# Creating a Linux Kernel Module (LKM) using C with a Beagleboard xM

**Cristian Castillo McQuiddy**
**(2014061245)**

**Jairo Mendez Martnez**
**(2014050475)**

*Instituto Tecnolgico de Costa Rica*

As part of this project a Loadable Kerne l Module (LKM) for Linux kernel 3.2.8 is going to be written to interact with the hardware in an embedded device (beagleboard XM).A loadable kernel module (LKM) is a mechanism for adding code to, or removing code from,the Linux kernel at run time. They are ideal for device drivers, enabling the kernel to communicate with the hardware without it having to know how the hardware works. The alternative to LKMs would be to build the code for each and every driver into the Linux kernel.

## 0 INTRODUCTION

Linux Loadable Kernel Module will be interacting within a Beagleboard. The kernel module will need to have all its natural capabilities: add it to the system , check the status of the LKM and remove it from the system. All the code will have to be developed using C. This LKM will be written in a Beagleboard. The LKM will be interacting with a few LEDs and buttons and it will be managed from the kernel space using the LKM.

A kernel is the lowest level of easily replaceable software that interfaces with the hardware in your computer. It is responsible for interfacing all of your applications that are running in user mode down to the physical hardware see, e.g., [1].Kernel modules run in kernel space and applications run in user space, as illustrated in Figure 1. Both kernel space and user space have their own unique memory address spaces that do not overlap. This approach ensures that applications running in user space have a consistent view of the hardware, regardless of the hardware platform. The kernel services are then made available to the user space in a controlled way through the use of system calls. The kernel also prevents individual user-space applications from conflicting with each other or from accessing restricted resources through the use of protection levels, see, e.g. [2]

.

The BeagleBoard is a low-power open-source hardware single-board computer.The BeagleBoard measures approximately 75 by 75 mm and has all the functionality of a basic computer.The OMAP3530 includes an ARM Cortex-A8 CPU (which can run Linux, Minix,FreeBSD,OpenBSD,RISC OS,or Symbian; Android is also being ported), a TMS320C64x+ DSP for accelerated video and audio decoding, and an Imagination Technologies PowerVR SGX530 GPU to provide accelerated 2D and 3D rendering that supports OpenGL ES 2.0. Video out is provided through separate S-Video and HDMI connections. A single SD/MMC card slot supporting SDIO, a USB On-The-Go port, an RS-232 serial connection, a JTAG connection, and two stereo 3.5 mm jacks for audio in/out are provided [3]

.

Built-in storage and memory are provided through a PoP chip that includes 256 MB of NAND flash memory and 256 MB of RAM (128 MB on earlier models) [3]

.

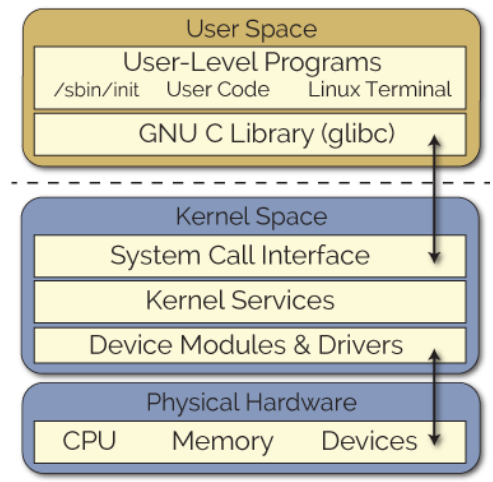


Fig. 1. Linux user space and kernel space.

The board uses up to 2 W of power and can be powered from the USB connector, or a separate 5 V power supply. Because of the low power consumption, no additional cooling or heat sinks are required [3]

.

The BeagleBoard is designed specifically to address the Open Source Community. It has been equipped with a minimum set of features to allow the user to experience the power of the processor and is not intended as a full development platform as many of the features and interfaces supplied by the processor are not accessible from the BeagleBoard. By utilizing standard interfaces, the BeagleBoard is highly extensible to add many features and interfaces [4]

.

There are a few operating systems you can choose to use on your BeagleBoard.Like: Angstrom, Ubuntu, and Debian all appear to have stable images

..

Angstrom is the default Linux distribution that is preinstalled on the eMMC on the BeagleBoard. It's a stripped down version of Linux specifically designed for embedded devices.Ubuntu have quite a few users, and a stable image with the 1.6.38 Linux kernel [5]

.

Debian is a Unix-like computer operating system that is composed entirely of free software,Debian includes popular free programs such as LibreOffice,(Firefox) web browser, Evolution mail, K3b disc burner, VLC media player, GIMP image editor and Evince document viewer. [6]

For this project it is being used Ubuntu, this one can be downloaded in [7], is a Debian-based Linux operating system,. It is based on free software and Ubuntu desktop.

The beagleboard is going to controller a circuit with 3 LEDs and one Button, them LEDs are connected to gpios 3,5,7 and the button to on 9.All the components are connected in the same and last pin ground, as illustrated in Figure 2 . The LEDs are default on/off, and the button active the function of them, like burst or OFF and ON all.

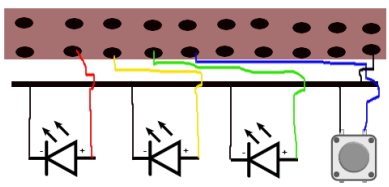

Fig. 2. Final circuit test.

## 1 Development environment

The kernel module was implemented in Ubuntu 12.04 armhf OMAP3/OMAP4 images, and was developed using C language, which is an imperative language, this one can be implemented in any C IDLE, for the development for this investigation SublimeText3 is the one that was used, this one can be downloaded in [8], or instead of using an IDLE almost all of the Linux distribution can integrate any code from terminal with the command *nano filename.termination*. The beagle-board's display is shown by an HDMI-DVI connection with a screen.

The kernel is going to run in a beagle-board XM and controller a circuit with 3 LEDs and one Button. As was mentioned the beagle-board is running in Ubuntu 12.04 armhf OMAP3/OMAP4, this can be downloaded and installed accorded to this link [7].

For the implementation of the interaction between the kernel module and the beagle-board it was necessary to use gpio. Gpio's are digital pins that are provided for the embedded, that acts like "interfaces" and allow the communication (in for reading and out for turning on/of leds, motors, etc) between a circuit, the embedded system and the software[9].

For the beagle-board xM C2, it's important to checkout the characteristics of the hardware, it can be found in [**?**], and the gpio signal connection it's give it by the Figure 3.

| EXP | Processor | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | VIO_1V8 | | | | | | | |
| 2 | | DC_5V | | | | | | | |
| 3 | AE3 | MMC2_DAT7 | * | * | * | GPIO_139 | * | * | Z |
| 4 | AB26 | UART2_CTS | McBSP3_DX | GPT9_PWMEVT | X | GPIO_144 | X | X | Z |
| 5 | AF3 | MMC2_DAT6 | * | * | * | GPIO_138 | * | X | Z |
| 6 | AA25 | UART2_TX | McBSP3_CLKX | GPT11_PWMEVT | X | GPIO_146 | X | X | Z |
| 7 | AH3 | MMC2_DAT5 | * | * | * | GPIO_137 | * | X | Z |
| 8 | AE5 | McBSP3_FSX | UART2_RX | X | X | GPIO_143 | * | X | Z |
| 9 | AE4 | MMC2_DAT4 | * | X | * | GPIO_136 | X | X | Z |
| 10 | AB25 | UART2_RTS | McBSP3_DR | GPT10_PWMEVT | X | GPIO_145 | X | X | Z |
| 11 | AF4 | MMC2_DAT3 | McSPI3_CS0 | X | X | GPIO_135 | X | X | Z |
| 12 | V21 | McBSP1_DX | McSPI4_SIMO | McBSP3_DX | X | GPIO_158 | X | X | Z |
| 13 | AG4 | MMC2_DAT2 | McSPI3_CS1 | X | X | GPIO_134 | X | X | Z |
| 14 | W21 | McBSP1_CLKX | X | McBSP3_CLKX | X | GPIO_162 | X | X | Z |
| 15 | AH4 | MMC2_DAT1 | X | X | X | GPIO_133 | X | X | Z |
| 16 | K26 | McBSP1_FSX | McSPI4_CS0 | McBSP3_FSX | x | GPIO_161 | X | X | Z |
| 17 | AH5 | MMC2_DAT0 | McSPI3_SOMI | X | X | GPIO_132 | X | X | Z |
| 18 | U21 | McBSP1_DR | McSPI4_SOMI | McBSP3_DR | X | GPIO_159 | X | X | Z |
| 19 | AG5 | MMC2_CMD | McSPI3_SIMO | X | X | GPIO_131 | X | X | Z |
| 20 | Y21 | McBSP1_CLKR | McSPI4_CLK | X | X | GPIO_156 | X | X | Z |
| 21 | AE2 | MMC2_CLKO | McSPI3_CLK | X | X | GPIO_130 | X | X | Z |
| 22 | AA21 | McBSP1_FSR | X | * | Z | GPIO_157 | X | X | Z |
| 23 | AE15 | I2C2_SDA | X | X | X | GPIO_183 | X | X | Z |
| 24 | AF15 | I2C2_SCL | X | X | X | GPIO_168 | X | X | Z |
| 25 | 25 | REGEN | | | | | | | |
| 26 | 26 | Nreset | | | | | | | |
| 27 | 27 | GND | | | | | | | |
| 28 | 28 | GND | | | | | | | |

Fig. 3. Expansion Signal Connection gpio.

The system must be prepared to build kernel code, and to do this you must have the Linux headers installed on your device. On a typical Linux desktop machine you can use your package manager to locate the correct package to install. For example, under 32-bit Ubuntu 12.04 armhf you can use:

sudo apt-get update

sudo apt-get install linux-headers-$(uname -r)

## 2 Libraries

- #include $< linux/init.h >$ : Macros used to mark up functions e.g. __init __exit.
- #include $< linux/module.h >$ : Core header for loading LKMs into the kernel.
- #include $< linux/device.h >$ : Header to support the kernel Driver Model.
- #include $< linux/kernel.h >$ : Contains types, macros, functions for the kernel.
- #include $< linux/fs.h >$ : Header for the Linux file system support.
- #include $< asm/uaccess.h >$ : Required for the copy to user function.
- #include $< linux/gpio.h >$ : Required for the GPIO functions.
- #include $< linux/kobject.h >$ : Using kobjects for the sysfs bindings.
- #include $< linux/kthread.h >$ : Using kthreads for the flashing functionality.
- #include $< linux/delay.h >$ : Using this header for the msleep() function.

## 3 Data Structures

a) static struct *kobj_attribute mode_attr* = __ATTR(LEDMode ,0666 , *mode_show*, *mode_store*); : Use these helper macros to define the name and access levels of the *kobj_attributes*.The *kobj_attribute* has an attribute attr (name and mode), show and store function pointers.The mode variable is associated with the LEDMode variable and it is to be exposed with mode 0666 using the *mode_show* and *mode_store* functions above.

b) static struct *kobj_attribute period_attr* = *__ATTR*(*blinkPeriod* ,0666 , *period_show*, *period_store*); : Use these helper macros to define the name and access levels of the *kobj_attributes*.The *kobj_attribute* has an attribute attr (name and mode), show and store function pointers.The period variable is associated with the blinkPeriod variable and it is to be exposed with mode 0666 using the *period_show* and *period_store* functions above.

c) static struct *kobj_attribute burst_attr* = *__ATTR*(*BurstRep* ,0666 , *burst_show*, *burst_store*); : Use these helper macros to define the name and access levels of the *kobj_attributes*.The *kobj_attribute* has an attribute attr (name and mode), show and store function pointers.The burst variable is associated with the BurstRep variable and it is to be exposed with mode 0666 using the *burst_show* and *burst_store* functions above.

d) static struct *attribute_group attr_group* = {*.name* = ledName, *.attrs* = *ebb_attrs*,}; : The attribute group uses the attribute array and a name, which is exposed on sysfs , which is automatically defined in the *ebbLED_init*() function below using the custom kernel parameter that can be passed when the module is loaded.

*.name* = ledName : The name is generated in *ebbLED_init*().
*.attrs* = *ebb_attrs* : The attributes array defined just above.

e) static struct attribute $*ebb\_attrs[]$ = {&*period_attr.attr*, &*mode_attr.attr*, &*burst_attr.attr* , NULL ,}; : The *ebb_attrs*[] is an array of attributes that is used to create the attribute group below.The attr property of the *kobj_attribute* is used to extract the attribute struct.

&*period_attr.attr* : The period at which the LED flashes.
&*mode_attr.attr* : Is the LED on or off.
&*burst_attr.attr* : Burst the LEDs.

f) static struct kobject $*ebb_kobj$; : The pointer to the kobject.
g) static struct *task_struct* $*task$; : The pointer to the thread task.

## 4 Functions

a) static *ssize_t mode_show*(struct kobject $*kobj$, struct *kobj_attribute* $*attr$, char $*buf$) : A callback function to display the LED mode.
Param kobj: Represents a kernel object device that appears in the sysfs filesystem.
Param attr: The pointer to the *kobj_attribute* struct.
Param buf: The buffer to which to write the number of presses.
Return: Returns the number of characters of the mode string successfully displayed.

b) static *ssize_t mode_store*(struct kobject $*kobj$, struct *kobj_attribute* $*attr$, char $*buf$, *size_t* count) : A callback function to store the LED mode using the enum above.
Param kobj: Represents a kernel object device that appears in the sysfs filesystem.
Param attr: The pointer to the *kobj_attribute* struct.
Param buf: The buffer to which to write the number of presses.
Param count:Indicate when the characters line end,is

important as otherwise the \n is used in the comparison.

c) static *ssize_t burstRep_show*(struct kobject **kobj*, struct *kobj_attribute *attr*, char **buf*) : A callback function to display the LED period.
Param kobj: Represents a kernel object device that appears in the sysfs filesystem.
Param attr: The pointer to the *kobj_attribute* struct.
Param buf: The buffer to which to write the number of presses.
Return: Returns the number of characters of the mode string successfully displayed.

d) static *ssize_t burstRep_store*(struct kobject **kobj*, struct *kobj_attribute *attr*, char **buf*, *size_t* count) : A callback function to store the LED period value.
Param kobj: Represents a kernel object device that appears in the sysfs filesystem.
Param attr: The pointer to the *kobj_attribute* struct.
Param buf: The buffer to which to write the number of presses.
Param count:Indicate when the characters line end,is important as otherwise the \n is used in the comparison.

e) static *ssize_t period_show*(struct kobject **kobj*, struct *kobj_attribute *attr*, char **buf*) : A callback function to burst the LEDs n times.
Param kobj: Represents a kernel object device that appears in the sysfs filesystem.
Param attr: The pointer to the *kobj_attribute* struct.
Param buf: The buffer to which to write the number of presses.
Return: Returns the number of characters of the mode string successfully displayed.

f) static *ssize_t period_store*(struct kobject **kobj*, struct *kobj_attribute *attr*, char **buf*, *size_t* count) : A callback function to store the LEDs n times.
Param kobj: Represents a kernel object device that appears in the sysfs filesystem.
Param attr: The pointer to the *kobj_attribute* struct.
Param buf: The buffer to which to write the number of presses.
Param count:Indicate when the characters line end,is important as otherwise the \n is used in the comparison.
Return: Returns the number of characters of the mode string successfully displayed.

g) static int flash(void **arg*) : The LED Flasher main kthread loop.
Param arg: A void pointer used in order to pass data to the thread.
Return: Returns 0 if successful.

h) static int __init *ebbLED_init*(void ) : The LKM initialization function.The static keyword restricts the visibility of the function to within this C file. The __init macro means that for a built-in driver (not a LKM) the function is only used at initialization time and that it can be discarded and its memory freed up after that point.In this example this function sets up the GPIOs and the IRQ.

Return : Returns 0 if successful.

## 5 Instructions to use the program

For build the kernel module is necessary to generate a Makefile. Then you have to follow the next steps:

- From terminal access to the carpet where the LKM and the Makefile are saved.
- Enter the command *make*
- Then for loaded the kernel module enter *sudo insmod module-kernel-name.ko*
- Access to /sys/BBLKM/led137
- For turning on all the leds enter: *echo 1 ¿ LEDMode*and then enter *echo 2 ¿ button*
- For generate a burst in the leds enter: *echo burst ¿ LEDMode*and then enter *echo 2 ¿ button*
- For catch any file value as: LEDMode, LEDStats, button, number, burstRep, enter: *cat file-name*
- For unloaded the kernel module enter: *echo 1 ¿ LEDMode*and then enter *sudo rmmod BBLKM,ko*

## 6 Project final status

Finally it could not, work with a button, this button must control the LEDs, when push the button, the LEDs turning off/on.So the files like difftime, lasttime and button status doesn‘t work.

The files like LEDMode, LEDStatus,blinkperiod,burst and burstRep( this should work with the button, but the button doesn‘t so is working with the kernel commands) is working good.

### 6.1 Some issues or bugs

1) When the user insert a 0 number in the button, some times the kernel, doesn‘t work very well.

2) Some time the kernel create multiples mayornumber, and this one only should exist one, so the kernel doesn‘t work.

3) Some LEDs are always a little on o lighting.

4) The burstRep only work with the kernel commands.

5) The kernel never take the time very well.

6) When the burst is true,two of the three LEDs turning on like the same time, but is no the same time.

### 6.2 Challenges during the development

1) Boot the beagleboard:kernel panic not syncing: Fatal exception in interrupt.The error hapened because the computer doesn't give enough electric current to the beagleboard, which caused that device is turning off when was booting.

   Solution: Buy a adapter of 5 volts.

2) Kernel beagleboard: The device had a linux kernel 2.6.32, which it's no best option to run the linux modules.

   Attemps: Research how install other operating system or image like debian on the beagleboard.

3) Format the miscro sd: Problem trying format the miscro sd, the device its no detected by the computer.Is very important format the device because be need a S.O comfortable to build the LKM.

   Attemps: It was used diferent S.O on the computer and no one works, never detected the tarjet sd, doesn't worked format from the terminal,also it was used the microsd with other adapter.

   Solution: Change the adapter micro sd for a new one.

4) Testing makefile with the computer: When it's use the command make, the terminal show the error "makefile:15: *** missing separator . Stop", where it's mean that error is in the line 15 on the code and one caracter was missing, can be a space or a tab, finally the problem had no solution.

5) Error Module.h: When it's use the command make the terminal show "fatal error: Linux/module.h: No such file or directory".

   Solution:Such error was beacause it wrote Linux with uper case y the kernel only reco admit lower case see, e.g. [11],[12],[13].

6) Terminal Command from BeagleBoard: When it is connecting the beagle to a monitor with adapter HDMI-VGA, hte monitor show the terminal each 5 min.

   Attemps: Change the resolution of the monitor, changing the file of the resolution from the image debian, but that file needs permision to write, so was edited from terminal.

   Solution: Buy a new microsd and connect the beagleboard to a monitor directly wtih cable HDMI or serial.

7) Testing Circuit with Debian: Installing and configuring Debian again, because angstrom doesn't worked. Designing a circuit with one led,resistence,button and transistor, as illustrated in Figure 4, but the led never turning on.

   Attemps: Designing a new circuit, with only one led and one resistence, the led was connected in the pin ground(24) and the pin of 5v(1), the led turned on but,it's never let controlling by the kernel.

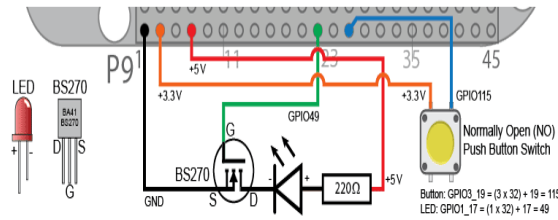


Fig. 4. First circuit test.

8) Angstrom Test: The makefile doesn't work, the system show that "make: coomand not found".

   Attemps: Edit the files of opkg folder, changing the command "armv7a" to "armv7a-vfp-neon", and now it's can update the system and upgrade using opkg. The time of upgrade was so long, after that the command make doesn't work but the problem is that the file build doesn't exist. The attemps was changing the command of the makefile, install make, install kernel, create a build folder, reboot the system but the beagleboard never was the same, the final attemp was changes to hte debian operating system again.

## 7 Conclusions and Suggestions

The BeagleBone is more famous than BeagleBoard,so while the research, there are more information about BeagleBone.

The newest is the BeagleBone Black, is almost of fast than the beagle board, the second newest is the Besagle-

Bone but is the lowest

The BeagleBoard is the oldest but is the best, for multiples functions o problems.

Angstrom is a Operating System that is better than Debian detecting the gpio or pin of the BeagleBoard, but is very hard to configure the kernel,unlike Debian is better that Angstrom configuring the kernel but can not detected the gpio or pins.

Angstrom can spend 6 hours configuring and preparing the system to run the kernel and in the end maybe not will finish.When the system turning off o shutdown, the BeagleBoard maybe never again,will reboot.

Each gpio give volts varied.So the LEDS are without resistances and some LEDS are always a little lighting.

The best Operating System is Ubuntu, configuring and preparing the system was very fast and easy.It can detect the gpio pins, this system is like the almost the same of the PC ubuntu.

## 8 REFERENCES

[1] Howtogeek.com. (2016). What is the Linux Kernel and What Does It Do?. [online] Available at: http://www.howtogeek.com/howto/31632/what-is-the-linux-kernel-and-what-does-it-do/ [Accessed 8 May 2016].

[2] derekmolloy.ie. (2016). Writing a Linux Kernel Module Part 1: Introduction — derekmolloy.ie. [online] Available at: http://derekmolloy.ie/writing-a-linux-kernel-module-part-1-introduction/ [Accessed 8 May 2016].

[3] Wikipedia. (2016). BeagleBoard. [online] Available at: https://en.wikipedia.org/wiki/BeagleBoard [Accessed 9 May 2016].

[4] Anon. (2016). BeagleBoard-xM Rev C System Reference Manual. [online] Available at: http://beagleboard.org/static/BBxMSRM-latest.pdf [Accessed 9 May 2016].

[5] Justin Cooper. ( 2015-01-16). BeagleBone Black: Installing Operating Systems [online] Available at: https://learn.adafruit.com/downloads/pdf/beaglebone-black-installing-operating-systems.pdf

[6] Wikipedia. (2016). Debian. [online] Available at: https://en.wikipedia.org/wiki/Debian [Accessed 9 May 2016].

[7] ”ARM/OMAP - Ubuntu Wiki”. Wiki.ubuntu.com. N.p., 2016. Web. 12 May 2016.

[8] 2016. [Online]. Available: https://www.sublimetext.com/3. [Accessed: 10- May- 2016].

[9] ”RPi Low-level peripherals - eLinux.org”, Elinux.org, 2016. [Online]. Available: http://elinux.org/RPi_Low-level_peripherals. [Accessed: 13- May- 2016].

[10] Beagle-board - x M Rev C System Reference Manual. 2010, p. http://beagleboard.org/static/BBxMSRM_latest.pdf.

[11] Directory, linux/module.h. ”Linux/Module.H No Such File Or Directory”. Stackoverflow.com. N.p., 2016. Web. 10 May 2016.

[12] ”Question 141319 . Questions . Linux Package . Ubuntu”. Answers.launchpad.net. N.p., 2011. Web. 10 May 2016.

[13] ”Linux/Module.H”. Linuxquestions.org. N.p., 2016. Web. 10 May 2016.

[14] ”Writing A Linux Loadable Kernel Module (LKM) - Interfacing To Gpios — Derekmolloy.Ie”. derekmolloy.ie. N.p., 2016. Web. 10 May 2016.

[15] ”Writing A Linux Kernel Module Part 2: A Character Device — Derekmolloy.Ie”. derekmolloy.ie. N.p., 2016. Web. 10 May 2016.

## THE AUTHORS

Table 1. Details of the activities performed by Cristian Roberto Castillo Mcquiddy

| Date. | Activity | Description | Hours |
|---|---|---|---|
| 29/05/2016 | LKM Research | Research a LKM of how they work,how they are created and What are their main functions. | 3 |
| 02/05/2016 | Write a Kernel Module | Preparing system to build kernel code,Linux headers installing,create a module code that print a "Hello World" and a makefile, finally testing the LKM custom parameter see, e.g. [2]. | 6 |
| 03/05/2016 | Testing BeagleBoard | Connecting the beagleboard with a cable adapter HDMI-VGA and booting | 3 |
| 03/05/2016 | File Operations Data Structure | Changing the device driver source code, adding functions like: write and read files, and too the function receive and send message to user space with the kernel to know when create the files see, e.g. [14]. | 5 |
| 07/05/2016 | Debian Test | Installing and configuring the operating system Debian to test the kernel | 2 |
| 07/05/2016 | Angstrom Test | Installing and configuring the operating system Angstrom to test the kernel | 6 |
| 07/05/2016 | Testing Circuit with Debian | Installing and configuring the operating system Debian to test the kernel and connecting the circuit with one led,resistence to the bealgleboard. Testing the circuit or off and on the led with the kernel of debian see, e.g. [15]. | 3 |
| 10/05/2016 | Testing Ubuntu | Installing and configuring the operating system Ubuntu on the beagleboard | 3 |
| 10/05/2016 | Testing Button | Designing a circuit with one led and button, when the button is pressed the led turning on | 1 |
| 10/05/2016 | Testing 3 LEDs | Designing a circuit with 3 LEDs and configuring the kernel.Turning on the 3 LEDs with kernel commands and make the burst LEDs | 4 |
| | | Total amount of hours: | 30 |

Table 2. Details of the activities performed by Jairo Mndez Martnez

| Date. | Activity | Description | Hours |
|---|---|---|---|
| 30/04/2016 | BeagleBoard Boot Investigation | Started the investigation about how the beagleboard work, how to booted,hot install a new OS and how display it. | 6 |
| 1/05/2016 | Continue with the investigation | Working with troubles, because the beagle didn't boot | 3 |
| 2/05/2016 | LKM investigation | Installing a new OS in the micro SD with the adapter of a classmate, because mine didn't work. Started the investigation of how to generate my own module linux kernel and started the document | 4 |
| 3/05/2016 | Trying to display the beagle | Tried to boot the beagle and it displayed the image using a conversor of HDMI to VGA, and it generate some problem because the configuration of the beagle. A simple example of LKM was write | 5 |
| 4/05/2016 | Read and write files | Implementing a simple LKM who can read and write some files | 2 |
| 6/05/2016 | Try to generate a .ko | Trying to create a file .ko with a Makefile in Debian and it does works, but using the gpios doesn't because of the kernel version | 5 |
| 7/05/2016 | Installing another OS | tried to install the most actual Angstrom OS, because it that moment it was the only OS that was known for using the gpios but it doesn't create an .ko file with make | 3.5 |
| 9/05/2016 | Tried to use angstrom | Still tried to make work angstrom to link the BBLKM code and the gpios, gaste 4h just in upgrade, and when it reboot doesn't work | 5 |
| 10/05/2016 | Installing ubuntu 12.04 armhf | Ubuntu 12.04 was installed and configuring the OS for working | 6 |
| 11/05/2016 | Trying to coupling the code with the gpio | Coupling the code code with the gpio, the code has implemented all leds functions, but the button is not implemented | 5 |
| | | Total amount of hours: | 44.5 |