Universidad del País Vasco Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

RED INALÁMBRICA DE SENSORES PORRADIOFRECUENCIA

ANEXO II – FORMACIÓN DE LA RED

Alumno/Alumna: Dieguez Martín Alexander

Director/Directora (1): Oleagordia Aguirre Iñigo Javier

Curso:2018-2019

Fecha:<XXXX, día, mes, año>

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

A continuación se va a presentar las posibles opciones y requerimientos para la formación de una red inalámbrica de sensores, donde en este caso además del sensor utilizado en la memoria DHT11, se va a incorporar un sensor ultrasónico.

El número de nodos emisores que se pueden ir incorporando a la red lo limitan las tarjetas de adquisición, encargadas de poder controlar y suministrar energía a dichos sensores. Estas tarjetas son de Arduino y concretamente son NANO. En caso de querer formar una red mucho más grande de sensores, Arduino dispone de tarjetas mucho más potentes y con mayor número de entradas tanto digitales como analógicas con las que poder controlar un mayor número de nodos.

Se expondrá mediante este anexo algunos problemas que se pueden encontrar así como las posibles colisiones que se pueden producir a la hora de enviar la información desde varios nodos de forma simultánea, y del uso de protocolos para poder subsanar este problema. Además de explicar de forma más detallada el tipo de comunicación que disponen los módulos de radiofrecuencia de 433mhz

2. TIPO DE COMUNICACIÓN

Los módulos de radiofrecuencia de 433mhz tienen un tipo de comunicación llamado simplex, es decir, en un solo canal y unidireccional (uno envía y otro recibe). Con ello tienen una baja velocidad de transmisión (entorno a los 2400bps). Se realiza básicamente por modulación ASK (Amplitude Shift Keying). No disponen de filtro ni ID por hardware, por lo que si se quiere una comunicación robusta hay que implementarlo programando el propio Arduino para ello. Es por ello que este tipo de redes son ideales para realizar proyectos de bajo coste y el alcance de la señal no sea muy amplia. Para proyectos en los que este tipo de características sean de total relevancia existen otro tipo de redes inalámbricas en las que permite realizar el envío de información a largas distancias y disponen de un software propio donde poder configurar la red de una forma mucho más compleja y robusta, como pueden ser los módulos XBee.

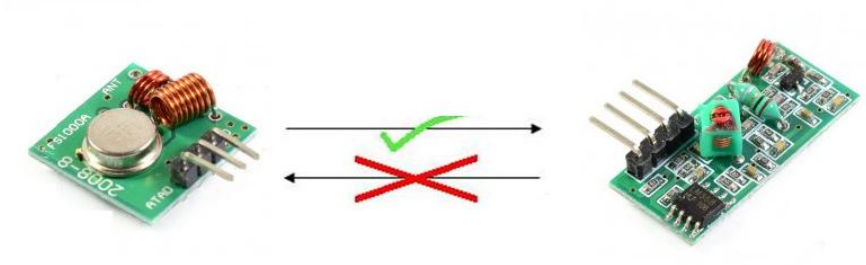

Figura 2.1 Tipo de comunicación Simplex.

3. CONFIGURACIÓN DE LA RED

Existen varios métodos con los que configurar una red inalámbrica , según los módulos a utilizar, distancia límite de la red con la que se quiere establecer la comunicación, instrumentación a utilizar o tipos de sensores a incorporar. Es por ello que según las características de nuestra red en los que se utilizan los módulos de radiofrecuencia, se ha provisto a la red con varios nodos emisores y un solo receptor, es decir, por cada sensor a incorporar a la red se dispone de su nodo de radiofrecuencia emisor con el que recoger los datos o información del sensor en cuestión mediante una tarjeta de Arduino, concretamente NANO.

Lo bueno de configurar una red donde se tienen x nodos emisores en función del número de sensores a utilizar, es que en caso de tener un fallo en alguno de ellos, únicamente dejaría de funcionar ese nodo afectado y no afectaría al funcionamiento del resto de ellos, ya que están implementados de forma independiente.

A la hora de recoger la información en el nodo receptor de radiofrecuencia hay que tener en cuenta que el tipo de comunicación que utilizan estos módulos es simplex, por lo que se tiene que configurar un protocolo de comunicación para poder evitar así posibles colisiones en la recepción de los mensajes.

4. PROTOCOLOS DE COMUNICACIÓN

Para poder mantener una comunicación eficaz entre los distintos nodos que componen la red, y así evitar colisiones entre ellos es necesario implementar en el sistema un protocolo de comunicación.

Existen varios tipos en los que poder escoger en función a las necesidades de la red en cuestión. A continuación se expondrán tres de los protocolos que mejor se ajustan a las necesidades de nuestra red.

4.1 PROTOCOLO DE COMUNICACIÓN POLLING O POR SONDEO

Este método de acceso se caracteriza por contar con un dispositivo controlador central, que se trata de un ordenador inteligente, en nuestro caso se trata de una tarjeta de Arduino NANO. En caso de implementar muchos sensores a la red y tener la necesidad de gestionar toda esa información, se podría realizar un cambio de la tarjeta NANO a MEGA, el cual dispone de mejores prestaciones tanto en el microcontrolador, como en el tipo de encapsulado.

Este ordenador inteligente haría la función de un servidor, el cual pasa lista a cada nodo en una secuencia predefinida solicitando el acceso a la red. Si tal solicitud se realiza, el mensaje se transmite; de lo contrario, el dispositivo central se mueve a pasar lista al siguiente nodo. Como principal desventaja que posee, se tiene que si falla el

control central, la red no funciona. Se trata de un método equitativo, pues cada nodo dispone del mismo tiempo que el resto.

Este método no dispone de prioridades, es por ello que en caso de que un nodo necesite enviar su información de manera urgente, deberá esperar hasta que le llegue el turno para poder mandar esa información. Para este tipo de protocolos en los que se necesitan distribuir una serie de prioridades entre los distintos nodos según su importancia en el funcionamiento del sistema, se tiene un tipo de sondeo por lista. Se trata de un control centralizado, donde el controlador dispone de una lista de las direcciones de los nodos. Se seleccionan aquellos nodos por orden de lista. Si se desea que un nodo posea mayor prioridad, éste se incluye varias veces en la lista.

Se trata de una forma tradicional de repartir un canal entre varios nodos que compiten entre sí para el envío de información. Es por ello que así se evitan las colisiones por medio de un controlador central que cuestiona a cada uno de los nodos para ver si tienen datos que transmitir, en caso de si tenerlos les permite el uso del canal, de lo contrario se sigue preguntando al siguiente nodo.

4.2 PROTOCOLO DE COMUNICACIÓN POR SELECCIÓN

Esta técnica también puede ser centralizada o distribuida, no se producen colisiones dado que no se accede al medio hasta que el canal es asignando al nodo, asegurando que un único nodo accede en cada momento al bus. Dentro del protocolo de comunicación existen dos técnicas diferenciadas, en los que uno de ellos se ha expuesto en el apartado anterior "*Polling*". La otra técnica se llama "*Daisy Chainning*".

La técnica Daisy Chainning es la misma que se utiliza en los buses internos de los ordenadores. Necesita un canal extra (hilo) que recorra en anillo las estaciones o nodos, siendo un bus el canal que utilizan para enviar los datos. A través de este hilo extra se envían pulsos, cuando un nodo es seleccionado mediante un pulso, toma el control del medio para enviar sus mensajes, devolviendo el pulso a la siguiente estación física en el anillo al finalizar su transmisión. Si al recibir el pulso no tiene nada que transmitir, lo pasa a la siguiente estación. El usuario toma el control del canal, avisando cuando finaliza su utilización.

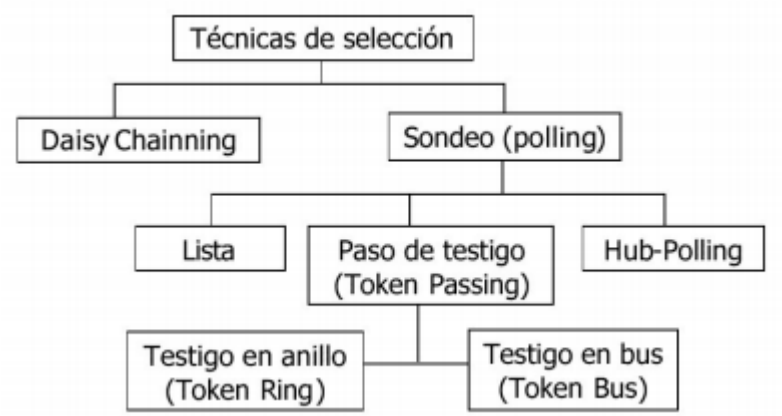

Figura 4.2.1 Árbol de técnicas de protocolos de comunicación.

4.3 PROTOCOLO DE COMUNICACIÓN POR INTERRUPCIÓN

Este tipo de comunicación está disponible para aquellos dispositivos que demandan mover muy poca información y con poca frecuencia. Su paquete de datos tiene las mismas dimensiones que el de las transmisiones de control.

Los sensores avisan al nodo principal o receptor cuando están listos para ser atendidos. El nodo receptor una vez ha iniciado al sensor lo que desea, puede dedicarse a realizar otras cosas. Al diseñar un sistema de entrada-salida mediante interrupciones, hay que tener en cuenta los siguientes factores:

- Cómo se solicita la interrupción.
- Cómo se indica la aceptación de la interrupción.
- Cómo se identifica al sensor que ha interrumpido.
- Cómo se resuelve el esquema de prioridad si varios sensores interrumpen de manera simultánea.

El proceso para la transmisión de daros es el siguiente. El sensor realiza la petición de servicio mediante una señal de control específica. Cuando la interrupción es aceptada por el nodo receptor o principal, ésta abandona momentáneamente el programa principal para ejecutar la rutina de tratamiento de la interrupción y realizar así la transferencia de información con dicho sensor.

Cuando existen varios sensores que pueden interrumpir se deben gestionar las prioridades si realizan peticiones simultáneas. Para ello se debe tener en cuenta lo siguiente:

- Cómo conectar los sensores al nodo receptor o principal.
- Cómo establecer la política de prioridades.
- Cómo determinar la dirección de la rutina del tratamiento de interrupción.

Conexión de todos los sensores a una única línea de interrupción. Para ello todos los sensores solicitan ser atendidos por la misma línea. La dirección de la rutina de tratamiento de la interrupción es fija y común para todos ellos. La rutina de tratamiento identifica mediante sondeo cuál de los sensores ha interrumpido al nodo receptor o principal. Por último la prioridad se da por el orden en el que se sondea a los sensores (0, 1, 1, 1, 2, 3, n, 0, 1, 1, 1, 1).

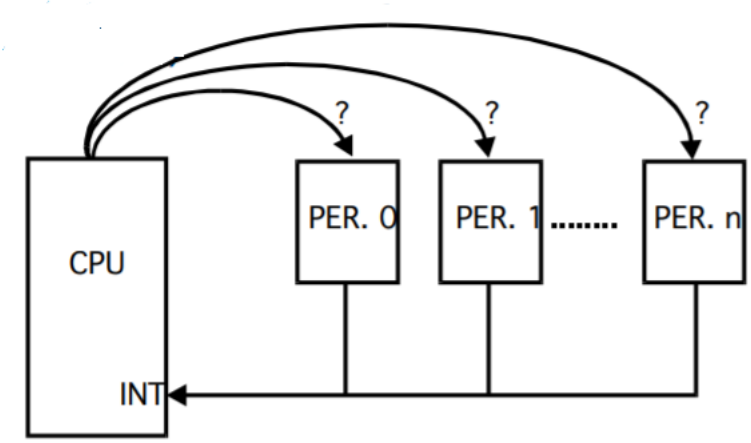

Figura 4.3.1 Esquema con una única línea de interrupción.

5. IMPLEMENTACIÓN EN ARDUINO

Como la comunicación entre los diferentes módulos de radiofrecuencia es unidireccional "Simplex", a partir de la incorporación de más de un nodo emisor a la red, es necesario la implementación de un protocolo de comunicación en la programación para evitar así colisiones en el transcurso de información entre los distintos emisores y el receptor. Para la verificación de que la información a llegado al destino de manera satisfactoria es necesario agregar tramas de validación.

Una de las librerías más comunes y eficientes para la transferencia de datos es la "VirtualWire". A continuación se van a exponer las funciones principales de las que dispone esta librería.

- **void vw_setup(uint16_t speed):** Inicializa el software VirtualWire, como parámetro hay que indicarle la velocidad de operación, que representa los bits por segundo para la transmisión RF.
- **void vw_set_tx_pin(uint8_t pin):** Estable el pin IO digital por donde se va a transmitir los datos.
- **void vw_set_rx_pin(uint8_t pin):** Estable el pin digital IO por donde se va a recibir datos.
- **void vw_rx_start():** Empieza a escuchar los datos provenientes por el pin_rx, es necesario llamar a esta función para poder recibir los datos.

- **uint8_t vw_send(uint8_t * buf,uint8_t len):** Enviar un mensaje con la longitud dada. La función termina rápido, pero el mensaje será enviado en el momento adecuado establecido por las interrupciones. Donde **buf**, es el puntero al vector para transmitir, y **len** es el número de bytes a transmitir.
- **void vw_wait_tx():** Hace una pausa hasta que se transmitan todos los datos.
- **uint8_t vw_get_message(uint8_t * buf, uint8_t * len)**: Si un mensaje está disponible (con buena suma de comprobación o no), almacena el mensaje en **buf**, devuelve true si la comprobación es correcta, **buf** es el puntero a la ubicación para guardar los datos de lectura y **len** es un puntero a la cantidad de bytes disponibles de **buf**.

Como es posible que la red inalámbrica tenga más de un sensor, y por ello la necesidad de enviar varios datos, existen dos formas con las que poder hacerlo. Un método es empaquetar todo en una sola trama y enviarlo, el receptor debe de desempaquetar y obtener los datos. La otra forma más sencilla que la anterior, pero no por ello menos efectiva, es enviar un inicio de trama diferente para cada dato, el cual indicara al receptor que dato es el que se está enviando. A continuación se añade un pequeño *script* de Arduino donde se puede apreciar la configuración de la trama.

En primer lugar el script del nodo emisor:

```
#include <VirtualWire.h>
#include "LowPower.h"
#include "DHT.h"

#define DHTPIN 8
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
int ledPin = 13;
char Msg[30];
const int dataPin = 2;
void setup()
{
  Serial.begin(9600);
  dht.begin();                  //Inicializamos el sensor DHT
  pinMode(ledPin,OUTPUT);
  vw_setup(2000);
  vw_set_tx_pin(dataPin);
```

```
}


void loop()
{
  //delay(5000);
  int humidity = dht.readHumidity();
  int temp = dht.readTemperature();
  int f = dht.readTemperature(true);
  int hi_f = dht.computeHeatIndex(f,humidity);       //Indice de calor en fahrenheit
  int heat_index  =(hi_f-32)*5/9;              //Pasamos el indice de calor a grados Celsius

  String str;
  char buf[30];

  sprintf(buf, "%d,%d,%d", humidity,temp ,heat_index);

  int dataInt = millis() / 1000;
  str = "i" + String(dataInt);                  //Convertir a string
  str.toCharArray(buf,sizeof(buf));             //Convertir a char array
  vw_send((uint8_t *)buf, strlen(buf));         //Enviar array
  vw_wait_tx();                                 // Esperar envio

  delay(200);
}
```

En segundo lugar y por último se presenta el script para la parte del receptor:

```
#include <VirtualWire.h>
#include <LiquidCrystal_I2C.h>
#include <VirtualWire.h>
```

```
const int dataPin = 2;
int humidity=0;
int temp=0;
int heat_index=0;
char MsgReceived[21];
int led = 13;

void setup()
{
   Serial.begin(9600);
   lcd.begin(20,4);
   lcd.backlight();
   pinMode(led, OUTPUT);
   vw_setup(2000);            //Velocidad de bits por segundo
   vw_set_rx_pin(dataPin);   //Pin 2 como entrada del RF
   vw_rx_start();             //Se inicia como receptor

   lcd.begin(20,4);
   lcd.backlight();
   lcd.createChar(1, thermometer);
   lcd.createChar(2, droplet);
   lcd.createChar(3,hi);
   lcd.clear();
}

void loop()
{
   uint8_t buf[VW_MAX_MESSAGE_LEN];
   uint8_t buflen = VW_MAX_MESSAGE_LEN;

   // Recibir dato y se verifica si hay uin dato valido en el RF
```

```
  if (vw_get_message((uint8_t *)buf,&buflen))
  {
    String dataString;
     if((char)buf[0]=='i')                //Se verifica el inicio de la trama
     {
        for (int i = 1; i < buflen; i++)
        {
        dataString.concat((char)buf[i]);
        Serial.print((char)buf[i]);
        MsgReceived[i] = char(buf[i]);
        }
        int dataInt = dataString.toInt();                    //Convertir a int
        sscanf(MsgReceived,"%d,%d,%d",&humidity,&temp,&heat_index);
        //Convierte de un string a un array
        digitalWrite(led, LOW);
        lcd_display();
        memset( MsgReceived, 0, sizeof(MsgReceived));
        //Linea donde se resetea el StringReceived
     }
     else if((char)buf[0]=='f')        //Se verifica el inicio de trama del sensor ultrasonico
    {
      for (i = 1; i < buflen; i++)
      {
      DatoCadena.concat((char)buf[i]);
      }
        dato2=DatoCadena.toFloat();
        Serial.print("Dato2 recivido: ");
        Serial.println(dato2);
      }
    }
}
```

De esta forma cada sensor a incorporar en la red inalámbrica se le asigna un principio de trama. Como en este caso al sensor DHT11 de temperatura y humedad la letra 'i' y al sensor ultrasónico la letra 'f'. En anterior script el receptor efectúa una pasada entre todos los sensores asignados a la red, que en este caso son dos. Empezando desde el sensor DHT11 y después el ultrasónico, es decir, realiza una recepción de información de forma secuencial. Ningún sensor de la red tiene prioridad ante el resto.

En caso de que se le quiera asignar prioridad a un sensor frente al resto. Se programa mediante el IDE de Arduino sabiendo previamente la letra asignada a ese sensor sobre la trama, para que a la hora de realizar la programación éste primero revise la información de este sensor frente al resto, o realice varias recogidas de información en un periodo de tiempo asignado antes de pasar al siguiente sensor. Esto dependería de la demanda de la red inalámbrica y de las necesidades del cliente de mantener según qué información de manera prioritaria.

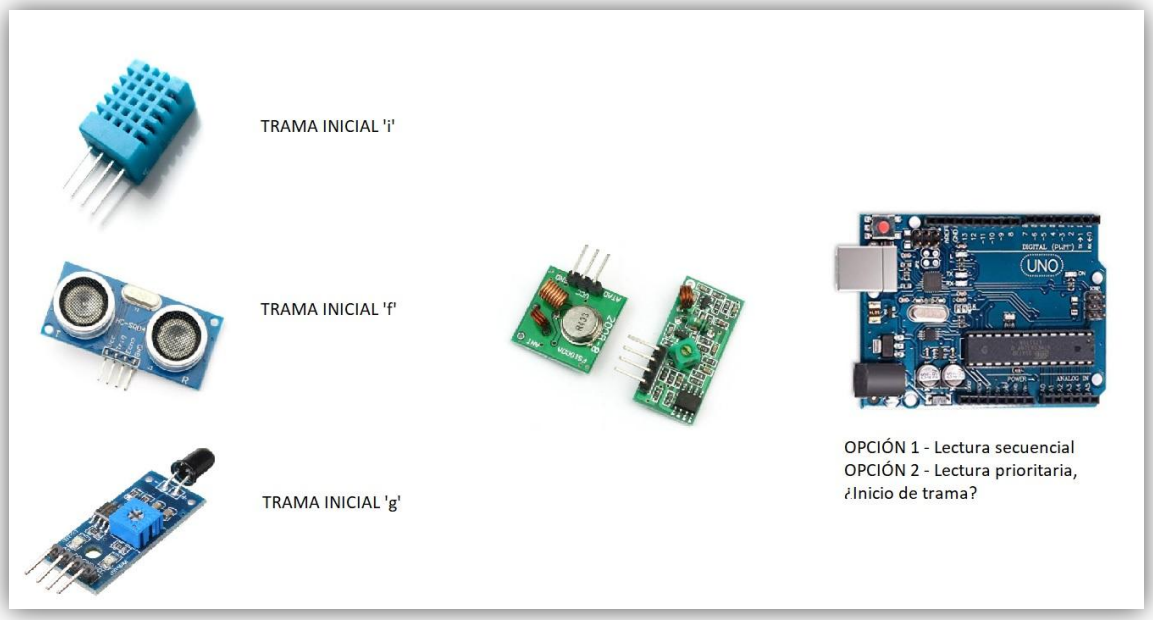

Figura 5.1 Esquema protocolos de comunicación.