2.5	25	GPIO_INTERNAL1	GIO2	Int. GPIO, ROSC	Input (IRQ)
P2.6	26	BT_PIO	BT_PIO	Bluetooth	Input (IRQ)
P2.7	27	RADIO_CCA	RADIO_CCA	802.15.4	Input (IRQ)
P3.0	28	SPI0_CS_FLASH_N	SD_CS_N	FLASH (SPIO0 CS)	Output
P3.1	29	SPI0_SIMO0	SIMO0	USART0: SPIO0/I2C	Output
P3.2	30	SPI0_SOMI	SOMI0	USART0: SPIO0	Input
P3.3	31	SPI0_SCLK	UCLK0	USART0: SPIO0/I2C	Output
P3.4	32	SER0_TXD	UTXD0	USART0: SERIAL	Output
P3.5	33	SER0_RXD	URXD0	USART0: SERIAL	Input
P3.6	34	BT_TXD	UTXD1	USART1: Bluetooth	Output
P3.7	35	BT_RXD	URXD1	USART1: Bluetooth	Input
P4.0	36	LED_RD_N	RED_LED	LED	Output
P4.1	37	PMUX_SEL	SEL_PMUX	ADC	Output
P4.2	38	LED_YE_N	YELLOW_LED	LED	Output
P4.3	39	LED_GR_N	GREEN_LED	LED	Output
P4.4	40	ACCEL_SEL0_N	ACCEL_SEL0	Accelerometer	Output
P4.5	41	SW_FLASH_N	SW_SD_PWR_N	Flash	Output
P4.6	42	SW_BT_N	SW_BT_PWR_N	Bluetooth	Output
P4.7	43	1WIRE_DATA	ONEWIRE	802.15.4	Bi-directional
P5.0	44	ACCEL_SLEEP_N	ACCEL_SLEEP_N	Accelerometer	Output
P5.1	45	SPI1_SIMO	SIMO1	USART1: SPI1	Output
P5.2	46	SPI1_SOMI	SOMI1	USART1: SPI1	Input
P5.3	47	SPI1_SCLK	UCLK1	USART1: SPI1	Output
P5.4	48	SPI1_CS_RADIO_N	RADIO_CSN	802.15.4 CS (SPI1)	Output
P5.5	49	BT_RESET	BT_RESET	Bluetooth	Output
P5.6	50	RADIO_VREG_EN	RADIO_VREF	802.15.4	Output
P5.7	51	RESET_RADIO_N	RADIO_RESET	802.15.4	Output
P6.0	59	VSENSE_ADC0	ADC_0	ADC	Analog Input
P6.1	60	VSENSE_ADC1	ADC_1	ADC	Analog Input
P6.2	61	VSENSE_ADC2	ADC_2	ADC	Analog Input
P6.3	2	VSENSE_ACCEL_Z	ADC_ACCELZ	Accelerometer	Analog Input
P6.4	3	VSENSE_ACCEL_Y	ADC_ACCELY	Accelerometer	Analog Input
P6.5	4	VSENSE_ACCEL_X	ADC_ACCELX	Accelerometer	Analog Input
P6.6	5	VSENSE_ADC6	ADC_6	Int. GPIO, ADC or DAC	Unassigned
P6.7	6	VSENSE_ADC7	ADC_7	Ext. GPIO, ADC or DAC	Unassigned

2.4 Hardware Sub-System Detail

Combined with the board information in Section 2.3, Vendor datasheets and application notes are the best source of detailed operational information, training and errata on components designed into Shimmer. However, functional description of Communication, MicroSD storage, and power subsystems will be detailed.

2.4.1 Radio Communication

One of the key functions of the Shimmer board is its ability to communicate as a wireless platform. Shimmer has both 802.15.4 and Bluetooth radio solutions but the radio cannot be operated simultaneously.

The features and goals of either radio technology are sufficiently differentiated to lead to a choice based on application needs and available resources according to the coarse matrix below:

Metric	802.15.4	Bluetooth
Power Consumption	Better	Worse
Agility/Connection Speed	Better	Worse
Out-of-box compatibility with mass-market devices	NO	YES
Prebuilt Application	NO	YES
Number of nodes	GOOD	POOR
Range	OK	OK
Mesh Implementations	YES	NO
Ability to customize	YES	NO
FCC Modular Certification	YES	YES
Data Rate	Worse	Better

2.4.1.1 802.15.4 Radio

IEEE 802.15.4 is a specification of a very low-power wireless personal area network (WPAN) protocol. It specifies the physical layer (air interface-layer 1) and the accompanying MAC protocol layers. 802.15.4 is a CSMA/CA MAC based system with a total of 27 channels specified in the frequency bands of 2.4 GHz, 902-928 MHz, and 868.3 MHz. Three different over-the-air data rates can be allocated: 16 data channels with a data rate of 250 kb/s, 10 channels with a data rate of 40 kb/s and 1 channel with a data rate of 20 kb/s. Such a network can choose one of the 27 channels depending on availability, congestion state and the data rate of each channel. It is optimised for short range (typically 30-50 meters), low data throughput with a 30 ms network join time and supports flexible topologies, i.e. star or peer-to-peer topologies. It also supports a very large numbers of nodes. A single 802.15.4 network can accommodate up to 216 devices, which are assigned during the association procedure. It is designed to achieve good energy efficiency both in the physical and MAC layers. The duty cycle of communications in an 802.15.4 network is around 1 percent, resulting in very low average power consumption for static and dynamic environments. However, it is also up to higher protocol layers to observe the low duty cycle. Most power saving mechanisms in 802.15.4 are based on beacon-enabled mode. The simplicity, low-cost, low-power features of 802.15.4 are intended to enable broad deployment of wireless networks which are able to run for years on standard batteries, for a typical monitoring application [3].

802.15.4 Channels 15, 20, 25, and 26 are recommended as they are relatively free from interference from common 802.11b networks [29].

For IEEE 802.15.4 compliant wireless communication the Shimmer platform contains the SR7 Radio module, a Shimmer Research devices that integrates a TI Chipcon CC2420 radio transceiver [14] and chip-type antenna. SR7 ideal for Shimmer as it is designed for size-constrained low-power and low-current applications (current usage 17.4mA for transmission and 18.8 mA for reception). The radio may also be turned off by the MSP430 for low power operation. Users can expect to achieve 5-10M of indoor communication range and 30M in a free space environment. Users are encouraged to experiment with orientation and placement as antenna is polarized.

The CC2420 radio has an inherent direct sequence spread spectrum modem and a theoretical data rate of 250Kbps. The CC2420 is controlled by an SPI connection over the USART1 and with the CC2420 having support for applications such as packet handling, data transmissions, data encryption, received signal strength, link quality and packet timing, the work load on the MSP430 controller is reduced. The CC2420 requires only a few extra additional components such as a reference crystal oscillator and does not necessitate and external filters.

2.4.1.2 Bluetooth (IEEE 802.15.1)

Bluetooth is a low-cost, low-power, robust, short-range wireless communication protocol which was initially founded by Ericsson in 1994 to replace traditional mobile phone and computer cables with wireless links. It operates in the license free 2.4 GHz ISM (industrial, scientific, medical) band with a short range (power-class-dependent: 1 metre, 10 metres, 100 metres) based on low-cost transceiver microchips in each device. With the introduction of the (EDR) Enhanced Data Rate feature [9] devices can communicate with each other at up to 3Mbps. The Bluetooth special interest group (SIG) was founded in 1998 by companies such as Ericsson, Nokia and Intel and the core system consists of an RF transceiver, baseband, and protocol stack. Bluetooth radios are designed for busy environments with many users. Up to eight Bluetooth devices can communicate together in a network called a piconet. The piconet is a point to multipoint network consisting of one master and up to seven slave devices. Multiple piconets can coexist and join together to form scatternets. Bluetooth uses 79 1MHz channels to transmit data. Interference between other ISM band devices (802.11 and 802.15.4 devices) and other Bluetooth piconets is minimised using frequency hopping spread spectrum (FHSS), where the carrier is rapidly switched (hops) among the 79 available channels. The frequency hopping sequence is controlled by the master within the piconet. Other Bluetooth interference reduction techniques include adaptive power control, Channel Quality Driven Data Rate (CQDDR) and Adaptive Frequency Hopping (AFH) [10]. Extensive documentation and analysis of Bluetooth and its applications can be accessed from the Bluetooth SIG’s website at www.Bluetooth.org.

Microsoft Windows currently only supports a single Bluetooth Piconet, limiting users to seven simultaneously attached devices. Linux support multiple Piconets and exposes the entire Bluetooth stack in open source software, for users interested in doing advanced or special purpose development with Bluetooth.

The Shimmer platform uses the Roving Networks RN-42 [16] Class 2 Bluetooth module to communicate via an integrated 2.4GHz antenna. This module was found to be well engineered and very configurable [17], reliable and robust. This module contains a full Version 2 Bluetooth Protocol Stack and supports the Serial Port Profile which facilitates rapid application development. The Bluetooth module is connected to the MSP430 directly via the USART1 serial connection. It can also be controlled by ASCII strings over the Bluetooth RF link. The RN-42 has a range of more than 10 metres (33 feet) and the transmitted power can be adjusted depending on the application distance. The system has seventy-nine channels with channel intervals of 1MHz and offers a robust secure link via frequency hopping spread spectrum (FHSS) and error correction schemes. Users can expect to communicate with the Shimmer USART at speeds up to 230kbaud, with 115kbaud as the default and recommended value.

As in 802.15.4 implementation, the antenna on the RN-42 module is polarized. Users are encouraged to experiment with orientation and placement.

2.4.2 MicroSD Flash Storage

2.4.2.1 MicroSD Overview

The Shimmer baseboard contains a MicroSD card socket to incorporate extra memory resources, with capacities up to with 2Gbytes. This allows the additional storage of data while the Shimmer is not streaming and ensures no loss of data while mobile, during network outages or while changing batteries.

For Shimmer compatibility the MicroSD card chosen must implement 1-bit SPI mode. The Shimmer research website (www.shimmer-research.com) contains a list of tested cards [37] or you can purchase directly from the online store.

2.4.2.2 Host Data bypass Functionality

To improve usability, Shimmer incorporates a wide bandwidth analog MUX and tri-state logic buffering on certain signals routed to the external connector to provide direct and immediate access to flash memory using an external SD-flash card controller (SDHOST) for high-speed data transfer

When placed in a powered dock, the battery charger indicates power-good. This signal notifies the CPU of a dock event and can be used to toggle the MUX switches. The external ADC inputs, Serial flow-control pins, MicroSD flash card and external connector SPI bus functions are no longer available to Shimmer applications. This switch normally happens automatically when a Shimmer is placed in a dock when the MicroSD is unused by the Shimmer application.

Shimmer Applications that use the MicroSD card will require firmware that allows for the SD specification's requirement of a power-cycle to change from SPI mode- the card talking to the MSP430- back to SDIO mode, when the card is controlled by the USB flash media controller. Power-cycling the card requires explicit control of the card's interface pins, setting them to *LOW* (zero Volts).

Specifically, the DOCK_N pin must be set to output and GPIO function, set to *HIGH* (3.0 Volts), to ensure MSP430 control of the card during docking events. Normal operation configures the DOCK_N pin as an input and GPIO function with interrupt enabled. A detailed treatment of this operation is out of scope here, but can be found in the TinyOS driver for the SD device.

In TinyOS-1.x, consult *contrib/handhelds/tos/lib/SD/SDM.nc*

In TinyOS-2.x, consult *tos/platforms/shimmer/chips/sd/SDP.nc*

The Host data activity indicator on the USB Reader dock presented earlier in this manual provides helpful status information.

2.4.3 Power

Device operating life will depend on application and battery selection however design goal for use as a long-term motion capture device is 1-10 days of operating life from a 280mAh cell while acquiring multichannel data with periodic radio communication.

The design supports both Lithium-Ion/Lithium-Poly cell chemistry and lithium coin cells and alkaline batteries. Device safety must be maintained by integrating battery polarity protection, charge monitoring and failsafe battery over/under voltage and over-discharge limits in common mobile environments and while AC-powered.

A push-button power controller is used to control board power-on sequence. From an "off" state, board reset is low and the board power regulator is disabled. When the reset button is pushed, the regulator is enabled and the processor is brought out of reset after a short delay. Short subsequent reset button pushes will generate a board reset. A long button push (preset to 6 seconds but HW customizable) will shutdown the board regulator. When the battery voltage drops below ~3V a "Kill" signal is also generated to power-off the board. Combined with software battery monitoring, these features simplify user interaction and firmware power-monitoring while facilitating deployment. Occasional user indication of on-power state is recommended for all applications.

Soft power switching is provided for both the Bluetooth radio module and MicroSD socket (see SD card discussion in 5.2 for more information on MicroSD card power control). When Shimmer is placed in a powered dock, logic will over-ride any application settings and force these switches "on" for use with an external SDHOST controller. The Accelerometer and 802.15.4 Radios have integrated shutdown functionality. The Digital serial number IC also has an automatic power-down feature to save power when the device is idle.

A Shimmer that has been powered-off using the reset button or due to battery discharge cannot be programmed or have its flash accessed by the dock until it has been turned back on. Pressing the reset button will turn the board-power back on, restoring all functionality (even if the battery is dead). Given the micro-power characteristics of Shimmer, the board power switch is primarily intended for periods of long-inactivity.

Shimmer's on-board regulator can provide 100mA continuous current and tolerates surges until a thermal limit is reached. Expansion devices must be current limited or have independent regulation running off of the PV_REG expansion pin. The following table provides estimates of peripheral power draw:

Case	Regulator Current*	Notes
CPU Active	5mA	Worst case, high application load at 8MHz
CPU Active	.1mA	Typical use
CPU Idle	.02mA	Idle (TinyOS)
CPU Suspended	.002mA	Suspend to RAM—not supported by TinyOS
ADC Active	1mA	During S/H Operation
ADC idle	<.010mA	During power-down
Bluetooth™ Active	45mA	Worst case-- ~20mA once paired regardless of data payload
802.15.4 radio Active	20mA	Worst case per data sheet
802.15.4 Typical application	.9mA	Measured using IP-over-802.15.4 stack for device control and data transfer
802.15.4 radio in sleep mode	<.03mA	See CC2420 Datasheet for more detail
MicroSD Peak	47mA	Peak, Worst Case
MicroSD Typical application	3mA	40mA peak, linear rise/fall over 400uS per R/W
Flash Idle	.150mA	
Sleep	.140mA	Entire board

* Expected values

Exceeding the 100mA limit of the regulator while possible is not recommended without detailed analysis and qualification of both electrical and thermal design margin.

2.4.4 Voltage and power measurement

When the PMUX is configured for voltage measurement (prempting ADC channels 6 and 7 which are normally available on the dock connector), it is possible to measure scaled battery voltage and regulator input voltage. The difference between these two values can be used to calculate the instantaneous current.

Battery voltage is divided by two using the battery voltage divider resistors. The accuracy of this value is +/-2% based on resistor tolerances. The Battery also passes thru an SBR130S3 diode before another divider (Regulator Voltage) is used for measurement of board regulator input voltage. That measurement is also +/- 2%.

The SBR130S3 diode, like all diodes has an exponential relationship between voltage drop, and current. As a result it is possible to measure current across a wide operating range. The table below can be used to estimate current or calculate coefficients for an exponential curve fit. When using the current measurement feature, Developers need to chose an appropriate sample rate or use sophisticated sample averaging methods or triggered data acquisition. For example, a radio transaction may happen in a few ms, a flash write in 10ms.

SBR130S3 Diode

I (ma)	Vf (mv)
163.5	297
140.8	293
117.6	287
98.2	280
81.3	273
69.9	270
59.52	264
50.01	259
41.6	254
34.44	248
28.53	242
24.25	237
19.47	231
16.34	226
13.52	220
11.09	215
9.13	209
7.74	204
6.45	199
5.44	195
4.45	190
3.72	185
3.06	180
2.793	177
2.367	172
1.931	167
1.633	162
1.346	156
1.105	151
0.907	146
0.776	142
0.6445	137
0.5466	132
0.4512	126
0.3803	123
0.3125	117
0.2611	113
0.2097	108
0.1749	101
0.1437	97
0.1181	92

3 Embedded Software

3.1 Software Introduction

The TinyOS operating environment is highly recommended for design, implementation, testing and validation of Shimmer embedded software (firmware). TinyOS offers economy due to the extensive cross-platform open-source code library. Reusing applications on Shimmer is a key advantage of the TinyOS environment and significantly accelerates the software development and validation process. Shimmer's platform code is actively maintained at:

https://tinyos-main.googlecode.com/svn/trunk/ (tinyos-2.x)

http://tinyos.cvs.sourceforge.net/tinyos/ (tinyos-1.x and tinyos-2.x-contrib)

TinyOS-1.x code can be found in the contrib/handhelds/ directory.

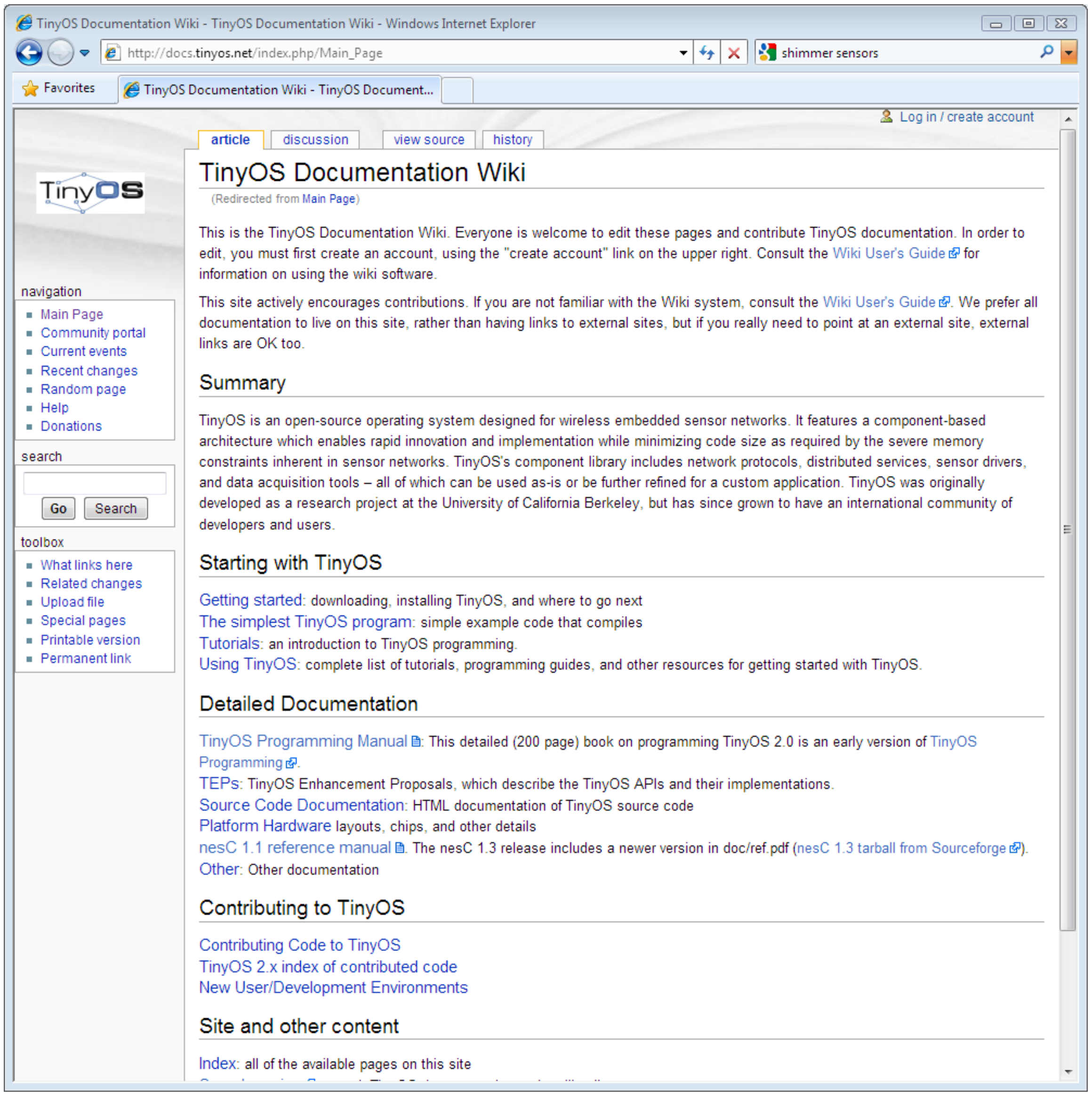
Current functionality includes:

- MicroSD flash storage
- FAT File system
- IP stack for 802.15.4
- Bluetooth configuration, connection management and streaming data transfer
- Time and clock configuration
- Peripheral control and configuration
- Power supply monitoring

3.2 TinyOS/nesC Programming

This section will explore the TinyOS operating environment and its programming language, nesC. The TinyOS developer community maintains a documentation wiki:

http://docs.tinyos.net



TinyOS Documentation Wiki

(Redirected from Main Page)

This is the TinyOS Documentation Wiki. Everyone is welcome to edit these pages and contribute TinyOS documentation. In order to edit, you must first create an account, using the "create account" link on the upper right. Consult the Wiki User's Guide for information on using the wiki software.

This site actively encourages contributions. If you are not familiar with the Wiki system, consult the Wiki User's Guide. We prefer all documentation to live on this site, rather than having links to external sites, but if you really need to point at an external site, external links are OK too.

Summary

TinyOS is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools – all of which can be used as-is or be further refined for a custom application. TinyOS was originally developed as a research project at the University of California Berkeley, but has since grown to have an international community of developers and users.

Starting with TinyOS

Getting started: downloading, installing TinyOS, and where to go next
The simplest TinyOS program: simple example code that compiles
Tutorials: an introduction to TinyOS programming.
Using TinyOS: complete list of tutorials, programming guides, and other resources for getting started with TinyOS.

Detailed Documentation

TinyOS Programming Manual: This detailed (200 page) book on programming TinyOS 2.0 is an early version of TinyOS Programming.
TEPs: TinyOS Enhancement Proposals, which describe the TinyOS APIs and their implementations.
Source Code Documentation: HTML documentation of TinyOS source code
Platform Hardware layouts, chips, and other details
nesC 1.1 reference manual. The nesC 1.3 release includes a newer version in doc/ref.pdf (nesC 1.3 tarball from Sourceforge).
Other: Other documentation

Contributing to TinyOS

Contributing Code to TinyOS
TinyOS 2.x index of contributed code
New User/Development Environments

Site and other content

Index: all of the available pages on this site

For a brief introduction to TinyOS programming, you should go through the "Getting Started" tutorials and "TinyOS Programming" manual [19], [20]. nesC coding conventions can be found in the official coding standards [21].

3.2.1 Introduction to TinyOS and nesC

TinyOS is an event based operating environment designed for use with networked sensors. This programming environment supports a variety of low power devices, with a few kilobytes of memory and wireless communication capabilities [19]. It is designed to support the concurrency intensive operations required by networked sensors with minimal hardware requirements. TinyOS is based on the nesC programming language. It is an extension to C [22] designed to embody the structuring concepts and execution model of TinyOS [23] and uses the custom nesC compiler. The nesC language supports programming structured component based applications. The source code for both nesC and TinyOS is available on SourceForge. The TinyOS system, libraries and applications are written in nesC. The language supports the TinyOS concurrency model based on tasks and hardware event handlers. TinyOS executes a program using two threads, one containing tasks and another containing hardware event handlers [24]. The nesC compiler detects data races at compile time. Tasks are scheduled by TinyOS but are run to completion and do not pre-empt each other. Hardware event handlers are initiated by hardware interrupts and may pre-empt tasks or other event handlers and also run to completion. The problem with threads in real time operating systems is that relatively speaking they require a large amount of RAM. Each thread has its own private stack which has to be stored when a thread is waiting or idle. RAM is a constrained resource on sensor node platforms and therefore TinyOS only uses the two threads discussed [20].

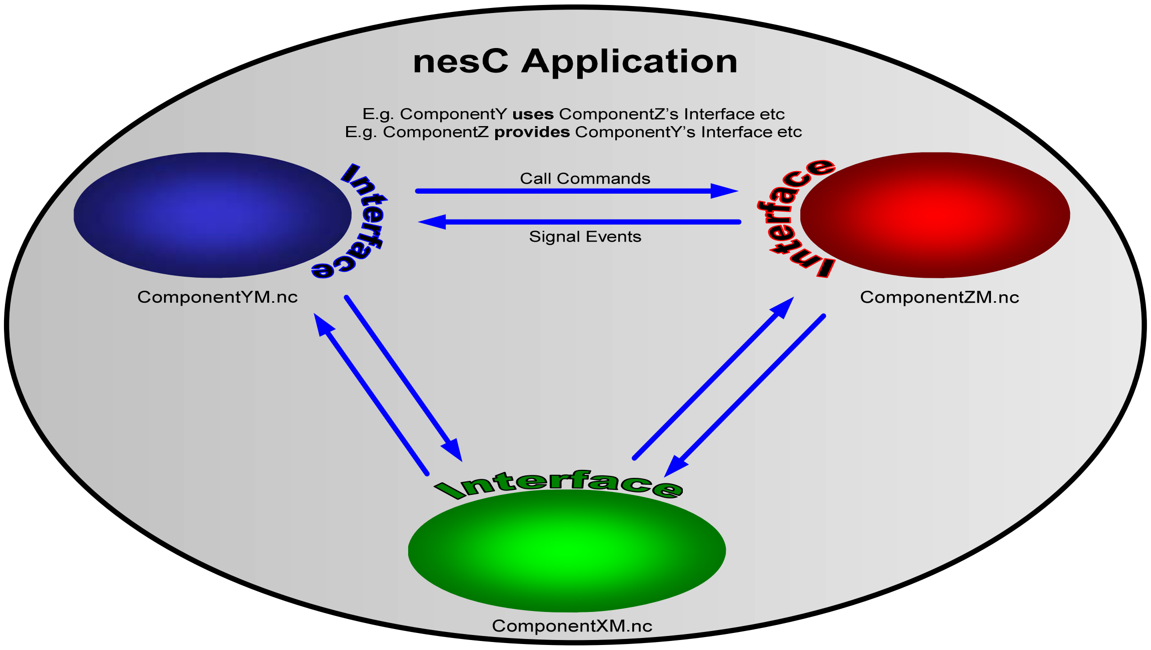
3.2.2 TinyOS Program Architecture

For experienced C or C++ developers, writing a small nesC programs is a relatively simple task, requiring implementation of one or more modules and wiring them together. The difficulty comes when building larger applications. TinyOS modules are fairly analogous to C source files, but configurations – which stitch modules together– are not. We will discuss configurations in more depth in the "Configurations and Wiring" section but before beginning to understand TinyOS and nesC, there are a number of important definitions that need to be understood. Also, the following keywords are new for nesC. Please see the Language Reference Manual [23] for full details.

as, call, command, components, configuration, event, implementation,

3.2.2.1 Applications

An application is composed of one or more components linked together to form an executable. Every nesC application is described by a top-level configuration that wires together the components inside.



3.2.2.2 Components

Components are the building blocks of nesC applications. A component provides and uses well-defined bidirectional interfaces. In some ways, nesC components are similar to C++ or Java objects [25]. For example, they encapsulate state and couple state with functionality. However, unlike C++ and Java objects, which refer to functions and variables in a global namespace, nesC components use a purely local namespace. This means that in addition to declaring the functions that it implements, a component must also declare the functions that it calls. Every component has a "specification", a code block that declares the functions it provides (implements) and the functions that it uses (calls). For example, this is the specification for a fictional component SmoothingFilterC, which smoothes raw data:

```
module SmoothingFilterC {

  provides command uint8_t topRead(uint8_t* array, uint8_t len);

  uses command uint8_t bottomRead(uint8_t* array, uint8_t len);

}
```

nesC has two kinds of components: configurations and modules. Configurations wire components together [26]. Modules, in contrast, are implementations. Configurations connect the declarations of different components, while modules define functions and allocate state.

3.2.2.3 Modules

A module provides application code, implementing one or more interfaces.

3.2.2.4 Configurations

Configurations are used to assemble other components together, connecting interfaces used by components to interfaces provided by others. This is called “wiring”.

3.2.2.5 Interfaces

An interface is bi-directional and acts as the only point of access to a component (See Figure 3.1). An interface declares a set of functions called “commands” that the interface provider must implement and another set of functions called “events” that the interface user must implement. For a component to call the commands in an interface, it must implement the events of that interface. A single component may use or provide multiple interfaces and multiple instances of the same interface [19].

3.2.2.6 Commands

Commands are functions that an interface provider must implement. Commands are called using the “call” keyword.

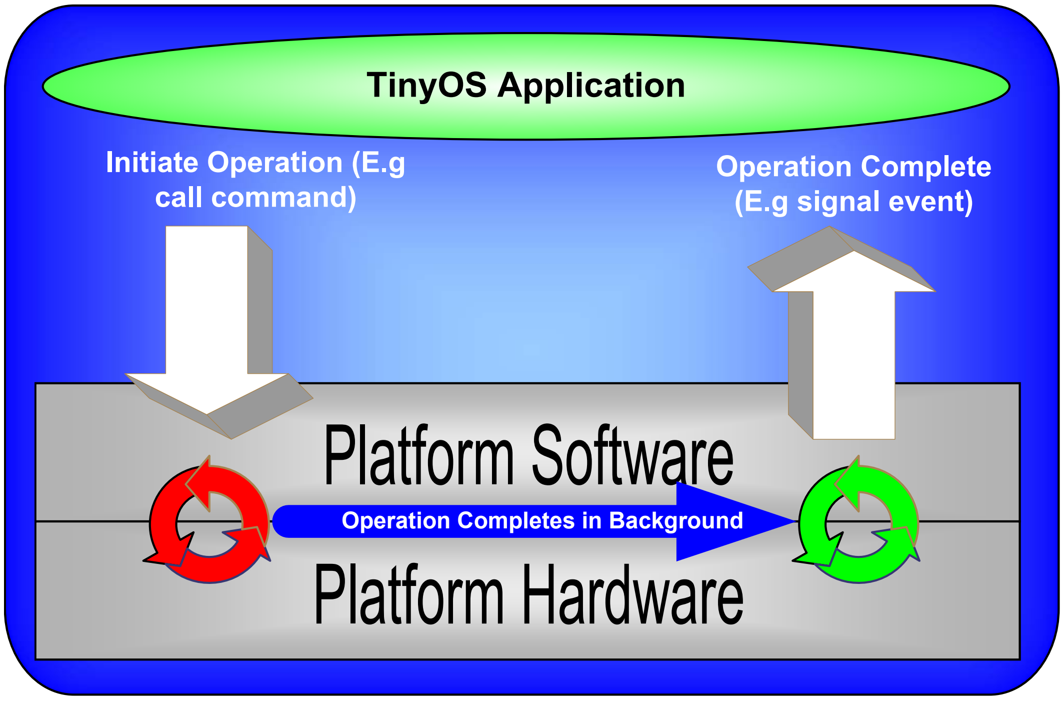
3.2.2.7 Events

Events are declared by an interface provider but the user of the interface must implement them based on their requirements. Events are triggered/signalled using the “signal” keyword.

NOTE: nesC uses the filename extension ".nc" for all source files including interfaces, modules, and configurations. Please see the official nesC coding standards for more details [21].

3.2.3 TinyOS Concurrency Model

As mentioned in the chapter Introduction, TinyOS executes a program using two threads, one containing tasks and another containing hardware event handlers. Tasks are functions whose execution is deferred. Once scheduled, they run to completion and do not pre-empt one another. Hardware event handlers are executed in response to a hardware interrupt and also run to completion, but may pre-empt the execution of a task or other hardware event handler. Because sensor nodes have a broad range of hardware capabilities, TinyOS has a flexible hardware/software boundary. In real-time embedded systems hardware operations are almost always split-phase rather than blocking [26]. An important characteristic of split-phase interfaces is that they are bidirectional, there is a down call to start the operation, and an up call (asynchronous event) that signifies that the operation is complete.


TinyOS Application
Initiate Operation (E.g call command)
Operation Complete (E.g signal event)
Platform Software
Operation Completes in Background
Platform Hardware

Blocking on the other hand refers to block an application completely and waiting (sometimes called busy waiting) until an operation completes. For example, to acquire a sensor reading with an analog-to-digital converter (ADC), software writes to a number of configuration registers to start a sample. When the ADC sample completes, the hardware issues an interrupt, and the software reads the value out of a data register. In TinyOS, operations that are split-phase in hardware are also split-phase in software. This means that many common operations, such as sampling sensors and sending packets, are split-phase.

3.2.3.1 Tasks

In some real time operating systems, tasks and threads can mean the same thing (or at least the two terms are used interchangeably) however tasks in TinyOS are not the same as threads; there are only two threads of execution and tasks make up one of these threads. Tasks are non-pre-emptive. This means that only one task runs at any time, and TinyOS doesn't interrupt one task to run another. Once a task starts running, no other task runs until it completes. This means that tasks run independently with respect to one another. This has the advantage that you don't need to worry about tasks interfering with one another and corrupting each other's data. However, it also means that tasks should be kept short. If a component has a very long computation to execute, it should be broken into multiple tasks. A module can post a task to the TinyOS scheduler using the post keyword. A task is not called immediately but rather at some point later, the scheduler will execute the task.

```
task void sendData() {

  call Bluetooth.write(msgbuf, strlen(msgbuf));
```

3.2.3.2 Hardware Event Handlers

Tasks allow software components to emulate the split-phase behaviour of hardware [26]. Event driven techniques like this are common in embedded software where a quick response time and handling of events is the cornerstone to efficient real time embedded systems. However they have much greater utility than these low level tasks. They also provide a mechanism to manage pre-emption in the system. As tasks run independently with respect to one another, code that runs only in tasks can be rather simple as there is no danger of another task suddenly taking over and modifying data inadvertently. However, interrupts do exactly that, they asynchronously interrupt the current execution and start running pre-emptively. The basic problem with pre-emptive execution is that it can modify state underneath an ongoing computation, which can cause a system to enter an inconsistent state. In nesC and TinyOS, functions that can run pre-emptively, from outside task context, are labelled with the "async" keyword.

```
  async event void Bluetooth. connectionMade () {

    call Leds.greenOn();

  }
```

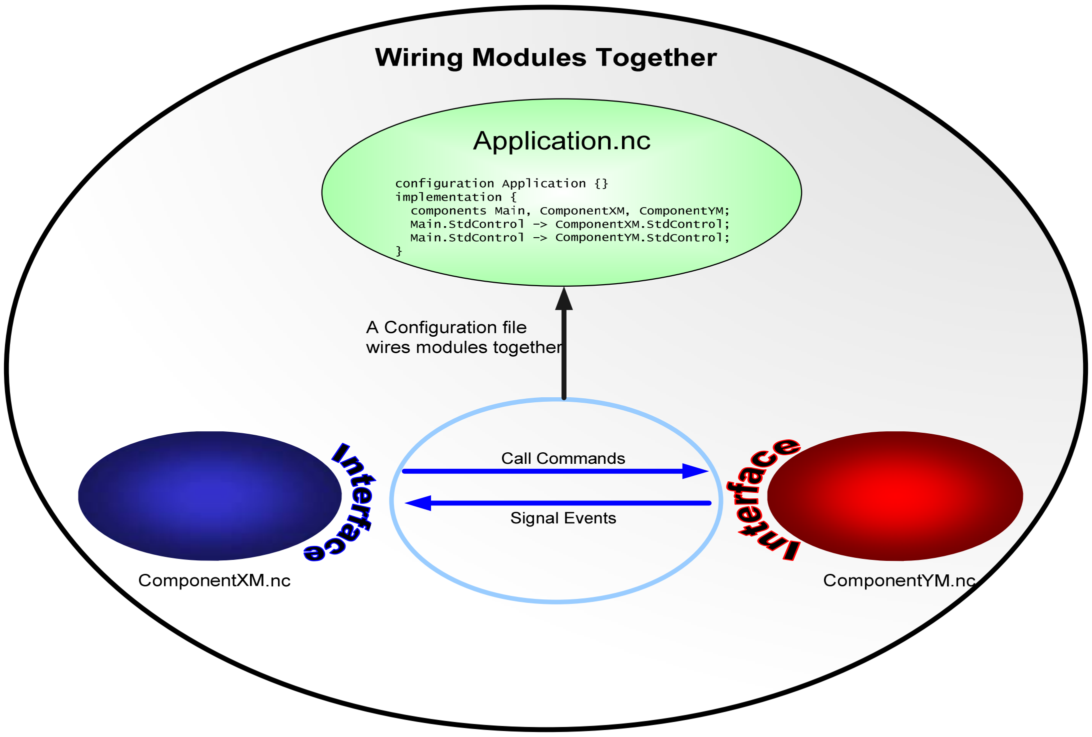
A function that is not asynchronous is synchronous (often call “sync” for short). By default, commands and events are “sync”, the async keyword specifies if they are not. Interface definitions specify whether their commands and events are async or sync. All interrupt handlers are async, and so they cannot include any sync functions in their call graph. The only way that an interrupt handler can execute a sync function is to post a task. A task post is an async operation, while a task running is sync. The problems interrupts introduce means that programs need a way to execute snippets of code that will not be pre-empted. NesC provides this functionality through atomic statements. For example:

```
command bool increment() {

  atomic {

    a++;

    b = a + 1;

  }

}
```

The atomic block promises that these variables can be read and written atomically. Note that this does not guarantee that the atomic block will not be pre-empted. Even with atomic blocks, two code segments that do not touch any of the same variables can pre-empt one another. The rule for when a variable has to be protected by an atomic section is simple; if it is accessed from an async function then it must be protected. nesC also checks to see whether variables are not protected properly and issues warnings at compile time when this is the case.

3.2.4 Configurations and Wiring

As discussed above under TinyOS Program Architecture, there are two types of components in nesC, “modules” and “configurations”. Modules provide application code, implementing one or more interface. Configurations however are used to assemble other components together, connecting interfaces used by components to interfaces provided by others. This process is called “wiring”. Every nesC application is described by a top-level configuration that wires together the components inside. For one module to interact with another, a set of names in one component, generally an interface, have to be mapped to a set of names in another component. In nesC, connecting two components in this way is called wiring. After modules, configurations are the other type of component in nesC whose implementation is component wirings itself. Modules will implement program and configurations compose modules into larger abstractions.



```
configuration Application {}
implementation {
  components Main, ComponentXM, ComponentYM;
  Main.StdControl -> ComponentXM.StdControl;
  Main.StdControl -> ComponentYM.StdControl;
}
```

4 Using the Live Distribution

4.1 General information on the Live Distribution

The Live Distribution offers a convenient way to start experimenting with the TinyOS programming environment - it is based on the stable and well-proven Ubuntu Linux distribution and it is very much just like having all the steps described in detail in the next few chapters already done for you! Using the existing Live Distribution you can:

- Boot from Live Distribution and have a uniform environment on any standard x86 computer at any time (the environment includes TinyOS 1.xx and now also TinyOS 2.xx but also many other programs including OpenOffice);
- Do a full installation of Linux including the development environment for Shimmer.
- In the /NON-LIVE directory of the Live Distribution, you will find VMware Player, and the Shimmer development virtual machine. See the Quickstart section of this manual or Readme file for instructions.

The next paragraphs will describe in detail each of the above options.

Important details:

- the username for logon is **tiny1** and the password is also **tiny1** ; more recent versions will log you automatically the first time but the password will still be needed for administrative actions; there is also a second login **tiny2** with password **tiny2** for TinyOS-2.x development.
- UNIX is case-sensitive so Blink is different from blink.
- By default the Live system will start with the US keyboard layout. Keyboard layout can be changed from the top toolbar and eventually saved in persistent mode.
- The time zone might not be correct by default
- **ALL THAT YOU DO IN A LIVE SESSION WILL BE LOST WHEN YOU RESTART** (unless you use the 'persistent' mode or you copy your work on some disk or over the network).

4.2 Booting from USB

Any modern x86 computer with at least 512 MB RAM that can read and boot from a USB Flash Drive should be able to use the Live Distribution. Some computers will require you to enter the BIOS menu and change the device priority so that a USB device- sometimes "removable media" - appears before the hard drive.

Initially a boot prompt will be displayed, and if <Enter> is pressed or no action is taken for about 15-30 seconds the boot will continue with the default configuration - the full boot sequence might take up to 1-5 minutes.

The username for logon is **tiny1** and the password is also **tiny1** - all the TinyOS build tools are already installed/preconfigured and in the home directory for that user the full CVS image of the TinyOS 1.x project is found in the **tinyos-1.x** folder - you can check it by opening a terminal window and entering:

```
cd ~/tinyos-1.x/ contrib/handhelds/apps/Blink
```

```
make shimmer2r
```

You will see a compilation message on the terminal. If you have a Shimmer (powered-on) in a programming dock connected to your computer via a mini-USB cable and no other 'serial-over-USB' port plugged-in you can also try:

```
make shimmer2r install bsl,/dev/ttyUSB0
```

The simplest way to confirm that you are using the correct device is to remove the dock and type at a prompt:

ls -l /dev/ttyUSB*

You might see nothing; that's OK. Now, plug the dock back in and the dock's two ports will also appear. When programming, choose the first of the two.

Congratulations, you just programmed your Shimmer!

Now you can start using TinyOS - but please remember that **ALL THAT YOU DO IN A LIVE SESSION WILL BE LOST WHEN YOU RESTART** (unless you use the 'persistent' mode or you copy your work on some disk - either a local disk - including Windows NTFS disks - or a network share).

TinyOS-2.x is also available; a second login of tiny2/tiny2 accesses this environment. Switch users by logging out and logging back in as tiny2. Enter tiny2 as the password.

As before, you can check the build entering:

```
cd /home/tiny2/tinyos-2.x/apps/Blink
```

```
make shimmer2r
```

and if you have a Shimmer in the dock and no other 'serial-over-USB' port plugged-in you can also try:

```
make shimmer2r install bsl,/dev/ttyUSB0
```

(the Blink application itself might be a little different in TinyOS v2 than in TinyOS v1).

4.3 Linux installation

The Live Distribution can also be used to install Ubuntu on a hard-drive - this way you will get a 'normal Linux environment' but with TinyOS already installed; and this mode will also be able to get automatic updates and specific optimizations.

In order to install Ubuntu you just boot from the Live Distribution and click on the 'Install' icon from the desktop.

The installer is pretty much the standard Ubuntu installer with a minor twist - any user that you will create in the installer will actually be lost and instead the **tiny1** user will be kept - after you finish the installation and you restart in the freshly-installed Ubuntu you should change the **tiny1** password and you can also create other users (but remember that the entire TinyOS-v1.x source is under /home/tiny1)!

4.4 Virtual Machine Installation

The Live Distribution includes a compressed image of a fully configured Linux development environment. To use it you must first uncompress the image with 7-Zip, we supply a windows installer for this application in the /NON-LIVE/VMware directory. Once 7-Zip installed, browse to the ShimmerLive_xxx.7z image and extract it by right clicking its icon and selecting 7zip->Extract to and completing the dialog box with the location of your choice.

The resulting VMware .vmx file is then executed by the VMware Player.

IMPORTANT: In order to access the programming dock as described in the next section, first expand the Virtual Machine menu on the VMware titlebar and open Removable Devices. Select a peripheral to connect/disconnect from host. When a peripheral is connected to VMware it is no longer visible to the host OS and vice versa. The future devices shimmer usb reader is required for programming and serial communication. If you wish to access the contents of the Shimmer MicroSD from VMware you should connect the Standard Shimmer2 Reader.

At this point you can open a terminal window and continue with the Live Distribution Quickstart Tinyos-1.x or Tinyos-2.x directions below.

The installed environment can be updated as with a conventional Linux distribution. CVS updates to the TinyOS directories can be performed in place. For more information see Section 5.1 below.

4.5 Other files on the Distribution

Other files are already placed on the Live Distribution in case you find yourself without an internet connection.

The files that are not related to the actual Live Distribution booting are located in a folder **/NON-LIVE** and under that you can find:

- **/NON-LIVE/DOCS** - various manuals, including a PDF version of this document, manuals for MSP430 and so on.

- **/NON-LIVE/offline_linux** - files needed for offline setup on Linux.

-**/NON-LIVE/setup_win** - files for other programs that can be installed in Windows including ShimmerConnect and QtOctave.

- **/NON-LIVE/FIRMWARE** - pre-compiled firmware for the Shimmer together with a small Windows executable application to program Shimmer.

- **/NON-LIVE/VMware** - A compressed image of the Linux VMware Player virtual machine that can be used as a preconfigured development environment - please note that when VMware products are first installed in Windows a restart is needed!

5 Developing Shimmer Firmware

5.1 Getting TinyOS Source

The Shimmer platform code is actively maintained at:

http://tinyos.cvs.sourceforge.net/tinyos/tinyos-1.x

https://tinyos-main.googlecode.com/svn/trunk/

http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib

For TinyOS-1.x, Shimmer platform support and application code examples are fond under the contrib/handhelds directory. TinyOS-2.x is the current version, and platform code is found under tinyos-main/tos/platforms, while application examples are found in tinyos-2.x-contrib/shimmer. Note that the tinyos-main repository now uses the subversion (svn) version control system, not cvs.

CVS and **SVN** are open source version control systems that facilitate software configuration management. They are used by many software developers to manage changes within their source code tree. They provide the means to store not only the current version of a source code element (E.g. a *.nc source file), but a record of all changes that have occurred to that source code. For further details on CVS commands, visit the CVS website [25] and http://sourceforge.net/docs/E04/. Subversion documentation is at http://subversion.apache.org/docs.

The latest TinyOS source can be downloaded via CVS from the TinyOS project page at SourceForge [27]. However the remainder of this manual assumes that a developer will maintain their own CVS tree. Once you have a CVS client on your system, then even without a SourceForge account setup the TinyOS CVS repository may be accessed through an anonymous server (pserver) with the following commands.

This command will log you in to CVS server anonymously, just hit enter when prompted for a password.

```
cvs -d:pserver:anonymous@tinyos.cvs.sourceforge.net:/cvsroot/tinyos login
```

This command will checkout a TinyOS repository of which there are three.

```
cvs -z3 -d:pserver:anonymous@tinyos.cvs.sourceforge.net:/cvsroot/tinyos
checkout -P <repository>
```

For TinyOS1, substitute *tinyos-1.x* for the repository field:

```
cvs -z3 -d:pserver:anonymous@tinyos.cvs.sourceforge.net:/cvsroot/tinyos
checkout -P tinyos-1.x
```

For TinyOS2-contrib, substitute *tinyos-2.x-contrib*

For the main TinyOS2.x repository, use

```
svn checkout https://tinyos-main.googlecode.com/svn/trunk/ tinyos-main
```

5.2 Creating a Build Environment on Linux

5.2.1 Update Environment Variables

Add the following environment variables to your environment permanently by adding to your .bashrc file:

```
export TOSDIR=~/tinyos-1.x/tos

export TOSROOT=~/tinyos-1.x

export MAKERULES=`ncc -print-tosdir`/../tools/make/Makerules

export TINYOS_NP=none
```

For TinyOS2, the variables are analogous except for MAKERULES:

```
export MAKERULES=`ncc -print-tosdir`/../support/make/Makerules
```

After modifying these environment variables, log out and log back in; otherwise run the tools from a shell that has updated its environment by including the following:

```
source ~/.bashrc
```

5.2.2 Install nesC

Prerequisite: Install Sun's Java SDK from your Linux distribution's package management system. RPM-based systems will have a package called **java-1.6.x-sun-devel** or a close variation; debian--based systems will have a package called **sun-java6-jdk**. The installation will be complete when the "javac" binary is visible in your environment.

```
    javac

    Usage: javac <options> <source files>
```

Download the latest nesC release from http://sourceforge.net/projects/nescc. Using the current nesC version (1.3.2) as an example- unpack the package (anywhere familiar is fine):

```
      tar zxvf nesc-1.3.2.tar.gz

      cd nesc-1.3.2
```

Install the package (you might also need the build-essentials package in case automake it is not already installed):

```
./configure

      make

      sudo make install
```

Next you need to install the ncc, mig, etc TinyOS front-ends for nesC. These are found in both tinyos-1.x and tinyos-2.x; Best to install from the tinyos-2.x directory as it is -- with the exception of Shimmer development -- the officially maintained TinyOS version. As user tiny2, which owns the tinyos-2.x code:

```
cd tinyos-2.x/tools

./Bootstrap

./configure

make

sudo make install
```

Press the <Enter> key when prompted. This will happen a few times.

5.2.3 Install MSP430 Tools

Extract the msp430-12Jan2005.tar.gz file from the "Linux" folder on the Shimmer Developer's CD and install the tools to /usr/local/msp430.

Instead of adding /usr/local/msp430/bin to your PATH, it would be best to symbolically link each file in /usr/local/msp430/bin to /usr/local/bin, which should already be in your path.

The command below symbolically links the msp430-gcc file

```
ln -s /usr/local/msp430/bin/msp430-gcc /usr/local/bin/msp430-gcc
```

Do this for all the files in /usr/local/msp430/bin except for msp430-bsl. You might want to use a shell script for this task.

```
unalias ls

cd /usr/local/msp430/bin

for file in `ls`

>do

>ln –s /usr/local/msp430/bin/$file /usr/local/bin/$file

>done
```

Then:

```
msp430-gcc

cd $HOME

msp430-gcc

msp430-gcc: no input files
```

This response from msp430-gcc means that it is now in your environment path.

5.3 Testing the Build Environment on Linux

5.3.1 Compiler Check

To check to see if the build environment is working:

Go to the Blink application folder:

```
cd ~/tinyos-1.x/contrib/handhelds/apps/Blink
```

Or for tinyos-2.x:

```
cd ~/tinyos-2.x/apps/Blink
```

```
make shimmer2r
```

You should see the "ncc" compiler compiling the files with no errors and near completion the following should be displayed:

```
compiled Blink to build/shimmer2r/main.exe

3170 bytes in ROM

44 bytes in RAM
```

5.3.2 Burning Shimmer Images

The correct bootstrap loader should already be installed.

Before connecting for the first time the dock do:

```
ls -l /dev/ttyUSB*
```

Connect the programming board (and let it automatically install drivers if this is the first time the board has been connected), then check if the right devices are now present:

```
ls -l /dev/ttyUSB*
```

Possible problems on certain Linux distributions:

- **brltty** package (for Braille support) might be already installed and will conflict with serial-over-USB drivers;

- under some distributions (most notably Fedora / RedHat) ttyUSB are created with user/group uucp as owner and in order to use the dock for programming under a normal privileges user you need to either add your user to the **uucp group** or the following steps have to be taken:

1. add a rule as /etc/udev/rules.d/99-my-devices.rules, owned by root, with

644 rights; this rule should contain a single line:

```
KERNEL=="ttyUSB[0-9]*", NAME="%k", SYMLINK="tts/USB%n", MODE="0666",

GROUP="users", OPTIONS="last_rule"
```

2. issue the following commands to reload the new rule:

```
su

/etc/init.d/udev-post start
```

Now we are ready to program a Shimmer device. Place the Shimmer in the dock (only one orientation will work) and press the pinhole Reset Button (pinhole either on your Shimmer or on the USB Dock). This operation will ensure that the Shimmer is powered- also indicated by an illuminated charge status LED. When you have the correct devices created you can type:

```
cd ~/tinyos-1.x/contrib/handhelds/apps/Blink

make shimmer2r install bsl,/dev/ttyUSB<n>
```

Where shimmer2r is the platform, bsl is the bootstrap-loader script, and /dev/ttyUSB<n>, where <n> is the first of two numbered USB devices (viewed above using ls -l). The blinking LED application should now be running on your Shimmer hardware. Well Done!!

5.4 Creating a Build Environment on Windows with Cygwin

Cygwin is a Linux-like environment for Windows. Cygwin is no longer a recommended development environment solution.

We highly encourage use of the provided VMware virtual machine instead of Cygwin. The instructions that follow are a snapshot from 2007 and have not been tested since.

Please install Cygwin from http://www.cygwin.com into the c:\cygwin folder. When installing, please select "Install all" by default. Then open up the Cygwin command shell and proceed.

5.4.1 Update Environment Variables

In the rest of the instructions, it is assumed that the tinyos-1.x source tree is at /home/user/. Firstly, some environment variables need to be setup permanently in your system.

You can add those with an editor inside \CYGWIN\home\YOUR-USER-NAME\ .bashrc

at very end like:

…

export TOSROOT=/home/$USER/tinyos-1.x

export TOSDIR=/home/$USER/tinyos-1.x/tos

export MAKERULES=/home/$USER/tinyos-1.x/tools/make/Makerules

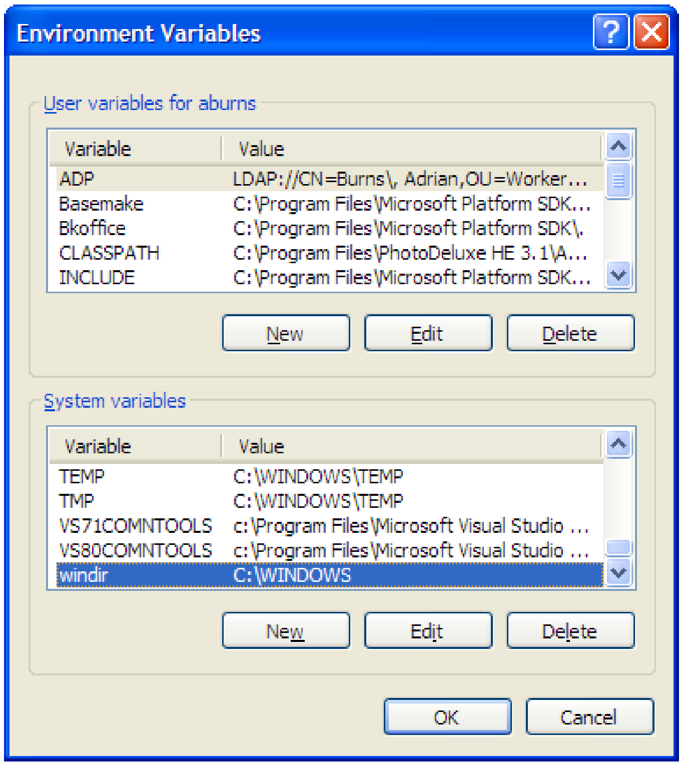
export MSPGCCROOT=/cygdrive/c/msp430-gcc

Alternatively you can add the environment variables permanently to **Windows** itself - go to Start Menu/Control Panel/System/Advanced/Environment Variables and add the 3 variables below, replacing <user> with your username.

```
TOSROOT=/home/<user>/tinyos-1.x
```

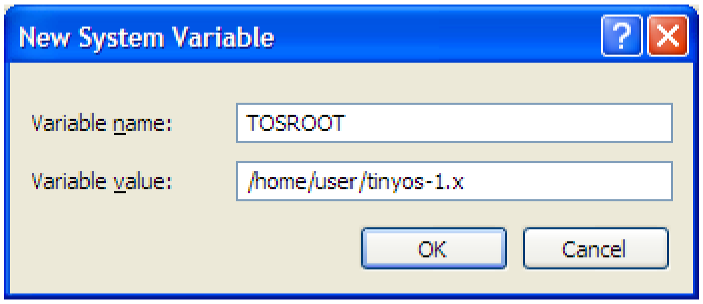
```
TOSDIR=/home/<user>/tinyos-1.x/tos
```

```
MAKERULES=/home/<user>/tinyos-1.x/tools/make/Makerules
```



Variable	Value
ADP	LDAP://CN=Burns\, Adrian,OU=Worker...
Basemake	C:\Program Files\Microsoft Platform SDK...
Bkoffice	C:\Program Files\Microsoft Platform SDK\.
CLASSPATH	C:\Program Files\PhotoDeluxe HE 3.1\A...
INCLUDE	C:\Program Files\Microsoft Platform SDK...

Variable	Value
TEMP	C:\WINDOWS\TEMP
TMP	C:\WINDOWS\TEMP
VS71COMNTOOLS	c:\Program Files\Microsoft Visual Studio ...
VS80COMNTOOLS	c:\Program Files\Microsoft Visual Studio ...
windir	C:\WINDOWS



Variable name: TOSROOT
Variable value: /home/user/tinyos-1.x

Note, that while in Bash you use Cygwin directory notation (/cygdrive/c/), in Windows you have to use DOS notation (C:\).

5.4.2 Install nesC

Please follow the Linux installation section.

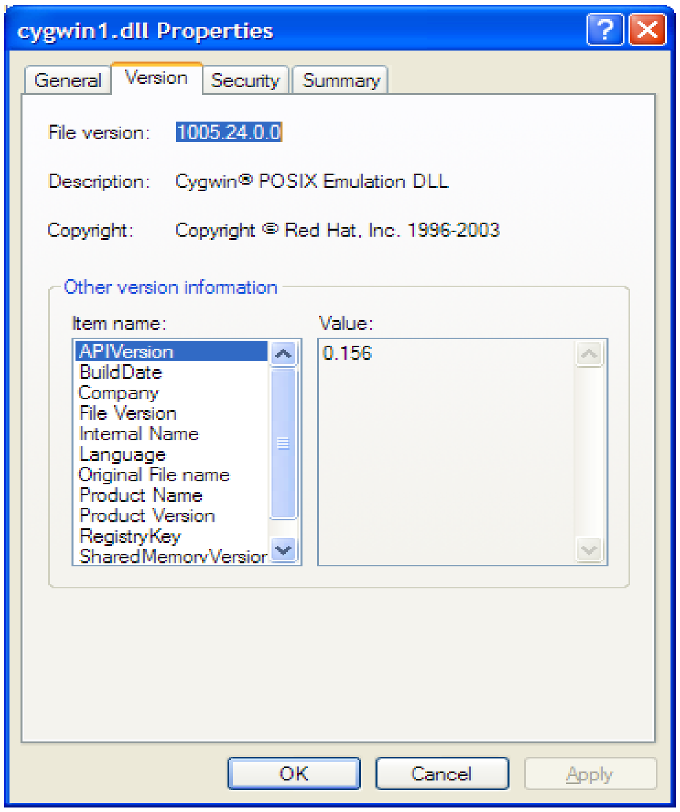
5.4.3 Install MSP430 Tools

Get the MSPGCC toolchain (mspgcc-20061126.exe) from http://mspgcc.sourceforge.net. Follow the download link http://sourceforge.net/project/showfiles.php?group_id=42303 and install the Windows version (mspgcc-win32).

Install the compiler at c:\msp430-gcc. If you use the default download folder (c:\mspgcc) then be sure to change it accordingly as you proceed through the details below. Then add the following environment variable:

```
MSPGCCROOT=/cygdrive/c/msp430-gcc
```

After installation, ensure that you restart the Cygwin shell again to pick up the new environment variable. Type "msp430-gcc" at the Cygwin command prompt and you should get a response "msp430-gcc: no input files". However, the more likely scenario is that you will get an error message stating that you have multiple cygwin1.dll files on your system and this is a problem. Find all versions of this file and remove all (there might be one in the mspgcc folder) except the newest version by right clicking on the file and checking properties as in Figure 3.4. The newest version has to be in c:\cygwin\bin (assuming that you installed Cygwin at c:\cygwin). Once you make sure that you have only one copy of cygwin1.dll in your file system, run msp430-gcc again and this should now work.


cygwin1.dll Properties

General | Version | Security | Summary

File version: 1005.24.0.0

Description: Cygwin® POSIX Emulation DLL

Copyright: Copyright © Red Hat, Inc. 1996-2003

Other version information

Item name:	Value:
APIVersion	0.156
BuildDate	
Company	
File Version	
Internal Name	
Language	
Original File name	
Product Name	
Product Version	
RegistryKey	
SharedMemoryVersion	

Figure 3.4 Checking the cygwin1.dll Properties

At this point go to the Blink application (/home/user/tinyos-1.x/contrib/handhelds/apps/Blink) and type:

make shimmer2r

You will probably run into some assembler errors which look like this:

```
/tmp/ccFOvuiZ.s: Assembler messages:

/tmp/ccFOvuiZ.s:10: Error: unknown opcode `disablecou'

/tmp/ccFOvuiZ.s:12: Error: unrecognized symbol type ""

...
```

or

```
/tmp/ccrlquXA.s: Assembler messages:

/tmp/ccrlquXA.s:111: Error: `,' required

/tmp/ccrlquXA.s:111: Error: constant value required

/tmp/ccrlquXA.s:111: Error: garbage at end of line
```

These errors occur because you need to patch the msp430 assembler to work with $ signs that nesC adds to the code. That is described in the next step.

If you tried “make clean shimmer” at this time, in addition to the above error, you may also get another assembler error “unknown MCU: msp430x1611”. This problem is also solved in the next step.

Get the patched MSP430 assembler (msp430-as.exe) from the “Windows” folder on the Shimmer Developer’s CD and copy it to the local directory. Then copy it to the following locations using the commands below.

```
cp msp430-as.exe $MSPGCCROOT/bin/msp430-as.exe

cp msp430-as.exe $MSPGCCROOT/msp430/bin/as.exe
```

5.4.4 Testing Build Environment on Windows/Cygwin

5.4.4.1 Compiler Check

To check to see if your build environment is working:

Go to the Blink application folder:

```
cd ~/tinyos-1.x/contrib/handhelds/apps/Blink

make shimmer2r
```

You should see the “ncc” compiler compiling the files with no errors and near the end you should see something like:

```
compiled Blink to build/shimmer2r/main.exe
```

```
3170 bytes in ROM

44 bytes in RAM
```

5.4.4.2 Burning Shimmer Images

To install Shimmer images, you will need to use the correct bootstrap loader.

Symbolically link to the correct BSL using the command below:

```
ln –s /home/user/tinyos-1.x/contrib/handhelds/tools/src/mspgcc-
pybsl/bsl.py  /usr/local/bin/msp430-bsl
```

Connect your programming board (and let it automatically install drivers if this is the first time you are connecting to the board) and go to the Blink application and type:

```
cd /home/user/tinyos-1.x/contrib/handhelds/apps/Blink

make shimmer2r install bsl,/dev/ttyS<n>
```

where <n> will be replaced by the port assigned to the programming board (e.g. if the board took COM5, then it will be ttyS4 because COM ports numbers start at 1 in Windows and 0 in CYGWIN). If the COM port is bigger than 15 you will need to go inside Windows device manager and remove some of the other COM ports (hidden or you might also need to set the Windows environment variable **devmgr_show_nonpresent_devices** to the value "1' since CYGWIN can only use the first 16 COM ports from under Windows!!! You should now be able to successfully compile and install Shimmer images. The blinking LED application should now be running on your Shimmer hardware. Well Done!!

5.5 Building Shimmer Firmware Applications

In this section we explore the process of developing our first application, a program that will transmit the "Hello World!" string wirelessly from Shimmer to a Bluetooth or 802.15.4 transceiver connected to a PC or Workstation. It seems like every programming book ever written begins with the same example, i.e. a program that prints "Hello World!" on the user screen. When dealing with embedded systems, printing of text strings can be more of an endpoint rather than a beginning point and some embedded systems will never have the capability to print text strings because of their specifically dedicated applications [28]. Of course Shimmer doesn't have an LCD to print to so we can print wirelessly to another device that has a screen. Many of the software components already exist to support this application and so we can go straight to writing a "Hello World!" type application immediately.

5.5.1 Application to Send "Hello World!" over Bluetooth

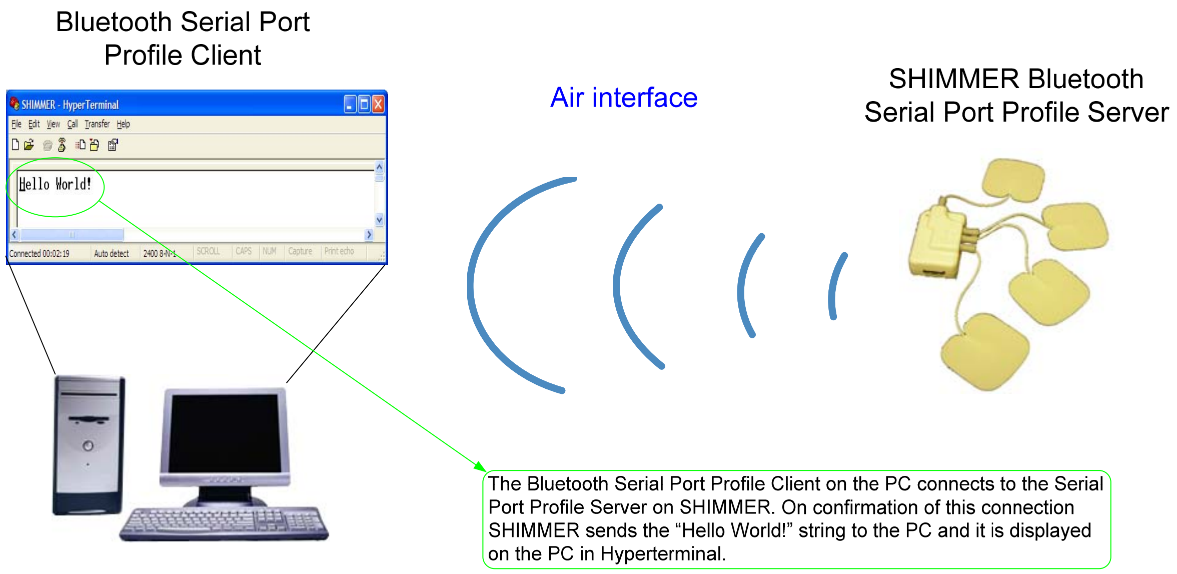
For this application we will send data from Shimmer over the Bluetooth radio to the native HyperTerminal program running on a Windows PC as can be seen in Figure 3.5. You will need a Bluetooth dongle on your PC and some Bluetooth software. For this example we will use the Windows Bluetooth Stack software that comes as standard with Windows XP SP2.

5.5.1.1 Write Application

Create a new folder in ~/tinyos-2.x-contrib/shimmer/apps called "HelloWorld".

Start by writing the HelloWorld Module in your favourite text editor or development suite. This module will implement the application. Copy the text code below into your file and save it as HelloWorldC.nc in the HelloWorld folder (you can also find a similar application in contrib/handhelds/swtest/TestBluetooth):

```
module HelloWorldC {
   uses {
     interface Boot;
     interface Leds;
     interface Init as BluetoothInit;
     interface StdControl as BTStdControl;
     interface Bluetooth;
     interface Timer<TMilli> as Timer;
   }
}

implementation {
   char msgbuf[128];
   boot ready;

   event void Boot.booted(){
     strcpy(msgbuf, "Hello, World!");

     atomic ready = FALSE;

     call BluetoothInit.init();

     call BTStdControl.start();
   }

   task void sendData() {
     call Bluetooth.write(msgbuf, strlen(msgbuf));
   }

   event void Timer.fired() {
     post sendData();
   }

   async event void Bluetooth.connectionMade(uint8_t status) {
     call Leds.led2On();

     if(ready)
       call Timer.startOneShot(1000);
   }

   async event void Bluetooth.connectionClosed(uint8_t reason){
     call Leds.led2Off();
   }

   async event void Bluetooth.dataAvailable(uint8_t data){
   }

   event void Bluetooth.writeDone(){
   }

   async event void Bluetooth.commandModeEnded() {
     atomic ready = TRUE;
   }
}
```


Bluetooth Serial Port Profile Client
SHIMMER - HyperTerminal
Hello World!
Air interface
SHIMMER Bluetooth Serial Port Profile Server

The Bluetooth Serial Port Profile Client on the PC connects to the Serial Port Profile Server on SHIMMER. On confirmation of this connection SHIMMER sends the “Hello World!” string to the PC and it is displayed on the PC in Hyperterminal.

Figure 3.5 Sending “Hello World!” from Shimmer to a PC

Now we need to “wire” our new HelloWorld module to other predefined modules in the system. Every application has a top level configuration file. Copy the text code below into your editor and save it as HelloWorldAppC.nc in the HelloWorld folder:

```
configuration HelloWorldAppC {
}

implementation {
   components MainC, HelloWorldC;
   HelloWorldC -> MainC.Boot;

   components LedsC;
   HelloWorldC.Leds -> LedsC;

   components new TimerMilliC() as Timer;
   HelloWorldC.Timer -> Timer;

   components RovingNetworksC;
   HelloWorldC.BluetoothInit -> RovingNetworksC.Init;
   HelloWorldC.BTStdControl -> RovingNetworksC.StdControl;
   HelloWorldC.Bluetooth    -> RovingNetworksC;
}
```

Lastly we need to point the new application to the core Makefile for TinyOS. Every application within TinyOS has a top level configuration file. Copy the text code below into your editor and save it as a file called Makefile in the HelloWorld folder:

```
COMPONENT=TestBluetoothAppC
include $(MAKERULES)
```

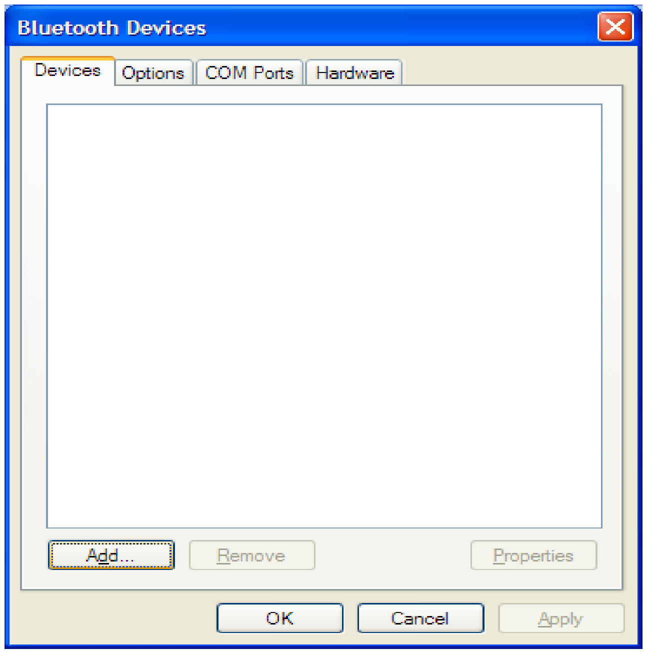
5.5.1.2 Compile and Load Firmware

Depending on whether you are using a Windows or Linux build environment, follow the instructions on Burning Shimmer Images in the “Creating a Shimmer Development Environment” chapter.

5.5.1.3 Test Application

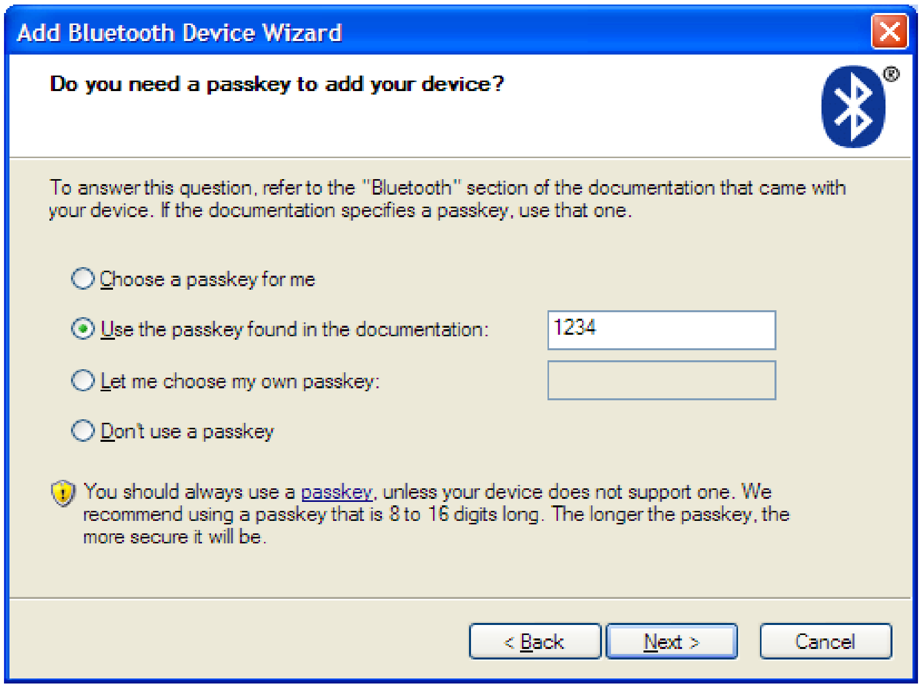
Ensure that the Shimmer is powered up. The Shimmer Bluetooth radio defaults into a discoverable mode so it should be seen by your dongle. Obviously, you can skip Bluetooth pairing if you have already done so in the Quickstart or with another application.

Open up the Windows Bluetooth Stack and search for your Shimmer by clicking on the “Add” button.



This shows all Bluetooth devices within range and your Shimmer device will be found and have a name in the format "RN42-XXXX". The XXXX represents the last 16 bits of the 48-bit Bluetooth Device Address.

Proceed to pairing (sometimes called bonding) with the Shimmer by clicking "Next". We should not have to use any security because authentication and encryption is disabled in Shimmer but there is a bug in the Microsoft Bluetooth Stack which requires a pass key anyway. So put in the Shimmer default passkey of "1234".



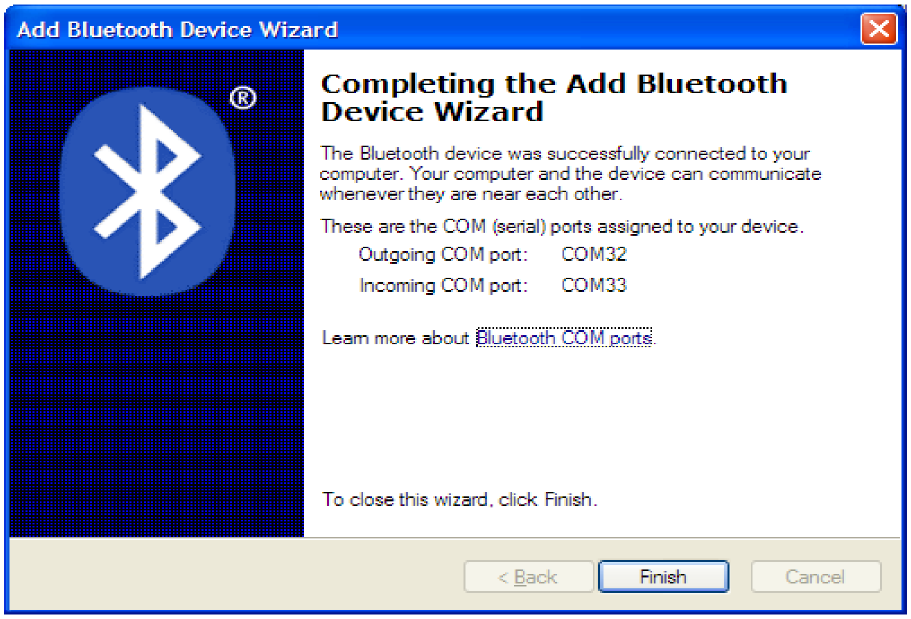
Do you need a passkey to add your device?

To answer this question, refer to the "Bluetooth" section of the documentation that came with your device. If the documentation specifies a passkey, use that one.

- ◯ Choose a passkey for me
- ● Use the passkey found in the documentation: 1234
- ◯ Let me choose my own passkey:
- ◯ Don't use a passkey

You should always use a passkey, unless your device does not support one. We recommend using a passkey that is 8 to 16 digits long. The longer the passkey, the more secure it will be.

The Bluetooth module on Shimmer supports the Bluetooth Serial Port Profile [9] and so Windows installs an Incoming COM port and outgoing COM port to communicate with the Shimmer. The outgoing COM port, COM32 in our case can now be used by any COM port application on the PC to connect to the Shimmer. This is simply a wireless emulation of a physical cable.



Completing the Add Bluetooth Device Wizard

The Bluetooth device was successfully connected to your computer. Your computer and the device can communicate whenever they are near each other.

These are the COM (serial) ports assigned to your device.

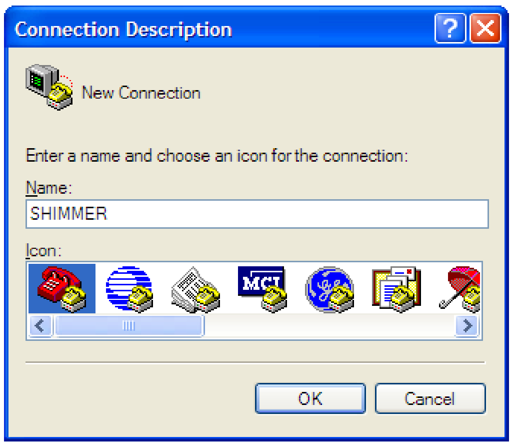
Outgoing COM port:	COM32
Incoming COM port:	COM33

Learn more about Bluetooth COM ports.

To close this wizard, click Finish.

Click "Finish" and go to Start->All Programs->Accessories->Communications and open up HyperTerminal. Make a new connection called Shimmer.

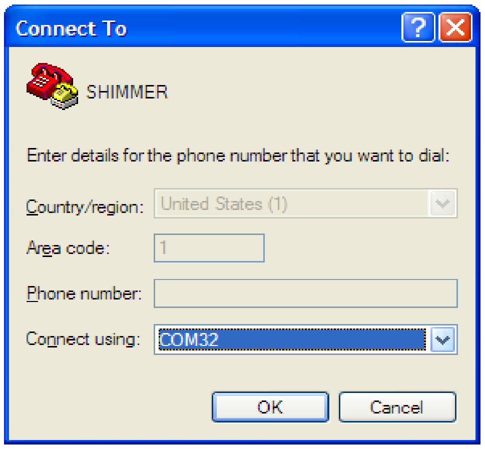
Connect to the outgoing COM port indicated in the previous step. In this case COM32.



Enter a name and choose an icon for the connection:

Name: SHIMMER

Icon:



Enter details for the phone number that you want to dial:

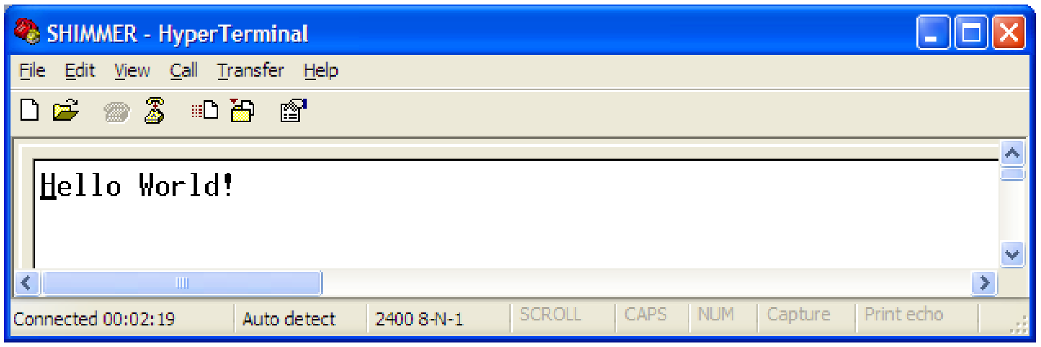
Country/region: United States (1)

Area code: 1

Phone number:

Connect using: COM32

You should see the "Hello World!" string printed out like below. You should also see the green LED on Shimmer coming on when the Bluetooth connections exists. Congratulations. You have now developed and tested your first Shimmer application.

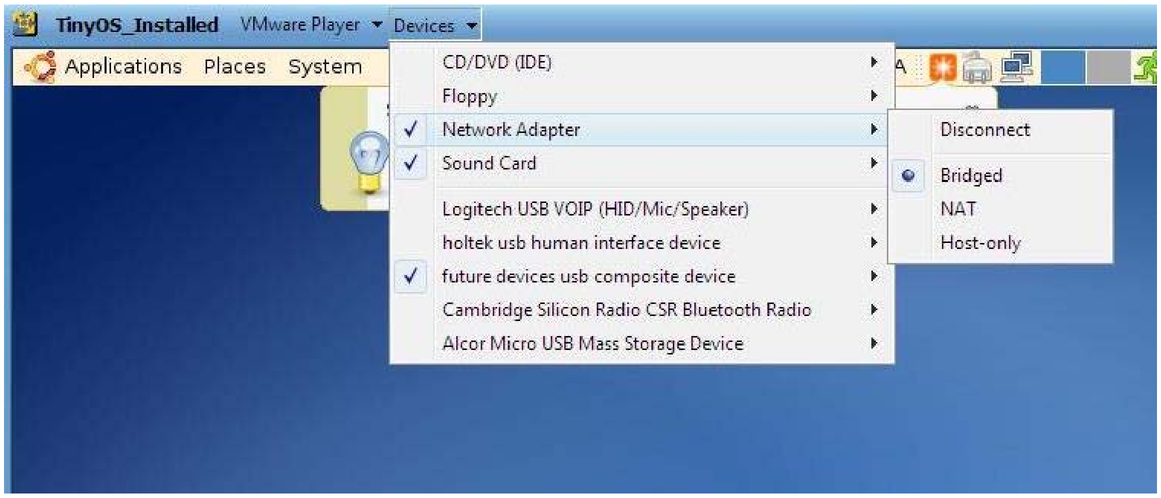


```
Hello World!
```

Connected 00:02:19 | Auto detect | 2400 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

5.5.2 Simple Application using TCP/IP over 802.15.4

A full-featured TCP/IP stack, including both main socket protocols, and a telnet server are available for the 802.15.4 radio [7]. At this time, placing the Shimmer onto the network requires a separate device attached to a Linux machine or Linux virtual machine on the user's LAN, as well as some manipulation of the Linux network stack and kernel. This separate device will serve as both as well as an access point for the 802.15.4 radio, which will service the Shimmer's client connection in the same fashion as an 802.11a/b/g access point. In the examples below we will use span as the target platform. If you are using VMware, please verify the Network adapter setting is in bridged mode:


TinyOS_Installed
VMware Player
Devices
Applications
Places
System
CD/DVD (IDE)
Floppy
✓ Network Adapter
✓ Sound Card
Logitech USB VOIP (HID/Mic/Speaker)
holtek usb human interface device
✓ future devices usb composite device
Cambridge Silicon Radio CSR Bluetooth Radio
Alcor Micro USB Mass Storage Device
Disconnect
● Bridged
NAT
Host-only

First, the Linux kernel module must be built and installed. If you do not maintain your own Linux kernel, you will need to have your distribution's kernel build tree package(s) (usually "kernel-source*") installed. Then:

```
cd tinyos-1.x/contrib/handhelds/apps/AccessPoint/kernel

make

sudo insmod span_ap.ko

echo 1 > /proc/sys/net/ipv4/conf/all/forwarding

echo 1 > /proc/sys/net/ipv4/conf/all/proxy_arp
```

Now build the spare Mote as a NIC/access point. Before doing so, you will need to understand how to address them on your LAN so that you will not interfere with local DNS or DHCP pools. Some users prefer to work behind a NAT router in crowded networks or where they can't acquire address allocation on the publicly-routable network. Working within this scenario, a desktop router might allocate DHCP from the range 192.168.1.1 – 20, leaving from 192.168.1.21 – 254 for static addressing. So, we can set the 802.15.4 NIC/access point to 192.168.1.50, and a Shimmer to

192.168.1.60. Attach the device you'll use for the A.P. (either a span or a Shimmer device in the USB dock) and do this:

```
cd ..  (now in tinyos-1.x/contrib/handhelds/apps/AccessPoint)

make span CC2420_CHANNEL=26 SSID=rti PAN_ID=0xfeed IP=192.168.0.100
install bsl,/dev/ttyUSB0
```

Then we can attach the 802.15.4 radio to the kernel module, and begin listening for associations:

```
cd daemon

make

sudo bash

./zattach -v -n /dev/ttyUSB0
```

You'll see output similar to this:

```
Device name span0
got socket for binding: 5
binding success!

No HW Address found for this interface!

Event: Reset
IP:    192.168.0.100
Addr:  a0:a0:00:00:0d:69:59:50
PanID: 0xfeed
Freq:  2480 (channel 26)
SSID:  rti
setting host ip address
```

Now you can build a Shimmer with your favourite IP communications app. We'll use the boilerplate TelnetServer.

```
cd ~/tinyos-1.x/contrib/handhelds/swtest/TelnetServer

make shimmer2r CC2420_CHANNEL=26 SSID=rti PAN_ID=0xfeed
IP=192.168.0.110 install bsl,/dev/ttyUSB0
```

Once the programming completes, you'll see the Shimmer associate with the access point, and then you can do this:

```
telnet 192.168.0.110
Trying 192.168.0.110...
```

```
Connected to 192.168.0.110 (192.168.0.110)
Escape character is '^]'.
Featherweight command shell
Type '?' for help
Mote>
```

By browsing other applications' use of this interface and the companion ParamView, you'll notice that the command shell is completely extensible and allows the developer to offer the user direct access to any of the application's functionality and memory. As you can imagine, it's also a great debugging tool.

5.5.3 Other Useful Example Applications

5.5.3.1 Sleep Application

A built-in mechanism of TinyOS can take advantage of the MSP430's excellent ability to transition into a low power mode in one instruction, and to wake from deep sleep in 6 uSec. When the task queue is empty, TinyOS places the MCU into this state; on Shimmer, current consumption is then approximately 140uA. Programming a Shimmer with ~/tinyos-1.x/contrib/handhelds/apps/Sleep will enable this behaviour. In TinyOS2 there is naming change: ~/tinyos-2.x/apps/Null

5.5.3.2 Six Axis Transmitter

This is a very useful configurable application which samples 6 channels of ADC data (ADC port 1 to 6) and transmits this data over 802.15.4. The sample period can be adjusted by changing the sample_period variable. The scenario that occurs once this application starts running is depicted in Figure 3.5 below. The application code can be found in the contrib/handhelds/apps/SixAxisTransmitter/ folder of your source tree.

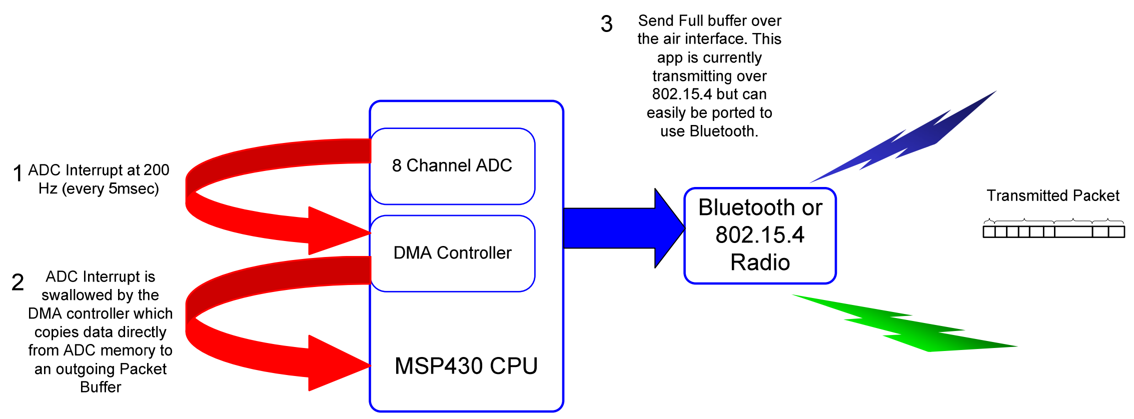

3 Send Full buffer over the air interface. This app is currently transmitting over 802.15.4 but can easily be ported to use Bluetooth.
1 ADC Interrupt at 200 Hz (every 5msec)
8 Channel ADC
DMA Controller
2 ADC Interrupt is swallowed by the DMA controller which copies data directly from ADC memory to an outgoing Packet Buffer
MSP430 CPU
Bluetooth or 802.15.4 Radio
Transmitted Packet

Figure 3.5 Internal Operation of the Six Axis Transmitter Application

5.5.3.3 Power Supply Monitor Application

When running applications that require long periods of active or power-intensive use of Shimmer, it is recommended that the PowerSupplyMonitor be used. This interface to the MCU's Supply Voltage Supervisor allows the programmer to set a threshold voltage for notification of a low power situation and a monitoring interval; an application event is triggered when the threshold is reached so that the application can evade data loss by unexpectedly draining the battery. The interface allows the programmer to automatically notify the user that a power-critical state has been reached and to shutdown the application before data loss becomes a risk.

5.5.3.4 Silicon Identifier DS2411

The DS2411 silicon identifier can be used for a MAC address of the 802.15.4 radio, The interface used with the Chipcon radio is provided as IDChip.nc under contrib/handhelds/tos/interfaces and tinyos-2.x/tos/platforms/shimmer/chips/ds2411.

5.6 nesC Documentation

The nesC compiler includes a facility for automatically generating documentation from nesC source code. The basic structure of the generated documentation is taken from the source code, with one HTML file produced from each nesC source file. You can view a graphical representation of the component relationships within an application [19]. TinyOS source files include metadata within comment blocks that ncc, the nesC compiler, uses to automatically generate html-formatted documentation. To generate the documentation, type:

```
make <platform> docs
```

from the application directory. The resulting documentation is located in docs/nesdoc/<platform>.docs/nesdoc/<platform>/index.html is the main index to all documented applications. There is more information about nesC documentation at:

http://www.tinyos.net/tinyos-1.x/doc/nesdoc

http://www.tinyos.net/tinyos-2.x/doc/nesdoc.

6 Using Labview

6.1 Shimmer LabVIEW Library

National Instruments LabVIEW is a graphical programming environment used by millions of engineers and scientists to develop sophisticated measurement, test, and control systems using intuitive graphical icons and wires that resemble a flowchart. LabView offers unrivaled integration with thousands of hardware devices and provides hundreds of built-in libraries for advanced analysis and data visualization.

http://www.ni.com/labview/

The *Shimmer 2 LabVIEW Instrument Driver Library* (or *Shimmer LabVIEW Library* for short) is a library of LabVIEW VIs designed to assist users of the Shimmer 2 and Shimmer 2r in the development of Shimmer based applications in LabVIEW. The *Shimmer Driver Library* is not intended to be the answer to all host side application requirements, but instead as a set of building blocks for developers.

The library offers a number of different low level *Instrument Driver VIs* for different Shimmer operations such as configuring, triggering and reading data. The library also includes a set of fully *Integrated Shimmer VIs* which are essentially higher level VIs integrating all of the functionality of the lower level *Instrument Driver VIs*. In addition the library includes a number of *Example Application VIs* to assist with Shimmer LabVIEW application development. The *Example Application VIs* may be used in their own right or may be modified by the LabVIEW developer to form the basis for additional applications.

6.2 Getting Started

Release version *Rev 0.4* of the *Shimmer LabVIEW Library* is included in the ShimmerLive distribution package at the location \Non-Live\setup_win\Labview. To download the latest release of the library visit www.shimmer-research.com, select *Technology* from the list of tabs and then select *LabVIEW* from the panel on the left. Here you will find further information including demonstration videos and the link to download the library.

For additional information email labview@shimmer-research.com.

7 Using Octave or Matlab

7.1 Introduction

Matlab is a numerical computing environment, developed by MathWorks, which allows matrix manipulations, plotting of functions and data, implementation of algorithms, etc.

Processing data collected by a Shimmer in Matlab is a common requirement, either in real-time, or post processing.

It is possible to import and process data in Matlab in real time but this is not always a straightforward task due to Matlab's lack of support for any and all Bluetooth devices. Some bluetooth radios (dongles) work with Matlab, and some don't and users who try this in Matlab report that this is very much a hit or miss affair.

Using the 802.15.4 radio (with a Span acting as the PC interface) is another option which should work without any problem (as it will simply be UART/serial communication in Matlab), but this has not yet been rigorously tested.

If your application requires Bluetooth communication there are a couple of work arounds that have been used in the past to overcome Matlab's BT limitation but these can be a bit unwieldy.

Section 7.3 explains one of these work arounds. Section 7.4 explains how to process data saved to a microSD card using one of the example Shimmer Applications.

7.2 GNU Octave

From the GNU Octave website (http://www.gnu.org/software/octave/):

"GNU Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab. It may also be used as a batch-oriented language." [38]

This document provides a quick introduction to data manipulation using Octave/Matlab. There is ample documentation on the web and the installation of Octave is straightforward. You may need to adjust filenames or directories to match your system.

For basic introduction to syntax and functions see:
http://www.math.iitb.ac.in/~pde08/matlab1.pdf

For those that prefer a matlab style GUI—QtOctave,
http://qtoctave.wordpress.com/what-is-qtoctave/

is available as both a distribution or in portable form for Windows which includes octave and can be run from a USB stick without an installation:
http://qtoctave.wordpress.com/2007/10/31/qtoctave-portable/

7.3 Streaming Data to Matlab over Bluetooth

As already mentioned, streaming data to Matlab over Bluetooth can be problematic. One way to overcome this is by handling the Bluetooth communication in a separate host side application, and forwarding the data of interest using a TCP (or UDP) connection.

This server application can be created in any programming language. Adding this functionality to the Shimmer Labview library described in the last chapter is an obvious choice.

The following python code can act as a very simple such solution. This has been tested in python 2.7 and requires the pyserial module to be installed. This application has been written to work with a Shimmer running the SimpleAccel application, but is easily modified to work with BoilerPlate if required (simply by changing the “BUFFER_SIZE” variable).

```
#!/usr/bin/python
import serial, SocketServer, sys

class SocketHandler(SocketServer.BaseRequestHandler):
   def handle(self):
      print "Connection Received From ",
      sys.stdout.write(self.client_address[0])
      sys.stdout.write(':')
      sys.stdout.write(str(self.client_address[1]))
      print

      ser = serial.Serial(sys.argv[1], 115200)
      ser.flushInput();
      BUFFER_SIZE = 120
      try:
         while True:
            self.request.send(ser.read(BUFFER_SIZE))
      except:
         ser.close()

if __name__ == "__main__":
   if len(sys.argv) < 3:
      print "Usage: simpleAccelServer.py <serial_port> <tcp_port>"
      print "<serial_port> is the Bluetooth serial port of the Shimmer to connect to"
      print "<tcp_port> is the TCP port to listen on"
      print "example:"
      print "  simpleAccel.py Com5 10001"
      print "or"
      print "  simpleAccel.py /dev/rfcomm0 10001"
   else:
      server = SocketServer.TCPServer(("", int(sys.argv[2])), SocketHandler)
      try:
         server.serve_forever()
      except KeyboardInterrupt:
         print
         print "All done"
```

Save this to a file, for example to simpleServer.py, and run it from the command line. E.g. (in Windows):

```
python simpleServer.py COM5 10001
```

where “COM5” is the serial port of the Shimmer you wish to connect to, and “10001” is the TCP port you wish to use.

A very simple Matlab script, which will connect to simpleServer, read 100 packets of data from the Shimmer, plot them, and exit is as follows. This assumes Matlab is running on the same machine as the python server. If this is not the case change the “HOST” variable to the IP address of the server machine. Also if a port other than “10001” was chosen when starting simpleServer.py, change the “PORT” variable appropriately.

```
BUFFER_SIZE = 120;
DATA_TO_DISPLAY = 3;
HOST = 'localhost';
PORT = 10001;

sock = tcpip(HOST, PORT);

set(sock, 'InputBufferSize', BUFFER_SIZE);

fopen(sock)

DataAll = [];

i = 0;
figure;
while i<100
    DataReceived = fread(sock);
    if(length(DataReceived) ~= BUFFER_SIZE)
         error('Incorrect sized packet');
    end
    Data=reshape(DataReceived,6,20)';

    AccelX=((Data(:,2)*256)+Data(:,1));
    AccelY=((Data(:,4)*256)+Data(:,3));
    AccelZ=((Data(:,6)*256)+Data(:,5));

    AccelAll=[AccelX,AccelY,AccelZ];

   DataAll=[DataAll;[AccelAll]];

   if(length(DataAll)>DATA_TO_DISPLAY*BUFFER_SIZE) % Cuts the Data matrix so can
be easily displayed
      DataAll=DataAll((length(DataAll)-DATA_TO_DISPLAY*BUFFER_SIZE-
1):length(DataAll),:);
   end
   figure(gcf);
   plot(DataAll(:,1:3)); % The raw data from the shimmer
   i = i + 1
end

% Disconnect and clean up the server connection
fclose(sock)
delete(sock)
clear sock
```

7.4 JustFATLogging

JustFATLogging is an example program for logging data with Shimmer. For storage and power economy, data is stored as a unsigned 16-bit binary values in x,y,z tuples and is easy to manipulate and translate into other data formats using Octave or Matlab.

7.4.1 Opening the binary data file

It is recommended (but not required) to copy your data files from the Shimmer to a directory on your computer. As JustFATLogging writes multiple files per session, you may want to combine data files-- most operating systems have a method to do this. For Windows users, open an MS-DOS Command window and in your data directory use the command: **copy /b [filelist] [outputfile]**

For example: **copy /b *. accel.bin** (hint: don't forget /b)

Next start up Octave. First we will assign a macro so we don't have to type as much and we get rid of the annoying terminal promts:

fid = fopen('accel.bin');

more off

to extract x-axis data to a variable called "accelx" use the command:

accelx = fread(fid, 'uint16', 4);

In this example the final parameter '4' skips the y and z data as they are 2 bytes each. For more information on fread, see:
http://www.mathworks.com/access/helpdesk/help/techdoc/ref/fread.html

At this point, Octave is looking at the end of the datafile, so you need to rewind back to the beginning of the file if you want to access it again:

frewind(fid);

However, we are actually going to load accely, which starts after a 2 byte offset, so we can use the fseek command:

fseek (fid, 2, 'bof');

accely = fread(fid, 'uint16', 4);

fseek (fid, 4, 'bof');

accelz = fread(fid, 'uint16', 4); (looks like we should have a macro for the fread command)

Alternatively you can load a matrix containing the data in separate rows. Again, make sure you use frewind.

The generic command is: **matrix_var = fread(fid, [n,Inf], 'uint16');** where n = number of channels sampled. So for 3 channel accelerometer logging use:

Accelxyz = fread(fid, [3,Inf], 'uint16');

For 6 channel logging replace the 3 with 6.

You can verify the data by outputting the beginning of the matrix:

Accelxyz ([1:3],[1:10]);

Or by typing **whos** and inspecting the size of the variable. In QtOctave, the variables list pane also shows this list—make sure you expand the appropriate variablespace.

7.4.2 Plotting data

To plot the data, Octave uses GNUPlot. The hold command is used to allow multiple plots on the same axis. Using a comma allows multiple commands to be entered at the command line:

hold, plot(accelx, "1"), plot(accely, "2"), plot(accelz, "3"), hold off;

or

subplot(2,2,1); plot(accelx, "1"); title 'Accel X';

subplot(2,2,2); plot(accely, "1"); title 'Accel Y';

subplot(2,2,2); plot(accelz, "1"); title 'Accel Z';

If you have loaded up a matrix, you can plot by row (or column):

subplot(2,2,1); plot(Accelxyz(1,:), "1"); title 'Accel X';

subplot(2,2,2); plot(Accelxyz(2,:), "2"); title 'Accel Y';

subplot(2,2,3); plot(Accelxyz(3,:), "3"); title 'Accel Z';

7.4.3 Manipulating data

Here is a trivial example:

accelx_nulloffset = accelx - 2048;

You can transpose the matrix (useful before outputting a CSV) with the **' operator:**

AccelxyzT = Accelxyz'

A more extensive example is calculating tilt based on the measured ADC value. A required value is the sensitivity of the accelerometer. On Shimmer, accelerometers typically run with a 3.0V ADC range and g-setting of 1.5. Looking at the datasheet for the accelerometer and scaling for 3.0V the ideal sensitivity is .727V per g. It is also required to know the midpoint of the accelerometer, this should be 1.5V. Both of those values will vary in practice so calibration is advised.

accelx_volts = accelx * 3 / 4095;

calc_accelx_volts = (accelx_volts – 1.5) / .727;

angle_accelx = asind(calc_accelx_volts);

```
angle_accelxReal = real(angle_accelx);

subplot(2,1,1); plot(accelx);

subplot(2,1,2); plot(angle_accelxReal);
```

7.4.4 Exporting data

Octave/Matlab use the commands ***csvwrite*** or ***dlmwrite*** to output text/ascii files. You can use ***fwrite*** for binary output.

Examples:

```
    csvwrite('accelx.csv', accelx);

    dlmwrite('accelx.dat', accelx, '\d');


    fid2 = fopen('accelx.bin', 'w');

    fwrite(fid2, accelx);
```

8 Calibration

The compact size, long operating life, and value of Shimmer is a direct result of recent developments in component technology. Depending on your application requirements and the sensor input source, calibration may be required or greatly improve sensor accuracy. A brief overview of static sources error and calibration procedure for various Shimmer peripherals is listed below. For additional information on the Shimmer ADC or comprehensive analysis please consult vendor websites and datasheets for the parts presented in Section 2:

8.1 Accelerometer, Gyroscope and Magnetometer Calibration

8.1.1 Common Sources of Error

The accelerometer, gyroscope and magnetometer have three common sources of error; Offset, Sensitivity and Axis Alignment, each of which can be corrected for using the appropriate calibration method.

Offset error is a shift in the zero output from the expected zero output. This source of error can be calibrated during or after data collection by measuring the offset value.

Sensitivity error is a variation in the signal conditioning within the sensor.

Axis Alignment error occurs based on placement variation of the Shimmer within the enclosure.

8.1.2 Shimmer 9DOF Calibration Application

The *Shimmer 9DOF Calibration Application* is an application to allow users to calculate the accelerometer Offset, Sensitivity and Axis Alignment and thus allow for auto-mated calibration of the accelerometer signals. The documentation which provided with the application also provides information on how to calibrate the gyroscope and magnetometer sensors. A future release of the application will provide for auto-mated calibration of the gyroscope and magnetometer in a manner similar to that currently implemented for the accelerometer.

Release version *Rev 0.1* of the *Shimmer 9DOF Calibration Application* is included in the ShimmerLive distribution package at the location \Non-Live\setup_win\Labview. To download the latest release of the library visit www.shimmer-research.com, select *Technology* from the list of tabs and then select *9DOF Calibration* from the panel on the left. Here you will find further information including a demonstration video and the link to download the application.

8.1.3 Voltage reference Skew

The Accelerometer or any other DC-coupled analog sensor used with Shimmer suffers from 4 sources of error as a result of the MEMs/IC manufacturing process: voltage reference skew, offset, sensitivity, and axis alignment [30].

Voltage reference skew occurs when a sensor is powered from a different supply than the analog voltage reference used in the Shimmer ADC. Any difference will result in a mismatch between the sensor full-scale output, and the Shimmer ADC's 12bit digital full-scale output. The simplest mitigation is programming your Shimmer to use the AVCC setting in the ADC and using a sensor powered from the same ADC level (3.0V on Shimmer). This scheme is called a ratiometric supply, as any change in the Shimmer ADC reference will ratiometrically effect the sensor output voltage according to the gain of the front-end. With the Shimmer Accelerometer implementation, the ratio is 1:1-- a trivial calculation!

8.2 ECG Sources of Error and Calibration

ECG signal error analysis is a lengthy subject. Shimmer users are encouraged to familiarize themselves with the impact of skin-electrode impedance (the front end circuit on the Shimmer-ECG is extremely high impedance), common signal artifacts, and noise sources.

Shimmer offers some advantages for ambulatory ECG monitoring. The compact size allows short lead lengths to reduce noise. Also the Shimmer ECG circuit uses an AC-coupled topology to improve signal quality when the subject is in motion. Periodic shifts in the zero-signal readout during periods of motion or lead-manipulation are normal and easily eliminated in post-processing.

If the user wishes to calibrate the ECG front end, a patient simulator device [35] can be used to calculate gain, offset voltage, and frequency response. Conventional laboratory signal generators may be used, but they must be attenuated to 1-10mV Amplitude for ECG., and .1-1mV for EMG.

9 Legacy Support

9.1 Revision Changes

Efforts have been made to provide a straightforward migration of applications, accessories, and daughter cards across revisions. Hardware is forward compatible and applications are easily recompiled for Shimmer2R vs. Shimmer2 vs. Shimmer1. The table below lists feature changes, and suggested migration/support strategy

Platform Feature	**Shimmer2R**	**Shimmer 2**	**Shimmer 1**	**Mitigation**
Orange LED	Not present	Not present	Present	• Use previously compiled applications with proper platform suffix • comment out or remap orange LED calls in application code
Pin 20, internal expansion Connector	PV_REG 2.75-4.7V raw board power (battery or USB	PV_REG 2.75-4.7V raw board power (battery or USB	Regulated 1.8V	• No known expansion boards use the 1.8V pin • Future expansion boards that use PV_REG will be designated "Not for use with Shimmer1" • Some boards may include a jumper to select Shimmer1 or Shimmer2 mode per documentation
Pin 11, internal expansion Connector	GPIO_INTERNAL1	GPIO_INTERNAL1	PV_CHG	• Shimmer2 only expansion boards have access to an additional GPIO pin with a 100k pull-up

				• No commercial expansion boards us PV_CHG on the internal expansion connector • Future expansion boards that use PV_CHG will be designated "For use with Shimmer1 only" • Applications that use GPIO_Internal1 (ROSC in hardware.h) will need to be modified for use with Shimmer1
CPU pin-out changes	Yes	Yes	Yes	• Use previously compiled applications with proper suffix • Compile for target platform
Tilt/vibration sensor	Yes	Yes	No	• Shimmer1 applications can't use the tilt/vibration sensor • Legacy support per Application on a case-by-case basis
Bluetooth power switch	Yes	Yes	No	• The Bluetooth module on Shimmer1 is always powered. Bluetooth while in reset but switched on consumes over 100uA • Shimmer2R uses a more efficient Bluetooth module, but the power switch is still the most efficient control.
MicroSD power switch	Yes	Yes	No	• The MicroSD card is always powered on a Shimmer1 when inserted. An idle

				MicroSD card consumes over 100uA • Shimmer1 cannot support SD data bypass • MicroSD cards that are inserted but unused will add to platform power consumption and should be removed
Dock Signal and MicroSD Data bypass	Yes	Yes	No	• Shimmer1 applications will not have access to a "Dock" signal • Shimmer2 applications must consider the behavior of the Dock signal and MicroSD card data bypass-- see JustFAT application for a programming example • Shimmer2 developers can set Dock as an output and drive high to emulate Shimmer1 MicroSD functionality (obviously the MicroSD data bypass is disabled) • The following Shimmer2 signals are not available when docked: ○ ADC7 ○ ADC0 ○ SER0_RTS
Power-on control	Yes	Yes	No	• Shimmer1 can't be switched off • Shimmer1 does not have a hardware low-battery protection-- Developers must provide SW battery monitoring. Find examples in the

				BioMOBIUS application directories • Shimmer2 can only be programmed or charged when switched on
Bluetooth/SD concurrency bug	No	No	Yes	• Shimmer2 applications that use both Bluetooth and MicroSD or an expansion board with SPIO may not work on Shimmer1. Upgrade to Shimmer2 is recommended • Shimmer1 has a hardware concurrency bug that prevents use of Bluetooth and an SPI0 device without a hardware fix (removal of R18) • Shimmer1 devices that have been fixed no longer route SPI0 clock to the internal expansion connector • If the application can handle the performance penalty of holding the Bluetooth module in reset between SPI0 bus transactions there is no need to perform the HW fix
Accelerometer axis orientation	N/A	N/A	N/A	• The polarity of the Z-axis and Y-axis is flipped on Shimmer2 when compared with Shimmer1 devices
Modular 802.15.4 radio	Yes	No	No	• No functional difference. • Shimmer2R FCC, ETSI,

				Industry Canada Compliant.
Accelerometer range	No	Yes	Yes	• Shimmer2R does not use Accel_SEL_1. Invalid Accelerometer settings result in a compiler warning and selection of the lowest range setting.
Battery and power measurement	Yes	No	No	• Shimmer and Shimmer2 do not support scaled battery voltage measurement, regulator input measurement, or derived current calculations. • ADC6 and ADC7 will not be available on Shimmer2R when SEL_PMUX is enabled.
Bluetooth power forced on while docked	No	Yes	N/A	• On Shimmer2, the Bluetooth power switch is enabled while docked. This "feature" was eliminated from Shimmer2R • Shimmer does not have a Bluetooth power switch
SFD signal routed to timer capture	Yes	No		• Shimmer2R routes the SFD signal to a timer capture pin for improved 802.15.4 physical-layer compliance in communication stacks

10 References

Some references omitted intentionally

[3] Intel Corp, Intel Mote, Sensor Nets/ RFID, Available: http://www.intel.com/research/exploratory/motes.htm, (Accessed: 27th September 2006).

[7] A. Christian, J. Hicks, B. Avery, B. Kuris, D. Denning; S. Ayer, J. Ankcorn, HPL-2005-114 Fingertips of the Network: Featherweight Communicators and Sensors, HP Labs Tech Report, June 2005, Available at http://www.hpl.hp.com/techreports/2005/HPL-2005-114.html

[9] Bluetooth Special Interest Group, Serial port Profile Specification, Vol. 7 Part B of the Bluetooth Specification Version 2, November, 2004.

[10] C. Hodgdon, Adaptive frequency hopping for reduced interference between Bluetooth and wireless LAN, Ericsson Technology Licensing, May 2003.

[12] Texas Instruments, MSP430F1611 Microcontroller Datasheets, Available at http://focus.ti.com/docs/prod/folders/print/msp430f1611.html, (Accessed: 19th April 2007: Last Update: 21st August 2006).

[13] Freescale Semiconductor, MMA7260Q 3-Axis Accelerometer, Available at http://www.freescale.com

[14] Texas Instruments, Chipcon CC2420 RF Transceiver, Available at http://www.chipcon.com/files/CC2420_Brochure.pdf, (Accessed: 19th April 2007).

[15] Shimmer Research SR7 module datasheet. Available on request. Email info@shimmer-research.com

[16] Roving Networks, Bluetooth Module RN-42, Available at http://www.rovingnetworks.com

[17] Roving Networks, Bluetooth Module RN-46 Command Set, Available at http://www.rovingnetworks.com

[18] HIROSE Internal Expansion Connector DF12 Series, Available at http://www.hirose-connectors.com/connectors, (Accessed: 19th April 2007).

[19] Official TinyOS Website, Available at http://www.tinyos.net/.

[20] A. Christian, The User's Manual for HP TinyOS Software Development, Available: http://tinyos.cvs.sourceforge.net/*checkout*/tinyos/tinyos-1.x/contrib/handhelds/docs/usermanual.pdf?revision=1.1

[21] Best Current practice for TinyOS Coding Standards. Available at http://www.tinyos.net/tinyos-2.x/doc/html/tep3.html.

[22] B. W. Kernighan and D. M. Ritchie, The C Programming Language, Second Edition, 1988.

[23] D. Gay, P. Levis, D. Culler, Eric. Brewer, nesC 1.1 Language Reference Manual.

[24] W. Wolf, High-Performance Embedded Computing: Architectures, Applications, and Methodologies, p19-20, 2007.

[25] Official (CVS) Concurrent Versioning System Website, Available at http://www.nongnu.org/cvs/.

[26] P. Levis, TinOS Programming, Available at http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf, June 2006.

[27] SourceForge TinyOS Project Source Code Website, Available at http://sourceforge.net/projects/tinyos/ .

[28] M. Barr, Programming Embedded Systems in C and C++, First Edition, p13-18, January 1999.

[29] T. Hubler, Worry-Free Wireless Networks, Available at http://www.us.sbt.siemens.com/bau/products/Wireless/HPACEprint.pdf, 2005.

[30] A. Burns, F. Tuffy, et al. Intel Corporation Digital Health Group, Shimmer Workshop Manual, 2008-09.

[31] Technology Research for Independent Living, BioMOBIUS White Paper. Available at http://www.biomobius.trilcentre.org, (Accessed: 26th October, 2009).

[32] Technology Research for Independent Living, BioMOBIUS Shimmer Tutorial. Available at http://www.biomobius.trilcentre.org, (Accessed: 26th October, 2009).

[33] F. Ferraris, U. Grimaldi, and M. Parvis, Procedure for effortless in-field calibration of three-axis rate gyros and accelerometers. Sens. Mater. 1995; 7: 311-30

[34] InvenSense, IDG-00 Integrated Dual-axis Gyro Datasheet, Available at http://www.invensense.com/shared/pdf/DS_IDG300.pdf, (Accessed: 26th October, 2009).

[35] Medi Cal Instruments, Inc., http://www.ecgsimulators.com

[36] Shimmer Research Kinematics User Guide. Available at www.shimmer-research.com

[37] Shimmer Research MicroSD Guide. Available at www.shimmer-research.com

[38] GNU Octave. Available at http://www.gnu.org/software/octave/

Notes

Notes

Notes