TRABAJO PRÁCTICO INTEGRADOR

TRABAJO PRÁCTICO N° 3

COMUNICACIÓN DE DATOS

DOCENTES: MG. ING. MARTIN PICO, ING. MILTON POZZO
CHIABO FRANCO, KUHN LAUTARO, PALOMEQUE MATEO

Índice

Objetivos a cumplir

A lo largo de esta asignatura hemos recopilado diferentes conocimientos que nos ayudaran en la realización de este último trabajo. Continuando con la actividad anterior, debemos poder llevar a cabo una comunicación inalámbrica que conecte tres dispositivos, receptor, emisor y controlador. El receptor contará con un sensor infrarrojo el cual podrá captar las señales ópticas enviadas, codificarlas y controlar un led RGB. El emisor será el responsable de emitir, mediante un led infrarrojo, esas señales. Para poder realizar el envío de las señales y qué comunicará se necesita de un controlador que al presionar un botón o pulsador determine qué acción realizará la comunicación y que señal emitirá el emisor para que el receptor la pueda captar y realizar la operación.

El controlador será un dispositivo que se pueda comunicar vía bluetooth, por ende, nosotros usaremos un celular. Este contará con una aplicación, hecha por nosotros, que tendrá diferentes botones en la pantalla y que cada uno realizará una función tal como lo hacia el teclado de membrana del práctico anterior. El celular se conectará inalámbricamente al dispositivo emisor, que tendrá el módulo bluetooth y el led emisor infrarrojo. Con esta comunicación podremos decidir las señales que deseamos enviar desde una ubicación remota. También hay que aclarar que el emisor guardara el estado del led RGB que se encuentra en el receptor, esto con el fin de poder enviar esta información al celular e indicar cual es el color en el que se encuentra. Esto nos servirá por si el celular que esta controlando el dispositivo se desconecta y necesitamos continuar con la comunicación, al usar otro celular el emisor enviará el estado del led y podremos continuar desde donde se guardo por ultima vez. De esta forma no se genera una dependencia del controlador y toda la información se encontrará en el emisor.

Dispositivo Emisor

Como se explicó anteriormente, en esta instancia, se solicita que el dispositivo emisor pueda conectarse mediante bluetooth a otro dispositivo que actuara como controlador y será el encargado de proporcionar el color deseado al led RGB del dispositivo receptor.

Para poder entender el funcionamiento de forma más detallada, en primer lugar, se explicará el código implementado, detallando diferentes aspectos técnicos y luego su estructura física.

Bluetooth

En esta comunicación inalámbrica hemos decidido utilizar un módulo bluetooth llamada Serial HC-05. Antes de dar sus especificaciones, nos gustaría explicar ¿qué es bluetooth?

Bluetooth es una tecnología ad hoc, lo que significa que no se necesita una infraestructura fija y es sencilla de instalar y configurar. Es implementada para transmitir datos de forma inalámbrica, a través de distancias cortas. Es popular por la alta velocidad que ofrece, mientras consume muy poca energía. Sigue el estándar IEEE 802.15.1 para transferir datos a una distancia de hasta 10 metros.

El Bluetooth envía y recibe datos en una banda de frecuencia de 2,45 GHz, ofreciendo una velocidad máxima de datos de 721 kbps. Las velocidades de transmisión han mejorado con cada versión. Por ejemplo, Bluetooth 2.0 + EDR (Enhanced Data Rate) permite hasta 3 Mbps, mientras que Bluetooth 5.0 puede llegar hasta 50 Mbps en condiciones óptimas.

La arquitectura de red considera que los dispositivos bluetooth que estén dentro del límite de cobertura de otros dispositivos bluetooth pueden ser configurados logrando así formar redes ad hoc punto a punto o bien redes que establezcan conexiones punto a multipunto.

Los dispositivos Bluetooth deben emparejarse antes de comunicarse, lo que generalmente implica una autorización por parte del usuario. Esto otorga un cierto nivel de seguridad ya que impide la conexión de dispositivos o personas no deseadas. Los métodos de seguridad incluyen cifrado y autenticación.

Crea una Red de Área Personal (PAN) dentro de la cual la comunicación tiene lugar. Los dispositivos se conectan en una configuración maestro-esclavo, donde el dispositivo maestro es el que inicia el proceso de comunicación con otros dispositivos y también controla el proceso de comunicación.

El proceso de emparejamiento de Bluetooth se inicia cuando un dispositivo comienza a buscar otros dispositivos en su rango. Una vez que encuentra el dispositivo deseado, puede iniciar una solicitud para conectarse a ese dispositivo.

Un dispositivo esclavo no puede comunicarse sin el permiso de su dispositivo maestro y debe sincronizar sus tiempos y frecuencia de salto con la del dispositivo maestro. El tráfico dentro de la red Bluetooth es administrado y supervisado por el dispositivo maestro. Las conexiones entre un maestro y sus esclavos existen hasta que es desconectado por el usuario o los dispositivos viajan fuera del rango de cobertura.

Modulación Bluetooth

La modulación que emplea bluetooth es GFSK (Gaussian Frequency Shift Keying/ Modulación por Desplazamiento de Frecuencia con Filtrado Gaussiano) con un producto ancho de banda por tiempo BT= 0.5.

Este tipo de modulación permite un bajo coste y alcanza tasas de transmisión de 1Mbps. El índice de modulación debe estar entre 0.28 y 0.35. Un "1" binario se representa por una desviación positiva de frecuencia y un "0" binario como una desviación negativa. La desviación mínima no debe ser menor de 115 kHz.

En el dispositivo receptor bluetooth el nivel de sensibilidad es el aspecto más importante. Para lograr la medición de una tasa de error de bit, el dispositivo receptor envía de vuelta la información decodificada. Para una tasa de error o BER (Bit Error Rate) del 0.1% se define el nivel de sensibilidad de un receptor bluetooth mayor o igual a - 70dBm.

Módulo Bluetooth Serie HC-05

Ya en contexto de la tecnología a utilizar, el módulo Bluetooth HC-05 es quien nos facilita la comunicación a distancia. Este componente se puede utilizar en modo maestro o esclavo. El HC-05 tiene 6 pines, status, RXD, TXD, GND, VCC y EN. Los pines RXD y TDX se conectan a los pines digitales para realizar la comunicación con el Arduino, el pin VCC a 5 voltios y el pin GND a la tierra. Es muy usado en el para dar conectividad inalámbrica a través de una interfaz serial TTL entre Microcontroladores y otros dispositivos como PC, laptops o celulares Smartphone.

Figura 1. Módulo Bluetooth Serie HC-05

Las principales características que nos brinda este módulo son:

- Protocolo: Bluetooth especificación V2.0+EDR
- Protocolo comunicación: UART
- Tensión de comunicación: 3,3V
- Tensión de Alimentación: 5V
- Frecuencia: 2.4Ghz Banda ISM
- Velocidad de transmisión en baudios ajustable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- Configuración por Defecto: 9600 baud rate, N, 8,1, contraseña 1234
- Distancia Cobertura Bluetooth: Aproximadamente 10 metros
- Tamaño compacto: 4.3cm x 1.6cm x 0.7cm

Comunicación UART

Como se aclara en las características de HC-05, este componente utiliza un protocolo de comunicación UART. Un UART (Universal Asynchronous Receiver-Transmitter) es un tipo de circuito integrado que se usa para enviar y recibir datos a través de un puerto serie en un equipo o dispositivo periférico. Los UART son ampliamente utilizados y conocidos por su sencillez. Sin embargo, a diferencia de SPI e I2C, los UART no admiten múltiples dispositivos subordinados.

La comunicación UART no requiere una señal de reloj compartida. En su lugar, utiliza bits de inicio y parada para definir los límites de cada paquete de datos. Los datos se transmiten bit a bit. Un paquete típico incluye un bit de inicio que indica el comienzo de un paquete, los bits de datos que generalmente son entre 5 y 9 bits por paquete, un bit de paridad (opcional) para detección de errores y por último un bit de parada que pueden ser uno o dos bits que indican el final del paquete. La velocidad de transmisión, medida en baudios (bits por segundo), debe ser la misma para el transmisor y el receptor. Los UARTs suelen permitir comunicación Full Duplex, lo que significa que pueden enviar y recibir datos simultáneamente a través de dos líneas diferentes.

Hardware Emisor

En la elaboración del circuito a comunicar se utilizó un Arduino Mega para el receptor, el cual esta conectado al Led RGB con sus respectivas resistencias y el sensor receptor de infrarrojo. No se explicará con detalles las conexiones y funcionamiento del dispositivo 1 debido a las reiteradas utilizaciones en prácticos anterior, esto se debe a que no hay cambios significativos ni en el hardware como en el código del programa y retomarlo solo sería repetitivo y poco provechoso.

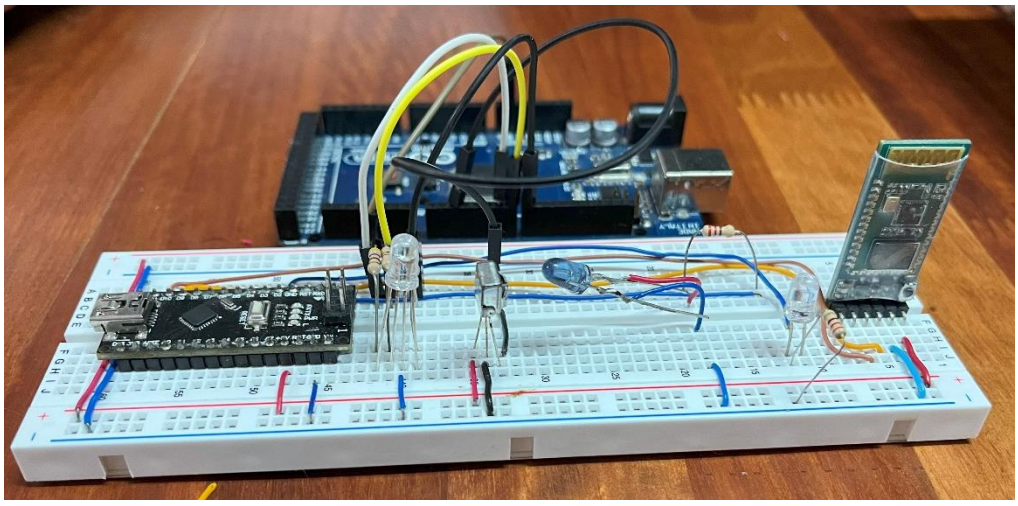

Figura 2. Dispositivo Receptor y Emisor, vista lateral.

El dispositivo 2 es un Arduino Nano que en sus pines 10 y 11 esta conectado el módulo bluetooth, el pin 10 para RXD y el 11 para TXD, cabe aclarar que se lo conecta

de forma directa a los 5V de la placa y también es esencial el GND a masa. En el pin 2 se conecta un led con una resistencia de 220Ω que se utilizará para notificar acerca del estado en que se encuentra la conexión entre el dispositivo 2 y 3, encendido para conexión y apagado para desconexión. Y por último en el pin 9 dedicado para el Led emisor infrarrojo que esta conectado con una resistencia de 220Ω.

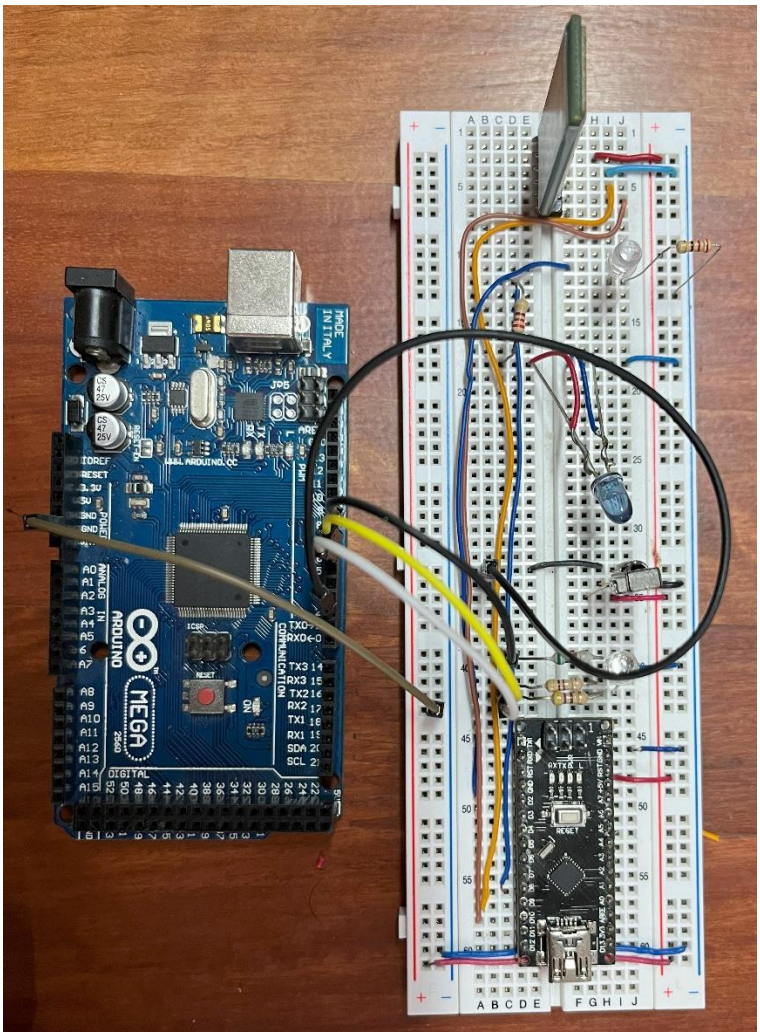

Figura 3. Dispositivo Receptor y Emisor, vista superior.

La complejidad electrónica no es alta, pero es necesaria su correcta elaboración para evitar errores y no producir un daño en el sistema, en los componentes o en las placas de Arduino.

Código del Programa Emisor

Para la programación del dispositivo 2, el Arduino Nano, se implementó diferentes librerías y funciones que se complementan con la aplicación utilizada, se profundizará en cada parte a continuación.

Al inicio del código incluimos 3 librerías, *IRremote*, *EEPROM* y *SoftwareSerial*. La librería *IRremote* es quien nos permitirá poder captar y recrear las señales infrarrojas

de la comunicación, sirve para configurar el sensor infrarrojo del dispositivo 1 para la captación y muestra de las señales, y para el envió de señales ópticas por el led infrarrojo del dispositivo 2. En este último caso, nosotros enviaremos las señales mediante el protocolo NEC, de igual forma, esta librería de código abierto también nos permite trabajar con muchos otros tales como SONY, Panasonic, Philips, etc.

La librería *EEPROM* que, antes de hablar de esta, necesitamos saber ¿qué es EEPROM? El microcontrolador de la placa de Arduino tiene una memoria EEPROM, una memoria que permite almacenar variables de forma permanente. Incluso cuando se reinicia o se apaga Arduino, mantendrá sus valores. Es como si fuera el disco duro de tu computadora, pero con una capacidad mucho más pequeña.

Las memorias se dividen en posiciones o huecos y cada una puede almacenar un byte lo que quiere decir que cada posición puede almacenar un número de 8-bit (un número de 0 a 255). La capacidad de la memoria EEPROM depende del microcontrolador, en nuestro caso, el Arduino Nano dispone de un ATmega328 por lo que tiene una capacidad de 1024 bytes. La memoria EEPROM tiene una vida limitada. En Arduino su vida es de 100.000 ciclos de escritura por cada posición de memoria, es decir, la posición 1 solo podrás escribir o actualizar un valor 100.000 veces.

La memoria EEPROM está dividida en huecos. Cada hueco representa un byte. Es la unidad mínima que puede almacenar cada hueco. Recuerda que 1 byte es un número entre 0 y 255.

En el caso de Arduino Nano, el microcontrolador ATmega328 tiene 1024 bytes, es decir, 1024 posiciones de memoria. Para guardar un dato tenemos que indicar la posición de la memoria y el dato. ¿qué pasa si quieres almacenar un número mayor que un byte o una cadena de texto? Ocupará más de una posición de memoria. Un tipo de dato *int* ocupa 2 bytes y por lo tanto ocupará dos huecos, un tipo *float* ocupa 4 bytes y una cadena de texto depende de los caracteres, cada letra ocupa un byte.

La memoria EEPROM cuenta con una librería con la que puedes leer, escribir o actualizar un valor de la memoria EEPROM muy fácilmente. Esta librería está preparada para leer byte a byte o tipos de datos más complejos como *int* o *float.*

Por último, la librería *SoftwareSerial* que ha sido desarrollada para permitir la comunicación serie en otros pines digitales del Arduino, más allá de los pines 0 y 1, usando el software para replicar la funcionalidad. Es posible tener múltiples puertos serie de programas con velocidades de hasta 115200 bps. El soporte nativo serie pasa a través de un componente de hardware (integrado en el chip) llamado UART. Este

hardware permite el chip ATmega recibir la comunicación serie incluso mientras trabaja en otras tareas, siempre que haya espacio en la memoria intermedia serie de 64 bytes.

```
#include <IRremote.h>
#include <EEPROM.h>
#include <SoftwareSerial.h>
```

Figura 4. Librerías incluidas.

A continuación de las bibliotecas utilizamos la función ***IRsend*** que proporciona métodos para enviar diferentes tipos de señales infrarrojas utilizando varios protocolos estándar. Estos métodos permiten modular la señal infrarroja para que sea compatible con los dispositivos que se desean controlar.

```
IRsend irsend;
```

Figura 5. Función IRsend.

Luego se define los pines de comunicación para el módulo bluetooth HC-05. Para poder utilizar los pines digitales 10 y 11, hacemos uso de la función ***SoftwareSerial*** propia de la librería del mismo nombre, esto con el fin de poder convertir estas entradas en RX y TX tal como se explicó con anterioridad.

```
int BT_Rx = 10;
int BT_Tx = 11;

SoftwareSerial BT(BT_Rx, BT_Tx);
```

Figura 6. Pines módulo bluetooth y conversión a comunicación serie.

Se definen el pin 2 para el Led que indicará la conexión y desconexión de la comunicación, el pin 9 para el Led emisor infrarrojo y el pin 3 que leerá la salida del pin de estado del módulo HC-05. Este último nos indicará si la conexión bluetooth se estableció o si el dispositivo se desconectó. También se definen otras variables que tendrán un uso futuro y que no son relevantes de especificar en este momento.

En el *void setup ()* empezamos definiendo el pin del Led que indicará el estado de la comunicación como salida y como entrada será la señal enviada por el pin de estado del módulo bluetooth. También implementamos la función *irsend.begin()* que como parámetro usamos el nombre de la variable que definimos para el pin del Led emisor. Junto a esto, se añade la función *Serial.begin()* y *BT.begin()*, ambas con 9600 baudios como parámetro. Esto sirve para poder indicar la velocidad de transmisión de datos en la comunicación y poder sincronizar los puertos serie.

```
void setup() {
  pinMode(LED_con, OUTPUT);          //defino el pin como salida
  pinMode(control_estado, INPUT);    //defino el pin como entrada

  irsend.begin(IR_emitter);    //funcion de la libreria IRremote para definir el pin donde se conecta el emisor
  Serial.begin(9600);          //seteo la velocidad de comunicacion serial con la PC en 9600
  BT.begin(9600);              //seteo la velocidad de comunicacion con el modulo BT en 9600
}
```

Figura 7. Pines y velocidad de transmisión definidos.

En void loop () solo añadiremos las funciones requerirá el programa. Estas serán definidas posteriormente y fuera del bucle principal, solo las llamaremos dentro de él.

```
void loop() {
  arranque();                 //funcion que le la memoria eeprom

  control_conexion();         //funcion deteccion del pin de estado del modulo BT

  lectura();                  //funcion encargada de recibir variables tipo texto desde el modulo, a su vez tambien
  //transforma esos datos en variables numericas aptas para su trabajo

  impresionControl();        //funcion que manda datos por el serial de la PC como forma de control

  envio();                    //funcion que envia los datos por infrarojo

  enviarValoresRGB();         //envia datos al modulo BT para retroalimentacion a la aplicacion del celular
}
```

Figura 8. Llamada a funciones en bucle principal.

La primera función llamada **arranque()** es utilizada para poder ejecutarse, mediante una variables incremental, solo una vez en el programa. Esto lo hace, como bien lo indica su nombre, al inicio de la comunicación. Tiene como objetivo leer los valores de los colores que se guardaron en la memoria EEPROM para así enviarlos mediante el led infrarrojo y mostrar el color del led RGB al encender la comunicación.

```
void arranque() { //funcion que lee los datos
  if (x == 0) {
    registroR = EEPROM.read(0);
    registroG = EEPROM.read(1);
    registroB = EEPROM.read(2);
    x = x + 1;
  }
}
```

Figura 9. Función encargada de enviar los colores almacenados al comienzo de la comunicación.

La función control_conexion() debe encargarse de comprobar la conexión establecida, o no, entre el módulo bluetooth y el dispositivo controlador. Se emplea una variable que guardará la lectura del pin de estado del HC-05, en caso de obtener un “1” en la lectura se enviará una señal HIGH para encender el led encargado de notificarnos el estado de la comunicación, en caso de recibir un “0” la señal será LOW para apagar dicho led.

```
void control_conexion() {
  disp_estado = digitalRead(control_estado);
  if (disp_estado == 1) {
    digitalWrite(LED_con, HIGH);
  }
  if (disp_estado == 0) {
    digitalWrite(LED_con, LOW);
  }
  Serial.print("Estado modulo");
  Serial.println(disp_estado);
}
```

Figura 10. Función que determina el estado de conexión establecida con el controlador.

La función **lectura()** es utilizada para poder leer los datos del módulo bluetooth que esta conectado a nuestro Arduino. La clase *BT.available()* devuelve el número de bytes disponibles para leer desde el buffer serial y en cuanto la condición se cumpla, se añade una de las variables definidas previamente, esta se encargará de guardar el valor obtenido mediante la clase *BT.read()* que lee un byte del puerto serial.

Los valores que se pueden obtener provienen de la aplicación utilizada desde el dispositivo controlador. Estos serán diferentes y cada uno habilitará un condicionar que ejecute una acción dada. En caso de leer el valor “R” se le asignará a una función, previamente definida, el valor “1”. Luego se hace una llamada a otra función llamada **obtencionEntero()**, que se detallará más adelante, pero a grandes rasgos es la encargada de convertir el formato del valor del color del led RGB a uno en el que podamos trabajar. Esto lo hará también para los valores “G” y “B” que son los asignados para los colores del led RGB.

Al obtener el valor “S”, se nos indica la acción de guardar el color del led RGB. Para esto se cambia el valor de una variable booleana para poder ejecutar la función **guardado()** que es llamada a continuación, esta función también se explicará con más detalles posteriormente.

El valor “P” es quien determina la señal de encendido del led RGB. Esto se lleva a cabo de forma similar que la función **arranque()**, busca y lee los valores almacenados en la memoria EEPROM y los almacena en las variables encargadas de contener los valores de los colores del led RGB.

Al leer el valor “O”, el programa sabrá que el controlar nos da la señal de apagar el led RGB. A las variables que almacenan los valores de los colores RGB se les establece guardar el valor 0, que al estar todas en esta condición, el led contará con una intensidad nula la cual se podrá presenciar como apagado.

Por último el valor “E” se encargará de poner una variable predefinida en un valor determinado que permitirá la ejecución de la función **envioValoresRGB()**, función que se encargará de notificarle a la aplicación el color actual de led RGB. Este valor se enviará cuando la conexión con la aplicación es exitosa. Esta parte cobrará más sentido en el momento de desarrollar el funcionamiento de la aplicación.

```
void lectura() {
  if (BT.available()) { //Si el puerto serie (Bluetooth) está disponible
    valor = BT.read();  //Lee el dato entrante via Bluetooth
    Serial.println(valor);

    if (valor == 'R') { //Si el dato entrante es una B
      g = 1;                //en funcion del valor de g se guarda en un registro diferente de color
      obtencionEntero();    //Llama la función que controla el valor a guradar el valor numerico de cada color
    }

    if (valor == 'G') {  //Si el dato entrante es una B
      g = 2;
      obtencionEntero();    //Llama la función que controla el valor a guradar el valor numerico de cada color
    }

    if (valor == 'B') {  //Si el dato entrante es una B
      g = 3;
      obtencionEntero();    //Llama la función que controla el valor a guradar el valor numerico de cada color
    }

    if (valor == 'S') {  //Si el dato entrante es una S
      flag = true;
      guardado();        //se llama a la funcion de guardar valores en la eeprom
      Serial.println("guardado");
    }

    if (valor == 'P') {  //Si el dato entrante es una P
      registroR = EEPROM.read(0);  //lee los valores guradados en la eeprom y prende el led con esos valores
      registroG = EEPROM.read(1);
      registroB = EEPROM.read(2);
    }

     if (valor == 'O') {  //Si el dato entrante es una O
       registroR = 0;     //los registros se ponen en 0 ya que es la señal de apagado
       registroG = 0;
       registroB = 0;
       Serial.println("Apagado");
     }

     if (valor == 'E') {  //Si el dato entrante es una E
       t = 0;             //pone t en 0, cuando se recibe una E indica que la concecion de la app android fud exitosa
       //al ponerse t en 0 la funcion de enviarValoresRGB(); se podra activar
     }
   }
}
```

Figura 11. Función de lectura.

Como se mencionó anteriormente, dentro de la función **lectura()**, se hace llamado de la función **obtencionEntero()**, esta inicia con un *delay* de 300ms con el objetivo de tomarse un tiempo para que se puedan recibir todos los valores, una vez transcurrido dicho intervalo, dentro de un condicional *while* que determine que mientras el bluetooth esté disponible, se almacenan los datos leídos en una variable tipo carácter que luego se utiliza para crear una lista. Después se verifica mediante la función *.length()* que la cadena cuenta con al menos un dato, y dependiendo del valor almacenado en la variable “g” que se empleó en la función lectura(), se guardará en las variables que contienen los colores del led RGB mediante la conversión de la cadena a un valor entero. Se utiliza la función *.toInt()* que convierte los datos tipo carácter a entero, esto se debe a que nosotros trabajamos con este tipo para las intensidades de los colores. Al finalizar la variable “g” se le asigna el valor 0 para “limpiarla” y permitir su uso para futuros nuevos datos.

```
void obtencionEntero() {
  delay(30);
  while (BT.available()) {
    char c = BT.read();   //Lee el dato entrante y lo almacena en una variable tipo char
    estado += c;      //Crea una cadena tipo String con los datos entrates
  }

  if (estado.length() > 0) { //Se verifica que la cadena tipo String tenga un largo mayor a cero

    if (g == 1) {                  //dependiendo del valor de g
      registroR = estado.toInt();  //Guarda en un registro el dato en forma de entero (int)
    }

    if (g == 2) {
      registroG = estado.toInt();  //Guarda en un registro el dato en forma de entero (int)
    }

    if (g == 3) {
      registroB = estado.toInt();  //Guarda en un registro el dato en forma de entero (int)
    }

    g = 0;             //vuelve la variable g a 0
    estado = "";       //Limpia la variable para poder leer posteriormente nuevos datos
  }
}
```

Figura 12. Función para convertir datos tipo carácter a enteros.

La función **guardado()** no es incluida en el bucle principal pero si pertenece a la composición dentro de otras funciones por lo que de interés explicarla antes. Esta función tiene como objetivo poder almacenar los valores que conforman el color del Led RGB en la memoria EEPROM. Utiliza un condicional con una variable booleana para que cuando esta esta en true, inicializada en false, escriba dentro de un especifico hueco de la memoria EEPROM los valores que contienen las variables, inicializadas anteriormente, usadas de parámetros. Para esta acción se usa la clase ***EEPROM.write(,)***. Luego la variable booleana vuelve a false para evitar que se guarden repetidamente los datos.

```
void guardado() {
  if (flag == true) {
    EEPROM.write(0, registroR);
    EEPROM.write(1, registroG);
    EEPROM.write(2, registroB);
    flag = false;
  }
}
```

Figura 13. Función para guardar los valores de los colores del Led RGB.

impresionControl() es una función opcional y que no influye directamente con el envío o recepción de los datos. Se utiliza para poder controlar, mediante el puerto serie, que los datos enviados sean los correctos y poder detectar algún posible error en la transmisión. Al comprobar la eficiencia del programa y de la comunicación, si uno lo desea, puede eliminar esta función sin generar ningún cambio significativo.

```
void impresionControl() {
  Serial.print("rojo");
  Serial.print(registroR);
  Serial.print(" verde");
  Serial.print(registroG);
  Serial.print("  azul");
  Serial.println(registroB);
}
```

Figura 14. Función para el control de envío de datos.

La función **envio()** que es la responsable de enviar los datos como señales infrarrojas por el led emisor al dispositivo 1 que capturará y realizará la acción en el led RGB tal como nosotros lo establezcamos. Dentro de la misma se encontrará la clase *irsend.sendNEC()* que se utiliza para enviar una señal infrarroja con un protocolo en específico, en nuestro caso NEC. Se le pasará como parámetro el comando que interpretará el sensor receptor, la variable donde se encuentra el valor del color del led RGB y la cantidad de repeticiones del comando. Esto se repite tres veces, una por color y se envían todas en conjunto, de esta forma se forma el color final del Led RGB.

```
void envio() {
  // irsend.sendNEC(0x64, 0x64, 1);
  irsend.sendNEC(32, registroR, 1);

  // irsend.sendNEC(0x64, 0x64, 1);
  irsend.sendNEC(33, registroG, 1);

  // irsend.sendNEC(0x64, 0x64, 1);
  irsend.sendNEC(34, registroB, 1);
}
```

Figura 15. Función para enviar los valores de los colores del Led RGB.

Por último, la función enviarValoresRGB(), mencionada previamente, se encarga de enviar los valores de los colores del led RGB a la aplicación del dispositivo controlador conectado vía bluetooth y presenciar los datos en tiempo real en ambas partes. Para esto la aplicación envía el dato “E”, tal como se explicó en la función de lectura, y al contener la variable con el mismo valor en ambas funciones, esta puede ejecutar la acción del envío de datos. La aplicación solo recibe valores en tipo texto, por eso, nosotros debemos enviar los valores de las intensidades de los colores separadas mediante un carácter a elección, que en nuestro caso es “|”, esto permite a la aplicación poder detectar este carácter y dividir los tres valores del RGB. Al final de esta cadena se añade el envío del valor “\n” que define el final de línea y es muy importante para que el receptor sepa el final del paquete recibido. Para finalizar se incrementa el valor de la variable inicial para que solo se actualice el color en pantalla cuando esta sea 0.

```
void enviarValoresRGB() {
  // Enviar los valores RGB actuales a la aplicación a tr
  // Se ejecuta cada vez que se recibe el caracter que in
  if ( t == 0) {                    //La transmicion se
    BT.print(registroR);            // | para delimitar
    BT.print("|");                  //eso no ayudara par
    BT.print(registroG);
    BT.print("|");
    BT.print(registroB);
    BT.print("\n"); // Fin de línea. Importante.
    t = t + 1;
  }
  /*
```

Figura 16. Función encargada de actualizar el color del led RGB en la aplicación.

Aplicación Móvil

Acabamos de explicar el funcionamiento del dispositivo 2 y como funciona en relación al receptor y el controlador, pero por el momento no se dio a conocer la apariencia ni funcionamiento interno de la aplicación que será la encargada mediar entre el usuario y toda la comunicación. Esta intervendrá desde la conexión al módulo bluetooth hasta definir el color del Led RGB a preferencia.

El controlador será una aplicación instalada en un celular mediante bluetooth, cabe aclarar que es un APK y no una aplicación descargable en tiendas virtuales. Esto se debe a que es desarrollada por nosotros usando una página web llamada “MIT App Inventor”. Esta nos permite diseñar una aplicación mediante la composición de diferentes componentes, tales como botones, cuadros de texto, imágenes y sliders. En la parte visual se pueden ubicar y dar formato en base a nuestro gusto sin ningún tipo de dificultad y para el funcionamiento o la lógica de la aplicación se emplea una programación en bloques que lo hace intuitivo y fácil de comprender.

Figura 17. Pantalla principal de la aplicación móvil.

Con lo primero con lo que nos encontraremos es con un botón con una leyenda que dice “SELECCIONAR BT”, al presionarlo se espera que se despliegue un listado de dispositivos bluetooth cerca de nuestra área. Al seleccionar el adecuado, módulo HC-05, se establece la conexión y en la zona debajo, el cuadro que se encontraba en rojo y

decía “DESCONECTADO” pasará a ser verde y con el mensaje de “CONECTADO”. Tal como se comento en el programa de Arduino, la aplicación envía el valor “E”, tal como se puede apreciar en la programación en bloque, para poder indicarle al emisor que le envíe los valores de color del led RGB.

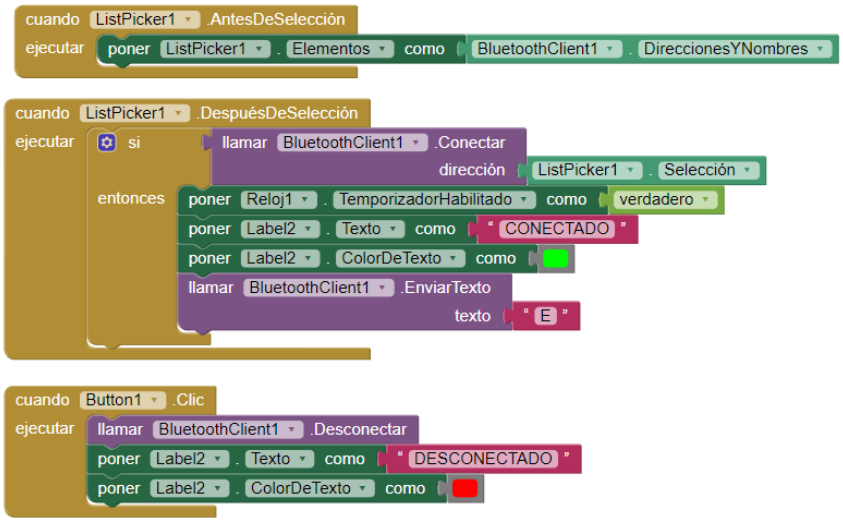

Figura 18. Programación en bloque para conexión y desconexión al módulo bluetooth.

Luego en la parte del centro de la pantalla se pueden ver tres sliders, uno para cada color del led RGB, rojo, verde y azul. Estos serán los encargados de definir el valor del color final con su combinación. En cada bloque de color se envía su respectivo valor al emisor para así poder actualizar la combinación del color final que tendrá el led RGB. El cuadro contenedor de dicha combinación se encontrará debajo de las sliders y cambiará cada vez que se detecte un incremento o disminución en la intensidad de cualquier color.

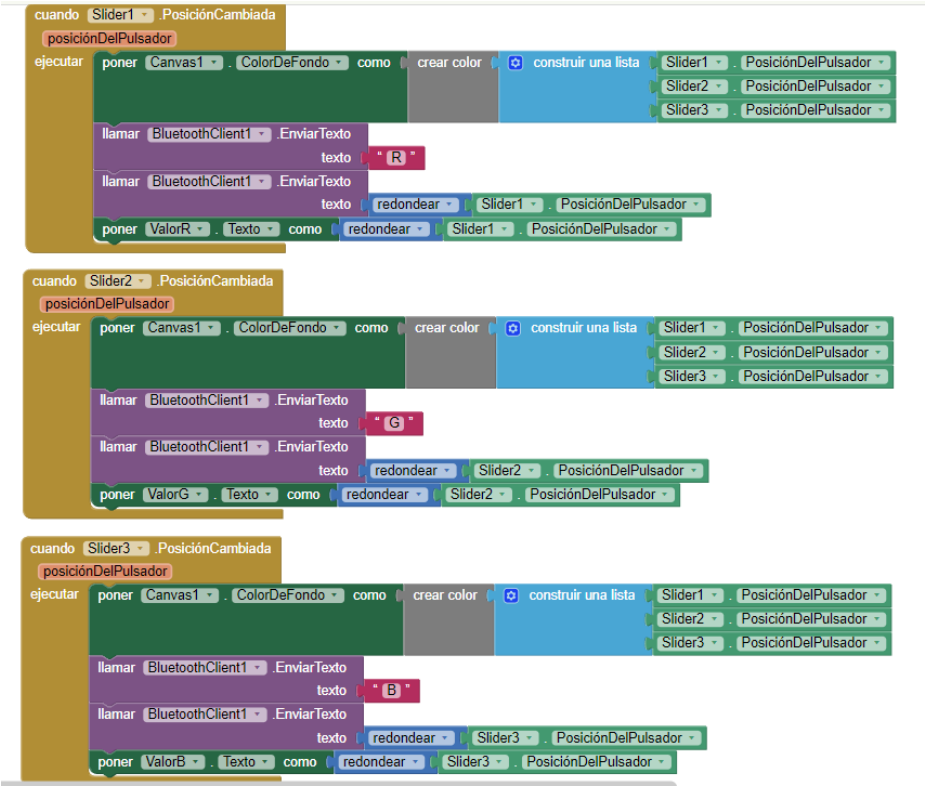

Figura 19. Programación en bloque para sliders que definen los colores del Led RGB.

Luego están los botones que definirán las principales acciones de la comunicación, el primero dice “Prender” y bien como se intuye su escrito, cumple con la función de encender el led RGB, el segundo denominado “Guardar” realiza la acción de almacenar los valores actuales del led RGB en la memoria EEPROM y, por último, el tercero “Apagar” es quien, valga la redundancia, apaga el RGB. Estos pulsadores enviarán un valor característico que le notificará al emisor que acción realizar a la hora de enviar los datos.

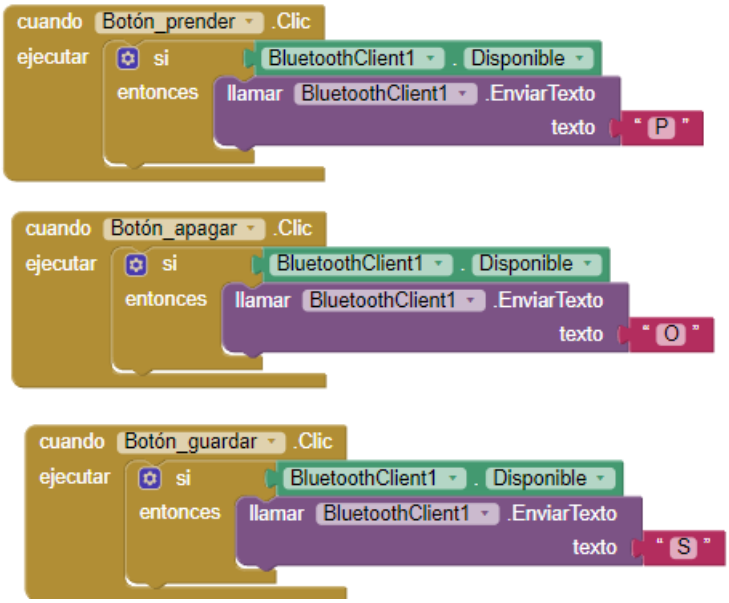

Figura 20. Programación en bloque para los botones de “Prender”, “Guardar” y “Apagar”.

Para el funcionamiento general de la aplicación se realiza una combinación de bloques dentro de un temporizador, el cual se encarga muestrear y sincronizar el dispositivo controlador con el emisor en la comunicación. Este conjunto tiene como finalidad la recepción de los valores de intensidad de los colores para poder almacenarlos en variables globales predefinidas, así formar el color del led RGB, mostrarlo en pantalla y actualizar la posición de los sliders. Para esto utilizamos el bloque “recortar texto en” que nos permite dividir la cadena de valores mediante un carácter predefinido en la programación de Arduino, el cual es “|” como se mencionó antes. La aplicación recibe texto, lo divide en valores según un carácter específico y lo almacena en diferentes variables para cada color. También, al final de los bloques, en caso de que el bluetooth no este conectado, se envía el valor “H”, esto se hace para poder detectar la desconexión de la comunicación causada por otros medias que no sean mediante el botón “DESCONECTAR” que esta al final de la pantalla.

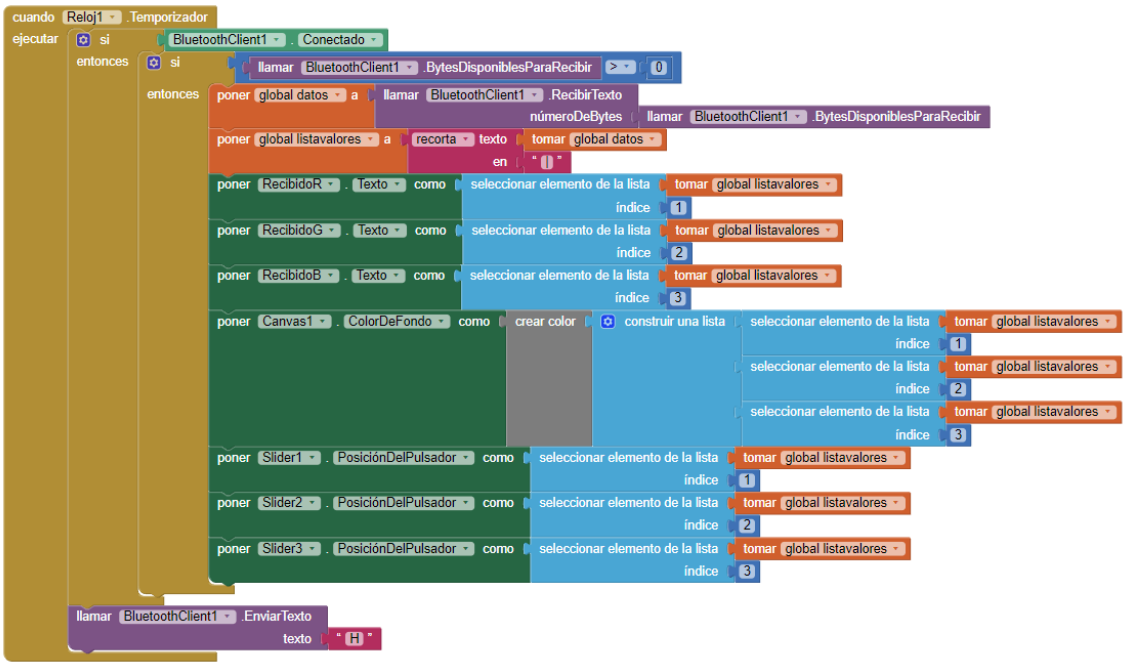

Figura 21. Conjuntos de bloques para recepción de los colores y presentación en pantalla de la combinación final.

Para finalizar, se emplea un bloque capaz de detectar e identificar diferentes erros que se producen en la comunicación. Cuando se produce el error 516, la aplicación actualiza la etiqueta que muestra el estado actual entre el emisor y el controlador a “DESCONECTADO” en color rojo. Este error se produce cuando la comunicación es interrumpida de forma externa por falta de alimentación, desconexión por distanciamiento de los dispositivos, alguna desconexión automática de una de las dos partes, etc. Por esto se envía el valor “H” anteriormente, esto se debe a que, al no estar definido en el programa del emisor, no se puede interpretar y ejecuta este error.

Figura 22. Conjuntos de bloques para detección de error 516.

Conclusiones y Agradecimientos

A lo largo del informe hemos indagado sobre la realización y funcionamiento de la comunicación solicitada, también se detallaron aspectos técnicos sobre él envió de los datos, características de los componentes, protocolo de comunicación, entre otros aspectos. Pero también, hemos recorrido a lo largo de esta materia diversos proyectos que nos brindaron las herramientas y conocimientos para culminar con este práctico final. Tratamos de ser lo más claros y breves en este informe ya que la parte fuerte de este proyecto está en el programa y elaboración del hardware. Queremos cerrar el escrito con un agradecimiento a los docentes de la cátedra, Mg. Ing. Martín Picó e Ing. Milton Pozzo, por su colaboración y asesoría en este práctico como en toda la materia a lo largo del cuatrimestre.

Bibliografía

M&V TECHNOLOGIE EC. (30 de octubre de 2023). Clase 13: Control de LED RGB con App Inventor y Arduino + Bluetooth con deslizadores por cada color. https://www.youtube.com/watch?v=OE6ITRHaz1g

Editronikx. (13 de junio de 2016). visualizacion de multiples sensores con arduino y App inventor (primera revision). https://www.youtube.com/watch?v=Jjha1pyXrZc

Aprendamos Ingeniería. (25 de mayo de 2020). Configurar módulo Bluetooth a Arduino | Módulo HC-05, HC-06 | Comandos AT. https://www.youtube.com/watch?v=YRWitbRWevY&t=1s

Javier Arévalo. (16 de mayo de 2021). Controlar LED RGB desde el Celular por Bluetooth || Módulo HC-05, APP Inventor || Proyectos Arduino. https://www.youtube.com/watch?v=3r3yBe6PoBM&t=1s

ELECTRONOOBS en Español. (23 de julio 2022). Memoria EEPROM Arduino - Guarda Cualquier Cosa. https://www.youtube.com/watch?v=PUE4-jwesWs&t=696s

Solectro. (22 de julio de 2020). Una guía detallada para usar el Bluetooth en el ecosistema de Arduino. https://solectroshop.com/es/blog/una-guia-detallada-para-usar-el-bluetooth-en-el-ecosistema-de-arduino-n36

MCI Electronics. Módulo Bluetooth Serial HC05. https://mcielectronics.cl/shop/product/modulo-bluetooth-serial-hc05-26967/

Diana Franco y Francisco Castillo. (01 de septiembre de 2009). Revista UTP. Comunicaciones Inalámbricas Bluetooth. https://revistas.utp.ac.pa/index.php/prisma/article/view/412/html#:~:text=La%20modulaci%C3%B3n%20que%20emplea%20bluetooth,tasas%20de%20transmisi%C3%B3n%20de%201Mbps.

Luis del Valle. (23 de noviembre de 2022). Programarfacil. Memoria EEPROM de Arduino. https://programarfacil.com/blog/arduino-blog/eeprom-arduino/

Manuel Delgado Crespo. (01 de marzo de 2016). Arduino y su documentación en español. https://manueldelgadocrespo.blogspot.com/p/biblioteca.html

Juan A. Villalpando. (01 de agosto de 2015). Tutorial de iniciación de App Inventor 2 en español. http://kio4.com/appinventor/9A5_bluetooth_fallo_conexion.htm

Repositorio GitHub: https://github.com/equipoPI/ComunicacionDatosTP3