SENSORES Y ACTUADORES – TST 2024

TP9-e2- Implementar la comunicación RF entre y RX #76

TAREA:

Implementar la comunicación RF entre TX y RX

- Descripción: Establecer la conexión de radiofrecuencia (RF) entre el transmisor y el receptor.
- Tareas:
  - Configurar el código en el ESP32/Arduino para enviar datos desde TX.
  - Configure el código en el ESP32/Arduino para recibir datos en RX.
  - Validar la transmisión de datos entre ambos.

Proyecto: Comunicación RF entre TX y RX

Objetivo

Establecer una conexión de radiofrecuencia (RF) entre un transmisor y un receptor para enviar y recibir datos de manera inalámbrica.

Materiales Necesarios

1. **Dos módulos nRF24L01**
2. **Dos placas ESP32 o Arduino**
3. **Cables de conexión**
4. **Fuente de alimentación (opcional para ESP32)**

Diagrama de Conexiones

Las conexiones para el módulo nRF24L01 son las siguientes:

Módulo nRF24L01	**ESP32/Arduino**
VCC	3.3V
GND	GND
CE	GPIO 9
CSN	GPIO 10
SCK	GPIO 13
MOSI	GPIO 11
MISO	GPIO 12

Nota: Puedes cambiar los pines GPIO (CE y CSN) según tus necesidades, pero asegúrate de actualizar el código en consecuencia.

Plan del Proyecto

1. **Configuración del Hardware**: Conectar los módulos nRF24L01 a las placas ESP32/Arduino.
2. **Codificación del Transmisor (TX)**: Escribir y cargar el código en el dispositivo que enviará datos.
3. **Codificación del Receptor (RX)**: Escribir y cargar el código en el dispositivo que recibirá datos.
4. **Validación de la Comunicación**: Probar el sistema para asegurar que la transmisión y recepción de datos funciona correctamente.

1. Configuración del Hardware

Conectar los módulos nRF24L01 a los ESP32 o Arduino como se detalla en la tabla anterior. Asegúrar de realizar las conexiones correctamente y de que los módulos estén alimentados con 3.3V.

2. Código para el Transmisor (TX)

Cargar el siguiente código en el dispositivo que actuará como transmisor:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9, 10); // CE, CSN

void setup() {
    Serial.begin(9600);
    radio.begin();
    radio.openWritingPipe(0xF0F0F0F0E1LL); // Dirección del receptor
    radio.setPALevel(RF24_PA_HIGH); // Potencia alta
    radio.setChannel(76); // Canal de comunicación
    radio.stopListening(); // Dejar de escuchar, solo enviar
}

void loop() {
    const char text[] = "Hola desde TX!";
    bool report = radio.write(&text, sizeof(text)); // Enviar datos

    if (report) {
        Serial.println("Datos enviados con éxito");
    } else {
        Serial.println("Error al enviar datos");
    }

    delay(1000); // Enviar cada segundo
}
```

3. Código para el Receptor (RX)

Carga el siguiente código en el dispositivo que actuará como receptor:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9, 10); // CE, CSN

void setup() {
    Serial.begin(9600);
    radio.begin();
    radio.openReadingPipe(1, 0xF0F0F0F0E1LL); // Dirección del transmisor
    radio.setPALevel(RF24_PA_HIGH); // Potencia alta
    radio.setChannel(76); // Canal de comunicación
    radio.startListening(); // Iniciar escucha
}

void loop() {
    if (radio.available()) {
        char text[32] = ""; // Buffer para recibir datos
        radio.read(&text, sizeof(text)); // Leer datos
        Serial.print("Datos recibidos: ");
        Serial.println(text); // Mostrar datos recibidos
    }
}
```

4. Validación de la Comunicación

1. **Cargar el Código**:
   - Carga el código del transmisor en uno de los dispositivos (TX).
   - Carga el código del receptor en el otro dispositivo (RX).
2. **Monitorear la Salida**:
   - Abre el Monitor Serial en el entorno de desarrollo (IDE) de Arduino o en el IDE de ESP32.
   - Configura la velocidad del Monitor Serial a **9600 bps**.
3. **Prueba de Comunicación**:
   - Asegúrate de que ambos dispositivos estén encendidos y en un rango de operación.
   - Observa el Monitor Serial del receptor para verificar que esté recibiendo los mensajes enviados por el transmisor.

Posibles Problemas y Soluciones

- **Sin Datos en el Monitor Serial**:
  - Verifica las conexiones y asegúrate de que están bien alimentados.
  - Asegúrate de que las direcciones (pipes) coincidan entre el TX y el RX.
- **Datos Corruptos**:
  - Asegúrate de que la distancia entre los módulos no sea demasiado grande.
  - Verifica que no haya interferencias de otros dispositivos que operen en la misma frecuencia.
- **No Se Envía el Mensaje**:
  - Asegúrate de que el módulo esté bien conectado y funcione correctamente.

Extensiones Futuras

- **Enviar Diferentes Tipos de Datos**: Modificar el código para enviar diferentes tipos de datos (números, caracteres, etc.).
- **Implementar un Protocolo de Confirmación**: Asegurarse de que los datos se han recibido correctamente.
- **Usar Sensores**: Conectar sensores a uno de los dispositivos y enviar datos del sensor al otro dispositivo.

Con este proyecto, deberías poder establecer una comunicación efectiva entre un transmisor y un receptor utilizando RF.

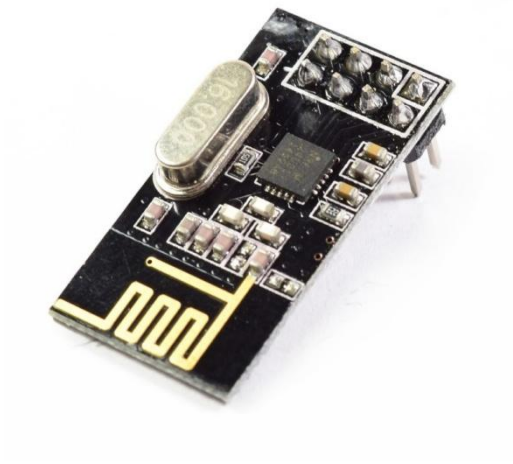
Conclusión

La implementación de la comunicación de radiofrecuencia (RF) entre un transmisor y un receptor utilizando módulos nRF24L01 y placas ESP32 o Arduino es un proyecto educativo valioso que permite explorar los conceptos de la transmisión de datos inalámbrica. A través de este proyecto, se ha aprendido a configurar hardware y software para establecer una conexión RF, enviar y recibir datos, y validar la comunicación entre dispositivos.

Este tipo de tecnología es esencial en aplicaciones de IoT (Internet de las Cosas), donde la comunicación entre dispositivos es fundamental. Las habilidades adquiridas en este proyecto son aplicables a una variedad de aplicaciones, desde la automatización del hogar hasta el monitoreo remoto de sensores. Al entender los fundamentos de la comunicación RF, se abre un amplio espectro de posibilidades para futuros proyectos e innovaciones.

ANEXO : TUTORIAL BÁSICO NRF24L01 CON ARDUINO 309759

En este tutorial se realizará la comunicación inalámbrica entre dos Arduino usando los módulos transceiver NRF24L01, explicaremos lo básico de estos transceptores y como usar su librería en Arduino.



En el mercado existen varios modelos de módulos del NRF24L01, todos operan en la banda de 2.4GHz, son muy usados por su funcionalidad, bajo consumo y bajo costo, los más populares son los que se muestran en la imagen anterior, El más básico y económico es el que se muestra en la imagen de la izquierda, básicamente es el chip NRF24L01 y sus componentes necesarios para su funcionamiento. El otro modelo es más completo, aparte del NRF24L01 posee un circuito amplificador de potencia (PA), un circuito amplificador de bajo

ruido (LNA) además de una antena SMA que en conjunto le permiten lograr un rango de hasta 1000m
Existe bastante información en Internet sobre el uso de estos transceptores, en el caso de Arduino, existen varias librerías (aquí las recomendadas) Pero para nuestro tutorial usaremos la librería RF24.

Lo primero a hacer es descargar la última versión librería RF24 e importarla a tu IDE de Arduino, para esto en la barra de menús vas a Sketch>Importar Librería>Add Library, y seleccionas la carpeta o archivo ZIP descargado, después de esto deberás cerrar y volver a abrir el IDE.

Ahora explicando las funciones que usaremos de la librería; solo se explicaran las usadas en este tutorial.

RF24 (uint8_t _cepin, uint8_t _cspin)

Función Constructor: Crea una nueva instancia (Objeto) de este dispositivo. La instancia se crea con los pines de comunicación SPI pero es necesario especificar los pines de control que están conectados al módulo.
Parámetros
_cepin : Pin del Arduino conectado al pin Chip Enable (CE) del módulo
_cspin : Pin del Arduino conectado al pin Chip Select (CS) del módulo
Ejemplo:

```
#define CE_PIN 9
#define CSN_PIN 10

RF24 radio(CE_PIN, CSN_PIN);
```

void begin(void)
Inicializa el objeto creado, generalmente se llama a esta función en setup (), antes de llamar a cualquier otro método.
Ejemplo:

```
radio.begin();
```

void openWritingPipe(const uint8_t * *address*)
Abre un canal de comunicación de escritura. La dirección se asigna a través de una matriz de 5 byte de longitud.
Parámetros
Address : La dirección del canal para abrir.
Ejemplo

```
byte direccion[5] ={'c','a','n','a','l'};

radio.openWritingPipe(direccion);
```

bool escribir(const void * buf ,uint8_t len)
Envía un dato por el canal definido en **openWritingPipe(),** el dato es de máximo 32bytes
Parámetros
Buf: Puntero al datos que se enviará

Len: Número de bytes a enviar

Devuelve
True si la carga útil fue entregada con éxito y falso si no se logró enviar
ejemplo:

```
int dato=65;

bool ok = radio.write(&dato, sizeof(dato));
```

void openReadingPipe(uint8_t number, const uint8_t * address)
Abre un canal de comunicación de lectura. La direccion se asigna a través de una matriz de 5 byte de longitud. Se pueden abrir hasta 6 canales de lectura pero sus direcciones solo pueden ser diferentes en el primer byte
Parámetros
Number : número de canal del 0-5 (canal 0 generalmente reserva para escritura).
Address : Dirección del canal para abrir.
Ejemplo:

```
byte direccion[5] ={'c','a','n','a','l'};

radio.openReadingPipe(1, direccion);
```

void startListening(void)

Al llamar esta función se empieza a escuchar por los canales definidos como lectura, después de llamar a esta función no podemos hacer escrituras, para esto antes debemos llamar a stopListening ()
Ejemplo:

```
radio.startListening();
```

bool available (void)

Compruebe si hay bytes disponibles para ser leídos
Devuelve
True si existen datos disponibles en el canal de lectura y falso si no hay ningún dato recibido
Ejemplo

```
if(radio.available()){

  radio.read(&data,sizeof(data));

}
```

void read(void * buf, uint8_t len)
Leer los datos disponibles en el canal de lectura
Parámetros
Buf : Puntero a un buffer donde los datos deben ser escritos
Len : El número de bytes a leer en el búfer

Ejemplo

```
int dato;

if(radio.available()){

  radio.read (&dato,sizeof(dato));

}
```

Una vez explicado lo necesario para empezar a usar los NRF24L01, vamos a ver un ejemplo:

Enviar variables desde un arduino a otro usando NRF24L01

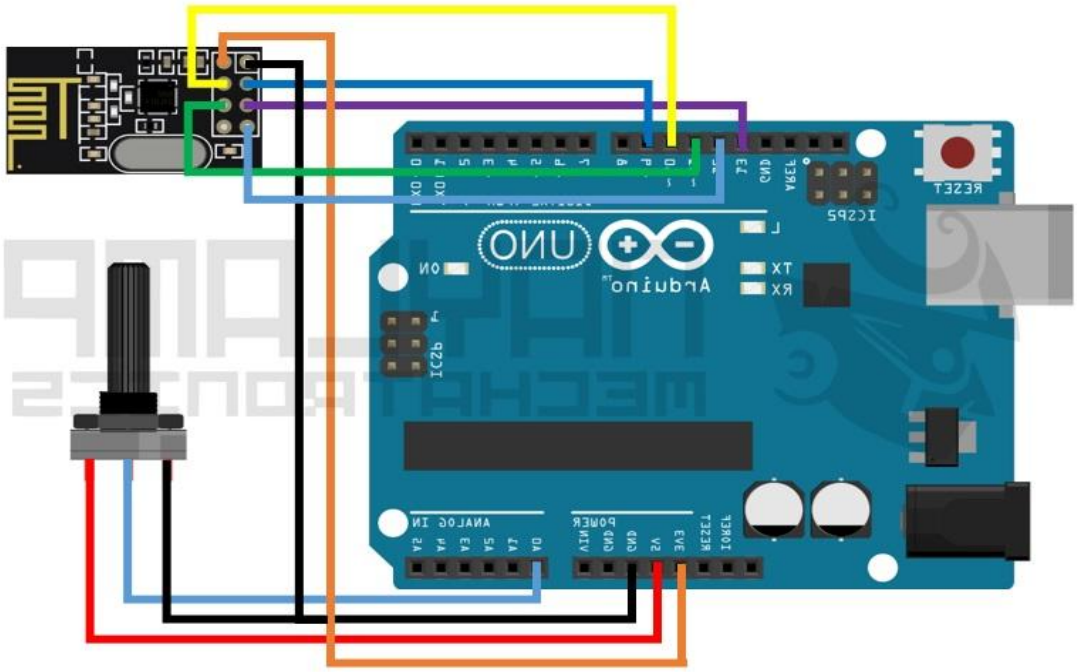
Se enviaran desde un Arduino hacia otro Arduino tres datos, estos serán: el voltaje leído por el pin analógico A0, el valor de millis(), y un dato adicional que podría ser el de cualquier sensor.

Para esto necesitamos 2 Arduinos, dos módulos NRF24L01 (con o sin antena), un potenciómetro y cables Dupont

Haciendo las conexiones entre el arduino y el NRF24L01

Las conexiones serán las mismas tanto en el Emisor y Receptor, con la única diferencia que en el receptor no usaremos el potenciómetro.

NRF24L01	Arduino UNO,Nano	Arduino Mega
1: GND	pin GND	pin GND
2: VCC	pin 3V3	pin 3.3V
3: CE	pin 9	pin 9
4: CSN	pin 10	pin 10
5: SCK	pin 13	pin 52
6: MOSI	pin 11	pin 51
7: MISO	pin 12	pin 50



Como se observa en la imagen el pin 2 del NRF24L01 (VCC) va conectado al pin 3V3 del Arduino, esto porque el modulo funciona con 3.3V. NO conectar a 5V porque podemos quemar al módulo, los pines de datos lo estamos conectando directamente al Arduino a pesar que los niveles lógicos del NRF24L01 son también de 3.3V, esto con el tiempo podría afectar al NRF24L01 por lo que se recomienda usar un adaptador para que trabaje con valores de 5V

El código para el emisor es el siguiente:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

//Declaremos los pines CE y el CSN
#define CE_PIN 9
#define CSN_PIN 10

//Variable con la dirección del canal por donde se va a transmitir
byte direccion[5] ={'c','a','n','a','l'};

//creamos el objeto radio (NRF24L01)
RF24 radio(CE_PIN, CSN_PIN);

//vector con los datos a enviar
float datos[3];

void setup()
{
  //inicializamos el NRF24L01
  radio.begin();
```

```
 //inicializamos el puerto serie
 Serial.begin(9600);

//Abrimos un canal de escritura
 radio.openWritingPipe(direccion);

}

void loop()
{
 //cargamos los datos en la variable datos[]
 datos[0]=analogRead(0)* (5.0 / 1023.0);;
 datos[1]=millis();
 datos[2]=3.14;
 //enviamos los datos
 bool ok = radio.write(datos, sizeof(datos));
  //reportamos por el puerto serial los datos enviados
  if(ok)
  {
    Serial.print("Datos enviados: ");
    Serial.print(datos[0]);
    Serial.print(" , ");
    Serial.print(datos[1]);
    Serial.print(" , ");
    Serial.println(datos[2]);
  }
  else
  {
    Serial.println("no se ha podido enviar");
  }
  delay(1000);
}
```

Y el código Para el recetor:

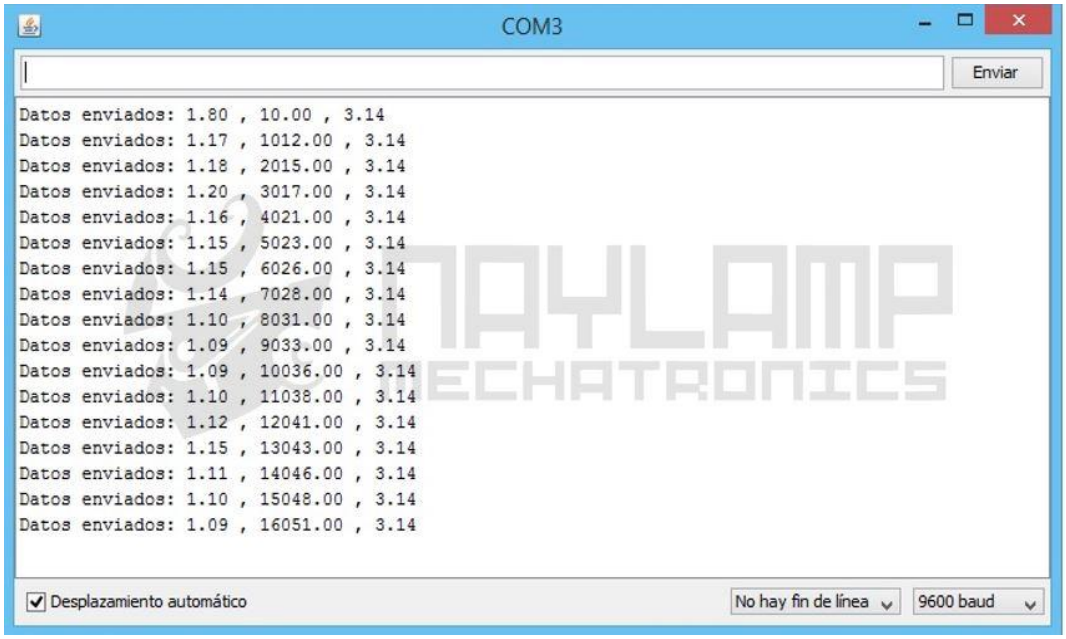
```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

//Declaremos los pines CE y el CSN
#define CE_PIN 9
#define CSN_PIN 10

//Variable con la dirección del canal que se va a leer
byte direccion[5] ={'c','a','n','a','l'};

//creamos el objeto radio (NRF24L01)
RF24 radio(CE_PIN, CSN_PIN);

//vector para los datos recibidos
float datos[3];

void setup()
{
 //inicializamos el NRF24L01
  radio.begin();
  //inicializamos el puerto serie
```

```
 Serial.begin(9600);

 //Abrimos el canal de Lectura
 radio.openReadingPipe(1, direccion);

  //empezamos a escuchar por el canal
 radio.startListening();

}

void loop() {
 uint8_t numero_canal;
 //if ( radio.available(&numero_canal) )
 if ( radio.available() )
 {
    //Leemos los datos y los guardamos en la variable datos[]
    radio.read(datos,sizeof(datos));

    //reportamos por el puerto serial los datos recibidos
    Serial.print("Dato0= " );
    Serial.print(datos[0]);
    Serial.print(" V, ");
    Serial.print("Dato1= " );
    Serial.print(datos[1]);
    Serial.print(" ms, ");
    Serial.print("Dato2= " );
    Serial.println(datos[2]);
 }
 else
 {
    Serial.println("No hay datos de radio disponibles");
 }
 delay(1000);
}
```

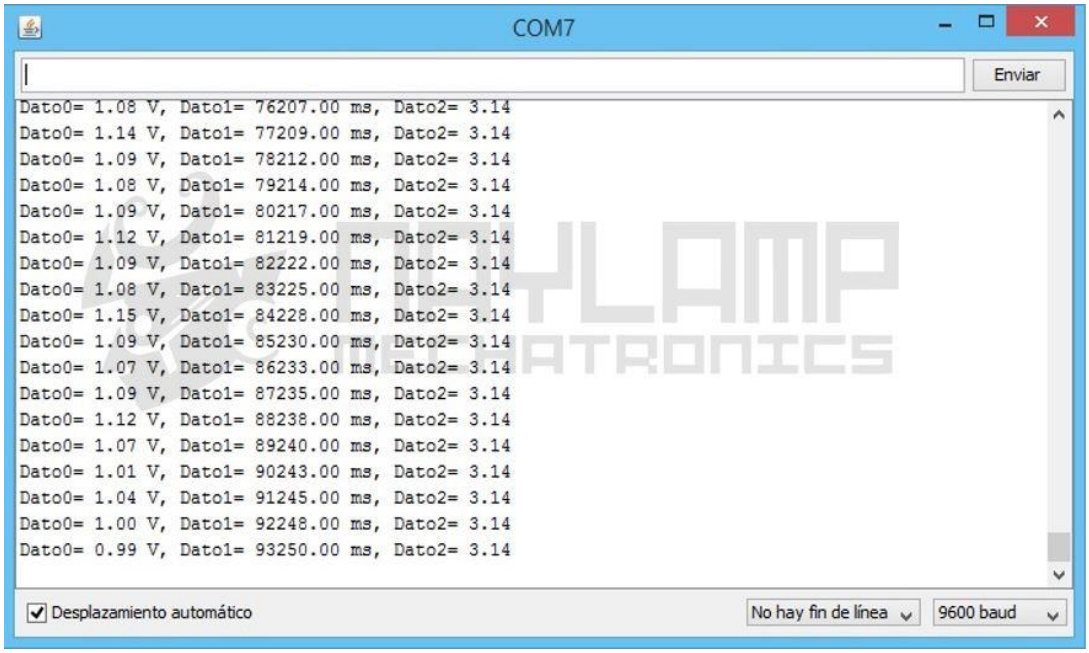
Si analizamos el código lo que hacemos es inicialmente configurar el módulo y luego enviar o leer los datos transmitidos por el módulo NRF24L01, la variable que se va a transmitir puede ser un solo dato o un array de datos como se lo está haciendo en este tutorial, pero siempre la variable o vector que se va a recibir tiene que ser del mismo tamaño y tipo que la variable enviada, de lo contrario se perderán datos.

Si han hecho bien todo lo descrito en el tutorial, al abrir el monitor serial les debería mostrar los siguientes resultados

Monitor serial del Emisor:



Monitor serial del receptor



La distancia o alcance entre módulos NRF24L01 dependerá del modelo que están usando y del lugar en donde están trabajando, si hay muros, ruido, o si estan en lugares abiertos. Los que vienen sin antena son de poco alcance un poco menos que la señal WIFI, pero los modelos que vienen con antena y amplificador de potencia pueden comunicarse hasta un 1km de distancia.

Referencias Bibliográficas

1.Libros y Manuales:
Monk, S. (2019). Programming Arduino: Getting started with sketches. McGraw-Hill Education.
Margolis, M. (2011). Arduino cookbook. O'Reilly Media.
2. Documentación y Tutoriales:
TUTORIAL BÁSICO NRF24L01 CON ARDUINO
https://naylampmechatronics.com/blog/16_tutorial-basico-nrf24l01-con-arduino.html
3. Artículos y Blogs:
Stack Overflow. (n.d.). nRF24L01 module - getting started.
https://stackoverflow.com/questions/tagged/nrf24l01
Random Nerd Tutorials. (n.d.). Getting started with nRF24L01 and Arduino: Complete guide.
https://randomnerdtutorials.com/
4. Foros y Comunidades:
Arduino Forum. (n.d.). Arduino forum - nRF24L01 discussion. https://forum.arduino.cc/