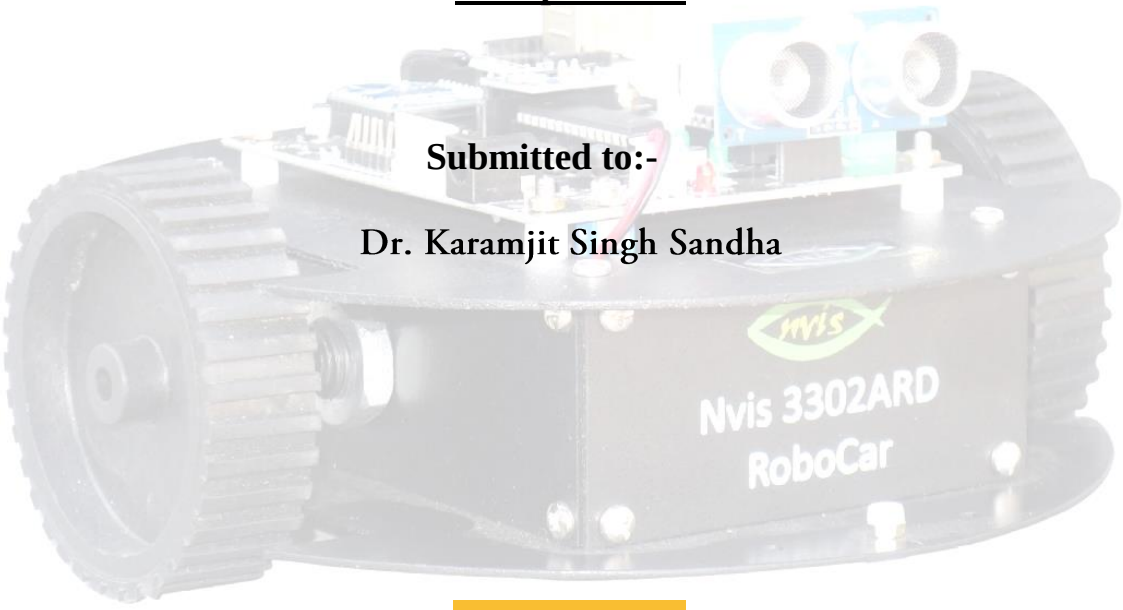
BUGGY PROJECT

An Interim Report submitted

for UTA 011-Engineering Design –III
(Second Year)

Submitted by :-

SHIVAM SHARMA(101503208)
Group:COE-9



Submitted to:-

Dr. Karamjit Singh Sandha

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT
THAPAR UNIVERSITY, PATIALA-147004, PUNJAB
INDIA
May 2017

THEORY

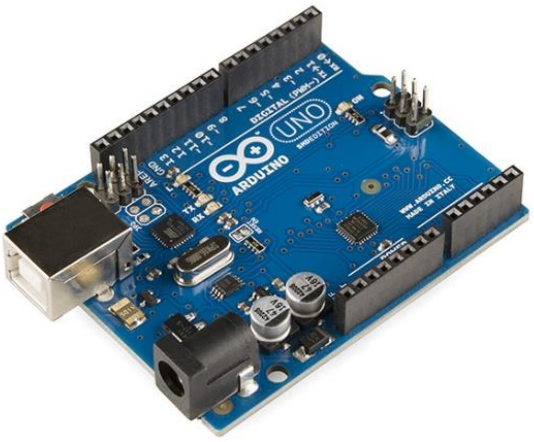
ARDUINO UNO BOARD DETAILS

What is a Microcontroller?

A micro-controller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals The important part for us is that a micro-controller contains the processor (which all computers have) and memory, and some input/output pins that you can control. (often called GPIO - General Purpose Input Output Pins).

Technical specs:

- Microcontroller ATmega328
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V
- Input Voltage (limits) 6-20V Digital
- I/O Pins 14 (of which 6 provide PWM output)
- Analog Input Pins 6
- DC Current per I/O Pin 40 Ma
- DC Current for 3.3V Pin 50 mA
- Clock speed 16 MHz
- Flash Memory 32 KB of which 0.5 KB.
- SRAM 2 KB
- EEPROM 1 KB



PIN CONFIGURATION:

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

The power pins are as follows:

• **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

• **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

• **3.3V**. A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

• **GND**. Ground pins.

• **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip. • External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

• **PWM:** 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite () function.

• **LED: 13**. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off

HOW TO USE ARDUINO SOFTWARE?

• Download Arduino software on your system.

• The coding is similar to c programming language.

• Code the program.

• Compile it.

• Change the serial port.

• After that upload it to the Arduino device via usb cable.

ADVANTAGES OF ARDUINO TECHNOLOGY

• It is cheap

. • It has an open supply hardware feature that permits users to develop their own kit

• The software of the Arduino works well with all kinds of in operation systems like Linux, Windows, and Macintosh, etc.

• It also comes with open supply software system feature that permits tough software system developers to use the Arduino code to merge with the prevailing programing language libraries and may be extended and changed.

• It is user friendly

EXPERIMENT 1

AIM:

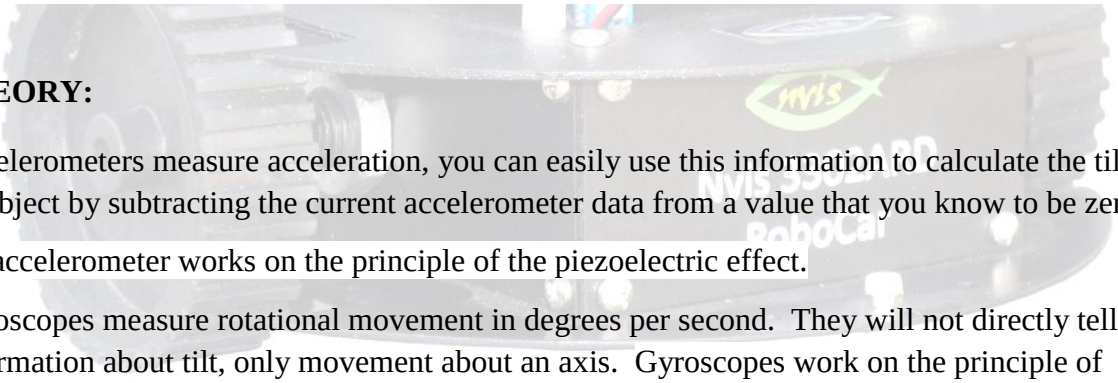
To interface gyroscope and accelerometer sensor with Arduino board

HARDWARE:

- MPU6050
- Jumper wires
- Arduino board
- Usb cable

SOFTWARE:

1. Arduino IDE: Arduino



THEORY:

Accelerometers measure acceleration, you can easily use this information to calculate the tilt of an object by subtracting the current accelerometer data from a value that you know to be zero tilt.

An accelerometer works on the principle of the piezoelectric effect.

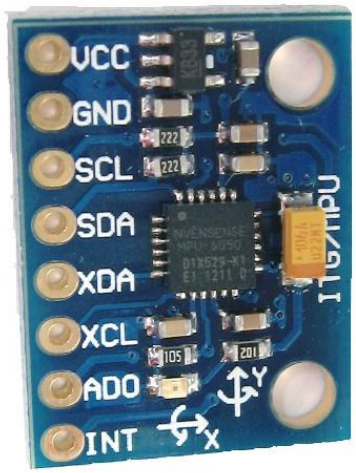
Gyroscopes measure rotational movement in degrees per second. They will not directly tell you information about tilt, only movement about an axis. Gyroscopes work on the principle of Coriolis acceleration.

The InvenSense MPU-6050 sensor contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefor it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino.

Applications:

- Gestures and Movements Detection
- Motion-activated user interface
- Gaming Human Interface
- Navigation Boards

- Platform Stability
- Accurate angular-rate detection



CODE:

```
// MPU-6050 Short Example Sketch

//By Arduino User JohnChi

// August 17, 2014

// Public Domain

#include

<Wire.h>

const int MPU_addr=0x68;

// I2C address of the MPU-6050

int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

void setup()

{

Wire.begin();

Wire.beginTransmission(MPU_addr);

Wire.write(0x6B);  // PWR_MGMT_1 register

Wire.write(0);    // set to zero (wakes up the MPU-6050)

Wire.endTransmission(true);

Serial.begin(9600);

}
```

```
void gyro()

{

Wire.beginTransmission(MPU_addr);

Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers

AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)

AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)

AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)

Tmp=Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)

GyX=Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)

GyY=Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)

GyZ=Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

Serial.print("AcX = ");

Serial.print(AcX);

Serial.print(" | AcY = ");

Serial.print(AcY);



Serial.print(" | AcZ = ");

Serial.print(AcZ);
```

```
Serial.print(" | Tmp = ");

Serial.print(Tmp/340.00+36.53);

Serial.print(" | GyX = "); Serial.print(GyX);

Serial.print(" | GyY = "); Serial.print(GyY);

Serial.print(" | GyZ = "); Serial.println(GyZ);

//delay(100);

}

void loop()

{

gyro();

}
```

PIN DIAGRAM:

VDD : 5V on UNO

XDA

XCL

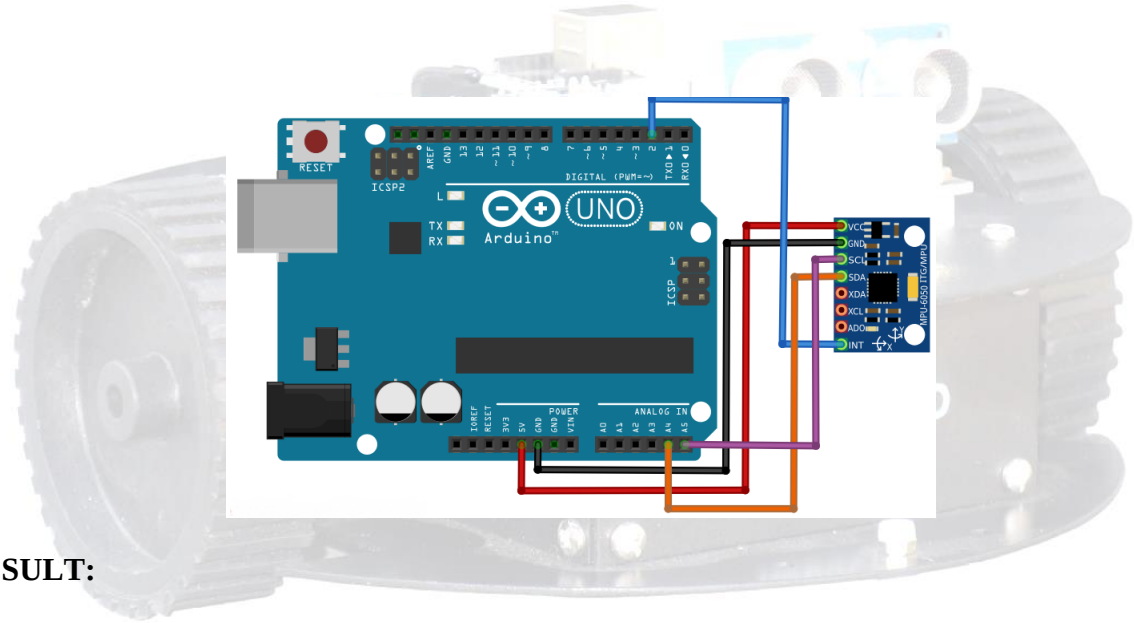
ADO

GND : GND on UNO
SCL : A5 on UNO

SDA : A4 on UNO
INT : Digital Pin 2 on UNO

CONNECTION DIAGRAM WITH ARDUINO:

The MPU 6050 communicates with the Arduino through the I2C protocol. The MPU 6050 is connected to Arduino as shown in the following diagram. If your MPU 6050 module has a 5V pin, then you can connect it to your Arduino's 5V pin. If not, you will have to connect it to the 3.3V pin. Next, the GND of the Arduino is connected to the GND of the MPU 6050. The program we will be running here, also takes advantage of the Arduino's interrupt pin. Connect your Arduino's digital pin 2 (interrupt pin 0) to the pin labeled as INT on the MPU 6050. Next, we need to set up the I2C lines. To do this, connect the pin labeled SDA on the MPU 6050 to the Arduino's analog pin 4 (SDA) and the pin labeled as SCL on the MPU 6050 to the Arduino's analog pin 5 (SCL).



RESULT:

We learnt about the features that are associated with the sensor MPU6050. We could see how the coordinates change on moving the sensor. The results were clearly visible on the serial monitor window.

EXPERIMENT 2

AIM:

To control buggy movements like moving forward, backward, rotation it clockwise and counter clockwise using zigbee.

HARDWARE:

THEORY:

Zigbee is a wireless communication module which use IEEE 802.15.4 standard. 802.15.4 is a IEEE standard for low power applications of radio frequency. It used in many products now a day for wireless communication functionality. It can be used as a transmitter and receiver both. It used serial communication to send and receive data. It has two series, series1 and series 2. Series 1 is comparatively easy to use and it is recommended for beginners. Series 1 zigbee module cannot work in mesh network. Mean it cannot talk to more than one zigbees buddies.

As I have already mentioned it use serial port to send and receive data. So its mean it can be easily interface with Arduino Uno R3, any type of microcontroller and computer. Because they all support serial communication and they all have serial port to send and receive data. It can also communicate with other Zigbee to form a mesh. Zigbee can also be used to make a local area network. It has many applications. But some of the famous applications of Zigbee is given below.

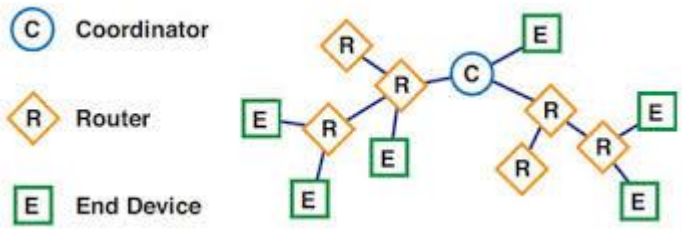
Applications of Zigbee:

- Wireless communication
- wirelessly controlled robot
- Remote monitoring system
- Wireless home automation system
- wireless temperature sensor and many others.

Zigbee alone can't do anything. You have to interface it with some intelligent device like microcontrollers, Arduino and computer. These devices will tell it what to do or what no to do through already fed program inside microcontrollers and Arduino Uno R3. These digital devices are no such intelligent. But you can make them intelligent by writing few lines of instructions. Let's move forward and learn how to interface Zigbee with Arduino.

- **Zigbee Network Formations:**

Zigbee defines three difference device types: coordinator, router and end device.



Coordinator: Start a new personal area network (PAN) by selecting the channel and PAN ID. Allow routers and end devices to join the PAN, transmit and receive RF data transmission and route the data through the mesh network. In charge of setting up the network. Can never sleep.

Router: Transmit and receive RF data transmission, and route data packet through the network. Can relay signals from other routers/EPs. Can never sleep.

End Device: Cannot assist in routing the data transmission but transmit or receive RF data transmission and intended to be battery powered devices. Cannot relay signals. Can sleep to save power

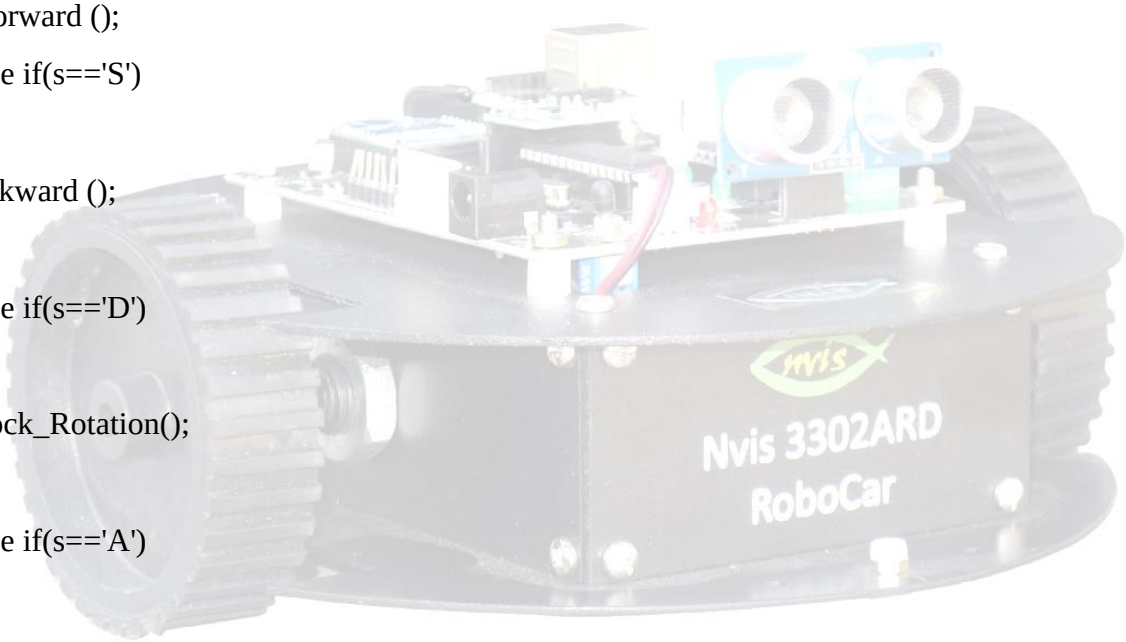
Transceiver: Transceiver is Transmitter and Receiver as one module, XBee can transmit data and receive data. Arduino(A) transmit data to XBee(A) via wire and XBee(A) transmit that data wirelessly to XBee(B); upon receiving data from XBee(A) wirelessly, XBee(B) will then transmit that data to Arduino(B) via wire. Same applied when data is transmitted from Arduino (B).

CODE:

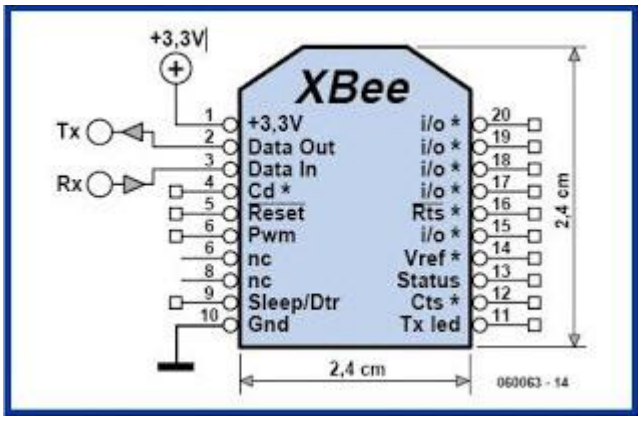
```
void setup ()
{
  Serial.begin(9600);   //serial initlization here we are using 9600 band rates.
  pinMode(5, OUTPUT);   // Right +ve)
  pinMode(6, OUTPUT);   // Right -ve
  pinMode(7, OUTPUT);   // Left -ve
  pinMode(8, OUTPUT);   // Left +ve
}
//functions for forward, backward, Antiprotection, Clock Rotation and stop movments
void forward()
{
  digitalWrite (5, HIGH);  // Right +ve
  digitalWrite (6, LOW);    // Right -ve
```

```
  digitalWrite (7, LOW);    // Left -ve
  digitalWrite (8, HIGH);   // Left +ve
}
void Backward()
{
  digitalWrite (5, LOW);
  digitalWrite (6, HIGH);
  digitalWrite (7, HIGH);
  digitalWrite (8, LOW);
}
void Anti_Rotation()
{
  digitalWrite (5, HIGH); // turn the LED on (HIGH is the voltage level)
  digitalWrite (6, LOW); // turn the LED on (HIGH is the voltage level)
  digitalWrite (7, HIGH); // turn the LED on (HIGH is the voltage level)
  digitalWrite (8, LOW); // turn the LED on (HIGH is the voltage level)
}
void Clock_Rotation ()
{
  digitalWrite (5, LOW); // turn the LED on (HIGH is the voltage level)
  digitalWrite (6, HIGH); // turn the LED on (HIGH is the voltage level)
  digitalWrite (7, LOW); // turn the LED on (HIGH is the voltage level)
  digitalWrite (8, HIGH); // turn the LED on (HIGH is the voltage level)
}
void stop ()
{
  digitalWrite (5, LOW); // turn the LED on (HIGH is the voltage level)
  digitalWrite (6, LOW); // turn the LED on (HIGH is the voltage level)
  digitalWrite (7, LOW); // turn the LED on (HIGH is the voltage level)
  digitalWrite (8, LOW); // turn the LED on (HIGH is the voltage level)
```

```
}
// the loop function runs over and over again forever
void loop ()
{
  char s=' ';
  if (Serial. Available ()>0) //here we check serial data on serial port and then read it.
  {
    s=Serial. Read ();
    if(s=='W')
    forward ();
  else if(s=='S')
{
Backward ();
}
  else if(s=='D')
 {
 Clock_Rotation();
}
  else if(s=='A')
  {
 Anti_Rotation();
  }
  s=' ';
  }
}
```



PIN DIAGRAM:

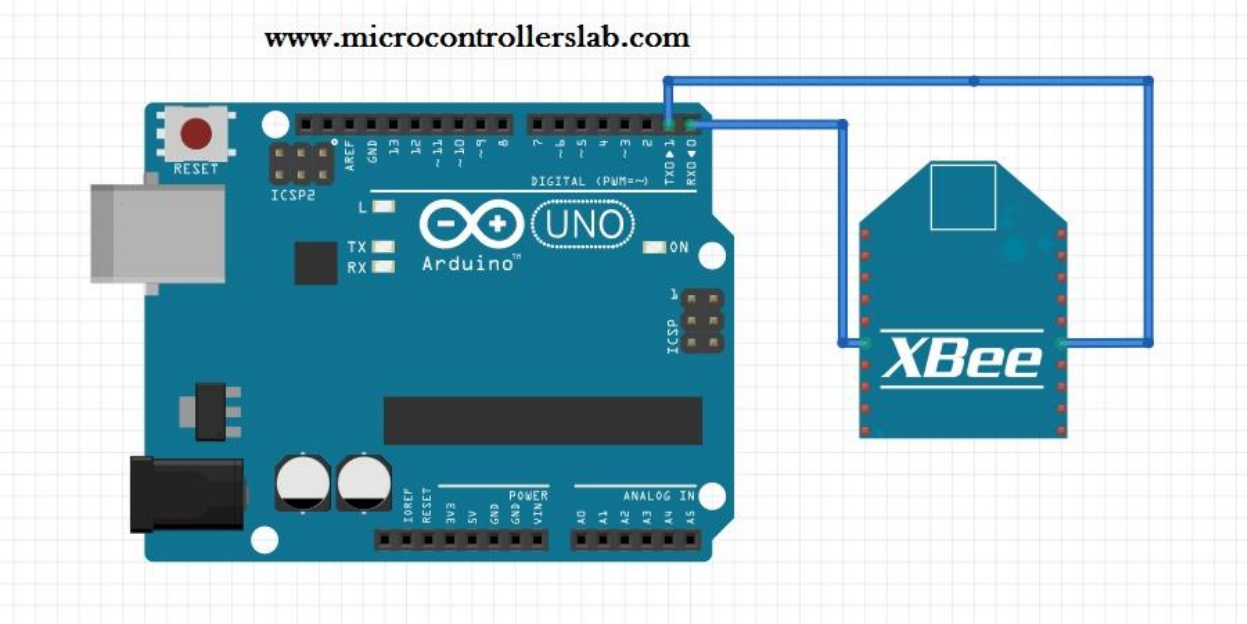


INTERFACING WITH ARDUINO:

Two Zigbee modules can talk with each other if both are of same type. To communicate to it module with each other, obtain to its modules. Connect one module to Arduino and other module to either sensor or any microcontroller or computer.

API and AT Modes:

The XBee modules can be configured in two ways: Transparent Mode (AT) and API Mode (API).In AT mode you are limited to point-to-point communication between two XBees. In API mode, we can trivally send and receive from both the COORDINATOR and many XBees out in the world. Additionally, API mode will expose a variety of additional information encoded in each packet.



RESULT:

We understood the reason of calling coordinator as master and router as slave. We could control the movements in buggy for eg on pressing W, S, D, A from the keyboard our buggy moved forward, backward, rotated clockwise and anticlockwise respectively.