Proyecto 2

Pablo Arias Mora Rodrigo López González
Kenneth Fernández Esquivel

23 Noviembre

1 Introducción

El sistema a realizar para el segundo proyecto del curso de Sistemas Operativos de la Carrera de Ingeniería en Computadores es realizar la implementación de un device driver de un dispositivo creado por el estudiante utilizando el lenguaje C para un Sistema Operativo GNU/Linux. Se requiere de la creación de un robot, el cual tenga como objetivo principal la automatización de pruebas físicas sobre un celular o tableta, la cual posea un tipo de pantalla capacitiva.
El robot se encargará de interactuar con la pantalla emulando el comportamiento de una persona. Dicho comportamiento se debe basar en tres tipos de interacción:

- Tocar: Esta interacción consiste en solo tocar la pantalla y soltar.
- Presionar: Esta interacción consiste en dejar presionada la pantalla por un periodo de tiempo especificado.
- Arrastrar: Consiste en presionar y moverse en cualquiera de los ejes X y Y.

Se debe crear un software de prueba para el dispositivo móvil en el que se muestre un teclado numérico. El software seleccionará un PIN aleatorio y el robot deberá de introducir este PIN aleatorio para pasar la prueba. El programa debe adaptarse al tamaño de la pantalla.

2 Descripción del proyecto

Se debe de crear un driver que permita el control de un robot el cual controle una aplicación de desbloqueo de un teléfono

2.1 Requerimientos Técnicos

2.1.1 Dispositivo (Robot)

El objetivo principal del dispositivo es la automatización de pruebas físicas sobre un celular o tableta, la cual posea un tipo de pantalla capacitiva. El robot se encargara de interactuar con la pantalla emulando el comportamiento de una persona (ver sección de Tipos de Interacción)

2.1.2 Tipos de Interacción

- Tocar: Esta interacción consiste en solo tocar la pantalla y soltar.
- Presionar: Esta interacción consiste en dejar presionada la pantalla por un periodo de tiempo especificado.
- Arrastrar: Consiste en presionar y moverse en cualquiera de los ejes X y Y.

2.1.3 Software de prueba para el dispositivo móvil

Se deberá de crear un software el cual muestre un teclado numérico. El software seleccionara un PIN aleatorio y el robot deberá de introducir este PIN aleatorio, para pasar la prueba. El programa debe adaptar al tamaño de la pantalla.

2.1.4 Interfaz con la computadora

La interfaz física la deberá de decidir el estudiante, puede utilizar entre: USB, puerto Paralelo o puerto de Comunicaciones (COM).

2.1.5 Device Driver

Se deberá de construir un device driver en el lenguaje C, que funcione sobre el SO que aparece en el enunciado. Este device driver se encargara de proporcionar al usuario una serie de primitivas que le permita la interacción con el robot.

2.1.6 Biblioteca

Se deberá de crear una biblioteca de Python que permita la implementación de las funciones provistas por el device driver.

2.1.7 Lenguaje

Se deberá de describir un lenguaje común que permita la descripción de cualquier tipo de interacción. En este lenguaje se tendrán conceptos como (no limitado a):

- Tipo de interacción.
- Posición de X y Y inicial.
- Posición de X y Y.

2.1.8 Interprete

Se deberá de escribir un programa que permita la implementación del pequeño lenguaje y que sirva para la programación del robot.

2.1.9 Otras consideraciones

Además de las definiciones anteriores tome en cuenta:

- El robot debe adaptarse a distintas pantallas.
- La resolución mínima del robot deber 1cm x 1cm, lo cual permite partir la pantalla en matrices de 1cm x 1cm.
- La interacción debe ser lo suficientemente rápida como para permitir un comportamiento fluido.
- Debe elegir una licencia para la documentación y el código de la tarea.

2.2 Requerimientos Técnicos

- El desarrollo se debe de realizar utilizando el lenguaje de programación C para GNU/Linux.
- Todo el proyecto debe de funcionar utilizando GNU/Linux.

La sintaxis del intérprete es:

```
$ robot -c configuracion
```

Nótese que el cliente debe de recibir los parámetros desde la terminal. El signo de dólar representa el Shell del SO.

3 Ambiente de desarrollo

3.1 Interacción desarrollada

- El hardware utilizado para controlar el dispositivo externo fue un arduino
- La comunicación con la computadora se realizó mediante un USB device driver.
- La computadora se comunica con el USB device driver mediante una biblioteca programada en C
- La biblioteca en C fue integrada con el intérprete de Python.
- Se desarrolló un lenguaje de programación basado en JSON el cual es procesado por Python.

Figure 1: Secuencia Propuesta

3.2 Software utilizado

Geany
Geany es un editor de texto pequeño y ligero basado en Scintilla con características básicas de entorno de desarrollo integrado (IDE). Utiliza bibliotecas GTK para su funcionamiento. Está disponible para distintos sistemas operativos, como GNU/Linux,Mac OS X, BSD, Solaris y Microsoft Windows. Es distribuido como software libre bajo la Licencia Pública General de GNU.
Utilizado principalmente en el desarrollo de código C utilizado en el driver, se utilizaron las librerías USB, module y kernel para la implementación del device driver

Sublime
Sublime Text es un editor de texto y editor de código fuente está escrito en C++ y Python para los plugins. Desarrollado originalmente como una extensión de Vim, con el tiempo fue creando una identidad propia, por esto aún conserva un modo de edición tipo vi llamado Vintage mode.
Se distribuye de forma gratuita, sin embargo no es software libre o de código abierto, se puede obtener una licencia para su uso ilimitado, pero él no disponer de ésta no genera ninguna limitación más allá de una alerta cada cierto tiempo.
Utilizado principalmente en la biblioteca de C, en el intérprete y en el código de prueba.

Android Studio
Android Studio es un entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014. Está basado en el software IntelliJ IDEA de JetBrains, y es publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux.
Utilizado principalmente para el desarrollo del software de prueba en el teléfono.

Arduino
Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. Utilizado para el control lógico de los motores involucrados en el robot.

3.3 Como Utilizar el Dispositivo

Driver
Los comandos para compilar el device driver se encuentran en el archivo Makefile adjunto en la misma carpeta del driver.
Con lo anterior, para compilar se utiliza el comando make.
Una vez compilado, se debe incorporar el driver a los módulos del kernel. Lo anterior se realiza con la instrucción:

- insmod pen_driver.ko

Una vez montado el driver se puede iniciar el envío de datos hacia el dispositivo físico. **Robot de Pruebas**
El diseño del lenguaje a utilizar es el siguiente:
"movements": [

"X": 10,
"Y": 10,
"DRAG": 0

]

En donde se tienen tres datos específicos para cada movimiento, los cuales son:

- X: Posición final a la que se desea mover en X (dado en centímetros).
- Y: Posición final a la que se desea mover en Y (dado en centímetros).
- DRAG: Tipo de interacción: 0 significa presionar y soltar mientras que 1 significa arrastrar.

El mismo debe encontrar dentro de un archivo .ro
El script de Python robot debe ser ejecutado de acuerdo con las especificaciones.

4 Estructuras de datos utilizadas y funciones

Para el device driver se utilizaron las siguientes estructuras:

- File_operations: Struct estático que describe las operaciones sobre el archivo (open, reléase, read y write).
- Usb_device_id: Contiene un USB_DEVICE con la información del idVendor e idProduct del arduino para asociar el driver al dispositivo utilizado.
- Usb_driver: Contiene las características primordiales del driver, tales como el nombre, el probe (función que se ejecuta cada vez q se conecta el dispositivo, una vez con cada endpoint), la función para desconectar y el usb_device_id mencionado anteriormente.

Las funciones principales utilizadas por el device driver son las siguientes:

- Init: Función que se utiliza al insertar el driver en la computadora. Básicamente registra el driver como una instancia del controlador USB.
- Exit: Función que se utiliza al remover el driver en la computadora. Registra la salida del driver de la computadora.
- Probe: Función que se ejecuta cada vez que se inserta el dispositivo, una vez por endpoint. Crea un nuevo dispositivo con una interfaz USB dada para proceder al registro del dispositivo como un dispositivo USB para asignar su minor number apropiado.

- Disconnect: Función utilizada en la extracción del dispositivo de la computadora. En ella se registra la salida del dispositivo del sistema.
- Read: Función que se utiliza para leer datos enviados por el dispositivo. Utiliza el modo de envío de mensajes bulk de USB para verificar el dato recibido, haciendo uso del bulk point in.
- Write: Función que se utiliza para enviar datos al dispositivo. Utiliza también el modo de envío de mensajes bulk de USB para verificar el dato enviado, haciendo uso del bulk point out.

Funciones de la Biblioteca de C y del Intérprete de Python

- Arduino_send_message: Encargado del envío de mensajes para ser interpretados por el protocolo entre USB y Serial.
- • Arduino_start_com: Inicializa el puente de comunicación con el Arduino.

Funciones del código del Arduino:

- El Arduino usa dos loop() y el serialEvent(). Las cuales cumplen con los propósitos por defecto de cada caso.

El loop corre para mover al robot en la dirección indicada por el mensaje del driver. Y el serialEvent va recolectando la data del mensaje byte por byte y dependiendo de lo que llegue así se arma una respuesta para ser corrida por el loop en este caso.

4.1 Diseño

4.1.1 Diseño Hardware

Para el diseño del hardware se usó el siguiente concepto:

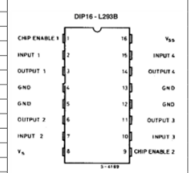
Pin	Nombre	Descripción	Patillaje
1	Chip Enable 1	Habilitación de los canales 1 y 2	
2	Input 1	Entrada del Canal 1	
3	Output 1	Salida del Canal 1	
4	GND	Tierra de Alimentación	
5	GND	Tierra de Alimentación	
6	Output 2	Salida del Canal 2	
7	Input 2	Entrada del Canal 1	
8	Vs	Alimentación de las cargas	
9	Chip Enable 2	Habilitación de los canales 3 y 4	
10	Input 3	Entrada del Canal 3	
11	Output 3	Salida del Canal 3	
12	GND	Tierra de Alimentación	
13	GND	Tierra de Alimentación	
14	Output 4	Salida del Canal 4	
15	Input 4	Entrada del Canal 4	

Figure 2: Descripción de los Pines del L293B

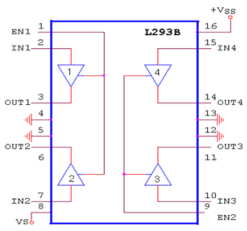


Figure 3: Diagrama de Bloques del L293B

Y donde el circuito resultante o a utilizar es el siguiente:

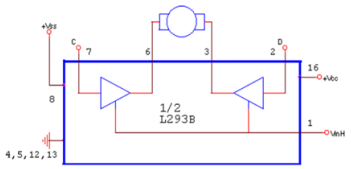


Figure 4: Circuito de control para el doble giro de un motor de corriente continua

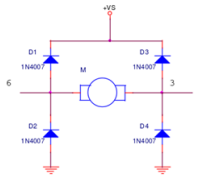


Figure 5: Circuito de protección para el L293 para evitar corrientes inversas al arrancar el motor

En esta caso la tabla de funcionamiento es la siguiente:

V_{inh}	A	B	M
H	L	L	Parada rápida del motor
H	H	H	Parada rápida del motor
H	L	H	Giro a la Izquierda
H	H	L	Giro a la derecha
L	X	X	Motor desconectado, giro libre

Figure 6: Circuito de control para el doble giro de un motor de corriente continua

4.1.2 Diseño Sistema Completo

Se desarrolló un diseño basado en Capas, el mismo permite modificaciones a futuro en caso de que el mismo se quiera readaptar a otros entornos.

Figure 7: Diagrama de Capas

4.2 Instrucciones para ejecutar el programa

Para ejecutar el programa se debe insertar el Arduino a la computadora, desmontar el driver del Arduino con la instrucción rmmod cdc_acm, montar el driver con la instrucción insmod pen_driver.ko y correr el intérprete de Python.

5 Actividades realizadas por estudiante

Rodrigo López:
Jueves 12: Investigar acerca de device drivers. 4 horas
Viernes 13: Implementar driver hello world versión 1. 1 hora
Sábado 14: Implementar driver hello world versión 2. 1 hora
Lunes 16: Implementación device driver uart tty fallida. 5 horas
Martes 17: Otros dos intentos de implementación de device driver uart tty fallida. 8 horas
Miércoles 18: Implementación inicial USB device driver. 2 horas
Jueves 19: Implementación funcional USB device driver (manda basura pero manda algo): 8 horas
Viernes 20: Pruebas conjuntas con lenguaje, intérprete y Arduino (sin dispositivo todavía): 7 horas.
Sábado 21: Inicio documentación: 1 hora
Total: 37 horas

Kenneth Fernández:
Lunes 2: Diseñando posibles opciones para la maqueta. 2 horas.
Jueves 5: Investigando sobre posibles opciones para la maqueta. 2 horas.
Sábado 8: Investigando como implementar el hardware. 3 horas.
Martes 10: Ir a comprar los componentes para el hardware y probar el circuito. 4 horas.
Jueves 12: Seguetear unidad de cd/DVD para usarla en la maqueta. 3 horas.
Viernes 13: Montar la maqueta y pegando las partes. 4 horas. Miércoles 18: Ir a comprar legos y unos diodos de protección. Y probar los cambios 4 horas.
Jueves 19: Montar más partes de la maqueta y modificación de la misma. 3 horas.
Viernes 20: Implementando el software en Arduino. 6 horas.
Domingo 22: Pruebas finales de hardware y maqueta, 5 horas.
Domingo 22: Documentación: 2 horas.
Total: 38 horas

Pablo Arias:
Jueves 12: Investigar de cómo implementar bibliotecas en C dentro de código Python. 8 horas
Viernes 13: Investigar sobre JSON y YAML. 4 horas
Sábado 14: Implementar lenguaje de programación basado en JSON dentro de Python. 4 horas
Miércoles 18: Implementar demo de biblioteca en C dentro de Python. 4 horas
Viernes 20: Diseño del lenguaje de comunicación entre Arduino y driver. 3 horas.
Sábado 21: Pruebas conjuntas con lenguaje, intérprete y Arduino (sin disposi-

tivo todavía): 7 horas.
Domingo 22: Documentación: 2 horas.
Total: 32 horas.

6 Comentarios finales (estado del programa)

El proyecto se entrega a un 100%.
Problemas encontrados: La velocidad de transferencia del USB es mucho mayor a la velocidad de transmisión del puerto serial dentro del Arduino. De esta manera la transferencia de datos de uno con el otro no es la esperada. Por lo que el equipo decidió realizar un protocolo que solucionara esta limitante, el mismo se basa en la cantidad de caracteres enviados, mas no así en el mensaje como tal.

7 Conclusiones

La implementación de un device driver específico para un hardware no es nada sencillo, ya que se debe investigar acerca de cómo funciona específicamente el dispositivo en términos de protocolos de envío, velocidad de envío y medio de comunicación utilizado. Una vez investigado lo anterior se debe adaptar el driver a estas características.
Se recomienda el uso de tutoriales de la red para iniciar con los primeros pasos de implementación del device driver, ya que aunque es posible entender de la literatura, los libros que explican el funcionamiento completo del driver son de extensiones grandes (200 páginas en adelante) por lo que si se tiene una ventana de tiempo limitado en los tutoriales explican lo básico (que en el libro también explican básicamente lo mismo, con algunas añadiduras a ciertos aspectos) y necesario para realizar el esqueleto inicial del driver.
Se recomienda utilizar un dispositivo que se comunique con el mismo protocolo que el driver, ya que si se utiliza por ejemplo un Arduino (que utiliza un driver UART para luego transformarlo a USB mediante un driver uart-to-usb) con un driver USB solamente, no se podrá tener la certeza de que los datos enviados mediante USB le van a llegar bien al Arduino (UART) ya que las velocidades de transmisión y recepción son diferentes, además que los protocolos utilizados pueden diferir también.
Se recomienda también realizar el propio dispositivo controlador de hardware, ya que así se podrá definir tanto el puerto de comunicación como el protocolo en el que se basará la comunicación, eliminando restricciones de operación.

8 Bibliografía

References

derekmolloy.ie,. (2015). Writing a Linux Kernel Module — Part 2: A Character Device — derekmolloy.ie. Retrieved 23 November 2015, from http:derekmolloy.iewriting-a-linux-kernel-module-part-2-a-character-device

DRIVER PUSH-PULL DE 4 CANALES. (2015). Retrieved 23 November 2015, from http:www.todopic.com.arutilesl293b_driver_en_puente.pdf

Es.wikipedia.org,. (2015). Android Studio. Retrieved 23 November 2015, from https://es.wikipedia.org/wiki/Android_Studio

Es.wikipedia.org,. (2015). Arduino. Retrieved 23 November 2015, from https://es.wikipedia.org/wiki/Arduino

Es.wikipedia.org,. (2015). Geany. Retrieved 23 November 2015, from https://es.wikipedia.org/wiki/Geany

Es.wikipedia.org,. (2015). Sublime Text. Retrieved 23 November 2015, from https://es.wikipedia.org/wiki/Sublime_Text

Henson, V. (2015). /dev/hello_world: A Simple Introduction to Device Drivers under Linux O'Reilly Media. Linuxdevcenter.com. Retrieved 23 November 2015, from http://www.linuxdevcenter.com/pub/a/linux/2007/07/05/devhelloworld-asimpleintroductiontodevicedriversunderlinux.html

Pugalia, A., Pugalia, A. (2014). USB Device Drivers Playing with Systems. Sysplay.in. Retrieved 23 November 2015, from https://sysplay.in/blog/tag/usb-devicedrivers/