Materia:

DISEÑO ELECTRÓNICO BASADO EN SISTEMAS EMBEBIDOS

Alumno:

Posadas Pérez Isaac Sayeg

Paniagua Rico Juan Julian

García Azzúa Jorge Roberto

Grado y grupo:

8°G

Profesor:

Garcia Ruiz Alejandro Humberto

Tarea 7:

Instrucciones que componen al paquete serial (readString, TimeOut, begin, etc.)

Instrucciones que Componen el Paquete Serial en Arduino: `readString`, `TimeOut`, `begin`, etc.

El paquete serial en Arduino es fundamental para la comunicación entre la placa y otros dispositivos, como computadoras, módulos y sensores. Permite la transmisión de datos de manera eficiente y es ampliamente utilizado en aplicaciones que requieren la transferencia de información. En esta investigación, abordaremos las instrucciones más comunes del paquete serial, incluyendo `Serial.begin`, `Serial.readString`, y el concepto de `TimeOut`, entre otros.

1. Introducción a la Comunicación Serial

La comunicación serial es un método de transmisión de datos en el que los bits se envían uno tras otro a través de un solo canal. En Arduino, la comunicación serial se realiza a través de pines específicos (por lo general, el pin 0 para RX y el pin 1 para TX) y se utiliza principalmente para la depuración y la interacción con otros dispositivos. La función `Serial` proporciona métodos para enviar y recibir datos a través de esta interfaz.

2. Funciones Comunes del Paquete Serial

2.1. `Serial.begin()`

La función `Serial.begin(speed)` se utiliza para iniciar la comunicación serial. Debe ser llamada en la función `setup()` para establecer la velocidad de transmisión, que se mide en baudios.

Sintaxis:

Serial.begin(baudrate);

- `baudrate`: La velocidad de transmisión en baudios (por ejemplo, 9600, 115200, etc.).

Ejemplo:

void setup() {

```
    Serial.begin(9600); // Inicia la comunicación serial a 9600 bps
}
```

2.2. `Serial.print()` y `Serial.println()`

Estas funciones se utilizan para enviar datos al monitor serie. La diferencia entre ellas es que `Serial.println()` agrega un salto de línea al final del mensaje.

Ejemplo:

```
void loop() {
    Serial.print("La temperatura es: "); // Envía el texto sin salto de línea
    Serial.println(25);                 // Envía el valor y agrega un salto de línea
    delay(1000);                        // Espera 1 segundo
}
```

2.3. `Serial.read()`

La función `Serial.read()` se utiliza para leer un byte de datos de la entrada serial. Si no hay datos disponibles, devolverá -1.

Sintaxis:

```
int dato = Serial.read();
```

Ejemplo:

```
void loop() {
    if (Serial.available() > 0) {       // Verifica si hay datos disponibles
        int dato = Serial.read();       // Lee un byte
        Serial.print("Dato recibido: "); // Imprime el dato recibido
        Serial.println(dato);
    }
}
```

2.4. `Serial.readString()`

`Serial.readString()` se utiliza para leer una cadena de caracteres de la entrada serial hasta que se encuentra un delimitador, como un salto de línea o un tiempo de espera.

Sintaxis:

String cadena = Serial.readString();

Ejemplo:

```
void loop() {
   String entrada = Serial.readString(); // Lee una cadena de caracteres
   Serial.print("Cadena recibida: ");    // Imprime la cadena recibida
   Serial.println(entrada);
}
```

2.5. `Serial.setTimeout()`

La función `Serial.setTimeout(timeout)` se utiliza para establecer el tiempo máximo de espera para la lectura de datos de la entrada serial. Esto es útil para evitar bloqueos en la ejecución si no se reciben datos en un período específico.

Sintaxis:

Serial.setTimeout(timeout);

- `timeout`: El tiempo de espera en milisegundos.

Ejemplo:

```
void setup() {
   Serial.begin(9600); // Inicia la comunicación serial
   Serial.setTimeout(2000); // Establece el tiempo de espera en 2 segundos
}

void loop() {
   String entrada = Serial.readString(); // Lee una cadena con tiempo de espera
   Serial.print("Cadena recibida: ");
   Serial.println(entrada);
}
```

3. Ejemplo Completo de Uso del Paquete Serial

A continuación, se presenta un ejemplo que combina varias funciones del paquete serial para leer datos de un sensor y enviarlos al monitor serie.

```
void setup() {
    Serial.begin(9600); // Inicia la comunicación serial
    Serial.setTimeout(1000); // Establece el tiempo de espera en 1 segundo
}

void loop() {
    if (Serial.available() > 0) { // Verifica si hay datos disponibles
        String comando = Serial.readString(); // Lee un comando
        Serial.print("Comando recibido: ");
        Serial.println(comando); // Imprime el comando recibido

        if (comando == "ON") {
            Serial.println("LED encendido.");
            // Código para encender un LED
        } else if (comando == "OFF") {
            Serial.println("LED apagado.");
            // Código para apagar un LED
        } else {
            Serial.println("Comando desconocido.");
        }
    }
}
```

4. Conclusiones

El paquete serial en Arduino es una herramienta para la comunicación entre dispositivos y la depuración de programas. Las funciones como `Serial.begin`, `Serial.readString`, y `Serial.setTimeout` son fundamentales para establecer una comunicación que sea efectiva y eficiente. Comprender cómo utilizar estas funciones permite a los usuarios desarrollar proyectos más interactivos y responsivos.

Bibliografía

1. Banzi, M., & Shiloh, M. (2014). *Getting Started with Arduino*. Maker Media, Inc.

- Este libro proporciona una introducción a la plataforma Arduino y explica el uso de la comunicación serial.

2. Monk, S. (2015). *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education.

    - Un recurso que presenta ejemplos prácticos sobre la programación en Arduino, incluyendo el uso del paquete serial.
3. Arduino. (n.d.). *Arduino Reference*. Retrieved from https://www.arduino.cc/reference/en/

    - Documentación oficial de Arduino que detalla las funciones del paquete serial.
4. Adafruit. (n.d.). *Learning System*. Retrieved from https://learn.adafruit.com/

    - Un recurso en línea que ofrece tutoriales sobre programación de Arduino y ejemplos prácticos.
5. Simon Monk. (2014). *Arduino Cookbook*. O'Reilly Media.

    - Un libro que ofrece una colección de recetas y ejemplos prácticos para trabajar con Arduino en diferentes proyectos.