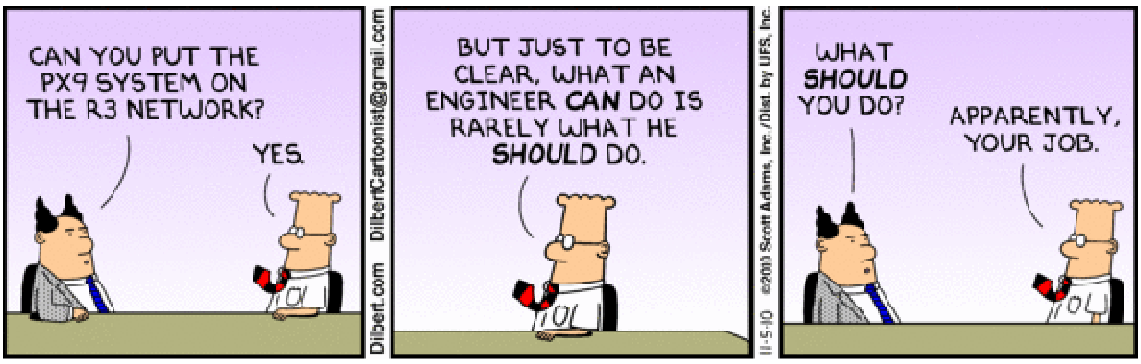
Arduino, Xbee and Matlab GUI Research (HOW TO)

By Roddy Posada and Eric Pak

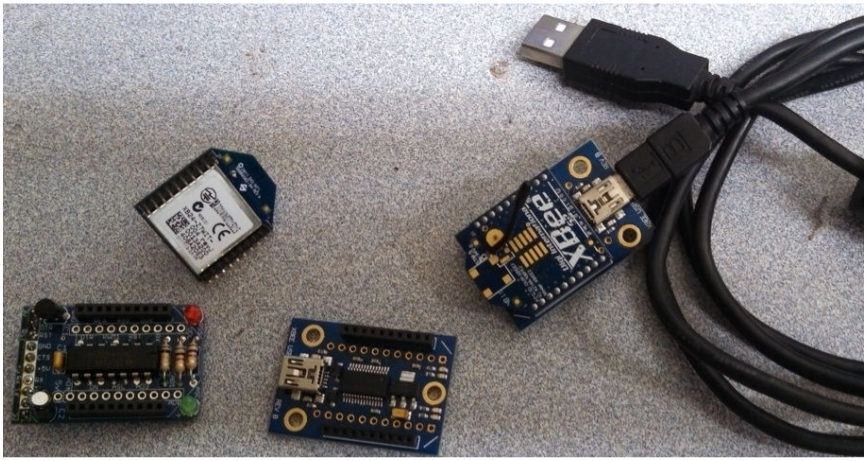
Table of Contents



1. Objective

The main task to be accomplished in this lab is to communicate or replace the USB connection (wire) between Matlab (PC) and the Rover (Arduino), this would be achieved by using wireless adapters, and also by modifying the code for MATLAB and Arduino. As mentioned in the original lab this it would be fun to "*drive around the rover*" using your PC without having the length limitation of the USB cable.
For this lab you will need the following materials:

- (1) Assembled Rover from Build.
- (2) XBee adapters plus breakout boards (2).
- Having completed previous "*Lab 1 Intro to Your Rover."*
- Instructions.
- (2) USB Cable for Xbee and for Arduino.
- MATLAB Software *Version 2008 or higher.*
- Arduino IDE.



1.1. Credits

This lab relies on previous work done by our Instructor on "*Lab 1 Intro to Your Rover*". The work done in this lab is an extension of the "*Creating a Matlab GUI for DC MOTORS.*" Our Instructor assign to us the responsibility of research and modify the original lab to be used in the Spring 2013 class.

1.2. General Comment(s)

If you have completed the previous lab, the most common complication other that coding typos would be the configuration of the Xbee units. In this lab there is a section dedicated to provide some advice for troubleshooting, please read it, and again the following would never be a bad advice "*When in doubt, close everything and unplug everything to start from square one*."

2. Xbee Configuration

2.1. Connecting Xbee to PC

Check all connections; use the USB-miniB cable to plug the FT232 Breakout into a USB port on your computer. The LEDs connected to the XBee should light up and stay on. The task here is to configure/update the firmware on the Xbee to make it work as a Router/End Device or as a Coordinator). Turn on the XBee module and open the software.

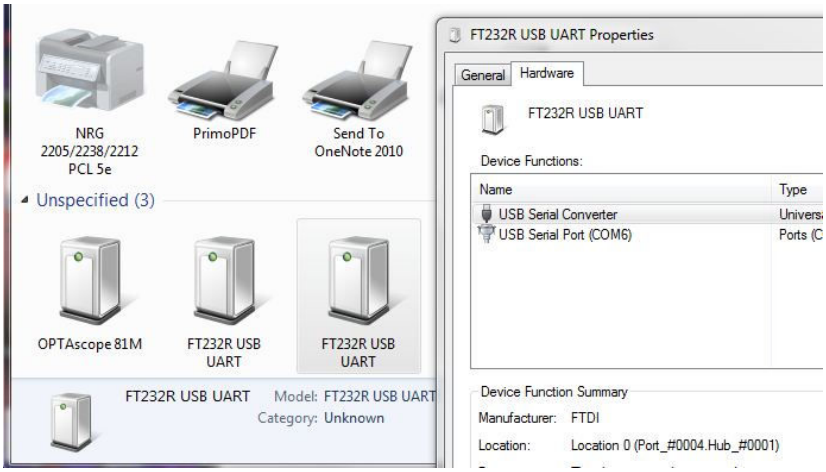


Figure 1

2.2. X-CTU

X-CTU is a tool provided by Digi that helps in the programming of the wide series of Xbee units. Once the software is installed, select the correct usb serial port where the Xbee module is connected. The" PC Settings" contains a Query button that will test the communication with the Xbee module providing information about the firmware version if the communication is successful

One of the most important parameters to setup is the baud rate that for our research is 57600, in simple term this is the communication speed between Xbee module and the PC Software.

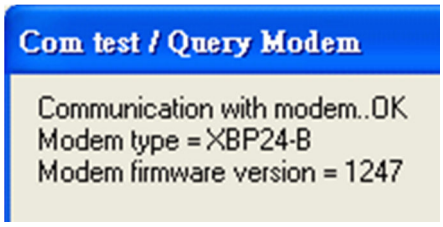


Figure 4

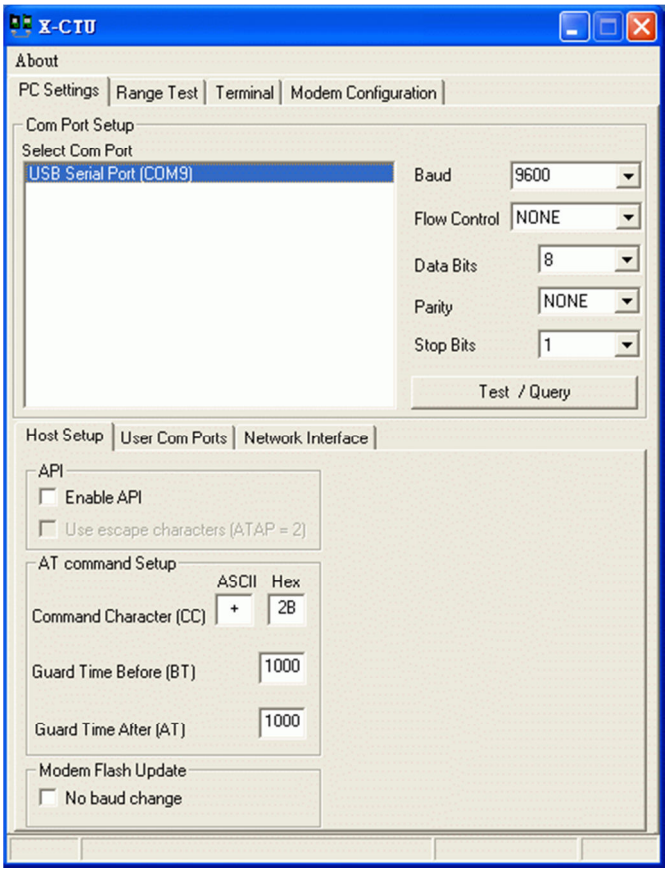


Figure 2

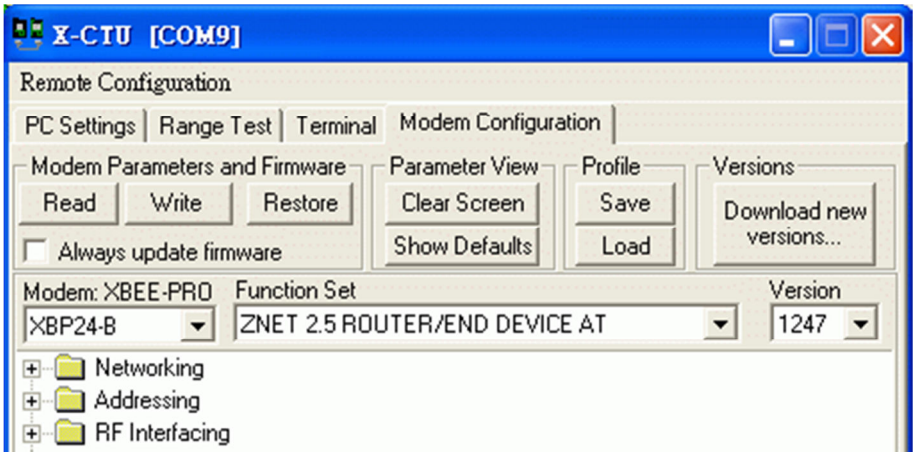


Figure 5

In X-CTU , click on TAB "Modem Configuration" the program has the user interface for firmware downloading and upgrading, also there is a section for Modem parameters and Profile setup.

2.3. Setting a ZigBee Network

ZigBee is a protocol that uses the 802.15.4 standard. ZigBee defines three different device types: coordinator, router, and end devices

A coordinator has the following characteristics: it

- Selects a channel and PAN ID (both 64-bit and 16-bit) to start the network
- Can allow routers and end devices to join the network
- Can assist in routing data
- Cannot sleep--should be mains powered.

A router has the following characteristics: it

- Must join a ZigBee PAN before it can transmit, receive, or route data
- After joining, can allow routers and end devices to join the network
- After joining, can assist in routing data
- Cannot sleep--should be mains powered.

A end device has the following characteristics: it

- Must join a ZigBee PAN before it can transmit or receive data
- Cannot allow devices to join the network
- Must always transmit and receive RF data through its parent. Cannot route data.
- Can enter low power modes to conserve power and can be battery-powered

ZigBee networks are called personal area networks or PANs. Each network is defined with a unique PAN identifier (PAN ID). If multiple ZigBee networks are operating within range of each other, each should have unique PAN IDs. In ZigBee networks, the coordinator must select a PAN ID and channel to start a network. Every Xbee has other critical parameters such as Destination Address High, Destination Address Low, this parameters have to be the same across all devices to be configured, as is shown below:

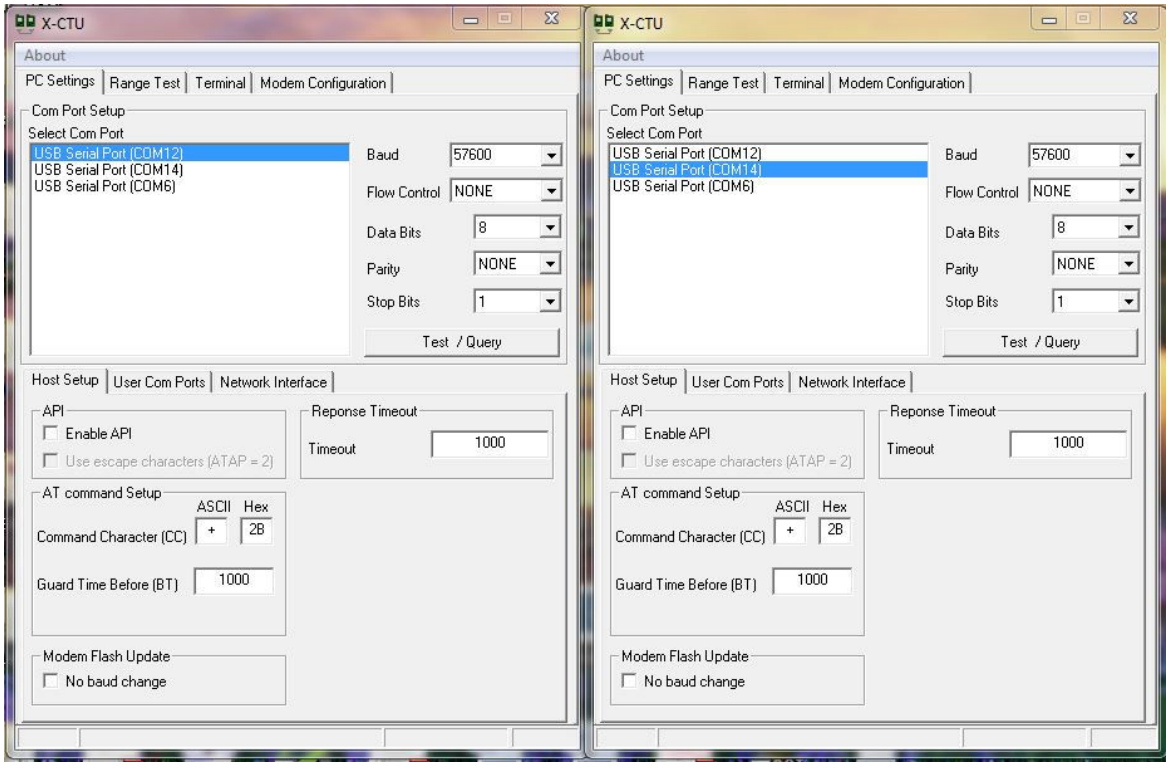


Figure 6

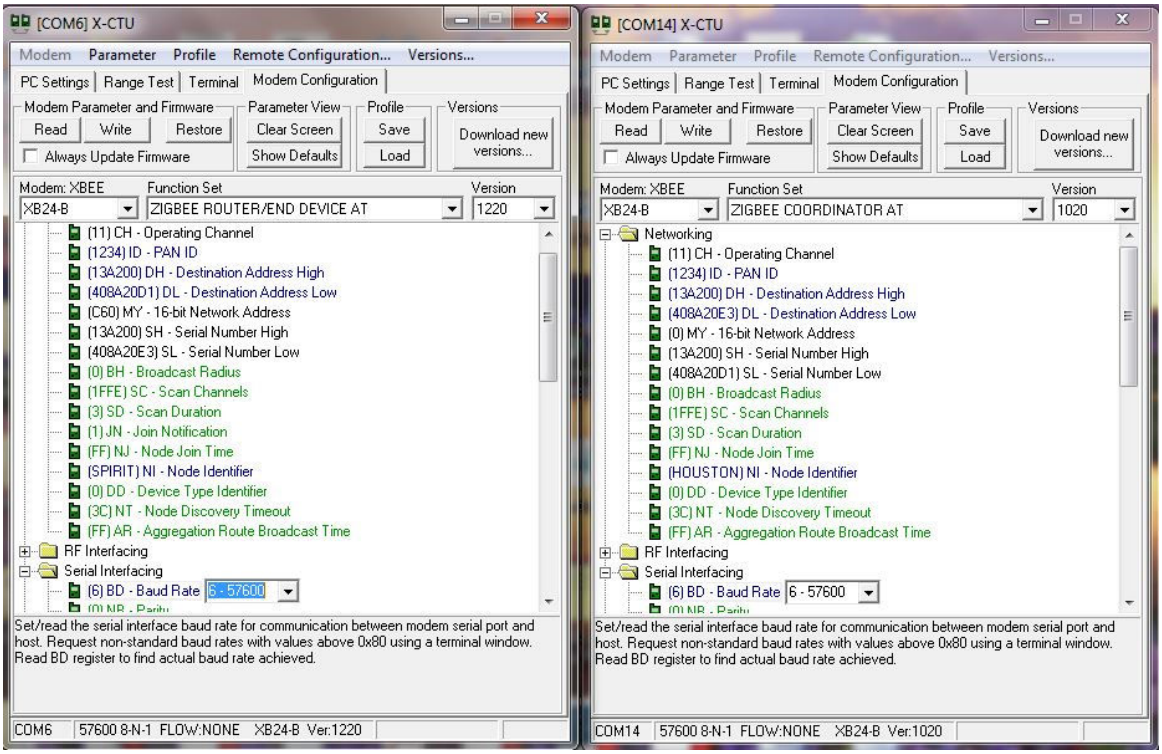
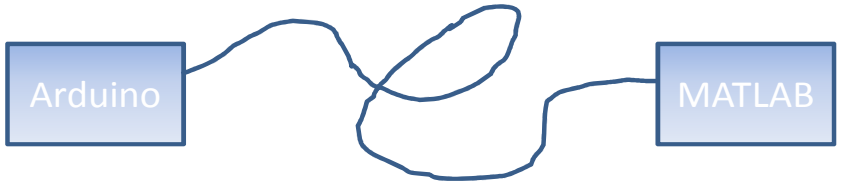


Figure 7

3. Arduino Programming

At this point is fair to mention why speed parameters have to be modified:

- ✓ When using the wired USB interface, the fastest data transmission speeds can be easily achieved, in this case the only limitation would be the processing speed on one end. Usually the speed used is 115000 bauds per second.



USB wire 115000 bauds

- ✓ When using the Xbee adapter we face a speed limitation due to the ability of the xbee to send and receive data, although 115000bauds speed can be achieved, it caused data loss and therefore erratic behavior of the rover.

3.1. Code addition/modification

```
void setup() {
  /* initialize serial
  //Serial.begin(115200);
  Serial.begin(57600);
}
```

4. The Matlab GUI

Matlab has a tools called Guide, for those that have used Microsoft software developing tools, it very similar to the interface for all Visual Studio and Visual Net tools (C++, Visual Basic, etc).

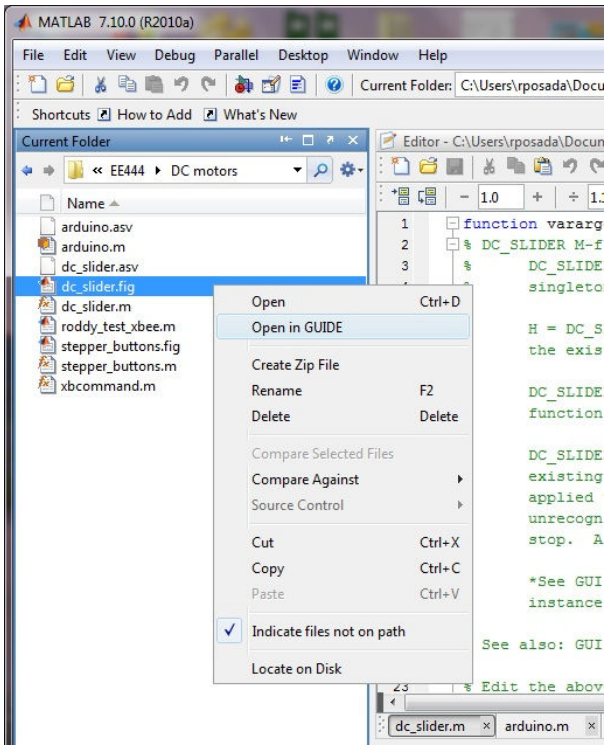


Figure 8

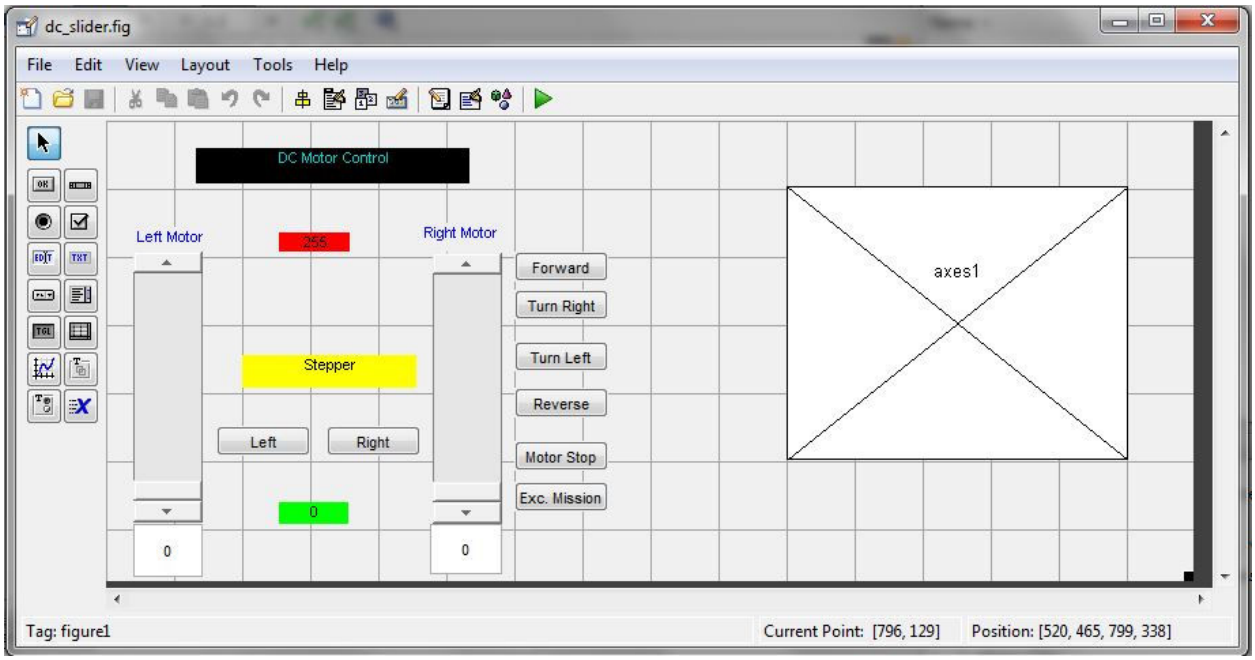


Figure 9

4.1. Code addition/modification

After a research and testing it was found that some parameter from the original code had to be adjusted:

```
File: Arduino.m
...

            % define serial object
            a.aser=serial(comPort,'BaudRate',57600); %original 115200

...
                % it takes several seconds before any operation could be
attempted

                fprintf(1,'Attempting connection ..');
                for i=1:4,
                    fprintf(1,'.');
                    pause(3); %original 1
```

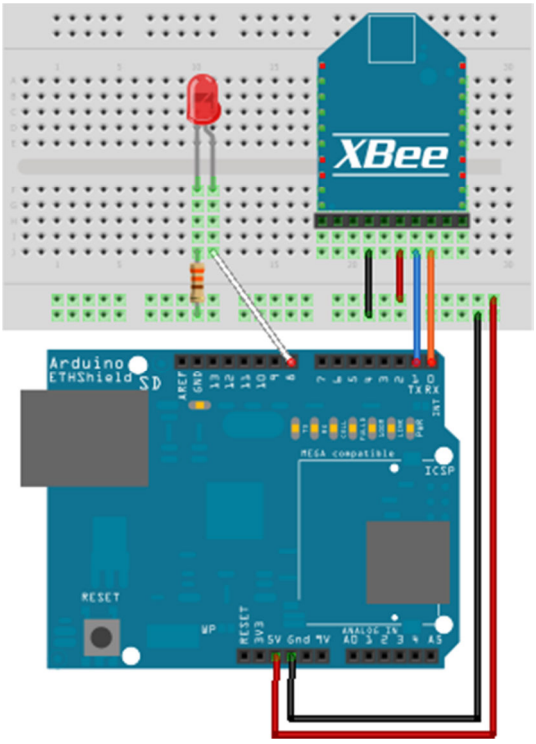
The code below is an automatic addition, to control a new added button instance:

```
DC_SLIDER M-file for dc_slider.fig
…
…

% --- Executes on button press in pushbutton8. // MISSION EXECUTION
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.a.mission()



File: Arduino.m
...


function Mission(a,t)

            % val=a.gyroRead(gaxis); Reads from the L3G4200D gyro over I2C.
            % The first argument before the function name, a, is the arduino
object.
            % The second argument, gaxis, is the axis to read (x, y, or z)
            % where the analog input needs to be performed.
            %
            % Example:
            % val=a.gyroRead('x'); % reads x-axis rotation
            %


            %%%%%%%%%%%%%%%%%%%%%%%%% ARGUMENT CHECKING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            % check arguments if a.chkp is true
            if a.chkp,

                % check nargin
                if nargin~=2,
                    error('Function must have the "t" argument');
                end

            end

            % check a.aser for validity if a.chks is true
            if a.chks,
                errstr=arduino.checkser(a.aser,'valid');
                if ~isempty(errstr), error(errstr); end
            end

            %%%%%%%%%%%%%%%%%%%%%%%%% PERFORM GYRO READ
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            if strcmpi(get(a.aser,'Port'),'DEMO'),
                % handle demo mode
```

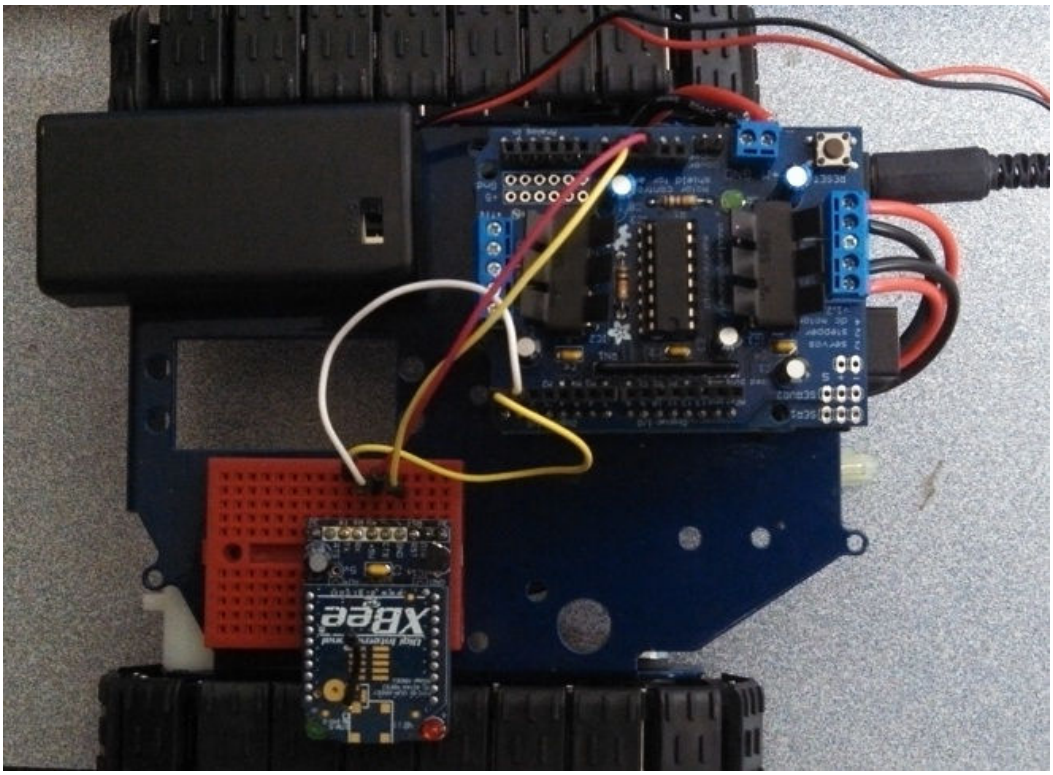
4.2. Xbee communication test code

```
% XBCOMMAND - Write a command to an XBee device

function xbcommand(s, command)

    fprintf('\n');
    timeoutVal = 3; % Number of seconds to wait for data before killing the
program

    if (~ regexp(s.Terminator,'CR')) % Reset the terminator to \r if needed.
        try
            fclose(s);
        end

        s.Terminator = 'CR';
        fopen(s);
    elseif (regexp(s.Status,'closed')) % if s wasn't open already, open it
now.
        fopen(s);
    end

    % Write the user-specified command
    if (regexp(command, '\+\+\+'))
        fwrite(s, command);
    else
        fprintf(s, command);
    end

    % fetch, process and print result
    timeout = 0;
    while (timeout < timeoutVal)
        if (s.BytesAvailable > 0)
            timeout = 0;
            if (s.TransferStatus == 'idle')
                pause(0.01);
                q = fread(s, s.BytesAvailable, 'char');
                q = q';
                fprintf('%s\n', char(q));
                return;
            end
        end
        timeout = timeout + 1;
        pause(1);
    end

    error('Command timed out\n');
```

5. Arduino Wiring



Note that in the following wiring diagrams, the XBees plug into the Adafruit adapter, which in turn plugs into the breadboard. The XBee pins do not (in fact, cannot, spacing is different) connect to the breadboard. The adapter plugs into the breadboard, this is represented by the 10-pin header below the XBee.



6. Help for troubleshooting; don't pull your hair ….. Yet.

Our previous lab experience showed to us that there are endless possibilities of making mistakes and even when everything seems to be fine a failing component could be the cause of many headaches, here is a link to a list of possibilities worth to check: http://www.faludi.com/projects/common-xbee-mistakes/

It is extremely recommended that first you establish the communication between the Xbee units and test both at the same time by connecting them from one USB port to another in the same computer. Once this action is executed successfully, continue onto the wiring for Arduino. A final advice would be to check that all interfaces are setup to the same transmission speeds.

7. Lab Deliverable(s)

- ✓ Demonstrate the operation of your rover using Xbee adapters.
- ✓ Demonstrate proper functionality of the communication interface by executing "There and Back" mission.

8. Future Work

- ✓ Use the Xbee technology to update firmware on the rover (change mission on the go)
- ✓ Show both ways of communication by presenting data from Arduino sensors on the Matlab interface.
- ✓ As USB ports/interfaces are being used, mouse could be easily replace to read from a joystick/gamepad offering better accessibility than clicking.

9. References

- ✓ http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#overview
- ✓ http://arduino.cc/blog/category/wireless/xbee/
- ✓ https://sites.google.com/site/xbeetutorial/xctu
- ✓ Practical Arduino Cool Projects for Open Source Hardware
- ✓ Alasdair Allan, iOS Sensor Apps with Arduino
- ✓ Beginning Arduino Programming
- ✓ Arduino Cookbook
- ✓ https://sites.google.com/site/xbeetutorial/xbee-introduction/zigbee
- ✓ https://sites.google.com/site/xbeetutorial/example/arduino-test-program-for-xbee
- ✓ http://www.jeremyblum.com/2011/02/27/arduino-tutorial-9-wireless-communication/
- ✓ http://arduino.cc/forum/index.php?topic=22324.0
- ✓ http://www.faludi.com/projects/common-xbee-mistakes/
- ✓ EE-440D Fall 2012. Nanocopter Project. Xbee communication Document

- ✓ http://diydrones.com/profiles/blog/show?id=705844%3ABlogPost%3A88961&page=2#comments
- ✓ http://www.ladyada.net/make/xbee/point2point.html
- ✓ http://adventuresinarduinoland.blogspot.com/2011/04/minimal-arduinoxbeepachube-sensor.html

```
% --- Executes on button press in pushbutton8. // MISSION EXECUTION
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%handles.a.motorSpeed(1,200)
%handles.a.motorRun(1,'forward')
%handles.a.motorSpeed(2,200)
%handles.a.motorRun(2,'forward')
handles.a.Mission(251)
for j=1:7
    set(handles.gyro_x,'String',num2str(floor(handles.a.gyroRead('x'))));
    set(handles.gyro_y,'String',num2str(floor(handles.a.gyroRead('y'))));
    set(handles.gyro_z,'String',num2str(floor(handles.a.gyroRead('z'))));
    pause(1);
end


% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.gyro_x,'String',num2str(floor(handles.a.gyroRead('x'))));
set(handles.gyro_y,'String',num2str(floor(handles.a.gyroRead('y'))));
set(handles.gyro_z,'String',num2str(floor(handles.a.gyroRead('z'))));

function gyro_x_Callback(hObject, eventdata, handles)
% hObject    handle to gyro_x (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gyro_x as text
%        str2double(get(hObject,'String')) returns contents of gyro_x as a
double


% --- Executes during object creation, after setting all properties.
function gyro_x_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gyro_x (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function gyro_y_Callback(hObject, eventdata, handles)
% hObject    handle to gyro_y (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gyro_y as text
%        str2double(get(hObject,'String')) returns contents of gyro_y as a
double


% --- Executes during object creation, after setting all properties.
function gyro_y_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gyro_y (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function gyro_z_Callback(hObject, eventdata, handles)
% hObject    handle to gyro_z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gyro_z as text
%        str2double(get(hObject,'String')) returns contents of gyro_z as a
double

% --- Executes during object creation, after setting all properties.
function gyro_z_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gyro_z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1
```