Processing Controls R/C Car with XBee modules

by **sath02** on October 3, 2012

Table of Contents

Author:sath02
I am Electronic Visualization Artist. I look at things through the Looking Glasses.

Intro: Processing Controls R/C Car with XBee modules

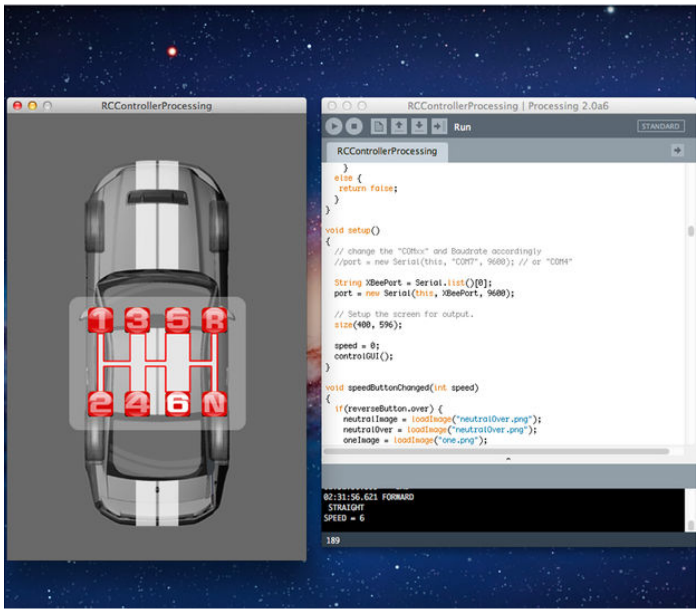
This instructable is a modification of my presentation to Processing Chicago community at the (EVL) Electronic Visualization Laboratory, University of Illinois at Chicago on Oct 1, 2012. In the topic of

Processing Library Series
Processing and Wireless XBee Wireless RF Module

In this presentation, attendants will learn about

- UART Serial Communication
- Serial Lib
- How to install FTDI Virtual Comm Port Driver
- XBee Wireless RF Module and how to configure the modules
- How to design the GUI (Graphics User Interface) in Processing (Used Processing to control R/C car as an example)
- Hands-on exercise configure the XBee modules and communicate among each other.

Note: I changed the title and modified my presentation to fit the Instructables' step-by-step format.



```
  }
 else {
  return false;
 }
}

void setup()
{
 // change the "COMxx" and Boudrate accordingly
 //port = new Serial(this, "COM7", 9600); // or "COM4"

 String XBeePort = Serial.list()[0];
 port = new Serial(this, XBeePort, 9600);

 // Setup the screen for output.
 size(400, 596);

 speed = 0;
 controlGUI();
}

void speedButtonChanged(int speed)
{
 if(reverseButton.over) {
  neutralImage = loadImage("neutralOver.png");
  neutralOver = loadImage("neutralOver.png");
  oneImage = loadImage("one.png");
```

```
02:31:56.621 FORWARD
 STRAIGHT
SPEED = 6
```

189

PROCESSING CHICAGO: LIBRARY SERIES
Processing and XBee Wireless RF Module
Artist Presentation and Workshop | October 1st @ 6pm
Twitter: @ProcessingCHI
http://www.evl.uic.edu/processing-chicago/

Image Notes
1. Digital Poster for my presentation

Step 1: Materials

The presentation was divided into two parts, first part was about my few past works:

Wireless (Bluetooth) Control Pop-Bot
Make Wired Robotic Arm Edge to "Wireless" with DIY Arduino + XBee and
Wireless Finger Drum ,
XB-Buddy and XB-Buddy Basic Kits

And introduced a new project. (this instructable - Processing Controls R/C Car with Wireless XBee modules)

Second part, was an in-depth workshop on working with the XBee configuration, the Processing's Serial library, Arduino's Serial and SoftwareSerial library.

In case, if you want to build the project, below are the list of Hardware, Software and Parts.
I also provided the step-by-step details (Step 11) to build the PCB, and modification of the R/C car (which was not shown at the presentation).

Hardware

Arduino or Arduino Compatibles
XBee Modules
FTDI cable or equivalent (i.e. XB-Buddy, Adafruit's FTDI Friend, Sparkfun's Breakoutboard, Librelium's ...)

Motors Driver Board or equivalent

Cheap R/C car with min. 6VDC. battery power (example used was 1:16 Scale Ford Mustang Shelby GT 2010, with 7.5VDC)

Driver & Softwares

FTDI - Virtual Comm Port Drivers
CoolTerm - Free Terminal Program
Processing IDE
Arduino IDE

Parts list for DIY Motors Driver PCB and DIY Arduino

If you're are scratch building DIY Arduino and DIY Motors Driver Following is the parts list, (See Step 11 for Schematics and Diagrams)

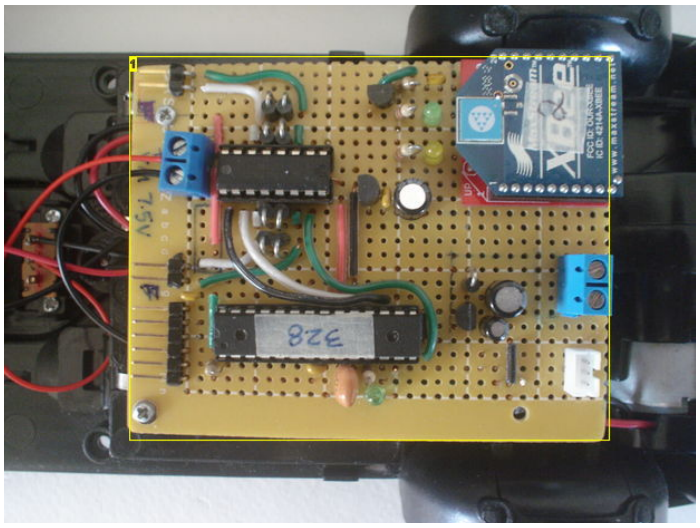
(x1) ATMega328P with Arduino bootloader
(x1) 28-pin IC Socket (Digikey #3M5480-ND)
(x1) 16Mhz Resonator
(x3) 0.1uF Ceramic capacitor
(x2) 100uF/16V capacitors
(x1) 10uF/16 capacitor
(x1) 10K resistortor
(x3) 1K Resistor
(x1) 6-pin right angle male header
(x1) 3mm green LED
(x1) 3mm red or yellow LED
(x2) 2 pin receptacle

(x1) L293D - Motor Drivers IC (or SN745)
(x1) 16-pin IC Socket

0.1" grid PCB (about 2-3/4" long x 3-1/2" wide)
Hook-up wire

Tools

Solder iron and Soldering Work Station
Solder - Rosin Core (Radio Shack #64-013)
Hookup Wire
Multimeter
Diagonal cutter
Pliers
X-Acto Knife
Wire Stripper
Solder Sucker



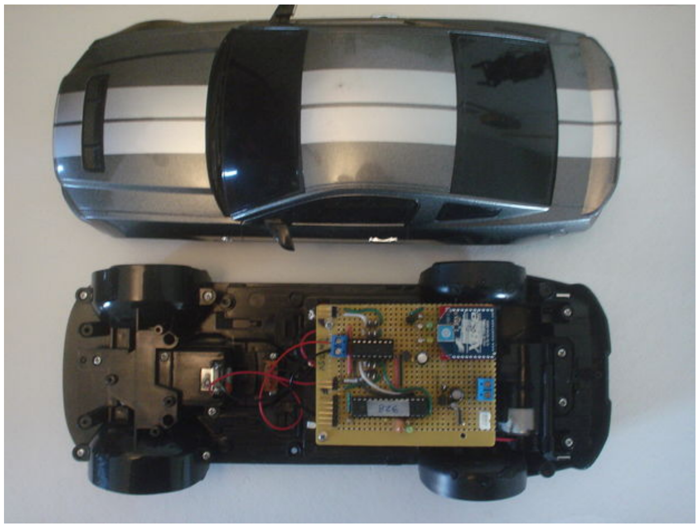


Image Notes
1. Actual built DIY PCB.

Step 2: Brief Introduction to Serial Communication

Brief Introduction to Serial Communication

- Serial communication means UART (Universal Asynchronous Receiver/Transmitter) or RS232
- UART is one of the oldest and most common interfaces
- Data is send out on UART TXD pin in a sequence at a predefined speed, one byte at a time
- When the transmitter sends out zeros(0) and ones(1), the receiver is checking for incoming data on RXD pin at the same speed that the transmitter is sending
- This complete one direction of transferring the data.
- Transmitter and receiver are using separate circuit
- To transfer data in opposite direction, a similar circuit is constructed at the opposite side
- Each circuit can work together or independently
- Each side can send/receive data at anytime with the same predefined speed
- This predefined speed of sending/receiving is called baud rate
- Baud rate is the number of transmitted bits per one second (bps)
- The UART baud rates are in the range of 300, ..., 9600, 19200, 38400, 115200, ..., 921600

- Two micro-controllers can directly connecting to each other, since this is just a digital I/O pins with the voltage levels between 0V and 3.3V (or 5V)
- To connect micro-controller to computer such as laptops, desktops that equipped with RS232 (with voltage level between -12V to +12V) we need a voltage level converter from TTL to RS232 or USB to Serial Converter.

Step 3: Installing FTDI Driver - Virtual Comm. Port

To use USB to Serial Converter i.e FTDI Cable, XB-Buddy, Adafruit's FTDI Friend, Sparkfun's as the Serial Communication between computer and micro controller we need to install a driver (Virtual Comm Port).
Note: I used XB-Buddy Kits and XB-Buddy Basic Kit in this presentation.

Following are the steps to do so:

Go to http://www.ftdichip.com/Drivers/VCP.htm

Scroll down, and select driver suitable for your operating system (In this presentation, we were using Mac OS X 2.2.18)

Download to your computer. It appeared in the "Downloads folder" as FTDIUSBSerialDriver_v2_2_18.dmg file.

Open the file, there are two packages contain in the .dmg file, FTDIUSBDriver_10_3 and FTDIUSBSerialDriver_10_4_10_5_10_6_10_7.

Open the FTDIUSBSerialDriver_10_4_10_5_10_6_10_7 package.

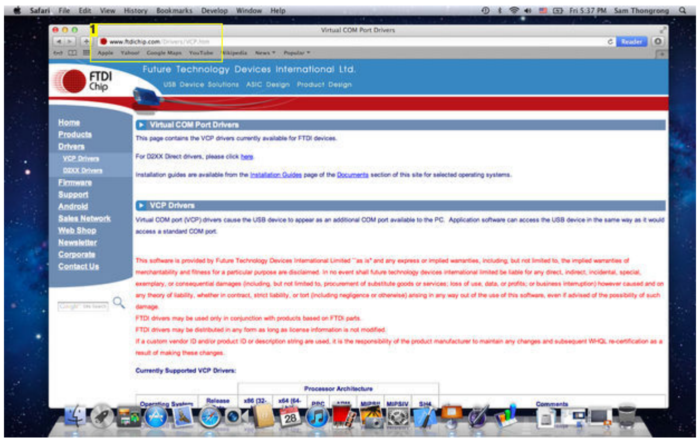
The installer window displays the installation steps. Select "Continue" button to proceed.

The Read Me page displays the read me message. After read through the read me messages, click "Continue" button.

The installation window page will ask you if you want to install the driver to different place than the default folder - Macintosh HD. Change the folder as require, otherwise keep it as default by click at "Continue" button.

Last step, the installer will ask for user name and password. Enter your user name and password. And click at "Install Software" button.

Click at "Close" button, after the installer successfully installed the driver.


Safari File Edit View History Bookmarks Develop Window Help
Fri 5:37 PM Sam Thongrong
Virtual COM Port Drivers
1
www.ftdichip.com/Drivers/VCP.htm
Reader
Apple Yahoo! Google Maps YouTube Wikipedia News Popular
FTDI Chip
Future Technology Devices International Ltd.
USB Device Solutions ASIC Design Product Design
Home
Products
Drivers
VCP Drivers
D2XX Drivers
Firmware
Support
Android
Sales Network
Web Shop
Newsletter
Corporate
Contact Us

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.

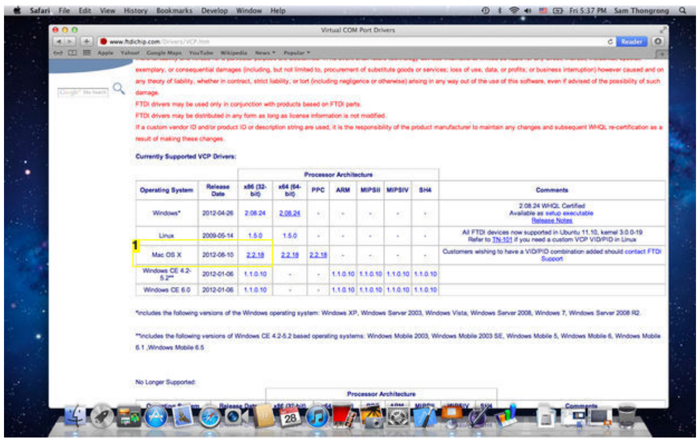
VCP Drivers

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 theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.
FTDI drivers may be used only in conjunction with products based on FTDI parts.
FTDI drivers may be distributed in any form as long as license information is not modified.
If a custom vendor ID and/or product ID or description string are used, it is the responsibility of the product manufacturer to maintain any changes and subsequent WHQL re-certification as a result of making these changes.

Currently Supported VCP Drivers:

Image Notes
1. Go to http://www.ftdichip.com/Drivers/VCP.htm


Safari File Edit View History Bookmarks Develop Window Help
Fri 5:37 PM Sam Thongrong
Virtual COM Port Drivers
www.ftdichip.com/Drivers/VCP.htm
Reader
Apple Yahoo! Google Maps YouTube Wikipedia News Popular
1

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 theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.
FTDI drivers may be used only in conjunction with products based on FTDI parts.
FTDI drivers may be distributed in any form as long as license information is not modified.
If a custom vendor ID and/or product ID or description string are used, it is the responsibility of the product manufacturer to maintain any changes and subsequent WHQL re-certification as a result of making these changes.

Currently Supported VCP Drivers:

		Processor Architecture							
Operating System	Release Date	x86 (32-bit)	x64 (64-bit)	PPC	ARM	MIPSII	MIPSIV	SH4	Comments
Windows*	2012-04-26	2.08.24	2.08.24	-	-	-	-	-	2.08.24 WHQL Certified Available as setup executable Release Notes
Linux	2009-05-14	1.5.0	1.5.0	-	-	-	-	-	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19 Refer to TN-101 if you need a custom VCP VID/PID in Linux
Mac OS X	2012-08-10	2.2.18	2.2.18	2.2.18	-	-	-	-	Customers wishing to have a VID/PID combination added should contact FTDI Support
Windows CE 4.2-5.2**	2012-01-06	1.1.0.10	-	-	1.1.0.10	1.1.0.10	1.1.0.10	1.1.0.10	
Windows CE 6.0	2012-01-06	1.1.0.10	-	-	1.1.0.10	1.1.0.10	1.1.0.10	1.1.0.10	

*includes the following versions of the Windows operating system: Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2.

**includes the following versions of Windows CE 4.2-5.2 based operating systems: Windows Mobile 2003, Windows Mobile 2003 SE, Windows Mobile 5, Windows Mobile 6, Windows Mobile 6.1 ,Windows Mobile 6.5

No Longer Supported:

Image Notes
1. Scroll down, and select driver suitable for your operating system (In this presentation, we were using Mac OS X 2.2.18)

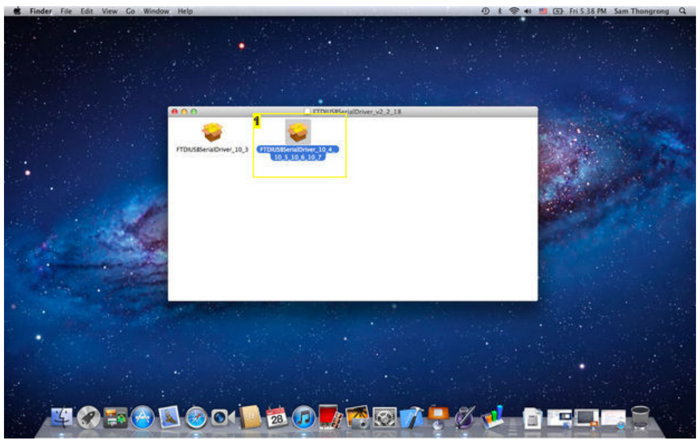


Image Notes
1. Download to your computer. It appeared in the "Downloads folder" as FTDIUSBSerialDriver_v2_2_18.dmg file.

Open the FTDIUSBSerialDriver_10_4_10_5_10_6_10_7 package.

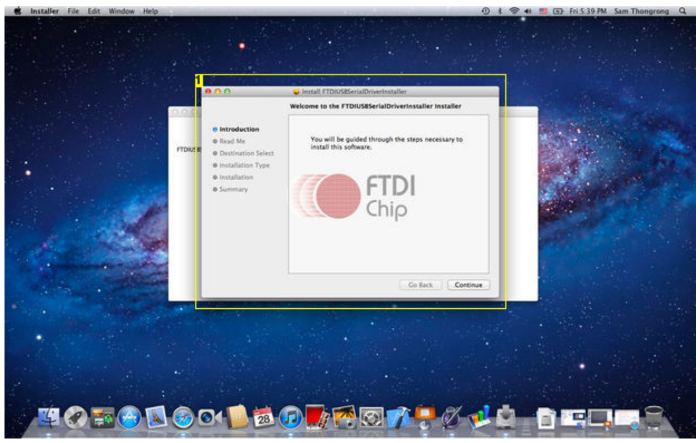


Image Notes
1. The installer window displays the installation steps. Select "Continue" button to proceed.

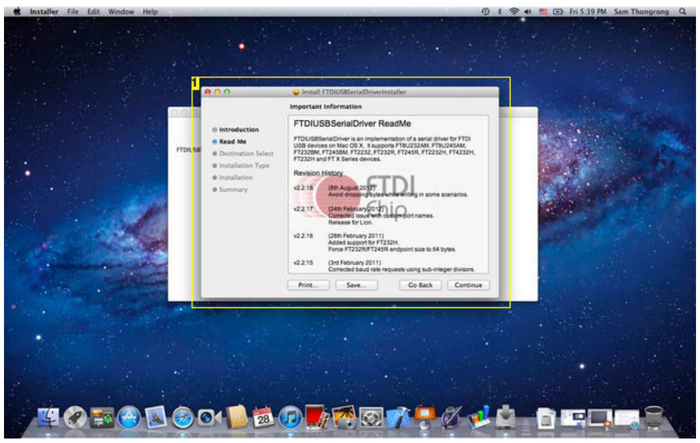


Image Notes
1. The Read Me page displays the read me message. After read through the read me messages, click "Continue" button.

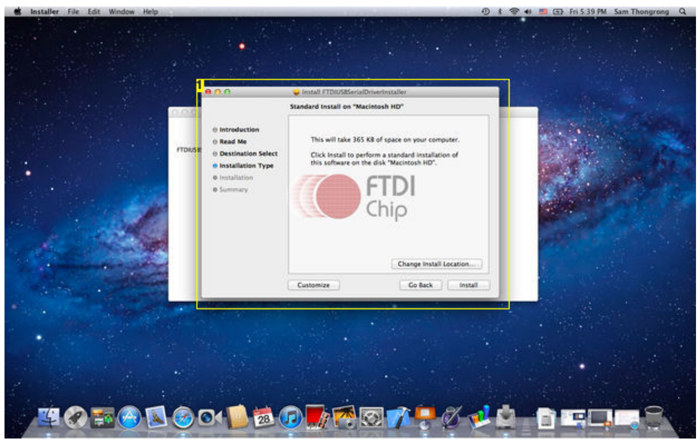


Image Notes
1. The installation window page will ask you if you want to install the driver to different place than the default folder - Macintosh HD. Change the folder as require, otherwise keep it as default by click at "Continue" button.

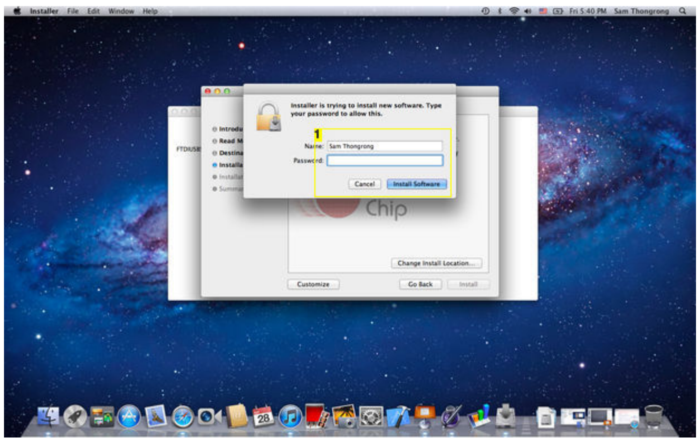


Image Notes
1. The installer will ask for user name and password. Enter your user name and password. And click at "Install Software" button.
Click at "Close" button, after the installer successfully installed the driver.

Step 4: Mac OSX Serial Comm.

How can we check if there is a Serial Comm. Port connected to Mac?

- Open the Launchpad
- Click on Utilities Icon
- Locate the Terminal App. Icon
- And click at it

The app. window open up
Type **ls /dev/tty.*** and press Enter key.
We will see that there is only one Serial port available, which is built-in bluetooth modem (image 5 below)

Now insert the USB cable to one of the USB port.

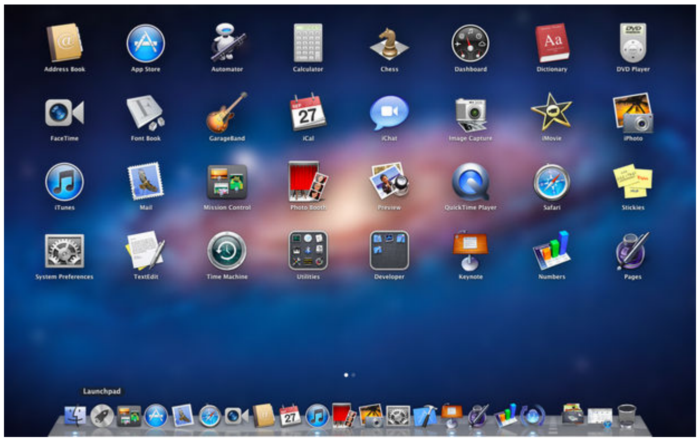
retype the command **ls /dev/tty.*** at prompt and press Enter key.

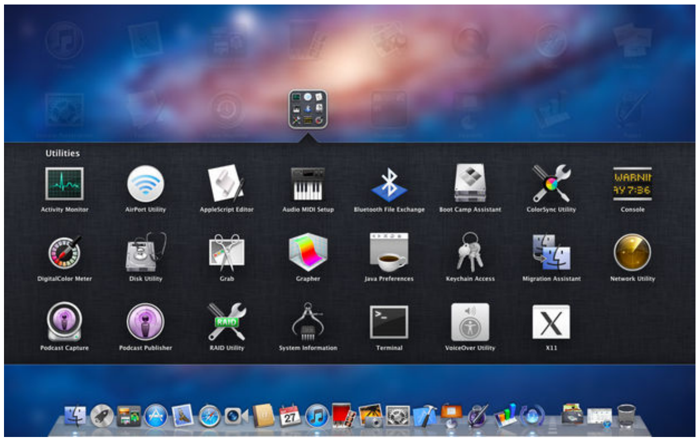
An alteration to check if there is any Serial Comm. Port connected.

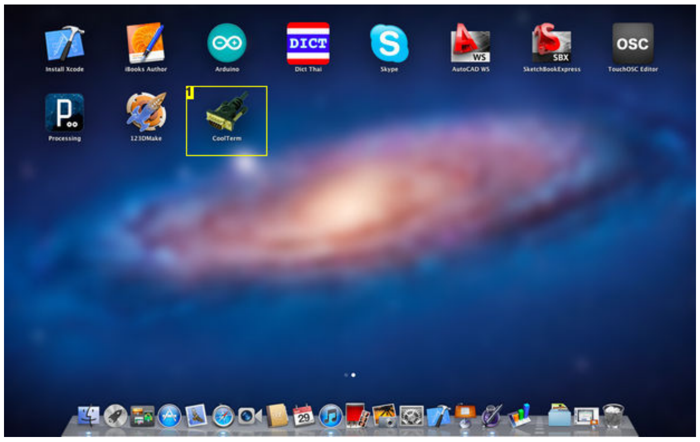
- Locate the Apple icon and click at it
- Select "About This Mac" tab
- Click at System Report button
- Select USB on the list

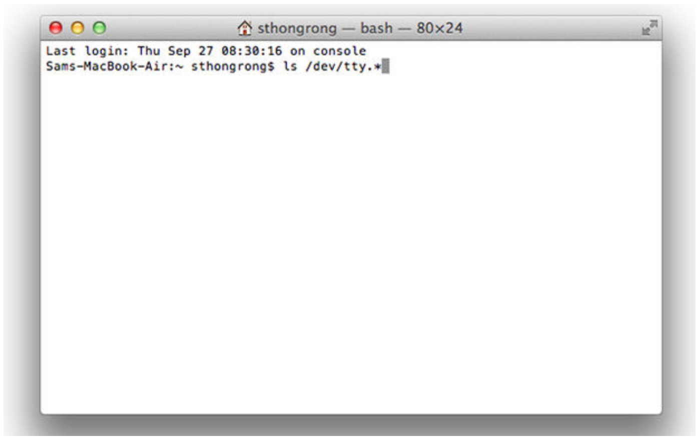
Now insert the FTDI cable

Close the **"System Report"** Window
And click at the **"System Report** " again
If everything OK, you should see the Serial Port in the USB list (As shown in the image 8 below)



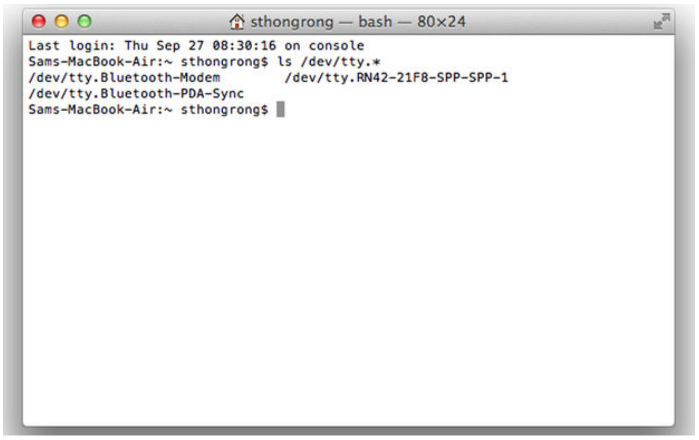




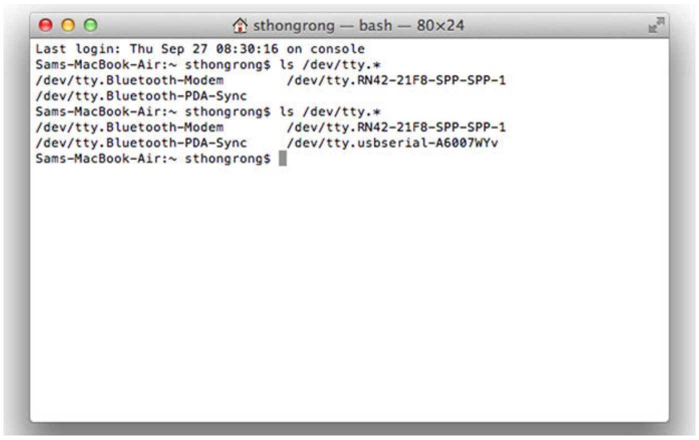


```
Last login: Thu Sep 27 08:30:16 on console
Sams-MacBook-Air:~ sthongrong$ ls /dev/tty.*
```

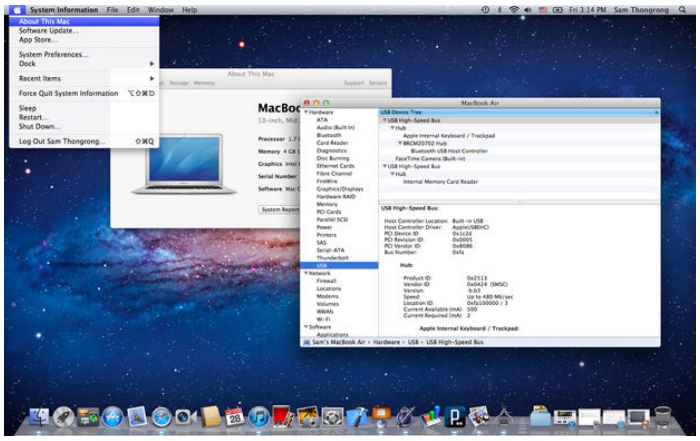
Image Notes
1. CoolTerm icon will appear on the Launchpad automatically.
Click at the icon to start the app.

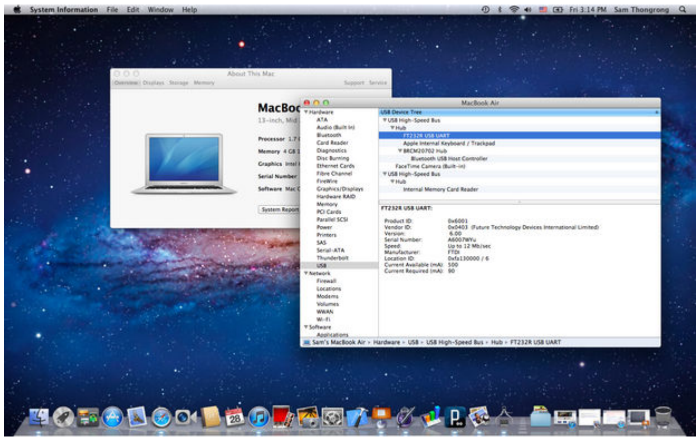


```
Last login: Thu Sep 27 08:30:16 on console
Sams-MacBook-Air:~ sthongrong$ ls /dev/tty.*
/dev/tty.Bluetooth-Modem        /dev/tty.RN42-21F8-SPP-SPP-1
/dev/tty.Bluetooth-PDA-Sync
Sams-MacBook-Air:~ sthongrong$
```



```
Last login: Thu Sep 27 08:30:16 on console
Sams-MacBook-Air:~ sthongrong$ ls /dev/tty.*
/dev/tty.Bluetooth-Modem        /dev/tty.RN42-21F8-SPP-SPP-1
/dev/tty.Bluetooth-PDA-Sync
Sams-MacBook-Air:~ sthongrong$ ls /dev/tty.*
/dev/tty.Bluetooth-Modem        /dev/tty.RN42-21F8-SPP-SPP-1
/dev/tty.Bluetooth-PDA-Sync     /dev/tty.usbserial-A6007WYv
Sams-MacBook-Air:~ sthongrong$
```





Step 5: Arduino Serial Library

Arduino is a simple computer with a small memory.
Arduino uses an USB connection that is used by the IDE to upload a sketch into processor.
With this connection, it can also be used by the sketches that we write in Arduino to send data back to the computer or to receive commands from it.

Arduino is already included Serial comm in it Program.
Serial Methods(functions)

On previous version of Arduino, Duemilanove, Diecimila, used FT232R chip.

On the Arduino UNO, there is an ATmega16U2 chip installed, this chip was pre-programmed to be used as USB-to-Serial converter.

[These two pins are built-in serial comm. port (TX, RX) on Arduino.]

/ /Arduino Sketch

// No need to add any header file or library in order to
// use the Serial port in Arduino sketch

int i;

void setup() {

// Set the Baud rate
Serial.begin(**9600**);

i = 1;
}

void loop() {

Serial.print(“ Hello, just saying for the “);
Serial.print(i);
Serial.print(“th time.\n”); // this line equivalent to Serial.println(“th time.”);

delay(1000);

i++;
}

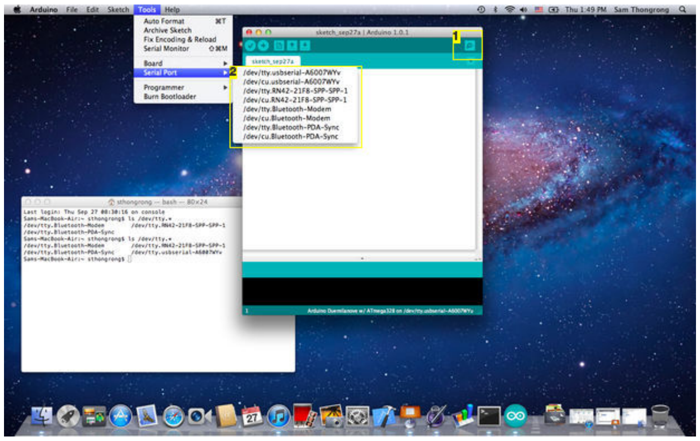
// Common serial methods (functions)
// **Serial.write(data)** - sends some data to the serial port.

// **int Serial.available()** - returns how many unread bytes are available on the serial port

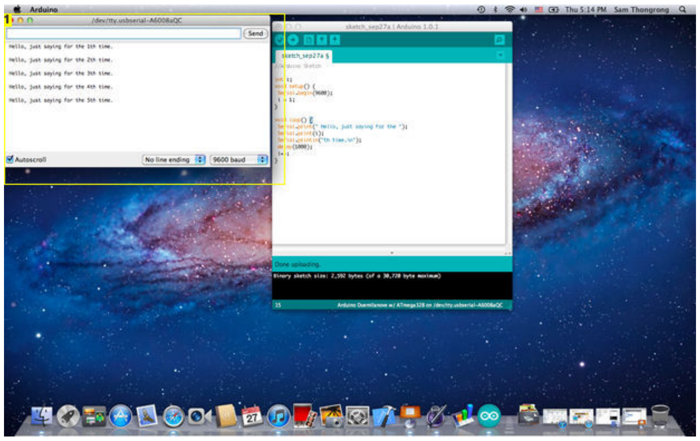
// for reading via read() function. After you have read() everything available,
// Serial.available() return 0 until new data arrived on the serial port.

// **Serial.read()** - Gets one byte of incoming serial data.

// **Serial.flush()** - clear the data in the serial buffer, and let it fill up with flesh data.


Auto Format
Archive Sketch
Fix Encoding & Reload
Serial Monitor
Board
Serial Port
Programmer
Burn Bootloader
1
2
/dev/tty.usbserial-A6007WYv
/dev/cu.usbserial-A6007WYv
/dev/tty.RN42-21F8-SPP-SPP-1
/dev/cu.RN42-21F8-SPP-SPP-1
/dev/tty.Bluetooth-Modem
/dev/cu.Bluetooth-Modem
/dev/tty.Bluetooth-PDA-Sync
/dev/cu.Bluetooth-PDA-Sync

```
Last login: Thu Sep 27 08:30:16 on console
Sams-MacBook-Air:~ sthongrong$ ls /dev/tty.*
/dev/tty.Bluetooth-Modem        /dev/tty.RN42-21F8-SPP-SPP-1
/dev/tty.Bluetooth-PDA-Sync
Sams-MacBook-Air:~ sthongrong$ ls /dev/tty.*
/dev/tty.Bluetooth-Modem        /dev/tty.RN42-21F8-SPP-SPP-1
/dev/tty.Bluetooth-PDA-Sync     /dev/tty.usbserial-A6007WYv
Sams-MacBook-Air:~ sthongrong$
```



```
Hello, just saying for the 1th time.
Hello, just saying for the 2th time.
Hello, just saying for the 3th time.
Hello, just saying for the 4th time.
Hello, just saying for the 5th time.
```

```
//Arduino Sketch

int i;
void setup() {
 Serial.begin(9600);
 i = 1;
}

void loop() {
 Serial.print(" Hello, just saying for the ");
 Serial.print(i);
 Serial.println("th time.\n");
 delay(1000);
 i++;
}
```

```
Done uploading.
Binary sketch size: 2,192 bytes (of a 30,720 byte maximum)
```

Image Notes
1. Serial Monitor Window is used to display the data send/receive between Arduino and Computer.
2. After plug in the USB to Serial Converter cable or breakout board.
We can see Serial port is now also available to Arduino.

Image Notes
1. Connected the Arduino to Mac USB Port. Uploaded the sketch and run, with Serial Monitor window opened.

Step 6: Arduino SoftwareSerial Library

//Arduino Sketch

// Software Serial Library is included in Arduino IDE standard library
//To use SoftwareSerial Library, add the following line to the sketch.
#include <SoftwareSerial.h>

//Specify the pins to use as serial input/output pin.
//Digital pin 10 is specified as TX (output), and Digital pin 9 is specified as RX (input).
SoftwareSerial xBeeSerial = SoftwareSerial(10, 9);

int incomingByte;

void setup() {

// We could use both Serial and SoftwareSerial at the same time,
// since we used different digital pins to send/receive data at the same time.

Serial.begin(9600); **// Set the baud rate of existing serial port to 9600 bps.**

xBeeSerial.begin(9600); **// Set the baud rate of SoftwareSerial port to 9600 bps.**
}

void loop() {

// Set the condition, If the data in the serial buffer available, read data from port.
if(xBeeSerial.available() > 0) {

// Use the SoftwareSerial port to read the data in the serial buffer.
incomingByte = xBeeSerial.read();

// At the same time, send the data to display the value on the Serial Monitor, if it is opened.
Serial.println(incomingByte);
...
}
}

// SoftwareSerial has similar methods as Serial Library, see details here (http://arduino.cc/en/Reference/SoftwareSerial)

Reference Language | Libraries | Comparison | Changes

SoftwareSerial Library

Step 7: Processing Serial Library

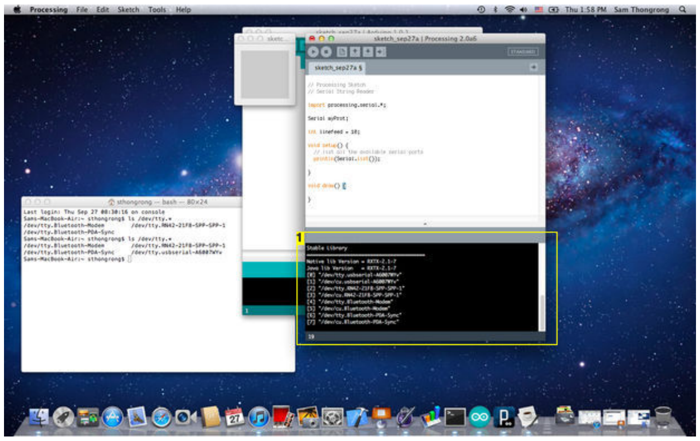
```
// Processing Sketch
// Serial String Reader #1

// In Processing, we need to import “Serial Library”
// and write a few lines of code in order to find out if we have any serial port available.
import processing.serial.*;

Serial myPort;

int linefeed = 10;

void setup() {

// Command line to list all the available serial port on the computer.
println(Serial.list());

}

void draw() {

}
```

Sample 2

```
// Processing Sketch
// Serial String Reader #2

import processing.serial.*;

Serial myPort;

int linefeed = 10;

void setup() {

// After we ran the sketch the first time.
// Now we know that our serial port is the first one on the list
// So we can comment out the println(Serial.list()); below
// List all the available serial ports
// println(Serial.list());


// and set our port to appropriate number
myPort = new Serial(this, Serial.list()[0], 9600);


}

void draw() {

}
```

To communicate between Processing (PC, Laptop) and Arduino the baud rate have to be the same.

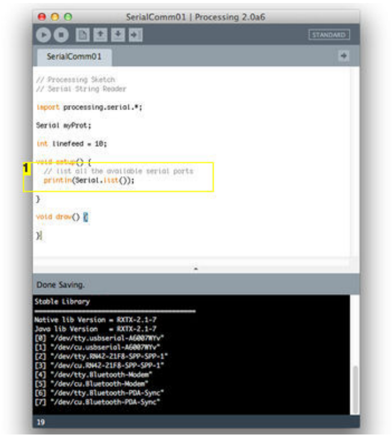
Processing
An open project initiated by Ben Fry and Casey Reas
© 2004-2011 Ben Fry and Casey Reas © 2001-2004 Massachusetts Institute of Technology


```
Last login: Thu Sep 27 08:30:16 on console
Sams-MacBook-Air:~ sthongrong$ ls /dev/tty.*
/dev/tty.Bluetooth-Modem        /dev/tty.RN42-21F8-SPP-SPP-1
/dev/tty.Bluetooth-PDA-Sync
Sams-MacBook-Air:~ sthongrong$ ls /dev/tty.*
/dev/tty.Bluetooth-Modem        /dev/tty.RN42-21F8-SPP-SPP-1
/dev/tty.Bluetooth-PDA-Sync     /dev/tty.usbserial-A6007WYv
Sams-MacBook-Air:~ sthongrong$
```

```
// Processing Sketch
// Serial String Reader

import processing.serial.*;

Serial myProt;

int linefeed = 10;

void setup() {
  // list all the available serial ports
  println(Serial.list());

}

void draw() {

}
```

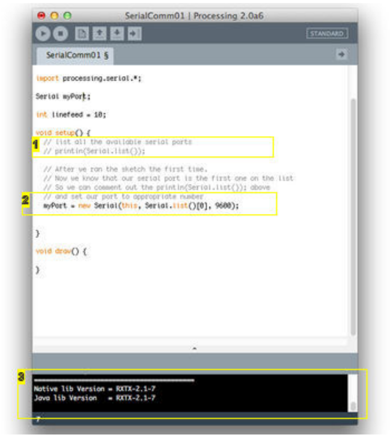
```
Stable Library
=========================================
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
[0] "/dev/tty.usbserial-A6007WYv"
[1] "/dev/cu.usbserial-A6007WYv"
[2] "/dev/tty.RN42-21F8-SPP-SPP-1"
[3] "/dev/cu.RN42-21F8-SPP-SPP-1"
[4] "/dev/tty.Bluetooth-Modem"
[5] "/dev/cu.Bluetooth-Modem"
[6] "/dev/tty.Bluetooth-PDA-Sync"
[7] "/dev/cu.Bluetooth-PDA-Sync"
```

Image Notes
1. List of serial ports available to use after we run the sketch above.


```
// Processing Sketch
// Serial String Reader

import processing.serial.*;

Serial myProt;

int linefeed = 10;

void setup() {
  // list all the available serial ports
  println(Serial.list());

}

void draw() {

}
```

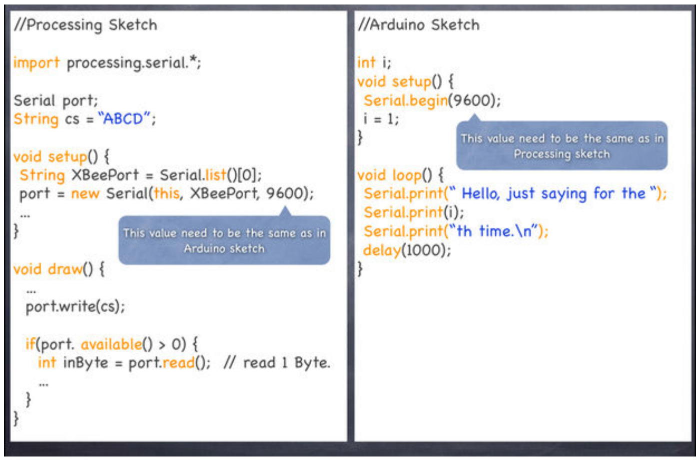
```
Done Saving.
Stable Library
=========================================
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
[0] "/dev/tty.usbserial-A6007WYv"
[1] "/dev/cu.usbserial-A6007WYv"
[2] "/dev/tty.RN42-21F8-SPP-SPP-1"
[3] "/dev/cu.RN42-21F8-SPP-SPP-1"
[4] "/dev/tty.Bluetooth-Modem"
[5] "/dev/cu.Bluetooth-Modem"
[6] "/dev/tty.Bluetooth-PDA-Sync"
[7] "/dev/cu.Bluetooth-PDA-Sync"
```

Image Notes
1. We only use this line once, to see the list of serial port available for Processing to be used.

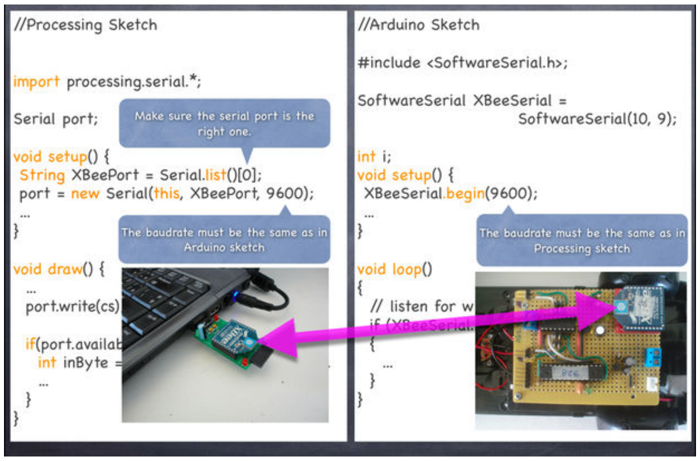

```
import processing.serial.*;

Serial myPort;

int linefeed = 10;

void setup() {
  // list all the available serial ports
  // println(Serial.list());

  // After we ran the sketch the first time.
  // Now we know that our serial port is the first one on the list
  // So we can comment out the println(Serial.list()); above
  // and set our port to appropriate number
  myPort = new Serial(this, Serial.list()[0], 9600);

}

void draw() {

}
```

```
=========================================
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
```

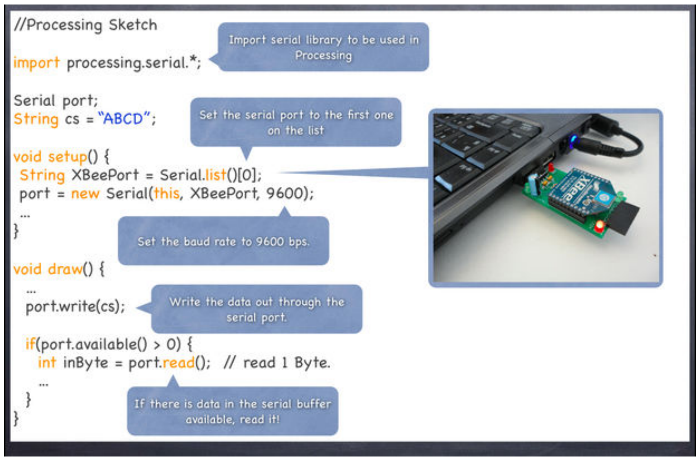
Image Notes
1. We comment out this line after found out about the serial port.
2. Set the Serial port to the first one on the list, Serial.list()[0], and set the speed to 9600 (bps)
3. If we run the code above. We will see these lines.
The Serial Port is ready!


```
//Processing Sketch

import processing.serial.*;

Serial port;
String cs = "ABCD";

void setup() {
 String XBeePort = Serial.list()[0];
 port = new Serial(this, XBeePort, 9600);
 ...
}
void draw() {
 ...
 port.write(cs);

 if(port. available() > 0) {
   int inByte = port.read();  // read 1 Byte.
   ...
 }
}
```

```
//Arduino Sketch

int i;
void setup() {
 Serial.begin(9600);
 i = 1;
}

void loop() {
 Serial.print(" Hello, just saying for the ");
 Serial.print(i);
 Serial.print("th time.\n");
 delay(1000);
}
```
This value need to be the same as in Arduino sketch
This value need to be the same as in Processing sketch


```
//Processing Sketch

import processing.serial.*;

Serial port;

void setup() {
 String XBeePort = Serial.list()[0];
 port = new Serial(this, XBeePort, 9600);
 ...
}
void draw() {
 ...
 port.write(cs)

 if(port.availab
   int inByte =
   ...
 }
}
```

```
//Arduino Sketch

#include <SoftwareSerial.h>;

SoftwareSerial XBeeSerial =
                    SoftwareSerial(10, 9);

int i;
void setup() {
 XBeeSerial.begin(9600);
 ...
}
void loop()
{
 // listen for w
 if (XBeeSerial.
 {
   ...
 }
}
```
Make sure the serial port is the right one.
The baudrate must be the same as in Arduino sketch
The baudrate must be the same as in Processing sketch



```
//Processing Sketch

import processing.serial.*;

Serial port;
String cs = "ABCD";

void setup() {
 String XBeePort = Serial.list()[0];
 port = new Serial(this, XBeePort, 9600);
 ...
}

void draw() {
  ...
  port.write(cs);

  if(port.available() > 0) {
    int inByte = port.read();  // read 1 Byte.
    ...
  }
}
```

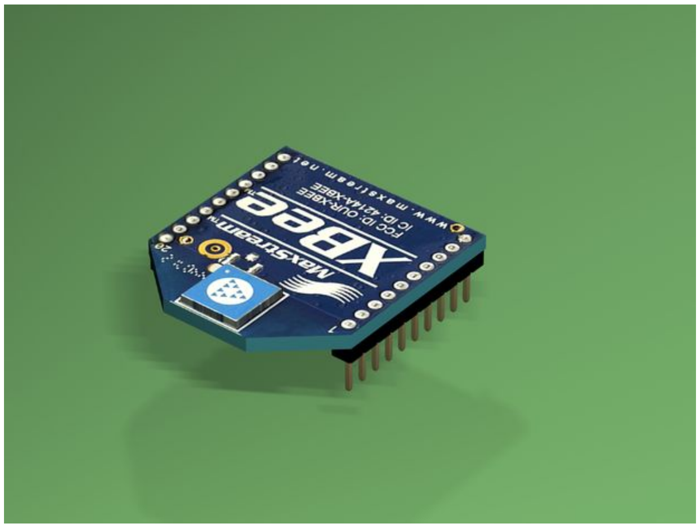
Import serial library to be used in Processing
Set the serial port to the first one on the list
Set the baud rate to 9600 bps.
Write the data out through the serial port.
If there is data in the serial buffer available, read it!

Step 8: Brief Introduction to XBee Module

XBee

- IEEE802.15.4 Network Protocol Wireless Communication Module
- Fast Point-To-Multipoint or Peer-To-Peer Networking
- Built by Digi to the ZigBee/802.15.4 Standard
- Longer range than Bluetooth
- Lower Power Consumption than WIFI
- Can directly talk to micro-controller
- XBee requires 3.3V power supply
- Use Serial Communication to send/receive data
- Requires USB to Serial Converter (i.e FTDI cable, xB-Buddy) to talk to PC or Mac
- XBee module needs adapter board to convert the pins spacing from 2mm to 2.54mm (0.1")
- Need two XBee modules to communicate
- Range ~100 ft. indoor, ~300 ft. outdoor (XBee Series 1)

We must configure XBee modules before using them!



Step 9: Install CoolTerm for MacOS

Install CoolTern for MacOS

Go to http://freeware.the-meiers.org to download CoolTerm.

Scroll down until you see CoolTerm, Then, click the version suitable for your computer. (We were using CoolTermMac)

Drag the CoolTermMac folder from Downloads folder to Applications folder.

CoolTerm icon will appear on the Launchpad automatically.

Click at CoolTerm icon to open the app.

Now, we are ready to start configuring the XBee module. But we need to set up the Terminal (CoolTerm) to be ready for the task:

Set Baudrate to **9600** ,
Data Bits to **8** ,
Parity to **none** ,
Stop Bits to **1**

Click at “Options” on the menu bar.

Insert xB-Buddy with XBee module installed, to USB port.
Click at “ **Options** ” on the menu bar.

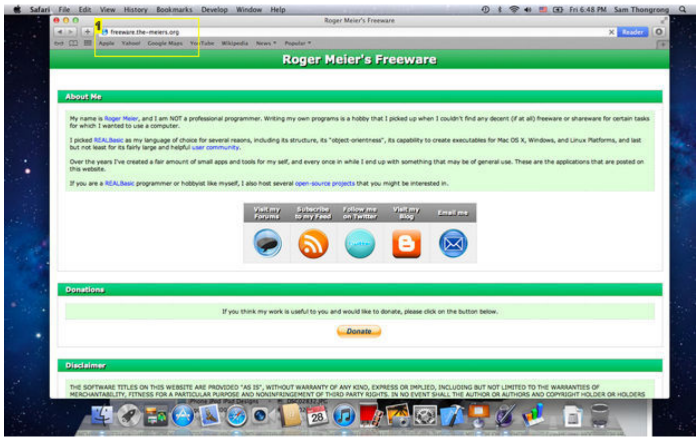
The available xB-Buddy serial port shows up in "**Serial Port Options:** "

If not, just click at **“Re-Scan Serial Ports”** button.

Set Baudrate to 9600,
Data Bits to 8,
Parity to none,
Stop Bits to 1
Click at “Options” on the menu bar.

Select "**Terminal** " from the commands list
Check "**Local Echo** "
Then click OK button to save the setting.

To make a connection between CoolTerm and XBee, Select "**Connect** " button on the Main Menu Bar.



Safari File Edit View History Bookmarks Develop Window Help
Fri 6:48 PM Sam Thongrong
Roger Meier's Freeware
1
freeware.the-meiers.org
Reader
Apple Yahoo! Google Maps YouTube Wikipedia News Popular
Visit my Forums
Subscribe to my Feed
Follow me on Twitter
Visit my Blog
Email me
Donate

Roger Meier's Freeware

About Me

My name is Roger Meier, and I am NOT a professional programmer. Writing my own programs is a hobby that I picked up when I couldn't find any decent (if at all) freeware or shareware for certain tasks for which I wanted to use a computer.

I picked REALBasic as my language of choice for several reaons, including its structure, its "object-orientness", its capability to create executables for Mac OS X, Windows, and Linux Platforms, and last but not least for its fairly large and helpful user community.

Over the years I've created a fair amount of small apps and tools for my self, and every once in while I end up with something that may be of general use. These are the applications that are posted on this website.

If you are a REALBasic programmer or hobbyist like myself, I also host several open-source projects that you might be interested in.

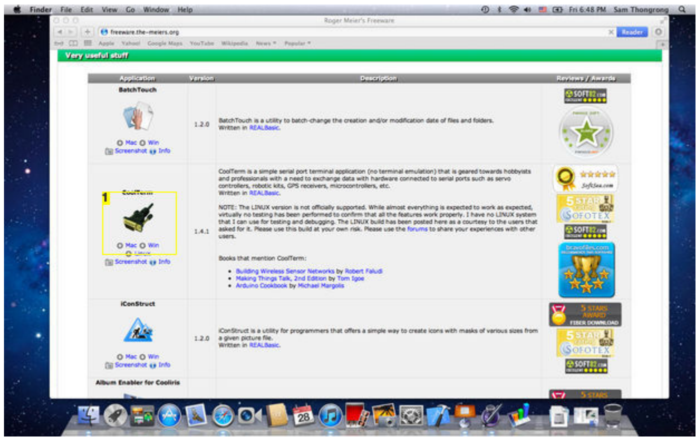
Donations

If you think my work is useful to you and would like to donate, please click on the button below.

Disclaimer

THE SOFTWARE TITLES ON THIS WEBSITE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHOR OR AUTHORS AND COPYRIGHT HOLDER OR HOLDERS

Image Notes
1. Go to http://freeware.the-meiers.org to download CoolTerm.

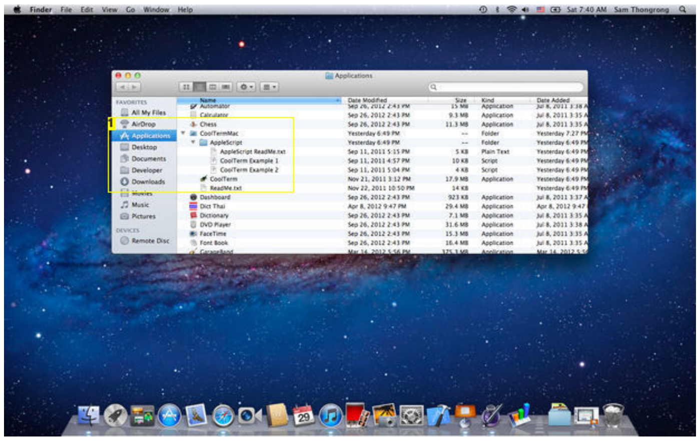


Finder File Edit View Go Window Help
Fri 6:48 PM Sam Thongrong
Roger Meier's Freeware
freeware.the-meiers.org
Reader
1

Very useful stuff

Application	Version	Description	Reviews / Awards
BatchTouch Mac Win Screenshot Info	1.2.0	BatchTouch is a utility to batch-change the creation and/or modification date of files and folders. Written in REALBasic.	
CoolTerm Mac Win Linux Screenshot Info	1.4.1	CoolTerm is a simple serial port terminal application (no terminal emulation) that is geared towards hobbyists and professionals with a need to exchange data with hardware connected to serial ports such as servo controllers, robotic kits, GPS receivers, microcontrollers, etc. Written in REALBasic. NOTE: The LINUX version is not officially supported. While almost everything is expected to work as expected, virtually no testing has been performed to confirm that all the features work properly. I have no LINUX system that I can use for testing and debugging. The LINUX build has been posted here as a courtesy to the users that asked for it. Please use this build at your own risk. Please use the forums to share your experiences with other users. Books that mention CoolTerm: • Building Wireless Sensor Networks by Robert Faludi • Making Things Talk, 2nd Edition by Tom Igoe • Arduino Cookbook by Michael Margolis	
iConStruct Mac Win Screenshot Info	1.2.0	iConStruct is a utility for programmers that offers a simple way to create icons with masks of various sizes from a given picture file. Written in REALBasic.	
Album Enabler for Cooliris			

Image Notes
1. Scroll down until you see CoolTerm, Then, click the version suitable for your computer. (We were using CoolTermMac)



Name	Date Modified	Size	Kind	Date Added
Automator	Sep 26, 2012 2:43 PM	15 MB	Application	Jul 8, 2011 3:38 A
Calculator	Sep 26, 2012 2:43 PM	9.3 MB	Application	Jul 8, 2011 3:35 A
Chess	Sep 26, 2012 2:43 PM	11.3 MB	Application	Jul 8, 2011 3:35 A
CoolTermMac	Yesterday 6:49 PM	--	Folder	Yesterday 7:27 PM
AppleScript	Yesterday 6:49 PM	--	Folder	Yesterday 6:49 PM
AppleScript ReadMe.txt	Sep 11, 2011 5:15 PM	5 KB	Plain Text	Yesterday 6:49 PM
CoolTerm Example 1	Sep 11, 2011 4:57 PM	10 KB	Script	Yesterday 6:49 PM
CoolTerm Example 2	Sep 11, 2011 5:04 PM	4 KB	Script	Yesterday 6:49 PM
CoolTerm	Nov 21, 2011 3:12 PM	17.9 MB	Application	Yesterday 6:49 PM
ReadMe.txt	Nov 22, 2011 10:50 PM	14 KB		Yesterday 6:49 PM
Dashboard	Sep 26, 2012 2:43 PM	923 KB	Application	Jul 8, 2011 3:37 A
Dict Thai	Apr 8, 2012 9:47 PM	29.4 MB	Application	Apr 8, 2012 9:47
Dictionary	Sep 26, 2012 2:43 PM	7.1 MB	Application	Jul 8, 2011 3:35 A
DVD Player	Sep 26, 2012 2:43 PM	31.6 MB	Application	Jul 8, 2011 3:38 A
FaceTime	Sep 26, 2012 2:43 PM	15.3 MB	Application	Jul 8, 2011 3:35 A
Font Book	Sep 26, 2012 2:43 PM	16.4 MB	Application	Jul 8, 2011 3:35 A
GarageBand	Mar 14, 2012 5:56 PM	375.1 MB	Application	Mar 14, 2012 5:5

Image Notes
1. Drag the CoolTermMac folder from Downloads folder to Applications folder.

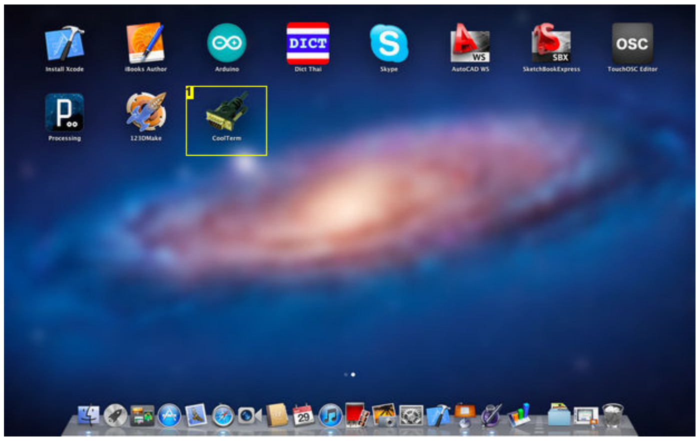


Image Notes
1. CoolTerm icon will appear on the Launchpad automatically.
Click at the icon to start the app.

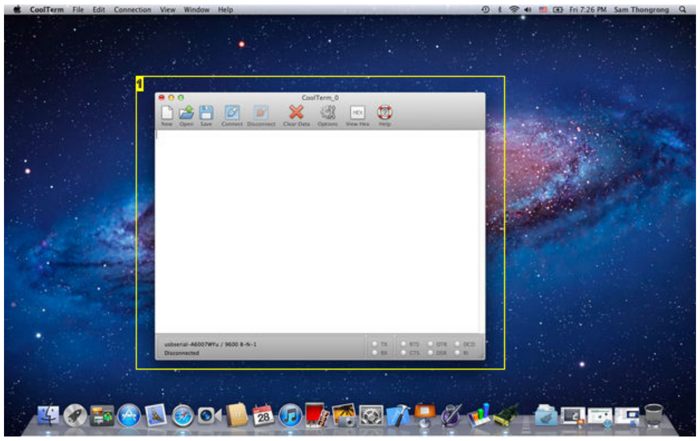


Image Notes
1. Now, we are ready to start configuring the XBee module.

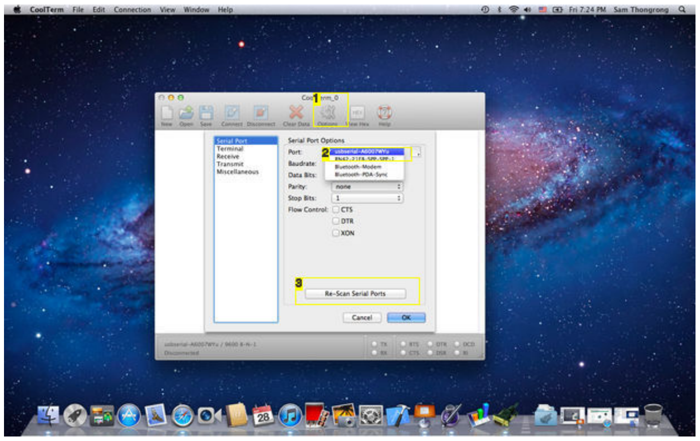
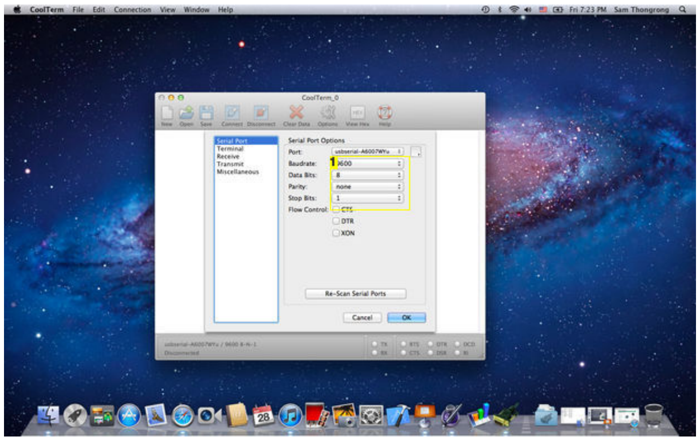


Image Notes
1. Insert xB-Buddy with XBee module installed, to USB port.
Click at âÂ€ÂœOptionsâÂ€Â• on the menu bar.
2. The available xB-Buddy serial port shows up here.
3. If not, just click at âÂ€ÂœRe-Scan Serial PortsâÂ€Â• button.



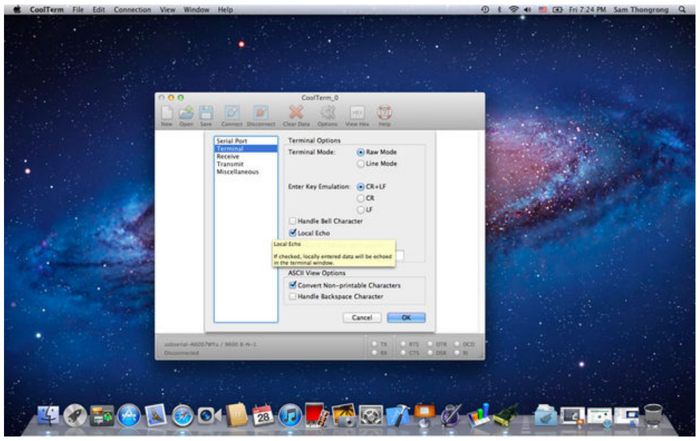


Image Notes
1. Set Baudrate to 9600,
Data Bits to 8,
Parity to none,
Stop Bits to 1
Click at âÂ€ÂœOptionsâÂ€Â• on the menu bar.

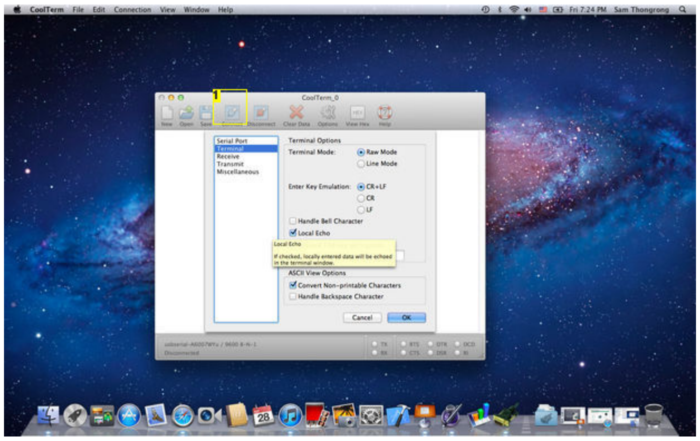


Image Notes
1. Click "Connect" to talke to XBee.

Step 10: XBee Configuration

XBee Configuration

Insert xB-Buddy with XBee module (XBee modules used in the presentation were Serial 1, different method of setting is needed if using XBee Pro) installed, to USB port. Click at **“Options”** button on the menu bar.

The available xB-Buddy serial port shows up on the Serial Port: list. If not, just click at **“Re-Scan Serial Ports”** button.
Then click **"OK"** button.

**NOTE: Assumed that the XBee module is newly bought. So we use the default baud rate (9600).
otherwise we have to find out what was the previously set baud rate.**

Set Baudrate to **9600**
Data Bits to **8**
Parity to **none**
Stop Bits to **1**
And select **"Terminal"** from the list on the left-hand window. And click **"OK"**

Check at the checkbox in front of **"Local Echo"** . Click **“OK”** to save the settings, then close the **“Options”** window.

Click at **“Connect”** icon.

Test XBee Modules

Since two XBee modules are required, and each Xbee module need to have it own configuration data.
We are going to configure each XBee module one at a time. We will assign an XBee as "a", and the other as "b".

Steps to Configuration the XBee

While in the CoolTerm App. First, we are going to ask the XBee module for it current configuration.

Type **+++** to enter command mode, CoolTerm will respond with OK

Type **ATID** and press Enter, CoolTerm respond with the four digit number (shown as 3322)

Type **ATMY** and press Enter, CoolTerm respond with one digit number (shown as 4)

Type **ATDH** and press Enter, CoolTerm respond with one digit number (Shown as 0)

Type **ATDL** and press Enter, CoolTerm respond with one digit number (Shown as 2)

Type **ATBD** and press Enter, CoolTerm respond with one digit number (Shown as 3)

Repeat the steps above to to find out about XBee "b" current configuration.

AT Commands

ATID - PAN ID (Personnel Area Network ID) value can be anything from 0000 to FFFF. The PAN ID on two XBee modules that use for communicating to each other must have same PAN ID number.

ATMY - The unique number use as the address of that XBee module

ATDH - Destination Address High, the first half of the address, we want to talk to

ATDL - Destination Address Low, the address, we will use to locate other XBee module

ATBD - Baud rate, this value need to be the same on both XBee modules, so they can talk to each other.

Complete list of AT commands and XBee configuration can be found here (http://examples.digi.com/wp-content/uploads/2012/07/XBee_802.15.4_AT_Commands.pdf)

Configuring the XBee "a" module:

While in the CoolTerm App.

Type **+++** to enter command mode, CoolTerm will respond with OK

Type **ATID 3322** and press Enter, CoolTerm respond with OK

Type **ATMY 1** and press Enter, CoolTerm respond with OK

Type **ATDH 0** and press Enter, CoolTerm respond with OK

Type **ATDL 2** and press Enter, CoolTerm respond with OK

Type **ATBD 3** and press Enter, CoolTerm respond with OK

Type **ATWR** and press Enter to save the configuration to XBee memory. Now XBee "a" is ready to use.

Configuring the XBee "b" module:

While in the CoolTerm App.

Type **+++** to enter command mode, CoolTerm will respond with OK

Type **ATID 3322** and press Enter, CoolTerm respond with OK

Type **ATMY 2** and press Enter, CoolTerm respond with OK

Type **ATDH 0** and press Enter, CoolTerm respond with OK

Type **ATDL 1** and press Enter, CoolTerm respond with OK

Type **ATBD 3** and press Enter, CoolTerm respond with OK

Type **ATWR** and press Enter to save the configuration to XBee memory. Now XBee "b" is ready to use.

Test XBee modules

After we configure both XBee "a" and Xbee "b". Let's connect XBee "a" and XBee "b" on two different laptops or computers.
Open up CoolTerm on both computer. Make sure that the setting are as mention earlier in this Step.

If you are still in the CoolTerm after you're done with the XBee configuration processes.
Type **ATCN** to get out of the (+++) command mode.

Start to type texts on the CoolTerm terminal on both computers!
Viola! Two computers are talking to each other through the Wireless XBee Serial Comm. Port.

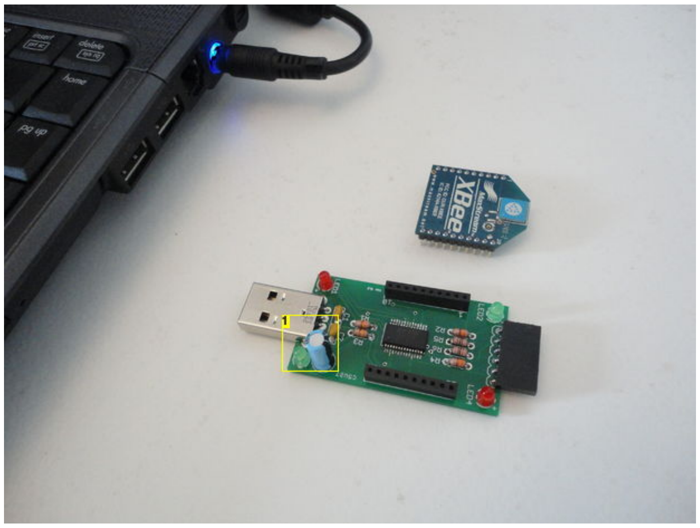


Image Notes
1. XB-Buddy Advance

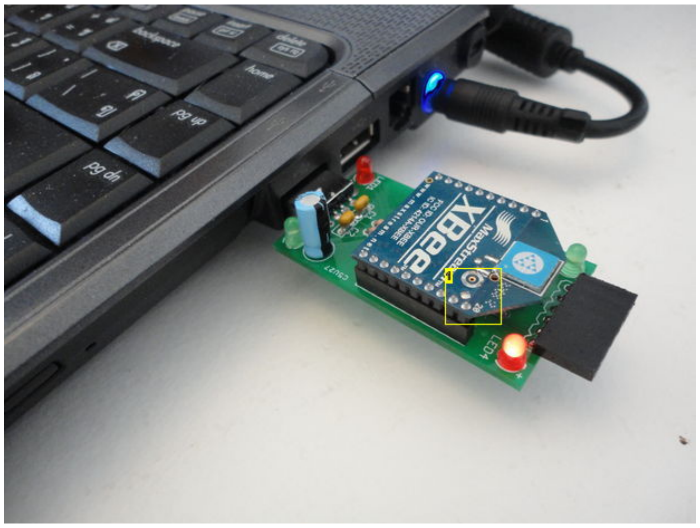
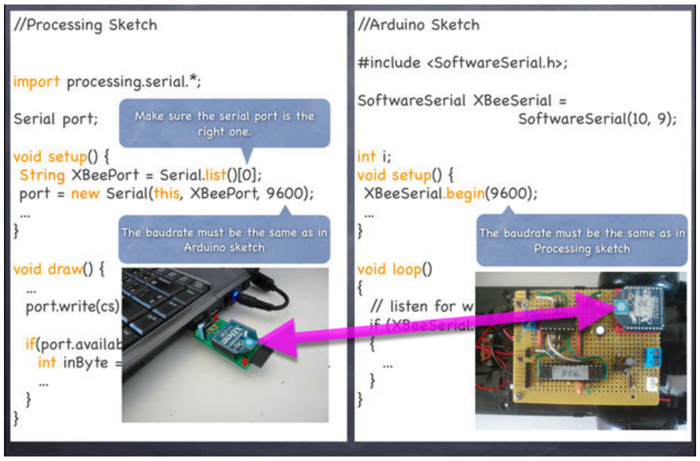


Image Notes
1. XB-Buddy Advance with XBee Module installed



```
//Processing Sketch

import processing.serial.*;

Serial port;

void setup() {
 String XBeePort = Serial.list()[0];
 port = new Serial(this, XBeePort, 9600);
 ...
}

void draw() {
  ...
  port.write(cs)

  if(port.availab
    int inByte =
    ...
  }
}
```

```
//Arduino Sketch

#include <SoftwareSerial.h>;

SoftwareSerial XBeeSerial =
                         SoftwareSerial(10, 9);

int i;
void setup() {
 XBeeSerial.begin(9600);
 ...
}

void loop()
{
  // listen for w
  if (XBeeSerial.
  {
    ...
  }
}
```

Make sure the serial port is the right one.
The baudrate must be the same as in Arduino sketch
The baudrate must be the same as in Processing sketch

Step 11: Built Arduino and Motors Driver PCB

In this project, I built the receiver PCB as all-in one board, meaning that all the component are on the same single board, Arduino compatible, XBee module pins converter board, and motors Driver board.
The whole board will use the power supply from ï¿¼7.5V from the existing battery enclosure. The receiver board was also equipped with ï¿¼3.3V regulator for XBee module, and ï¿¼5V regulator for Arduino Clone.

XBee module talks to Arduino via pin 10 and 9 using SoftwareSerial library.

ï¿¼Arduino Clone
I built the Arduino compatible board exactly as in this instructables - Build "The RevIO" (Arduino Clone) My Way .

ï¿¼XBee Breakout board and XBee Module
XBee module alone could not be connected directly to micro controller or to PC. Because of the pins spacing on the XBee module (2mm) is not compatible to standard 2.5mm spacing PCB. We need to put XBee module to breakout board to connect XBee module to micro controller. In this presentation, I use XB-Buddy basic kit for this purpose.

And to connect XBee module to communicate to PC (to talk to Processing serial) we are also need USB to Serial converter to make it works. So we also need breakout board with FTDI232 IC as the converter. In this presentation I used XB-Buddy Kit for this purpose.

ï¿¼Motors Driver
ï¿¼I built the Motors Driver board similar to the way this Instuctables - Control Your Motors with L293D and Arduino . The schematic (image 4) show the connection of the pins from Arduino to Motors Driver IC.

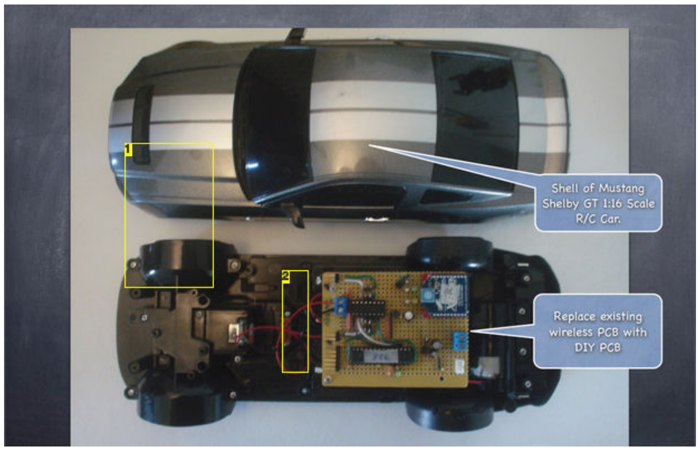
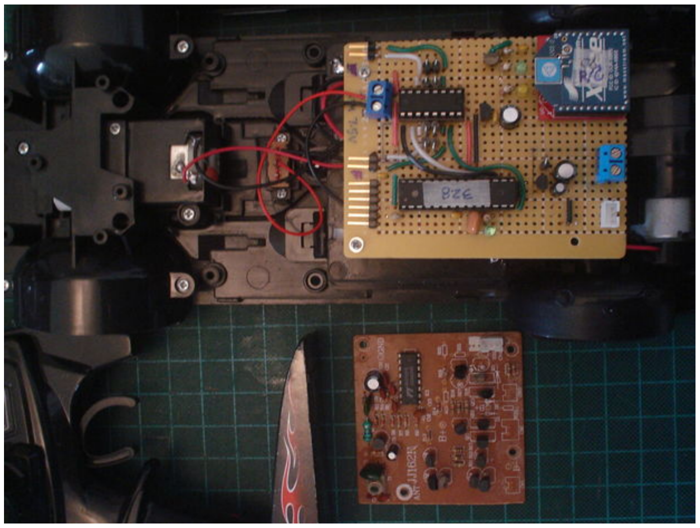

1
2
Shell of Mustang Shelby GT 1:16 Scale R/C Car.
Replace existing wireless PCB with DIY PCB

Image Notes
1. 1:16 Scale R/C car after modification.
2. Battery compartment located underneath the PCB, and can be accessed form the outside underneath the chassis.



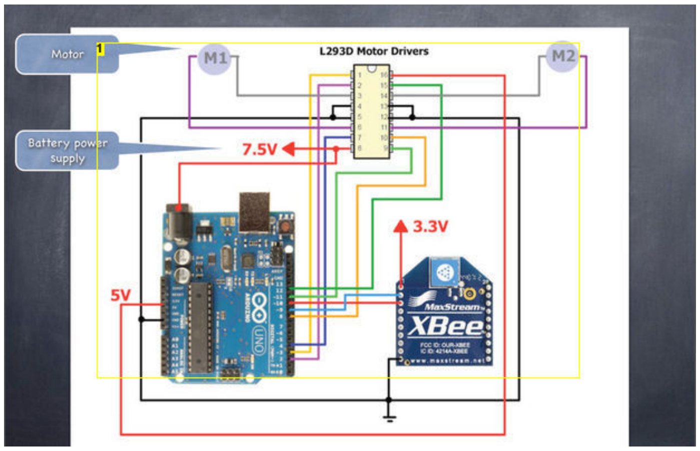


Image Notes
1. Basic Diagram of connection between Arduino, XBee Module and L293D Motors Driver IC

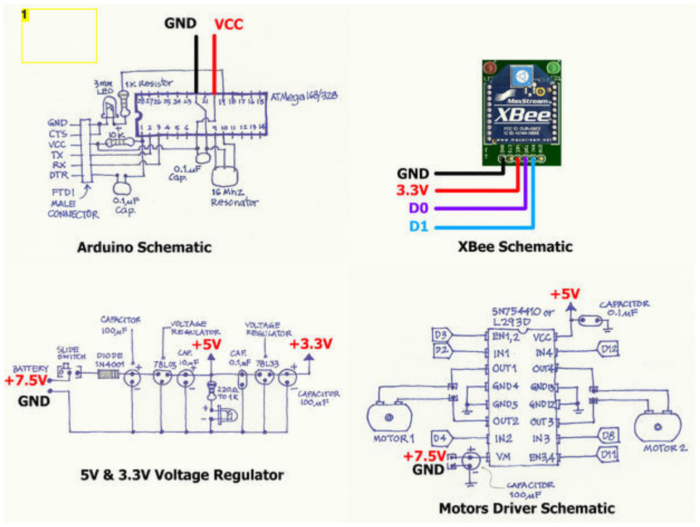
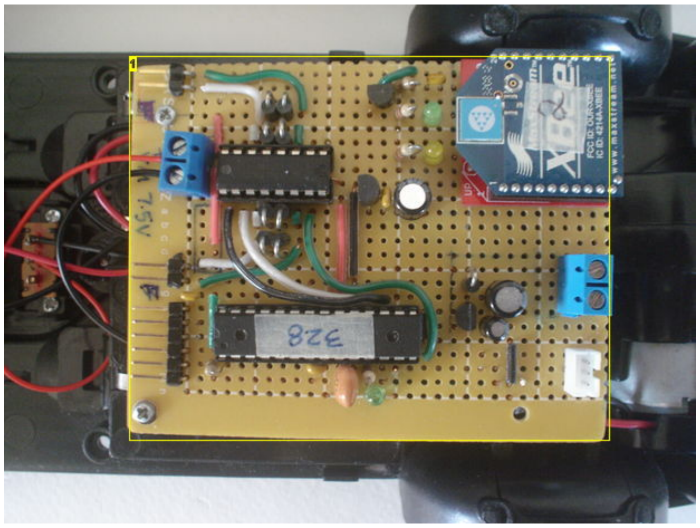


Image Notes
1. Actual built schematic of the Arduino Compatible, XBee Module, and Modtors Driver PCB.



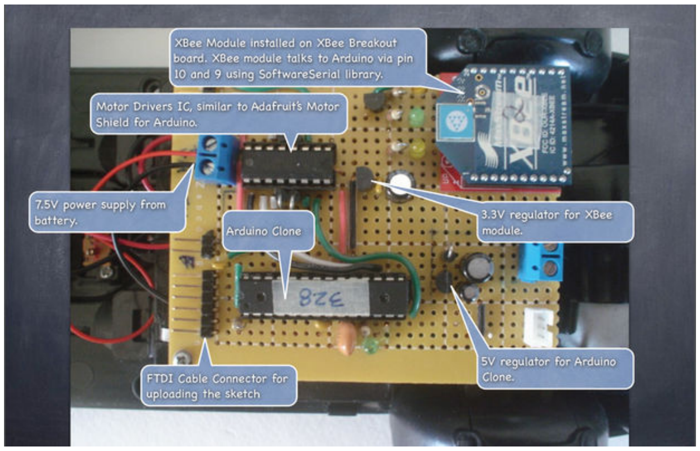

XBee Module installed on XBee Breakout board. XBee module talks to Arduino via pin 10 and 9 using SoftwareSerial library.
Motor Drivers IC, similar to Adafruit's Motor Shield for Arduino.
7.5V power supply from battery.
Arduino Clone
3.3V regulator for XBee module.
5V regulator for Arduino Clone.
FTDI Cable Connector for uploading the sketch

Image Notes
1. Actual built DIY PCB.

Step 12: Arduino Sketch

Arduino Sketch Explain

The explanation is included in the sketch display as the comment in bold typeface.

// Include SoftwareSerial library
#include <SoftwareSerial.h>

// Set XBee module to use SoftwareSerial Library, and set the Arduino pins to be used as TX (10) and RX (9)
SoftwareSerial XBeeSerial = SoftwareSerial(10, 9);

// Set the Arduino pins that connected to L293D - Motor Drivers for Left and Right Turn
#define TURN_ENABLE_PIN 3 // use PWM for this pin
#define LEFT_PIN 4
#define RIGHT_PIN 2

// Set the Arduino pins that connected to L293D - Motor Drivers for Forward and Backward
#define DRIVE_ENABLE_PIN 11 // use PWM for this pin
#define BACKWARD_PIN 8
#define FORWARD_PIN 12

// Input Commands
#define BEGIN_COMMAND 0x7F // decimal = 127, binary = 0111 1111
#define FORWARD 0x1 // decimal = 1, binary = 0000 0001
#define BACKWARD 0x2 // decimal = 2, binary = 0000 0010
#define LEFT 0x4 // decimal = 4, binary = 0000 0100
#define RIGHT 0x8 // decimal = 2, binary = 0000 1000

// Set command variable to be two bytes array,
// command[0] = turn left or right, and drive forward or backward
// command[1] = speed.

int command[2];

void setup() {

// Set XBee SoftwareSerial baud rate to 9600 bps
XBeeSerial.begin(9600);

// Assigns Arduino motors control pins as the output
pinMode(TURN_ENABLE_PIN, OUTPUT);
pinMode(LEFT_PIN, OUTPUT);
pinMode(RIGHT_PIN, OUTPUT);

pinMode(DRIVE_ENABLE_PIN, OUTPUT);
pinMode(BACKWARD_PIN, OUTPUT);
pinMode(FORWARD_PIN, OUTPUT);
}

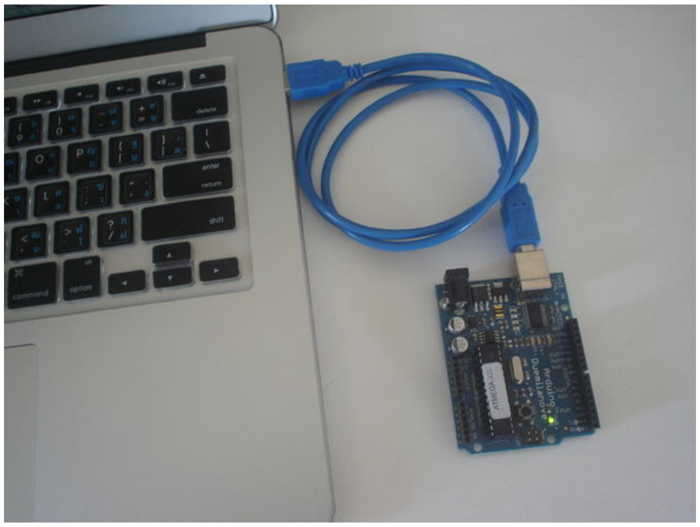
void loop() {

// XBee SoftwareSerial listening for commands from Processing GUI,
// by set the condition to see if there is a data in the serial buffer
if(XBeeSerial.available() > 0) {

// and if the readCommand() functions return value more than 0,
// then executeCommand()
if(readCommand() > 0) {

```
executeCommand();
}
}
}

// Read Command() Method,
// return integer value of 1,
// if there are three bytes begin with 0x7F in the serial buffer
// or return integer value of 0, otherwise.
//
int readCommand() {

int b = XBeeSerial.read();

if(b == BEGIN_COMMAND) { // BEGIN_COMMAND = 0x7F
command[0] = readByte(); // command[0] is either 0x1, 0x2, 0x4, or 0x8
command[1] = readByte(); // command[1] is the speed varies from 0 - 6

return 1;
} else {

return 0;
}
}

int readByte() {

while (true) {
if(XBeeSerial.available() > 0) {
return XBeeSerial.read();
}
}
}

// Translate the commands receive from Processing GUI
// and turn the motor control pins ON/OFF

void executeCommand() {

int c = command[0];
int speed = command[1];

// Control Forward & Backward
// DRIVE_ENABLE_PIN = 11
// BACKWARD_PIN = 8
// FORWARD_PIN = 12
digitalWrite(DRIVE_ENABLE_PIN, LOW);

// Compare (logical and) first byte of the command array (command[0]) with FORWARD (0x01)
// if the result is TRUE, then set the Motor Controls Pins to drive motor forward
if (c & FORWARD) {
digitalWrite(BACKWARD_PIN, LOW);
digitalWrite(FORWARD_PIN, HIGH);
}

// Compare (logical and) first byte of the command array (command[0]) with BACKWARD (0x02)
// if the result is TRUE, then set the Motor Controls Pins to drive motor backward
//

if (c & BACKWARD) {
digitalWrite(FORWARD_PIN, LOW);
digitalWrite(BACKWARD_PIN, HIGH);
}

// Compare (logical and) first byte of the command array (command[0]) with
// the result of logical or between FORWARD (0x01) and BACKWARD(0x02)
// if the result is TRUE, then set the speed to the second byte of the command array (command[1])
//
if (c & (FORWARD | BACKWARD)) {

analogWrite(DRIVE_ENABLE_PIN, speed);

// Control Left & Right turn
// TURN_ENABLE_PIN = 3
// LEFT_PIN = 4
// RIGHT_PIN = 2 digitalWrite(TURN_ENABLE_PIN, LOW);

if (c & LEFT) {
digitalWrite(RIGHT_PIN, LOW);
digitalWrite(LEFT_PIN, HIGH);
}

if (c & RIGHT) {
digitalWrite(LEFT_PIN, LOW);
digitalWrite(RIGHT_PIN, HIGH);
}
```

```
if (c & (LEFT | RIGHT)) {
digitalWrite(TURN_ENABLE_PIN, HIGH);
}
}
```



Step 13: Processing GUI

Processing GUI Explain

I designed theProcessing GUI to be used with specific model of R/C car, the 1/16 Scale model of Ford Mustang Shelby 2010.
I created a background image by took the top view photo of the R/C car then manipulated the image in Photoshop, so I have exactly the same model of the car in the Processing GUI windows.

I created three pairs of front wheels, for right turn, left turn and straight run.
Also I created the button icon for the speed controls of the vehicle.

I incorporated the Processing GUI to the plain original processing code from "Wireless Robotics Platform: Cheap R/C Vehicle + Arduino + XBee + Processing" by nootropicdesign.com.

ï¿½Right Wheels Controls
Move cursor over right wheel to turn "Right". Front wheels are displayed in right-turn direction.

ï¿½Left Wheels Controls
Move cursor over left wheel to turn "Left". Front wheels are displayed in left-turn direction.
ï¿½ï¿½
ï¿½Speed Controls
Speed varies from 1 (slow) to 6 (fast), when cursor is over the number, the number is highlighted in white. The car move "Forward" with that speed.
When move cursor over R, car move in "Reverse" with the speed of 3.
Move cursor over N, make the car "Stop" moving.

Processing Sketch Explain

//
//In order to communicate between Processing (PC, Laptop) and Arduino the baud rate have to be the same.

//ï¿½Import serial library to be used in Processing
import processing.serial.*;

Serial port;

void setup() {

// Set up the port
String XBeePort = Serial.list()[0];

// Set Baud Rate to 9600 bps
port = new Serial(this, XBeePort, 9600);

// Setup the screen for output.
size(400, 596);

// Assign variable called speed to be 0 (zero)
speed = 0;

// Call method (function) controlGUI() to display Graphics User Interface to
// display GUI on the Processing display windows
controlGUI();
}

void draw() {

```
//ï¿½Displays the background image ( ShelbyGT2010Background.png )
background(carImage);

// Call method (function) speedButtons() to refresh the images of the buttons change according to the user input
speedButtons();

// Call method updateFrontWheel(mouseX, mouseY) to update
// the image of the front wheels if the mouse cursor is moved over
updateFrontWheel(mouseX, mouseY);

// ï¿½Method to reflesh the GUI window, when there is a change in the command.
directionButtons();

// Test run
testRun();
}

// serialEvents() - monitor the input from
// the serial comm. port
void serialEvent(Serial p)
{
int input = p.read();
lastInput = input;
}


/ ============================
// Button Class
// ============================

class Button
{
int x, y;
int w, h;
color basecolor, highlightcolor; color currentcolor;
boolean over = false;
boolean pressed = false;

void pressed() {
if (pressed) {
currentimage = down; }
else if (over) {
currentimage = roll; }
else {
currentimage = base;
}
}

void over() {
if ( overRect(x, y, w, h) ) {
over = true; }
else {
over = false;
}
}

void display() {
image(currentimage, x, y);
}
}


// ============================
// ImageButtons Classes
// ============================
class ImageButtons extends Button
{
PImage base;
PImage roll;
PImage down;
PImage currentimage;

ImageButtons(int ix, int iy, int iw, int ih,
PImage ibase, PImage iroll,
PImage idown) {
x = ix; // image origin X
y = iy; // image origin Y
w = iw; // image width
h = ih; // image height
base = ibase; // base image
roll = iroll; // roll or over image
down = idown; // down or selected image
currentimage = base;
}
```

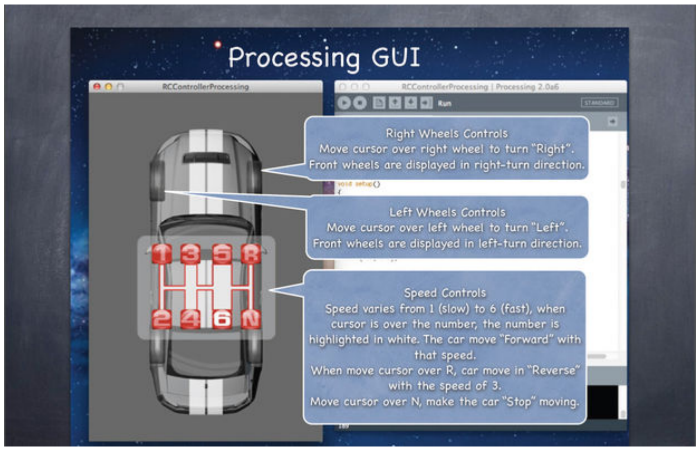
```
void update()
{
over();
pressed();
if (pressed) {
currentimage = down;
} else if (over){
currentimage = roll;
} else {
currentimage = base;
}
}

void over()
{
if ( overRect(x, y, w, h) ) {
over = true;
} else {
over = false;
}
}
void display()
{
image(currentimage, x, y);
}
}
```

// We needed to have at least three images
// to use as an icon- base, roll, and down
// in ImageButtons() method.

```
int buttonX = 34;
int buttonY = 34;
...
void controlGUI() {
carImage = loadImage(
"ShelbyGT2010Background.png");
selected =
loadImage("gearSelected.png");

turnSelected =
loadImage("turnSelected.png");

// one
oneImage = loadImage("one.png");
oneOver = loadImage("oneOver.png");
oneButton = new ImageButtons(108, 258,
buttonX, buttonY,
oneOver, oneImage,
selected);
```

// These last three variables are the placeholder for the images.

```
// two
twoImage = loadImage("two.png");
twoOver = loadImage("twoOver.png");

...

}
```



Processing GUI

Right Wheels Controls
Move cursor over right wheel to turn "Right".
Front wheels are displayed in right-turn direction.

Left Wheels Controls
Move cursor over left wheel to turn "Left".
Front wheels are displayed in left-turn direction.

Speed Controls
Speed varies from 1 (slow) to 6 (fast), when cursor is over the number, the number is highlighted in white. The car move "Forward" with that speed.
When move cursor over R, car move in "Reverse" with the speed of 3.
Move cursor over N, make the car "Stop" moving.

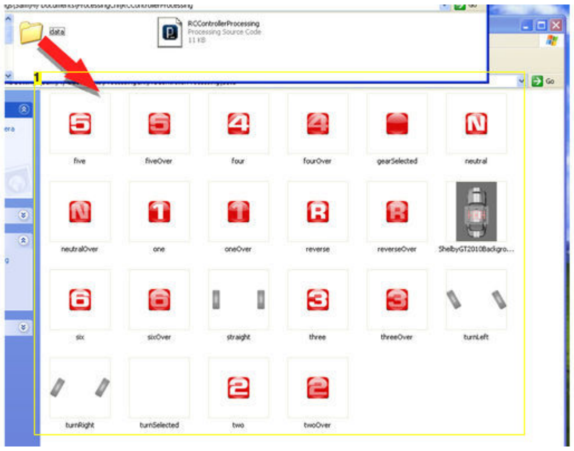
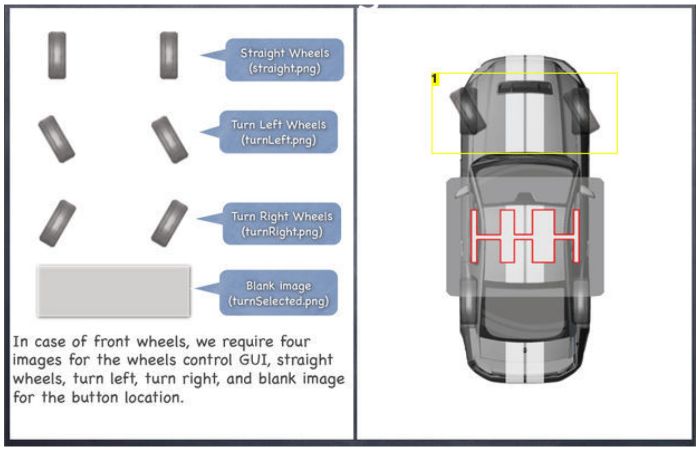


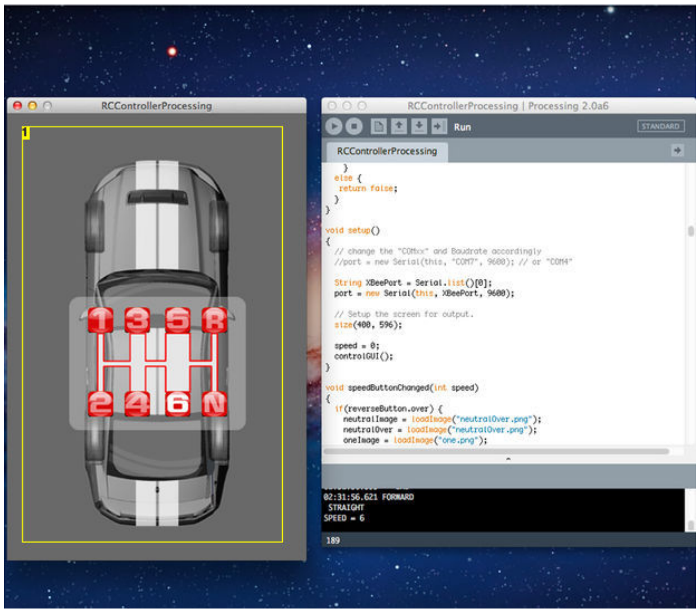
Image Notes
1. All necessary icons, wheels turning icons, and background are resided in the data folder.



Straight Wheels (straight.png)
Turn Left Wheels (turnLeft.png)
Turn Right Wheels (turnRight.png)
Blank image (turnSelected.png)
1

In case of front wheels, we require four images for the wheels control GUI, straight wheels, turn left, turn right, and blank image for the button location.

Image Notes
1. Wheels turning left.

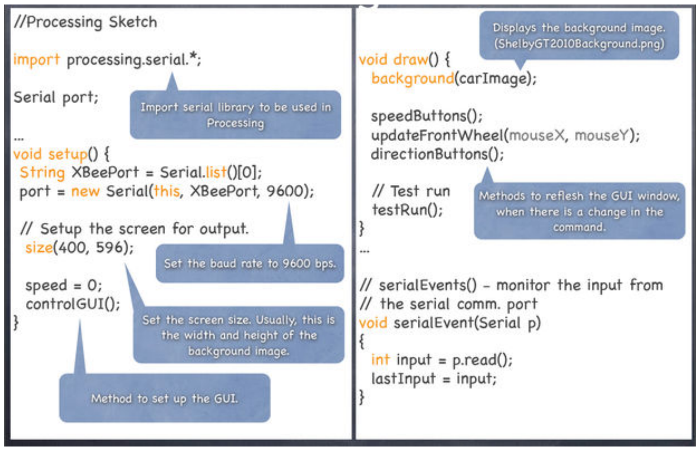


```
  }
  else {
   return false;
  }
}

void setup()
{
  // change the "COMxx" and Baudrate accordingly
  //port = new Serial(this, "COM7", 9600); // or "COM4"

  String XBeePort = Serial.list()[0];
  port = new Serial(this, XBeePort, 9600);

  // Setup the screen for output.
  size(400, 596);

  speed = 0;
  controlGUI();
}

void speedButtonChanged(int speed)
{
  if(reverseButton.over) {
    neutralImage = loadImage("neutralOver.png");
    neutralOver = loadImage("neutralOver.png");
    oneImage = loadImage("one.png");
```

```
02:31:56.621 FORWARD
 STRAIGHT
SPEED = 6
```

189

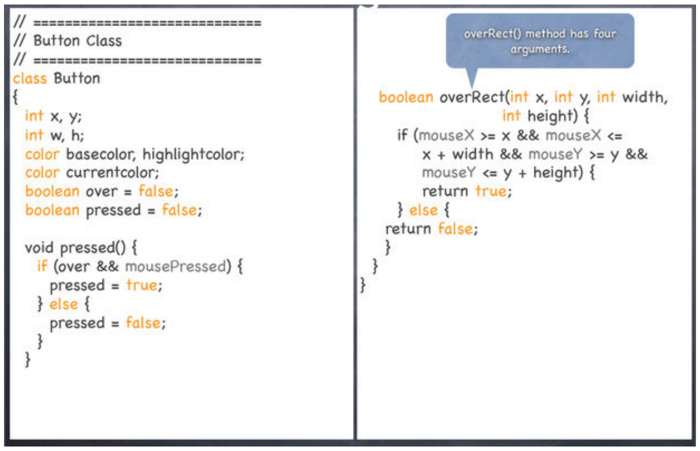
Image Notes
1. GUI in used.



```
//Processing Sketch

import processing.serial.*;

Serial port;

...
void setup() {
 String XBeePort = Serial.list()[0];
 port = new Serial(this, XBeePort, 9600);

 // Setup the screen for output.
  size(400, 596);

  speed = 0;
  controlGUI();
}
```

```
void draw() {
  background(carImage);

  speedButtons();
  updateFrontWheel(mouseX, mouseY);
  directionButtons();

  // Test run
  testRun();
}
...

// serialEvents() - monitor the input from
// the serial comm. port
void serialEvent(Serial p)
{
  int input = p.read();
  lastInput = input;
}
```

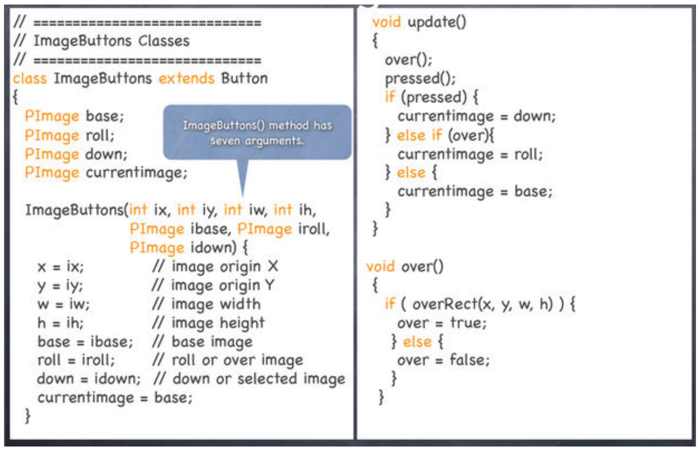
Import serial library to be used in Processing
Set the baud rate to 9600 bps.
Set the screen size. Usually, this is the width and height of the background image.
Method to set up the GUI.
Displays the background image. (ShelbyGT2010Background.png)
Methods to reflesh the GUI window, when there is a change in the command.



```
// ============================
// Button Class
// ============================
class Button
{
  int x, y;
  int w, h;
  color basecolor, highlightcolor;
  color currentcolor;
  boolean over = false;
  boolean pressed = false;

  void pressed() {
    if (over && mousePressed) {
      pressed = true;
    } else {
      pressed = false;
    }
  }
```

```
  boolean overRect(int x, int y, int width,
                   int height) {
    if (mouseX >= x && mouseX <=
        x + width && mouseY >= y &&
        mouseY <= y + height) {
        return true;
    } else {
     return false;
    }
  }
}
```

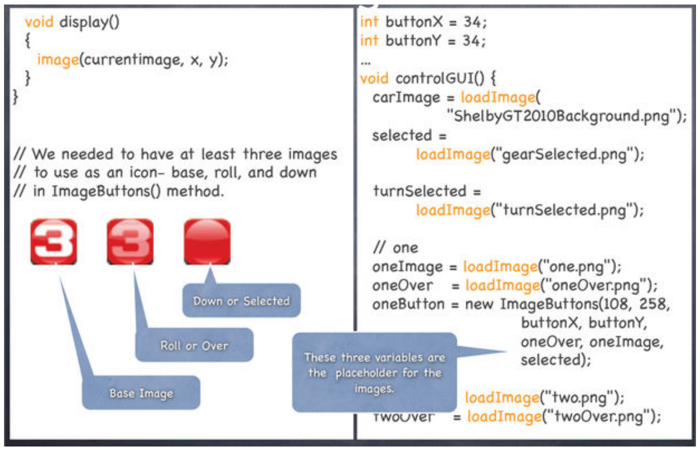
overRect() method has four arguments.



```
// ============================
// ImageButtons Classes
// ============================
class ImageButtons extends Button
{
  PImage base;
  PImage roll;
  PImage down;
  PImage currentimage;

  ImageButtons(int ix, int iy, int iw, int ih,
               PImage ibase, PImage iroll,
               PImage idown) {
    x = ix;          // image origin X
    y = iy;          // image origin Y
    w = iw;          // image width
    h = ih;          // image height
    base = ibase;    // base image
    roll = iroll;    // roll or over image
    down = idown;    // down or selected image
    currentimage = base;
  }
```

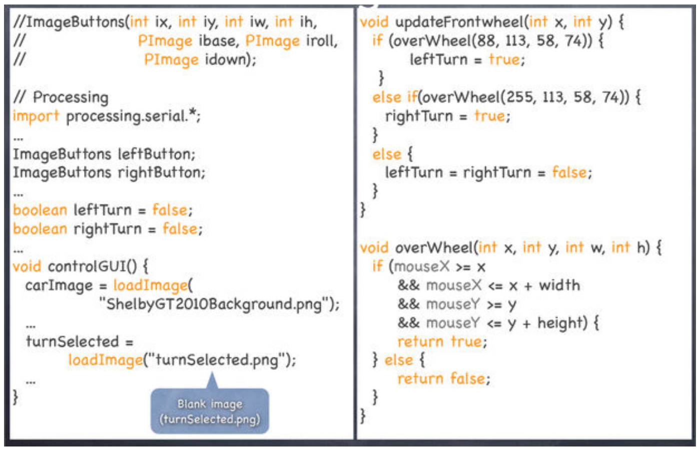
```
  void update()
  {
    over();
    pressed();
    if (pressed) {
      currentimage = down;
    } else if (over){
      currentimage = roll;
    } else {
      currentimage = base;
    }
  }

 void over()
  {
    if ( overRect(x, y, w, h) ) {
      over = true;
    } else {
      over = false;
    }
  }
```

ImageButtons() method has seven arguments.



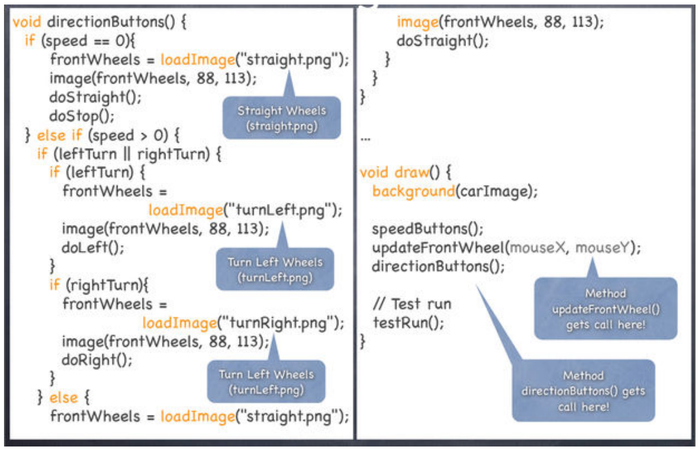
```
  void display()
  {
    image(currentimage, x, y);
  }
}

// We needed to have at least three images
// to use as an icon- base, roll, and down
// in ImageButtons() method.
```

```
int buttonX = 34;
int buttonY = 34;
...
void controlGUI() {
  carImage = loadImage(
          "ShelbyGT2010Background.png");
  selected =
          loadImage("gearSelected.png");

  turnSelected =
          loadImage("turnSelected.png");

  // one
  oneImage = loadImage("one.png");
  oneOver   = loadImage("oneOver.png");
  oneButton = new ImageButtons(108, 258,
                      buttonX, buttonY,
                      oneOver, oneImage,
                      selected);

                loadImage("two.png");
  TwoOver    = loadImage("twoOver.png");
```



```
//ImageButtons(int ix, int iy, int iw, int ih,
//                PImage ibase, PImage iroll,
//                 PImage idown);

// Processing
import processing.serial.*;
...
ImageButtons leftButton;
ImageButtons rightButton;
...
boolean leftTurn = false;
boolean rightTurn = false;
...
void controlGUI() {
  carImage = loadImage(
             "ShelbyGT2010Background.png");
  ...
  turnSelected =
        loadImage("turnSelected.png");
  ...
}
```

```
void updateFrontwheel(int x, int y) {
  if (overWheel(88, 113, 58, 74)) {
        leftTurn = true;
   }
  else if(overWheel(255, 113, 58, 74)) {
    rightTurn = true;
  }
  else {
    leftTurn = rightTurn = false;
  }
}

void overWheel(int x, int y, int w, int h) {
  if (mouseX >= x
      && mouseX <= x + width
      && mouseY >= y
      && mouseY <= y + height) {
      return true;
  } else {
      return false;
  }
}
```



```
void directionButtons() {
  if (speed == 0){
    frontWheels = loadImage("straight.png");
    image(frontWheels, 88, 113);
    doStraight();
    doStop();
  } else if (speed > 0) {
   if (leftTurn || rightTurn) {
    if (leftTurn) {
     frontWheels =
                loadImage("turnLeft.png");
     image(frontWheels, 88, 113);
     doLeft();
    }
    if (rightTurn){
     frontWheels =
               loadImage("turnRight.png");
     image(frontWheels, 88, 113);
     doRight();
    }
   } else {
    frontWheels = loadImage("straight.png");
      image(frontWheels, 88, 113);
      doStraight();
    }
  }
}

...

void draw() {
  background(carImage);

  speedButtons();
  updateFrontWheel(mouseX, mouseY);
  directionButtons();

  // Test run
  testRun();
}
```

Step 14: Demo and References

Live Demo!
I did "Live Demo" at the presentation, But we did not recorded the demo.
(Addition Demo Video will be post as soon as possible.)

References

XBee
Digi XBee Examples & Guides http://examples.digi.com
Arduino Set Up - http://xbee.wikispaces.com/Arduino+Set+up

FTDI
Future Technology Devices International Inc. - http://www.ftdichip.com/index.html

Arduino SoftwareSerial Library
Arduino SoftwareSerial Library - http://www.arduino.cc/en/Reference/SoftwareSerial

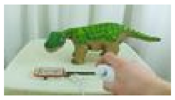
Processing Serial
Processing Serial Library - http://processing.org/reference/libraries/serial/index.html

CoolTerm
CoolTerm App - http://freeware.the-meiers.org

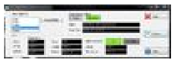
Motors Driver
Control your motors with L293D and Arduino - http://www.instructables.com/id/Control-your-motors-with-L293D-and-Arduino/

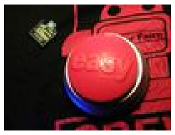
"Wireless Robotics Platform: Cheap R/C Vehicle + Arduino + XBee + Processing" by nootropicdesign.com.

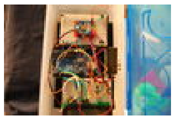
Related Instructables


Remote Control Pleo with Wii Nunchuck by mikhalchuk

Use xbees (series 2) to control a motor by gabriellalevine


Wireless outdoor Arduino weather station with PC logging and Graphs by zmashiah


Make an Easy Button Wireless by nomuse


Wireless Vital-monitoring Armband by Wesley03

Wireless Finger Drum by sath02