Año: 2023 Volumen: 1
Número 9, ISSN: 2448-623X

Diseño de protocolo comando – parámetro mediante comunicación serie asincrónica universal aplicado al control de servomotores

Machado-Díaz, E[1]; Coto-Fuentes, H[1]; Alvarado-Tovar, N[1].

Datos de Adscripción:
[1]Tecnológico Nacional de México (TecNM): Instituto Tecnológico Superior de Lerdo, División de Investigación y Desarrollo Tecnológico, Av. Tecnológico No. 1555 Sur Periférico Gómez - Lerdo Km. 14.5, Ciudad Lerdo, Durango; México. C.P. 35150.
eduardo.md@itslerdo.edu.mx

Resumen **- El protocolo serie asincrónico universal (UART) es ampliamente utilizado en el diseño electrónico para la comunicación entre dispositivos, especialmente microcontroladores. Su facilidad de uso y compatibilidad con numerosos sistemas lo convierten en una opción popular. Sin embargo, su naturaleza plantea desafíos en el manejo de cadenas de caracteres y el intercambio de información.**

Con el objetivo de simplificar la comunicación entre dispositivos con microcontrolador, se ha desarrollado un programa de ejemplo para la plataforma Arduino. Este programa utiliza el protocolo UART para el intercambio de información entre dos microcontroladores.

El protocolo consiste en un comando y un parámetro, lo que permite el envío y recepción de valores numéricos de manera ordenada y sencilla.

Una aplicación destacada de este programa de ejemplo es el control de servomotores. Mediante una interfaz visual, los comandos son enviados desde un dispositivo externo y recibidos por una placa Arduino. La información es procesada y utilizada para controlar múltiples servomotores. Esto proporciona una solución eficiente para el control de movimientos precisos en diversos proyectos.

La capacidad de manejar grandes cantidades de información de forma estructurada y reproducible hace que este programa de ejemplo sea una herramienta versátil para aquellos que trabajan con el protocolo UART. Además, al estar diseñado para la plataforma Arduino, puede ser fácilmente implementado en otros sistemas que utilicen este protocolo de comunicación.

El protocolo es ampliamente utilizado en el diseño electrónico y se destaca por su facilidad de uso y compatibilidad.

Palabras Clave **– Comando, Comunicación, Parámetro, Serie, Servomotor**

Abstract **- Abstract— The Universal Asynchronous Receiver-Transmitter (UART) serial protocol is widely used in electronic design for communication between devices, especially microcontrollers. Its ease of use and compatibility with numerous systems make it a popular choice. However, its nature poses challenges in handling character strings and exchanging information.**

With the aim of simplifying communication between microcontroller devices, an example program has been developed for the Arduino platform. This program utilizes the UART protocol for information exchange between two microcontrollers.

The protocol consists of a command and a parameter, enabling the transmission and reception of numerical values in an organized and straightforward manner.

One notable application of this example program is servo motor control. Through a visual interface, commands are sent from an external device and received by an Arduino board. The information is processed and used to control multiple servo motors. This provides an efficient solution for precise motion control in various projects.

The ability to handle large amounts of structured and reproducible information makes this example program a versatile tool for those working with the UART protocol. Additionally, being designed for the Arduino platform, it can be easily implemented in other systems that use this communication protocol.

The protocol is widely used in electronic design and is distinguished by its ease of use and compatibility.

Keywords **– Command, communication, parameter, Serial, Servomotor.**

I. INTRODUCCIÓN

Debido al incremento en los sistemas automatizados y el desarrollo e implementación de dispositivos encaminados a la industria 4.0, cada vez es más común encontrar proyectos en los que se utilizan servomotores como el actuador principal para generar un movimiento de precisión y con cierto torque de una forma fácil y rápida, realizando el control a través de microcontroladores de bajo costo como lo pueden ser Arduino, PIC, STM, etc (Domínguez Mínguez, 2020).

En este tipo de desarrollos intervienen también protocolos de comunicación, pues los dispositivos de control deben de realizar generalmente el intercambio de información con una interfaz. Para llevar a cabo esto, se utilizan comúnmente protocolos seriales, siendo el asíncrono universal uno de los más populares. Sin embargo, al ser una comunicación simple, existen desventajas al momento de utilizarlas en el manejo de grandes cantidades de información ya que, por la naturaleza de su protocolo, el envío de datos de forma serial puede ser complicado al momento de clasificar tramas de datos, en especial si se envían muchos parámetros al mismo tiempo (Bariain, 2017).

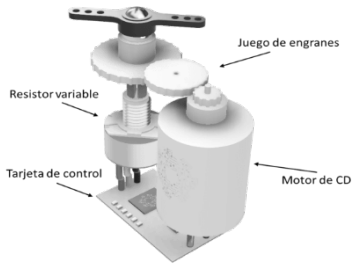
Debido a lo anterior, es de vital importancia crear formas de clasificar la información que se recibe y se transmite a través del medio de comunicación, es por ello que se propone un protocolo comando – parámetro para facilitar el control de tramas a través

de palabras reservadas que el microcontrolador es capaz de identificar y posteriormente almacenar un valor numérico relacionado a él para su uso en el programa en general.

1.1 Servomotor

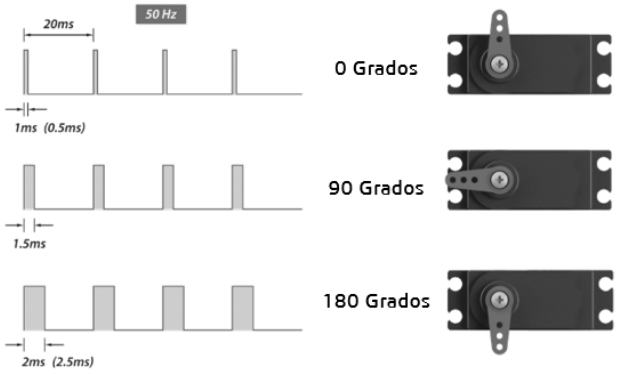
Son dispositivos de accionamiento que, mediante un sistema de control electrónico, permiten movimientos angulares de precisión. Contienen en su interior un encoder, conocido como decodificador, que convierte el movimiento mecánico (giros del eje) en pulsos digitales interpretados por un controlador de movimiento. También utilizan un driver, que en conjunto forman un circuito para comandar posición, torque y velocidad (Corona, 2014). La estructura general de un servomotor se muestra en la Figura 1.

Figura 1
Estructura interna general de un Servomotor



El control de posición angular se realiza mediante pulsos de una frecuencia constante y ancho variable, parecido a un PWM, utilizando los tiempos que se muestran en la Figura 2. Estos pulsos se generan a través de microcontroladores y de acuerdo a la duración del tiempo en alto se varía la posición general del eje del motor.

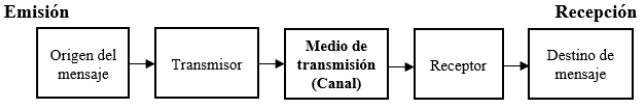
Figura 2
Pulsos para control de posición de servomotores



1.2 Protocolos de comunicación

Lo fundamental en cualquier tipo de comunicación es resolver el problema de cómo llevar el mensaje desde un punto A hacia un punto B sin errores o con la mínima probabilidad de ellos, esto se logra mediante la codificación del mensaje de la mejor manera posible. Este tipo de codificación recibe el nombre de medio de comunicación, que será el encargado de establecer la unión entre los dos puntos. Un ejemplo de lo mencionado anteriormente es el que se muestra en la Figura 3. (Coto, 2008).

Figura 3
Esquema básico de la comunicación entre dos puntos.



El protocolo se define como las reglas para la transmisión de la información a través del medio de intercambio de información. Un protocolo de red de comunicación es un conjunto de reglas que gobiernan el intercambio ordenado de datos dentro del sistema para la comunicación.

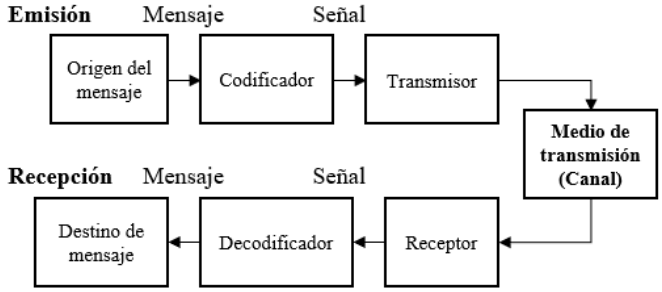
Los elementos básicos de un protocolo de comunicación son:

- Conjunto de símbolos o caracteres.
- Reglas para la secuencia y sincronización de los mensajes.
- Procedimientos para determinar cuándo se ha enviado el mensaje correctamente o se ha detectado un error

Para que exista comunicación entre ambos puntos por el canal se debe enviar la misma configuración en cada uno de ellos (Pérez, 2012).

Según Pérez, para el establecimiento de reglas es necesario identificar los siguientes puntos: Un emisor y receptor, método de comunicación acordado, idioma común (tipo de cifrado), velocidad y momento de entrega y requisitos de confirmación o acuse de recibo. Habiendo establecidos los puntos anteriores, el diagrama de la comunicación, una vez que se ha aplicado el protocolo, deberá ser similar al que se presenta en la Figura 4.

Figura 4
Diagrama de comunicación aplicando un protocolo.

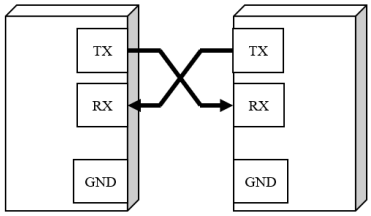


1.3 Comunicación asíncrona universal

La comunicación Transmisor – Receptor Asíncrono Universal (UART por sus siglas en inglés) define un protocolo o un conjunto de reglas que se establecen para el intercambio de datos entre dispositivos de forma serial (es decir, se envía un dato tras otro). Utiliza dos conductores entre el transmisor y el receptor para crear una comunicación bidireccional y se basa en el envío de información en forma de tramas, Así mismo, si la conexión es física, se debe de tener una referencia o tierra en las conexiones. la Figura 5, muestra un diagrama de comunicación básica en UART física (Díaz, 2009).

Año: 2023 Volumen: 1
Número 9, ISSN: 2448-623X

Figura 5
Esquema de funcionamiento del protocolo UART.



Existen diferentes tipos de comunicación UART.

- Comunicación Simplex: Los datos únicamente se envían en una sola dirección.
- Semidúplex: Existe una comunicación bidireccional pero un dispositivo a la vez.
- Full – dúplex: Ambos dispositivos transmiten simultáneamente.

La UART fue uno de los primeros protocolos de tipo serial pues se utiliza con interfaces RS-232. Sin embargo, a pesar de existir otros protocolos como el SPI, I2C o USB, sigue siendo común utilizarla para realizar comunicación entre microcontroladores por su facilidad de uso, fácil implementación y bajo coste, ya que se encuentra incrustado en la mayoría de los dispositivos de prototipado (Méndez de la Torre, 2022).

Una de las mayores ventajas que se pueden encontrar en el uso de este tipo de comunicación, es que es asíncrona, es decir que el transmisor y el receptor no comparten una señal de reloj.

1.4 Plataforma Arduino y su importancia en la actualidad

Arduino es una plataforma de desarrollo electrónico de código abierto que se ha convertido en una opción popular para implementar proyectos interactivos. Consiste en una placa con un microcontrolador y un entorno de desarrollo que facilita la programación y la interacción con diferentes dispositivos y sensores (Artero, 2013).

La versatilidad de Arduino radica en su capacidad para leer entradas de varios sensores y controlar salidas para interactuar con el mundo físico. La placa está equipada con pines de entrada/salida digitales y analógicos que se pueden programar para leer datos de sensores o enviar señales a actuadores como motores, luces y pantallas.

Una de las principales ventajas de Arduino es su simplicidad y facilidad de uso, lo que lo hace accesible tanto para principiantes como para desarrolladores experimentados. El lenguaje de programación de Arduino se basa en Wiring, una variante simplificada de C/C++, lo que facilita escribir código para controlar las funciones de la placa (Reyes Cortés, 2015).

Las placas de Arduino vienen en diferentes variantes, ofreciendo diversas características y capacidades. Además, hay una amplia biblioteca de funciones de software preconstruidas, conocidas como bibliotecas, que permiten a los usuarios intercalarse fácilmente con una amplia gama de sensores, módulos y protocolos de comunicación.

Arduino ha encontrado aplicaciones en numerosos campos, incluyendo robótica, domótica, proyectos de Internet de las Cosas (IoT), instalaciones de arte interactivo y proyectos educativos. Su precio asequible, simplicidad y amplio apoyo de la comunidad lo convierten en una opción popular tanto para aficionados, estudiantes como profesionales (López, 2016).

Este dispositivo es una versátil plataforma de desarrollo electrónico que permite a los usuarios crear proyectos interactivos mediante la programación de una placa de microcontrolador. Su facilidad de uso, amplio soporte de bibliotecas y amplia gama de aplicaciones lo convierten en una herramienta poderosa para dar vida a ideas en el mundo de la electrónica y los sistemas embebidos.

II. PARTE TÉCNICA

El protocolo de comunicación comando-parámetro que se diseñó fue implementado en la plataforma Arduino con la finalidad de verificar su funcionamiento. Sin embargo, puede ser adaptado para diferentes plataformas que contengan instrucciones similares.

2.1 Descripción general del protocolo.

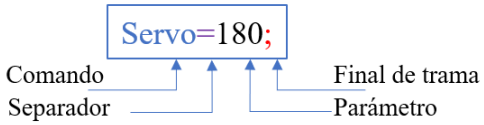
Para la descripción se tomará a la placa Arduino como receptor de información, y como transmisor puede ser cualquier dispositivo que contenga el protocolo UART y sea capaz de enviar cadenas de texto.

La trama principal del protocolo consta de 4 partes

- Comando: Es la palabra reservada que se encuentra programada internamente en el receptor y que será detectada por el programa para establecer si la trama es válida o no.
- Separador: Corresponde a un carácter especial (es decir, que no sea común su uso) que será el encargado de ser la división entre el comando y el valor numérico que se le va a asignar. Este carácter se define por el usuario, por lo que puede cambiarse de acuerdo a las necesidades.
- Parámetro: Es el valor numérico que se le va a asignar al comando que se ha establecido.
- Final de trama: Indica el final de la trama serial, permitiendo al receptor terminar la recepción de datos y comenzar con el análisis.

En la Figura 6, se muestra la estructura básica de la trama de forma visual, puede decirse que es una asignación de un valor numérico a una variable establecida. El ejemplo se presenta para el manejo de servomotores, donde el comando corresponde al servomotor al que quiere asignar un valor y el parámetro al valor en grados a asignar.

Figura 6
Ejemplo de estructura de control de servomotores



2.2 Descripción del algoritmo del protocolo.

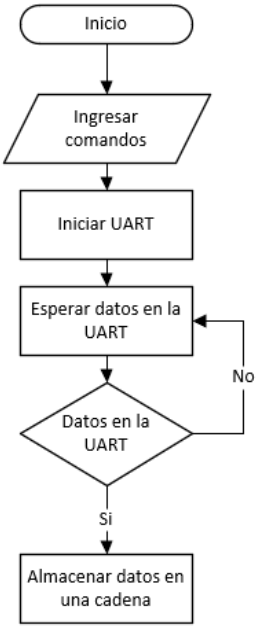
El algoritmo puede dividirse en varias etapas de acuerdo a las tareas que se realizan en cada una de ellas.

1. Primera fase: Ingreso de comandos y recepción de datos por la UART.
2. Segunda fase: Procesamiento de la cadena de caracteres.
3. Tercera fase: Identificación de comandos y almacenamiento de parámetros y toma de decisión para cada comando.

2.2.1 Primera fase

Como primer paso es necesario decidir cuáles palabras se van a establecer como parámetro, esto mediante un arreglo de 1D. Posteriormente, se realiza la configuración necesaria para el uso de la UART del microcontrolador y se espera la recepción de cualquier trama que provenga del transmisor, si existe información en el buffer esta se almacena en una cadena de texto. El diagrama de flujo de la primera fase se muestra en la Figura 7.

Figura 7
Diagrama de flujo primera fase



2.2.2 Segunda fase

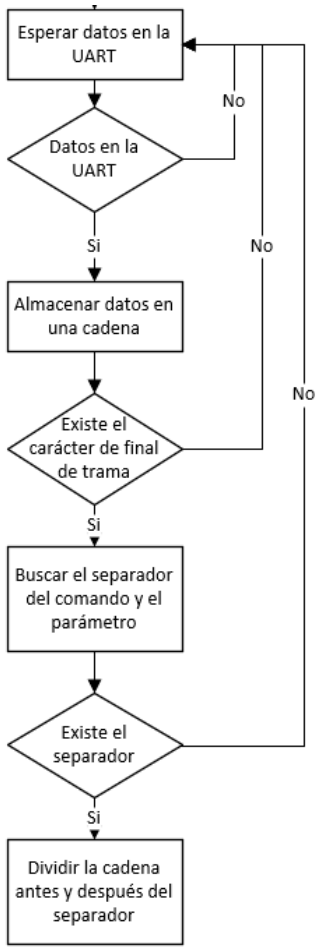
En esta fase del algoritmo es donde interviene el mayor manejo de datos, pues una vez que se tiene almacenada la cadena de texto proveniente del buffer de la UART es necesario verificar si es válida para su procesamiento. Para ello, lo primero que se busca es el carácter de final de trama establecido pues es la indicación para identificar el fin de los datos.

Enseguida, si el final de trama existe se realiza un algoritmo de búsqueda para el separador del comando y del parámetro, es importante decir que ambos caracteres pueden ser cambiados de acuerdo a las necesidades y comprensión del programador.

Una vez que se ha comprobado que el final de trama y el separador existen en la cadena que ha sido enviada por la UART, se cuentan los caracteres totales y el índice (ubicación numérica de los caracteres en la cadena) donde están ubicados el final de trama y el separador para, mediante diferencias de índices, realizar subtramas que serán almacenadas como comando y parámetro respectivamente. El algoritmo en diagrama de flujo de la segunda fase se ve en la Figura 8.

Como se muestra, si no existe en la trama la información requerida, el microcontrolador elimina la trama enviada hasta recibir alguna válida de acuerdo a los parámetros establecidos.

Figura 8
Diagrama de flujo segunda fase



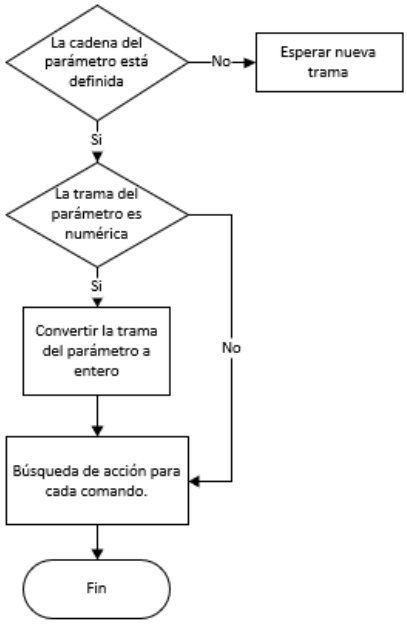
2.2.3 Tercera fase

Cuando ya se tienen divididas las cadenas de texto, éstas se almacenan en dos variables diferentes con la finalidad de realizar un procesamiento distinto para cada una de ellas. En primer lugar, se compara si la cadena recibida es igual a alguna que se ha ingresado como palabra reservada, en caso de ser verdadero, la trama almacenada como parámetro se convierte de variable tipo cadena a tipo entero y se guarda en otro tipo de dato con la finalidad de que pueda utilizarse para realizar operaciones matemáticas o utilizarlas en el código para algún otro fin.

. Así mismo, no todos los comandos deben de tener un parámetro numérico, pueden existir comandos en los que solo se requiera realizar una acción determinada por lo que cualquier parámetro es válido para poder tomar una acción. En caso de que lo almacenado en la variable del comando no se encuentre dentro de la lista definida al principio del programa, la trama se deshecha y el parámetro se elimina para siguiente iteración. El resumen de la tercera parte puede verse en el diagrama flujo de la Figura 9.

Año: 2023 Volumen: 1
Número 9, ISSN: 2448-623X

Figura 9
Diagrama de flujo tercera fase



III. RESULTADOS Y DISCUSIÓN

3.1 Declaración de variables a utilizar.

Con el algoritmo diseñado, se procedió a la implementación para la verificación y validación de su funcionamiento utilizando la plataforma Arduino.

Para comenzar, es necesario definir las variables que se van a utilizar para almacenar los datos provenientes del buffer de la UART, así como los símbolos que se van a utilizar como final de carrera y como separador o limitador. La declaración de estar variables se muestra en el Código Fuente 1.

Código Fuente 1
Declaración de variables

```
//VARIABLES PARA COMUNICACION SERIAL
String Cadena; //Variable para datos del buffer
//Variables de final de línea y separador
char final_cadena = ';';
char delimitador = '=';
//Variable para identificación de comandos
byte NUM_CMD = 15;
boolean FLAG = 0;
int DATA[10]; //Arreglo para almacenar datos de parámetro
                    //DATA[0], DATA[1]  DATA[2]
const String CMD[] = {"MOV1", "MOV2", "MOV3", "MOV4",
"MOV5", "MOV6", "MOV7", "HOME"};
```

En estas declaraciones también se encuentran dos arreglos, el primero (DATA) es de tipo entero, en él se almacenan los datos numéricos de cada uno de los parámetros. El segundo arreglo (CMD) de tipo cadena constante es en donde se colocan las palabras que estarán definidas como parámetros.

La identificación del parámetro que le corresponde a cada uno de los comandos se realiza mediante los índices del arreglo. Es decir, el índice del comando en su arreglo, corresponde también al índice del arreglo del parámetro por lo que es fácil conocer la posición con la que se está trabajando.

Para tener una mejor comprensión y funcionalidad del programa se optó por programar las tareas de cada una de las fases en diferentes funciones de manera que posteriormente se pudiera crear una librería para su uso.

3.2 Función para detección de comandos.

Una vez que se han declarado las variables y se ha iniciado la comunicación serie se comienza con la función del Código Fuente 2 para identificar si la trama contiene un comando disponible. La función es de tipo vacío por lo que no devuelve ningún valor y únicamente se utiliza para realizar tareas determinadas

Código Fuente 2
Función de búsqueda de comandos

```
void Comando_disponible()
{
  //Se almacenan los datos recibidos en Cadena
  Cadena += Read_dato();
  //Si existe el final de trama
  if (FLAG == 1)
  {
    //Se busca si existe un comando válido
    Find_dato(Cadena);
    //Reinicia la búsqueda de final de trama
    FLAG = 0;
    //Se vacía la trama para la siguiente iteración
    Cadena = "";
  }
  delay(5);
}
```

En esta función intervienen otras dos funciones que permiten la búsqueda tanto del final de trama como del separador.

3.3 Función para lectura de datos de la UART.

Como se ha mencionado anteriormente, la función de búsqueda de comandos depende de otras dos funciones, la primera de ella es la que permite realizar la lectura del buffer de la UART y saber si existe el carácter de final de trama.

La función se presenta en el Código Fuente 3 y en este caso corresponde a una función de tipo cadena, por lo que se va a retornar toda la trama que se ha leído del buffer del puerto serie. Para llevar a cabo la búsqueda se utiliza una variable auxiliar que permite almacenar las tramas temporales para ir acumulando los caracteres hasta que se tenga el final de línea. Si el final de línea existe, entonces se activa una bandera para que las demás funciones realicen las tareas determinadas.

Código Fuente 3
Función de lectura de datos.

```
String Read_dato()
{
  //Variable para subcadena
  String Serial_Dato;
  //Variable para conteo de datos en buffer
  int IncomingBytes;
  IncomingBytes  = SerialBT.available();
```

Año: 2023 Volumen: 1
Número 9, ISSN: 2448-623X

```
  //Se limpia la cadena
  Serial_Dato = "";
  //Existen datos en el buffer?
  if (IncomingBytes > 0)
  {
    //Se analiza cada uno de los caracteres
    for (int i = 1 ; i <= IncomingBytes; i++)
    {
      char c = SerialBT.read();
      //Si se detecta el final de cadena
      if (c == final_cadena)
        //Se activa la bandera
        FLAG = 1;
      //Se almacena cada caracter en una cadena
      Serial_Dato += c;
    }
  }
  //Se retorna la cadena completa.
  return Serial_Dato;
}
```

3.4 Función para detección de comandos y acciones.

El último paso del protocolo es conocer si los comandos que se han enviado son válidos y, a partir de la validación, realizar las tareas que se requieran para el programador. En este caso, debido a que se utiliza para el control de servomotores, actualizar la posición de cada actuador que se tenga conectado. En Código Fuente 4 se presenta el código implementado.

Código Fuente 4
Función de detección de comandos

```
void Find_dato(String BUFFER)
{
  //Variable para ancho de trama
  int anchoCadena;
  //Posición del índice
  int PosIndex;
  //Cadenas para separar comando y parámetro
  String Temp_CMD;
  String Temp_Par;
  //Se pregunta el ancho total de la cadena
  anchoCadena = BUFFER.length();
  //Búsqueda del índice del separador
  PosIndex = BUFFER.indexOf(delimitador);
  //La cadena del comando es del inicio hasta el separador
  Temp_CMD = BUFFER.substring(0, PosIndex);
  //La cadena del parámetro es del separador hasta el
final menos el final de trama
  Temp_Par = BUFFER.substring(PosIndex +1,anchoCadena-1);
  //Bucle for para búsqueda de posición del comando en el
arreglo
  for (int i = 0; i <= NUM_CMD; i++)
  {
    //Si el comando se encuentra en el arreglo
    if (Temp_CMD.equals(CMD[i]) == 1)
    {
      //El índice del bucle indica la posición del arreglo
      switch (i)
      {
        //ACCIONES DE ACUERDO A CADA COMANDO
        case 0:
        //Se convierte de cadena a entero el parámetro
          DATA[i] = Temp_Par.toInt();
          //Se puede realizar una acción adicional
          Serial.println("CMD1 RECONOCIDO");
          break;
      }
    }
  }
}
```

El código separa el comando del parámetro de acuerdo a la posición del separador. Posteriormente se verifica si el comando existe en el arreglo que se tiene almacenado y se guarda el índice en el que se encuentra. Si el comando es válido, se convierte la trama de tipo cadena a tipo numérico correspondiente al parámetro en la misma posición del arreglo de datos lo que permite identificar los valores de cada uno de ellos.

3.5 Control de Servomotores utilizando el protocolo.

Para la comprobación y validación del funcionamiento del protocolo se utilizó un ejemplo para el control básico de servomotores que pudiese utilizarse para el control de diversos proyectos de robótica (principal aplicación de los servomotores miniatura). Se contó con 6 servomotores conectados a la placa Arduino y el mismo número de comandos que permitían el control de posición de cada uno de ellos.

El parámetro de cada comando correspondía a un valor numérico de entre 0 a 180 que son los números de grados en que estos motores pueden realizar movimiento. En el Código Fuente 5 se muestra una función donde se utilizan los valores de datos obtenidos de los parámetros para el ajuste del movimiento de los servomotores en los rangos establecidos anteriormente.

Es importante tomar en cuenta que el ejemplo mencionado se utilizó solamente para validar que la información entre el transmisor y el receptor funcionara de forma correcta. Sin embargo, la aplicación del protocolo puede extenderse para otros fines siempre que se respeten las normas que se han descrito a lo largo del desarrollo del proyecto.

Código Fuente 5
Función para movimiento de servomotores mediante el protocolo

```
void adjustMove(int j)
{
  if (DATA[j] > 0 && valor_pos < 0)
    DATA[j] = DATA[j] + valor_pos;
  if (DATA[j] < 180 && valor_pos > 0)
    DATA[j] = DATA[j] + valor_pos;
    printData();
}
```

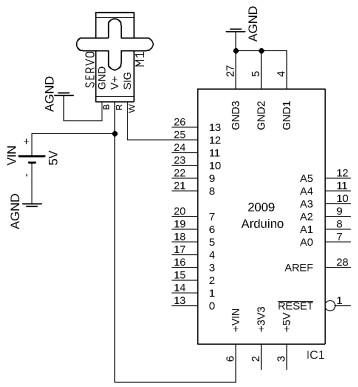
Para realizar la prueba se utilizaron servomotores MG995 como los que se muestra en a Figura 10 que son de los más comunes en proyectos de electrónica y micro robótica.

Figura 10
Servomotor MG995

Año: 2023 Volumen: 1
Número 9, ISSN: 2448-623X

El circuito de conexión utilizado para las pruebas se muestra en la Figura 11. Es necesario colocar una fuente de alimentación externa para el servomotor y unir la línea del común (GND) con la de la placa Arduino. Esto debido a que el servomotor consume más corriente que la entregada por las salidas digitales de la tarjeta. Así mismo, la salida digital utilizada debe de tener característica de modulación por ancho de pulso (PWM) para que pueda generarse el movimiento.

Figura 11
Circuito esquemático para control de servomotor utilizando Arduino.



El protocolo está diseñado para poder mover "n" servomotores, es decir, no se limita a uno solo. Por lo que, de ser necesario, solamente se requiere replicar el circuito para el número de actuadores a utilizar tomando en consideración las salidas PWM del Arduino.

IV. CONCLUSIONES

La comunicación serial de tipo UART es una de las más utilizadas para la transmisión de datos entre microcontroladores y otros tipos de interfaces por su facilidad de configuración e implementación en estos sistemas. Sin embargo, puede complicarse su entendimiento conforme se va incrementando el número de caracteres o números que se quieren analizar, siendo, en muchas ocasiones, el impedimento principal para su aplicación.

Si bien es posible encontrar en bibliografía y en diferentes fuentes en la red códigos de ejemplo para el manejo de grandes tramas de información, muchos de ellos se encuentran diseñados para una aplicación en particular, lo que no permite que pueda adaptarse a diferentes circunstancias que puedan darse. Así mismo, no se encuentra definido de forma escrita el algoritmo que se utilizó para el diseño, siendo complicado replicarlo en otras tareas.

El uso del algoritmo propuesto, abre un área de oportunidad para reducir el tiempo de desarrollo de programas que requieran una transmisión de datos desde la UART. De igual forma, el uso de instrucciones de librerías comunes en la mayoría de lenguajes de programación, lo vuelve versátil para su uso en diferentes plataformas haciendo cambios mínimos a la estructura básica presentada.

Como futuros trabajos, se recomienda hacer una librería que permita sintetizar todas las funciones del algoritmo de una forma más clara y que pueda ser entendible para personas que tengan poco conocimiento en programación. Pues si bien el código se planteó de la forma más sencilla posible, su aplicación en sistemas didácticos puede dificultarse para la mayoría de personas que no cuenten con un curso básico de programación como base.

La aplicación propuesta en el control de servomotores puede darse como una introducción a propuestas de sistemas robóticos que utilicen este tipo de actuadores y que son muy comunes en proyectos didácticos de nivel media superior y superior en carreras afines a la mecatrónica. Sin embargo, también puede extenderse para la creación de sistemas de adquisición de datos al poder enviar una gran cantidad de información de diferentes variables de interés. El hecho de utilizarlo desde la UART no limita su uso a microcontroladores, pues también sistemas de automatización como PLC e inclusive programas de diseño de interfaces como MATLAB, LabVIEW, entre otros, cuentan con este protocolo por defecto

V. AGRADECIMIENTOS

Agradecimientos al Tecnológico Nacional de México (TecNM): Instituto Tecnológico Superior de Lerdo por las facilidades brindadas para la realización del presente trabajo.

También a las divisiones de Sistemas Automotrices, Electrónica por su valiosa ayuda, así como al Departamento de Investigación y Desarrollo Tecnológico.

VI. REFERENCIAS

Artero, Ó. T. (2013). *ARDUINO. Curso práctico de formación.* RC libros.

Bariain, C. (2017). *Programación de Microcontroladores PIC en lenguaje C.* México: Alfaomega.

Corona, L. (2014). *Sensores y actuadores. Aplicaciones con Arduino.* México: Patria.

Coto, A. (2008). *Protocolos y comunicaciones de Red.* México: Cisco Networking Academy.

Díaz, J. M. (2009). *Microcontroladores PIC. Principios y aplicaciones.* México: ASEUC.

Domínguez Mínguez, T. (2020). *Desarrollo de aplicaciones IoT en la nube para Arduino y ESP8266.* España: Marcombo.

López, P. P. (2016). *Robótica y domótica básica con Arduino.* Ra-Ma Editorial.

Méndez de la Torre, R. E. (2022). *Diseño e Implementación de un Módulo de entrenamiento utilizando el procesador ESP32 para aplicaciones enfocadas a la Domótica.* Ecuador: Universidad Politécnica Salesiana.

Pérez, C. (2012). Plataforma embebida multipropósito para comunicación mediante protocolo MIL-STD. *CASE*, 237-242.

Reyes Cortés, F. (2015). *ARDUINO - Aplicaciones en Robótica, Mecatrónica e Ingenierías.* México: Alfaomega.