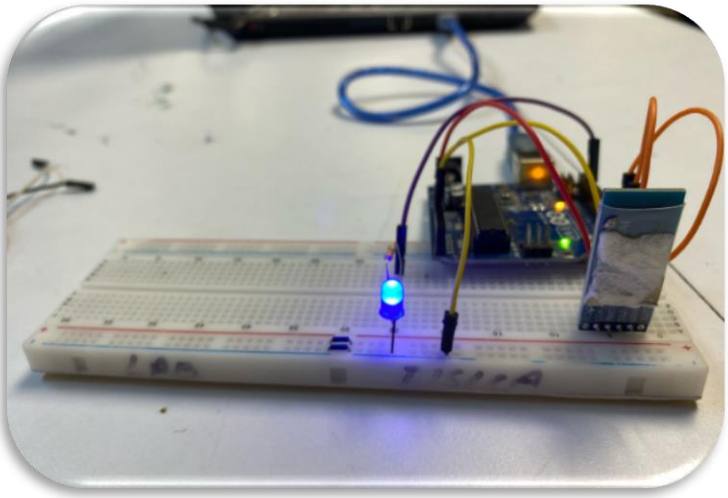
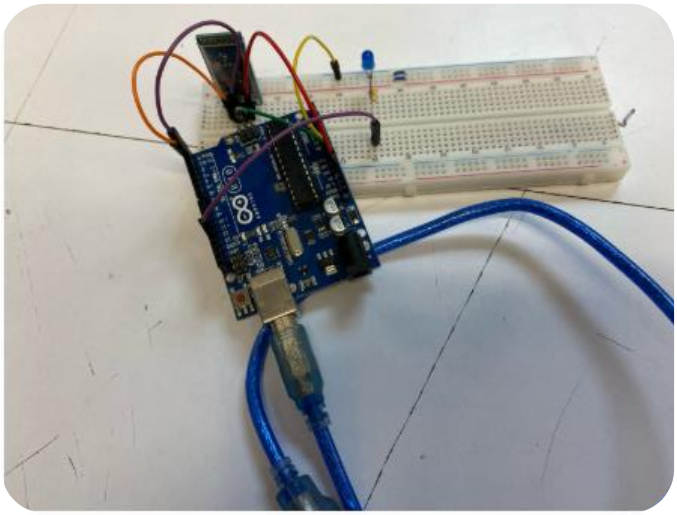
ESTABLECER UNA COMUNICACIÓN UART CON ARDUINO





INTEGRANTES DEL EQUIPO:

VÁZQUEZ HERNÁNDEZ EMILY EDITH

HERNANDEZ HENANDEZ MILCA LIZBETH

CORPUS RENTERIA ANTONIO

VARGAS JIMENEZ JESSICA JANETH

MALDONADO VAZQUEZ CELESTE

VERGARA TORREZ CRHISTIAN DE JESUS

PROFESOR:

DR. DANIEL LOPEZ PIÑA

MATERIA:

DISEÑO ELECTRONICO BASADO EN SISTEMAS EMBEBIDOS

CARRERA:

ISC

UNIDAD ACDÉMICA MULTIDISCIPLINARIA MANTE

8EJF

INTRODUCCIÓN

Esta práctica nos permitirá explorar y comprender cómo establecer una comunicación inalámbrica entre un microcontrolador Arduino y dispositivos Bluetooth, utilizando el protocolo UART (Universal Asynchronous Receiver/Transmitter). La comunicación UART es un método común de comunicación serial que permite el intercambio de datos entre dos dispositivos. Al combinarlo con un módulo Bluetooth, podemos lograr una comunicación inalámbrica bidireccional, abriendo un abanico de posibilidades para proyectos de control remoto, monitoreo y automatización.

Materiales necesarios

- Placa Arduino (por ejemplo, Arduino Uno)
- Módulo Bluetooth (por ejemplo, HC-05 o HC-06)
- Cables de conexión macho-macho (6 unidades)
- Resistencia de 330 ohmios
- LED (del color que prefieras)

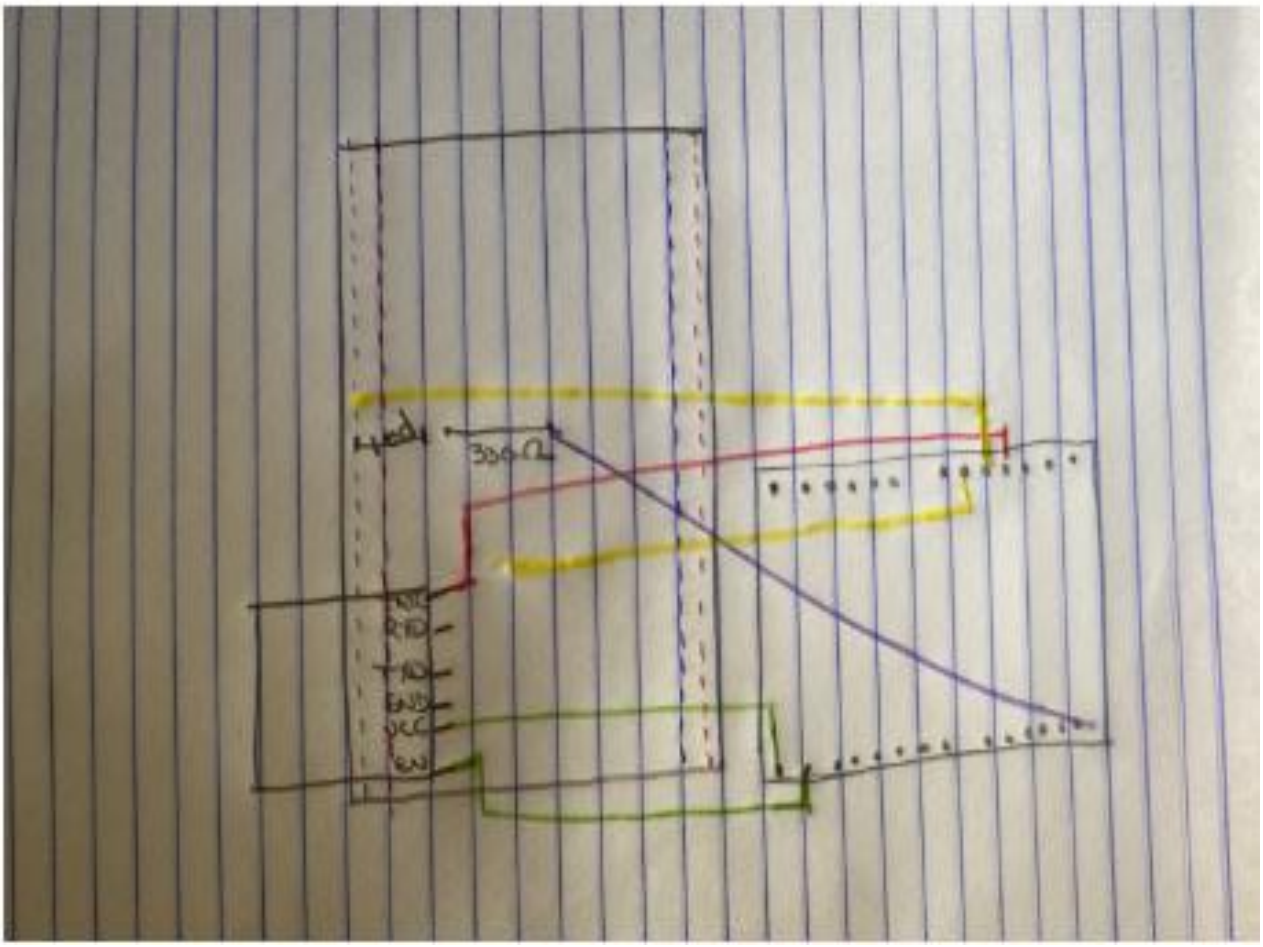
Conexión del módulo Bluetooth a Arduino

- VCC: Conectar a 5V de Arduino.
- GND: Conectar a GND de Arduino.
- TX del módulo Bluetooth: Conecte a RX de Arduino.
- RX del módulo Bluetooth: Conecte a TX de Arduino.

Aplicaciones utilizadas

- Serial bluetooth terminal Android
- Arduino IDE

DIAGRAMA



CÓDIGO

Modulo-Bluetooth | Arduino IDE 2.1.0

File Edit Sketch Tools Help

Arduino Uno

Modulo-Bluetooth.ino

```
1   const int ledPin = 13; // Pin al que está conectado el LED
2
3   void setup() {
4     Serial.begin(9600);
5     pinMode(ledPin, OUTPUT); // Configurar el pin del LED como salida
6   }
7
8   void loop() {
9     if (Serial.available()) {
10      char dato = Serial.read();
11      Serial.print("Recibido: ");
12      Serial.println(dato);
13      digitalWrite(ledPin, HIGH); // Encender el LED
14      delay(200); // Esperar 200 milisegundos
15      digitalWrite(ledPin, LOW); // Apagar el LED
16    }
17  }
```

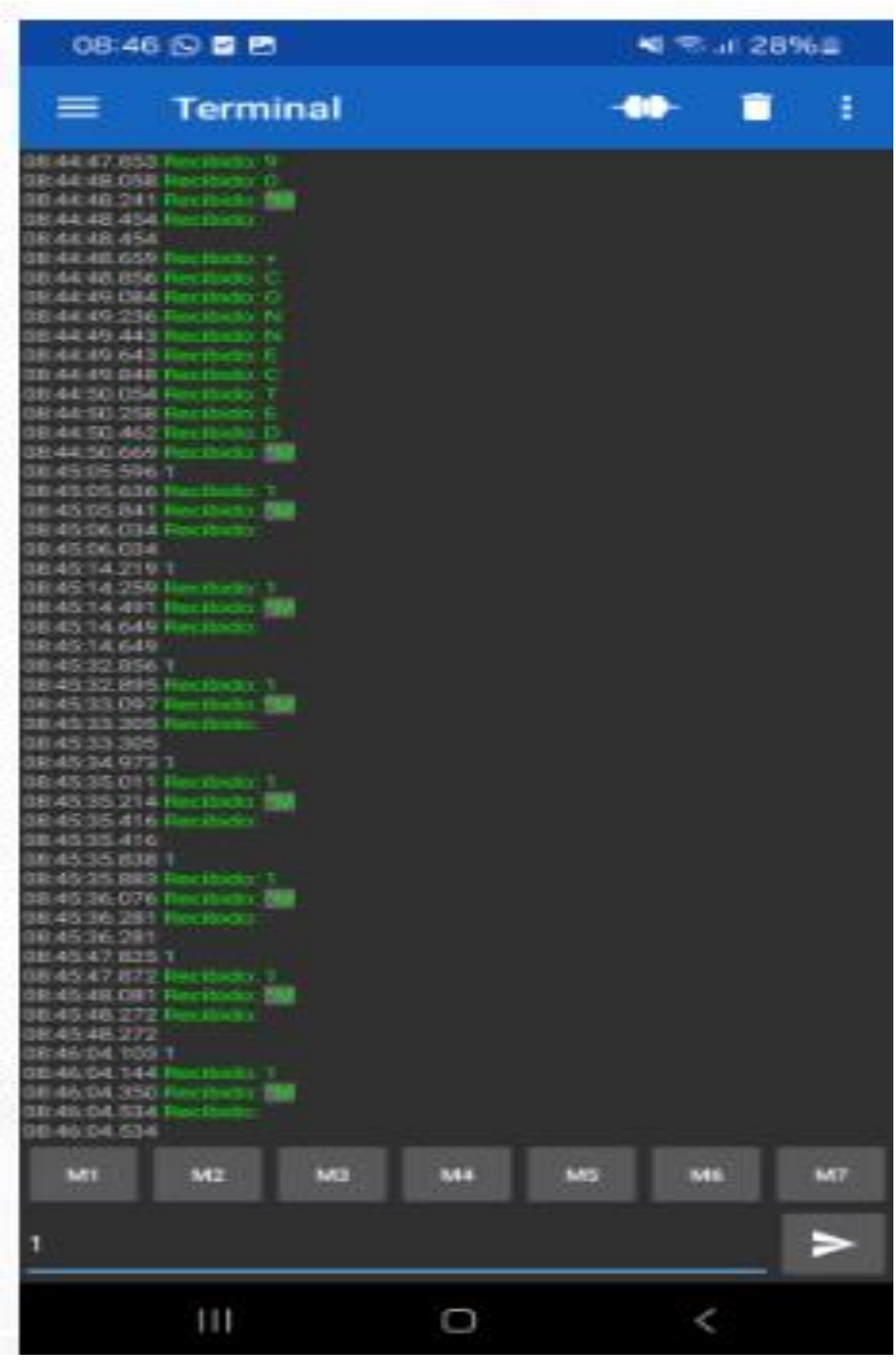
Output

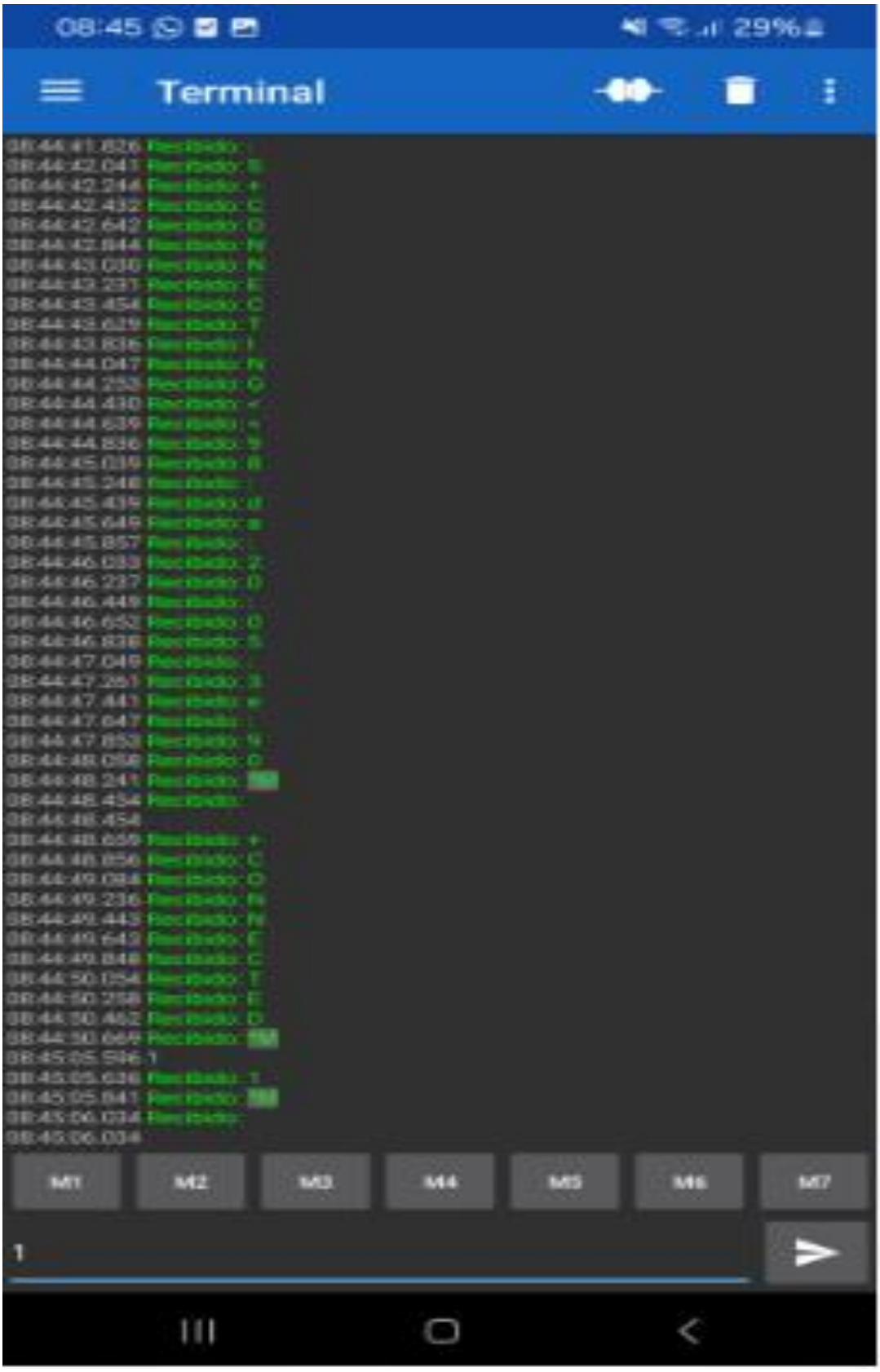
```
El Sketch usa 1958 bytes (6%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 198 bytes (9%) de la memoria dinámica, dejando 1850 bytes para las variables locales. El máximo es 2048 bytes.
```

Explicación del código actualizado:

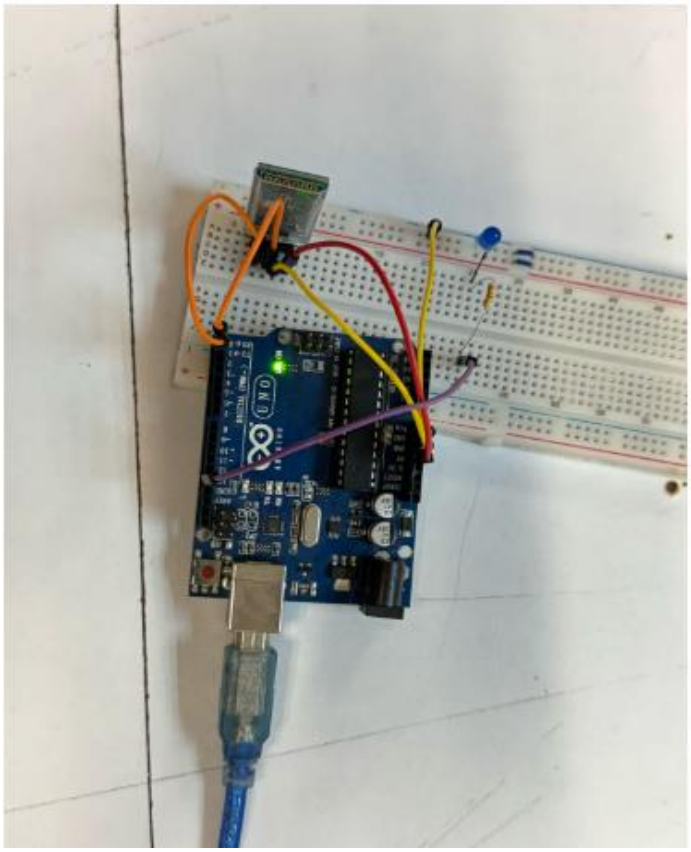
- Se define una constante `ledPin` para representar el pin digital al que está conectado el LED.
- En la función `setup()`, se configura `ledPin` como una salida.
- En la función `loop()`, cuando se recibe un dato a través de Bluetooth:
  - Se imprime el dato recibido en el monitor serial.
  - Se enciende el LED (`digitalWrite(ledPin, HIGH)`).
  - Se espera 200 milisegundos (`delay(200)`).
  - Se apaga el LED (`digitalWrite(ledPin, LOW)`).

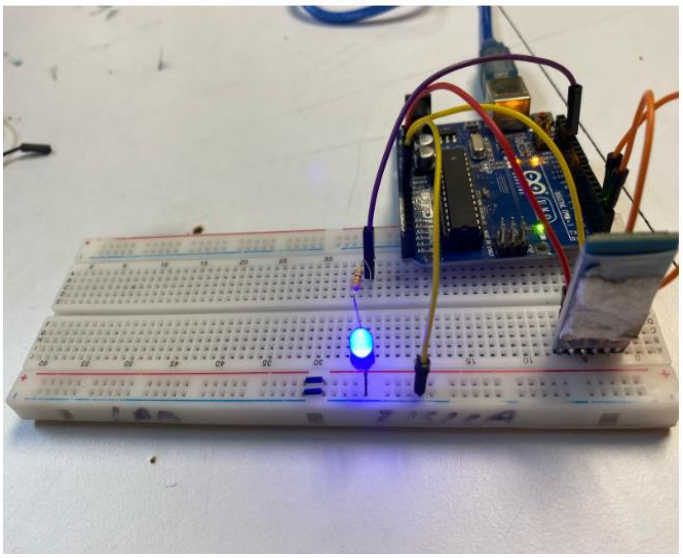
SERIAL BLUETOOTH TERMINAL ANDROID

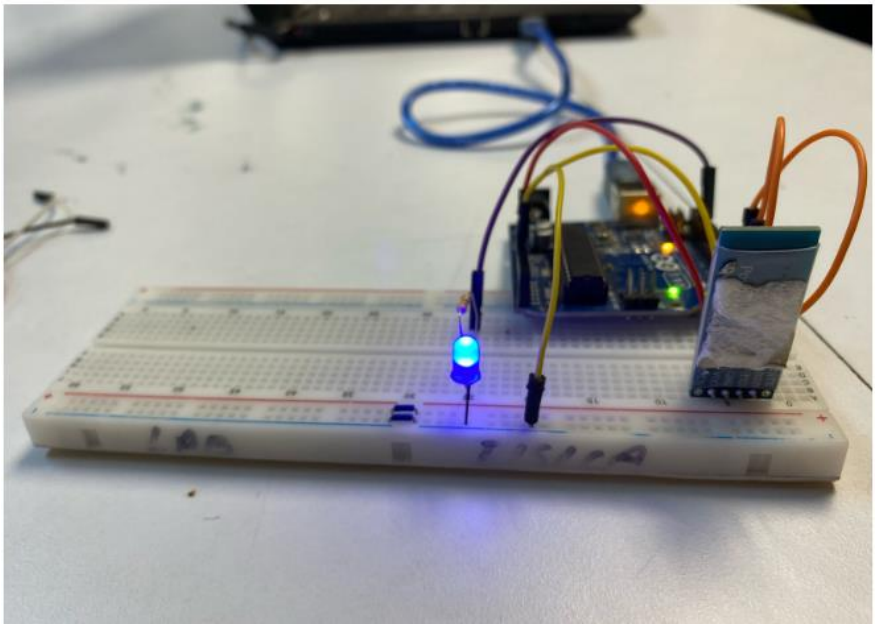



08:45
29%
Terminal
M1
M2
M3
M4
M5
M6
M7
1

EVIDENCIAS







MODIFICACIÓN DE CÓDIGO PARA QUE PARPADEE EL LED

BluetoothParpadeo | Arduino IDE 2.1.0

File Edit Sketch Tools Help

Arduino Uno

BluetoothParpadeo.ino

```
int ledpin = 13;
bool parpadeoActivo = false; // Variable para controlar el parpadeo

void setup() {
  Serial.begin(9600);
  pinMode(ledpin, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    char data = Serial.read();

    if (isAlpha(data)) { // Si es una letra, activa el parpadeo
      Serial.println("Parpadeo activado");
      parpadeoActivo = true;
    } else if (data == '0') { // Si recibe '0', detiene el parpadeo
      Serial.println("Parpadeo detenido");
      parpadeoActivo = false;
      digitalWrite(ledpin, LOW);
    }
  }

  // Si el parpadeo está activo, hacer parpadear el LED
  if (parpadeoActivo) {
    digitalWrite(ledpin, HIGH);
    delay(500);
    digitalWrite(ledpin, LOW);
    delay(500);
  }
}
```

1. Declaración de variables:

- int ledpin = 13;: Esta línea declara una variable entera llamada ledpin y la inicializa con el valor 13. Esta variable representa el número del pin digital al que está conectado el LED.
- bool parpadeoActivo = false;: Esta línea declara una variable booleana llamada parpadeoActivo y la inicializa con el valor false. Esta variable se utiliza para controlar si el LED debe parpadear o no.

2. Función setup():

- Serial.begin(9600);: Esta línea inicializa la comunicación serial a una velocidad de 9600 baudios. Esto permite que el Arduino se comunique con otros dispositivos, como un módulo Bluetooth o una computadora, a través del puerto serial.
- pinMode(ledpin, OUTPUT);: Esta línea configura el pin digital ledpin como una salida. Esto significa que el Arduino puede enviar señales eléctricas a este pin para encender o apagar el LED.

3. Función loop():

- if (Serial.available() > 0): Esta línea verifica si hay datos disponibles en el puerto serial. Si hay datos disponibles, el código dentro del bloque if se ejecuta.
  - char data = Serial.read();: Esta línea lee el primer byte de datos disponibles en el puerto serial y lo almacena en la variable data de tipo carácter.
  - if (isAlpha(data)): Esta línea verifica si el carácter data es una letra (alfabético).
    - Si data es una letra, se ejecuta el siguiente bloque:
    - Serial.println("Parpadeo activado");: Se imprime el mensaje "Parpadeo activado" en el monitor serial.
    - parpadeoActivo = true;: Se establece la variable parpadeoActivo en true, lo que indica que el LED debe comenzar a parpadear.
  - else if (data == '0'): Esta línea verifica si el carácter data es el número '0'.
    - Si data es '0', se ejecuta el siguiente bloque:
    - Serial.println("Parpadeo detenido");: Se imprime el mensaje "Parpadeo detenido" en el monitor serial.
    - parpadeoActivo = false;: Se establece la variable parpadeoActivo en false, lo que indica que el LED debe dejar de parpadear.

- digitalWrite(ledpin, LOW);: Se apaga el LED.
- if (parpadeoActivo): Esta línea verifica si la variable parpadeoActivo es true.
  - Si parpadeoActivo es true, se ejecuta el siguiente bloque:
- digitalWrite(ledpin, HIGH);: Se enciende el LED.
- delay(500);: Se espera 500 milisegundos.
- digitalWrite(ledpin, LOW);: Se apaga el LED.
- delay(500);: Se espera 500 milisegundos.

En resumen:

- El código recibe datos a través del puerto serial.
- Si recibe una letra, activa el parpadeo del LED.
- Si recibe el carácter '0', detiene el parpadeo del LED y lo apaga.
- Mientras el parpadeo esté activo, el LED parpadea a intervalos de 500 milisegundos.

CONCLUSIÓN

A través de esta práctica, hemos logrado establecer una comunicación inalámbrica bidireccional entre un Arduino y un dispositivo móvil mediante Bluetooth, utilizando el protocolo UART. Este proceso no solo nos permitió enviar y recibir datos, sino que también exploramos la capacidad de controlar dispositivos externos, como un LED, de forma inalámbrica.

Además de simplemente encender y apagar el LED, hemos ampliado la funcionalidad para incluir el parpadeo del LED, lo que demuestra la versatilidad de la comunicación Arduino-Bluetooth para crear sistemas interactivos y dinámicos.

En resumen, esta práctica nos proporcionó una base sólida para comprender y aplicar la comunicación UART y Bluetooth en proyectos futuros, abriendo un mundo de posibilidades para la creación de dispositivos inalámbricos y sistemas de control remoto.