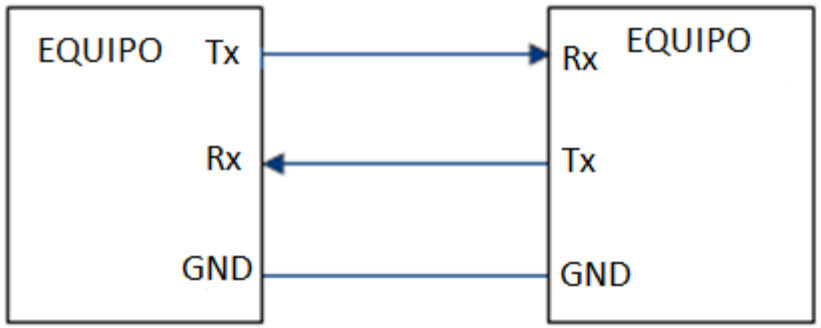
COMUNICACIÓN SERIAL ALAMBRICA E INALAMBRICA USANDO EL XBEE

Lenguajes Técnicos de Programación

UNIVERSIDAD EAFIT®

COMUNICACIÓN SERIAL

La comunicación serial sólo utiliza tres líneas, una para recibir los datos Rx, otra para trasmitir los datos Tx y la línea común GND.
Cada dato se transmite bit a bit, un bit a la vez, por lo tanto se hace mucho más lenta, pero tiene la ventaja de necesitar menos líneas y las distancias a las cuales se puede transferir la información son mayores, además con el uso de los módem se puede trasmitir a cualquier parte del mundo



Existen dos formas de comunicación serial:

- Sincrónica
- Asincrónica

UNIVERSIDAD EAFIT®

COMUNICACIÓN SINCRÓNICA Y ASINCRÓNICA

- COMUNICACIÓN SINCRÓNICA

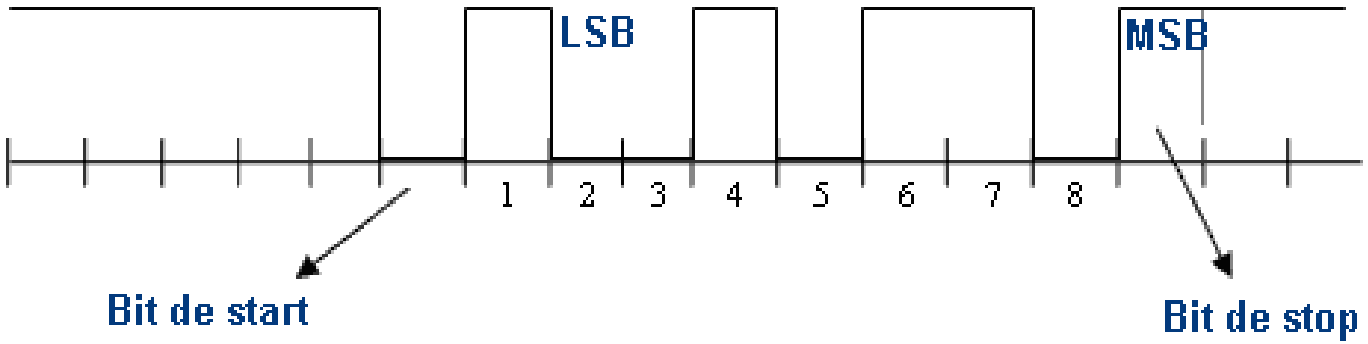
En esta comunicación además de una línea sobre la que se transfieren los datos, se necesita otra que contenga pulsos de reloj que indiquen que el dato es válido; la duración del bit está determinada por la duración del pulso de sincronismo.

- COMUNICACIÓN ASINCRÓNICA

En esta comunicación los pulsos de reloj no son necesarios y se utilizan otros mecanismos para realizar la transferencia de datos. La duración de cada bit está determinada por la velocidad con la cual se realiza la trasferencia de datos, por ejemplo si se transmite a 1200 bits por segundo (baudios), la duración de cada bit es de 833 microsegundos.
Las velocidades de transmisión más comunes son 300, 600,1200, 2400, 9600, 14400 y 28800 baudios

UNIVERSIDAD EAFIT®

COMUNICACIÓN SINCRÓNICA Y ASINCRÓNICA



En la figura se muestra la forma como se trasmite un dato cuando se realiza alguna transferencia, la línea del transistor permanece en alto. Para empezar a transmitir datos esta línea se pone en bajo durante un bit, lo cual se conoce como bit de Start, y luego comienza a transmitir los bits correspondientes al dato, empezando por el bit menos significativo (LSB) y terminando con el más significativo (MSB). Al finalizar se agrega el bit de paridad, si está activada esta opción, y por último los bits de stop, que pueden ser 1 o 2, en los cuales la línea regresa a nivel alto.

UNIVERSIDAD EAFIT®

COMUNICACIÓN SERIAL

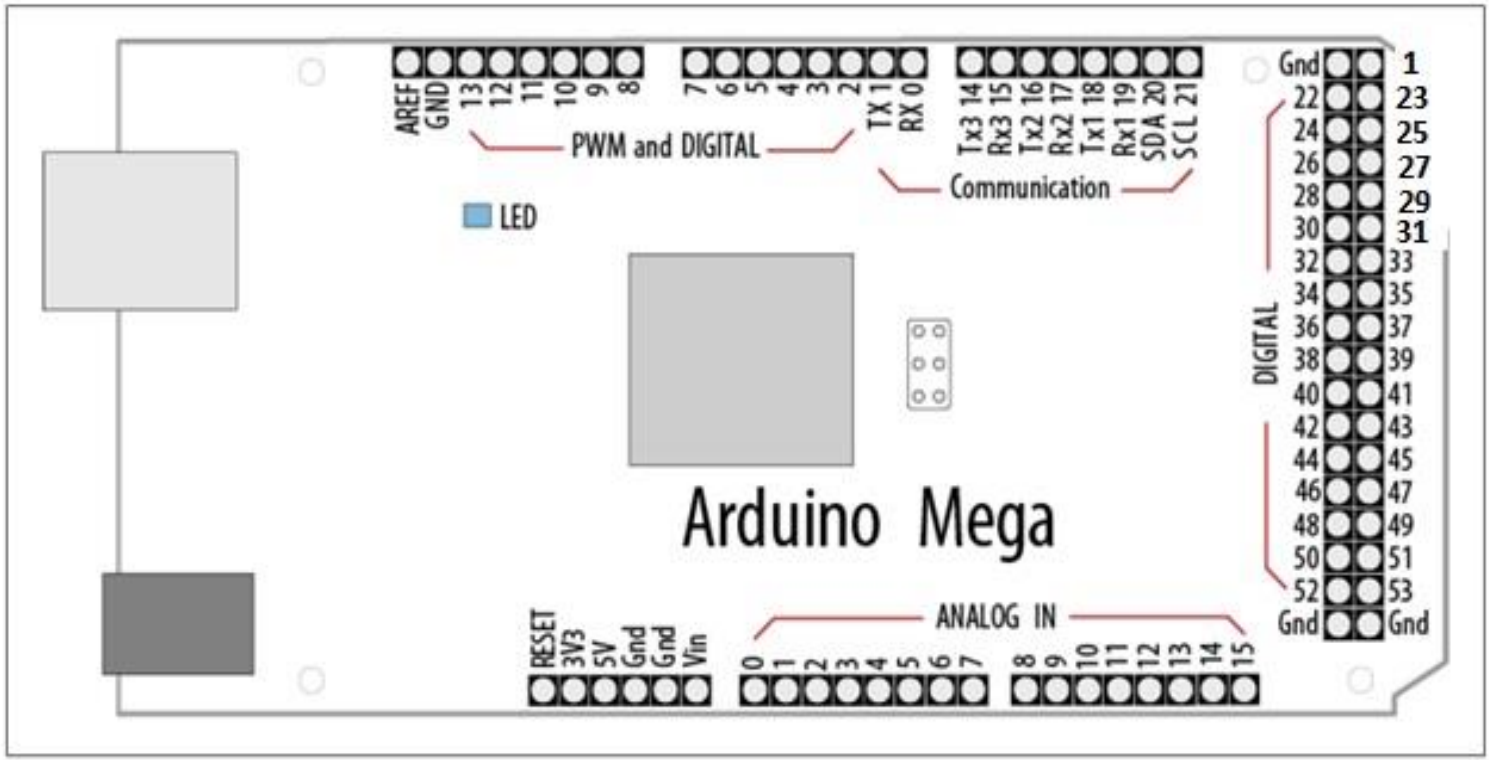
La comunicación serial se utiliza para la comunicación entre la placa Arduino y un ordenador u otros dispositivos. Todas las placas Arduino tienen al menos un puerto serie (también conocido como un UART o USART): Serial. Se comunica a los pines digitales 0 (RX) y 1 (TX), así como con el ordenador a través de USB. Por lo tanto, si utiliza estas funciones, no se pueden usar los pines 0 y 1 para la entrada o salida digital.

La placa Arduino Mega tiene tres puertos adicionales de serie: Serial1 en los pines 19 (RX) y 18 (TX), Serial2 en los pines 17 (RX) y 16 (TX), Serial3 en los pines 15 (RX) y 14 (TX).

Se puede utilizar el entorno de Arduino monitor incorporado de serie para comunicarse con la placa Arduino. Haga clic en el botón de serie del monitor en la barra de herramientas y seleccione la misma velocidad de transmisión utilizada en la llamada a comenzar begin().

NOTA: Las velocidades de transmisión serial del Arduino mega que se pueden utilizar son los siguientes 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 o 115200 **baudios**

Pines de comunicación del Arduino Mega



UNIVERSIDAD EAFIT®

FUNCIONES COMUNES EN LA COMUNICACIÓN SERIAL DEL ARDUINO

Serial.begin(): Establece la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie. Para comunicarse con el computador, utilice una de estas velocidades: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 o 115200.

Serial.available(): Devuelve el número de bytes (caracteres) disponibles para ser leidos por el puerto serie. Se refiere a datos ya recibidos y disponibles en el buffer de recepción del puerto (que tiene una capacidad de 128 bytes).

Serial.read(): Lee los datos entrantes del puerto serie. En el Arduino Mega: Serial1.read() , Serial2.read() , Serial3.read()

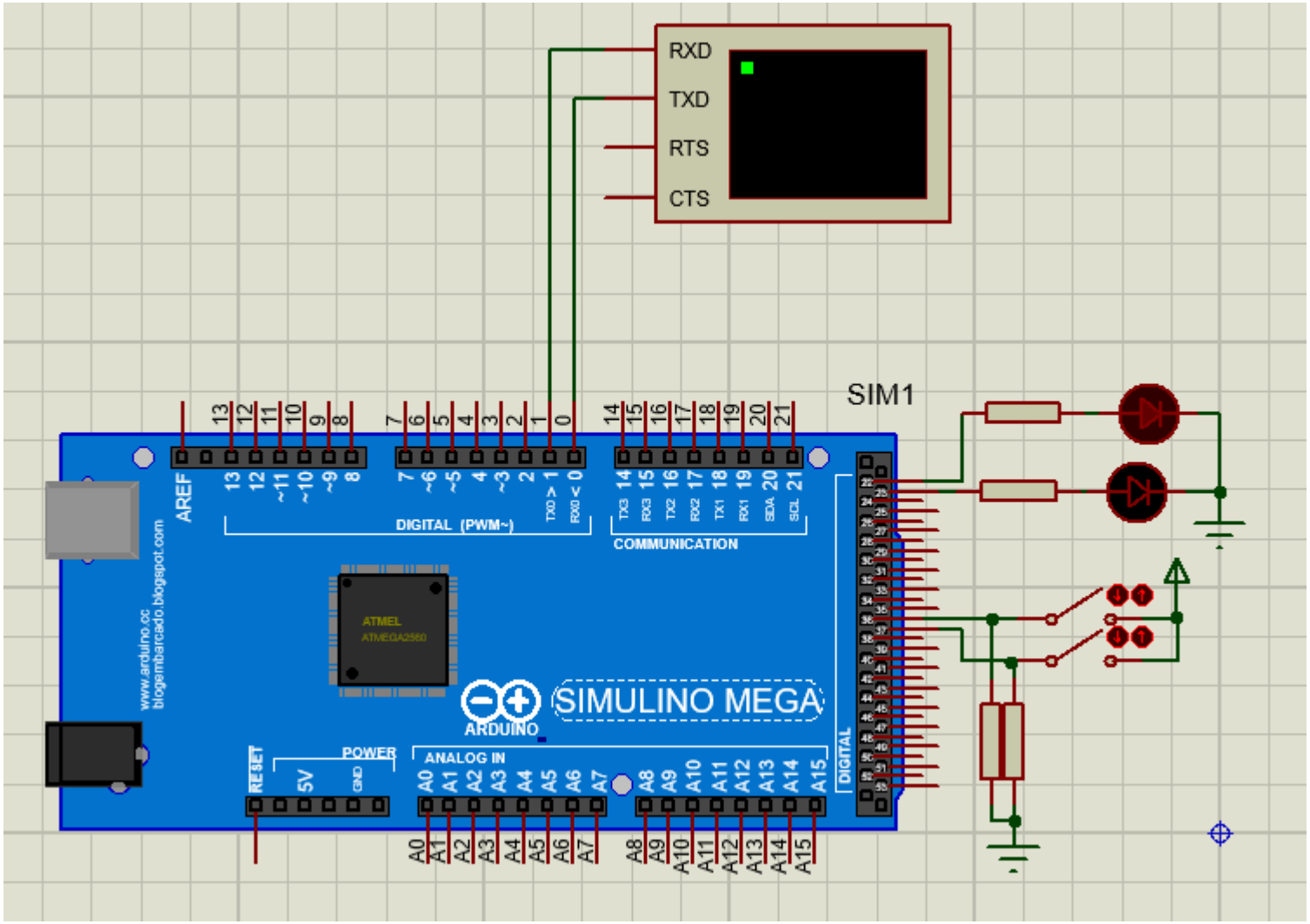
Serial.print(): Imprime los datos al puerto serie como texto ASCII. Este comando puede tomar muchas formas. Los números son impresos mediante un juego de caracteres ASCII para cada dígito.

Serial.println(): Imprime los datos al puerto serie como texto ASCII seguido de un retorno de carro (ASCII 13, o '\r') y un carácter de avance de línea (ASCII 10, o '\n'). Este comando tiene la misma forma que serial.print()

Serial.flush(): Limpia el buffer una vez que todos los caracteres de salida han sido enviados

UNIVERSIDAD EAFIT®

CIRCUITO EN PROTEUS COMUNICACIÓN ALÁMBRICA



UNIVERSIDAD EAFIT

Ejemplo 1

Enciende un led conectado a A0 si el numero recibido del puerto serial es 1 y apaga el led si el numero recibido del puerto serial es 2.

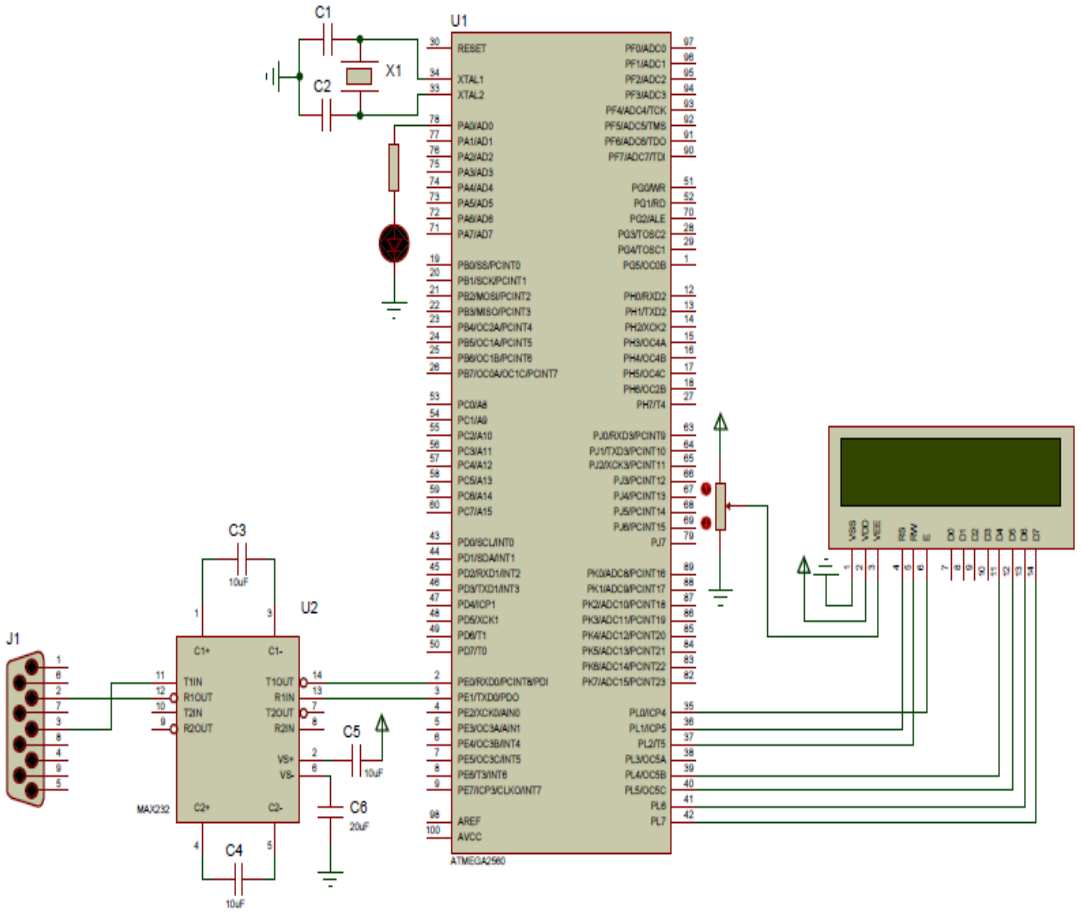
```
int led = 22;                          // led en el pin 22
int dato;


void setup()
 {
  Serial.begin(9600);
  pinMode(led, OUTPUT);                // configura el led como salida
 }
void loop()
 {
  if (Serial.available()>0)
  {
   dato=Serial.read();
   if(dato=='1')
     {
      digitalWrite(led, HIGH);         // activa el led
     }
   if(dato=='2')
     {
      digitalWrite(led, LOW);          // desactiva el led
     }
  }
 }
```

UNIVERSIDAD EAFIT®

EVENTO SERIAL

Para programar un evento por puerto serial lo único que se debe hacer es agregar al programa la rutina void serialEvent() antes del void **setup** o después del void **loop**, esta rutina es la que se encarga de sensar o preguntar si llego algún dato en el puerto serial, en esta rutina se agrega el programa que se quiere que ejecute el Arduino cuando se genere la interrupción.



UNIVERSIDAD EAFIT

EVENTO SERIAL

Para atención de la rutina de evento en el puerto serial conectado a TX0 Y RX0 usar:

```
Void serialEvent()
{
Sentencia;
}
```

Para atención de la rutina de interrupción en el puerto serial conectado a TX1 Y RX1 usar:

```
void serialEvent1()
{
Sentencia;
}
```

Para atención de la rutina de interrupción en el puerto serial conectado a TX2 Y RX2 usar:

```
void serialEvent2()
{
Sentencia;
}
```

Para atención de la rutina de interrupción en el puerto serial conectado a TX3 Y RX3 usar:

```
void serialEvent3()
{
Sentencia;
}
```

Ejemplo 2

Hacer un programa en encienda y apague un led cada segundo y cuando se envié el dato 1 o 0 encienda o apague un led diferente

```
int led1 = 22;                                  // led1 en el pin 22
int led2 = 23;                                  // led2 en el pin 23
char valor;


 void setup()
  {
    pinMode(led1, OUTPUT);                      // configura el led1 como salida
    pinMode(led2, OUTPUT);                      // configura el led2 como salida
    Serial.begin(9600);                         //Define la velocidad de la comunicación a 9600 baudios
    Serial.println(" 1 para encender el led2"); //Indica mensaje en el monitor serial
    Serial.println(" 0 para apagar el led2");
  }

void loop()
   {
     digitalWrite(led1, HIGH);                  // activa el led1
     delay(1000);
     digitalWrite(led1, LOW);                   // desactiva el led1
     delay(1000);
   }
```

UNIVERSIDAD EAFIT®

Ejemplo 2

```
void serialEvent()                      //Revisa que al menos se haya recibido un carácter en el puerto serie
   {
       valor=Serial.read();
       if(valor=='1')                   //si la tecla pulsada es el carácter 1
        {
            digitalWrite(led2, HIGH);   // activa el led2
        }
      if(valor=='0')                    //si la tecla pulsada es el carácter 0
        {
           digitalWrite(led2, LOW);     // desactiva el led2
        }
    }
```

Ejemplo 3

Hacer un programa donde el usuario pueda enviar el tiempo en milisegundos del retardo para encender y apagar el led

```
int led = 22;                                   // led en el pin 22
 int valor,cen,dec, uni;


 void setup()
 {
   pinMode(led, OUTPUT);                        // configura el led1 como salida
   Serial.begin(9600);                          //Define la velocidad de la comunicación a 9600 baudios
   Serial.println("Digite el tiempo de tres dígitos");
   valor=100;                                   // Define valor 100 por defecto
 }

void loop()
    {
       digitalWrite(led, HIGH);                 // activa el led
       delay(valor);
       digitalWrite(led, LOW);                  // desactiva el led
       delay(valor);
    }
```

UNIVERSIDAD EAFIT®

Ejemplo 3

```
void serialEvent()                          //Revisa que al menos se haya recibido un carácter en el puerto serie
    {
            cen=Serial.read();                          //Lee centenas
            dec=Serial.read();                          //Lee Decenas
            uni=Serial.read();                          //Lee unidades
            cen=cen-48;                                 //Convierte el carácter centenas a numero
            dec=dec-48;                                 //Convierte el carácter decenas a numero
            uni=uni-48;                                 //Convierte el carácter unidades a numero
            valor=cen*100+dec*10+uni;                    //agrupa los caracteres recibidos en velocidad (dato)

    }
```

UNIVERSIDAD EAFIT®

Uso de la funcion Serial.parseInt()

Serial.parseInt()

Devuelve el primer número entero (de tipo long) válido a partir de la posición actual, en la entrada serie. Los caracteres iniciales que no son enteros (o el signo menos) son ignorados. La función parseInt() termina cuando se encuentra el primer carácter que no es un dígito.

En el ejemplo anterior los cambios el evento serial quedaría de la siguiente manera:

```
void serialEvent()
 {
    valor=Serial.parseInt();          //Lee el valor entero
 }
```

ZIGBEE

Zigbee es un protocolo de comunicaciones inalámbrico basado en el estándar de comunicaciones para redes inalámbricas IEEE_802.15.4. Creado por Zigbee Alliance, una organización, teóricamente sin ánimo de lucro, de más de 200 grandes empresas (destacan Mitsubishi, Honeywell, Philips, _ ODEM_ do, Invensys, entre otras), muchas de ellas fabricantes de semiconductores. Zigbee permite que dispositivos electrónicos de bajo consumo puedan realizar sus comunicaciones inalámbricas. Es especialmente útil para redes de sensores en entornos industriales, médicos y, sobre todo, domóticos.

LAS COMUNICACIONES ZIGBEE

Las comunicaciones Zigbee se realizan en la banda libre de 2.4GHz. A diferencia de bluetooth, este protocolo no utiliza FHSS (Frequency hooping), sino que realiza las comunicaciones a través de una única frecuencia, es decir, de un canal. Normalmente puede escogerse un canal de entre 16 posibles. El alcance depende de la potencia de transmisión del dispositivo así como también del tipo de antenas utilizadas (cerámicas, dipolos, etc) El alcance normal con antena dipolo en línea vista es de aproximadamente (tomando como ejemplo el caso de MaxStream, en la versión de 1mW de potencia) de 100m y en interiores de unos 30m. La velocidad de transmisión de datos de una red Zigbee es de hasta 256kbps. Una red Zigbee la pueden formar, teóricamente, hasta 65535 equipos, es decir, el protocolo está preparado para poder controlar en la misma red esta cantidad enorme de dispositivos.

XBEE

UNIVERSIDAD EAFIT

PROPIEDADES XBEE

Entre las necesidades que satisface el módulo se encuentran:

Bajo costo.
Ultra-bajo consumo de potencia.
Uso de bandas de radio libres y sin necesidad de licencias.
Instalación barata y simple.
Redes flexibles y extensibles.

El uso del protocolo Zigbee va desde reemplazar un cable por una comunicación serial inalámbrica, hasta el desarrollo de configuraciones punto a punto, multipunto, peer-to-peer (todos los nodos conectados entre sí) o redes complejas de sensores.

El coordinador

Es el nodo de la red que tiene la única función de formar una red. Es el responsable de establecer el canal de comunicaciones y del PAN ID (identificador de red) para toda la red. Una vez establecidos estos parámetros, el Coordinador puede formar una red, permitiendo unirse a él a dispositivos Routers y End Points. Una vez formada la red, el Coordinador hace las funciones de Router, esto es, participar en el enrutado de paquetes y ser origen y/o destinatario de información.

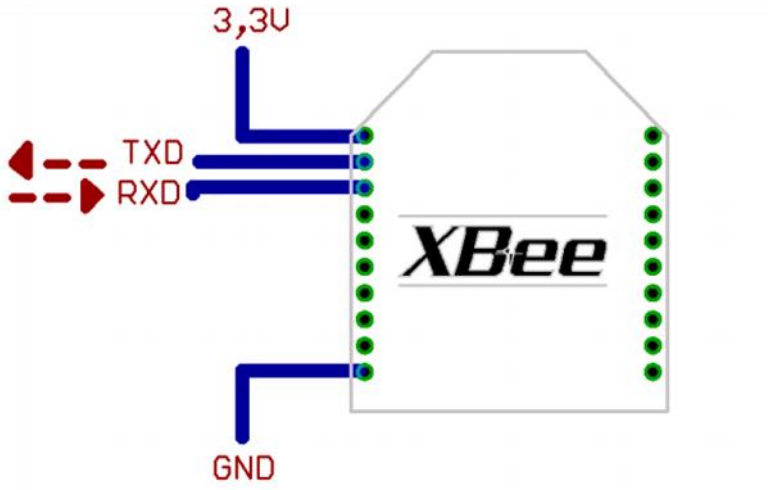
Los Routers

Es un nodo que crea y mantiene información sobre la red para determinar la mejor ruta para transmitir un paquete de información. Lógicamente un router debe unirse a una red Zigbee antes de poder actuar como Router retransmitiendo paquetes de otros routers o de End points.

UNIVERSIDAD EAFIT®

End Device

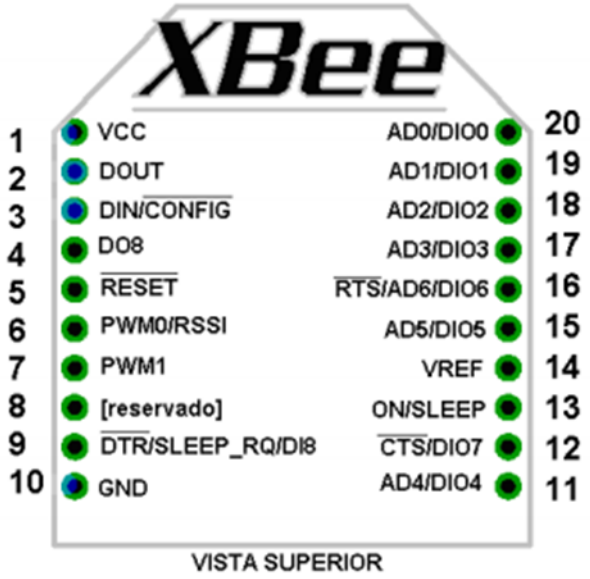
- Los dispositivos finales no tienen capacidad de enrutar paquetes. Deben interactuar siempre a través de su nodo padre, ya sea este un Coordinador o un Router, es decir, no puede enviar información directamente a otro end device. Normalmente estos equipos van alimentados a baterías.



UNIVERSIDAD EAFIT

Xbee

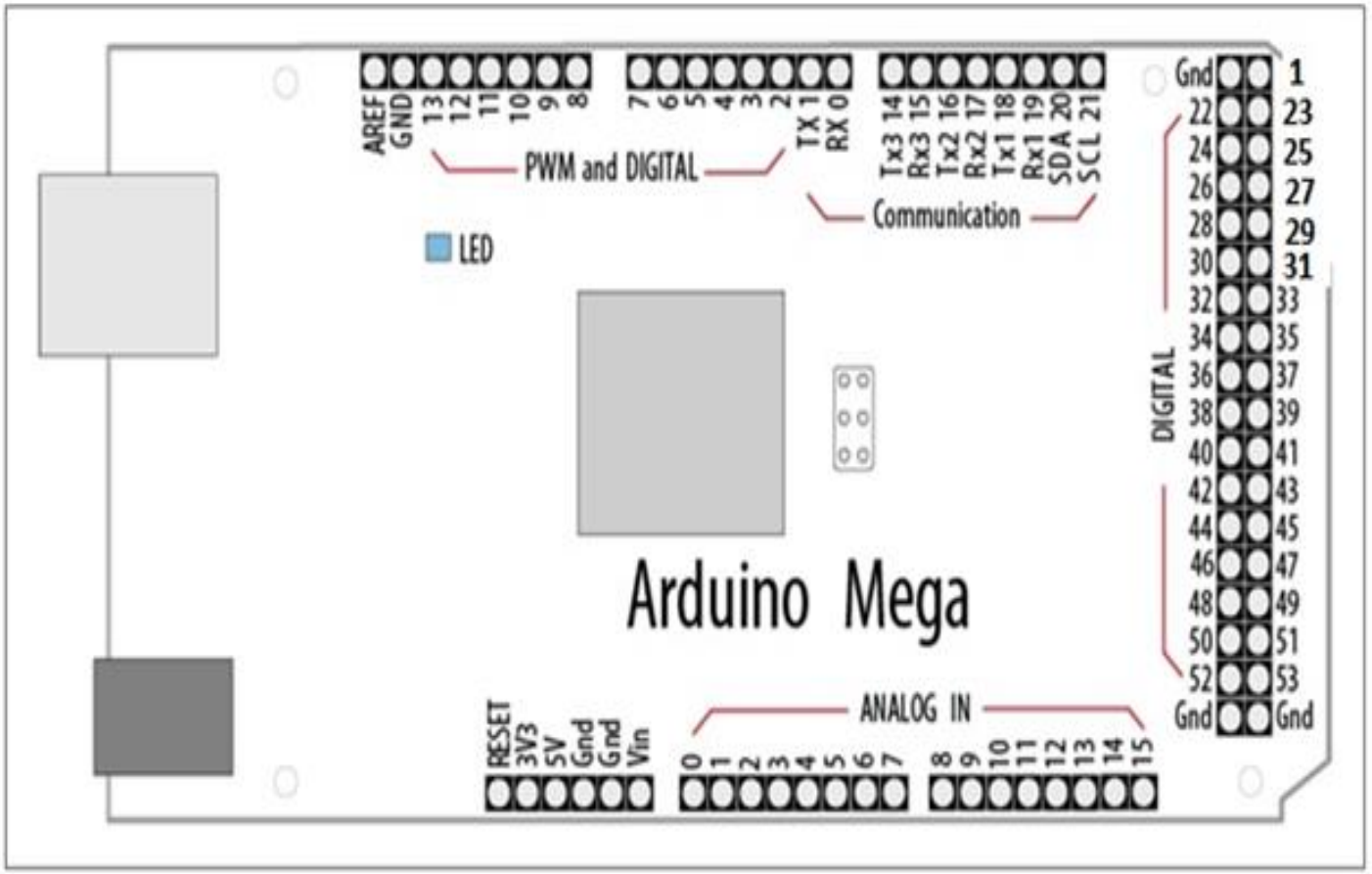
- Los módulos XBee se pueden emplear para recibir o transmitir datos digitales a través de sus pines sin la necesidad de un microcontrolador. Estos módulos pueden ser configurados para realizar estas operaciones, pero están limitados, ya que no pueden utilizar operaciones lógicas, y tampoco pueden transmitir señales análogas, pero si pueden recibirlas



VISTA SUPERIOR

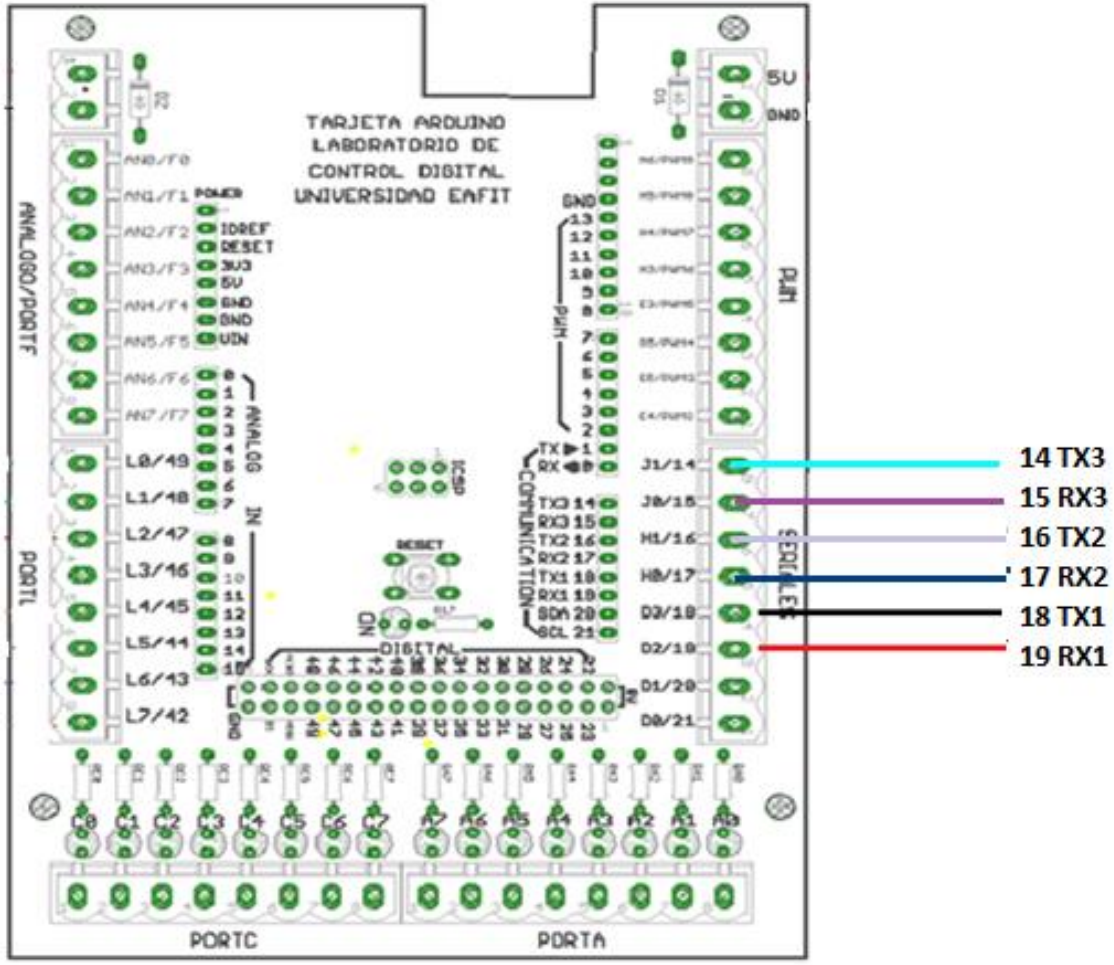
UNIVERSIDAD EAFIT

IDENTIFICACIÓN PINES DE COMUNICACIÓN SERIAL



UNIVERSIDAD EAFIT

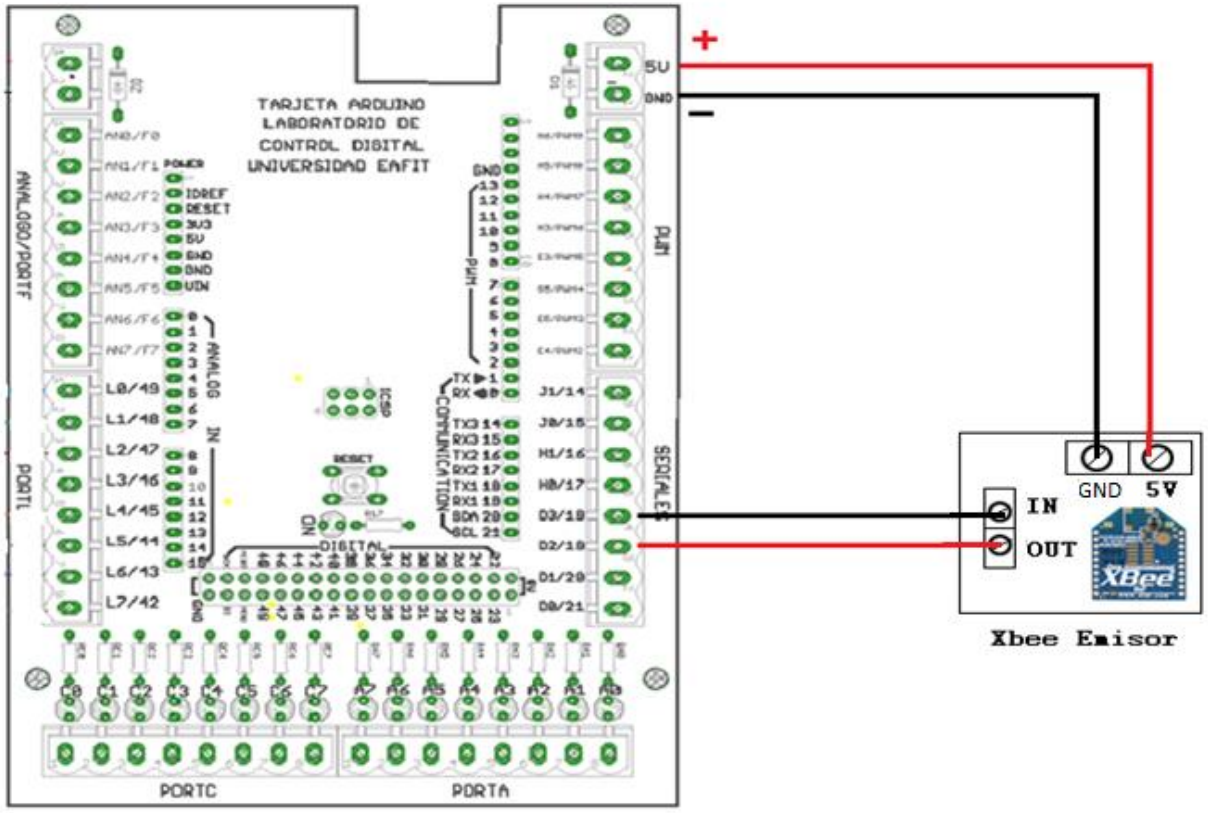
IDENTIFICACIÓN PINES DE COMUNICACIÓN SERIAL



UNIVERSIDAD EAFIT®

CONEXIONES

CONEXIÓN TRANSMISOR



UNIVERSIDAD EAFIT®

Aclaraciones

- La comunicación usando los Xbee se hace por transmisión serial. En el Laboratorio ya se tiene los Xbee configurados, la mayoría a una velocidad de 9600 baudios, por lo tanto los ejemplos que se muestra a continuación funcionan a esta velocidad.

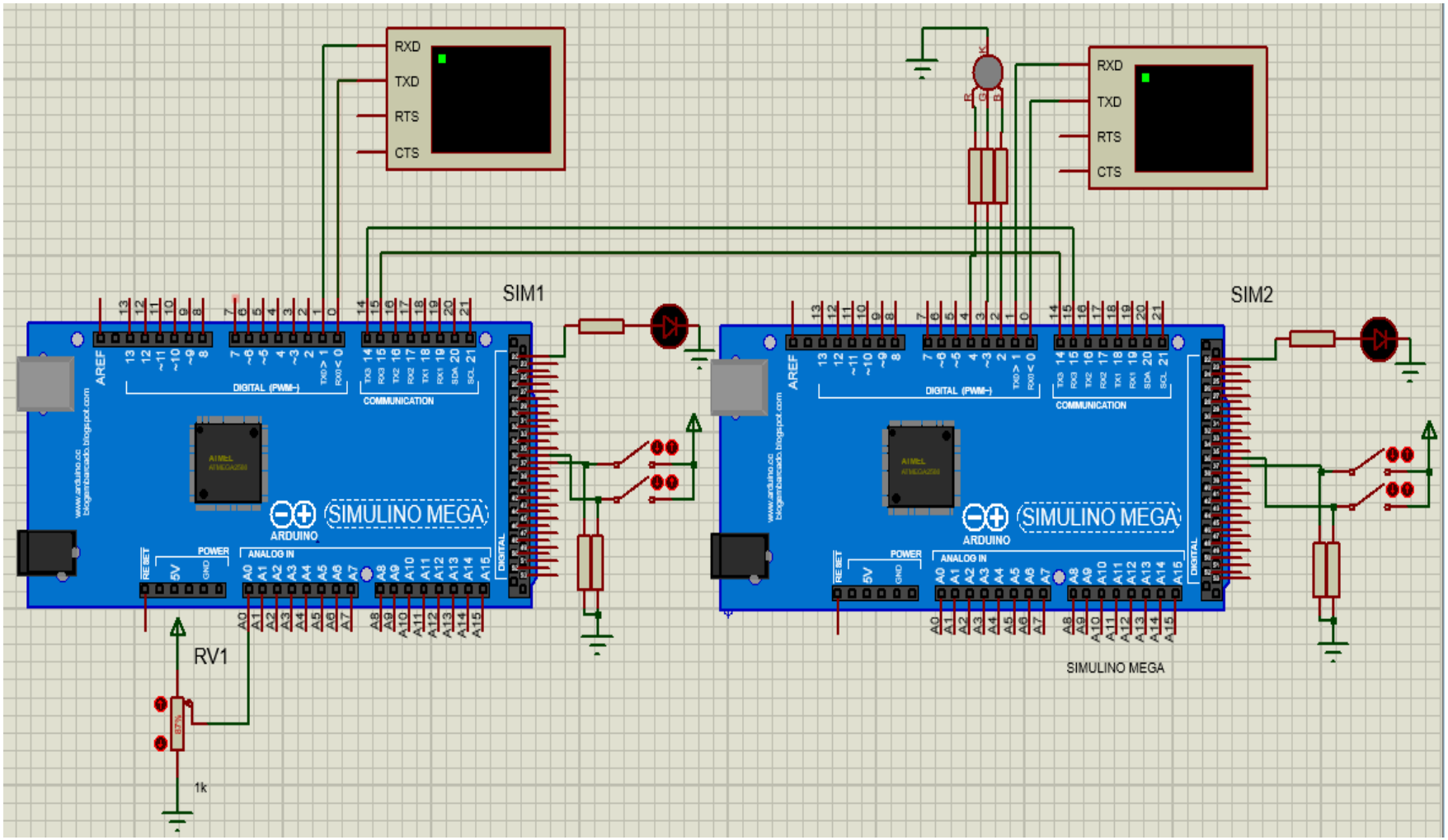
- Se puede enviar un valor máximo de 255, sin embargo al enviar el dato por comunicación serial, este protocolo envía carácter por carácter.

- Cuando se comunican dos elementos usando comunicación serial, siempre se debe establecer un protocolo de comunicación entre los dos dispositivos.

- En cada ejemplo se muestra el programa para el emisor y para el receptor.

UNIVERSIDAD EAFIT®

CIRCUITO EN PROTEUS COMUNICACIÓN INALÁMBRICA



UNIVERSIDAD EAFIT

Ejemplo 4

Comunicar dos Arduinos por Xbee de tal manera que los dos funcionen igual. Con un "1" enciende el led y con un "0" apaga el led. (Este programa es el mismo para el emisor y el receptor)

```
int sw1 = 37;                    // sw1 en el pin 37
int sw2 = 36;                    // sw2 en el pin 36
int led = 22;                    // led en el pin 22

int dato;
void setup()
 {
    Serial3.begin(9600);
    Serial.begin(9600);
    pinMode(sw1, INPUT);         // configura el sw1 como entrada
    pinMode(sw2, INPUT);         // configura el sw2 como entrada
    pinMode(led, OUTPUT);        // configura el led como salida
 }

void loop()
{
  if(Serial3.available()>0)
   {
    dato=Serial3.read();
    if(dato=='1')
     {
       digitalWrite(led, HIGH);  // activa el led
       Serial.println("Led encendido");
     }
```

UNIVERSIDAD EAFIT®

Ejemplo 4

```
if(dato=='0')
    {
      digitalWrite(led, LOW);              // desactiva el led
      Serial.println("Led apagado");
    }
  }

   if(digitalRead(sw1)==HIGH)
   {
     Serial3.print('1');
   }

   if(digitalRead(sw2)==HIGH)
   {
     Serial3.print('0');
   }
}
```

UNIVERSIDAD EAFIT®

Ejemplo 5 (Emisor)

Encender el led RGB de tres colores diferentes de acuerdo al carácter enviado..

```
int dato;
void setup()
 {
   Serial.begin(9600);
   Serial3.begin(9600);
   Serial.print("Comenzamos");
 }
void loop()
 {
   if (Serial.available()>0)
   {
     dato=Serial.read();
     if(dato=='r')
      {
```

Ejemplo 5 (Emisor)

```
Serial3.print('r');
    }
    if(dato=='v')
    {
      Serial3.print('v');
    }
    if(dato=='a')
    {
      Serial3.print('a');
    }
  }
}
```

UNIVERSIDAD EAFIT®

Ejemplo 5 (Receptor)

```
int dato;


void setup()
 {
   Serial3.begin(9600);
 }
void loop()
 {
   if (Serial3.available()>0)
   {
     dato=Serial3.read();
     if(dato=='v')
      {
        analogWrite(3,55);
        analogWrite(2,0);
        analogWrite(4,0);
      }
```

UNIVERSIDAD EAFIT®

Ejemplo 5 (Receptor)

```
if(dato=='r')
    {
      analogWrite(4,255);
      analogWrite(3,0);
      analogWrite(2,0);
    }
    if(dato=='a')
    {
      analogWrite(3,255);
      analogWrite(2,0);
      analogWrite(4,0);
    }
  }
}
```

Ejemplo 6 (Emisor)

Enviar el valor de la señal análoga que entra por A0

```
int valor;  // declaración de constantes


void setup()
 {
   Serial3.begin(9600);
   Serial.begin(9600);
}
void loop()
{
  valor = analogRead(0);    //lee datos sensor
   Serial.println(valor);
   Serial3.println(valor);    //los imprime en el serial 0
   delay(500);
}
```

UNIVERSIDAD EAFIT®

Ejemplo 6 (Receptor)

```
int valor;// declaración de constantes
void setup()
{ //iniciacion de puertos serial
  Serial3.begin(9600);
  Serial.begin(9600);
}
void loop()
{
  if (Serial3.available() > 0 )
  {
    valor = Serial3.parseInt();   //Lee el valor entero
     Serial.println(valor);
  }
}
```

UNIVERSIDAD EAFIT®