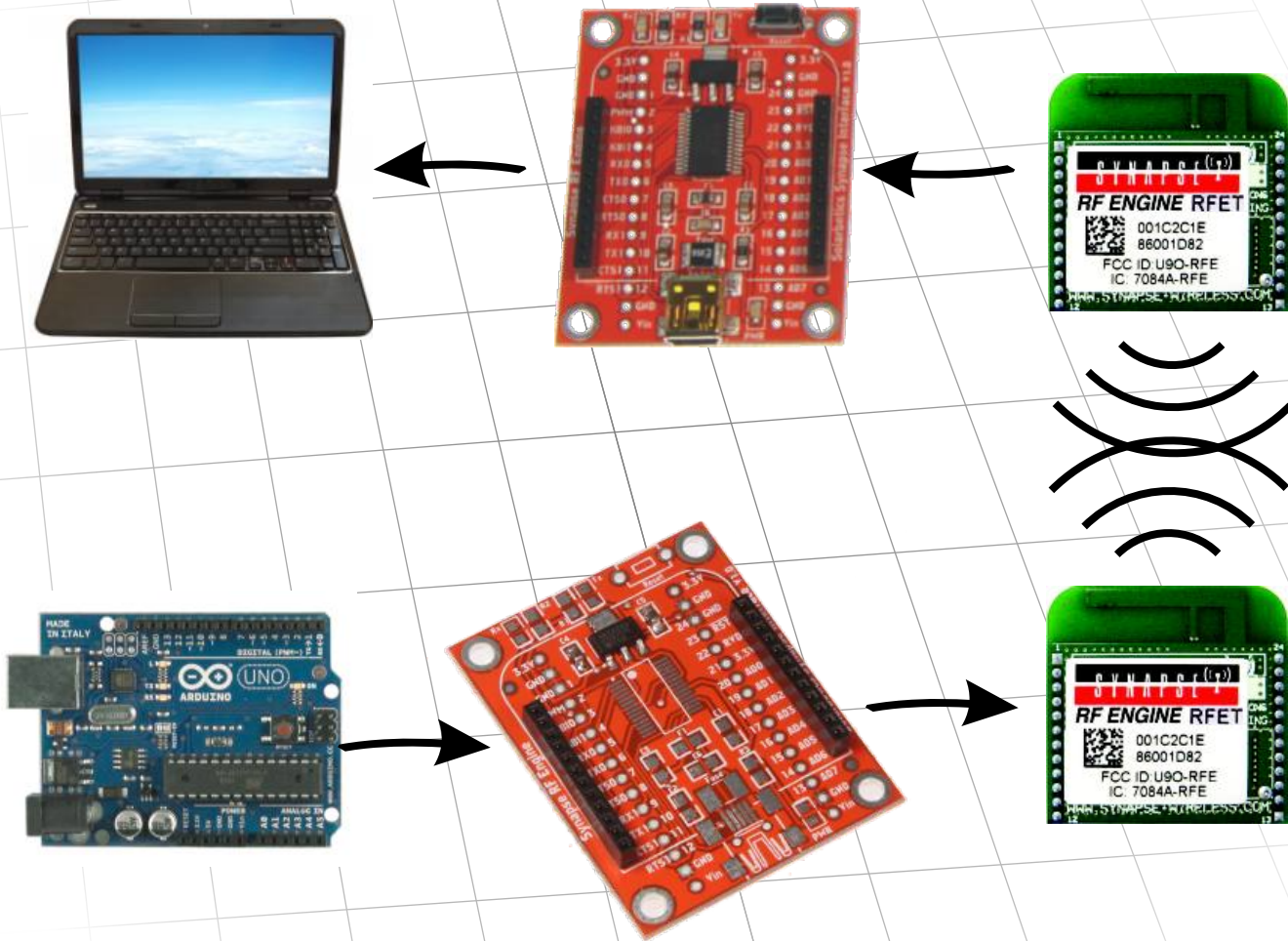


Getting Started with Synapse Wireless

Tutorial 1: Building a Wireless Serial Link to your microcontroller Project



open source
hardware



SOLARBOTICS[®]
Ltd

www.solarbotics.com
1-866-276-2687

Synapse Wireless - What is it?

Most people are already familiar with Zigbee & XBee. Synapse falls into the same family of 802.15.4 parts, using the 915MHz & 2.4GHz spectrum to build wireless data communication devices.

What makes them different is what they bring new to the table is their feature set, range, and price. They offer the ability to mesh-network *natively*. Power them up, and they mesh automagically! Other noteworthy advantages of Synapse nodes are:

- Up to 3 miles line-of-sight (LOS) range. Need more range? Add another module in between!
- No central “coordinator” node required.
- 2 hardware UARTS (1 more than XBee modules) runnable at different baud rates.
- up to 20 I/O pins (4 more than XBee), driving up to 8mA/pin (vs 4mA on XBee)
- Cross-module compatible on same frequency (RF100 series talk with RF200 series), unlike XBee 1 vs 2 vs 2.5 devices
- Intelligent **embedded controller** with built-in Python. No need for a μ C for simple applications!
- Supports AES-128 bit encryption
- Supports SPI, I2C, CBUS & JTAG communication interfaces to other nodes (series dependant)
- Supports battery-saving “Sleepy Mesh” mode, which uses only 2 μ A per node!

One huge feature is that Synapse RF Engines are configured via programming rather than drop down selections or AT commands. Synapse Engines are programmed via python-based SNAPpy scripts.

What are we doing with this Tutorial:

We will be using a pair of Synapse RF Engines to communicate a serial data from an Arduino Microcontroller back to the Arduino IDE.

Do not confuse this with being able to *reprogram* the Arduino wirelessly - it is similar, but requires some additional steps and circuitry to accommodate Arduino bootloader quirks, and is covered in another tutorial.

More information on Synapse RF engines:

Synapse Customer Forums: <http://forums.synapse-wireless.com/index.php>

Synapse Youtube channel: <http://www.youtube.com/user/SynapseWirelessInc>

Solarbotics Youtube channel: <http://www.youtube.com/user/Solarbotics>

RF Engines are will be referred to as RF modules or nodes throughout this manual. Perhaps even as “data squirrels”, but only in the most affectionate sense.

Special thanks to J.C. Woltz who has helped out Solarbotics with RF Engine setup on a number of occasions.

Step 1: Parts!

- 1 x Solarbotics Synapse RF100PC6 Coordinator Bundle (39285)
- 1 x Synapse RF100PC6 Module with 'F' Antenna (51752)
- 1 x Solarbotics Synapse Breakout Board Kit (39255)
- 1 x USB A to Mini-B Cable (14080)
- 1 x 3 'AA' Battery Holder with 15 cm Leads (17020)
- 1 x 10k Resistor (Brown / Black / Orange)
- 1 x 15k Resistor (Brown / Green / Orange)

PC Equipment

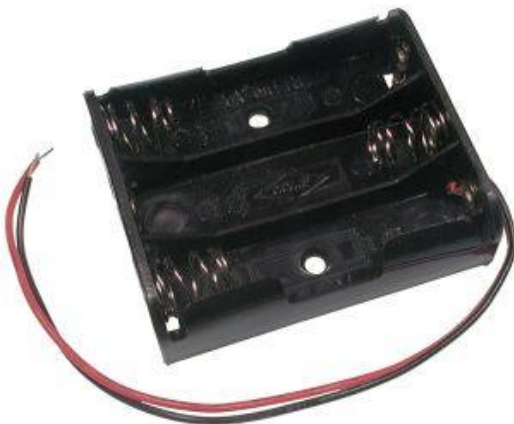


SKU: 14080



SKU: 39285

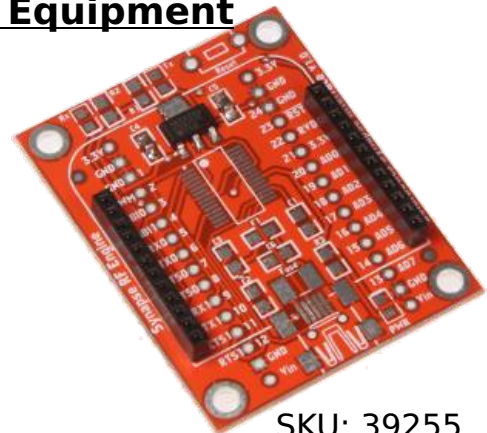
Remote (Target/Application) Equipment



SKU: 17020



SKU: 51752



SKU: 39255



10k & 15k

Step 2: Set up the Hardware

PC Hardware: Plug the RF Engine into the Breakout board kit, and use the USB cable to connect it to your PC. Simple! Now the Synapse (controller) node is powered, and will shortly be able to create your serial link.



+



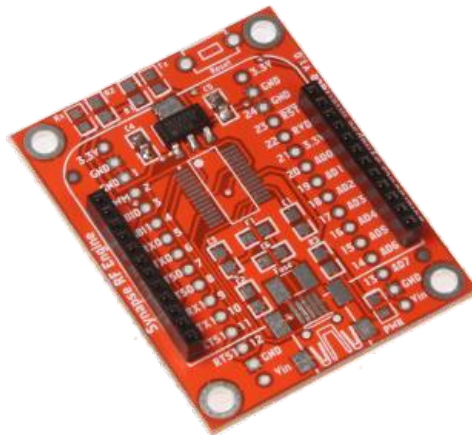
+



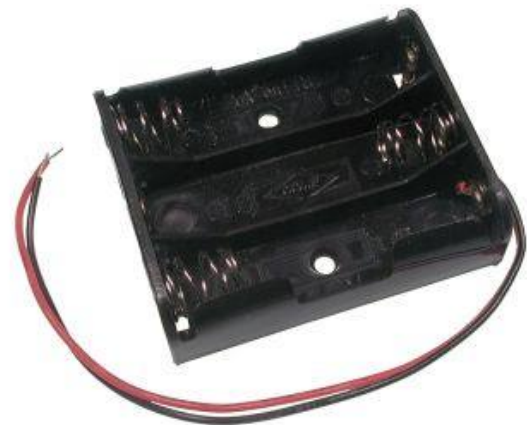
Application Hardware: As the Synapse module requires 3.3 volts, you have to supply the breakout board with 3.6 to 6 volts. The regulator is very efficient, but cannot handle more than 6 volts input! To achieve a stable 3.3V output you'll want to keep your input voltage above 3.55V.



+



+



Solder the 3 'AA' Battery Holder to the Synapse Breakout board with the **RED** wire going on the "Vin" line and the **BLACK** wire going to "GND". You can place either 3 'AA' alkaline batteries (3 x 1.5V = 4.5V) or 3 'AA' Rechargeable batteries (3 x 1.2V = 3.6V) to power your breakout board. Both of these voltages are under 6V but above 3.55V, the prime range we want to be in.

Step 3: Download the Software / Installing Drivers

Open up the Synapse webpage at www.synapse-wireless.com, and click on **"Support" / "Software/Manual Download"** links.

You will have to sign up for a free account to get the latest setup files, but this also gives you full access to their customer forums as well.



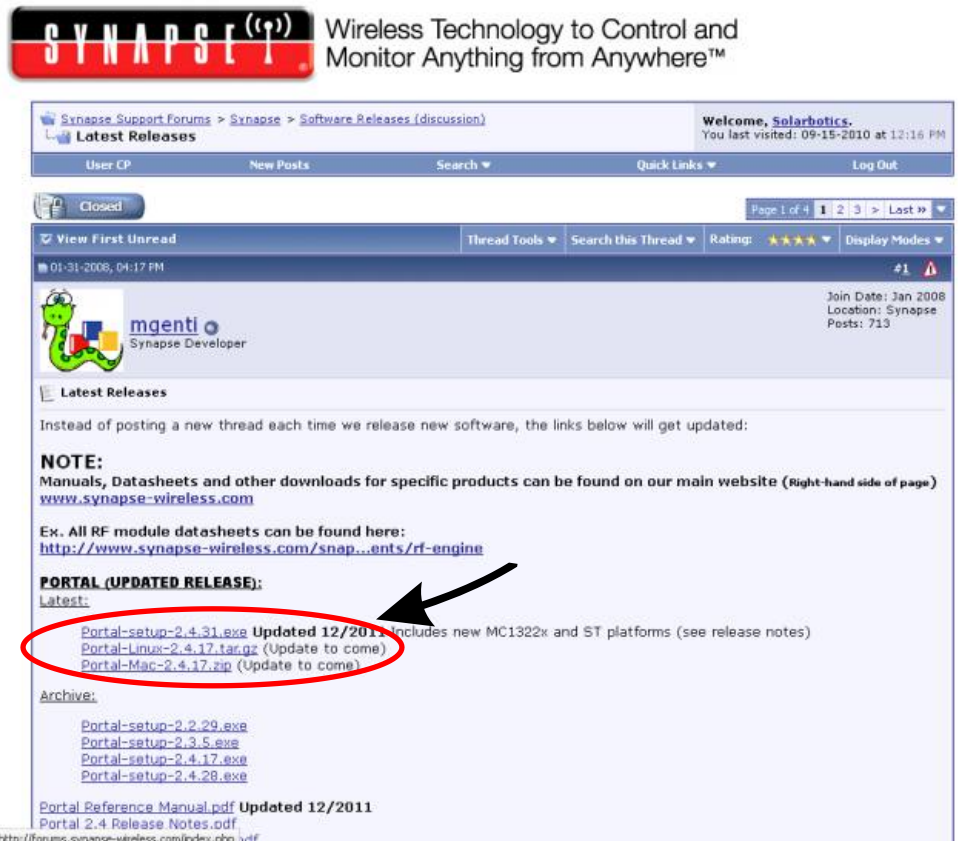
Clicking on that link, you'll be piped straight to the community forum where all the latest download links are listed.

You will have to sign-up for a free forum account.

Download the setup file, install it (don't attach the nodes yet!), and start the software

The Portal software is your key interface to all Synapse functions like firmware upgrade, network monitoring, coding and node setup.

Now that it's ready to run, plug your Synapse node into the FTDI carrier and plug it into your USB jack!

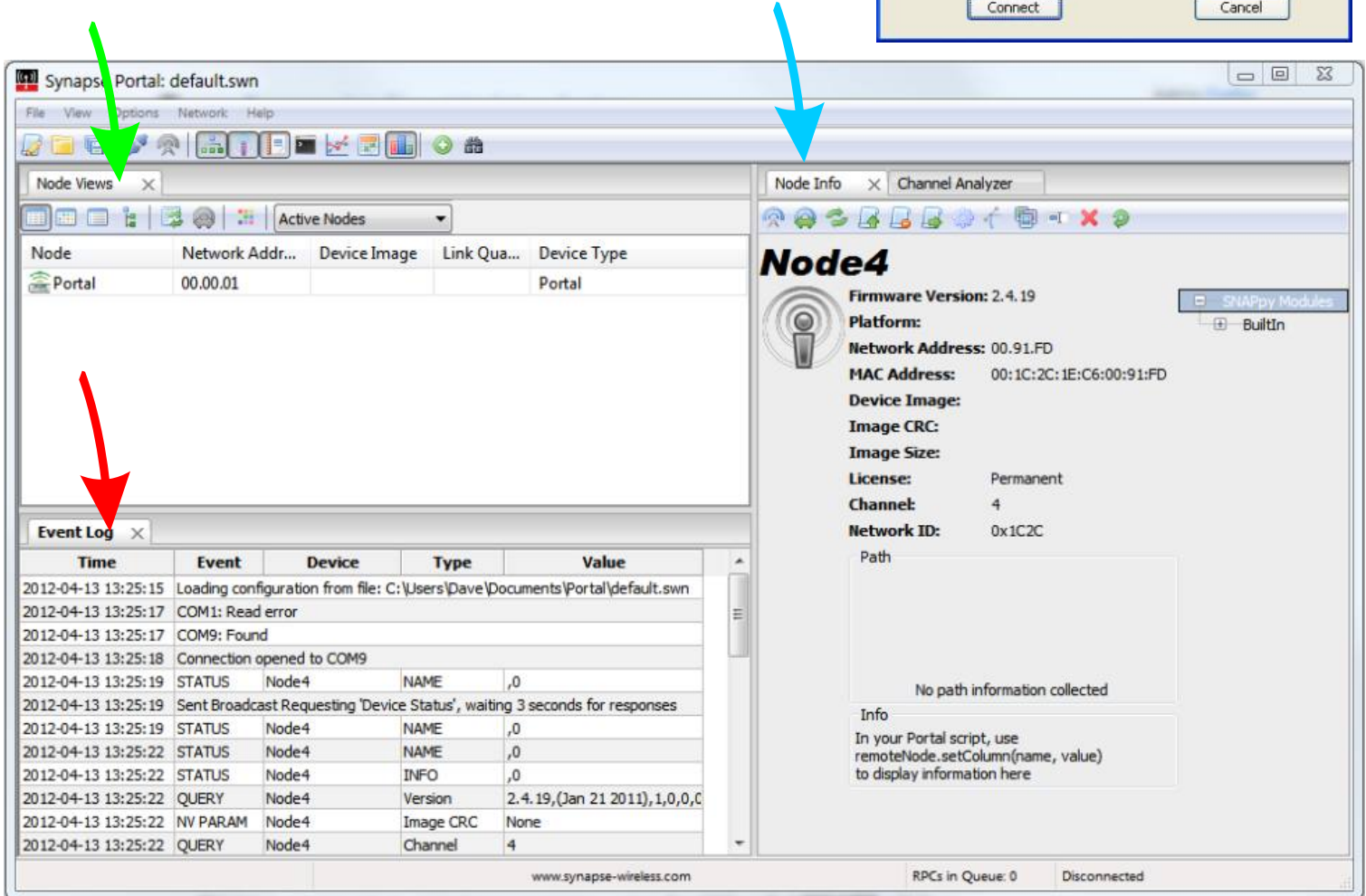
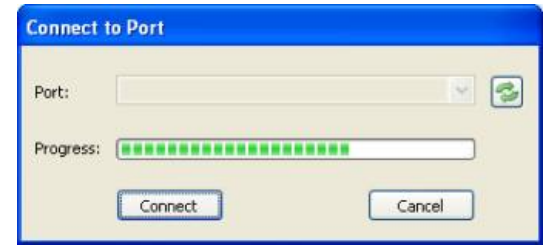


If this is the first time you've installed an FTDI-based USB-to-Serial adapter, you'll see this window pop up. FTDI drivers are built into Windows from XP onwards, so a simple "continue" will install the driver.

Now we're ready to start using the Portal software!

Step 4: Exploring the Portal Software

When starting the software, you'll see a pop-up that attempts to find a Synapse Node on a local COM port. Assuming you've plugged your node in, you'll *automatically* have a option to connect to that virtual COM port. If not, the search will just time out and let you know that no SNAP bridge device was found.



The main interface of the Synapse Portal software. It features a menu bar (File, View, Options, Network, Help) and a toolbar. The "Node Views" tab is active, showing a table of active nodes. The "Event Log" tab is also visible, showing a list of events. The "Node Info" tab is selected, displaying details for "Node4".

Node	Network Addr...	Device Image	Link Qua...	Device Type
Portal	00.00.01			Portal

Time	Event	Device	Type	Value
2012-04-13 13:25:15	Loading configuration from file: C:\Users\Dave\Documents\Portal\default.swn			
2012-04-13 13:25:17	COM1: Read error			
2012-04-13 13:25:17	COM9: Found			
2012-04-13 13:25:18	Connection opened to COM9			
2012-04-13 13:25:19	STATUS	Node4	NAME	,0
2012-04-13 13:25:19	Sent Broadcast Requesting 'Device Status', waiting 3 seconds for responses			
2012-04-13 13:25:19	STATUS	Node4	NAME	,0
2012-04-13 13:25:22	STATUS	Node4	NAME	,0
2012-04-13 13:25:22	STATUS	Node4	INFO	,0
2012-04-13 13:25:22	QUERY	Node4	Version	2.4.19,(Jan 21 2011),1,0,0,0
2012-04-13 13:25:22	NV PARAM	Node4	Image CRC	None
2012-04-13 13:25:22	QUERY	Node4	Channel	4

Node4
Firmware Version: 2.4.19
Platform:
Network Address: 00.91.FD
MAC Address: 00:1C:2C:1E:C6:00:91:FD
Device Image:
Image CRC:
Image Size:
License: Permanent
Channel: 4
Network ID: 0x1C2C
Path:
No path information collected
Info:
In your Portal script, use remoteNode.setColumn(name, value) to display information here

The **Node Views** tab is an “at-a-glance” summary of the synapse nodes active in your local area (listed by the last 6 digits of their specific MAC address). Power up your application node, and it will show up here (click the “Ping” radio-tower button to refresh the network)

The **Node info** tab allows modification of code and settings of individual nodes on the network. Clicking a node here allows you to upload code via the “file & vertical-green arrow” upload icon in the Node info tab. As long as you program the PC's controller node last, you can wirelessly reprogram *all* the application nodes on your network.

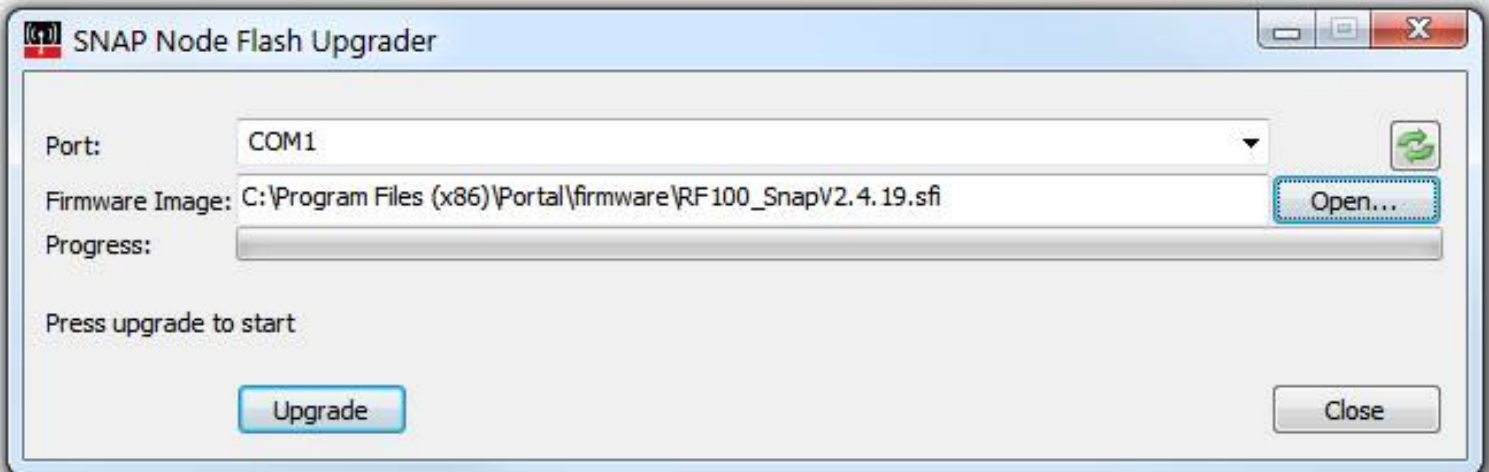
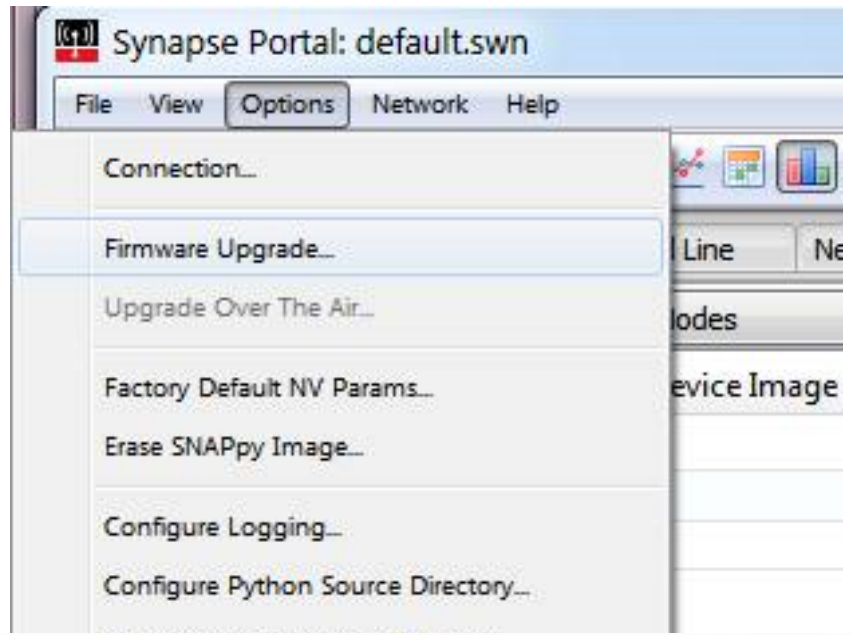
The **Event log** contains a description of what is happening to the network, including node firmware versions, your connected COM ports and much more.

Step 5: Upgrading Firmware

At present, the easiest way to get the latest firmware is to re-download the Synapse Portal software, which always contains the latest firmware (version 2.4.19 shown below).

Of note is the greyed out “Upgrade Over The Air” option, which teases us with the ability to *not* have to physically attach your modules to your PC to do the upgrade. Stay tuned!


Until then, you’ll need your FTDI adapter board, as upgrading firmware is one of the cases where the RF module will require a switch to reset the module.



Upgrading firmware: Please, please, *please* do the firmware upgrade to your modules at your earliest convenience. Minor inconsistencies between firmware can cause issues between nodes on your network!

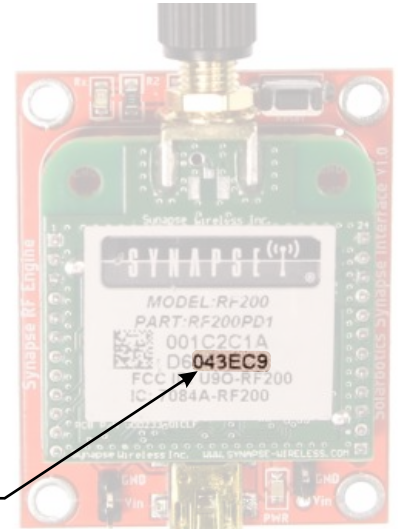
- If you haven't already, plug your node / FTDI adapter set into your PC. Note what COM port it activates as.
- You will have to *disconnect* from the COM port you're using. Click on the 4th icon (which looks like two plug-ins) on the top icon menu bar. When prompted to, click "Yes".
- From the Options Menu select "Firmware Upgrade..."
- Then select your COM port from the Option drop down list.
- Open the firmware image corresponding to your node (In this case "**RF100_SnapV2.x.x.sfi**" which is found in the Portal\firmware folder), and then click the **Upgrade** button.
- If the node is found, you are then prompted to press and release the physical Reset button on the FTDI board, which starts the firmware upload.
- Repeat as necessary for the remaining nodes at the "Upgrade another node" prompt.

Step 6: How to Upload Code (but don't do it quite yet!)

Click on the "connect" icon  and reconnect to your COM port and nodes. In a few seconds, your new firmware version number should show up in the **Event Log tab** at the bottom.

Power up the remote application breakout board. Once connected, you will see the MAC address of both your modules. If not, hit the ping icon (antenna icon) and it will refresh the list of nodes you have available. After you can see *both* RF nodes in Portal, you can program them!

The last 6 hexadecimal digits listed in the MAC addresses in **Node Views tab** will match the numbers on the sticker on top each Synapse node.



MAC Address

1. Network Refresh

2. Select Node

3. Upload Button

Node	Network Addr...	Device Image	Link Qua...	Device Type
Portal	00.00.01			Portal
Node4	00.91.FD		100%	None
Node5	00.92.3D		100%	None

Time	Event	Device	Type	Value
2012-04-13 13:43:18	NV PARAM	Node4	MAC Address	00:1C:2C:1E:C6:00:91:FD
2012-04-13 13:43:18	QUERY	Node4	SNAPpy Space	13248
2012-04-13 13:43:18	NV PARAM	Node4	Device Type	None
2012-04-13 13:43:18	STATUS	Node5	INFO	,18
2012-04-13 13:43:18	QUERY	Node5	Version	2.4.19,(Jan 21 2011),1,0,0,0
2012-04-13 13:43:19	NV PARAM	Node5	Image CRC	None
2012-04-13 13:43:19	QUERY	Node5	Channel	4
2012-04-13 13:43:19	QUERY	Node5	Network ID	0x1C2C
2012-04-13 13:43:19	NV PARAM	Node5	MAC Address	00:1C:2C:1E:C6:00:92:3D
2012-04-13 13:43:19	QUERY	Node5	SNAPpy Space	13248
2012-04-13 13:43:19	NV PARAM	Node5	Device Type	None

Node Info

Node4

Firmware Version: 2.4.19

Platform: SNAPpy Modules

Network Address: 00.91.FD

MAC Address: 00:1C:2C:1E:C6:00:91:FD

Device Image: BuiltIn

Image CRC:

Image Size:

License: Permanent

Channel: 4

Network ID: 0x1C2C

Path

No path information collected

Info

In your Portal script, use `remoteNode.setColumn(name, value)` to display information here

1. Refresh the nodes, if you can't see the one you want to program.
2. Select a node in the **Node views tab**. You now can modify its code via the **Node Info tab**.
3. Pick upload, select the code you want to upload (see step 9), and away it goes!

“Hey! Where'd my Portal node go!?!?”

When programming for serial communications, program your controller node (a.k.a. the *PC USB-connected node*) **last**, as loading this code will stop it from acting as a standard mesh-connecting node, and turn it into a blind wireless link that can only be restored by erasing the PC-connected node code (use menu item *Options / Erase Snappy Image*).

Step 7: Preparing the Arduino for Serial Data Output

Although this method works for any microcontroller sending serial data, we're using the common Arduino platform. Setting up your Arduino and programming it is outside the scope of this tutorial, but it is covered in the "ARDX Arduino Experimenters Kit" in the free documentation.

We'll use the Arduino 1.0 built-in example sketch "ASCIITable" located under:

```
Files
├── Examples
│   ├── L
│   │   ├── 4.Communicatio
│   │   │   └── n
│   │       └── ASCIITable
```

This Arduino sketch generates a simple table of ASCII values in decimal, hex, octal and binary, and dumps it back to your Arduino IDE terminal screen. Everytime the Arduino is reset, it generates the table and stops. In this example, we'll be having these bytes fly wirelessly from the Arduino to your PC!

You'll notice this string at the beginning of the sketch, which sets the baud rate it will be communicating:

```
Serial.begin(9600);
```

We'll match this 9600 baud rate in our Synapse setup. **Load up this sketch into Arduino, and confirm you can make it echo back to your Arduino terminal via the USB connection before continuing.**

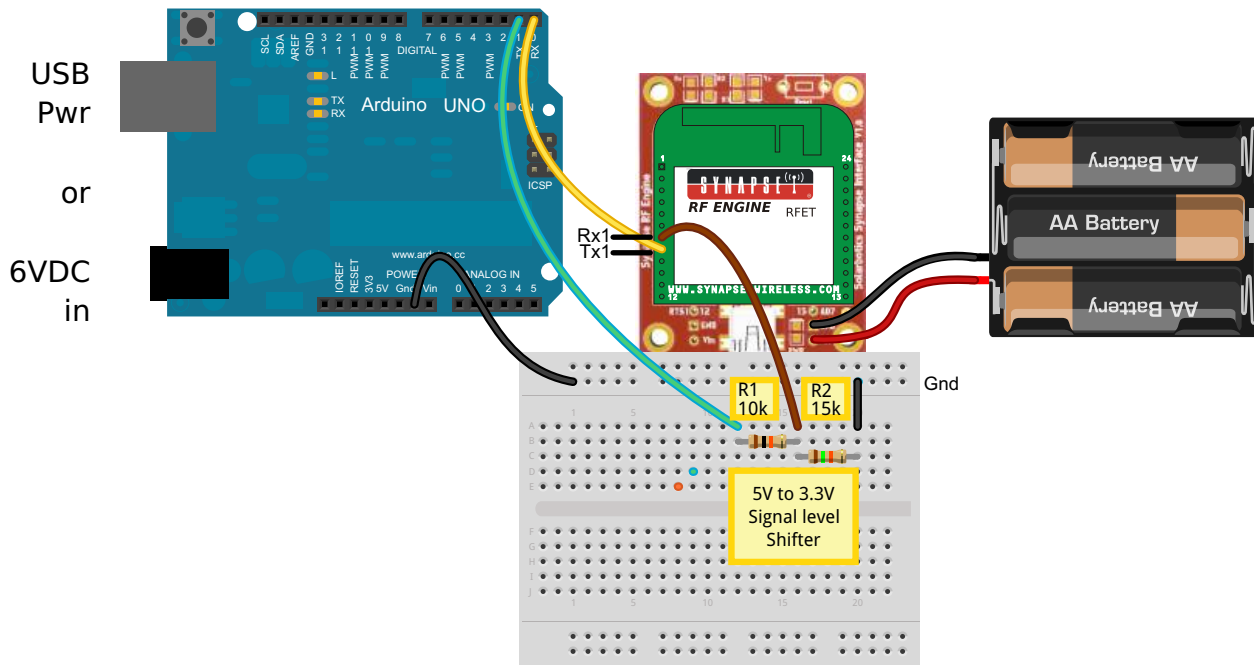
When running normally via your USB connection, the output looks like this:

```
ASCII Table ~ Character Map
!, dec: 33, hex: 21, oct: 41, bin: 100001
", dec: 34, hex: 22, oct: 42, bin: 100010
#, dec: 35, hex: 23, oct: 43, bin: 100011
$, dec: 36, hex: 24, oct: 44, bin: 100100
%, dec: 37, hex: 25, oct: 45, bin: 100101
&, dec: 38, hex: 26, oct: 46, bin: 100110
', dec: 39, hex: 27, oct: 47, bin: 100111
(, dec: 40, hex: 28, oct: 50, bin: 101000
), dec: 41, hex: 29, oct: 51, bin: 101001
*, dec: 42, hex: 2A, oct: 52, bin: 101010
+, dec: 43, hex: 2B, oct: 53, bin: 101011
,, dec: 44, hex: 2C, oct: 54, bin: 101100
-, dec: 45, hex: 2D, oct: 55, bin: 101101
., dec: 46, hex: 2E, oct: 56, bin: 101110
/, dec: 47, hex: 2F, oct: 57, bin: 101111
0, dec: 48, hex: 30, oct: 60, bin: 110000
1, dec: 49, hex: 31, oct: 61, bin: 110001
2, dec: 50, hex: 32, oct: 62, bin: 110010
3, dec: 51, hex: 33, oct: 63, bin: 110011
4, dec: 52, hex: 34, oct: 64, bin: 110100
5, dec: 53, hex: 35, oct: 65, bin: 110101
6, dec: 54, hex: 36, oct: 66, bin: 110110
7, dec: 55, hex: 37, oct: 67, bin: 110111
8, dec: 56, hex: 38, oct: 70, bin: 111000
9, dec: 57, hex: 39, oct: 71, bin: 111001
:, dec: 58, hex: 3A, oct: 72, bin: 111010
;, dec: 59, hex: 3B, oct: 73, bin: 111011
<, dec: 60, hex: 3C, oct: 74, bin: 111100
=, dec: 61, hex: 3D, oct: 75, bin: 111101
>, dec: 62, hex: 3E, oct: 76, bin: 111110
?, dec: 63, hex: 3F, oct: 77, bin: 111111
@, dec: 64, hex: 40, oct: 100, bin: 1000000
A, dec: 65, hex: 41, oct: 101, bin: 1000001
B, dec: 66, hex: 42, oct: 102, bin: 1000010
C, dec: 67, hex: 43, oct: 103, bin: 1000011
```

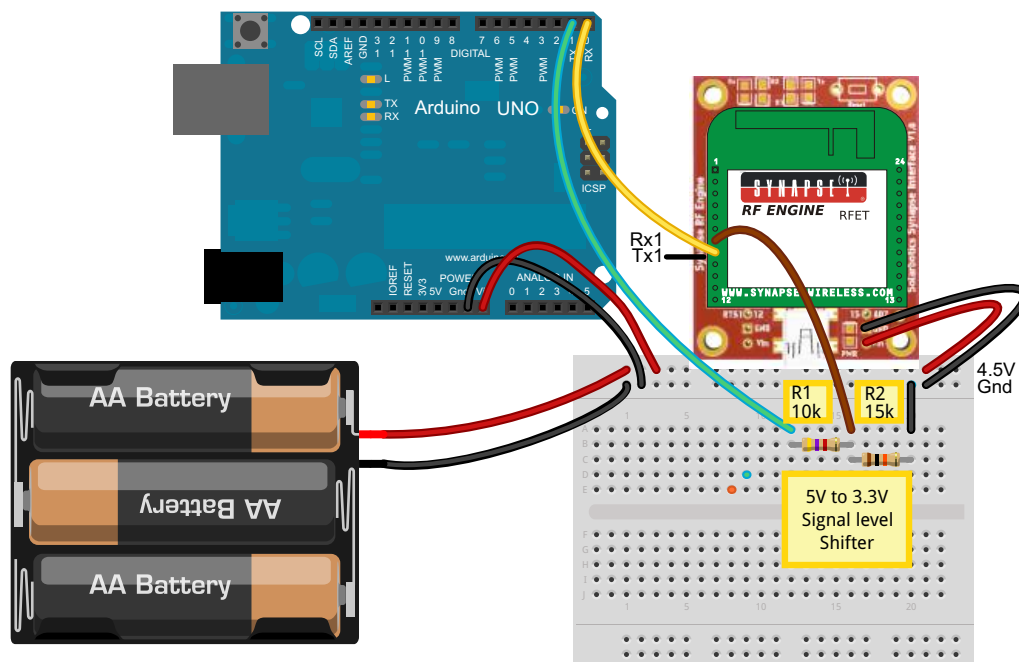
Step 8: Wiring up the Synapse Node to the Arduino

Since the Arduino will be putting out a 5V serial signal into the 3.3V Synapse input, we're piping that signal through a *shift-leveler*, which is made up from 2 resistors. We don't worry about the shift-leveler going *from* the Synapse to the Arduino, as the Arduino understands the 3.3V "high" voltage as a proper signal input. Remember to power your Arduino from it's own power source (or run the 4.5V battery pack through the Arduino, and use the Arduinos 3.3V output to power the Synapse node).

Powering Arduino & Synapse Powered Separately



Powering Arduino & Synapse From same 4.5V Power Supply



Step 9: Prepare & Upload Synapse Controller & Remote Code

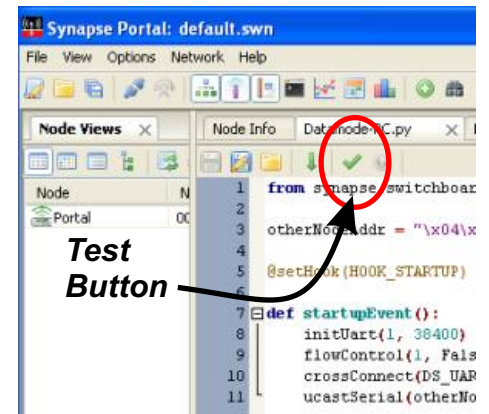
Let's prepare the SNAPpy Scripts for the controller (PC-side) and remote (Arduino-side) Synapse nodes. The code listed below is all you need for each. Eight lines, that's it. Either type them in yourself, or download these scripts from our website. Use the "File / New Script" menu option in Portal to start a new script. To check your code, use the "test script" icon in the tool bar.

Note the bold text in the script listings below - there are two important settings:

1) "otherNodeAddr" - this is the MAC address of the other Synapse node being talked to in the serial link. It needs to know where to address the data to, just like the address on a letter being mailed. YOU MUST change this address to match your Synapse Node address! This example shows the MAC address of the nodes we are using, which are unique among the available 16 million addresses.

2) "initUart(1, 9600)" - We're referring to the 9600 - this is the speed the nodes are talking to each other. This has to match the same baud rate selected in your Arduino sketch, in the line "Serial.begin(9600);".

Until you load this new transparent serial code to your controller (PC-side) node, you'll be able to still see all the other nodes and what is loaded on them. When you load it to the controller, you'll have to disconnect the serial line (4th icon on top bar) so you can link your Arduino to the same port.



Datamode-Arduino.py SNAPpy Script - Upload this code to the remote Arduino-attached node

```
from synapse.switchboard import *

otherNodeAddr = "\x04\x3E\xC3" # <= put the address of the PCs USB-attached node here

@setHook(HOOK_STARTUP)

def startupEvent():
    initUart(1, 9600) # <= put your desired baud rate here!
    flowControl(1, False) # <= set flow control to True or False as needed
    crossConnect(DS_UART1, DS_TRANSPARENT)
    ucastSerial(otherNodeAddr)
```

Watch your event log - don't worry if you see a "Did not get an expected response within timeout period from xxxxD". That's just the node resetting.

Datamode-PC.py SNAPpy Script - Upload this code to the local USB-connected PC node

```
from synapse.switchboard import *

otherNodeAddr = "\x04\x3E\xC9" # <= put the address of the Arduino-attached node here

@setHook(HOOK_STARTUP)

def startupEvent():
    initUart(1, 9600) # <= put your desired baud rate here!
    flowControl(1, False) # <= set flow control to True or False as needed
    crossConnect(DS_UART1, DS_TRANSPARENT)
    ucastSerial(otherNodeAddr)
```

Now that you have the nodes configured to communicate serially over a wireless link, let's test it out!

Step 10: Testing your link!

Just to confirm - you've done the following, right?:

- You've loaded the ASCIItable Arduino code to your Arduino.
- You've confirmed it works, echoing the table back to your Arduino serial console.
- You connected your PC Synapse node to the PC, and can see it in the Portal software.
- You powered your remote Arduino Synapse node, and can see it in the Portal software.
- You've modified the *Datamode-Arduino.py* SNAPpy with the PC node MAC address & loaded it to the Arduino node.
- You know what COM port the PC node Synapse Adapter acts as (i.e.: COM4).
- You know what baud rate the programs are expected to talk (i.e.: 9600) and confirmed the nodes are configured to talk at this speed.
- You've modified the *Datamode-PC.py* SNAPpy script with the Arduino node MAC address & loaded it to the PC node (and **poof** you can't see your nodes anymore).

Let's get these talking wirelessly!

- 1) In your **Portal software**, click on the "disconnect" icon. You have to free up the COM port so you can connect Arduino to it.
- 2) in your **Arduino IDE software**, click on "Tools / Serial Port" and select the COM port of the PC node Synapse Adapter (i.e.: COM4). This will be different from the COM port you used to originally test the ASCIItable code when using the Arduino directly with a USB cable.
- 3) Open the Arduino "Serial Monitor" with either CTRL+SHIFT+M, or use the Arduino toolbar icon.
- 4) Take a deep breath
- 5) Press the "Reset" button on the Arduino board.

Assuming everything is connected correctly, blinky lights will blink on the Arduino board, and then you'll see a burst of text show up in the serial monitor window, just like when you did it wired up with a USB cable. Congratulations! You're talking serial, wirelessly!

Troubleshooting (not that you'll need it, right?)

Recheck the Step 10 checklist. No luck? Here's some tips:

Can't find which COM port your adapter is installed on or Node not showing up in Portal?

Use Your Windows Device Manager to find if the FTDI adapter board is showing up under the COM port section. (Windows7 - Start button / Run / "Device Manager")



FTDI virtual COM port drivers didn't install properly?

This shouldn't be a problem with Windows XP or better, but you can download the drivers directly and reinstall them from here: <http://www.ftdichip.com/Drivers/VCP.htm>

FTDI adapter is working, but still can't get the Nodes to show up in Portal?

Try using the **erase function** under the **Option Menu** to refresh a node back to it's default code. To do this you'll need to hook it up directly to your FTDI board. If this doesn't seem to work, re-inspect your solder connections to the headers on the FTDI adapter board and reflow the solder if needed.

What is this "name 'DS_TRANSPARENT' is not defined" mean?

Synapse can be picky, so don't forget the top line where you have to have:
from synapse.switchboard import *

Other Applications

Data Transmission: So you have now successfully *sent* data from the Arduino to the host PC, but it's bi-directional - you can send data back to it too (assuming you hooked up the Synapse Tx line to the Arduino Rx line).

EZ-B Communication: The EZ-B is an impressive host PC / slave robot controller, which leverages the power of the PC to do most of the heavy number crunching. This means it uses wireless communication to send/receive data to the mobile robot, which is usually done with the default Bluetooth serial link. This solution replaces the Bluetooth serial link with the *much* more robust and longer range Synapse solution. You can use it to allow your robot to move outside your house to explore the backyard and head on down the street! Just boot up EZ-builder and connect to the COM port that your Synapse adapter is on, and you are ready to go - no bluetooth pairing required!

To interface to an EZ-B, remove Bluetooth module. Wire up the Tx from Synapse node to Rx on EZ-B, and Rx from Synapse *through the resistor voltage divider* to Tx, just like you did for the Arduino. Don't forget wiring the power, being the node "Vin" to EZ-B "V+" and GND to GND.

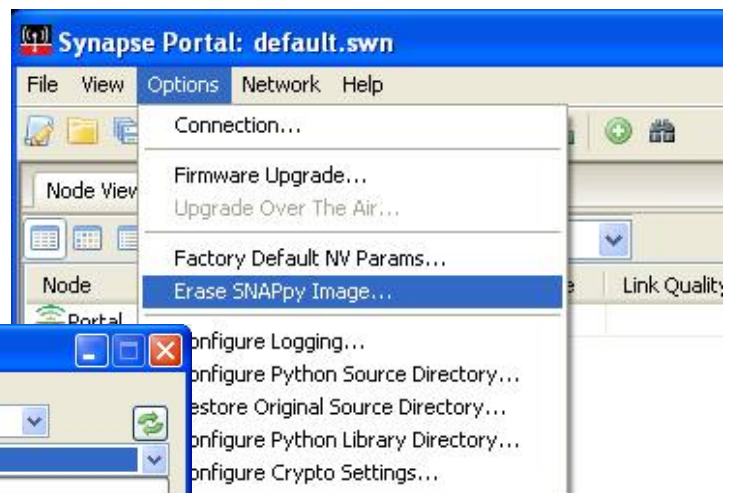
Wireless Arduino Programming: A Synapse wireless link between your Arduino and host PC is actually fast enough to allow for **wirelessly programming an Arduino Uno or Mega!** No more cables! Although you can do this with RF100 nodes (for the Uno but not the Mega), the **RF200** nodes better sustain the required 115200 baud necessary to satisfy the bootloader. There's some minor hardware adjustments to add (reset line transistor) and some other tricks to accomplish this, and will be the topic of another tutorial.



Done with the transparent link, and want to go back to mesh-networking defaults again? Remember - just erase and reset the nodes, and you're back to regular mode!

Reminder: How to Erase & Reset Synapse Nodes

- Go into the "Options" menu, select "Erase SNAPpy Image"
- Select your COM port, Platform, and then click the Erase button.
- You will again be prompted to push the reset button, hit the button on the FTDI adapter and your node will then start erasing. The node will be erased once the progress bar is finished.



Where to From Here?

We've already touched on some of the aspects that they Synapse wireless nodes offer as a dumb transparent serial, and how it can even be leveraged to remote wireless programming Arduino modules. What is of particular interest is the *Embedded Python Interpreter*.


We are experimenting on what the processors in each node are capable of, using the I/O lines on the nodes for active participants in the circuits instead of just dumb reporters/receivers of data. Solarbotics loves these modules, and will keep developing around them!


Visit us online for more info and cool stuff:

www.solarbotics.com

Solarbotics Ltd.

3740D - 11A Street NE, Suite 101
Calgary, Alberta T2E 6M6
Canada

 **Toll Free:** 1-866-276-2687
International: +1 (403) 232-6268

 **Fax:** +1 (403) 226-3741

 Made in Canada