Operating Systems Project

Pablo Rodriguez - 2014084649

Diego Solís Jimenez - 2014027244

Geovanny Espinoza Quiros - 2014046508

1 Introduction

1.1 Drivers

A driver is a software application that allows you to implement routines for control and access to modules and hardware devices. In Linux, the implementation of the driver is done in a C language file, which, by compiling it, generates a kernel module (.ko), corresponding to a device driver.[1]

Kernel space

The kernel handles the hardware of a machine directly, simply and efficiently, providing a uniform programming interface (API). Any subroutine or function that is part of the kernel, such as those found in modules and device drivers, are considered part of the kernel space. The interaction with hardware, in kernel space, is done through writing and readings to memory addresses, reserved for the use of hardware (memory maps or ports)[2]

User space

User-focused applications, such as UNIX shells, are part of the user space. These applications do not interact directly with the hardware, but through kernel functions for that purpose, or system calls (syscalls). The interaction with the hardware, in user space, is done through writing and readings to files of the file system provided by the operating system (for UNIX-based systems).[2]

1.2 Program description

The project is a robotic arm through which you can play checkers, the game field is a matrix of 10x10 posible positions. The robotic arm is controlled by a driver that must be implemented using the c lenguaje. In an user-friendly scenario, it is necessary to implement a graphical user interface, that means that the user can play checkes using a board display on the screen and all movements must be reflected or executed by the arm. The robotic arm can also be controlled with a list of commandsthat are previously validated by an interpreter. The following image shows the flow.

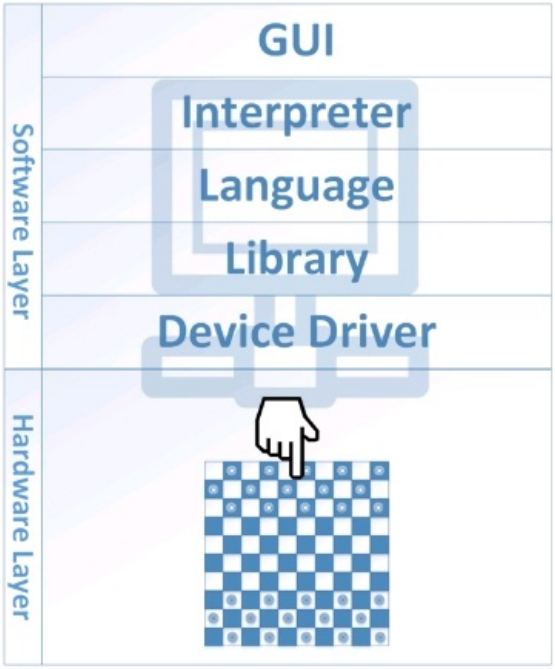


Fig 1.Flow of the project.

2.Development environment and tools.

Fig 2.Distribution of the operating system used.

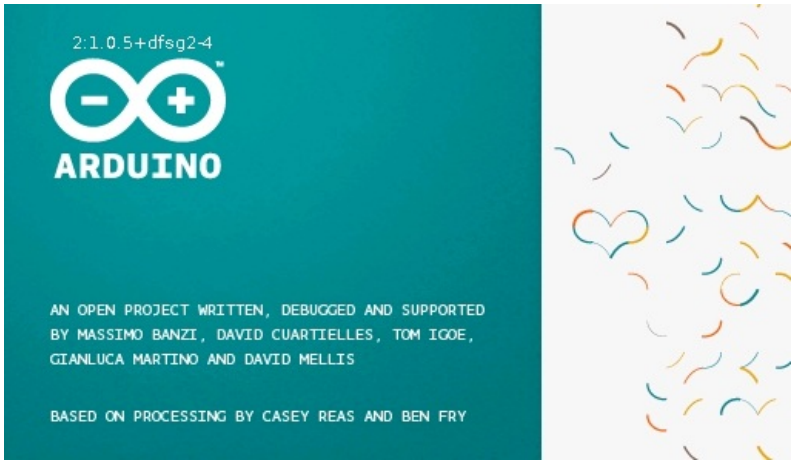


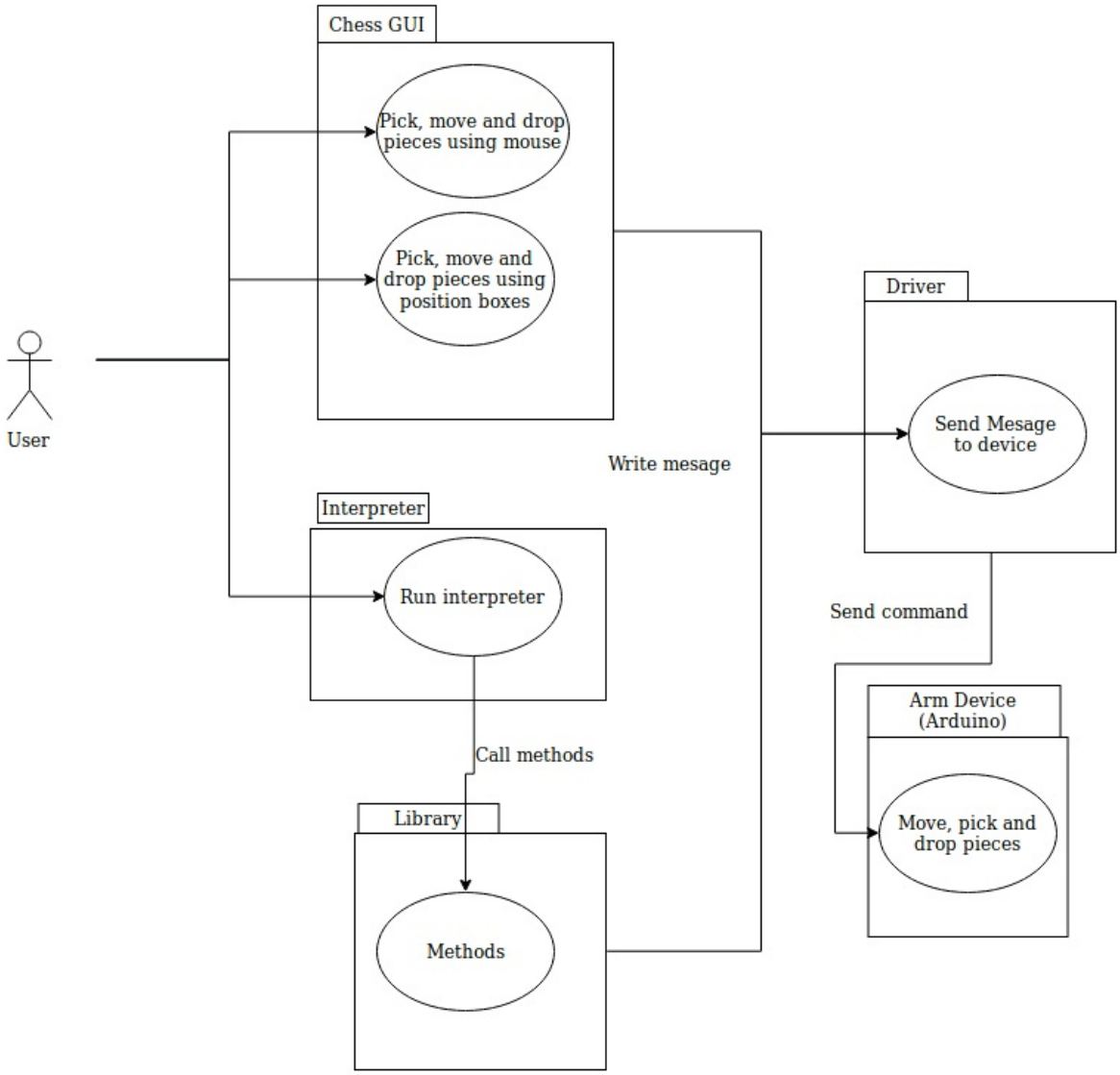
Fig 3.Arduino version.

3. Continuos Learning Attribute Analysis

This project provided the tools and directions needed to archive a better understanding of how device drivers function and how are they implemented and used by the operating system. Thanks to knowledge adquired in other courses like Analogic Design Workshop, Languaces, Interpreters and Compilers, and Algorithms and Data Structures 1&2 we could design and develop a fair implementation of the project, having the requiered understanding of hardware and software needed. But also, there was a new learning process, because we had to investigate and test concepts that we were completely unfamiliar with, like how drivers work and how they are implemented, the way you code them and how to interact with the devices. By developing this project, we could explore in the world of operating systems even deeper, we archieved new understandings about kernel space, user space, drivers, devices, arduino, team work and task scheduling for better development.

4.Provide the details of your program design

4.1 UML Diagram



4.2 Robotic Arm

The Robotic Arm is based on an open source mechanical device, the pieces were modified to satisfy the requirements. We use four servo motors to controll all the rotational movements, this kind of devices have the advantage of be very precise but they are really expensive and difficult to control. We use a Arduino Mega 2560 board and had some problems with the power because the arduino did not have enough to provide the energy that the four servos needed, so we had to connect using a external power source, another problem was that because of the robotic Arm weight, the servo motors get hot and lose strength, so we could not used for long periods of time or they can be damaged.

4.3 Language & Interpreter

Language

The language is simple, it consists of 5 instructions: move, pick, drop, move&pick, move&drop. These instructions are not case sensitive and can be written in capitals or not. Example: MOVE, move, MoVe. All these are correct.

In the case of move, move&pick and move&drop, the instruction word must be followed by a space and then two values from 0 to 9, these are the position in the board, separated by a comma "," and sorrounded by parenthesis, there can't be any spaces in this or any other characters. Example:

```
Move (1,2)
Move&pick (3,4)
```

All instruction lines must end with a line jump directly after the word (including the last instruction in the configuration file) spaces are not allowed. An example of a valid code is as follows:

```
move (1,2)
pick
move (3,4)
drop
move&pick (1,1)
move&drop (2,2)
```

Interpreter

The interpreter can be manually compiled by using the following command line on a terminal located inside it's folder:

```
gcc interpreter.c -o interpreter
```

To call the interpreter manually it must be done as follows:

```
./interpreter -c "configuration_file_path"
```

The interpreter will print error mesages depending on the error and will specify the line in which the error was found. If no errors where found it will make a call from the library to send the instruction using the device driver. This interpreter was developed using C language.

4.4 Device Driver

The device driver is based on Linus Torvalds Skeleton for USB. The driver creates a char device under the name of /dev/ardu%d. With this interface we can execute write and read operations to the device.

When the device is connected, the driver performs a probe and the first task is to identify the two interfaces for the USB device. One interface is for CDC control and the other one for CDC data.

Once the probe function performs the first flight and gets the bulk-in and bulk-out addresses, then we are ready to read and write to the device.

4.5 Robotic Arm Library

This library was developed using C language. It contains the necesary code to send instructions to the device through the device driver. This library is implemented in the language interpreter.

5.Using the Robotic Arm

Provide a description of the code of your different programs. ◦ Provide the details of the design of your project using UML. ◦ Provide the design details of your circuit created to implement the robotic checkers player. • Instructions of how to use the program. ◦ Provide detailed instructions of to setup and use the program and the robotic player.

6.Student activity log

TimeTable for Geovanny Espinoza Quiros

Activity	Date	Hours
Review and Analyze the specification document	26-4-19	2.5
Create the GUI	28-4-19/5-5-19	10
Build the robotic arm harware and assemble it	5-5-19/ 15-5-19	15
Create arduino code to developt test on the robotic arm	15-5-19/20-5-19	7
Research about drivers	20-5-19/22-5-19	4
Connect the GUI with C Languaje	28-5-19	1
Improve the robotic arm movements precision	29-5-19	3
Test the driver	30-5-19	1.5
Documentation	31-5-19	3

Total hours:	--------	47

TimeTable for Diego Solís Jiménez

Activity	Date	Hours
Review and Analyse the specification document	26/4/19	2
Begin design of arm device	28-4-19	2
First 3D print of arm device(several pieces for testing)	29-4-19	2
Arm fixing and corrections	1-5-19	2
Game Board elaboration using carboardbox	4-5-19	2
Game Board decoration	5-5-19	2
Driver testing and setting	10-5-19	2
Arduino servo configuration measuring	12-5-19	3
Arduino and servomotor feeding circuit elaboration	14-5-19	3
Arm 3D print and building	15-5-19	2
Language design	20-5-19	2
Interpreter design	22-5-19	2
Group Reunion	23-5-19	2
Begin Interpreter Development	24-5-19	3
Completed Interpreter Development	26-5-19	3
Interpreter fixes and improvements	28-5-19	2
Interpreter conection with library	30-5-19	3
Document Elaboration	30-5-19	2
Document Elaboration	31-5-19	4
Total hours:	--------	45

TimeTable for Pablo Rodríguez Quesada

Activity	Date	Hours
Research Device Drivers	17/05/2019	6
First try driver skeleton	18/05/2019	4
Debugging driver	22/05/2019	5
Try driver with Diego	23/05/2019	2
Change to serial driver	24/05/2019	5
Debug and read more about drivers	26/06/2019	5
Try with Geovanny driver and worked	27/06/2019	2
Calculate Polar mapping and create board	28/06/2019	3
Calibrate Arduino with Geovanny	30/06/2019	2
Total hours:	--------	40

7.Project final status

Project Current Status

- [x] The GUI application works.
- [ ] The GUI connects with the hardware
- [x] The Robotic Arm work with some precision problems.
- [x] The driver works and sends to the arduino.
- [x] The interpreter works and implements the library.
- [x] The library is functional and implemented.

We faced some problems that are worthwhile to mention. 1-We developed the first Robotic Arm release, after some test we noticed that it did not reach enought space to move thought all the posible position on the board, after some improvements trying to fix the robotic arm, we decided to create another and we had the same problem, so, it was until the third release when we found a usable one but on the other hand, strech the pieces caused that the screws were loose, so, we lost precision on the movements, at the end we had to deal with them because we could not find some mechanical engineer that could correct the pieces 2-The group required about 15 days trying to fix the driver, a lot of test were made, it was tried to change the arduino board, the Operative system and there was no way to execute it, at the

end, because of some reason, the driver was never wrong. It was just that it works on some specific computers.

8.Conclusions

Hardware is complex to build and design, but at the end it doesn't matter how good the design if there is not a way of controlling it's functioning in an agile and simple way. Software layers used for interactions with hardware are very important for computers as we know them. Without those layers, interacting with external hardware could be very difficult or maybe impossible. Providing the user with an interface that allows him or her to use an I / O device is a good way to limit what the user can do in order to reduce damages to the system. Interpreters are relevant because they allow for continuos execution of instructions while check for errors in the code, stoping execution when an error appears but having completed all of the previous instructions. Arduinos are not good for feeding energy to several devices, it is better to use it as a trigger that allows for another energy source to feed the devices (such as servomotors).

9.Suggestions, recommendations.

It is not recommended to create a driver in a virtual machine because a lot of errors were obtained and they were not trivial. It is recommended to test it directly in the operating system installed on the architecture. About the process of printing parts in 3D technology. It is recommended that the design can be reviewed in detail because the full impression of the robotic arm took around 24 hours and it was expensive. It is recomended that the calibration of servomotors is done with enough time and with precise elements because of how catastrophic an imprecision can be.

10.References

[1] What is a Device Driver?, January 2014. [Online]. Available: http://web.archive.org/web/20141021035445/https://www.pc-gesund.de/it-wissen/what-is-a-device-driver. [2] P. Silberschatz and G. Gagne, Operating System Concepts Essentials. Danvers, MA: John Wiley & Sons.Inc, 2011. [3] R. Arpaci-Dusseau and A. Arpaci-Dusseau. Operating systems: Three easy pieces. [Online]. Available: http://pages.cs.wisc.edu/ř̃emzi/OSTEP/cpu- sched.pdf [4] A. Corbet and G. Kroah-Hartman, Linux Device Drivers. Sebastopol, CA: O'Reilly Media, 2005.