Arduino Duemilanove Workbook, Wireless and Control

Contents

[Introduction 1](#_Toc297401962)

[Wireless 1](#_Toc297401963)

[Setting up XBee Devices 2](#_Toc297401964)

[XBee – Arduino Hardware Interface 3](#_Toc297401965)

[Programming Arduino for XBee 4](#_Toc297401966)

[Hello World… Again Again 5](#_Toc297401967)

[Off on a Tangent? – The Caesar Cipher **Error! Bookmark not defined.**](#_Toc297401968)

[Assignment 1 – Caesar Cipher **Error! Bookmark not defined.**](#_Toc297401969)

[Adding Electronics – Remote Control 7](#_Toc297401970)

[Exercise 1 8](#_Toc297401971)

[A Last Piece of Advice 8](#_Toc297401972)

[Go Forth and Prosper – Final Assignment **Error! Bookmark not defined.**](#_Toc297401973)

# Introduction

This book is to be used in conjunction with the Arduino Duemilanove Workbook. It builds on material previously studied and therefore assumes a certain amount of previous knowledge. Feel free to refer to the Arduino Duemilanove Workbook if you get stuck (which is quite normal!).

Having previously used the Arduino to interface with computers and simple electronic circuits, we’re now going to investigate the wireless transmission of data between Arduinos. This workbook includes two exercises which, with an accompanying report, will form your assessment for this module.

# Wireless

There are several technologies available that provide wireless communications between hardware devices, many of which are covered in the lectures. We’ve already used IR with the Arduino, but you’ll have seen how susceptible this is to sunlight and physical interference (IR really only works well with a line of sight signal). Much more reliable and much longer range are the radio technologies. We’re going to play with Zigbee.

# Setting up XBee Devices

Our device of choice is the XBee series 2 wireless transceiver. Setup of the firmware on these devices is made through an application that can be found here:

<http://www.digi.com/support/productdetl.jsp?pid=3352&osvid=57&s=316&tp=5&tp2=0>

It is likely that your devices will have already been set up to work together, as programming them requires a separate piece of hardware (It is possible to use the Arduino to do this, but requires the removal of the main chip. This is annoying). However, if you think that your system is working incorrectly, see your tutor for the kit.

Attach the programmer to a spare USB port, open X-CTU and select the ‘PC Settings’ tab. Select correct COM port.

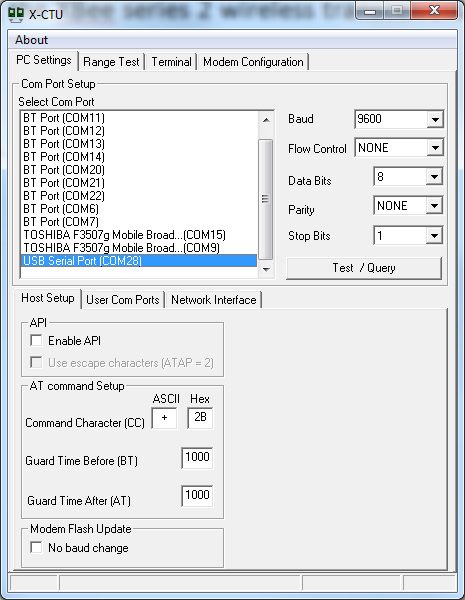
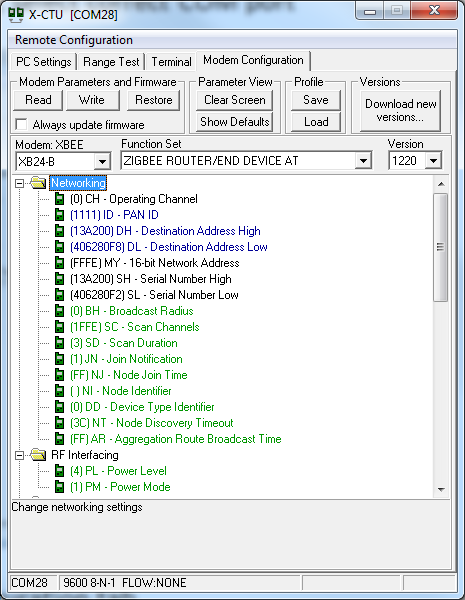


Fig 1: X-CTU Setup, First Screen

Move to the ‘Modem Configuration’ tab and follow the instructions in the text boxes below.

**BEAR IN MIND: One device must be set as ‘Co-ordinator’, the other as ‘Router/End device’. The devices you have been supplied with will have a ‘Tx’ sticker on the co-ordinator.**



For co-ordinator select ‘ZNET 2.5 COORDINATOR AT’ function set. For receiver node, select ‘ZNET 2.5 ROUTER/END DEVICE AT’ function set

For our requirements, Destination address is the address of the other XBEE device in the network

If more than one network is being used within range, PAN ID should be set to a unique value for each network

Select ‘XB24-B’ from Modem menu

Fig 2: X-CTU Setup, Second Screen

Once you have performed the 4 operations outlined in Fig. 2, click the ‘Write’ button. Once complete. The XBee is ready for service.

# XBee – Arduino Hardware Interface

Using these devices with the Arduino requires an interface board (also known as a ‘shield’). Amongst other things, this interface board allows us to communicate between Arduinos, through the XBee, as if using serial communications.



Orientation of Interface board on Arduino

XBee transceiver

Interface board, showing orientation of XBee device

Fig 3: Arduino XBee Interface/Shield Use

One more thing to remember!

**When uploading a sketch to the Arduino, you’ll get an error. Unless you unplug the two jumpers on top of the shield. See below. Note that you could also unplug the whole XBee/Shield unit. This may be a better option once you have overcome the serial interface issue detailed later.**



Remove these two jumpers for programming. BUT MAKE SURE YOU REPLACE THEM AFTERWARDS. Connecting the two ‘inner’ pins on the board

Fig 4: Illustrating Jumpers on XBee Shield

# Programming Arduino for XBee

We will be using the XBee to communicate between Arduinos as if through serial. Set up as we have discussed so far, the Arduino communicates with the XBee through the hardware serial pins (0 and 1). This means that any serial command that we use in a sketch (e.g. Serial.print) is transmitted directly to (and therefore, by) the XBee and anything we receive on the XBee can be read directly using serial commands (e.g. Serial.read). Equally, whenever Serial.print() is used, we will see the result on the serial monitor too.

The problem comes when we want to communicate with the Arduino using both the computer and the Xbee. Both cannot use the same serial pins. To avoid this, we use a software serial library and tie the middle pins of the two jumper sets of the XBee shield to unused I/O pins on the Arduino (Commonly pins 2 and 3). If you’re cunning, you can do it using the jumpers themselves!

The figure below illustrates how the jumpers can be used with two wires to tie the serial pins on the XBee to pins 2 and 3 on the Arduino through the shield. We can then use a software-serial library to emulate serial communications to the XBee, leaving the standard hardware serial pins (0 and 1) for communication between the host computer and the Arduino.

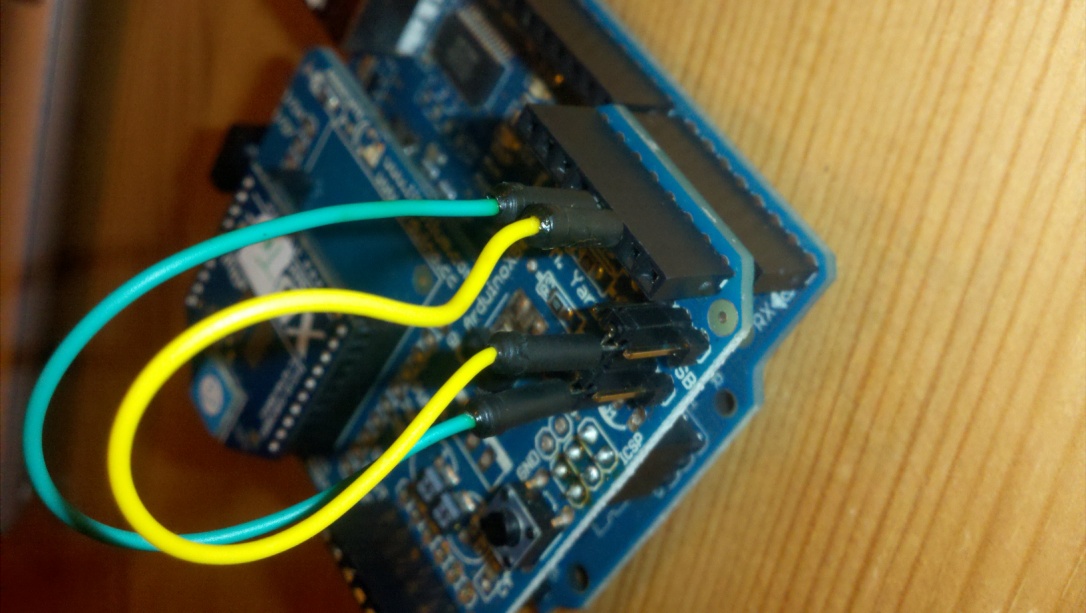


Fig 5: Illustrating the use of Jumpers and wires to use software serial library

As an example of good practice, we’ll include the use of this software serial library as part of and Sketch that uses Xbee from now on, whether we use it or not. We will also assume that the mod in Fig 5 is used unless otherwise stated.

# Hello World… Again Again

As always, we’ll start with a simple ‘Hello World’ sketch. Except that now we need to create two sketches, two XBee loaded Arduinos and at least one computer.

The sketch in Fig 6 will simply transmit ‘Hello World’. In fact, if we simply used Serial.println rather than zigBee.println (remember, zigBee is now the name for our software serial connection), we would simply be printing ‘Hello World’ to the serial monitor.

Upload this to an Arduino. Remove the Arduino from the computer. Attach the ‘Rx’ XBee module and Shield (modified as in Fig 5). Power up the unit separately (The easiest way is to connect the Arduino to another computer).

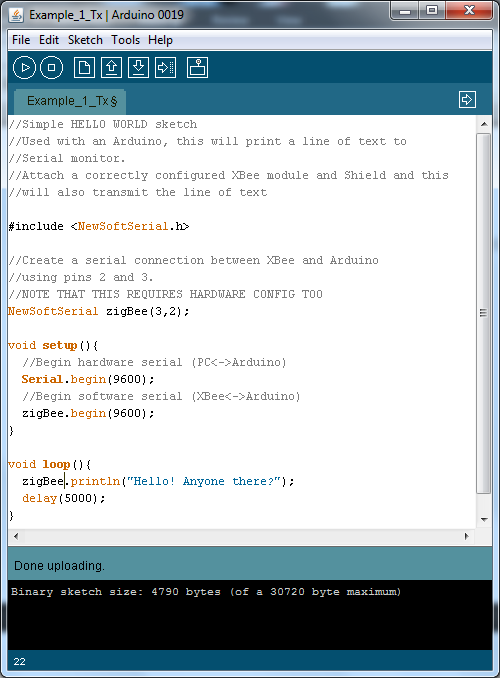


Fig 6: ‘Hello World’ sketch

Attach the other Arduino/Shield and ‘TX’ XBee module to the computer (remembering to change the serial port in the ‘Tools’ menu).

Upload the sketch in Fig 7 to this, second Arduino.

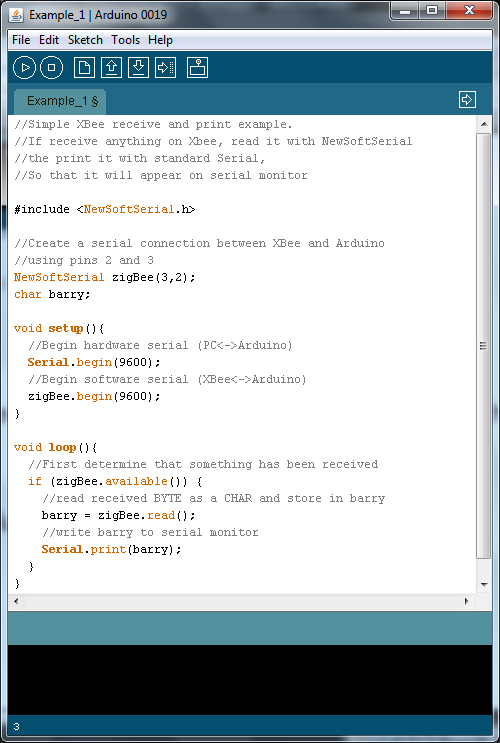


Fig 7: Simple XBee receiver/serial output

Once uploaded, open the serial monitor. You should get an output along the lines of Fig 8.

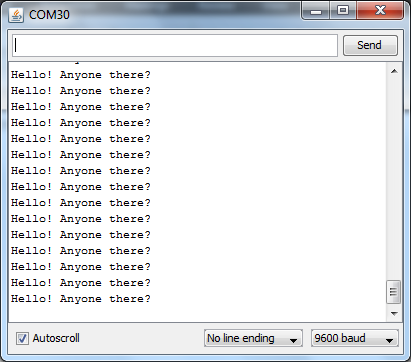


Fig 8: Expected Output of Simple XBee Receiver

# Adding Electronics – Remote Control

If we can transmit serial information between Arduinos, we can control electronics remotely. Simply by passing variables between the two devices. Extending our work, it is a simple matter to transmit an instruction from one arduino to another, switching an LED on and off at the receiver.

Transmitter code would look something like this:

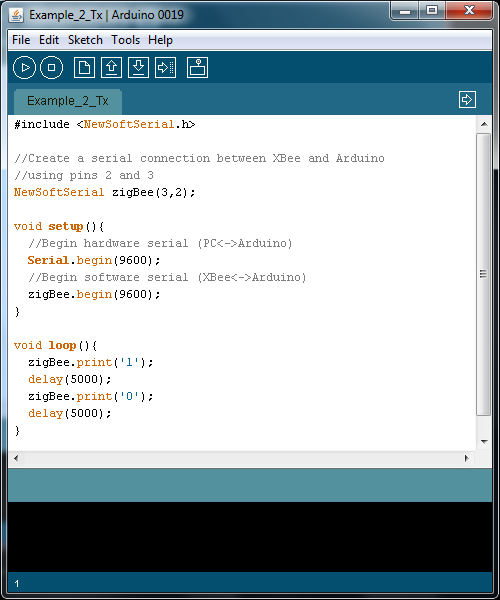


Fig 11: Transmitting ‘1’ and ‘0’

And the receiver code would look something like this:

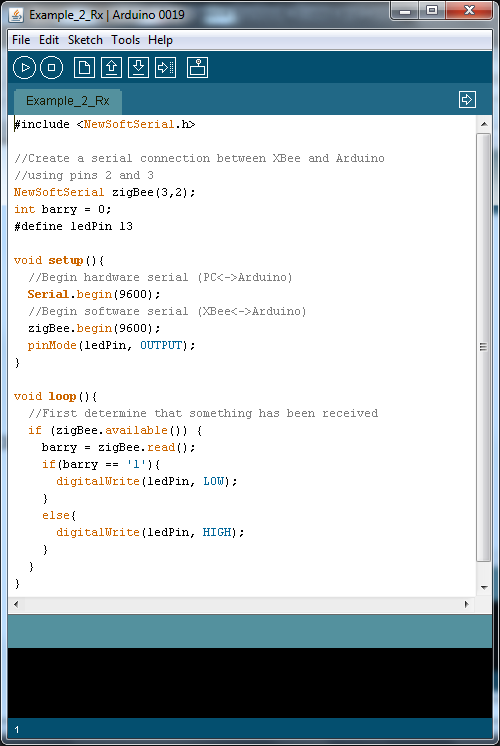


Fig. 12: Controlling an LED with Received Data

# Exercise

Use a potentiometer on the ‘Tx’ Arduino to control the brightness level on an LED connected to the ‘Rx’ Arduino. Refer to your notes from 5CC002 if required.

# A Last Piece of Advice

If you require more than ‘point to point’ communications (i.e. more than 2 wireless devices) with XBee, set your co-ordinator destination addres to DL = 00000000 and DH = 000FFFF. This is broadcast mode to all devices.

When transmitting in broadcast mode, remember that if you want one device to respond to an instruction, you have to identify it within the information you are sending. For example, I would not simply say to a class of 60 – ‘Switch the light on’, because that would mean everyone would get up and try to switch it on! Instead I would say something like – ‘Bob, turn the light on’, identifying Bob before giving the instruction.

Editing the example transmitter sketch of Fig. 11 to transmit two strings alternately (“02” and “12”), where the first value is the address of the device and the second value corresponds to the instruction “JUMP”, and modifying the reciever (See Fig. 13 below) gives you an idea of what we mean.

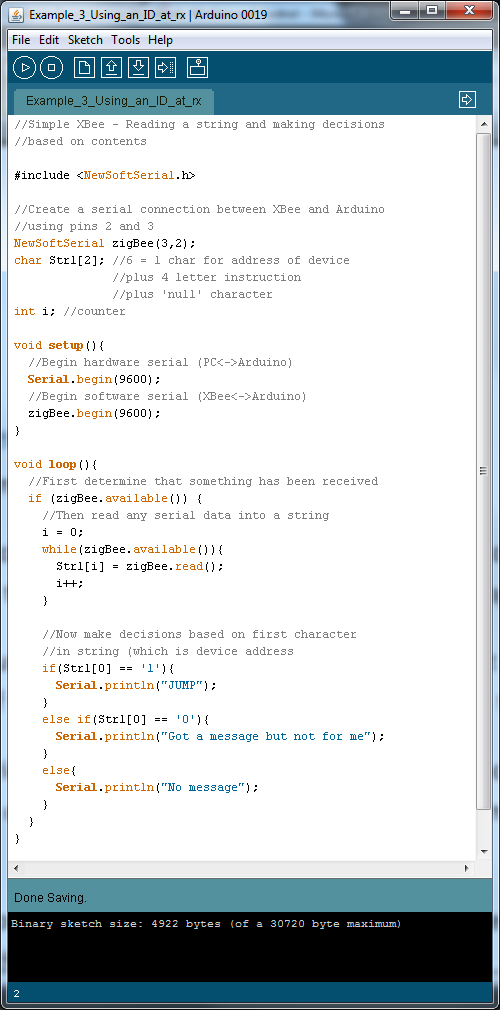


Fig. 13: Addressing each device within a sketch

# Hints and Tips for a Happy Assignment

Piezo buzzers are very simple to use. Refer to <http://www.arduino.cc/en/Tutorial/Melody>. All that they require is connection to a digital output and ground.

Arduino digital pin

Fig. 14: Buzzer Circuit Diagram

Using a Phototransistor. A phototransistor is a variable resistance. Use it like a potentiometer. A low value will occur when the sensor is well lit while a high value will occur when it is in darkness.