Instituto Tecnológico de Costa Rica

Principios de Sistemas Operativos

Proyecto 2

Estudiantes:

Michael Varela Suarez

Oscar Montes Fonseca

Melvin Gutierrez Vargas

Introducción

El siguiente proyecto tiene como objetivo principal la construcción de un dispositivo que permita la automatización de pruebas físicas sobre un celular o tableta, cuando éstas le solicitan al usuario el ingreso de un pin numérico. La idea está en simular el proceso de un token en una transacción bancaria suponiendo que la entidad financiera provee una aplicación móvil donde el usuario deberá ingresar su pin de seguridad dado a su computador que tiene acceso al dispositivo para indicarle cual es el pin valido para esa transacción. Entonces el robot se encarga automáticamente del ingreso de este y se logra abstraer por completo al usuario la existencia del pin.

Para el desarrollo de este proyecto se necesita que exista un driver capaz de controlar al dispositivo usando el sistema operativo Linux. Asimismo, de un lenguaje que permita la comunicación entre el driver y una biblioteca de acceso a las principales funciones para el control de este. A continuación, se detallarán las herramientas de desarrollo, el proceso del diseño y la construcción tanto del software como del hardware del dispositivo.

Ambiente de desarrollo

Las herramientas utilizadas para el desarrollo del proyecto fueron las siguientes:

- Python 3
- C
- Arduino 1.6.5
- LabView 2014
- Gedit 3
- USB.h

Instrucciones para ejecutar el programa:

1. Ejecutar el programa writePin.py en terminal mediante:
   $ python writePin.py

2. Digitar el pin deseado en el programa y la configuración "x,y,z" y presionar el boton OK.

3. Ejecutar el programa interpreter.py en terminal mediante:
   $ python interpreter.py

Estructuras de datos usadas y funciones:

- Vectores: Se utilizaron para la representación de los números del teclado del celular. Cada vector equivalía a la posición en la matriz que era una abstracción de la pantalla. Por ejemplo, para el número 8 se tenía el vector (2,1).
- Matriz: Se utilizó para la abstracción del teclado numérico del dispositivo.
- writePin(pin): Esta función recibe el pin de forma de una tupla y por cada número calcula los movimientos básicos que debe realizar el robot para transformarlos en el lenguaje diseñado por la aplicación writePin.py.
- write_msg_bulk (): Esta función es utilizada por el driver del dispositivo USB para poder enviar un "bulk" de información al USB. Para su configuración se utilizó el endpoint del Arduino y el que era específico de tipo bulk. Cabe mencionar que el Arduino presenta dos endpoints de bulk y uno de interrupción.

Diseño

En lo que respecta al diseño de la construcción del programa controlador del robot y la parte del driver junto con el intérprete del lenguaje para manipular el dispositivo se tiene las siguientes etapas de diseño (Fig. 1):

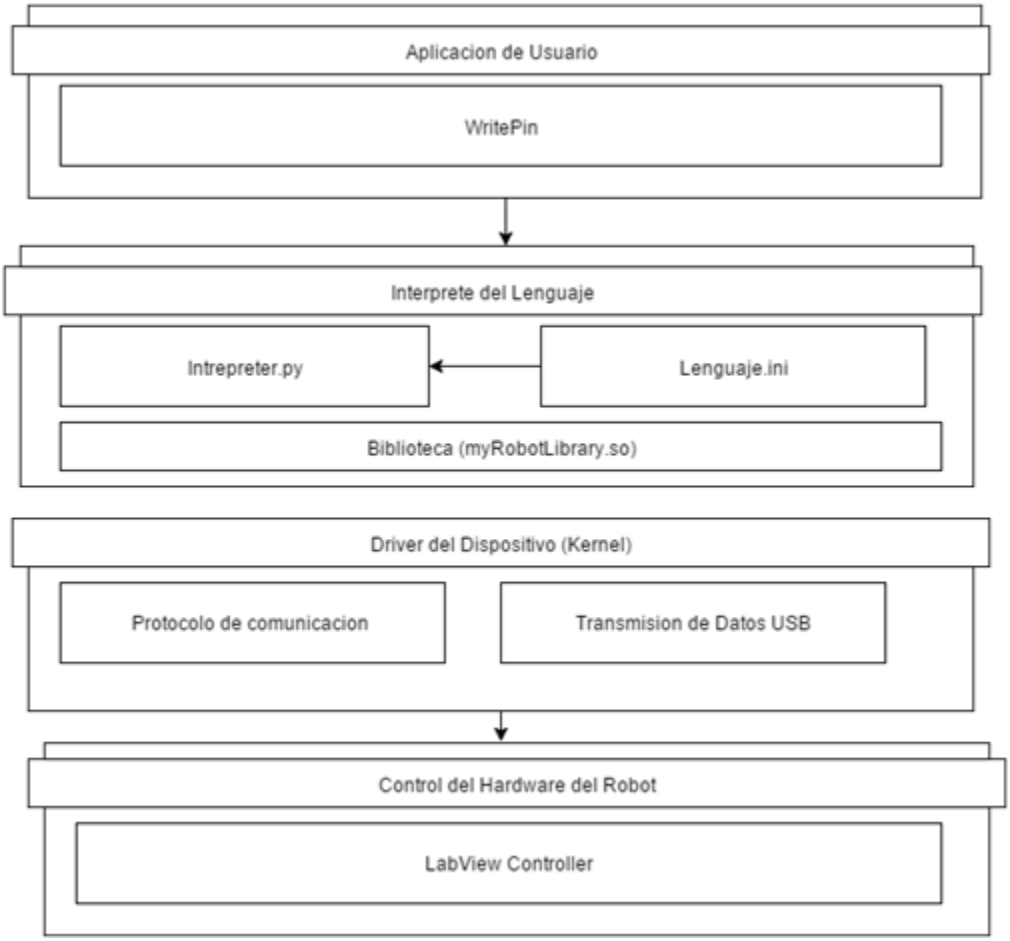

Fig. 1 Diagrama de diseño y arquitectura del programa Physical Test Automation.

WritePin.py

Consiste en la aplicación a nivel de usuario, que le presenta la oportunidad de ingresar un pin para generar el lenguaje que el intérprete deberá proporcionar al driver del dispositivo con la finalidad de que el robot se mueva de acuerdo al pin ingresado. Básicamente se genera un algoritmo que traduce el pin en unas primitivas básicas del lenguaje definido para su posterior uso en el intérprete del lenguaje.

Interprete del Lenguaje

*Lenguaje.ini: El lenguaje para el control del dispositivo se basó en los siguiente:
Movimientos básicos: Down, Up , Left ,Right
Variables de configuración: Adaptación de la pantalla en los ejes X, Y , Z.
Variables de Acción: Presionar(P), Subir (O), Click(C) y Mantener (M)
Variable de control: Fin movimiento (*), Fin Configuracion (&)

*Interpreter.py: Toma el lenguaje.ini definido según lo anterior y lo interpreta utilizando la biblioteca myRobotLibrary.so que se encarga de interactuar con el driver del dispositivo.

*Biblioteca para el control del driver: myRobotLibrary.so se encarga de interactuar con el dispositivo /dev/myRobot el cual es el encargado del driver de este.

Driver del dispositivo

- Protocolo de comunicación: Para poder interactuar con el dispositivo se utilizó un intento del protocolo UART construido a partir del envío de cantidad de caracteres. Entonces por ejemplo: si se quería enviar un movimiento derecha este se asociaba a un número 2. Con esto se logró establecer el protocolo que por decirlo de una manera simple está basado en cantidad de caracteres de tipo "a".

- Transmisión de Datos USB: Para la construcción de este módulo se requirió el apoyo del documento "Writing an USB Driver for linux" el cual explicaba como crear un driver para un dispositivo de llave maya y los conceptos básicos de cualquier dispositivo USB. Para el análisis de este código se puede consultar con más detalle el el siguiente link: http://opensourceforu.efytimes.com/2011/12/data-transfers-to-from-usb-devices/ Asimismo para montar el driver de tipo "char" sobre el de USB para poder codificar la biblioteca se utilizaron ciertas pruebas basadas en un driver "ebbchar" del autor Derek Molloy. Para mas detalle se puede consultar el link: http://derekmolloy.ie/writing-a-linux-kernel-module-part-2-a-character-device/

- Licencia del software: Se seleccionó la licencia de tipo GPL (General Public License) debido a que funciona sin ninguna restricción permitiendo a los usuarios garantizarse que pueden usar el módulo del kernel (LKM) de forma gratuita. Esta alternativa permite que su LKM sea tratado por el kernel sin ninguna advertencia y al ser no apropiativo permite que sea una alternativo "non-tainted". (Derek Molloy).

Control del Hardware

-Comunicación Serial I2C: Para la transmisión de los datos hacia el hardware de LabView se utilizó el envío por medio de I2C usando dos arduinos con una configuración slave-master. El diseño de este circuito se muestra a continuación:

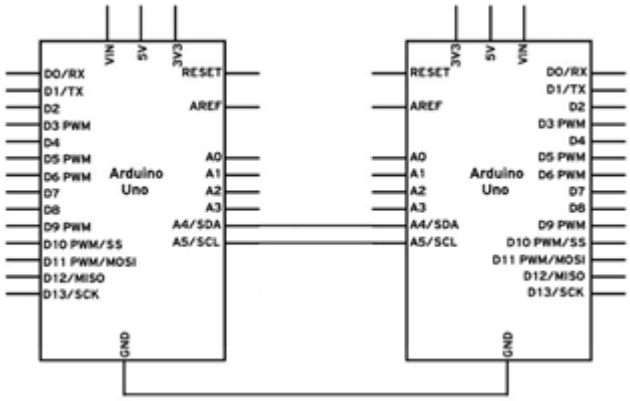

Fig. 2 Circuito para la comunicación I2C de los arduinos.

-LabView Controller: Este módulo se encarga del control absoluto de los motores(Lego) del dispositivo. Le indica la potencia con la cual deben trabajar y la velocidad de rotación de estos.

Actividades realizadas por estudiante:

Actividades realizadas por Melvin:

Fecha	Actividad Realizada	Horas Trabajo
15/11/2015	Se programó una aplicación en Android la cual incluía un teclado numérico	7
18/11/2015	Se programó la aplicación de generación de instrucciones del lenguaje diseñado	2
19/11/2015	Se programó la aplicación de interpretación del lenguaje	2
21/11/2015	Se realizaron pruebas del sistema completo	5
	Total	16

Actividades realizadas por Michael:

Fecha	Actividad Realizada	Horas Trabajo
16/11/2015	Implementación de la primera versión del robot y pruebas	10
17/11/2015	Realización de pruebas mediante el Device Driver	6
18/11/2015	Realización de pruebas mediante el Interprete y el Device Driver	6
19/11/2015	Implementación de la segunda versión del robot y pruebas	12
21/11/2015	Pruebas del robot con el sistema completo implementado	5
	Total	39

Actividades realizadas por Oscar:

Fecha	Actividad Realizada	Horas Trabajo
15/11/2015	Se investigó y se realizaron tutoriales acerca de Device Driver	6
16/11/2015	Se implementó un Device Driver UART y se realizaron pruebas	12
17/11/2015	Se implementó un Device Driver USB y se realizaron pruebas	8
18/11/2015	Se realizaron pruebas en conjunto con el robot	6
21/11/2015	Se realizaron pruebas generales del sistema	5
	Total	37

Comentarios finales (estado del programa):

Se implementó un programa en Android el cual se ejecuta en un teléfono inteligente, donde este programa genera códigos aleatorios de 4 dígitos, además mediante una interfaz gráfica es posible digitar estos valores.

Se creó un Robot mediante la utilización de Legos y controlado mediante el software LabView, el cual es capaz de digitar números en la aplicación Android.

Además, se implementó un programa escrito en Python el cual genera un conjunto de instrucciones de control para el Robot, estas instrucciones son generadas a partir de un código de 4 dígitos dado al programa, por último, se implementó un programa escrito en Python el cual interpreta estas instrucciones y envía al Robot la información y acciones que debe realizar.

Conclusiones

- Todo driver de USB en linux necesita de una capa vertical que puede ser de tipo "char" o "block" para interactuar con un programa en el espacio de usuario.
- Los prints en el espacio de kernel se tratan de manera distinta a un printf en el espacio de usuario, haciendo que sea necesario la revisión de logs en el kernel.
- Los endpoints de un dispositivo USB determinan si son de salida o de entrada y si les pueden enviar un paquete de información de tipo "bulk" o "interrupt".
- Todo dispositivo en linux es tratado como un archivo y por lo tanto se debe crear un espacio en /dev/miDispositivo.
- Para proveer a una aplicación de espacio de usuario la interacción con el archivo descriptor del driver se necesita implementar los métodos de una estructura de operaciones de archivos de información. Estas funciones son las primitivas básicas: open, read, write, release entre otras.
- Cuando se realiza una comunicación a través de USB con un dispositivo todo información debe pasar por los endpoints definidos según el fabricante de este. En esta se detalla su tamaño de paquete, si es de entrada o salida y el tiempo de "timeout" máximo.

Recomendaciones

- Cuando se trabaja a nivel de Kernel una aplicación se recomienda la verificación constante de los logs que se encuentran en /sys/var específicamente el kern.log.
- Se recomienda evitar el uso de printk en un driver para cuando ya se está seguro de su funcionamiento, esto debido a que pueden ocasionar errores de tipo "heisenbug" en los cambios de contexto y pueden afectar el envío de la información.
- Cuando se verifica si un driver de usb está instalado con el LKM del fabricante por medio de cd /dev/sys/log y cat devices se recomienda que el LKM propio que se desea montar aun no este activo. Con esto se evita que el driver del dispositivo analizado en cuestión no se vuelva a cargar de forma automática y se logre hacerle un rmmod para poder instalar el propio sin problemas.
- Para la compilación de las bibliotecas de python se recomienda el uso de estas en mayúscula y para su ejecución por medio de la terminal se recomienda solo el uso de python y no python3.5 para lo que sería esta aplicación.
- En el momento de probar un driver de tipo "USB" se recomienda estar muy seguro de los endpoints e interfaces con las que trabaja el dispositivo. Por ejemplo: el arduino presenta dos interfaces y solamente la segunda es la que utiliza el hardware para la comunicación Usb-Serial.

Bibliografía

[1] Molloy, D. (s.f.). derekmolloy.ie. Consultado el 20 de Noviembre de 2015, de http://derekmolloy.ie/writing-a-linux-kernel-module-part-1-introduction/

[2] Kumar, A. (s.f.). OpenSource. Consultado el 16 de Noviembre de 2015, de http://opensourceforu.efytimes.com/2011/12/data-transfers-to-from-usb-devices/