

Informe cobertura inalámbrica y detección de errores g6

Integrantes: Diego Carmona
Leonardo Rodríguez
Profesor: Christian Lazo
Asignatura: Info 239
grupo: 6

20 de junio, 2023 Valdivia

Introducción:

En la era digital en la que actualmente vivimos, la transmisión de datos se ha convertido en una parte fundamental de nuestras vidas. Sin embargo, el uso de diferentes medios para la transmisión de datos tiene una alta probabilidad de errores durante la transmisión. Este fenómeno se vuelve particularmente notorio en el campo de las comunicaciones inalámbricas, donde factores como la relación señal-ruido y la atenuación espacial de la señal, entre otros, pueden provocar cambios de bits en los paquetes de datos que se envían, provocando un error durante la transmisión de datos. La transmisión inalámbrica enfrenta desafíos especiales que implican mejorar su rendimiento y/o fiabilidad de transmisión y la reparación de errores.

Uno de los principales desafíos en las comunicaciones inalámbricas es el efecto de la relación señal-ruido mencionado anteriormente. Esta es la relación entre la potencia de la señal transmitida y el nivel de ruido presente en el medio de transmisión. A medida que la señal se debilita debido a la distancia y la atenuación, el ruido se vuelve más prominente, lo que reduce la calidad de la señal y aumenta la posibilidad de errores en la transmisión de datos. Además, la transmisión inalámbrica en el espacio también puede verse afectada por las condiciones ambientales y los obstáculos, lo que resulta en una degradación de la señal, la atenuación de la señal provoca errores durante la transmisión de datos, ya que los bits del paquete transmitido pueden modificarse en su recorrido.

Este trabajo busca experimentar de manera cercana con la transmisión de datos con señales inalámbricas, como asimismo implementar sistemas de control de errores en dicha transmisión que ayuden a mejorarla. Al estar trabajando con sistemas de comunicación basados en arduino, transmisores y receptores, estaremos trabajando sobre la capa de enlace de los datos. Nuestro objetivo principal será determinar la cobertura óptima de las antenas en posesión, para esto se deberá investigar, diseñar y construir métodos, como así también algoritmos que nos aproximen a lo buscado.

Para poder controlar los errores surgidos durante la transmisión de paquetes de datos, se usará CRC-5-USB el cual es un algoritmo de detección de errores, usa una representación polinómica: $x^5 + x^2 + 1$, para funcionar realiza algunas operaciones con los datos y el polinomio, luego se agrega a lo que se enviará para que el receptor calcule las mismas operaciones y comprueba el mensaje.

Al finalizar este informe se entregarán los resultados y se discutirá los puntos que podríamos mejorar y lo que podemos concluir, junto a lo que nos dejó esta experiencia de arduino, virtual wire, protocolos y comunicaciones inalámbricas.

Metodología:

¿Qué se quiere lograr? En esta investigación, nuestro objetivo principal es desarrollar un método efectivo para eliminar el ruido y garantizar una transmisión precisa de mensajes. Este enfoque tiene implicaciones más amplias, ya que exploramos la extensión de la propagación de mensajes y el impacto del tamaño de la antena. Además, en nuestro experimento final, buscamos lograr la transmisión exitosa de mensajes entre dos dispositivos en un entorno ruidoso donde varios arduinos están enviando mensajes y que aun así se logre mantener la integridad del mismo.

Diseño de la investigación: Este estudio es de naturaleza experimental y se centra en la configuración de dos dispositivos Arduino para comunicarse entre sí utilizando la biblioteca VirtualWire. En nuestro experimento, el dispositivo Arduino que actúa como transmisor se configura con el modelo RT11, mientras que el receptor se configura con el modelo RR10. Para facilitar la transmisión de señales, se han utilizado antenas de alambre recubiertas con goma, estas tienen un largo de 13 centímetros cada una. Este diseño permite que los dispositivos emisores generan una señal que se transmite a través de estas antenas caseras

Teóricamente hablando la frecuencia de una antena de látilo de longitud 13.01 cm sería aproximadamente 576.48 MHz, y para una longitud de 13 cm, la frecuencia sería aproximadamente 576.92 MHz

Usando el texto manual de antenas donde salía lo siguiente:

$$L(cm) = 7500 / \text{Freq.}(MHz)$$

A 433.92 MHz, un cuarto de longitud de onda es de 17 cm

Sin embargo nosotros necesitamos cuál es la cobertura real de nuestra antena, a lo cual se llevó a cabo experimentos a terreno, para así con los dispositivos hacer mediciones en varias direcciones para poder encontrar el tamaño práctico de cobertura de la antena.

El experimento se llevó a cabo en una cancha de fútbol, en este lugar tenía un suelo plano, y sin muchos objetos alrededor que pudiesen causar interferencia. Además se intentaba que la pantalla de los notebook estuviera fuera del rango para poder medir correctamente.

Debido a que en las primeras mediciones la distancia máxima era alrededor de 21 metros y eran muy irregulares, se decidió cortar las antenas para así facilitar las mediciones, así quedando ambas antenas de un largo aproximado 13 cm cada una. con cortes de cortes 3,4 cm y 2,2 cm.

Luego de esto se midió en 12 direcciones en el suelo y en 5 direcciones hacia el eje ZY para ver la altura. Las mediciones se hacían hasta donde los mensajes se reciben correctamente, como se hizo en un lugar sin interferencias de ningún tipo (cancha del barrio) no se detectaba ruido de ningún tipo, pero esto también fue contraproducente ya que en la práctica siempre se tiene que contar con que haya algún tipo de ruido, por lo que se espera poder probar de una manera más cercana a la realidad el dia de la prueba grupal de transmisiones y recepciones.

Para la codificación se investigó y se trató de estandarizar algunos tipos de datos, pero solo se consiguió hacer funcionar unos pares de scripts de distintos grupos, como lo veremos en la sección de resultados. Con tal de tener un control de errores de transmisión se uso un algoritmo conocido para este fin, el cual es crc-5-usb y podemos ver nuestro primer diseño del algoritmo en pseudocódigo.

```

181 Función crc5(data: arreglo de bytes, len: entero): entero sin signo de 5 bits
182     crc = 0 (inicializar el valor del CRC en 0)
183
184 Para cada byte en data:
185     crc ^= byte (realizar una operación XOR entre el byte actual y el valor
186     actual del CRC)
187
188 Para i = 0 hasta 7: (recorrer los 8 bits del byte)
189     Si el bit más significativo (MSB) de crc es 1:
190         crc = (crc << 1) XOR 0x25 (realizar un desplazamiento a la
191         izquierda y una operación XOR con 0x25)
192     Sino:
193         crc = crc << 1 (realizar un desplazamiento a la izquierda)
194 Devolver los 5 bits inferiores de crc (puede utilizar una máscara |bitwise para
195 obtener solo los 5 bits inferiores)
196 Fin de la función

```

El cual calcula el crc usando el polinomio generador 0x25, el cual su representación binaria es 100101, lo cual es $x^5 + x^2 + 1$, lo cual es lo solicitado debido a que los datos son enviados a través del usb del arduino. Luego como el resultado puede variar entre número de 1 o 2 cifras, lo que serían 1 o 2 bits, lo que se hace hacer control de esto, con una función que se asegura que crc siempre sea de 2 dígitos, en otras palabras toma el numero y si es de un dígito, le concatena un cero al inicio.

Para hacer el envío de datos a través de la función vw_send() presente en la librería virtualwire se tuvo que diseñar el paquete y seguir cierta estructura, para ello se ocupó la estructura sugerida por el docente a cargo, la cual es una cadena de pares de bits para la cabecera del mensaje y concatenado a este mismo el mensaje.

```

75 //definicion de parametros del paquete
76 const char* origen = "06";
77 const char* destino = "00"; //00 broadcast o 06 dirijo
78 const char* crc; //calculo de parametro crc
79 const char* secuencia[] = {"1","2","3","4","5"};
80 const char* total = "5";
81

```

Como se puede ver en la imagen, los datos fijos los cambiamos directamente en el código fuente según dicte la necesidad del momento, y el paquete se enviará en total 5 veces, la concatenación y el envío se hacen dentro del loop principal del script.

```

110  for (int i = 0; i < 5; i++){
111      char* partOne = concatenarParametros(origen,destino,crc,secuencia[i],total,msg);
112
113      vw_send((uint8_t *)partOne, strlen(partOne));
114      vw_wait_tx();
115      // esparamos a que el receptor verifique que es su destino = receptor
116      Serial.println("enviado paquete:");
117      Serial.println(partOne);

```

Luego de haber explicado a grandes rasgos lo que se diseñó e implementó para el envío, pasemos a ver el flujo de trabajo de la recepción. Antes de empezar cabe mencionar que el crc y la estructura de paquetes son las mismas, por lo que volver a explicarlo sería redundante, dicho esto el receptor tiene como tarea recibir paquetes de un emisor, éste puede ser 2 transmisores posibles, el mismo grupo (06) y broadcast(00), para controlar esto se usa una sentencia if de la siguiente manera:

```

148  if (((m[0] == origenA[0] && m[1] == origenA[1])||(m[0] == origenB[0] && m[1] == origenB[1]))&&primero{
149      //(
150      // trabajar broadcast || unicast(direccion) )&& primero
151      char primeraMitad[9]; // Suficiente espacio para 8 caracteres + 1 carácter nulo adicional
152      char segundaMitad[9]; // Suficiente espacio para 8 caracteres + 1 carácter nulo adicional
153      dividirCadena(m, primeraMitad, segundaMitad);
154      Serial.println(primerMitad);
155      Serial.println(segundaMitad);

```

Como se puede apreciar en los comentarios, se evalúa quién es emisor y además nuestro código usa la variable “primero” para comenzar a recibir únicamente si está llegando el primer paquete. La razón de porqué se evalúa índice a índice es que al tener estructura fija no tenemos que identificar qué dato es cual.

También se verifica si el crc presente en la cabecera de el mensaje es el mismo que el que calcula el receptor, para así medir si el mensaje no se adulteró mediante la transmisión, esto lo hacemos de la forma:

```

200
201     uint8_t mycrc = crc5((uint8_t *)segundaMitad, strlen(segundaMitad));
202     char charCrc[3];
203     sprintf(charCrc, "%u", mycrc);
204     const char* charValor = modificarString(charCrc); // validar y/o corregir longitud de crc (2 bytes)
205     if(charValor[0] == primeraMitad[4] && charValor[1] == primeraMitad[5])[] //valida el crc
206         msj = segundaMitad; // se almacena msj en variable global
207         Serial.println(msj);
208         cocrc+=1; // contador crc correctos |
209         []
210

```

Como última acotación puede resultar interesante de mencionar, la razón por la que mandamos los datos convertidos a binario, es porque al enviar datos a través de cualquier sistema de comunicación al final siempre se enviará mediante una cadena de 1s y 0s por lo que consideramos redundante hacer tal conversión.

El resto del código del receptor son el uso de contadores para verificar los datos de la transmisión, cuántos paquetes llegaron, se perdieron, crc correcto. El código fuente del trabajo está disponible en: https://github.com/leoxz98/arduino_virtual_wire_g6.git

Materiales:

A continuación se listan los materiales usados en las pruebas de campo, como también en la realización de las demás actividades.

Material	Cantidad
Arduino	2
Antenas	2
Emisor	1
Receptor	1
Cable usb-arduino	2
Protopboard	2
Cables	16
huincha	1

Antena	Largo(cm)
Emisor	13.01
Receptor	13
Cuaderno	2
Lápiz	2
Regla	1

Resultados:

En las siguientes tablas se presentarán los resultados de las mediciones de campo, realizadas para determinar el alcance práctico real de las antenas.

Grados	Largo medición(en metros)
0(norte)	10,59
30°	4,012
60°	4,012
90° (este)	13,32
120°	7,6
150°	5,83
180°(sur)	12,51
210°	7,87
240°	3,32
270°(oeste)	2,94
300°	2,88
330°	3,06

Tabla n°1: Mediciones en plano circular, cada 30°

Grados	Altura (en metros)
0°	12.51
30°	10.13
60°	7.71
90°	18.05
120°	12.01
150°	6.59
180°	10.59

Tabla n°2: medición de altura semiesfera cada 30°

Para poder visualizar de mejor manera se puede hacer un diagrama que muestra la longitud del radio de una antena, este tipo de diagrama o representación gráfica se le conoce como lóbulo de radiación de una antena.

Para la tabla 1 se graficó un círculo subdividido en cortes de 30° cada uno, esto guiado por la brújula y las 4 direcciones cardinales, esto también está anotado en la tabla mencionada. El lóbulo de radiación para nuestra antena resultó ser el siguiente.

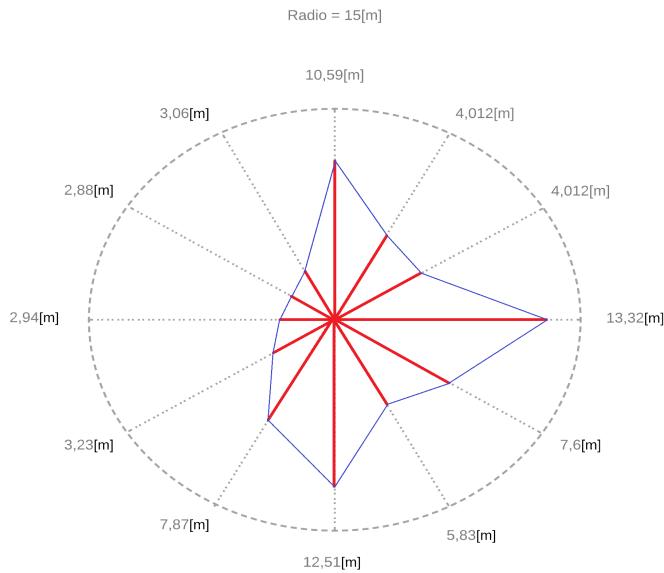


Fig n°1 Lóbulo de radiación plano

En la figura 1 se puede apreciar que la antena proporcionada no está con su lóbulo de radiación óptimo, por lo que podemos corroborar la información que se dio de que las antenas estaban modificadas para evitar obtener resultados “sacados del libro”.

Para la altura también se hicieron mediciones, como se especificó en la metodología, lo que se hizo fue voltear la antena transmisora para simular la altura, pero en el plano. Desde la tabla n°2 se desprende el siguiente lóbulo de radiación de la antena, pero mostrando la altura, cabe recordar que aquí también se separó cada 30° .

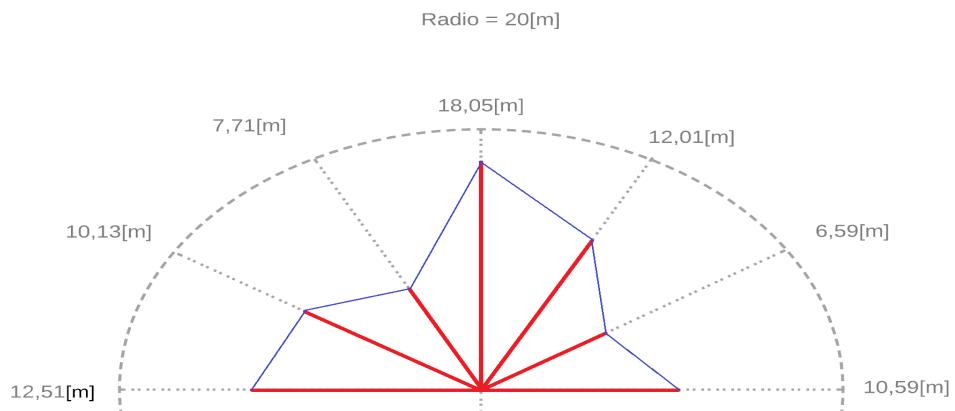


Fig n°2 Lóbulo de radiación altura

Para la altura obtuvimos resultados que nos llevan una conclusión igual que a las mediciones del plano, la antena no está operando con su capacidad normal

Todas estas mediciones fueron realizadas por los 2 miembros del grupo en terreno. Algunas fotografías del procedimiento estarán al final del informe, en un anexo

Para los experimentos con ruido se realizó lo siguiente(dia 20 de junio)

Al no haber cambiado el código, en el cual enviaremos la mensaje como un arreglo de caracteres, hubo problemas en la comunicación, esto fue debido a que lo que nosotros enviamos era como tal el char de los valores en sí, por lo tanto cuando enviabamos “1” no era el entero de “1” si no que era el carácter “1” por lo cual nos daba valores diferentes a los demás, nosotros estábamos guardando los valores en un arreglo char, esto hacía que cuando intenta insertar un uint8_t que era el valor que entregaba el buffer el arreglo en si no pudiera cambiarse. Para comenzar el comportamiento del código es el siguiente cuando enviamos dentro de nuestro propio grupo.

```
06061315
G 06
06061335
paquetes recibidos: 2
paquetes perdidos: 3
G 06
crc correctos: 2
```

Ejemplo de cuando se pierden paquetes de la secuencia de 5 elementos.

```
06061315
G 06
06061325
G 06
06061335
G 06
06061345
G 06
06061355
paquetes recibidos: 5
paquetes perdidos: 0
G 06
crc correctos: 5
```

Ejemplo de recepción completa de la secuencia sin ningún fallo.

Como puede observarse en las imágenes de la terminal serial de arduino, el algoritmo detecta una pérdida y la notifica, también entrega algunas estadísticas de la comunicación.

Para la siguiente muestra de resultados de resultados, se mostrará cómo se comportó el algoritmo conectarse con el transmisor del grupo 3.

```
00060915
G 03
00060925
G 03
00060935
G 03
00060945
G 03
00060955
G 03
paquetes recibidos: 5
paquetes perdidos: 0
crc correctos: 0
```

Ejemplo de conexión correcta con el broadcast del grupo 3 y siendo recibido por el nuestro.

```
03000915
G 03
03000955
G 03
paquetes recibidos: 2
paquetes perdidos: 3
crc correctos: 0
```

Ejemplo de conexión con pérdidas.

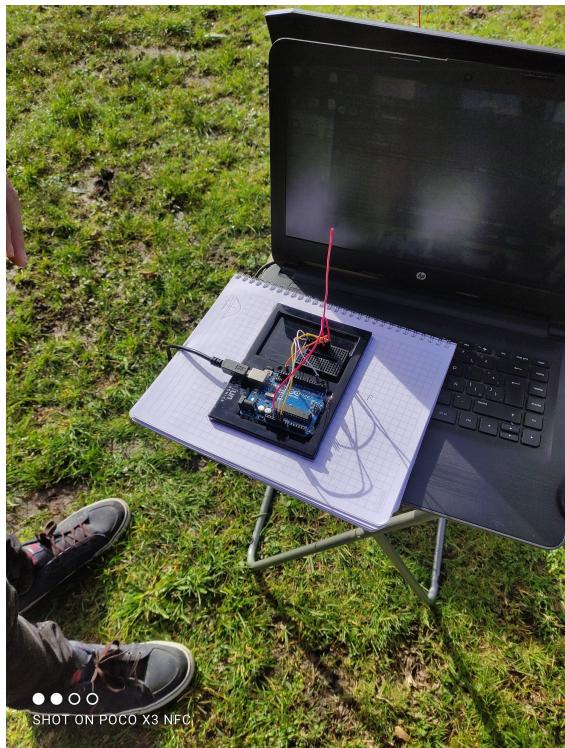
Algo a notar dentro de estos últimos ejemplos es que la estadística de “crc correctos” es igual a cero, esto se debe a que el grupo 3 y nuestro grupo 6 usan versiones distintas de la función crc, cada método tiene variaciones y de aquí nace el conflicto de que nuestro crc no puede comprobar si el mensaje se envió correctamente.

Discusión:

Nuestro estudio revela que la antena, en lugar de ser perfectamente circular como se podría esperar, tiene una forma similar a una estrella. Esto enfatiza la importancia de realizar mediciones directas en las antenas, en lugar de confiar únicamente en estimaciones teóricas. Al hacer esto, podemos determinar con más precisión su alcance real para así lograr experimentos correctamente. Lo cual confirma que la antena estaba modificada para no tener un lóbulo de radiación estándar. Ahora cuando realizamos las pruebas en sala solo nos pudimos comunicar solo con un grupo, el cual era el 3, pero con el resto no se logró, esto debido a que como se dijo en los resultados, nuestro código pasaba los datos como char en vez de uint_8, esto se pudo haber solucionado, haciendo un nuevo arreglo y que consiga procesar el buffer del mensaje, ya que si bien se podía leer el buffer, al estar usando arduino, que utiliza c++ como lenguaje de compilación, no se pueden cambiar los valores dentro de un array a otro tipo a diferencia de python, esto fue una mala práctica , por no haber buscado información,, y saber de antemano como se pasaba nuestro mensaje en nuestro código, para de esta manera poder haber realizado experimentos con otro grupo, con esto pudimos apreciar en la práctica el porqué se crearon ciertos protocolos a la hora de crear redes, para que de esta manera la conexión entre dispositivos sea cada vez mejor. Esto además es una mala implementación ya que esperar que un mensaje cambie los datos de un Array puede llevar a conexiones lentas y gastos en la memoria, al tener mensajes de largo fijo se aumenta la velocidad, algo muy importante por no decir esencial, en los sistemas de comunicaciones de hoy en día, por lo que mejorar nuestro programa le queda bastante por mejorar.

Como reflexión final podemos decir que como grupo y curso en general fallamos en el diálogo y el objetivo de crear un estándar para nuestras comunicaciones, esto puede deberse a varios factores tales como: falta de habilidades comunicativas y/o de liderazgo, como también el la ambigüedad de no saber si esto era copiar o no, esto generado por el ambiente evaluativo de la actividad. Sin embargo también sacamos cosas buenas de esta experiencia como los conocimientos adquiridos y la necesidad de mejorar nuestras fallas mencionadas anteriormente.

Anexo(Fotografías)



Izq - der : Diego, Leonardo



