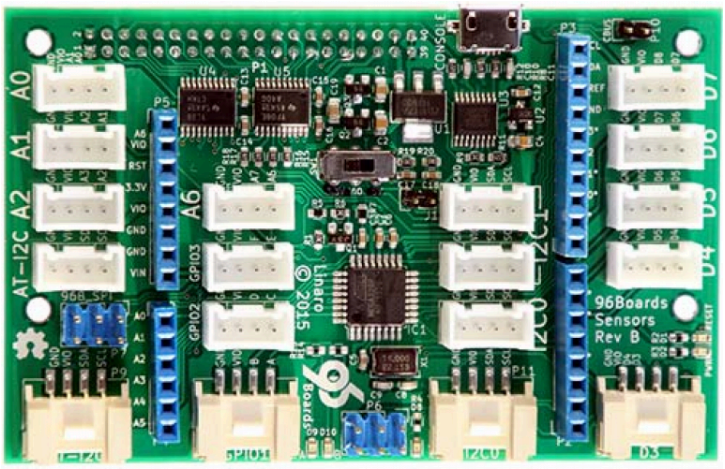
96Boards Sensors Adapter Getting Started Guide

For Rev B of the Sensors adapter board

96Boards Sensors Adapter Getting Started Guide

For Rev B of the Sensors adapter board

General Description



This is a getting started guide for using the 96Boards Sensors adapter. This guide will step you through the process of hooking up the Sensors mezzanine board to a 96Boards compliant baseboard and starting to hook it up to sensor devices.

Installation

Disconnect power from all boards before installing the adapter. Attach the Sensors large 40-pin header (J2) to the LS expansion connector on a 96Boards base board.

WARNING: Do not insert backwards. Doing so will connect the 96Boards +8-18V power supply rail directly to the UART IO pins and will destroy your Sensors adapter. It may also damage your base board.

HDMI and USB devices can be connected as normal to the baseboard. The “Console” Micro-USB connector on the Sensors board provides a USB-UART connection to the serial console. This is exactly the same circuit as found on the 96B-UART adapter, and can be used to control power and reset signal lines on the baseboard.

Connect power to the 96Boards baseboard as normal.

Software Setup

All of the following instructions assume Debian is installed on the baseboard. You will need to adapt these instructions if you are using a different OS, such as Android.

Install Arduino tools

Open a shell on the baseboard and Install the Arduino tools. You can use the Console USB port to get a shell prompt from the console UART.

For command line only Arduino tools:

$ sudo apt-get install arduino-mk

For full Arduino IDE to use from the desktop.

$ sudo apt-get install arduino

Both i2c tools and Git are also useful, install them too.

$ sudo apt-get install i2c-tools git

Install libsoc

libsoc is a C shared library that has been modified to use pin mapping for 96Boards, it is extensive and very useful accessing GPIO, I2C and SPI functions on Linux. As of this writing it only support C and C++, though Python and Java support is being added by Linaro.

$ sudo apt-get update
$ sudo apt-get install build-essential autoconf automake libtool libsoc-dev

libsoc is a cross platform library and will work on any of the 96Boards, and now handles the mapping between LS connector pins and Linux GPIO pin numbers. libsoc is a bit complex and if you want a more simplistic, easier set of calls to access to the 96Boards GPIO we recommend installing and using the 96BoardsGPIO library in conjunction with the libsoc library. The 96BoardsGPIO shared library has been rewritten to depend on libsoc but presents simpler calls similar to the Arduino digitalRead() and digitalWrite() type of calls.

Install 96BoardsGPIO

To build and install 96BoardsGPIO from source, use the following commands:

$ git clone https://github.com/96Boards/96BoardsGPIO.git
$ cd 96BoardsGPIO
./autogen.sh
$ ./configure
$ make
$ sudo make install

Using 96Boards GPIO

Grove connectors GPIO1, GPIO2 and GPIO3 provide level shifted access to GPIOs A, B, C, D, E and F. You can connect digital gpio Grove modules to these connectors and directly access them from Linux. GPIOs can be manipulated from the shell by using the sysfs interface.

When you installed libsoc library it should have exported 12 shell variables GPIO_A - GPIO_L into your shell environment. You should use these in all shell scripts so that they work seamlessly across any 96Boards device. If you don't have GPIO_A - GPIO_L in your shell environment you can go to https://github.com/96boards/96boards-tools where a version of the script is packaged.

GPIO read example

This example assumes you have the proper shell variables available. Connect Grove button module to GPIO1. To read a gpio you need to export the GPIO number, set the pin to input, and read the GPIO.

```
$ cd /sys/class/gpio
$ echo $GPIO_A | sudo tee export    # GPIO_A is 488 on a HiKey and 36 on a DragonBoard 410c
$ cd gpio$GPIO_A
$ cat value  # Button released
0
$ cat value  # Button pressed
1
```

GPIO write example

Connect Grove LED module to GPIO3. To write to a GPIO you need to export the GPIO, set the pin to output, and write the value:

```
$ cd /sys/class/gpio
$ echo $GPIO_C | sudo tee export  # GPIO_C is 490 on HiKey
$ cd gpio$GPIO_C
$ echo out | sudo tee direction
$ echo 1 | sudo tee value   # Turn LED on
$ echo 0 | sudo tee value   # Turn LED off
```

Using 96B I2C

The 4 Grove connectors labelled I2C0 and I2C1 are wired to the I2C0 and I2C1 busses on the LS expansion connector. You can connect I2C devices directly to these Grove connectors and access them directly from Linux

When using long Grove cables or several I2C modules, the I2C bus may suffer from crosstalk between the clock and data lines. If you experience difficulty communicating with a module then try spliting the conductors of the Grove

ribbon cable. Physically separating the clock and data lines reduces the crosstalk problem.

I2C Example with i2cset

Hook the RGB LED modules to connector I2C0. Use the following commands to initialize the backlight controller. Remember to compensate for the missing pull up resistors by pulling apart the CLK and DATA signals from the ribbon cable.

```
$ i2cset -y 0 0x62 0 0   # Write 0 to register 0 on RGB backlight controller)
$ i2cset -y 0 0x62 1 0   # Write 0 to register 1 on RGB backlight controller)
$ i2cset -y 0 0x62 8 0xaa   # Write 0xAA to register 0x8 on RGB backlight controller)
$ i2cset -y 0 0x62 2 0xff   # Turn on Blue component RGB backlight controller)
$ i2cset -y 0 0x62 3 0xff   # Turn on Green component RGB backlight controller)
$ i2cset -y 0 0x62 4 0xff   # Turn on Red component RGB backlight controller)
```

I2C Example with

Using the Arduino compatible processor

The four blue 0.1” 1xN header connectors, and 11 Grove connectors D3-D7, A0-A2, A6 and AT-I2C are connected to the on-board Atmel ATMEGA328P microcontroller. The microcontroller is compatible with the Arduino IDE, and is 100% compatible with all Arduino UNO sketches.

The Atmel is pre-loaded with the Arduino UNO bootloader, and can be programmed using UART0 on the LS expansion connector. The microcontroller can be programmed using the avrdude tool and the Arduino IDE.

Programming the ATMEGA328P using the Arduino IDE

This section assumes you have completed the *Software Setup* section earlier in this document.

The arduino IDE will not use the 96Boards UART0 for programming out of the box. It needs to be told explicitly[1] which UART to use. After installing the Arduino IDE, edit /usr/bin/arduino and change the java line to specify the UART name.

First let’s make sure you have the correct tty ports installed. 96Boards is a bit different unlike other development boards you can use the same command no matter which board you have, though that requires a bit of behind the scenes udev work in this case.

[1] http://angryelectron.com/rxtx-on-raspbian/

So first look at the results of the below command:

$ ls -l /dev/tty96B*

If you see:

/dev/tty96B0 /dev/tty96B1

$

you are good, jump down to “Locate the following line in /usr/bin/arduino:” If you don't see those 2 files you need to install a set of udev rules on your 96Board system. Please go to https://github.com/96boards/96boards-tools You will the 70-96boards-common.rules file, there is a Debian package you can install if you are running a Debian build, or you can simply download the file itself and copy it into /etc/udev/rules.d/ directory. You will need to be root to do so:

$ sudo mv 70-96boards-common.rules /etc/udev/rules.d/.

Than reboot and the standard 96Boards /dev/tty96B? ports will be in place.

Locate the following line in /usr/bin/arduino:

java -Dswing.defaultlaf=com.sun.java.swing.plaf.gtk.GTKLookAndFeel processing.app.Base "$@"

Change it to this:

java -Dswing.defaultlaf=com.sun.java.swing.plaf.gtk.GTKLookAndFeel -Dgnu.io.rxtx.SerialPorts="/dev/tty96B0" processing.app.Base "$@"

Then the Arduino IDE should be able to program the Atmel when Arduino UNO is selected. If you are using the Arduino IDE you are ready to go, the IDE will upload sketches into the ATMEGA328P and release the ATMEGA from reset. If you are programming the ATMEGA328P from the command line you may need the info below in the “**Releasing ATMEGA from reset**” and the “**Programming the ATMEGA from the command line**” sections.

Releasing ATMEGA from reset

The ATMEGA reset signal is wired to the UART0 RTS line. Avrdude uses the RTS signal to reset the ATMEGA at various points in the programming cycle. However, it often leaves the ATMEGA in reset after programming is complete. It will automatically if a program (ie. terminal emulator) opens the serial device. It can also be manually be released from reset by using the following stty command:

$ stty -F /dev/tty96B0 -hupcl # Release ATMEGA from reset

```
$ stty -F /dev/tty96B0 hupcl  # Place ATMEGA into reset
```

Programming the ATMEGA from the command line

Often the easiest way to load an Arduino sketch into the ATMEGA is to use the command line. The following example will load and run the example Blink sketch using only the command line:

```
$ mkdir -p sketchbook/Blink
$ cd sketchbook/Blink
$ cp /usr/share/arduino/examples/01.Basics/Blink/Blink.ino .
$ cat > Makefile << EOF
BOARD_TAG = uno
MONITOR_PORT = /dev/tty96B0
include /usr/share/arduino/Arduino.mk
> EOF
$ make
$ make upload
$ stty -F /dev/tty96B0 -hupcl   # Release Arduino from reset
```

Alternately, you can do “make monitor” to connect screen to the serial port which will also release the ATMEGA from reset. The serial connection can be used as an IO channel between Linux on the baseboard and sketches running on the ATMEGA.

Pretty much any Arduino UNO sketch should work on the Sensors board. You should have everything you now need to make use of the Sensors board.

Grove Connector Pinouts

For pinouts of Grove connectors both ARM and ATMEGA connections, please refer to the table below. You can also look at the design files linked below hosted in GitHub. There is a pdf copy of the schematic available in the git repository.

Grove Connector	CPU	Pins (3 & 4 are VIO and Ground respectively unless otherwise noted)
D3	ATMEGA	1. PD3 / INT1 2. PD4 3. VIO 4. Ground
D4	ATMEGA	1. PD4 2. PD5
D5	ATMEGA	1. PD5 2. PD6

D6	ATMEGA	1. PD6 2. PD7
D7	ATMEGA	1. PD7 2. PB0
A0	ATMEGA	1. PC0 2. PC1
A1	ATMEGA	1. PC1 2. PC2
A2	ATMEGA	1. PC2 2. PC3
A6	ATMEGA	1. ADC6 2. ADC7
AT-I2C (x2)	ATMEGA	1. PC5 / SCL 2. PC4 / SDA
GPIO1	ARM	1. GPIO-A 2. GPIO-B
GPIO2	ARM	1. GPIO-C 2. GPIO-D
GPIO3	ARM	1. GPIO-E 2. GPIO-F
I2C0 (x2) P11,P12	ARM	1. I2C0_SCL 2. I2C0_SDA
I2C1 (x2) P13,P14	ARM	1. I2C1_SCL 2. I2C1_SDA

Design files

The UART adapter board is an entirely Open Hardware, designed using KiCad, and with all of the design files provided under a BSD license. The files can be found on GitHub:

https://github.com/96boards/96boards-sensors

$ git clone https://github.com/96boards/96boards-sensors.git

$ git checkout rev-b

Using the Seeed Studio Sketchbook Starter Kit

The Seeed Studio's Sketchbook Starter Kit for Arduino works with the 96Boards Sensor board. You can get the sketch demos from: github.com/Seeed-Studio/Sketchbook_Starter_Kit_for_Arduino You can clone the archive via git or download the zip file. The titles of each sketch are listed below and the required code changed to make it work with the 96Boards Sensor Board.

AdaFruit 2.8" TFT Touch Shield for Arduino w/Capacitive Touch Display

PRODUCT ID: 1947

Plug the display into the Arduino connectors. See: https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2 for instructions and software download. Known to work as a display, testing still going on for touch screen.

github software links:

https://github.com/adafruit/Adafruit_ILI9341

https://github.com/adafruit/Adafruit-GFX-Library