Universidad Intercontinental de la Empresa

# Tecnologías de Automatización y Robotización para Empresas

# IoT

## Placa Arduino Uno

# Práctica de Laboratorio 06

## Comunicaciones

## Arduino

**Profesor Eladio Dapena Gonzalez**

**Técnico Pablo Tarrío**

**Curso 2024-2025**

Universidad Intercontinental de la Empresa

# Práctica 07: Comunicaciones

## 1 Introducción

Durante la actividad de laboratorio se experimenta con el funcionamiento y gestión de las comunicaciones con Arduino.

En esta actividad se realizan la configuración y uso de tareas básicas con comunicaciones serie y Bluetooth; para lo cual es necesario el uso de diversos componentes vistos en prácticas anteriores.

## 2 Objetivos

a) Establecer comunicación Serial entre dos Arduino **e intercambiar datos entre ellos.**
b) Configurar y probar el módulo de comunicación Bluetooth BLE HM10.

## 3 Materiales Requeridos

- 2 Arduino UNO.
- 1 Módulos Bluetooth BLE HM10
- 2 Protoboard.
- 4 Leds
- 2 pulsadores
- 1 motor DC
- 1 Resistencia 220 Ω
- 2 Resistencias 2K Ω
- 1 Resistencia 1 KΩ

# 4 Marco teórico

## 4.1 Comunicación entre dispositivos

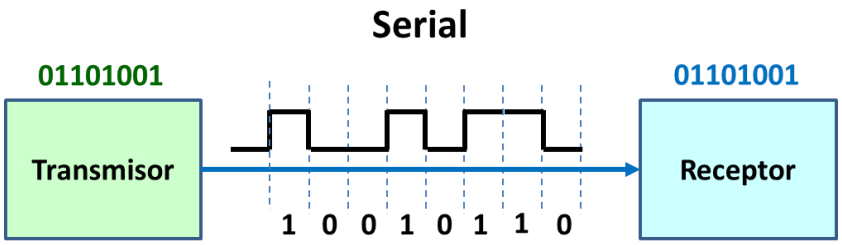
Es el proceso mediante el cual dos o más dispositivos intercambian información o datos entre sí. Este intercambio puede ser de diferentes formas y bajo distintos estándares, y es fundamental para que los dispositivos se sincronicen, envíen instrucciones, y trabajen en conjunto en sistemas electrónicos más complejos.

### 4.1.1 Comunicación Serial.

Los datos se transmiten bit a bit a través de un solo canal, ya sea en una o ambas direcciones.

Ejemplos:

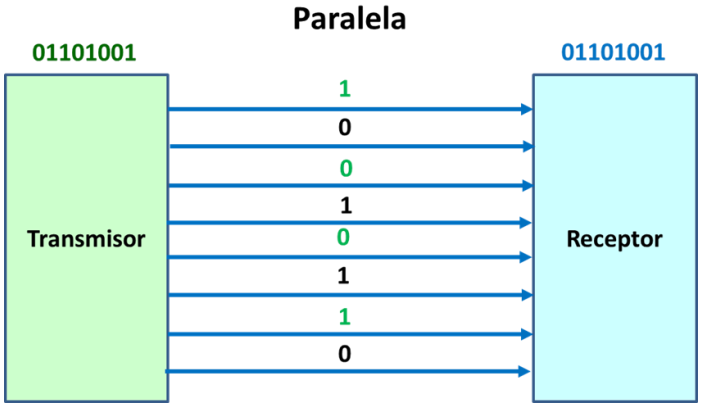
- **UART** (*Universal Asynchronous Receiver-Transmitter*): Comunicación asíncrona sin reloj compartido.
- SPI (*Serial Peripheral Interface*): Protocolo síncrono donde se utiliza un reloj compartido.
- **I²C** (*Inter-Integrated Circuit*): Comunicación síncrona que permite conectar múltiples dispositivos en un solo bus.




### 4.1.2 Comunicación Paralela.

Los datos se transmiten simultáneamente a través de varios canales (varios bits al mismo tiempo).

Ejemplo: Conexiones entre un microprocesador y su memoria (buses de datos y direcciones).



### 4.1.3 Comunicación Síncrona.

Los dispositivos se sincronizan utilizando una señal de reloj compartida.

El transmisor y el receptor deben estar coordinados a través de este reloj.

Ejemplos: SPI, $I^2C$, CAN Bus.

### 4.1.4 Comunicación Asíncrona.

No hay una señal de reloj compartida; los dispositivos deben acordar la velocidad de transmisión (*baud rate*).

Se usa en comunicaciones donde no es necesario sincronizar los datos en tiempo real.

Ejemplo: UART.

### 4.1.5 Comunicación Simplex, Half-Duplex y Full-Duplex.

- **Simplex:** La transmisión de datos se realiza en una sola dirección (de un transmisor a un receptor).
- **Half-Duplex**: Los datos pueden ir en ambas direcciones, pero no al mismo tiempo (un dispositivo transmite mientras el otro recibe).
- **Full-Duplex:** Los datos se transmiten y reciben simultáneamente en ambas direcciones.

## 4.2 Puerto UART (*Universal Asynchronous Receiver-Transmitter*)[1]

El puerto **UART** es un hardware utilizado para la comunicación ***serial asíncrona*** entre dispositivos electrónicos.

Su función principal es **convertir datos** en **formato paralelo** (que se utiliza dentro de un procesador o microcontrolador) a **formato serial** (que se usa para la transmisión de datos a través de cables).

Los puertos UART son comúnmente utilizados para la comunicación entre un ordenador y un microcontrolador (como Arduino), o entre microcontroladores y sensores avanzados.

**Características.**

1. **Asíncrono:** No necesita de una señal de reloj para sincronizar la transmisión y recepción de datos, lo que significa que los dispositivos deben acordar la velocidad de transmisión (baud rate) de antemano.
2. **Transmisión de bits en serie:** Los datos se transmiten bit a bit en un único canal, comenzando por un bit de inicio, seguido por los bits de datos, y terminando con un bit de parada (y a veces un bit de paridad).
3. **Estructura de los datos:**
   - **Bit de inicio:** Indica el comienzo de la transmisión.
   - **Bits de datos:** Generalmente 7 u 8 bits de datos que contienen la información útil.

[1] https://arduino.cl/como-configurar-la-comunicacion-uart-en-arduino/

- **Bit de paridad (opcional):** Sirve para verificar la integridad de los datos.
- **Bit de parada:** Marca el final de la transmisión.

4. **Velocidad:** La comunicación UART se mide en baud rate (bits por segundo).

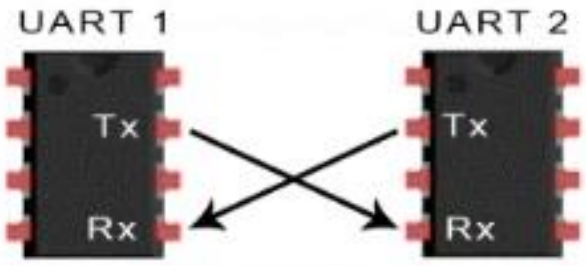
   Ejemplos comunes de velocidad son: 9600 bps, 115200 bps, entre otros.

5. **Tipos de comunicación:** Simplex, Half-Duplex o Full-Duplex:
   - **Simplex:** La comunicación solo ocurre en una dirección.
   - **Half-Duplex:** Los datos se transmiten en ambas direcciones, pero no simultáneamente.
   - **Full-Duplex:** Los datos se pueden transmitir y recibir simultáneamente.

## 4.3 Conexión / Envío / Recepción[2]

### 4.3.1 Conexión Serial

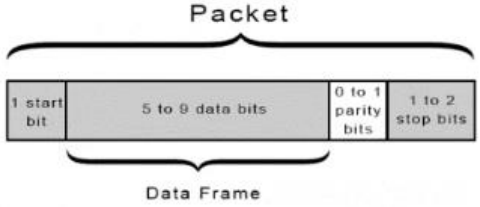
La conexión de dos dispositivos UART es sencilla y directa.
La Figura a continuación muestra un diagrama de conexión UART básico.




Un cable es para transmitir datos (llamado pin TX) y el otro es para recibir datos (llamado pin RX). Sólo podemos conectar dos dispositivos UART juntos.

### 4.3.2 Envío de Datos Serial

Antes de que el dispositivo UART pueda enviar datos, el dispositivo transmisor convierte los bytes de datos en bits. Después de convertir los datos en bits, el dispositivo UART los divide en paquetes para su transmisión. Cada paquete contiene un bit de inicio, una trama de datos, un bit de paridad y los bits de parada. La figura a continuación muestra un paquete de datos de ejemplo.




Después de preparar el paquete, el circuito UART lo envía a través del **pin TX**.

[2] https://arduino.cl/como-configurar-la-comunicacion-uart-en-arduino/

### 4.3.3 Recepción de Datos Serial

El dispositivo UART receptor comprueba si el paquete recibido (a través del pin RX) presenta errores, calculando el número de 1's y comparándolo con el valor del bit de paridad contenido en el paquete. Si no hay errores en la transmisión, procederá a eliminar el bit de inicio, los bits de parada y el bit de paridad para obtener la trama de datos.

### 4.3.4 Interfaz de UART Arduino

Arduino tiene uno o más pines UART dependiendo de la placa.

Para este caso de un Arduino Uno que tiene sólo una interfaz UART, la que se encuentra en el **pin 0 (RX0)** y el **pin 1 (TX0)**.

Los **pines 0 y 1 de Arduino también se utilizan para comunicarse con el IDE de Arduino a través del USB**. Así que, cuando se va a cargar sketches en el Arduino UNO, se debe **DESCONECTAR PRIMERO LOS CABLES DE LOS PINES 0 Y 1**.

En la figura muestra la ubicación de los pines **UART TX** y **RX** en la Placa Arduino Uno.


Pines Tx/Rx Arduino Uno

### 4.3.5 Nivel Lógico UART

Los niveles lógicos de UART pueden diferir entre fabricantes, en el caso de un Arduino Uno se tiene un nivel lógico de 5V pero el puerto RS232 de un ordenador tiene un nivel lógico de +/-12V. Conectar un Arduino Uno directamente a un puerto RS232 dañará el Arduino.

Si ambos dispositivos UART no tienen los mismos niveles lógicos, se necesita un circuito convertidor de nivel lógico adecuado para conectar los dispositivos.

### 4.4 Funciones.[3]

| Funciones | Descripción |
|---|---|
| ***Serial.begin(velocidad, config)*** | Establece la velocidad en bits por segundo (baudios) para la transmisión serie. |
| ***Serial.available()*** | Indica el número de bytes disponibles para la lectura desde el puerto serie. Los datos ya han llegado y se han almacenado en el buffer de recepción. |
| ***Serial.print*** | Imprime datos al puerto de serie como texto ASCII legible por humanos. |
| ***Serial.println*** | Imprime datos al puerto serie como texto ASCII legible por humanos seguido de un carácter de retorno de carro (ASCII 13, o '\ r') |
| ***Serial.read()*** | Retorna el primer byte de los datos de la entrada serie (o -1 si no hay datos disponibles) |
| ***Serial.write()*** | Escribe los datos binarios al puerto serie. |
| ***SerialEvent () {}*** | Se llama cuando se dispone de datos. |
| ***Serial.readStringUntil()*** | Lee los caracteres del buffer serie en una cadena. La función termina si se detecta el carácter terminador o el tiempo de espera se ha alcanzado. |

[3] https://docs.arduino.cc/language-reference/en/functions/communication/Serial/

### 4.5 Comunicación Bluetooth

Bluetooth es una especificación de comunicaciones industrial para redes inalámbricas de área personal creado por *Bluetooth Special Interest Group, Inc.*

Que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace de radiofrecuencia.

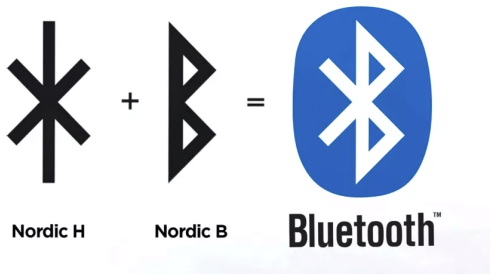
Objetivos:

- Facilitar comunicación entre dispositivos móviles
- Eliminar cables y conectores
- Ofrecer la disponibilidad de crear pequeñas redes inalámbricas y sincronizas dispositivos entre sí.

Etimología y Logo:

El nombre procede del rey danés y noruego *Harald Blatand*, cuya traducción al inglés es *Harald Bluetooth*, conocido por unificar las tribus danesas y convertirlas al cristianismo.

El logotipo es la combinación de las runas *Hagall* y *Berkana*, que corresponden con las iniciales del rey Harald.

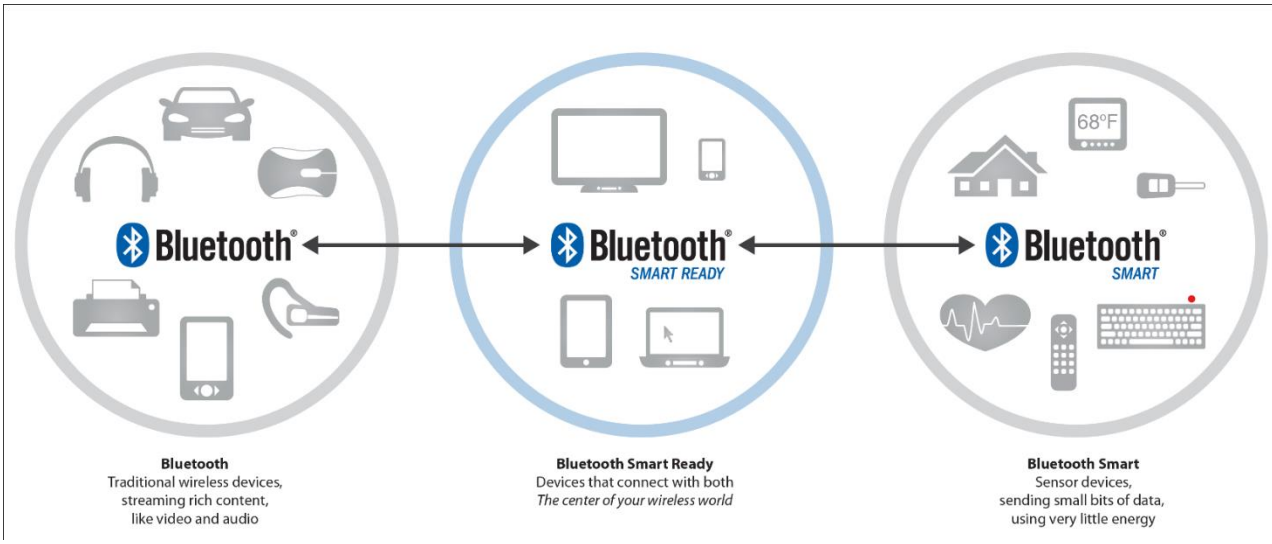



Usos y aplicaciones

Diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores (conjunto transmisor-receptor) de bajo coste.

Primer dispositivo Bluetooth

Primer teléfono móvil con Bluetooth




Se utiliza en un gran número de productos tales como teléfonos, impresoras, tabletas, altavoces, auriculares, etc.

Su uso es adecuado cuando se requiere conectar dos o más dispositivos en un área reducida sin grandes necesidades de ancho de banda.

Clasificaciones:

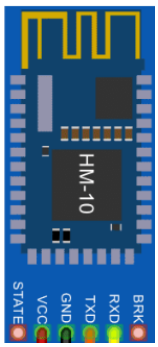
| **Potencia de transmisión** | | | |
|---|---|---|---|
| **Clase** | **Potencia Máx. Permitida (mW)** | **Potencia Máx. permitida (*dBm*)** | **Alcance aproximado** |
| **1** | 100mW | 20dBm | ~100m |
| **2** | 2,5mW | 4dBm | ~5-10m |
| **3** | 1mW | 0dBm | ~1m |
| **4** | 0.5mW | -3dBm | ~0.5m |

| Capacidad por canal | | |
|---|---|---|
| **Versión** | **Año** | **Ancho de Banda (BW)** |
| **V1.0** | 1999 | |
| **V1.1** | 2002 | |
| **V1.2** | 2003 | 721 kbit/s |
| **V2.0** | 2004 | 3 Mbit/s |
| **V2.1** | 2007 | 3 Mbit/s |
| **V3.0** | 2009 | 24 Mbit/s |
| **V4.0 (BLE)** | 2010 | 32 Mbit/s |
| **V5.0 (BLE)** | 2016-17 | |
| **V5.1 (BLE)** | 2019 | 32 Mbit/s |
| **V5.2 (BLE)** | 2020 | 50 Mbit/s |
| **V5.3 (BLE)** | 2022 | 32 Mbit/s |
| **V5.4 (BLE)** | 2023 | 50 Mbit/s |

### 4.5.1 Bluetooth Low Energy (BLE)

Comparado con Bluetooth clásico, Bluetooth Low Energy está diseñado para proporcionar un bajo consumo de energía, manteniendo un rango de alcance de comunicación similar.

*DataSheet* Módulo Inalámbrico AT-09 Bluetooth 4.0 CC2541




STATE – Pin digital que se activa con el funcionamiento del módulo.

VCC – Alimentación eléctrica 5Vcc

GND – Alimentación eléctrica GND

TXD – Transmisión comunicación serie

RXD – Recepción comunicación serie

BRK – Deshabilita el funcionamiento del módulo al alimentar la entrada.

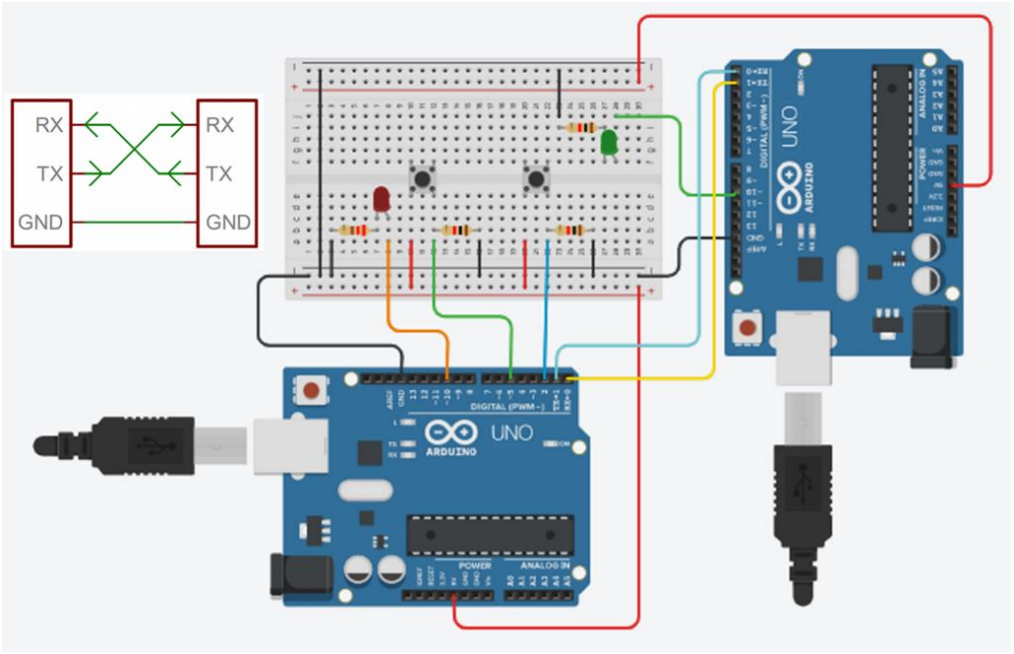
| Command | Description |
|---|---|
| **AT** | Check if the command terminal work normally |
| **AT+RESET** | Software reboot |
| **AT+VERSION** | Get firmware, bluetooth, HCI and LMP version |
| **AT+HELP** | List all the commands |
| **AT+NAME** | Get/Set local device name |
| **AT+PIN** | Get/Set pin code for pairing |
| **AT+LADDR** | Get local bluetooth address |
| **AT+DEFAULT** | Restore factory default |
| **AT+STATE** | Get current state |
| **AT+POWE** | Get/Set RF transmit power |
| **AT+ROLE** | Get/Set current role. |

## 5 Ejercicio 01. Conexión Serial de dos Arduino UNO. (Valoración 2 punto)

### 5.1 Objetivos.

- Conectar dos Placas Arduino Uno (A y B) para comunicación Serial.
- Encender y Apagar un Led, conectado a la Placa Arduino B desde dos pulsadores conectados en la placa Arduino A.

**Conectar los componentes como se muestra en el siguiente esquema:**




| **Nota:** **Verifique** sus conexiones con el **técnico del laboratorio** antes de continuar con la actividad. |
|---|

### 5.2 Programa.

- Abrir en su entorno de desarrollo (IDE Arduino) el programa **IoTPL06E01A.ino** descargado desde el material de la práctica para el ejercicio 01 y grabar en el Arduino A.
- Abrir en su entorno de desarrollo (IDE Arduino) el programa **IoTPL06E01B.ino** descargado desde el material de la práctica para el ejercicio 01 y grabar en el Arduino B.

### 5.3 Prueba.

Reiniciar ambos los Arduinos.

### 5.4 Validar Ejercicio.

Mostrar al técnico del laboratorio el funcionamiento del modelo para el **registro en la plantilla de valoración** de la práctica**.**

## 6 Ejercicio 02. Actividad Autónoma Grupal I. (Valoración 6 puntos)

**Requerimientos**

**a) Utilizando como base el esquema del ejercicio 01 realice lo siguiente:**

- Sustituya el Led del Arduino B por un Motor DC (Ver conexiones en el ejercicio 01 de la Práctica 05).
- Agregue un nuevo Led de color Verde conectado al Arduino A (Este se enciende cuando el Motor conectado al Arduino B está apagado).
- Retire el Botón de Apagado del Esquema conectado al Arduino A.
- Una vez que se pulsa el botón de encendido en el Arduino A, se comunica al Arduino B la orden de encendido del motor, se apagará el Led Verde y se enciende el Led Rojo para indicar que el Motor está trabajando.
- El motor es controlado desde el Arduino B y deberá completar un ciclo de 5 segundos cuando recibe la orden de iniciar.
- Cuando se complete el ciclo de funcionamiento deberá apagarse y el Arduino B deberá informar al Arduino A que ha finalizado su ciclo.
- El Arduino A deberá apagar el Led Rojo y encender el Led Verde.
- El Arduino A no podrá procesar nuevos intentos de encendido desde el botón mientras este el Led Rojo encendido.

**b) Programas Arduinos.**

Realice las modificaciones necesarias en los programas para que se cumplan los requisitos especificados en el apartado anterior y guardarlos como:

Arduino A: **IoTPL06E02A.ino** Arduino B: **IoTPL06E02B.ino**

**Nota:** **Verifique** sus conexiones con el **técnico del laboratorio** antes de continuar con la actividad.

### 6.1 Programa.

- Actualizar los nuevos programas en los arduinos.

### 6.2 Prueba.

Reiniciar ambos los Arduinos.

### 6.3 Validar Ejercicio.

Mostrar al técnico del laboratorio el funcionamiento del modelo para el **registro en la plantilla de valoración** de la práctica**.**

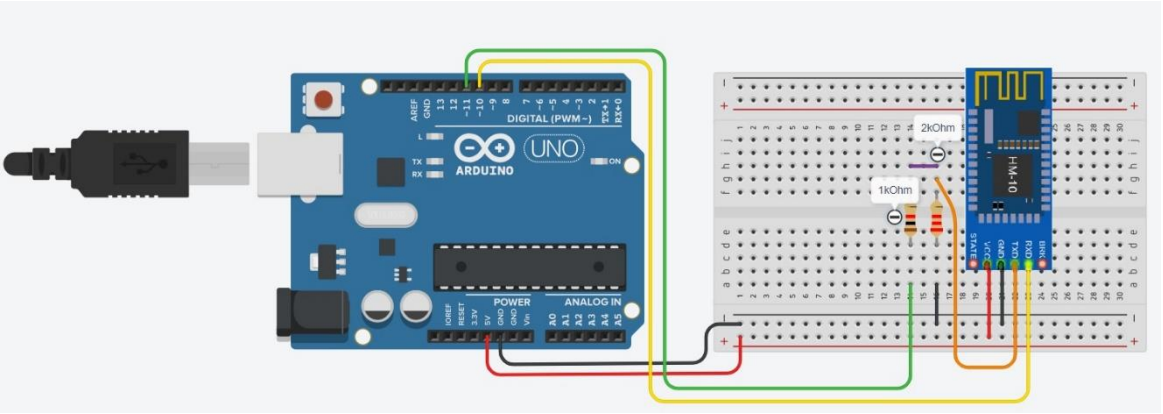
## 7 Ejercicio 03. Configuración de módulo Bluetooth BLE HM-10 (Valoración 2 punto)

- El objetivo de este ejercicio es comprender el funcionamiento del módulo Bluetooth ***HM-10*** (basado en el chip BLE ***CC2541***) y configurar el mismo para su correcto funcionamiento.
- La configuración del módulo puede realizarse a través de comunicación serie mientras el módulo no se encuentre emparejado o conectado a otro dispositivo.
- En este modo el led del módulo parpadea, indicando que no está emparejado.
- El monitor o terminal serie del Arduino IDE debe configurarse a la velocidad de 9600kbps y el envío de NL y CR (nueva línea y retorno de carro).

Las instrucciones o comandos se envían desde la terminal serie del Arduino IDE y están disponibles en su DATASHEET, en la tabla se muestran algunas de uso frecuente:

| Comando | Descripción |
|---|---|
| AT | Comprueba si la terminal de comandos funciona |
| AT + VERSION | Devuelve la versión de firmware del chip |
| AT + HELP | Muestra todos los comandos |
| AT + NAME | Devuelve/Establece el nombre local del dispositivo |
| AT + ADDR | Devuelve la dirección Bluetooth local del módulo. |
| AT + ROLE | Devuelve/Establece el rol actual (Maestro/Esclavo) |

### 7.1 Conectar los componentes como se muestra en el siguiente esquema:




- Realice las conexiones correspondientes según el esquema mostrado.

**Nota:** **Verifique** sus conexiones con el **técnico del laboratorio** antes de continuar con la actividad.

### 7.2 Programa.

Abrir en su entorno de desarrollo (IDE Arduino) el programa **IoTPL06E03.ino** descargado desde el material de la práctica 06.

### 7.3 Prueba.

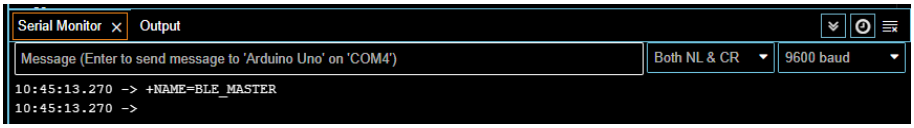
- Cargar el programa en la placa Arduino para su ejecución.
- Cambiar el nombre del Módulo por uno unívoco.
  a) Ejemplo 01 Nombre del Módulo.
  b) Ejemplo 02. Dirección del Módulo y Rol.
  c) Comprobar el resultado de los siguientes comandos AT desde la terminal serie, teniendo en cuenta los datos del DATASHEET.

**AT**:


Respuesta:

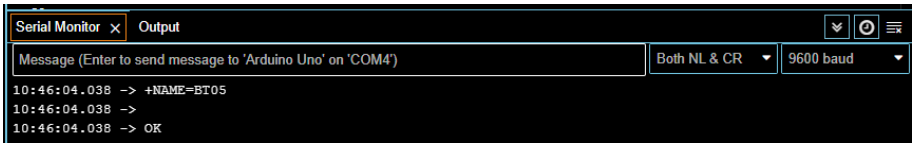

AT+NAME:


Respuesta:




CAMBIAR NOMBRE:


Respuesta:




### 7.4 Validar Ejercicio.

Mostrar al técnico del laboratorio el funcionamiento del modelo para el **registro en la plantilla de valoración** de la práctica**.**

## 8 Ejercicio 04. Actividad Complementaria Grupal I. (Opcional)

El objetivo de este ejercicio es gestionar componentes conectados a los terminales de Arduino desde un dispositivo inalámbrico mediante la conexión Bluetooth Low Energy (BLE)

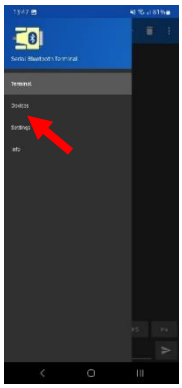
Descargar e instalar la aplicación «Serial Bluetooth Terminal» (Android) y mediante ella enviar y recibir mensajes desde Móvil – Terminal Serie de Arduino IDE.

**Serial Bluetooth Terminal**

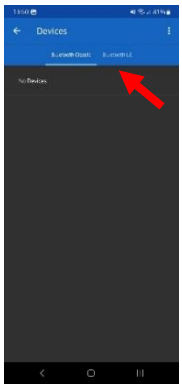
- Preparación de la aplicación para su uso:

1.- Abrir la aplicación y seleccionar el menú de la parte superior izquierda:


2.- Seleccionar “DEVICES” para buscar los dispositivos visibles Bluetooth

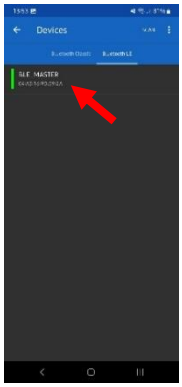


3.- Seleccionar visibilizar la búsqueda de dispositivos LE.

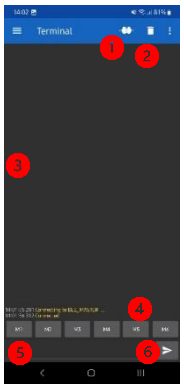



4.- En la lista de dispositivos seleccionar el dispositivo deseado para establecer la conexión.

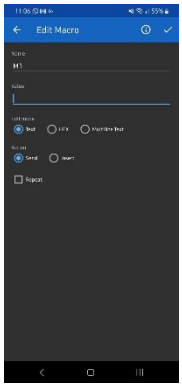
El nombre del dispositivo será el que hayamos configurado en el ejercicio anterior.




1. Conectar/Desconectar.
2. Limpiar historial de terminal.
3. Historial de intercambios con esclavo bluetooth
4. Botones configurables, pulsación larga entra en la configuración de los mismos.
5. Terminal de intercambio de comandos.
6. Enviar comando al esclavo bluetooth.




Desde la terminal de intercambio introducimos los comandos que hemos programado en Arduino que hacen encender o apagar el LED. Otra opción es configurar alguno de los botones configurables con el comando para que al pulsarlo envíe directamente.




Name: Nombre del botón

Value: Valor que queremos enviar

Edit Mode: Tipo de dato que enviamos

Action: Accion que debe hacer el botón al ser pulsado

Repeat: Habilitar o no la repetición de la acción al pulsar y configurar dicha acción.

## 8.1 Requerimientos

**a) Utilizando como base el esquema del ejercicio 03 realice lo siguiente:**

- Añada un led a la protoboard y realice las conexiones con Arduino necesarias.
- Con la base del programa **IoTPL06E03.ino** cree uno nuevo, guardándolo como **IoTPL06E04.ino**
- Realice las modificaciones necesarias en el programa para que, desde la terminal de comandos de la aplicación y a través de la conexión bluetooth logre encender y apagar el led añadido.
- Configure los botones necesarios en la aplicación “Bluetooth Serial Terminal” para que los comandos enviados desde los mismos sean los esperados en Arduino.

## 8.2 Programa.

- Actualizar el nuevo programa en Arduino

## 8.3 Prueba.

Reiniciar el Arduino.

## 8.4 Validar Ejercicio.

Mostrar al técnico del laboratorio el funcionamiento del modelo para el **registro en la plantilla de valoración** de la práctica**.**