**Technocrats Task Report for Week 3 and Week 4**

**Wireless Communication Using Xbee:**

Zigbee:

Zigbee communication is built for setting up a communicaton between sensor and control devices. Zigbee is low-cost and low-powered mesh network widely deployed for controlling and monitoring applications where it covers 10-100 meters within the range.

The whole zigbee system comprises of the zigbee coordinator, router and the end device. Every system must have atleast one coordinator which acts as a root and bridge of the network. This coordinates the transmitting, receiving and storing the data acquired.End devices have limited functionality to communicate with the parent nodes such that the battery power is saved as shown in the figure. The number of routers, coordinators and end devices depends on the type of network such as star, tree and mesh networks.

Their communication is over a serial port.

Connecting Zigbee module and arduino:

First install the XCTU software.

Then upload the bare minimum program to the arduino.

Connecting the xbee module with the shield is more preferable because the connection is more complex as shown in the figure.

After the wirings are done. The xbee is operated in USB mode. Then connect the arduino , open XCTU software and press the Test / Query button to see if the computer can talk to the XBee module.

There will be a dialog stating OK if it is able to communicate with arduino. Update the firmware if required.

To add your XBee(s), click the “Add device” icon – – in the upper-left part of the window. This window also allows you to specify more specific serial characteristics like baud rate, data bits, and stop bits. Assuming this is the first time you’ve used your XBee, you can leave those settings alone. So make sure those values look just as they do in the image above and click Finish. Now, Click that new module. To add your second Xbee again click the add module button. There are MAC IDs printed on the board to cross-check(in XCTU) their identity.

Click the “Switch to Consoles” icon. This will switch from the configuration tab to the console. We can use the console to send characters to an XBee, which will route that character over-the-air to any other XBee it’s connected to.

If Two zigbees are connected to a single computer then, open a serial connection on each device by clicking the connect icon – . The icon will change, and the border of the console will turn green.Next, click into the left half of the console, and type a letter or number. You should notice that character echoed in a blue font (the hexadecimal digits on the right represent the ASCII value). Now click into the other XBee’s console. As long as it was open, you should see that same character, but red. Try typing a different character into the second XBee’s console, and you should see it work the other way.

Once the test is done it is ready to code.

Sample code:

#include <XBee.h>

#include <SoftwareSerial.h>

/\*

This example is for Series 1 XBee Radios only

Receives I/O samples from a remote radio with 16-bit addressing.

The remote radio must have IR > 0, at least one digital or analog input enabled

and DL set to the 16-bit address of the receiving XBee (the one connected to the Arduino).

This example uses the SoftSerial library to view the XBee communication. I am using a

Modern Device USB BUB board (http://moderndevice.com/connect) and viewing the output

with the Arduino Serial Monitor.

\*/

// Define NewSoftSerial TX/RX pins

// Connect Arduino pin 8 to TX of usb-serial device

uint8\_t ssRX = 8;

// Connect Arduino pin 9 to RX of usb-serial device

uint8\_t ssTX = 9;

// Remember to connect all devices to a common Ground: XBee, Arduino and USB-Serial device

SoftwareSerial nss(ssRX, ssTX);

XBee xbee = XBee();

Rx16IoSampleResponse ioSample = Rx16IoSampleResponse();

// 64-bit response is same except api id equals RX\_64\_IO\_RESPONSE and returns a 64-bit address

//Rx64IoSampleResponse ioSample = Rx64IoSampleResponse();

void setup() {

Serial.begin(9600);

xbee.setSerial(Serial);

// start soft serial

nss.begin(9600);

}

void loop() {

//attempt to read a packet

xbee.readPacket();

if (xbee.getResponse().isAvailable()) {

// got something

if (xbee.getResponse().getApiId() == RX\_16\_IO\_RESPONSE) {

xbee.getResponse().getRx16IoSampleResponse(ioSample);

nss.print("Received I/O Sample from: ");

nss.println(ioSample.getRemoteAddress16(), HEX);

nss.print("Sample size is ");

nss.println(ioSample.getSampleSize(), DEC);

if (ioSample.containsAnalog()) {

nss.println("Sample contains analog data");

}

if (ioSample.containsDigital()) {

nss.println("Sample contains digtal data");

}

for (int k = 0; k < ioSample.getSampleSize(); k++) {

nss.print("Sample ");

nss.print(k + 1, DEC);

nss.println(":");

for (int i = 0; i <= 5; i++) {

if (ioSample.isAnalogEnabled(i)) {

nss.print("Analog (AI");

nss.print(i, DEC);

nss.print(") is ");

nss.println(ioSample.getAnalog(i, k));

}

}

for (int i = 0; i <= 8; i++) {

if (ioSample.isDigitalEnabled(i)) {

nss.print("Digtal (DI");

nss.print(i, DEC);

nss.print(") is ");

nss.println(ioSample.isDigitalOn(i, k));

}

}

}

}

else {

nss.print("Expected I/O Sample, but got ");

nss.print(xbee.getResponse().getApiId(), HEX);

}

}

else if (xbee.getResponse().isError()) {

nss.print("Error reading packet. Error code: ");

nss.println(xbee.getResponse().getErrorCode());

}

**A Better Alternative:**

A better alternative to zibgee communication can be communication over ESP8266 on which I have been working in the vacction. This module connects to the WiFi. This is also a power consuming module. If this can be connected to a WiFi network the data acquired can be uploaded to the cloud service and data analysis could be made. This may help in training the bot which makes it more efficient. It is also very easy to interface with arduino(not as complex as Xbee). Only Arduino IDE is sufficient. Platforms like Thingspeak and matlab can be interfaced with arduino and communication can be done.