1.Introduction

The project is about exploring the unseen arena using small, cheap swarm robots which are sent into the arena. The robots make random motion and send the information regarding distance traveled and the angle of rotation using zig-bee protocol and a corresponding mapping is done using Matlab, by plotting the coordinates which are sent by the bots. The distance After a certain duration, the output of the matlab contains dots where there are no obstacles and a plain area which has obstacles, since the bots can't go into the rigid area of obstacles.

Definitions

ZigBee is a specification for a suite of high level communication protocols using small, low-power digital radios based on an IEEE 802 standard for personal area networks. More information on zigbee can be found in appendix-a.

Swarm robotics is a new approach to the coordination of multi robot which consist of large numbers of mostly simple physical robots.

Wheel Encoders The wheel encoder is a sensor attached to a rotating object (such as a wheel or motor) to measure rotation. By measuring rotation your robot can do things such as determine displacement, velocity, acceleration, or the angle of a rotating sensor.

2.Problem Statement

Multiple robots which are cheap, unintelligent and random moving, also called swarm robots are released into an unseen arena. Each and every bot keeps sending the distance they have traversed and the angle of rotation, either when they meet obstacle or when the wheel does a complete rotation. The signals are sent using Zig-bee protocol. The matlab interprets these signals and converts them into coordinates by using polar coordinates and plots them onto the image. After some duration(which Is inversely proportional to the number of bots used), the image contains points where there are no obstacles and a distinguishable boundary of the obstacle.

3.Requirements

Functional requirementst

The swarm robots use IR sensors which reflect the light when an obstacle comes across the path(obstacle of white color). When an obstacle is met, the algorithm calculates the distance traversed, angle rotated and feds this as input to zigbee.

The input for the matlab is the signals received from the bots using XBEE (Based on zigbee protocol, IEEE 802.15.4) which gives information about coordinates.

The matlab processes these input and uses the algorithm embedded in it to find the polar coordinates and maps them onto the image. After a certain duration the image contains information about the obstacles present in the unseen arena.

Non-functional Requirements

Accuracy in determining and transmitting the coordinates

Less delay in transmitting and receiving the signals.

It should be acurate and efficient.

Communication overhead should be low

No data loss due to collision in signals received from various bots simultaneously.

Hardware requirements

Wheel encoder

Zig-bee USB adaptor

Onboard battery support

Swarm robots

Digital IR sensor

Fire-bird

4.Implementation

4.1 Matlab Algorithm Description:

Initially we will close the previously opened serial port by checking its existence. Then we will initialize the arena array and have initialized the serial port using serial function. Next, we need to define the input buffer size and the open the serial port. Opening the serial port may take 15-20 seconds. Just after this we will read the input buffer input buffer is updated automatically everytime data arrives at the USB port. This received data is stored in the inbuff array.

In while loop we will keep updating the inbufff array with the new packet and then we will start parsing of the packet using 3 while loops. This will extract the robot id in the packet, distance travelled in ticks and then theta in ticks. All three variables will be updated in string variable. After that all string variables will be converted into integer variables for the processing.

First we will convert theta into degree and r into distance traveled. Then we will update the arena or plot the new box in the arena as per the new coordinates before using x & y we will using polar coordinates system to convert them into x & y.

4.2 C Code Description:

The most important and best used feature of the ATMEGA -16 is ISR (interrupt service routine) . ATMEGA-16 is the basic building block of the SPARK-V . So lets have a look how an ISR works.

An **interrupt handler**, also known as an **interrupt service routine** (**ISR**), is a callback subroutine in microcontroller firmware, operating system or device driver whose execution is triggered by the reception of an interrupt. Interrupt handlers have a multitude of functions, which vary based on the reason the interrupt was generated and the speed at which the interrupt handler completes its task.
An interrupt handler is a low-level counterpart of event handlers. These handlers are initiated by either hardware interrupts or interrupt instructions in software, and are used for servicing hardware devices and transitions between protected modes of operation such as system calls.

We have used ISR to serve our real time need of data sending through zigbee protocol and count the shaft counts. Shaft count is used to perform all the basic operations like “TURNING BY SPECIFIED DEGREE” “MOOVING BY A SPECIFIED AMMOUNT OF DISTANCE”.

Another important thing is the sensing the environment which is done by ADC converters . They are used to sense obstacles in our case.

We sense the obstacles in a regular basis and send distance traveled information after moving some fixed distance or facing an obstacle .

The code has been discussed elaborately in the following section.

Robot reads the three sensors first. If it detects an obstacle it takes ome random turn. First it decides the random direction I.e, left or right then decides ammount of shaft counts by which it has to rotate. It then adds the rotated shaft count to a SUM variable (that sums all the rotations taken at a time) .The robot again goes for another execution of the while (true) loop. Where it again reads and checks the sensors and takes another random turn in the last decided direction and sums the value to the previously told SUM value .this process of turing again and again in the same place occures until the bot avoids the obstacle , after avoiding it in eneters to no obstacle section and runs for 6 shaft counts after that it sends the previous sum of turning information and distance measure of 6 .

The bot transmitts position information after every 6 shaft counts and when an obstacle is detected all these informations are read by the matlab program and processed to generate the image explaining the area being explored . The working of matlab code has been explained later .

The above explained procedure is repeated infinitely by the while(true) loop. And the matlab execution depends upon the sending of information by the bot. So both of them work synchronously.

There is no speed gap in processing because the matlab is executed in a general purpose pc which is 1000 times faster than ATMEGA-16 .

5. Testing Strategy and Data

We have implemented & tested the robots in spark 5. For testing we have tried circular and squared objects as an obstacle in the arena and we have found that round objects are showing more correct output.

We have also implemented the different version of speed and found the robot to be working more precisely on low speed. Considering different size of arena and different size & shape objects we have found the robot to be showing more correct output on round shape arena and round shape objects.Robot size has to be small to make more precise and correct output.

6. System description

The system consists of two parts : one is high end processing system and other consist of large numbers of mostly simple physical robots. The high end system is a general purpose computer containing matlab and XBEE wireless system. The simple physical robots used here are SPARK V robots. The least requirements of these robots which are required for project are precisely working wheel encoders, motors, XBEE wireless, obstacle detection sensors and processing component. The obstacle detection sensors used are IR sensors. The SPARK V contains 3 IR sensors.

These is a unidirectional communication between these two systems using XBEE, where robots send message as packets to system with matlab when detecting obstacle. There is no need of sending data from matlab program to robots.

The details of implementation of the system has been described in implementation part.

Working

The robots, when started, moves in forward direction until it detects a obstacle and when it detects the obstacle, then it sends message to the matlab using XBEE about its ID, the length covered in forward motion from the last time it detected obstacle and the total angle turned when it detected the obstacle previous time and then it chooses a random direction and then continuously selects a random angle and turns by that much angle until it does not detect the obstacle in that position and then again moves in the forward direction. This process continues until it covers the unseen arena.

When matlab receives the message, it extracts the data from the message and uses a mathematical formula to compute the co-ordinates of the robot, whose id is present in the message, in the arena and then map it to the image of arena. This process follows every time matlab receives the message from the robots.

So gradually the information about the arena increases and the map of the arena becomes clear. Gradually the portion where obstacle is not present is filled with black dots and the portion where obstacle is present remains white as robots can not cover those portion.

Things that worked as panned

The code has been written as per the requirements and it works fine and efficiently starting from giving forward motion,reading IR proximity sensors,sending data through XBEE from multiple robots without congestion,executing ISR to update wheel shaft counts on interrupt,turning robots at a random angle to give random motion to processing in matlab to update the co-ordinates of every robots and displaying map of arena with every update.

Also the simulation of project done using stage software works fine and produces the output as expected and is a good approximation of the arena it simulates.

Things that did not work as planned and their reason:

The spark V robot doesn't possess obstacle detection sensors, but IR proximity sensors. Hence it becomes mandatory to cover the walls of the arena boundary and the obstacles with white surface.

Further, swarm robots need to detect the each other in the arena as the obstacle. Since Spark V is not covered with a white surface, it has become difficult to place more than one robot on the arena.

Since there is a problem with motor and wheel-encoder , so the data plotted by matlab is too different from the actual scenario leading to a map of the arena plotted by matlab that can hardly represent the actual arena and also far below what was expected in spite of the fact that all software part were working perfectly. The reasons are explained below in detail.

There is an error with the wheel-encoder. In Spark V, one shaft count value = 12.85 degrees. Hence, when an obstacle is encountered, distance traveled and the previous angle rotated is sent as the input to mat-lab. Hence each time an obstacle is encountered, there is a an error of 6.5 degrees on an average. Thus, when the bot is given an instruction to rotated at an angle of 102 degrees, it may rotate 90 degrees and since, we are calculating the cumulative distance, the error keeps incrementing and the bot deviates from the path which is displayed by the image. So at some point the bot faces the obstacle and moves while that point is mapped in matlab to a point where there is no obsatcle or some other obstacle is present. So this problem gradually distorts the map.

Also initially we were under the impression that the error in calculating the coordinates could be due to slip in wheels. But we started to face problem with the variation in velocity of the two motors.

To overcome the above problem, we tried to calculate the difference in the velocity of the two motors and hard coded by reducing the velocity of one wheel with the difference calculated.

But it dint seem to work, as the variation in the velocity is not constant through out.

The other solution we tried out was to reduce compensate the variation in the velocity by reducing/increasing the velocity dynamically by using an algorithm which measures the difference in the shaft counts of left and right wheels and base on that value, decision is taken and velocity of the wheel which is moving at a faster pace is reduced.

But this algorithm wasn't able to solve the problem because of the error in the shaft count values.

<u>**One important Problem-faced**</u>

The algorithm that dynamically compensates the change in velocity has a serious problem. If the bot goes left by an angle of 45 degrees, it is compensated by the algorithm by rotating the bot by -45 degrees. But in the mean while, the bot deviated from the original path by a distance 'd' units. Now, in reality, the bot needs to take -90 degrees and move a distance equal to 2d in that direction and then take another 45 degrees to overcome this problem. But it is not feasible because the bot then keeps moving in a zig-zag path and hence the distance traversed by the bot is way more than what it needs to traverse, thus results in wrong results. Further, The algorithm is only used to DETECT the error but not PREVENT it, it becomes infeasible to get the coordinates with accuracy.

One solution to this problem with the wheel-encoders is to use "Triangulation method" but it needs accuracy and efficient hardware and this can be implemented in the future.

Simulation

To check if the algorithm is efficient enough we simulated the entire process using Stage simulator and found the results quite encouraging. In an area of 14*14 units, we put 6 Swarm robots with random rotation and these robots could trace the entire arena and send the coordinates in less than 100 seconds. We have taken these signals and used GNUPLOT to show the image by plotting these points.

During simulation we faced problem of swarm robots not detecting each other. To overcome this problem we increased the number of sensors to five(5) from three(3). The swarm robots now, could detect each other efficiently. Similarly, if the spark V can be accommodated with 5 sensors in the front, it could determine any obstacle covered with a white surface, else it becomes important that the obstacles need to be circular in shape to avoid collision.

7. Future Work :

Scalability with desired efficiency is one of the most important aspects of a project. When we are dealing with swarm robots, scalability should be the inborn nature of such projects. Because swarm robots are small cheap robots meant to be used in group to give at an expected result. Our Project is having an efficient scalability which can be extended for very large problem size (large problem size means exploring a large arena) .
It can be achieved just by programming each bot with an unique ID (The ID is just an integer currently its 3 digit max I.e, it supports 999 robots now) but it can be more .

A camera can be mounted on the bot which will give the image (and if multiple cameras mounted views from multiple directions can be taken and processed). By mounting the camera on top of these robots, we not only get obstacles in the arena, but also details such as dimensions of the room(such as height), objects in the room, multiple entries into the room and vulnerabilities of the arena which can be exploited. But this will need costly computing in the image processing end. More ever the processing is video processing . Which is far more complex than image processing than This will also increase the cost of communication by sending a lot of data . If all these issues are tackled efficiently, it fetches wonderful results . We will use a wider channel for sending images from the robot to high end processor. In the video processing use of parallel distributed computing will solve the issue. And the Parallel distributed systems need to be equipped with high speed graphics processors .

For positioning of the robot inside the arena we have used distance traveled from last know position and the direction of traveling with respect to a reference axis ,and initially the bots position and direction was known to the processor running matlab code. But better positioning methods like `` Triangulation method " can be used to provide less error. But this will increase the program complexity in robot side. For which we have to sacrifice some speed of the bot other wise the bot will be unable to process a large amount of data (like obstacle detection , taking random direction, receiving signals from 3 sonar radiators and replying them) in real time .

Another interesting approach is to make a boundary following robot . When it detects an obstacle it try to cover the obstacle's boundary by trying to keep the obstacle sensor's value in a predefined interval so that neither the robot gets collided with the obstacle nor it leaves it . When it reaches the same position again the bot is signaled to to leave that obstacle.

8.Conclusions:

This project can be used for serving military,household purposes. In military these cheap swarm robots can be used to spy enemy area and send information of the map of that area to base station.

For household uses swarm bots featured with this intelligence can prepare the map of a which can be used by other bots. This method of generating information automatically can lead us to design robots teaching other robots or we can say bots showing path to other robots.

Further, these swarm robots which are relatively cheaper, can be used in house hold applications to clean the houses, when they are free as they already possess the map of the house.

The things that seem to be achievable using software simulation is not always accomplished as expected. So the feasibility of the project (especially an embedded project) should be analyzed from a hardware perspective ,Real time analysis . Simulation is essential . Consulting an experienced person in the same area shows the best path.

9. References :

[1] MATLAB R2011b Documentation [url : http://www.mathworks.in/help/techdoc/index.html]

[2] STAGE 4.0.0 Reference manual [url : http://rtv.github.com/Stage/]

[3] WinAVR User Manual – 20100110 [url : http://dybkowski.net/download/winavr-user-manual.html]

[4] SPARK-V Hardware Manual

[5] SPARK-V Software Manual

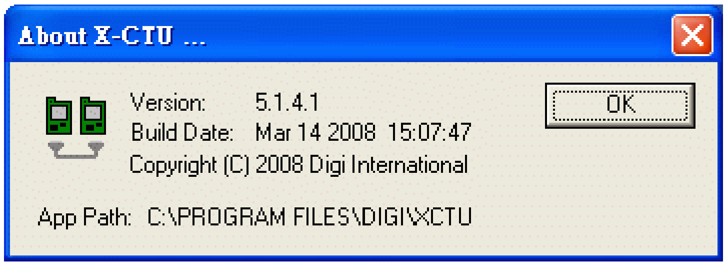
APPENDIX-A

XBee and XBee-PRO 802.15.4 OEM RF modules are embedded solutions providing wireless end-point connectivity to devices. These modules use the IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking. They are designed for high-throughput applications requiring low latency and predictable communication timing.
While Bluetooth® focuses on connectivity between large packet user devices, such as laptops, phones, and major peripherals, ZigBee is designed to provide highly efficient connectivity between small packet devices. As a result of its simplified operations, which are one to two full orders of magnitude less complex than a comparable Bluetooth® device, pricing for ZigBee devices is extremely competitive, with full nodes available for a fraction of the cost of a Bluetooth node.
ZigBee devices are actively limited to a through-rate of 250 Kbps, compared to Bluetooth's much larger pipeline of 1Mbps, operating on the 2.4 GHz ISM band, which is available throughout most of the world.

What is X-CTU?

Digi International offers a convenient tool for Xbee module programming - X-CTU. With this software, the user be able to upgrade the firmware, update the parameters, perform communication testing easily. The basic operations are listed below.

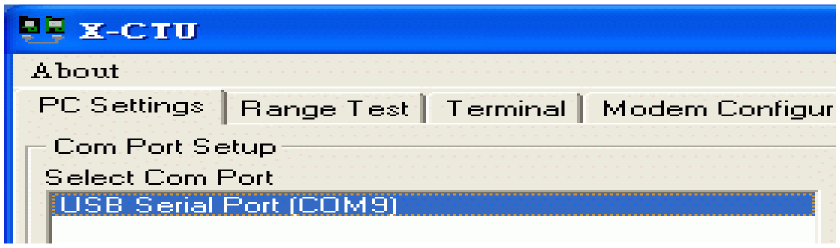
How To Use X-CTU?



Version: 5.1.4.1
Build Date: Mar 14 2008 15:07:47
Copyright (C) 2008 Digi International

App Path: C:\PROGRAM FILES\DIGI\XCTU

Before we can talk to an XBee, except USB cable we will need to get an USB adapter for XBee With the USB adapter, we can communicate with Xbee through **"USB Serial Port"**. We may have more than one device on serial port, for example, we want to test the wireless communication and connect two XBee to our PC. So, we will add more devices on serial port. In either case, we need to select the correct one that we want to perform operations.



Select Com Port
USB Serial Port (COM9)

In this case, it only has one Xbee connect to PC and it locates on com port 9 (COM9).

Xbee Query:

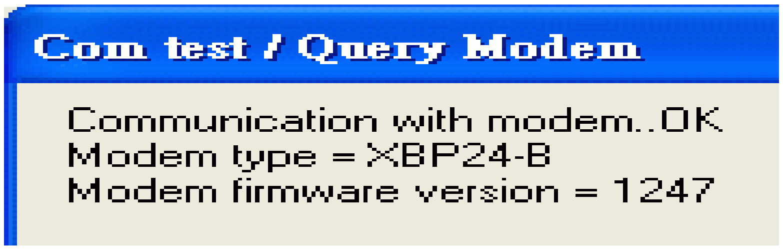
This is an easy way to test and check if an XBee is working and configuring properly. After parameter modification and firmware upgrading , we need to do this for checking if everything is ok. we will have a very little chance to get a problem Xbee. If we got problem to Test / Query an XBee , usually it is due to the wrong parameter setting.

For a successful two way communication, the most primary principle is the **"Baud Rate"** should match each other.

For a new Xbee module, follow this procedure to query.

8) Select com port in section : "PC Settings"
9) Baud Rate set to 9600 (only for the new Xbee module , set the value to your case)
10) Flow Control : NONE
11) Data Bits : 8
12) Parity : NONE
13) Stop Bits : 1
14) Enable API : Uncheck
15) Click "Test / Query"

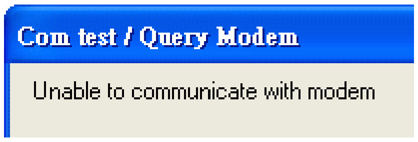
If we had set the Baud Rate to other value, then we should change to it.


Com test / Query Modem

Communication with modem..OK
Modem type = XBP24-B
Modem firmware version = 1247

When everything comes to the right place, this window will appear. If we see any other kind of message window, it means something wrong even we see an OK on it. It is not ok without this message and window. With the same setting, if we only got this occasionally , it means the communication is unstable. We may need to use an XBee adapter with a better quality , or try another XBee.

The modem type and firmware version means the firmware be programmed in this XBee. It is possible to have other types of firmware , depends on your usage for XBee.


Com test / Query Modem

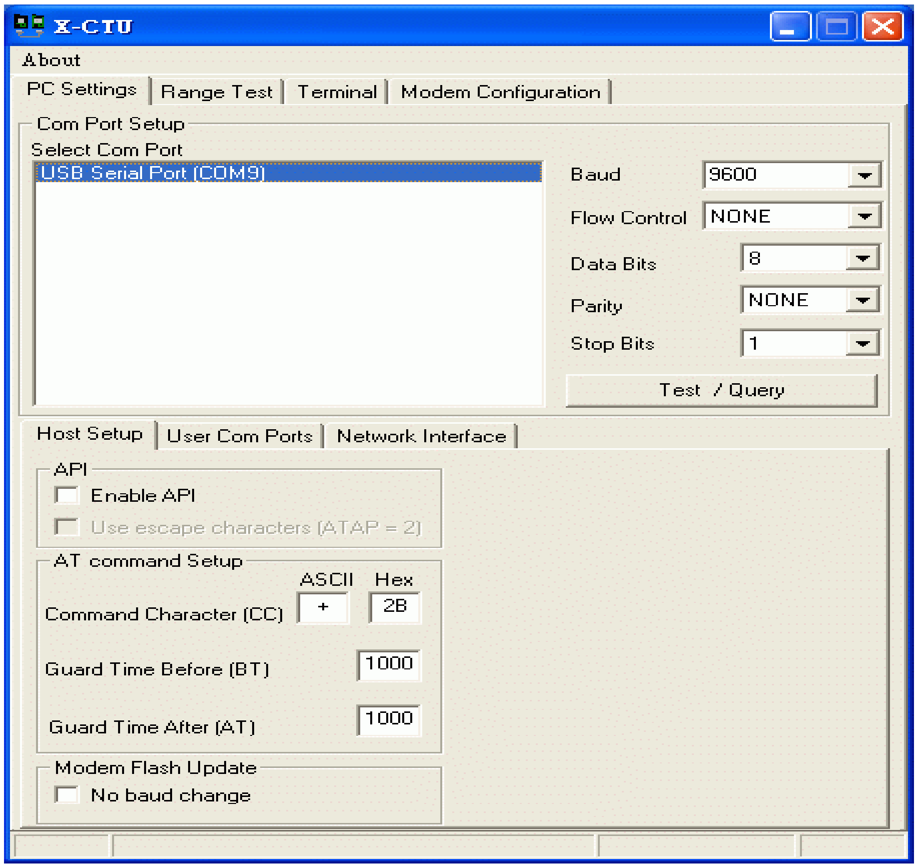
Unable to communicate with modem

If something goes wrong , then we will get this window. For most cases , it can be solved by just changing the Baud Rate and un/check API Mode. We will have a detailed information about this latter.

The wrong firmware in XBee will also result in this kind of message. If the setting and firmware are correct , then the hardware may have problem. In this case , check the adapter first then XBee. A hand made , bad quality Xbee adapter may result in this. Although the specification can not tell this , but XBee Pro apparently has a better tolerance on power source and adapter than normal XBee.

The next section will discuss the four main tabs in X-CTU and information about using XBee with Arduino.

<u>X-CTU User Interface:</u>



USB Serial Port (COM9)

Baud 9600
Flow Control NONE
Data Bits 8
Parity NONE
Stop Bits 1

API
Enable API
Use escape characters (ATAP = 2)

AT command Setup

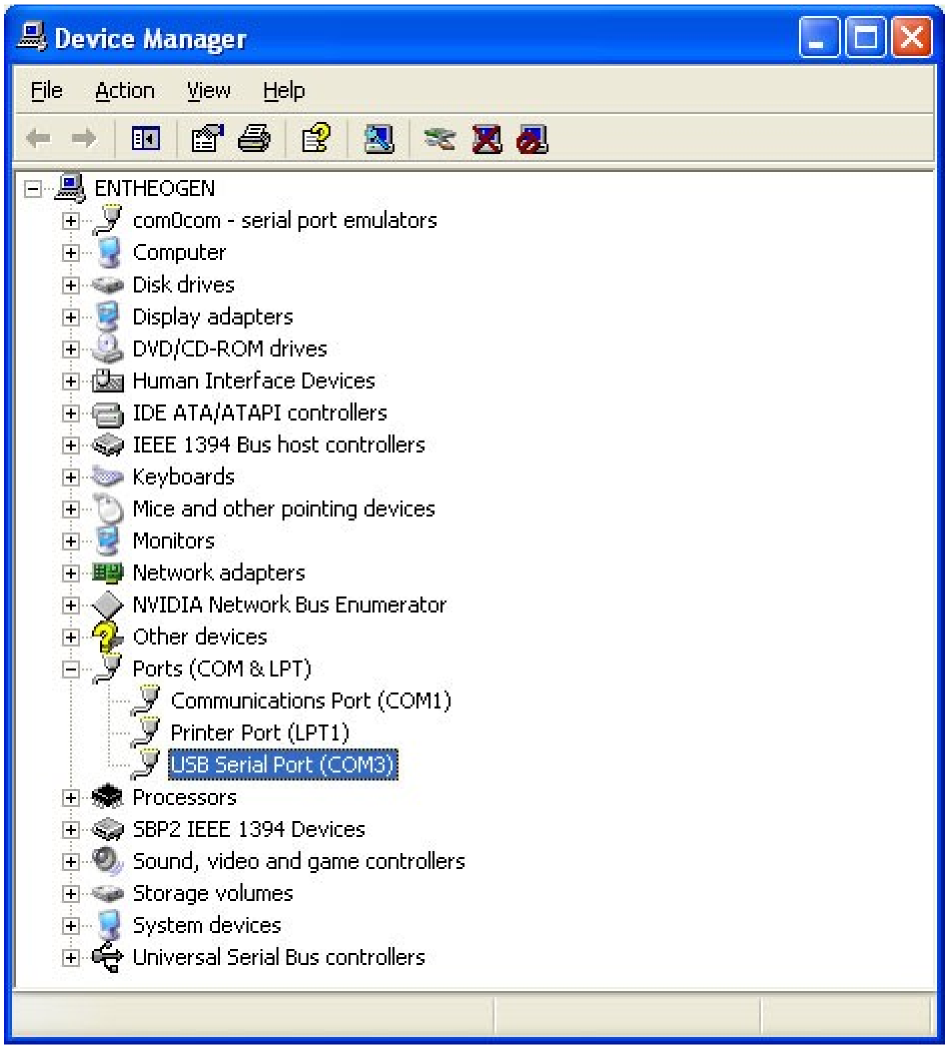
	ASCII	Hex
Command Character (CC)	+	2B
Guard Time Before (BT)		1000
Guard Time After (AT)		1000

Modem Flash Update
No baud change

<u>Connecting & Configuring XBee for the project:</u>

First, insert the xbee module in to the Xbee USB adapter.

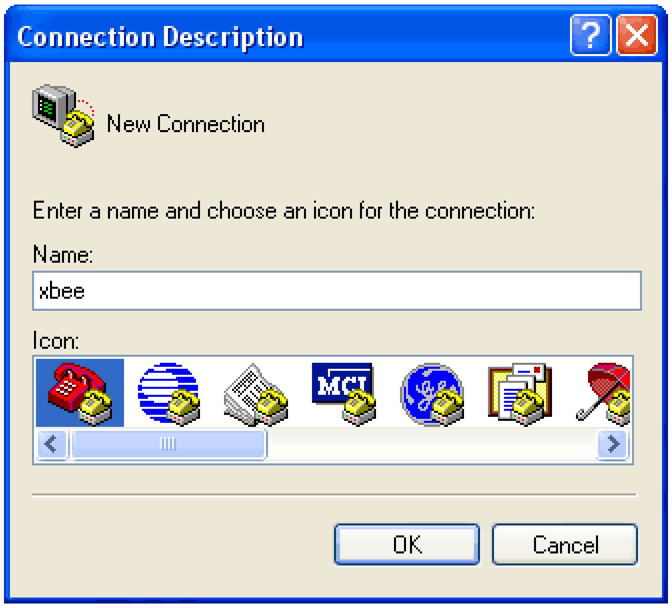
Next, we'll need to figure out which serial port (COM) we are using. Plug in the FTDI cable, USB adapter, Arduino, etc. Under windows, check the device manager, look for "USB Serial Port"



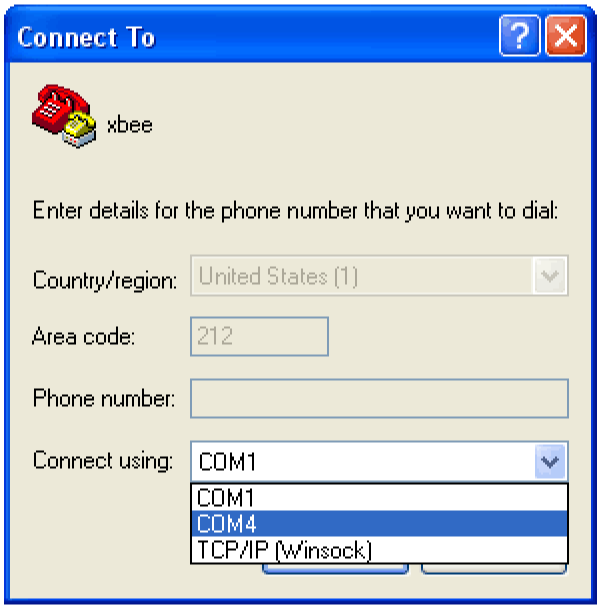
```
ENTHEOGEN
    com0com - serial port emulators
    Computer
    Disk drives
    Display adapters
    DVD/CD-ROM drives
    Human Interface Devices
    IDE ATA/ATAPI controllers
    IEEE 1394 Bus host controllers
    Keyboards
    Mice and other pointing devices
    Monitors
    Network adapters
    NVIDIA Network Bus Enumerator
    Other devices
    Ports (COM & LPT)
        Communications Port (COM1)
        Printer Port (LPT1)
        USB Serial Port (COM3)
    Processors
    SBP2 IEEE 1394 Devices
    Sound, video and game controllers
    Storage volumes
    System devices
    Universal Serial Bus controllers
```

Next we'll need to open up a terminal program. Windows comes with Hyperterminal, so just use that. Its under **Start->Programs->Accessories->Communications->HyperTerminal**. If we are running a different operating system just use whatever terminal program is available for it, such as ZTerm, minicom, etc.

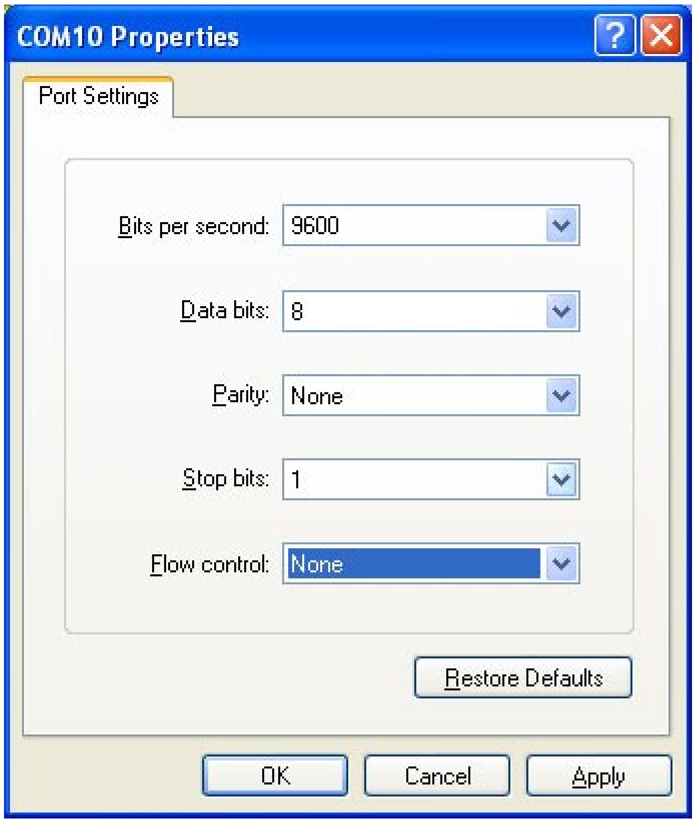
When we open it up, it should ask us for a new connection. Lets name it "xbee"



Next We will select the COM port from the drop down menu, in my case its COM4.


Connect To
xbee
Enter details for the phone number that you want to dial:
Country/region: United States (1)
Area code: 212
Phone number:
Connect using: COM1
COM1
COM4
TCP/IP (Winsock)

Next, set the properties. We select **9600 bps, 8 bit, No parity, 1 stop bit and no flow control.** Some programs may call this (9600 8N1). If the XBee has been configured for a different baud rate, of course, we should use that.


COM10 Properties

Port Settings

Bits per second: 9600

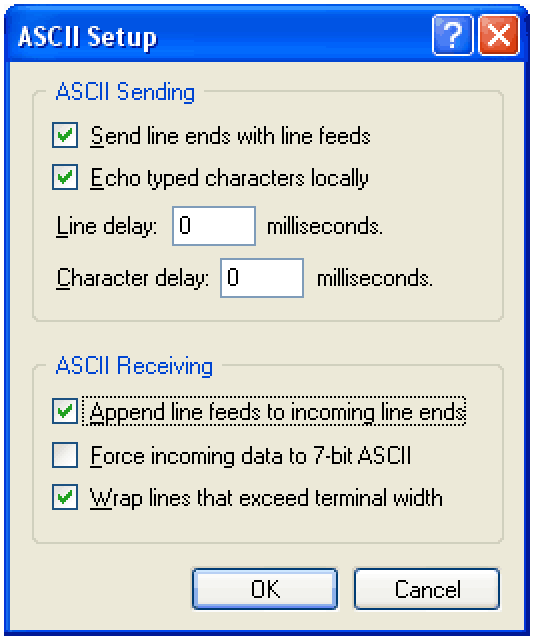
Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

We will get a blank screen that says "Connected" in the bottom left corner. Now, change the setup by selecting **File->Properties** and then going to the **Settings** tab and clicking the **ASCII Setup** button. Make sure we are sending line ends with line feeds and also echoing local characters


ASCII Setup

ASCII Sending

- [x] Send line ends with line feeds
- [x] Echo typed characters locally

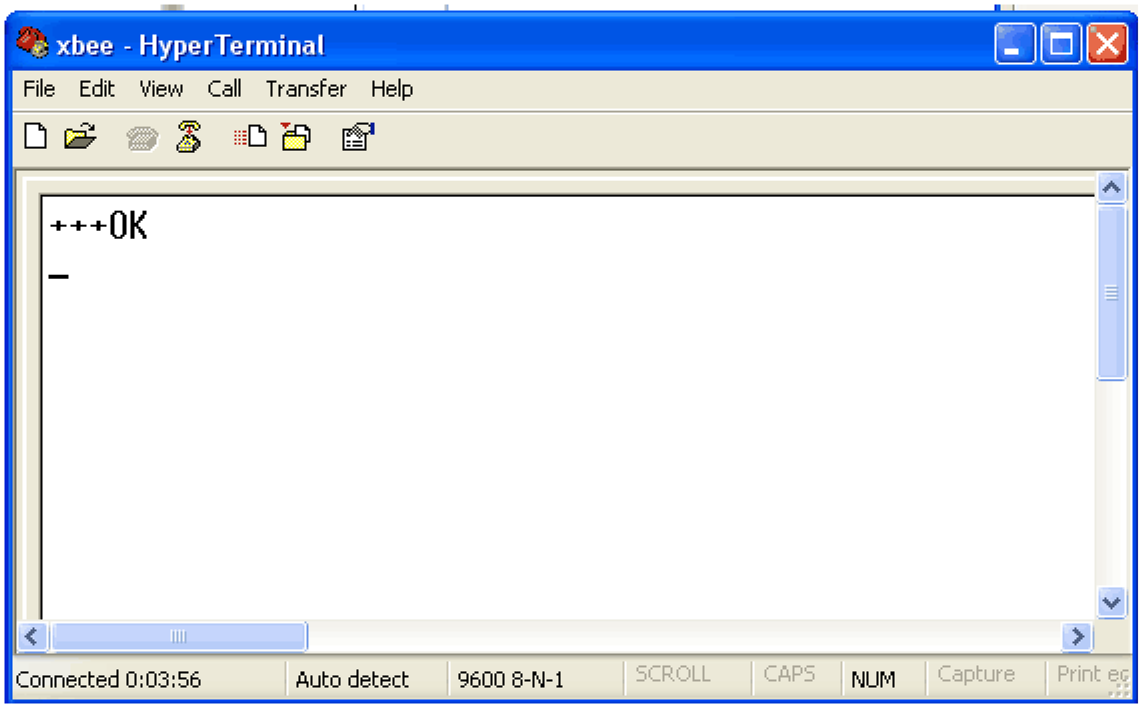
Line delay: 0 milliseconds.

Character delay: 0 milliseconds.

ASCII Receiving

- [x] Append line feeds to incoming line ends
- [ ] Force incoming data to 7-bit ASCII
- [x] Wrap lines that exceed terminal width

Now type in +++ (three plus signs) in quick succession. If the XBee is connected up properly we will get an **OK** in response

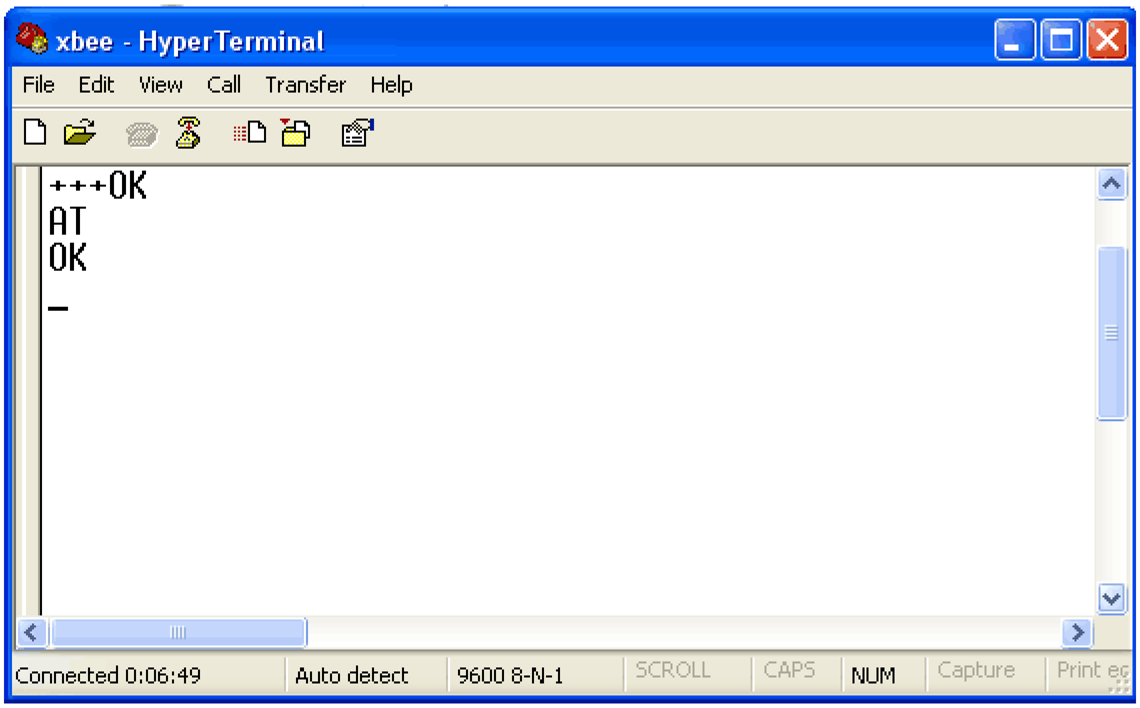


```
+++OK
_
```

Connected 0:03:56   Auto detect   9600 8-N-1

If we got an OK that means the XBee is powered and wired up correctly! If its not working, check:

1. Try again, be sure to wait 10 seconds between each attempt at typing in +++ and type the +'s quickly
2. Is the module powered? Green LED should be blinking
3. Are RX & TX swapped?
4. Do we have the correct baud rate? By default it should be 9600 baud 8N1 no hardware handshake but if it has been used for something else the baud rate might be different.



```
+++OK
AT
OK
_
```

Next try typing in +++ (receive **OK**) and then **AT** and press return to get another **OK** This is basically how we can configure the XBee, by sending it AT commands (they all start with AT for ATtention). After a while, the XBee times out of configuration mode and goes back to pass-through connection mode. So if we want to get back to config mode, just type in +++ and it will start responding again.

Configuring Xbee for the project:

In Robot, we need to fix the Xbee with the destination address as the Base station xbee. Similarly in all other Swarm Robots all Xbee will contain the base station Xbee's Serial number as the Destination address.

Base station Xbee can be configured to transmit to anyAddress as we are going to use this communication as the one way i.e swarm robots to base station.