BUGGY PROJECT

An Interim Report submitted

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

Submitted by

ANKUR SHARMA
Roll No. - 101510016
Group: CML-1

Supervisor

Dr. Rana Pratap Yadav
Assistant Professor
Electronics & Communication Engineering
Thapar University

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT
COMPUTER SCIENCE ENGINEERING DEPARTMENT
THAPAR UNIVERSITY, PATIALA-147004, PUNJAB
INDIA

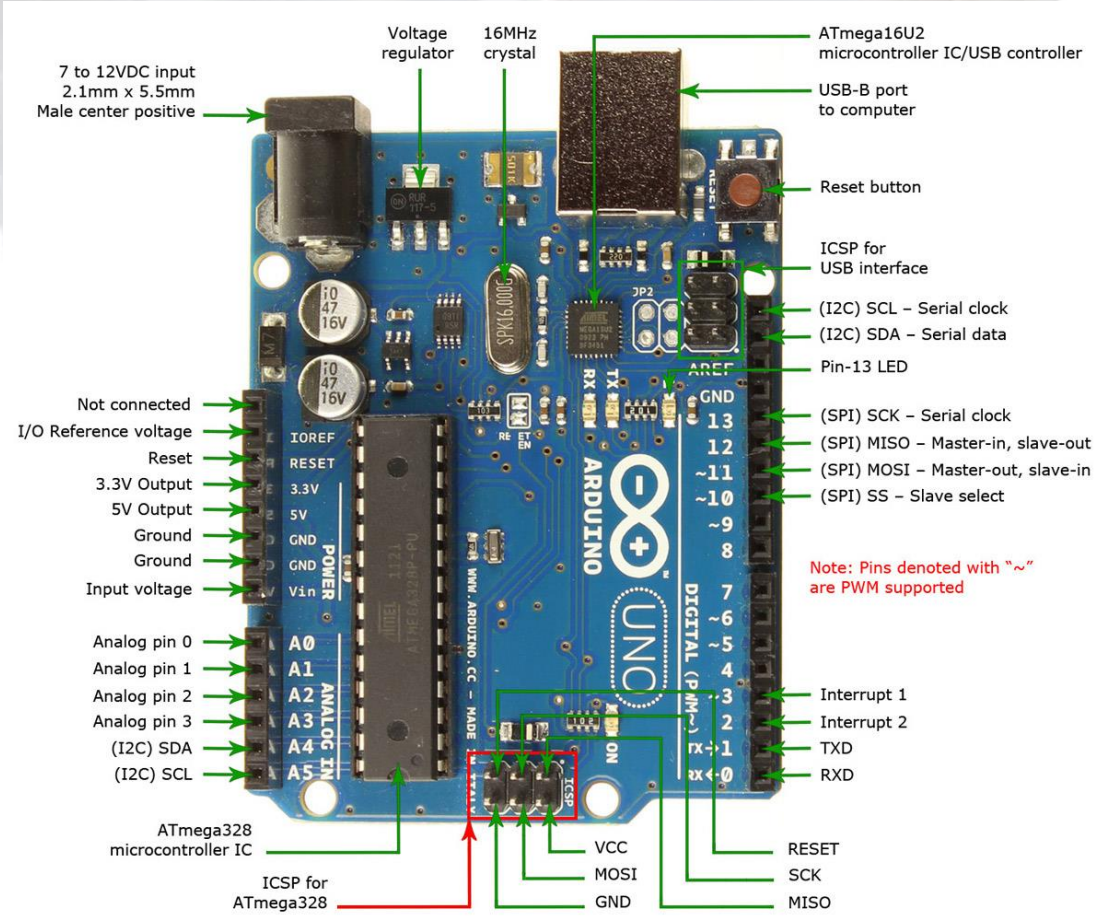
May 2017

INDEX

ARDUINO BOARD UNO

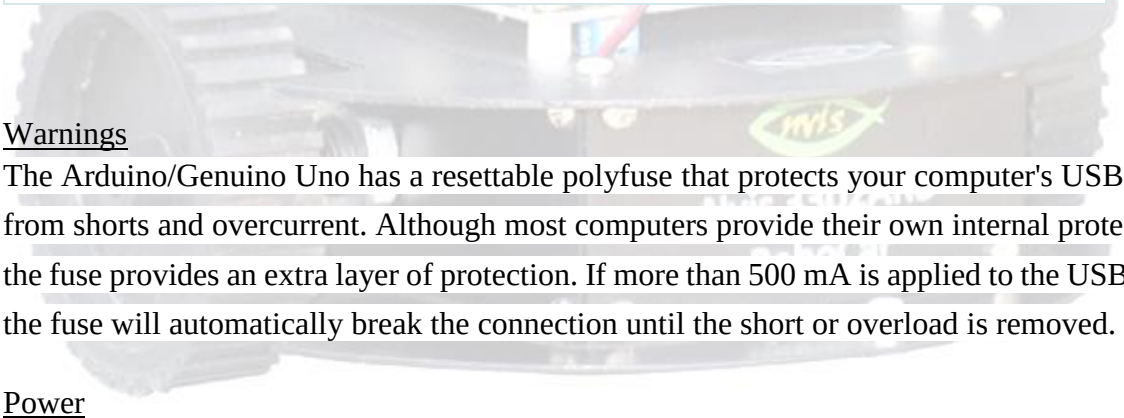
Arduino/Genuino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.



Technical specs (UNO Board)

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g



Warnings

The Arduino/Genuino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Power

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

An explanation of Arduino board:

Digital IO pins (pins 0–13)

These can be inputs or outputs, which is specified by the sketch you create in the IDE.

Analogue In pins (pins 0–5)

These dedicated analogue input pins take analogue values (i.e., voltage readings from a sensor) and convert them into a number between 0 and 1023.

Analogue Out pins (pins 3, 5, 6, 9, 10, and 11)

These are actually six of the digital pins that can be reprogrammed for analogue output using the sketch you create in the IDE.

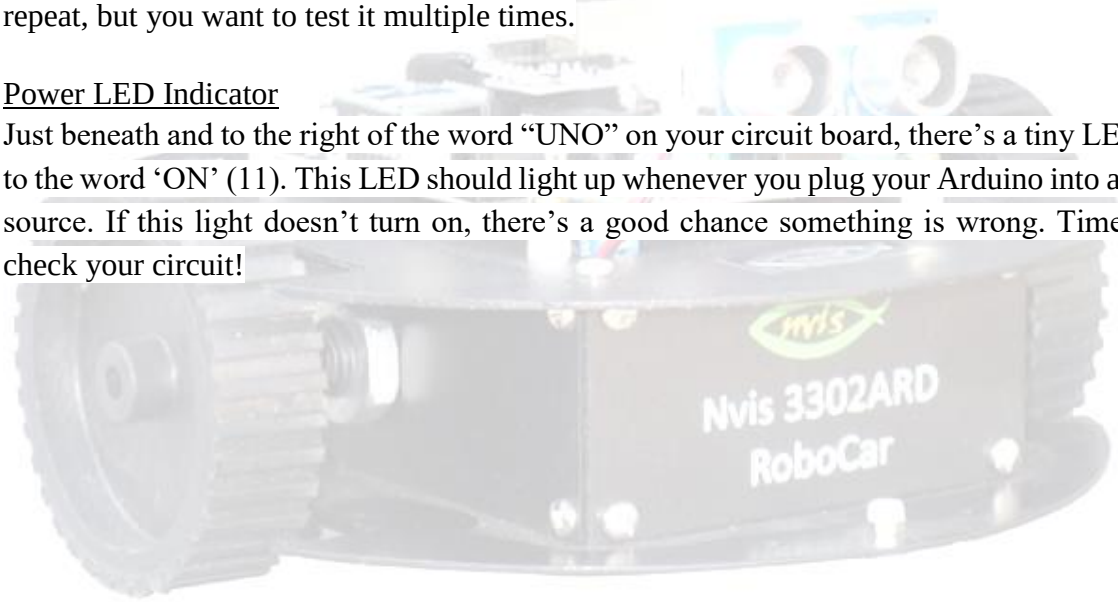
The board can be powered from your computer's USB port, most USB chargers, or an AC adapter (9 volts recommended, 2.1mm barrel tip, center positive). If there is no power supply plugged into the power socket, the power will come from the USB board, but as soon as you plug a power supply, the board will automatically use it.

Reset Button

The Arduino has a reset button (10). Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times.

Power LED Indicator

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' (11). This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!



Nvis 3302ARD RoboCar

Nvis 3302ARD RoboCar is an electro-mechanical platform with the capability of sensing the environment, processing the data, and acting according to the preprogrammed sequence. It is a miniature prototype car powered by batteries whose motion is controlled by microcontroller. Various Sensors can be interfaced like IR (Infrared) Sensor, Ultrasonic Sensor, Analog Sensor, and many more.

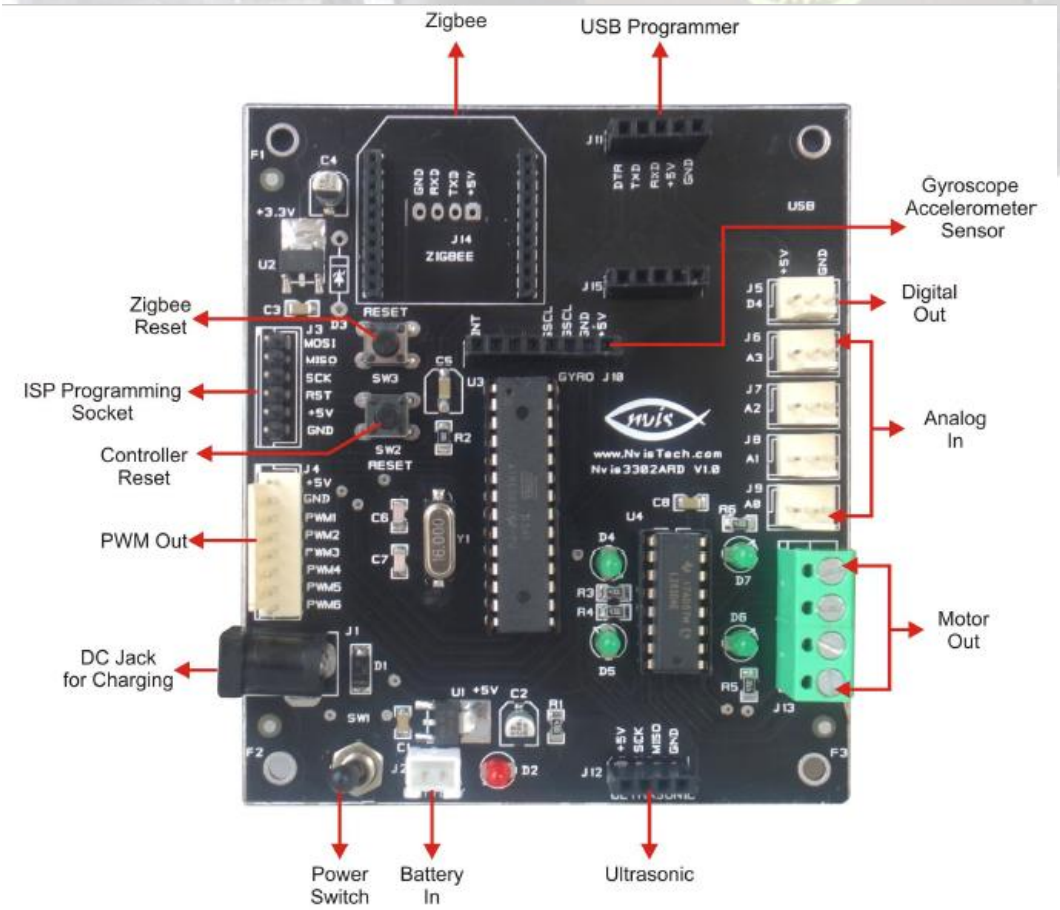
It is a multitasking Robocar that can perform actions such as line follower, obstacle detection with wireless operation with DC motor drive. Robocar displays all action on LCD. Nvis 3302ARDRoboCar is a Robot designed for Robotics students. It will help them to get comfortable with the world of Robotics and Embedded Systems.
Nvis 3302ARD, RoboCar is an ideal platform to enhance education, training, skills & development amongs our young minds.

Nvis 3302ARDis a ready assembled Robotic Buggy consisting of FR4 chassiswheels with different Sensor modules mounted on it.NvisRoboCaris capable of sensing environment using various sensor modules andacts accordingly. The machine is driven by DCmotors which are powered by rechargeable batteries. This Nvis 3302ARD isAtmega328P Microcontroller (Arduino based) RoboCar, is designed for users to startdeveloping smart robot which is capable of accelerometer balancing, Gyroscopeangular velocity sensing, Ultrasonic obstacle avoiding/detecting and distance measure and many more. There is Zigbee for wireless control your smart RoboCar.

Product Features:

- 16 x 2character LCD interface
- Ultrasonic Sensor interface
- Infrared(IR) Transmitter interface
- Buzzer interface
- IR Receiver interface
- DC Motor interface & control
- Switch interface
- Expansion connectors
- Onboard battery charger
- PC based programming
- RF module interface(Optional)
- Analog Sensor interface (Optional)

PCB Details:



Input Output Devices of RoboCar are:

EXPERIMENT 1

AIM:

To interface gyroscope and accelerometer sensor with Arduino board

HARDWARE:

- MPU6050
- Jumper wires
- Arduino board
- USB cable

SOFTWARE:

Arduino IDE: Arduino

THEORY:

MPU-6050 is an IMU(**Inertial Measurement Unit**) with 6 DOF(Degrees of Freedom) or a six axis IMU sensor, which means that it gives six values as output., 3-axis gyroscope and 3-axis accelerometer.

The MPU 6050 is a sensor based on **MEMS (Micro Electro Mechanical Systems)** technology with both the accelerometer and the gyroscope embedded inside a single chip. This chip uses I2C (Inter Integrated Circuit) protocol for communication.

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**.

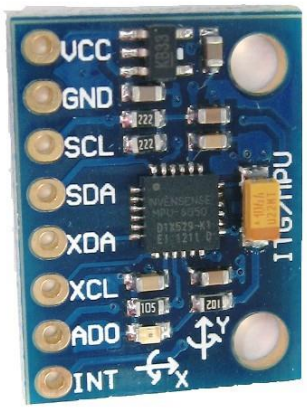
Gyroscopes measure rotational movement in degrees per second. They will not directly tell you information about tilt, only movement about an axis. Gyroscopes work on the principle of **Coriolis acceleration**.

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.

The full scale ranges of the accelerometer and gyroscope are user programmable and they are +/- 2g, 4g, 8g and 16g for the accelerometer and +/- 250 °/S, 500 °/S, 1000 °/S and 2000 °/S.

APPLICATIONS:

- Gestures and Movements Detection
- Motion-activated user interface
- Gaming Human Interface
- Navigation Boards
- Platform Stability
- Accurate angular-rate detection



CODE:

```
#include<Wire.h>
#include<LiquidCrystal.h>
const int MPU=0x68;                    //I2C address of MPU
int GyX,GyY,GyZ;
float pitch=0;
float roll=0;
float yaw=0;
float v_pitch;
float v_roll;
float v_yaw;
float a_pitch;
float a_roll;
float a_yaw;
void setup()
{
Wire.begin();
Wire.beginTransmission(MPU);
Wire.write(0x6B);                      //power management register 1
Wire.write(0);
Wire.endTransmission(true);
Serial.begin(9600);
}
 void loop()
{
Wire.beginTransmission(MPU);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU,6,true);          //requests 6 registers
GyX=Wire.read()<<8|Wire.read();
GyY=Wire.read()<<8|Wire.read();
GyZ=Wire.read()<<8|Wire.read();
v_pitch=(GyX/131);
if(v_pitch==-1)                        //error filtering
{v_pitch=0;}
v_roll=(GyY/131);
if(v_roll==1)                          //error filtering {v_roll=0;}
v_yaw=GyZ/131;
a_pitch=(v_pitch*0.046);
a_roll=(v_roll*0.046);
a_yaw=(v_yaw*0.045);
pitch= pitch + a_pitch;
roll= roll + a_roll;
yaw= yaw + a_yaw;
Serial.print(" | pitch = ");
Serial.print(pitch);
Serial.print(" | roll = ");
Serial.print(roll);
Serial.print(" | yaw = ");
```

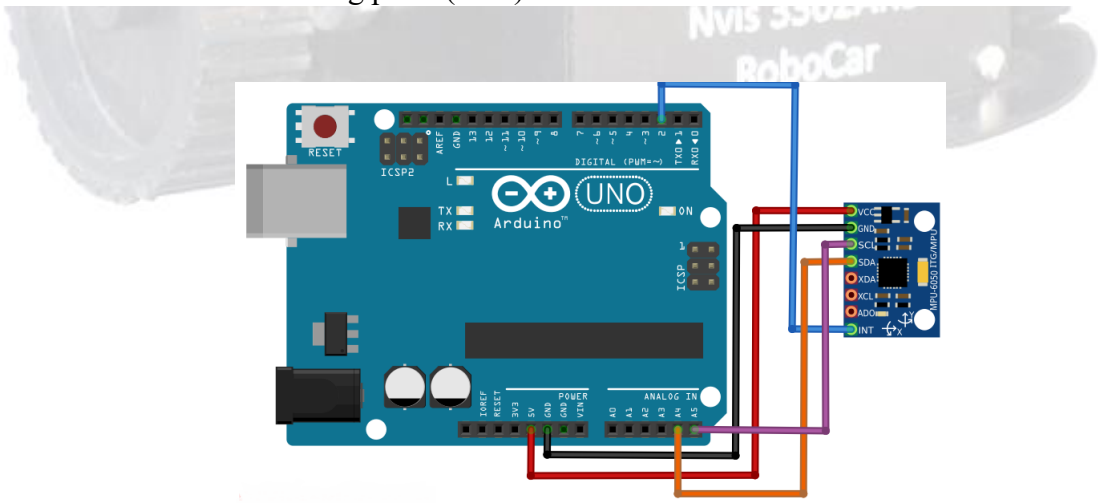
```
Serial.println(yaw);
}
```

PIN CONFIGURATION:

VDD : 5V on UNO
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 labelled as INT on the MPU 6050. Next, we need to set up the I2C lines. To do this, connect the pin labelled SDA on the MPU 6050 to the Arduino's analog pin 4 (SDA) and the pin labelled as SCL on the MPU 6050 to the Arduino's analog pin 5 (SCL).



RESULT:

We learnt about the various features that are associated with the sensor MPU6050 and can relate them with the various real life occasions where they can be used or is already being employed. We could see how the coordinates were changing on moving the sensor and 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:

- Arduino Uno
- DIGI XBee Module

SOFTWARE:

- Arduino IDE: Arduino

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 days for wireless communication functionality. It can be used as a transmitter and receiver both. It used serial communication to send and receive data. It have 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 ZigBee buddies.

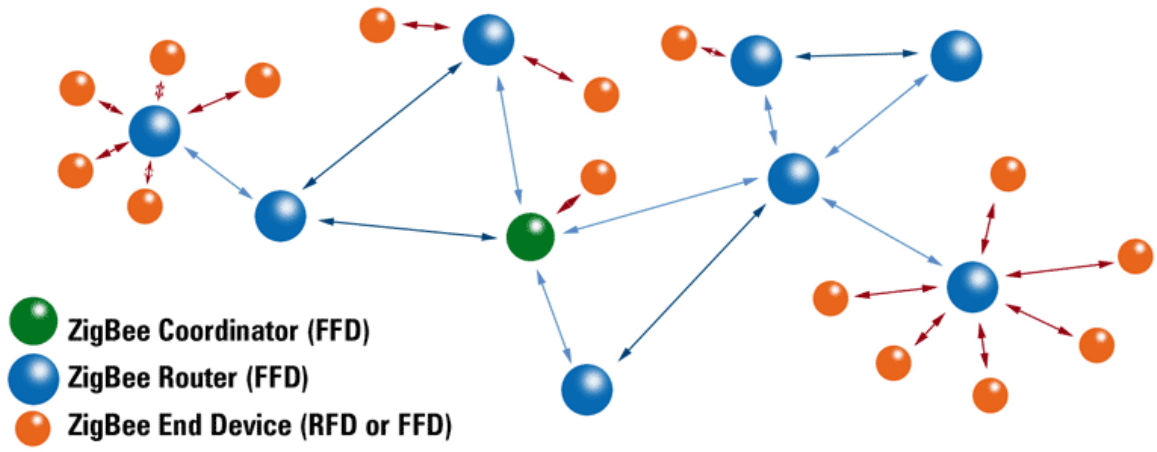
As already mentioned it use serial port to send and receive data. So its mean it can be easily interface with Arduino Uno R3, any type of microcontroller and computer. Because they all support serial communication and they all have serial port to send and receive data. It can also communicate with other ZigBee to form a mesh. ZigBee can also be used to make a local area network.

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.

APPLICATIONS:

- Wireless Communication
- Wireless Controlled Robot
- Wireless Monitoring System
- Wireless Temperature Sensor
- Wireless Home Automation System

ZIGBEE NETWORK FORMATION:



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).

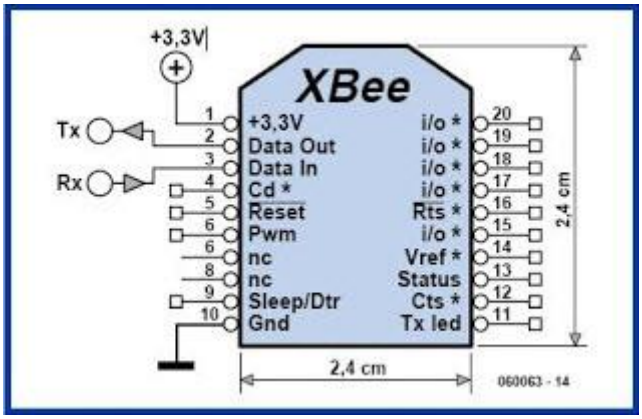
CODE:

```
void setup()
{
  Serial.begin(9600);   //serial initlization here we are using 9600 baud rate.
  pinMode(5, OUTPUT);   // Right +ve)
  pinMode(6, OUTPUT);   // Right -ve
  pinMode(7, OUTPUT);   // Left -ve
  pinMode(8, OUTPUT);   // Left +ve
}
```

```
//functions for forward,backward,Anti_Rotation,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 stops()
{
  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 than read it.
  {
    s=Serial.read();
    if(s=='F')
    forward();
  else if(s=='B')
  {    Backward();  }
  else if(s=='R')
  {    Clock_Rotation();  }
  else if(s=='L')
  {    Anti_Rotation();  }
  s=' ';
  }
}
```

PIN DIAGRAM:



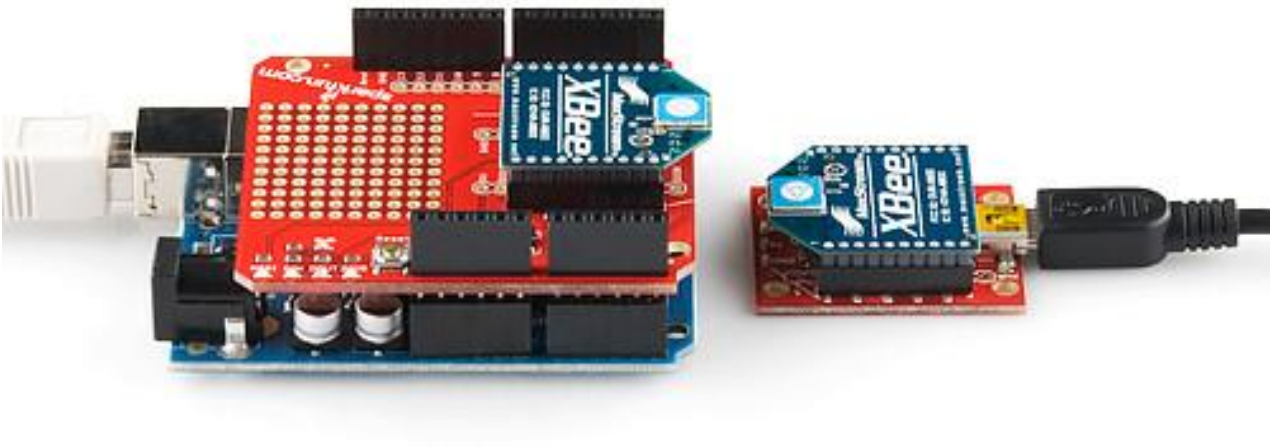
INTERFACING WITH ARDUINO:

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 many XBees out in the world. Additionally, API mode will expose a variety of additional information encoded in each packet.

XBEE MODULE:



RESULT:

We learnt about the various features that are associated with XBee module. We understood the reason of calling coordinator as master and router as slave. We could control the movements in buggy for e.g. on pressing F, B, R, L from the keyboard our buggy moved forward, backward, rotated clockwise and anticlockwise respectively. We could relate it to various real life applications where it is being used or can be employed in future.