Experiment 3.3

Student Name: Kanishk Soni **UID: 20BCS9398**
Branch: CSE **Section/Group: 20BCS_DM-708/B**
Semester: 6th **Subject Name: IOT LAB**
Subject Code:20CSP-358

Aim:

Study the Implementation of Zigbee Protocol using Raspberry Pi/Arduino.

Objectives:

- Learn about interfacing.
- Learn about IoT programming.
- Learn about XBee Pro S2 Transceiver

Components Required:

You will need the following components –

- 2 XBee Pro S2 Transceiver
- 2 UART to USB adapter board
- 1 USB Cord

About Zigbee:

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.

Fig1: XBee Module

Configuring the XBee modules:

The XBee modules (transmitter and receiver) need to be configured using the X-CTU Software. It can be downloaded from XTCU6.3, this software is provided by DigiKey, and they have given a detailed configuration guide. Therefore, there is no point of me reinventing the wheel here.

There's another one by Sparkfun that is adapted to the newer version of the X-CTU software.

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:

Once your XBee is configured, you can connect it to the Arduino via the breakout/Explorer board. In that case, the pinout will be slightly different depending on which board/ Explorer you are using. Here we will assume you are connecting the XBee directly to Arduino Uno, in which case, the connections will be −Connect the BT module's Rx pin to pin 11 on the Arduino

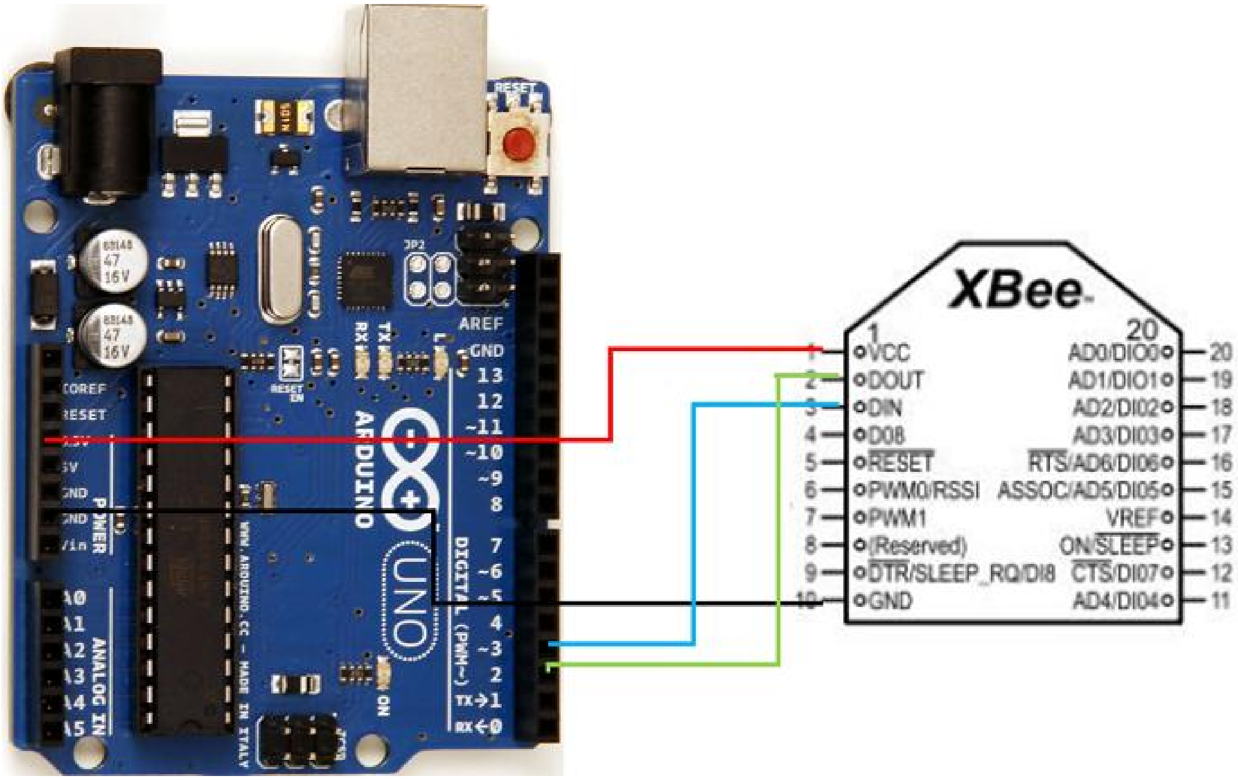


Fig2: Circuit Diagram

As you can see, 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, that's good, else you'll have to connect an antenna to the UFL connector.

Arduino Code:

```
#include <SoftwareSerial.h>
SoftwareSerial xbeeSerial(2,3); //RX, TX

void setup() {
  Serial.begin(9600);
  xbeeSerial.begin(9600);
}

void loop() {
  if(xbeeSerial.available() > 0){
    char input = xbeeSerial.read();
    Serial.print(input);
    }
}
```

Output:

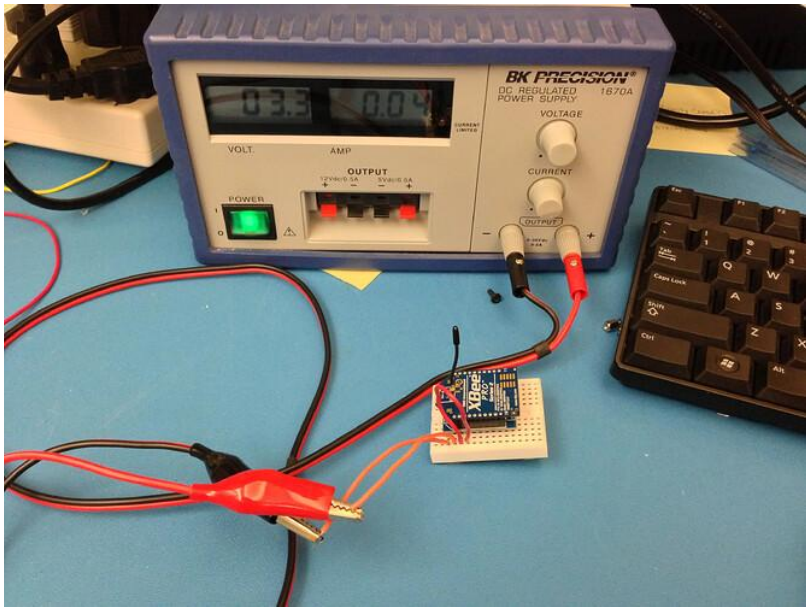
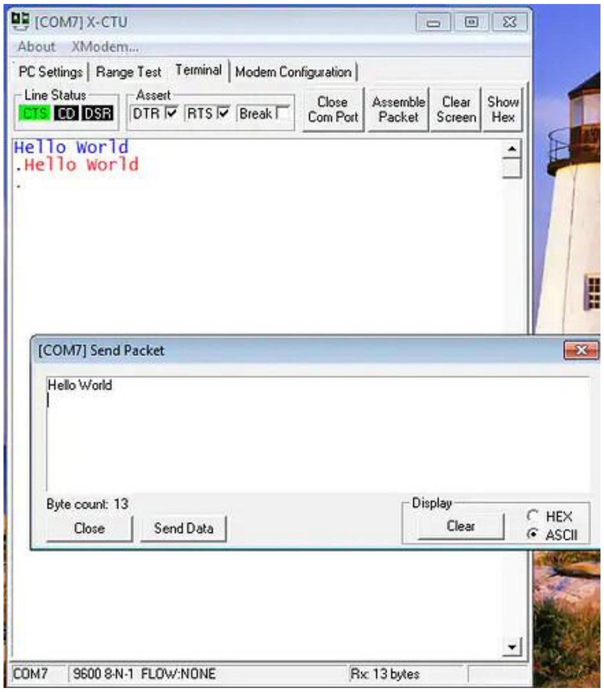


Fig3. XBee setup



```
Hello World
.Hello World
.
```

Hello World

Byte count: 13

COM7 9600 8-N-1 FLOW:NONE Rx: 13 bytes

Fig4: Packet Sent via XBee

Learning outcomes:

- Learnt about MQ-05.
- Learnt how to interface and limitations of MQ-05 sensors about its range.
- Learnt the basic features of IoT programming