Experiment No. – 8

Title: Communicate between Arduino and Raspberry PI using any wireless medium like ZigBee

Aim: Interfacing of Arduino and Raspberry PI using any wireless medium like ZigBee

Hardware Requirements: Raspberry Pi 3 B^{+}, Arduino Uno, ZigBee Module with shield

Software Requirements: Arduino IDE, X-CTU Software

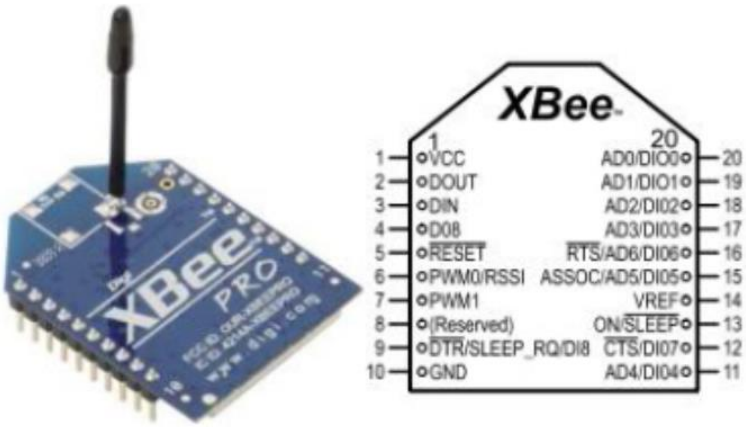
Theory:

Zigbee is a wireless communication protocol targeted for battery powered devices (it has both low power and low cost). It generally operates in the 2.4GHz range (although there are geographic variations), and supports data ranges from 20 to 250 kbits/s.

The transmission distance though, is small compared to the likes of LoRa. It is 10 to 100 m, whereas LoRa can transmit over a few kilometers. Another thing to note is that Zigbee communication doesn't work very well if there is no line of sight between transmitter and receiver.

Even minor obstacles have been observed to significantly degrade the communication. Keep these limitations in mind when using Zigbee. You may want to look out for other options if your application can't meet these constraints.

In order to make Zigbee work with Arduino, we will use the XBee module.



The DOUT and DIN pins in the figure above are the UART pins (TX and RX). They can be connected to two digital pins of Arduino (if you plan to use SoftwareSerial), or else to pins 0 and 1 of Arduino respectively (if you plan to use HW Serial).

Configuring the XBee modules:

The XBee modules (transmitter and receiver) need to be configured using the X-CTU Software. It can be downloaded from here. This software is provided by DigiKey, and they have given a detailed configuration guide.

Please note that the two XBee modules that intend to communicate with each other should belong to the same series.

Here are a few things to note about the configuration −

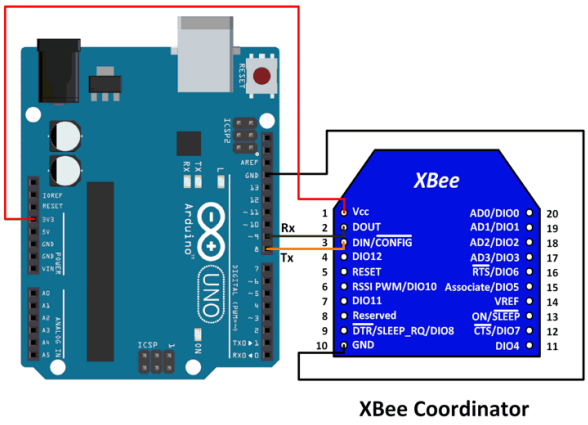
You will need a breakout board or an Explorer with a USB to UART converter for this configuration.

The PAN ID (Personal Area Network ID) has to be the same for the devices that want to communicate with each other.

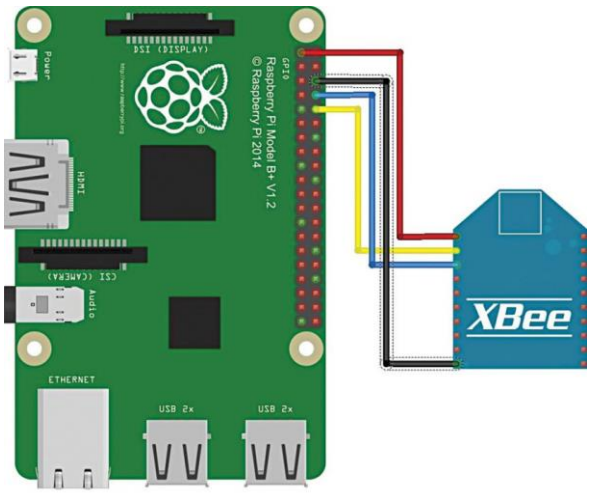
One module needs to be set as the transmitter and the other as the receiver (this is determined by the CE field).

Note the baud rate that you set. This will be used in the Arduino code, when configuring the Serial communication with XBee.

Circuit Diagram:



Interfacing of XBee Module with Arduino



Interfacing of XBee Module with Raspberry Pi

Test Setup:
XBee: configured two xbee, one as Coordinator and another as Router. Both in API mode 2 (AP =2). I used XCTU to configure both the device. Only reason to choose API 2 is because the Arduino library I used only support API mode 2.
Raspberry pi: connected Coordinator XBee to one of Raspberry Pi.
Arduino Uno: connected the Router xbee to one of my Arduino. The connection is pretty simple as below.

- XBee Rx –> Arduino Tx
- XBee Tx -> Arduino Rx
- XBee 3.3v-> Arduino 3.3v
- XBee Gnd –>Arduino Gnd

We have connected Vcc to 3.3V on Arduino, GND to GND, DOUT (TX) to pin 2, which will act as RX on the Arduino, and DIN (RX) to pin 3, which will act as TX on the Arduino.

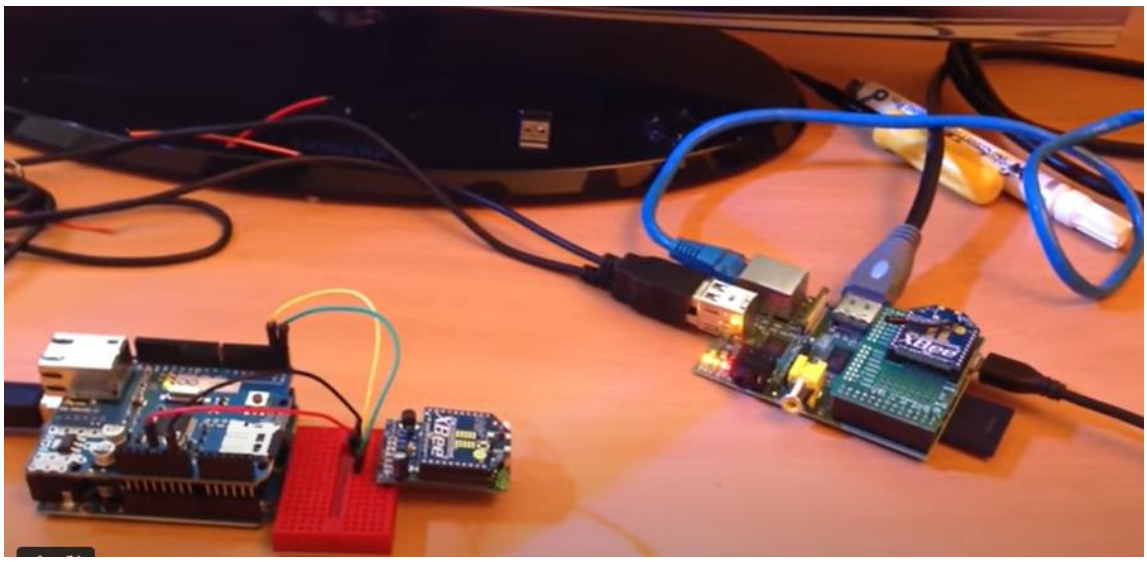
The connections will be similar on the receiving side as well. If you have an on-board antenna, else you'll have to connect an antenna to the UFL connector.

Over here, whatever is received from XBee is forwarded to the Serial Monitor. Thus, when testing out the combined system, whatever you type on the Serial Monitor on the transmitter side should be printed on the Serial Monitor on the receiver side.

Raspberry Pi Node js code

Modules used

- xbee-api: npm install xbee-api
- serialport: npm install serialport



Conclusion:

Code:

XbeeTx:-

```
#include <SoftwareSerial.h>

#define rxPin 6
#define txPin 7

// Set up a new SoftwareSerial object
SoftwareSerial mySerial =  SoftwareSerial(rxPin, txPin);

void setup()  {
  Serial.begin(9600);
    // Set the baud rate for the SoftwareSerial object
   mySerial.begin(9600);
}

void loop() {
 unsigned int temp = analogRead(A0);
 temp = map(temp,0,1023,0,500);    //10mV = 1'C

 mySerial.println(temp);
 delay(1000);
}
```

XbeeRx:-

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(6, 7); // RX, TX

void setup()
{
  Serial.begin(9600);
  while (!Serial) {
  }

  Serial.println("hello");

  mySerial.begin(9600);
  mySerial.println("Hello, world?");
}

void loop()
{
  if (mySerial.available())
    Serial.write(mySerial.read());
  if (Serial.available())
    mySerial.write(Serial.read());
}
```