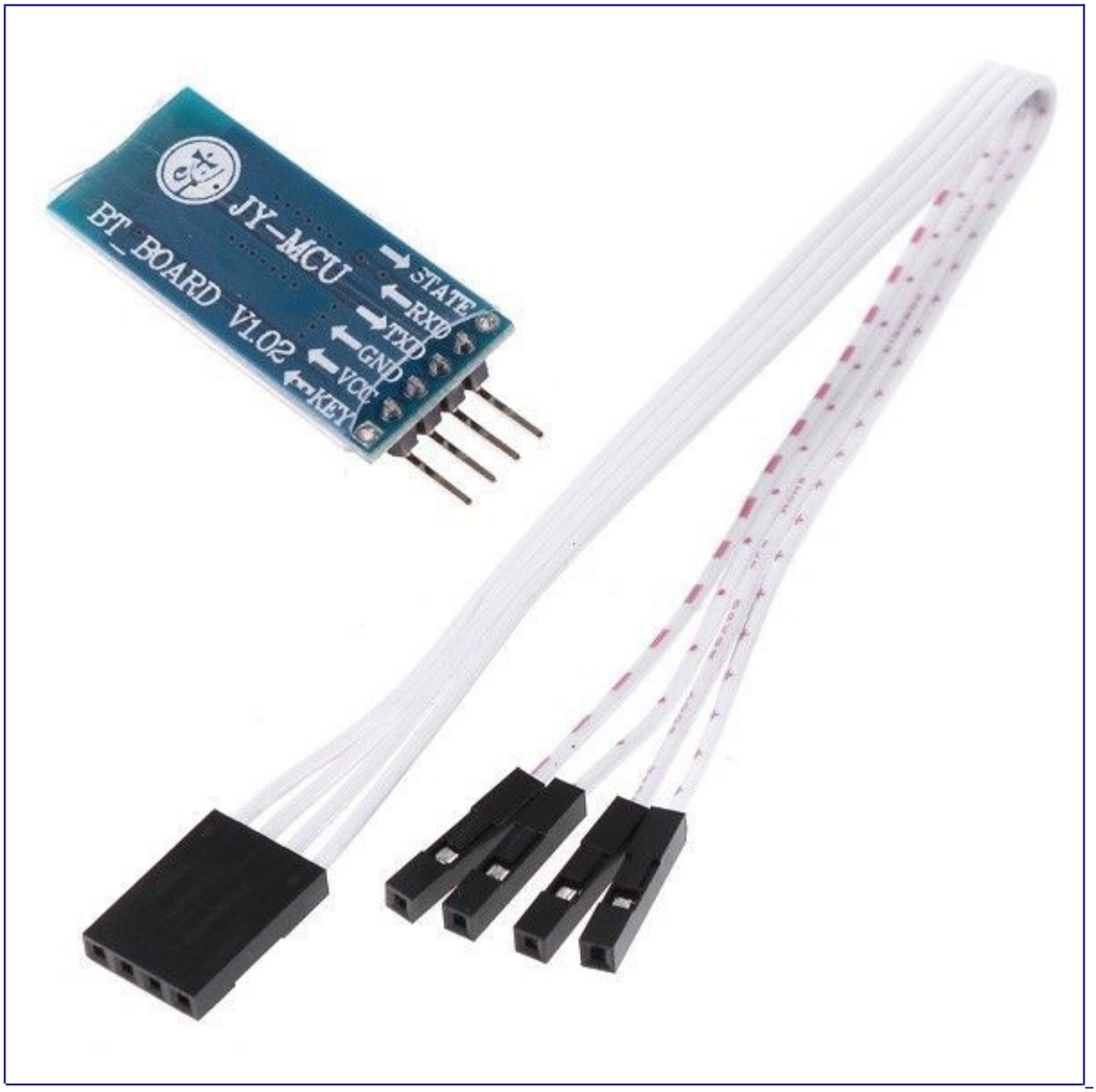
Success Using the JY-MCU (linvor) Bluetooth Module

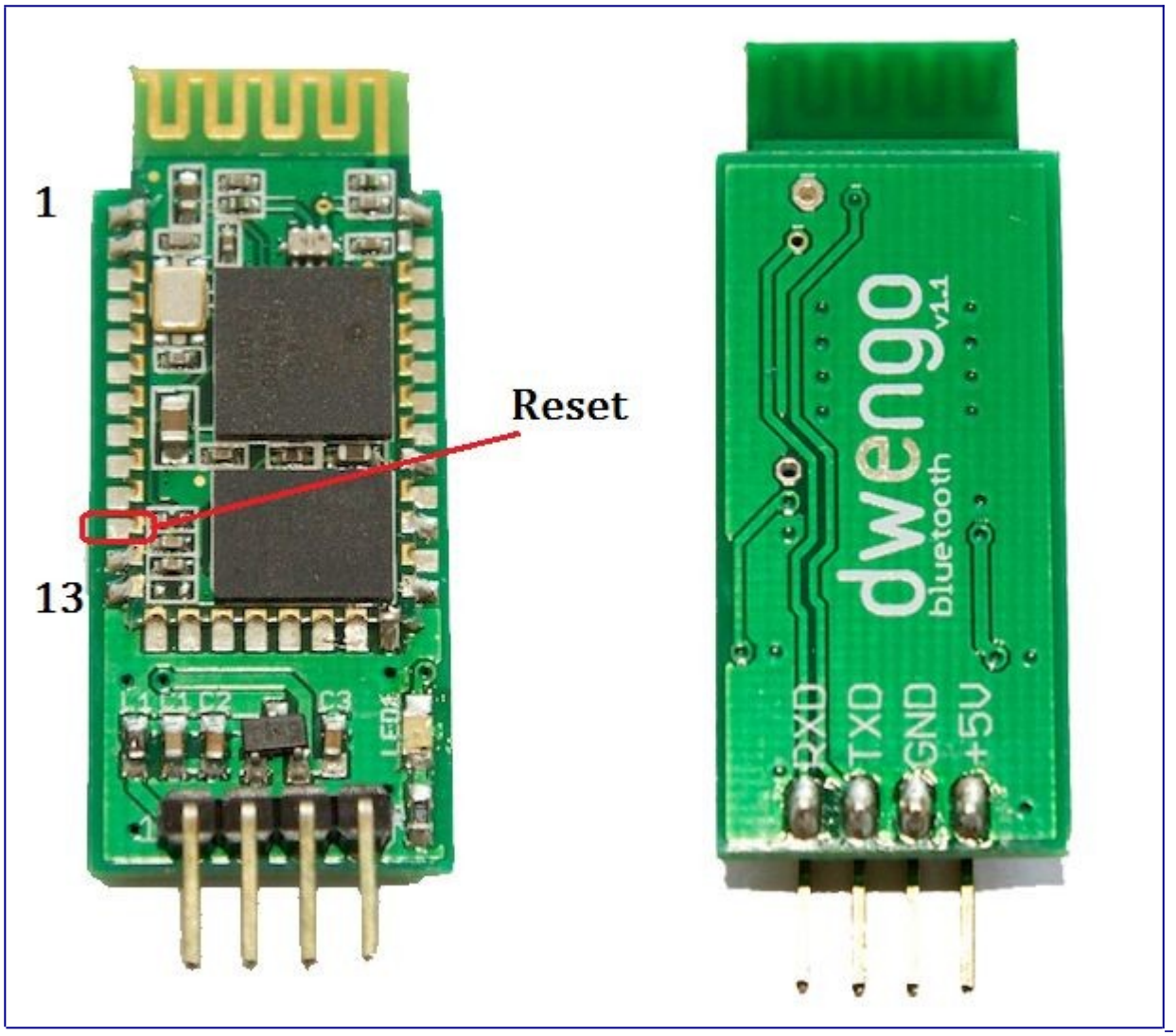


I'm writing this instructable because I have had nightmares getting this cheap, but well built bluetooth module to work. It was worth the challenge to save myself $50 for a simple SPP serial port (RS232 Emulator) service! I will go over the do's and don'ts I discovered while spending weeks with this board, both hardware and software.

I am assuming you are connecting this module to an Arduino or similar MCU development board.

Next up - The Pins...

Step 1: The Pins



The module has 6 pins labeled on the back, but most modules only have 4 of those populated with pogo pins. KEY & STATE seem to be not required, as KEY is used for flashing the device and STATE simply indicates if the device is awake or not. So that leaves only GND, VCC, TXD, RXD. Not shown is Pin 11 which is the RESET pin, resetting the module when pulled LOW.

Some boards have VCC labeled for working voltages up to ~6 volts. These modules DO NOT like anything except 3.3 volts on the VCC line. Also, some forums claim that the device works fine with 5 volt TTL levels. This is also not true and you should use a level converter to 3.3V on the RXD line. I used two resistors as a simple voltage divider to make the TTL level conversion. One 2.2k ohm resistor to ground, connected to a 1k ohm resistor, to the TXD line on the MCU. Connect the RXD pin in between the two resistors for an output of approx 3.4 volts. This about covers the hardware side of the module. As it turns out, the VCC voltage was one of the last things I tried to have full success with the module.

Next up - The Software...

Step 2: The Software

I have only found one library that works in Processing, and that is extrapixel's bluetoothDesktop library, which I will link to below. Others may work, but I had no luck with them. The library has an extensive reference page and is fairly straight forward.

I recommend setting up a softwareSerial virtual port on your Arduino for communicating with this module. Connecting it directly to the RX/TX lines gave me headaches and locked up ports! On the mega2560 I used pin 10 & 9 for RX & TX respectfully. The RX pin varies between different Arduino models, so check the Arduino forums to see which pins your Arduino work with softwareSerial.

The default parameters on the linvor module are: 9600 baud 8 N 1 None. AT commands can be sent to the board only when the module is NOT connected, or when the red LED is flashing. Only a handful of basic AT commands work, unless you have a HC-05 firmware. Linvor version 1.5, also known as HC-06, is a slave only device. Here are the known AT commands for version 1.5...

AT - Response OK
AT+NAMExxxx - Where xxxx is the friendly name of the module
AT+BAUDx - Where x sets the baud rate
(values & baud rates below)
AT+VERSION - Returns the firmware version

AT+PINxxxx - Sets a new pairing code

1——1200 2——2400 3——4800 4——9600
5——19200 6——38400 7——57600 8——115200

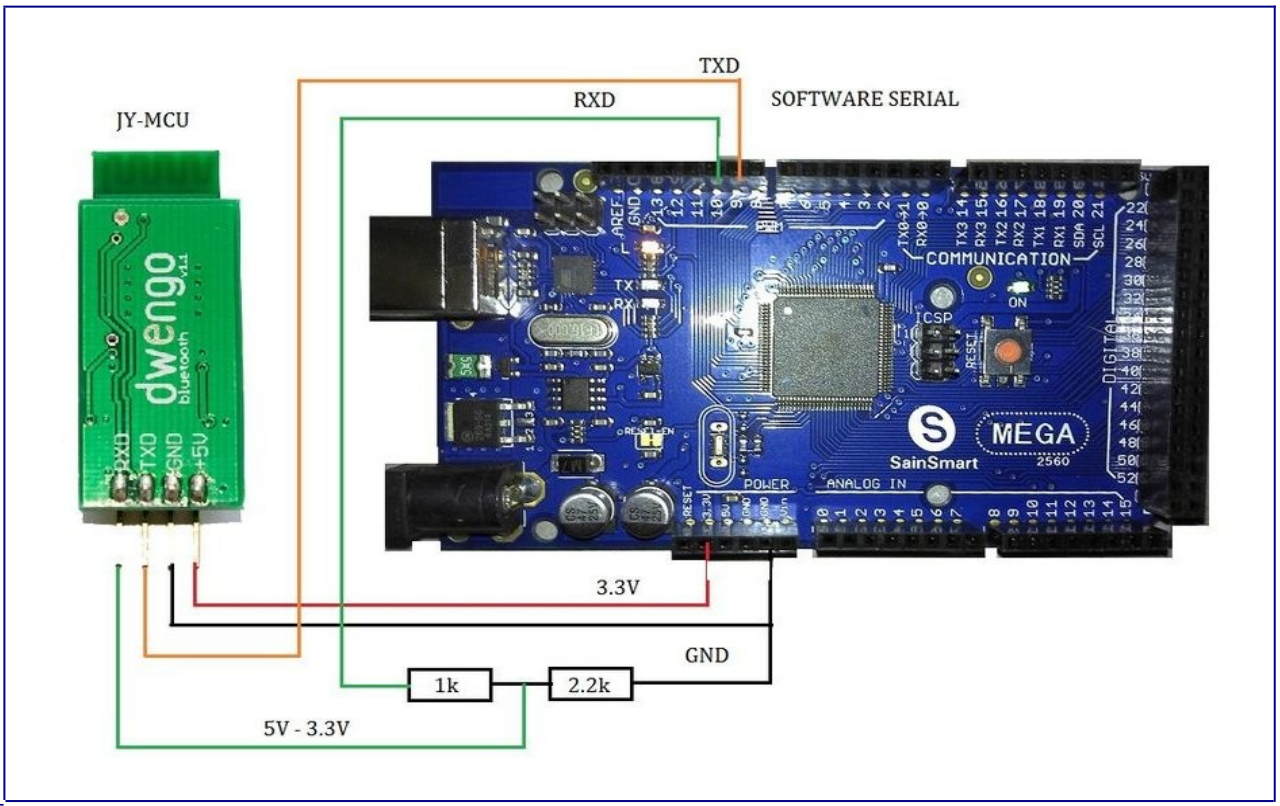
This example code will allow AT commands to be sent via the serial monitor. The bluetoothDesktop library is not required to connect with the BT module...

```
/*********************************************************
*********************************************************
  Created back in the mists of time
  Modified 25 May 2012
  by Tom Igoe
  based on Mikal Hart's example.

  Modified 10 July 2012
  by Peter Timinski
  based on Tom Igoe's changes to Mikal Hart's example.

  This example code is in the public domain.
  Nothing is guaranteed to work so PROCEDE at your OWN RISK!
  Forum: http://arduino.cc/forum/index.php?topic=101452.0
  *********************************************************
  *********************************************************
  */

#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 9); // RX, TX
String command = ""; // Stores response of bluetooth device
                                        // which simply allows \n
between each
                                        // response.
void setup()
{
   // Open serial communications and wait for port to open:
   Serial.begin(9600);
   Serial.println("Type AT commands!");
   // SoftwareSerial "com port" data rate. JY-MCU v1.03 defaults
```

```
to 9600.
   mySerial.begin(9600);
}

void loop()
{
   // Read device output if available.
   if (mySerial.available()) {
     while(mySerial.available()) { // While there is more to be
read, keep reading.
       command += (char)mySerial.read();
     }
     Serial.println(command);
     command = ""; // No repeats
   }

   // Read user input if available.
   if (Serial.available()){
       delay(10); // The DELAY!
       mySerial.write(Serial.read());
   }
}// END loop()
```

Find this library with reference at http://www.extrapixel.ch/processing/bluetoothDesktop/

Step 3: The Correct Library & Connections:



To get into some SPP service serial communication you will need the bluetoothDesktop library for Processing. The files and reference pages can be found here...

http://www.extrapixel.ch/processing/bluetoothDesktop/

Wiring the module for Software Serial:

Yes, use the 3.3 volt output for VCC. This is one of the last details I found to be quarky with this particular module, even though some boards say 5V. The green lines show the TTL level converter via the 2 resistors. This provides a logic level of about 3.45 volts.

Step 4: Services

For this application, we will want to use the SPP serial port service. When you connect/pair with the radio, your computer should make 2 virtual COM ports .. 1 for incoming and the other for outgoing traffic. It is a must that the outgoing port is present. It should also have the SPP serial service listed next to it in your bluetooth settings/ports. Mine says 'Dev B' and I would assume this is true for all of these modules. The next bit of code uses the library and software serial to actually connect with the radio for serial communication.

This makes a simple client/server using SPP service 'Dev B' as is listed next to the outgoing COM port. (found in bluetooth settings / ports). It simply searches for the service

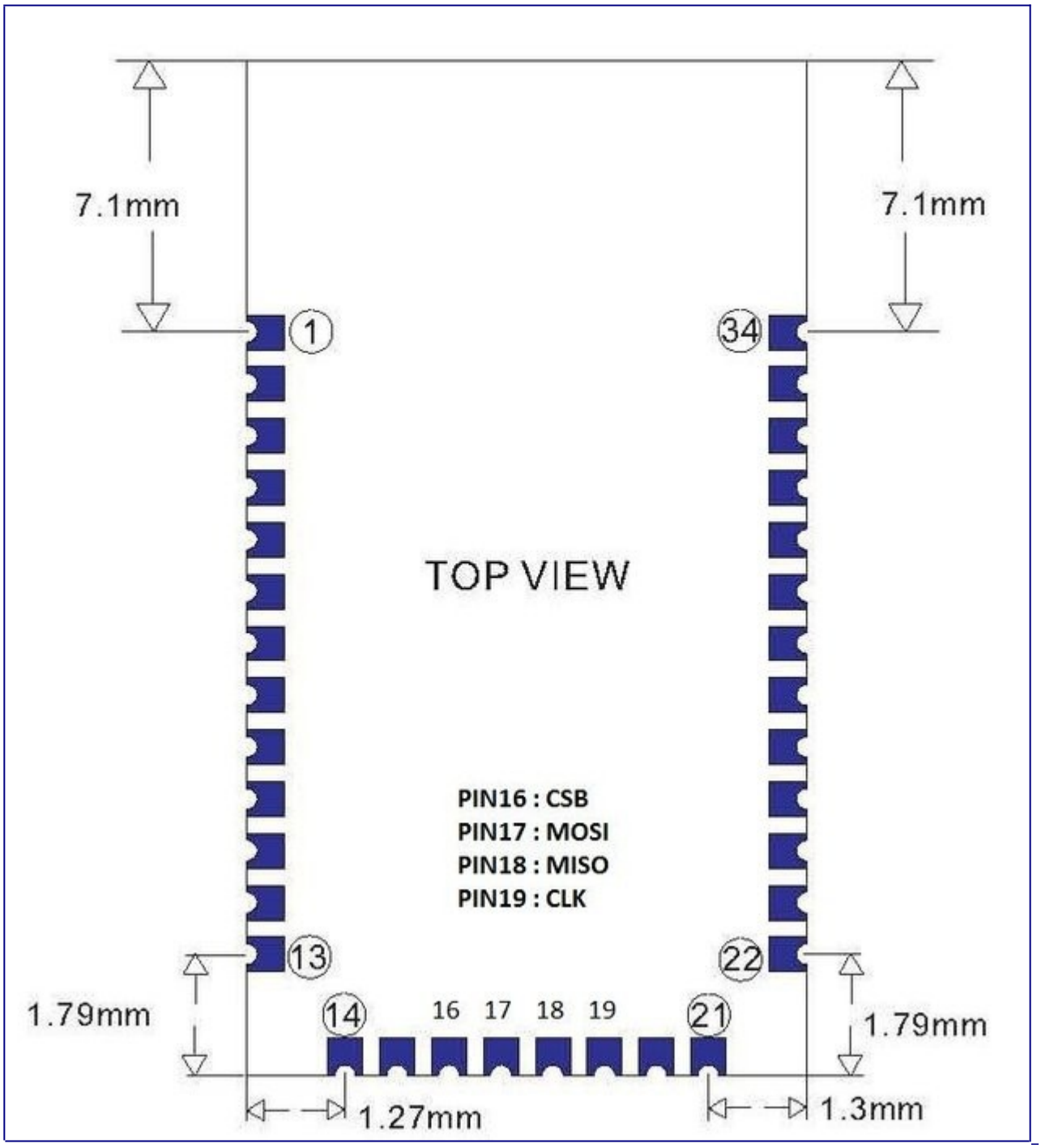
and attempts to connect with the radio. Upon connection the module's LED should stop blinking. Then AT commands may be entered using the Arduino serial monitor. Within the serial monitor, set the baud rate to 9600 and also choose 'no line ending' from the dropdown menu.

Processing Code

```
/*
Demonstration of a Processing client searching for and connecting
to a service.

  extrapixel, 2007
http://www.extrapixel.ch/processing/bluetoothDesktop/.

*/

import bluetoothDesktop.*;

PFont font;
Bluetooth bt;
String msg = "inactive";
Client server;
final String SERVICE_NAME = "Dev B";

void setup() {
  size(600,300);
  font = createFont("Courier", 15);
  textFont(font);
  try {
    bt = new Bluetooth(this, Bluetooth.UUID_RFCOMM); // RFCOMM

    // Start finding the service
    bt.find();
    msg = "searching...";
  }
  catch (RuntimeException e) {
    msg = "error. is your bluetooth on?";
    println(e);
  }

}

void draw() {
  background(0);
```

```
  fill(255);
  text(msg, 10, height/2);
}



// this gets called when the search process is over
void serviceDiscoveryCompleteEvent(Service[] s) {
  Service[] services = (Service[])s;

  msg = "Search completed.";

  // now search for the service we want
  for (int i=0; i<services.length; i++) {
    println(services[i].name);
    if (services[i].name.equals(SERVICE_NAME)) {
      msg = "Service " + SERVICE_NAME + " found";

      try {
        // we found our service, so try to connect to it
        // if we try to connect to it more than once, this will
throw an error.
        server = services[i].connect();
        msg = "Connected to service " + SERVICE_NAME + " on server
" + server.device.name;
        return;
      }
      catch (Exception e) {
        msg = "Found service " + SERVICE_NAME + " on Server " +
server.device.name + ", but connection failed";
        println(e);
        return;
      }
    }
  }

  msg = "Service " + SERVICE_NAME + " not found.";
}
```

Step 5: Application & Troubleshooting



Application:

This should be enough to get started sending data over bluetooth to processing and back again. Once you have a connection, you can send/receive via the Arduino softwareSerial ports using the commands in the bluetooth library!

If you can find and pair with the linvor module but cannot connect...

Using my own frustration as an example, a missing COM port and/or the absence of the SPP service is the most likely cause of a failed attempt to connect and communicate. This is a known problem with this particular module. The cure is simple, but took some time to figure out.

No SPP serial port service?

Go to your bluetooth devices and REMOVE the linvor radio module
Disconnect power Vcc from the linvor radio for 10 seconds
Reconnect Vcc to the radio
Search for the module again on your PC and pair with it

You should now have the SPP serial service available. To verify go to “Show bluetooth devices”. Right click on linvor and click properties. In properties click on the 'Services' tab and the SPP service should be listed.

Here is an example in a project I have worked on in Processing for quite a while. It's a demo of it's current revision and makes use of all of the steps shown in this tutorial...