OPERATIVES SYSTEMS

INSTITUTO TECNOLÓGICO DE COSTA RICA

COMPUTER ENGINEERING ACADEMIC AREA

Robotic Finger

Authors:
José Daniel Badilla Umaña
Luis Alonso Barboza Artavia
Jason Josue Leitón Jiménez
Carlos Peralta Coto

Date: November 25, 2017

1 Introduction

When it comes to computers, it refers to both software and hardware, but the technology is so advanced that using a system has become such a simple task that users do not realize what is happening with respect to the communication that must be between the physical device (hardware) and the program with which the user interacts (software).

It is important to mention that whenever there is interaction between a physical device and a program, there is a driver that allows communication between both parts as well as with the rest of the system. In general, without the drivers you could not print, listen to music, use mouse and keyboard among others.

The drivers are special programs that allow the operating system communicates with the parts of the computer, which contain the messages to their destination. They are messengers without which the operating system could not do almost anything. The printer, for example, is responsible for putting on paper everything that is seen on the screen of Word or Firefox, but OS does not know anything about how much ink there is what to use or where to put it. When ordering the printing, it is the driver who takes note and communicates with the printer. In other words, the driver gives instructions to the operating system, about how certain hardware should work and how the system should work together to provide the best results.

Device drivers can be defined as system files that translate the complex commands issued by people, the operating system, and programs into simpler instructions that the hardware components (“devices”) of the PC can actually understand and execute.

The applications talk with the core of the system which is represented by the yellow strip figure 1, who in turn passes the commands to the hardware through the device drivers. Everything is done through standard calls, or API.

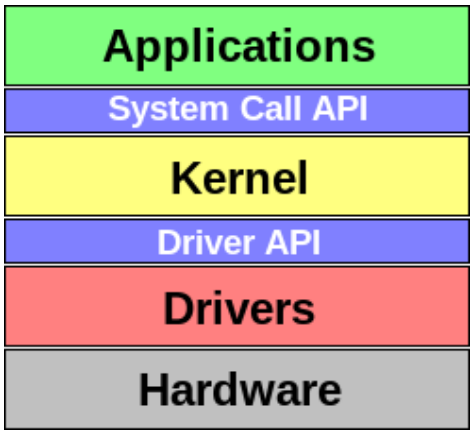

Figure 1: System Layers

Not all devices need driver installation. Those that are essential for the operation of a PC are included in the factory; This is the case of processors, disks and memories. When the device has an essential part and another does not, the support included by the system is basic; for example, when there are no drivers, the graphics cards show the windows, but not the special effects. It is as if the driver had only one basic dictionary available.

Figure 2 shows the different layers that exist from the highest level program to the hardware or device (low level), it is important to mention that you should not confuse driver with firmware; the second is inside the gray zone, because it is hardware.

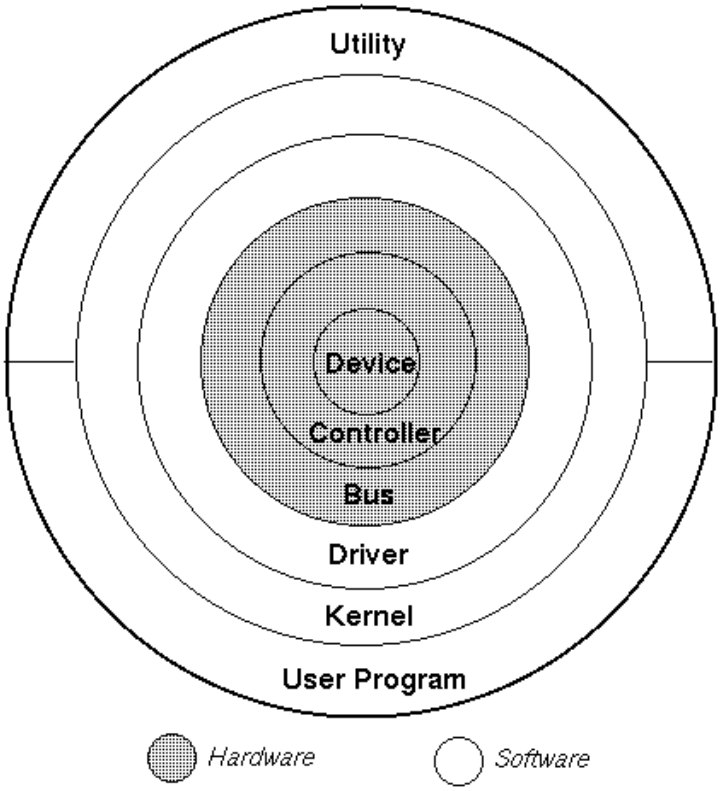

Figure 2: System Modules

Drivers make work at such a basic level (low level) that the smallest error can destabilize the system. For example, if the system asks the driver of the graphic card to change the size of the screen and the driver does not establish communication with the card, a critical error will appear, which can render the computer useless.

To make a device usable there must be a driver present for it. So let us understand how an application accesses data from a device with the help of a driver. We will discuss the following four major entities, these major are based in figure 1.

- User-space application: This can be any simple utility like echo, or any complex

application.

- Device file: This is a special file that provides an interface for the driver. It is present in the file system as an ordinary file. The application can perform all supported operation on it, just like for an ordinary file. It can move, copy, delete, rename, read and write these device files.
- Device driver: This is the software interface for the device and resides in the kernel space.
- Device: This can be the actual device present at the hardware level, or a pseudo device.

2 Development environment

To implement this project we used some tools that made easier the work. Some of the tools in our environment are:

- Sublime Text: very popular text editor that supports multiple programming languages. Some languages that can be used in this editor are: Python, Java, C, C++, PHP, ASM, and more.
- GNU C/C++ Compiler: most important compiler for languages such as C and C++. In this project it was used to generate the necessary executables from the .c and .h files.
- Arduino: open source platform which has the objective to bring an cheap option to learn about computing science.
- Android Studio: program used to create an Android App. It has the option to debug using a tablet or a virtual device to check the status of the application.
- Makefiles: files that have a command list that will be executed in order to compile the files automatic.

3 Continuous learning attributes analysis

For the resolution of this project, soft skills such as teamwork were worked, where a leader was assigned considering the requirement that was being sought at the time. We also sought to provide effective communication, first by obtaining the requirements of the system, then by designing the general model of the system and the way in which they communicate, and finally by implementing the different parts required. It was taken into account that each person is responsible for carrying out their tasks, communicating their progress and the obstacles encountered to find an effective solution as quickly as possible.

4 Program Design

The program has 4 components: device driver, device library, interpreter and the android app.
The project's objective is to control the robot finger using a certain language to write some instructions. These instructions are necessary to make the robotic finger to move on a numeric keyboard and complete a PIN generated by the android app or a security pattern defined by the user. It's important to say that the app is capable of making three different views of the input, so the robotic finger can adjust the variables to fit the correct view.

The data flow of the project is shown below:

- The data will enter through the interpreter with the instructions written by the user that wants to generate a movement in the robotic finger.
- These instructions are analyzed by the library that is in charge of writing a code for each type of instruction on the device. Each one has an unique code, so the hardware knows what action do.
- The device takes this code sent by the user and transform it in an action to be executed on the robotic finger.

The interpreter has the job to take the input given by the user and transform it to computer language. The process consists in a parser to take each element and analyze it. The first step checks if the mnemonic of the instruction is correct to ensure that the input is correct. When it is analyzed, then it searches the argument. In this case the argument is an integer which says the amount of steps or spaces that the finger has to move.

The library consists of several methods that are capable of writing a specific code depending on the input generated by the user. Each of these codes has to be unique because the device has to read the operation on memory to execute the correct instruction. The objective is to make a modular project. When the tasks are separated the communication is easier between all the modules. With this idea, the tasks of writing on the device is assigned to the library and not to the interpreter or another method.

The driver probes searching for the device that wants to connect. When it finds the required device, the driver makes the initialization of the device to notify the operative system that the devices is ready to use. The main operations of a driver are write and read. In this project these operations are necessary to send the instructions to the device. When the device needs to finish the connection, the driver provides methods to do all the process and eliminate the register on the operative system, so there's no option to communicate to the device.

The embedded system used in the project is Arduino. For the hardware implementation the main components used are motors. The reason for this is that it's necessary

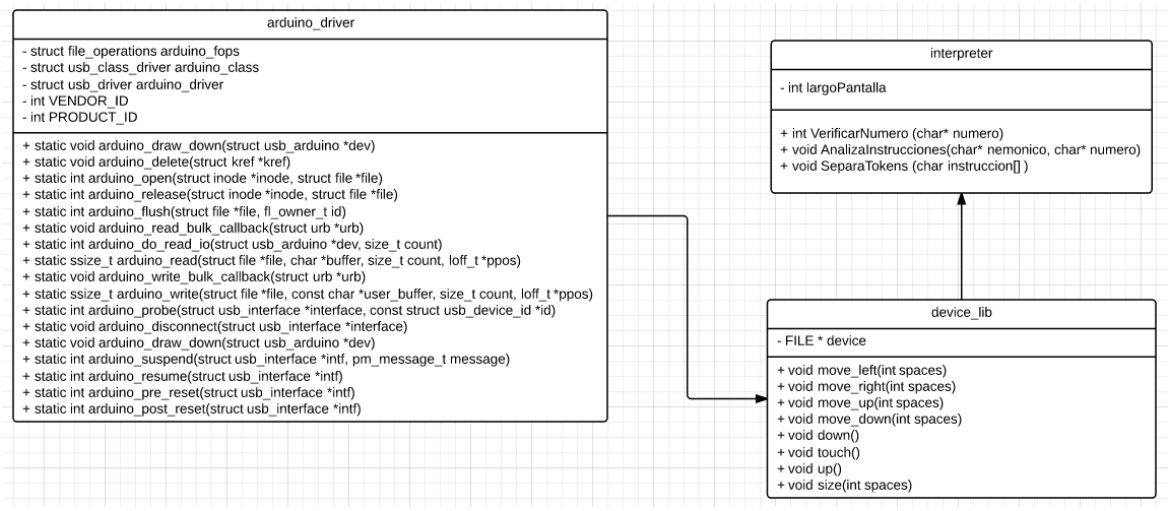

Figure 3: UML Diagram for the project

the movement of the robotic finger through the tablet.

The details of the communication between the different modules in the system are shown in the figure below.

5 How to use the program?

1. Execute this command sudo rmmod cdc_acm. Remove the factory Arduino driver

2. Execute this command cd driver && make. Enter the driver folder and compile it

3. Execute this command sudo insmod arduino_driver.ko .Insert the driver in the kernel

4. Execute this command sudo chmod a+rw /dev/arduino0. Read and write permissions

5. Generate the pin and pattern on movile device as follow image.

Figure 4: Pin

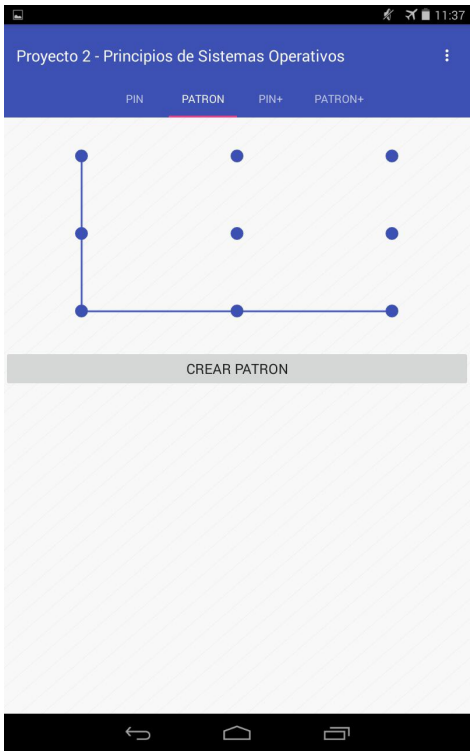

Figure 5: Pattern

6. Execute this command ./interprete on the correct directory. It will show the follow image.

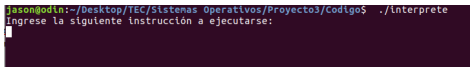

Figure 6: Interpreter

7. Type the instructions of lenguage.

6 Students activity Logs

Table 1: My caption

Member	**Activity**	**Description**	**Time (Hours)**
Jason, Jose, Luis, Carlos	UML Diagram	Meeting to discuss the UML diagram and the structure of the complete system.	1:00
Jason	Interpreter design	Research about the tokens for strings and creating the structure of the Interpreter's methods that allow the communication to the library.	3:00
Luis	Mobil App	Research about the Android Studio and it's variables, also about the libraries that allows the design of the pattern and numeric code.	2:30
Carlos	Hardware for Robotic Finger	Get the components of the hardware and create the design of the robotic finger. Also, begin with the implementation of the structure base .	8:00
Jason	Interpreter and Programing Language	Define the structure of the programming language with the correct syntax structure and it's semantic structure. Implement the up, down and push function in the interpreter.	3:00
Jose	Arduino Driver	Research about the design of Linux driver that allows the communication with an specific device also research about the communication protocol of an Arduino.	3:30
Jason	Interpreter	Define and implement the function of movement in any direction on the interpreter.	2:30
Luis	Library	Design and implement the functions that allows the writing bytes to the device into a library, It's writing in C language programming.	2:00
Jose	Arduino Driver	Implement the design of the structures and parameters that allows the identification of Arduino device. Also implement the methods to read and write data to the correct device.	5:00

Carlos	Hardware	Creates the design of the movement in any direction with the servo motors and stepper motor, also use a pulley to perform the movement. All this is done through an Arduino device.	8:00
Luis	Mobil App	Creates the screens of the numeric codes with a random algorithm that creates a security password, also creates the screen of the pattern that allows to set a specific password define by the user.	4:00

7 Project Final State

Interpreter

The final state of the interpreter is one hundred percent, it complies with identifying and verifying the sintaxys of the proposed language, as well as obtaining the data that the user wants to execute on the device.

Library

The state of the library is one hundred percent, it receives the data of the interpreter and writes the data in file of the device.

App

The state of the application is one hundred percent, it generates and verifies the pin, as well as the pattern.

Driver

The status of the driver is one hundred percent, it performs the reception and sending of data to the arduino.

Robotic Finger

The robotic finger is able to move and push the screen. If the finger is down, it's not always able to move. The functionality is around ninety percent.

8 Conclutions

1. It is concluded that the drivers not only need to write and read from and to memory but that it is also important to establish a protocol to send and receive data from the device.

2. It is concluded that modularity makes a better distribution of tasks and generates independence between each stage of the project, which makes it more portable and facilitates debugging.

3. It is concluded that interpreter is a tool which facilitates the interaction of the user with the application, translating from an everyday language to an abstract one so that the driver can understand the instructions that he wants.

9 Recomendations

1. Tools such as Flex and Jack can be used to implement the interpreter and analyze the language instructions.
2. You can use data structures defined for each device and each depends of own protocol.
3. You can use libraries for graphic generation in the mobile application as DeviceLib.
4. For the physical part you can use stepper motors to have better control of the movement in the two axes.

References

[1] Fisher, T. (2017). What Is a Device Driver?. [online] Lifewire. Available at: https://www.lifewire.com/what-is-a-device-driver-2625796 [Accessed 24 Nov. 2017]. pages

[2] Hope, C. (2017). What is a Device Driver?. [online] Computerhope.com. Available at: https://www.computerhope.com/jargon/d/driver.htm [Accessed 24 Nov. 2017]. pages

[3] Docs.microsoft.com. (2017). What is a driver. [online] Available at: https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/what-is-a-driver- [Accessed 24 Nov. 2017]. pages