# Ampliación 01: Módulo de radio APC220

En esta ampliación se realiza una descripción del funcionamiento del módulo de radio APC220, configuración y programación para la transmisión de datos obtenidos desde el sensor de temperatura y presión.

Material necesario

- 1 Arduino UNO
- Dos módulos radio APC220
- Convertidor USB to TTL
- Cables macho-hembra, macho-macho
- Sensor BMP280
- Protoboard o placa de prototipado

En la transmisión se tienen siempre dos componentes:

- Transmisor: Es el elemento encargado de codificar y enviar la señal que deseamos transmitir, en nuestro caso el CanSat es el elemento transmisor
- Receptor: Es el elemento encargado de recibir y decodificar la señal transmitida, nuestra estación base.

El módulo de radio APC220 permite ajustar sus parámetros, y ambos dispositivos pueden funcionar tanto de transmisor como de receptor, permitiendo una comunicación bidireccional.

### 1.- Configuración de módulos de radio

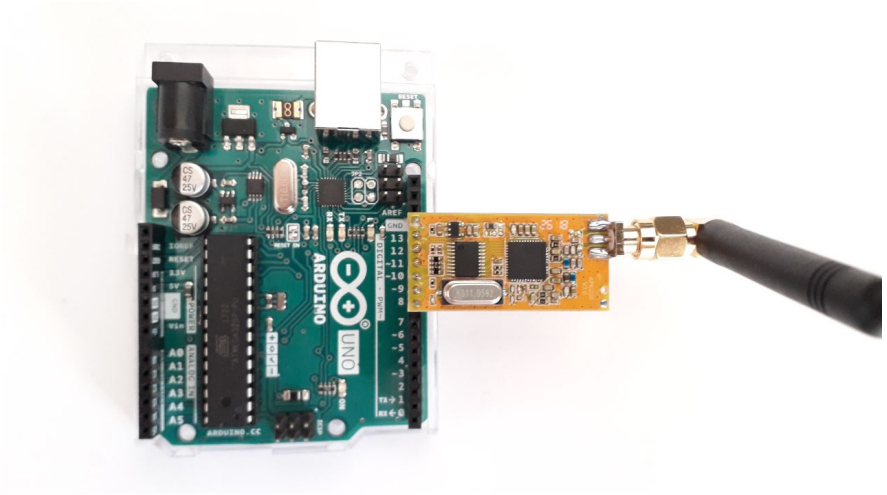
Para configurar la frecuencia de funcionamiento y otros parámetros necesarios para la comunicación, hay dos formas: a través de un programa ejecutable (RF-Magic) y a través de una programación en Arduino.

En este caso, vamos a utilizar la programación en Arduino, ya que se evitan problemas de compatibilidad y drivers. Vamos a configurar los dos módulos, primero se conecta uno, y se carga el programa; después se conecta el otro y se vuelve a cargar.

Conexión APC220 para configuración:

| Arduino UNO | APC220 |
|---|---|
| GND | GND |
| 13 | VCC |
| 12 | EN |
| 11 | RXD |

| | |
|---|---|
| 10 | TXD |
| 9 | AUX |
| 8 | SET |



Esquema de conexión para configuración APC220. También se puede realizar la conexión mediante cables

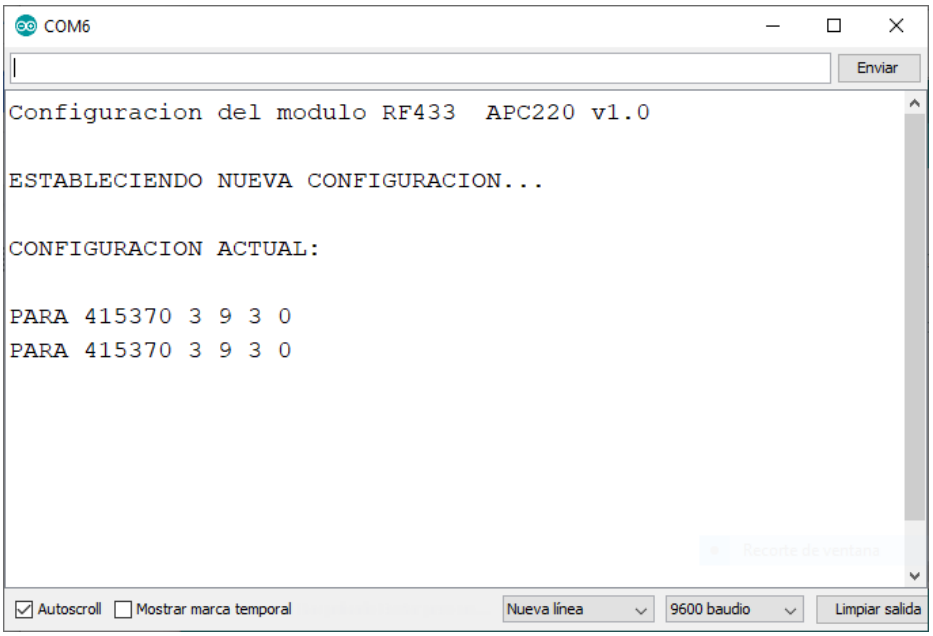
Conectar el Arduino al puerto USB de nuestro ordenador y cargar el programa APC220_config.ino. este código nos permite leer la configuración actual y cambiar los parámetros.

```
/*
 Programa configuración APC220
 Fuente: https://github.com/inopya/APC220_Transceiver

 Esquema de conexión módulo APC220 con Arduino

 ARDUINO    APC220
 GND   --->  GND
 13    --->  VCC
 12    --->  EN
 11    --->  RXD
 10    --->  TXD
  9    --->  AUX
  8    --->  SET

/*

 Cuando se lee la configuracion del modulo se obtiene un linea similar a esta:
PARAM 415370 2 9 3 0

  "PARAM AAAAAA B C D E"

  AAAAAA, es la frecuencia de trabajo del modulo expresada en KHz, Puede oscilar
entre 418MHz y 455MHz
  - en el ejemplo 415.370 MHz

  B, es la velocidad de transmision de radio frecuencia puede tomar los
```

```
siguientes valores
  1 (2400bps), 2 (4800bps), 3 (9600bps), 4 (19200bps)

  C, es la potencia de emision, puede tomar valores entre 0 y 9, siendo 9 la
mayor potencia

  D, velodidad de transferencia entre el modulo y arduino o PC  , toma valores
entre 0 y 6: 0 (1200bps), 1 (2400bps), 2 (4800bps),3 (9600bps), 4 (19200bps),  5
(38400bps), 6 (57600bps)
  - en el ejemplo 9600bps

  E, es el control de paridad de la informacion emitida por RF
  0 (sin control de paridad), 1 (paridad par), 2 (paridad impar)
  - sin control de paridad

  Para grabar informacion se ha de enviar una linea similar...
  WR 434000 3 9 3 0

  Esta configuracion seria: Frecuencia de emision 434MHz, velocidad RF 9600,
  maxima potencia, Puerto serie 9600 y sin control de paridad

*/


#define __VERSION__ "\Configuracion del modulo RF433  APC220 v1.0\n"

//Librerias
#include <SoftwareSerial.h>

//Declaracion de constantes
#define SET      8
#define AUX      9
#define TXD     11
#define RXD     10
#define EN      12
#define VCC     13
#define GND    GND

//definimos el puerto serie Software para comunicar con el modulo RF
SoftwareSerial APCport(RXD, TXD);

void setup()
{
  Serial.begin(9600);
  Serial.println(F(__VERSION__));

//iniciar el puerto serie software para comunicar con el APC220
  APCport.begin(9600);

  pinMode(SET,OUTPUT);
  pinMode(AUX,INPUT);
  pinMode(EN,OUTPUT);
  pinMode(VCC,OUTPUT);

  digitalWrite(SET,HIGH);
  digitalWrite(VCC,HIGH);
  digitalWrite(EN,HIGH);
```

```
  delay(1000);

  write_config();
  delay(1000);

  read_config();
  delay(5000);
}

void loop()
{
  //no hacemos nada en esta seccion
}

//  ESCRIBIR CONFIGURACION

void write_config()
{
  Serial.println(F("ESTABLECIENDO NUEVA CONFIGURACION...\n"));
  digitalWrite(SET, LOW);               // poner en modo configuracion
  delay(50);

//Parametros de configuración
  APCport.print("WR 415370 3 9 3 0");
  APCport.write(0x0D);
  APCport.write(0x0A);
  delay(100);
  digitalWrite(SET, HIGH);
}

//  LEER CONFIGURACION

void read_config()
{
  Serial.println(F("CONFIGURACION ACTUAL:\n"));
  digitalWrite(SET, LOW);          // poner en modo configuracion
  delay(50);                       // pausa para estabilizar
  APCport.print("RD");             // peticion de datos
  APCport.write(0x0D);             // fin de linea
  APCport.write(0x0A);
  delay(100);                      // pausa para estabilizar

  while (APCport.available()) {
    Serial.write(APCport.read());
  }
  digitalWrite(SET, HIGH);         // volver al modo normal
}

//Fin del programa
```

La salida en este caso es:




Esta respuesta significa que:

- **415370** es la frecuencia de trabajo del módulo expresada en KHz. Puede oscilar entre 418MHz y 455MHz
- El siguiente número **3**, es la velocidad de transmisión de radio frecuencia, puede tomar los siguientes valores: 1 (2400bps), 2 (4800bps), 3 (9600bps), 4 (19200bps)
- **9**, es la potencia de emisión, puede tomar valores entre 0 y 9, siendo 9 la mayor potencia
- **3**, velocidad de transferencia entre el módulo y arduino o PC, toma valores entre 0 y 6: 0 (1200bps), 1 (2400bps), 2 (4800bps),3 (9600bps), 4 (19200bps), 5 (38400bps), 6 (57600bps)
- **0**, es el control de paridad de la información emitida por RF: 0 (sin control de paridad), 1 (paridad par), 2 (paridad impar).

Cambiando estos parámetros en la línea `APCport.print("WR 415370 3 9 3 0")`, podemos cambiar la configuración de nuestro módulo de radio.

Los dos módulos, transmisor y receptor deben tener la misma configuración para funcionar adecuadamente.

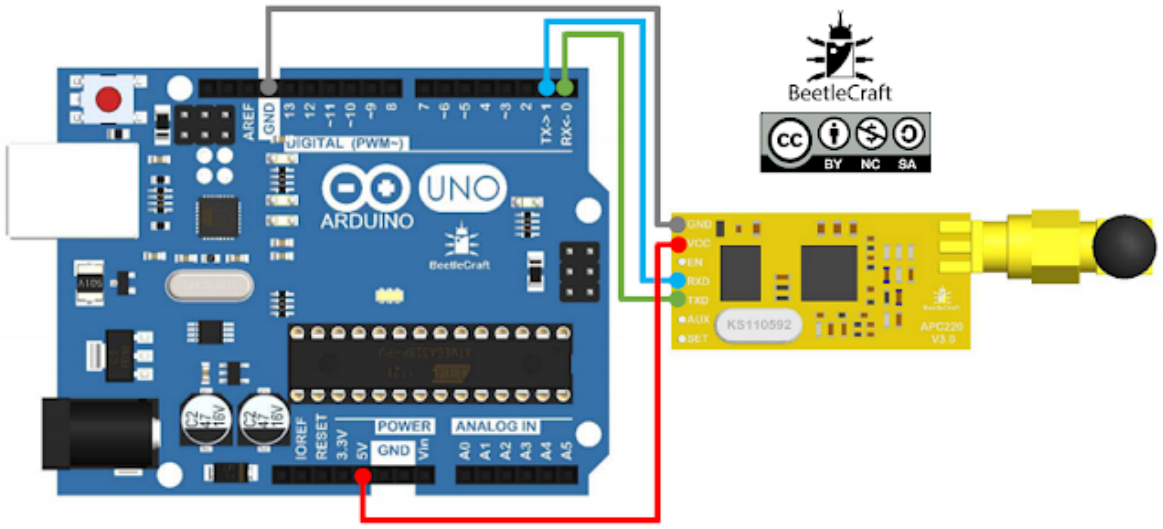
Para consultar cómo se realiza la configuración a través del programa RF-Magic, puedes visitar la wiki oficial del fabricante:
https://wiki.dfrobot.com/APC220_Radio_Data_Module_SKU_TEL0005_

## 2.- Test de funcionamiento

Una vez los dos módulos están configurados a la misma frecuencia, uno se conecta al Arduino y el otro a nuestro ordenador a través del conversor USB.

Esquema de conexión APC220 en Arduino para transmitir datos:

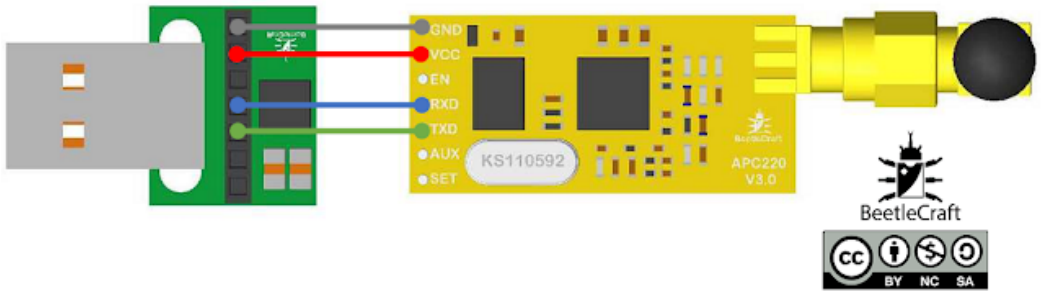
| Arduino UNO | APC220 Transmisor |
|---|---|
| GND | GND |
| 5V | VCC |
| TX | RXD |
| RX | TXD |




Esquema de conexión módulo transmisor en Arduino.
fuente:https://beetlecraft.blogspot.com/2015/10/tutorial-apc220.html

Esquema de conexión módulo receptor con conversor USB

| Conversor USB | APC220 Emisor |
|---|---|
| GND | GND |
| VCC | VCC |
| TX | TXD |
| RX | RXD |




Esquema de conexión módulo receptor con conversor USB.
Fuente:https://beetlecraft.blogspot.com/2015/10/tutorial-apc220.html

El programa de test sólo consiste en imprimir un *Hola Mundo* por el puerto serie:

```
/*
   PROGRAMA DE PRUEBA: APC220
   CONEXION:
             RXD: Arduino Pin 1
             TXD: Arduino Pin 0
             GND: Arduino GND
             VCC: Arduino 5V

   Autor: Renato H.
   http://beetlecraft.blogspot.pe/

   El siguiente programa es de uso publico, cualquier modificacion o mal uso del
mismo que pudiera ocasionar el mal funcionamiento de la plataforma de uso de la
misma no es responsabilidad del autor
*/

void setup(){
  Serial.begin(9600); // Velocidad de comunicacion
                      // La velocidad del puerto serial debe ser
                      // la misma que la de configuracion del modulo
}

void loop(){
 Serial.println("Hola mundo"); // Mensaje "Hola mundo"
 delay(1000);                 // Retraso de envio cada 1 segundo
}
```

Cargamos este programa en el sistema transmisor (Arduino + APC220 transmisor). Después conectamos el sistema receptor (Conversor USB + APC220 receptor) en uno de los puertos USB de nuestro ordenador. Al abrir el puerto serie de nuestro transmisor y receptor, podemos ver que imprime lo mismo, es decir, nuestro receptor nos muestra el puerto serie de nuestro transmisor.

Si alimentamos el sistema transmisor desde una batería externa o una pila, podremos ver que nuestro sistema receptor sigue recibiendo la información del puerto serie.

Nota: Al cargar el programa en Arduino, es necesario desconectar los pines TX y RX del APC220, ya que son los mismos que se utilizan para cargar los programas. Una vez cargado el programa, se vuelve a conectar.

**3.- Recibir los datos del sensor de presión y temperatura a través del módulo de radio.**

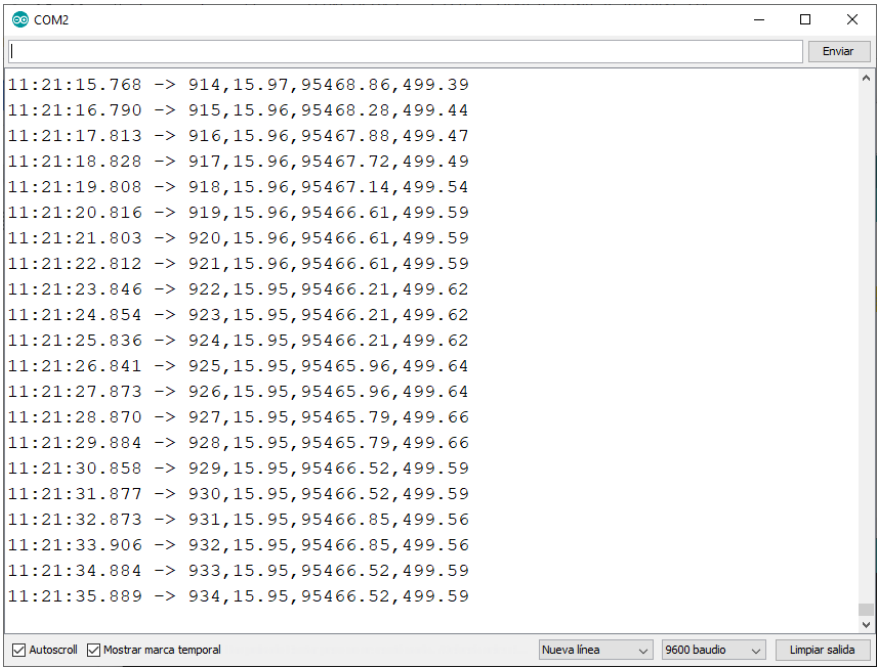
Sólo es necesario, conectar el sensor BMP280 y cargar la programación para obtener los datos. En esta programación de ejemplo, se ha añadido la variable paquete, un contador que se incrementa cada vez que se envían los datos. De esta forma, cada paquete de datos enviado tiene un número y podemos ver si hemos perdido alguno.

Ejemplo BMP280 con variable paquete:

```
//BMP280 con transmisor APC220

#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BMP280.h>

#define BMP_SCK  (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS   (10)

Adafruit_BMP280 bmp; // I2C
//Adafruit_BMP280 bmp(BMP_CS); // hardware SPI
//Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO,  BMP_SCK);

int paquete=0; //contador de paquete de datos enviado

void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
  while ( !Serial ) delay(100);   // wait for native usb
  Serial.println(F("BMP280 test"));
  unsigned status;

  //status = bmp.begin(BMP280_ADDRESS_ALT, BMP280_CHIPID);
  status = bmp.begin(0x76);
  if (!status) {
    Serial.println(F("Could not find a valid BMP280 sensor, check wiring or "
                     "try a different address!"));
    Serial.print("SensorID was: 0x"); Serial.println(bmp.sensorID(),16);
    Serial.print("        ID of 0xFF probably means a bad address, a BMP 180 or
BMP 085\n");
    Serial.print("   ID of 0x56-0x58 represents a BMP 280,\n");
    Serial.print("        ID of 0x60 represents a BME 280.\n");
    Serial.print("        ID of 0x61 represents a BME 680.\n");
    while (1) delay(10);
  }

  /* Default settings from datasheet. */
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode. */
```

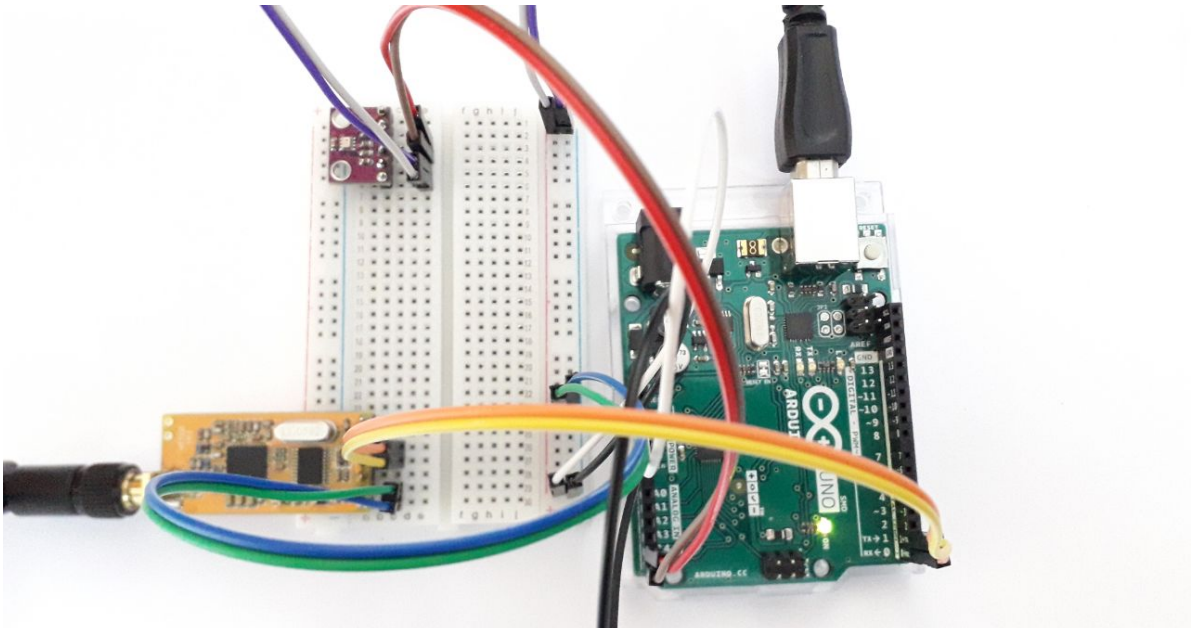
```
                    Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
                    Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
                    Adafruit_BMP280::FILTER_X16,      /* Filtering. */
                    Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */
}

void loop() {

    digitalWrite(13, HIGH);
    Serial.print(paquete);
    Serial.print(",");

    //Serial.print(F("Temperature = "));
    Serial.print(bmp.readTemperature());
    Serial.print(",");

    //Serial.print(F("Pressure = "));
    Serial.print(bmp.readPressure());
    Serial.print(",");

    //Serial.print(F("Approx altitude = "));
    Serial.print(bmp.readAltitude(1013.25)); /* Adjusted to local forecast! */
    Serial.println();

    digitalWrite(13, LOW);
    delay(1000);
    paquete++;
}
```

Resultado del puerto serie receptor con marca de tiempo activada.

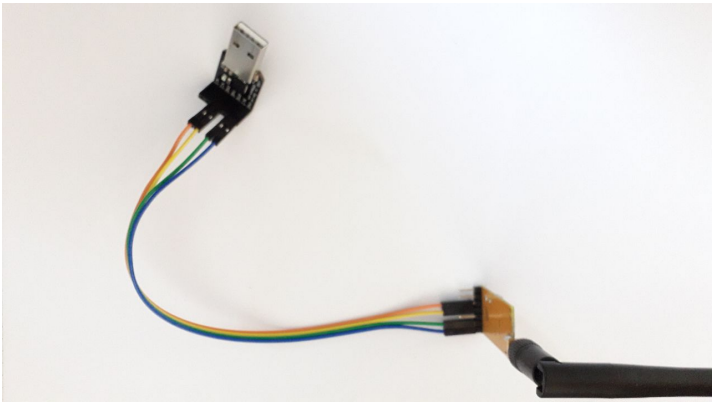



Resultado del puerto serie del sistema receptor

Circuito completo en placa de prototipado



Circuito transmisor: BMP280 y módulo transmisor de datos



Sistema receptor

Podemos capturar todos estos datos utilizando un programa como *Coolterm*
https://parzibyte.me/blog/2020/12/11/monitor-serial-arduino-coolterm/#Descargando_CoolTerm

Enlaces de interés:
Datasheet APC220: https://image.dfrobot.com/image/data/TEL0005/APC220_Datasheet.pdf
Configuración a través del software APC220 RF-Magic:
https://beetlecraft.blogspot.com/2015/10/tutorial-apc220.html
Software específico APC220 RF-Magic, para realizar configuración a través de puerto:
http://www.appcon.com.cn/en/soft.php?cid=34
Drivers
https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers