

# Replicador de TIC-TAC-TOE

Alejandra Castrillo Muñoz, Crisptofer Fernández Fernández, Bryan Masís Mora  
 ale.castrillom97@gmail.com, alberto12ff@gmail.com, bryanmasis87@gmail.com

Área académica de Ingeniería en Computadores

Tecnológico de Costa Rica

**Resumen**—Este documento busca servir como prueba de concepto acerca de la implementación de drivers en un sistema operativo basado en Linux. Esto con el objetivo de brindar al SO la habilidad de manipular dispositivos que son ajenos a los que él maneja por defecto.

**Palabras clave**—driver, meArm, linux, Ubuntu, arduino, react

## I. INTRODUCCIÓN

Actualmente, cualquier persona no especializada en desarrollo de software o afines a la informática puede utilizar un ordenador y la mayoría sus periféricos de forma sencilla e intuitiva, al contrario que en los primeros años de desarrollo del computador, pero esto no implica que la complejidad de dichos dispositivos haya disminuido con el tiempo, si no al contrario, ha incrementado muchas veces con respecto a los primeros computadores.

Este comportamiento es posible gracias a integración de los sistemas operativos modernos con las máquinas, los cuales brindan una capa de abstracción de todos los sistemas de software y hardware que conforman al computador, mostrándole al usuario una interfaz amigable y un conjunto de aplicaciones para interactuar con el sistema. Dentro de las funciones del sistema operativo, como se mencionó anteriormente, es la gestión de los dispositivos periféricos o de entrada-salida (I/O) y en este apartado es donde los controladores (Drivers) tienen un papel importante en el sistema.

Según [1] un controlador de dispositivo es un componente de software que provee una interfaz entre el sistema operativo y el dispositivo de hardware. Este componente administra y configura el dispositivo y además le da un formato legible a la información para que el dispositivo de hardware pueda comprenderla y viceversa. Los controladores dependen de tres interfaces:

- **Interfaz entre el controlador y el kernel:** Para comunicar solicitudes y acceder a los servicios del sistema operativo.
- **La interfaz entre el controlador y el dispositivo:** Permite la ejecución de operaciones.
- **Interfaz entre el controlador y el bus de comunicación:** Para administrar la comunicación entre el controlador y el dispositivo.

EL proyecto actual centra su objetivo general en el desarrollo de un sistema en el cual existen 3 capas principales, el cliente, el servidor y el dispositivo de hardware. De forma

más específica, el cliente es una aplicación móvil en la cual se le permite al usuario jugar el clásico juego de TIC-TAC-TOE, esto contra el computador u otra persona de forma local en el mismo dispositivo, este cliente le envía la información al servidor del estado del juego para que este mediante una biblioteca y el controlador, le envíe instrucciones al dispositivo de hardware el cual es un brazo robótico, este se encargará de replicar lo que hay en la matriz de juego en el cliente, a papel físico, es decir, traducir lo digital a lo real. Para este documento se espera exponer el proceso de desarrollo del proyecto, exponiendo las herramientas utilizadas para el desarrollo del mismo, así como los detalles de implementación del software y del hardware. Se describen los atributos relacionados al proyecto, es decir, como su desarrollo e implementación refuerzan las capacidades de un ingeniero en Computadores. Como parte del desarrollo del sistema, también se muestran los diagramas correspondientes a los sistemas de software y hardware. Se espera que con la finalización del documento se brinde al lector los resultados y conclusiones del desarrollo del proyecto, además de las herramientas necesarias para poder ejecutarlo y también sugerencias y recomendaciones para iteraciones futuras.

## II. AMBIENTE DE DESARROLLO

### II-A. Cliente

Para el desarrollo del cliente, se utilizó React Native, esto por debido a la versatilidad que el lenguaje ofrece, permite la compilación tanto para Android como iOS con el mismo código, además; tiene una sintaxis simple y no representa mucho desafío entender el significado de los componentes. Otra característica que justifica la escogencia de la biblioteca para el proyecto, es que existe una cantidad importante de información y ejemplos de todo, entonces es más fácil solucionar problemas, así como información sobre componentes y/o implementaciones que no se habían hecho con anterioridad.

### II-B. Servidor

El servidor fue implementado en el lenguaje de programación C. Es un sistema basado en sockets pero utiliza consultas del protocolo Http 1.1, con el método POST; el servidor se encargará de recibir los mensajes del estado de la matriz de juego en el cliente, procesará esta información para llamar a la biblioteca que se comunica con el driver para realizar la comunicación con el microcontrolador del dispositivo de hardware.

### II-C. Biblioteca

La biblioteca para el uso del driver y control del brazo se desarrollo en el lenguaje de programación C.

### II-D. Driver

Se desarrolló un driver en el sistema operativo Linux, Ubuntu 16.04 LTS para el control del brazo. El driver fue desarrollado en lenguaje de programación C.

### II-E. Dispositivo de hardware

Para el dispositivo de hardware se utiliza un brazo robot basado en el modelo de MeArm en su versión 0.4, este es movido por 4 servomotores, el brazo es controlado por el microcontrolador ATmega328P, el cual tiene el programa de software encargado de ejecutar la rutinas desarrolladas para duplicar el juego del cliente al espacio físico real. Es alimentado por una fuente de corriente directa de 5V, con un amperaje máximo de 10A.

### II-F. Cálculo de potencia

El consumo total del brazo está determinado por el número de servos que lo mueven, son cuatro servos, los cuales consumen 800mA de corriente cuando inician y cuando se encuentran en funcionamiento, lo cual nos da una corriente total neta de operación de:

$$4I_{servo} = 3,2A \text{ $}$$

Además, este arreglo requiere de 5 voltios para ser alimentado, por lo tanto podemos obtener la potencia consumida por el brazo usando la fórmula:

$$V_{total}I_{total} = 19,2W$$

Se puede observar que tiene un consumo de potencia bastante notable para ser un dispositivo de dimensiones pequeñas.

### II-G. Herramientas de ingeniería

Se utilizó la programación imperativa en la mayor parte del componente de software y programación orientada a objetos. Se realizó el calculo de consumo del brazo mediante la suma de corrientes obtenidas de las hojas de datos de los servomotores. Se requirieron conocimientos básicos en electrónica digital y analógica para ensamblar el brazo robótico y conocimientos intermedios en protocolos de comunicación y lenguaje de bajo nivel.

### II-H. Aprendizaje continuo

Se utilizó el modelo cascada para el desarrollo del proyecto, se debe reforzar la estrategia o utilizar metodologías ágiles como scrum. Se fortaleció el trabajo en equipo por la colaboración en el desarrollo de todas las partes. Se debe fortalecer las habilidades en programación en C y la comprensión de los protocolos utilizados para el desarrollo de controladores de dispositivos.

### II-I. Ingeniería y sociedad

La comprensión y desarrollo de drivers pueden afectar tanto positiva como negativamente a los computadores, por lo cual el desarrollo de controladores de calidad fomenta la seguridad de los dispositivos y evita la perdida de datos y la intrusión a los dispositivos sin permiso. Además, desarrollar drivers permite realizar dispositivos de hardware para resolver cualquier deficiencia o problema en la sociedad y poder controlarlos efectivamente mediante un computador.

## III. DETALLES DE DISEÑO

En la figura 1 puede observarse el diagrama de bloques del sistema donde se muestra la implementación de las capas, donde la capa cliente se aloja en un dispositivo móvil y corresponde a la capa de mayor nivel, luego la capa del servidor que se encarga de la comunicación del cliente con el brazo a través de una biblioteca y el controlador y por ultimo la capa de menor nivel que es el brazo mecánico, controlado por un microcontrolador ATmega328P. En la figura 2 se observa de forma detallada el circuito que implementa el brazo, el cual consta de 4 servomotores de 180 grados, una fuente de alimentación y el arduino con el microcontrolador mencionado anteriormente, este brazo utiliza la biblioteca MeArm para facilitar el mapeo el espacio físico.

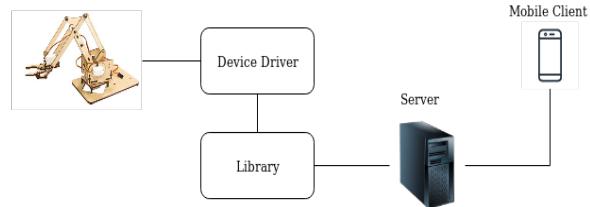


Figura 1. Diagrama de bloques del sistema

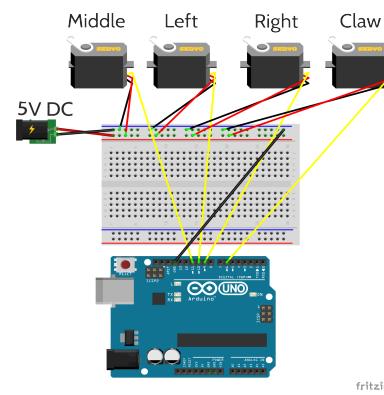


Figura 2. Diagrama del circuito del brazo

En caso del driver se decidió implementar a nivel de kernel debido a los permisos que este requiere para funcionar. Se realizó un driver de memoria con las operaciones comunes (open, close, write, read). Estas operaciones acceden al archivo tty del arduino llamando sus propias operaciones (file\_operations) por medio de las cuales se comunica el driver con el dispositivo[2]. Este driver está ligado a un archivo de

memoria llamado 'myarmdriver', creado para este propuesto, al cuál se le pueden ejecutar las operaciones desarrolladas en el driver como a cualquier otro archivo. Las operaciones del driver se encargan del acceso este archivo de memoria por medio de las funciones definidas dentro de este. Además de las funciones del archivo este driver contiene las funciones que se encargan de la escritura y lectura en el archivo tty del arduino, estas funciones son llamadas desde los métodos write y open de 'myarmdriver'.

#### IV. MANUAL DE INSTRUCCIÓN

Una vez instalada la aplicación, procedemos a abrirla. Al abrir la aplicación, tendremos esta pantalla:



Figura 3. Pantalla principal

Al presionar ese botón, nos llevará a la pantalla de selección, acá podremos escoger el sprite que usará cada jugador:



Figura 4. Configuración del juego

Cabe destacar que si se trata de escoger el mismo sprite para ambos jugadores, el juego avisará sobre este comportamiento e impedirá jugar hasta que se escojan diferentes. Al seleccionar los sprites de forma válida, nos guiarán a la siguiente pantalla que es la pantalla de juego.

Sin embargo, hay que colocar nuestro brazo robótico de forma tal que pueda dibujar, para esto, coloque el brazo a alrededor de 3cm de la base del brazo, esto con el objetivo de que calcen las coordenadas de la matriz del juego (incluida en el paquete) y las escritas dentro del código del robot, debe quedar más o menos así:

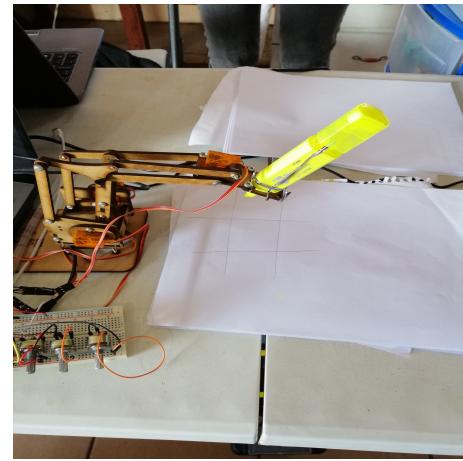


Figura 5. Ubicación del brazo.

Recordar que se debe alimentar nuestro brazo con el cargador incluido en el paquete. Ahora sí, es momento de jugar. La pantalla de juego se puede ver a continuación:

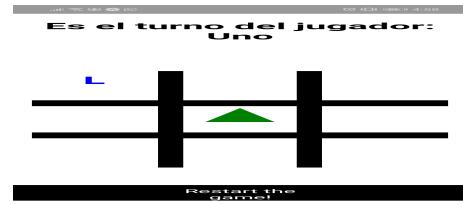


Figura 6. Pantalla de juego

La mecánica del juego es simple, en la pantalla se muestra cual de los dos jugadores tiene el turno actual, este jugador debe tocar cualquiera de las posiciones disponibles (que no han sido marcadas en el momento), al tocarlas, estas registrarán su carácter y le cederá el turno al siguiente jugador. En ese momento, el brazo procederá a pintar el sprite en la casilla seleccionada.

Cabe destacar que el juego puede ser reiniciado en cualquier momento mediante el uso del botón que se encuentra debajo del tablero de juego. Al presionarlo el juego nos dará una advertencia de que debemos cambiar el papel del brazo para dar inicio a una nueva partida. A continuación se puede observar el mensaje de alerta:

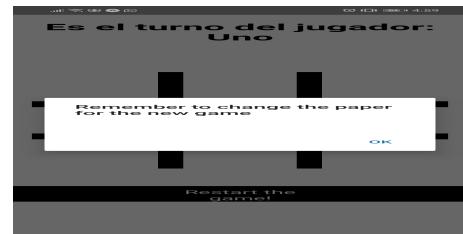


Figura 7. Mensaje de alerta

#### V. COEVALUACIÓN

Basados en la escala del 1 al 10 donde 10 es la nota más alta, puede observarse en los cuadros I, II las notas asignadas

en la coevaluación, estas porque existió un clima de trabajo de respeto, apoyo y dedicación al trabajo entre los compañeros del grupo, además el líder siempre ejerció el cargo de forma responsable, objetiva y con las cualidades de un líder nato.

Cuadro I  
EVALUACIÓN DE SURICATOS A JEFE.

Evaluadores	Bryan
Alejandra	10
Crisptofer	10

Cuadro II  
EVALUACIÓN DE JEFE A SURICATOS.

Evaluador	Bryan
Alejandra	10
Crisptofer	10

## VI. INSTRUCCIONES DEL PROYECTO

El código necesario para hacer uso de este proyecto, se encuentra en nuestro repositorio de github:

[https://github.com/SistOperativos/Proyecto\\_2](https://github.com/SistOperativos/Proyecto_2)

Cuando se abre el enlace, se debe buscar el botón que dice “clonar o descargar”, el cual mostrará el siguiente recuadro:



Figura 8. Opciones de descarga de Github

La opción que debemos escoger es “Descargar Zip”. Cabe destacar que debe asegurarse de que cumple todos los requisitos mencionados en la sección de Ambiente de desarrollo. Una vez descargados los archivos y asegurados los requisitos, se debe descomprimir, esto se puede hacer mediante programas de terceros como WinRAR, etc. Dentro de los archivos descargados se encuentra un archivo .apk, este archivo se debe copiar al teléfono celular con sistema android e instalarlo, ya que este contiene el juego. Para realizar la instalación basta con buscar el .apk desde el explorador de archivos del teléfono, presionarlo y pedirle que instale.

Una vez que se cuente con la aplicación se puede proceder a instalar el driver en la computadora. Para la ejecución del driver primero se debe crear el archivo sobre el cual se ejecutará el driver, esto se realiza con el siguiente comando:

```
sudo mknod /dev/myarmdriver c 0 0
```

Luego de esto se debe abrir la terminal en la carpeta del driver la cual se encuentra en el folder del proyecto con el nombre

de ‘Driver’. Estando en la terminal lo primero que se debe realizar es compilar el driver y generar los archivos necesarios para instalarlo en la computadora, para esto se debe ejecutar el siguiente comando:

```
make
```

Cuando se hayan generado los archivos necesarios se puede pasar a instalar el driver en la computadora con el comando:

```
make install
```

Para verificar que se haya instalado el driver se puede ejecutar el siguiente comando que muestra una lista de los drivers registrados:

```
lsmod
```

En caso de querer remover el driver se debe ejecutar el siguiente comando:

```
make uninstall
```

Ahora para permitir que el servidor pueda utilizar el driver se deben cambiar los permisos del archivo ‘myarmdriver’, para esto se ejecuta este comando:

```
make permission
```

Lo siguiente, es generar los archivos necesarios y poner a correr el server, esto se hace mediante el makefile. Para usar este archivo, necesitamos la terminal de nuestro sistema (Ubuntu), por lo tanto debemos acceder a la carpeta donde tenemos todos los archivos descomprimidos y ejecutar el siguiente comando:

```
make build
```

Una vez terminado este proceso, se utiliza el siguiente comando:

```
make run
```

Una vez terminado esto, el servidor estará corriendo y podremos jugar desde nuestro celular.

## VII. BITÁCORA

Cuadro III  
BITÁCORA CRISPTOFER.

Fecha	Actividad	Duración
12/11/19	Investigación para la implementación del servidor basado en sockets y http 1.1	2
13/11/19	Inicio de implementación del servidor.	3
15/11/19	Finalización del servidor y pruebas del brazo robótico con el equipo.	4
16/11/19	Compra de algunos componentes	2
19/11/19	Finalización de la documentación e integración del sistema.	12
Total		23 horas

Cuadro IV  
BITÁCORA BRYAN.

Fecha	Actividad	Duración(h)
27/10/19	Compresión del proyecto	4
30/10/19	División de tareas	1
2/10/19	Investigación sobre brazos robóticos	3
3/10/19	Investigación sobre modelos para corte láser de brazos	2
7/10/19	Inicio del ensamblamiento del brazo	4
8/10/19	Continuación del ensamblamiento del brazo	2
10/11/19	Planeación de la aplicación móvil para el juego	3
11/11/19	Inicio del desarrollo de la aplicación	5
13/11/19	Inicio de la programación del brazo	5
15/11/19	Mapeo del brazo a las celdas del juego	7
16/11/19	Continuación del desarrollo del app	5
17/11/19	Continuación del mapeo del brazo	3
18/11/19	Desarrollo de app. Inicio de la integración con server	8
19/11/19	Finalización del App/ Mapeo del brazo	15
20/11/19	Finalización de la documentación	5
<b>TOTAL</b>		<b>72</b>

Cuadro V  
BITÁCORA ALEJANDRA.

Fecha	Actividad	Duración(h)
29/10/19	Compresión del proyecto.	2
30/10/19	División de tareas.	1
3/10/19	Investigación sobre drivers en linux. [3]	3
4/10/19	Investigación sobre drivers en linux. [4]	1
5/10/19	Investigación sobre drivers en linux. [2]	2
8/11/19	Implementación de driver de memoria en modo usuario.	1
8/11/19	Implementación de driver de memoria en modo kernel.	1
9/11/19	Investigación sobre drivers de arduino.	2
13/11/19	Investigación sobre drivers con acceso a GPIOs.	1
13/11/19	Investigación sobre comunicación entre driver y arduino.	2
16/11/19	Implementación de driver de memoria con comunicación con arduino.	2
18/11/19	Implementación de driver de memoria con comunicación con arduino.	6
19/11/19	Implementación de driver de memoria con comunicación con arduino.	2
19/11/19	Implementación de biblioteca para driver.	1
19/11/19	Conexión de server con biblioteca.	1
19/11/19	Conexión de todo el sistema.	2
<b>TOTAL</b>		<b>30</b>

### VIII. CONCLUSIONES

Se concluye que la implementación del servidor puede ser más sencilla si se implementa un lenguaje o framework enfocado a esta función, dado que al generar consultas complejas, el manejo de los datos se vuelve complejo y difícil.

Utilizar react-native para el desarrollo de la aplicación cliente de este proyecto (y cualquier proyecto móvil en general), facilitó la implementación de la misma y brindó soporte multiplataforma (Android e iOS) lo cual es valioso.

El elevado consumo de corriente de los servos los hace difíciles de controlar en sistemas de recursos limitados (por ejemplo, que si el sistema dependiera únicamente de la energía suministrada por el arduino). Por esto es importante utilizar

fuentes de alimentación externa para alimentarlos de manera eficiente.

El conocimiento en desarrollo de drivers de la computadora aparte de permitir al usuario poder realizar la comunicación y el manejo de cualquier dispositivo con la computadora también permite tener una visión clara de los riesgos y los posibles software maliciosos que podrían llevar consigo los drivers de fuentes no confiables.

### IX. SUGERENCIAS Y RECOMENDACIONES

Se sugiere usar acrílico y tornillos con tuerca para la estructura del brazo, dado que por desgaste, el MDF se deteriora y la estructura queda muy floja. Se recomienda usar una biblioteca distinta a MeArm porque esta carece de documentación suficiente y esto dificulta su uso. Se sugiere realizar el driver en modo kernel y no usuario porque este posee más permisos. Se recomienda utilizar otro lenguaje de programación para realizar el servidor, dado que C es muy poco flexible. Se recomienda la utilización de React Native para el desarrollo móvil, esto por su capacidad de optimización y también porque permite migrar el programa a múltiples plataformas. Se recomienda trabajar con drivers en modo kernel debido a los permisos y comandos a los que tienen acceso el driver.

### REFERENCIAS

- [1] A. Kadav and M. M. Swift, "Understanding modern device drivers," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1, pp. 87–98, 2012.
- [2] (2015) Linux driver. [Online]. Available: <https://forum.arduino.cc/index.php?topic=348538.0>
- [3] D. Clinton. How to load or unload a linux kernel module. [Online]. Available: <https://opensource.com/article/18/5/how-load-or-unload-linux-kernel-module>
- [4] P. J. Salzman. The linux kernel module programming guide. [Online]. Available: <https://www.tldp.org/LDP/lkmpg/2.6/html/>