**Universidad Autónoma de Guadalajara**

Ingeniería Electrónica Biomédica

Architecture of Microcontrollers

*“*Practice 5. Development of a simple console”

Andrea Alejandra Mondragón Olivos

Jesús Arnoldo Zerecero Núñez

2915351

2885993

Practice 5. Development of a simple console

**Introduction**

The objective of this practice was creating a simple user-commands-terminal (known as “shell”) in which the user can communicate with the microcontroller and enable some of its functions. The user has multiple command options which he can type in the terminal. The communication was established using the microcontroller’s UART module, and the commands interact with multiple internal modules: GPIO, TPM/PWM, ADC and PIT.

The shell was implemented used the *Tera Term* software, a neat VT100 terminal emulator. The user must type a command name and may type one or two parameters to scroll through the different options that command may offer. In total, 10 commands were coded and implemented:

1. **cls:** Clears the text and its format from the terminal.
2. **rgb (*color*):** Turns on any of the 7 possible led colors. These colors are created using the combination of the RGB (red, green and blue) the microcontroller offers. How much of each color is used is modulated by the TPM/PWM module.
3. **led (*color*) (*on/off/toggle*):** Chooses a RGB color and turns it on, off or toggles it. The TPM/PMW is also used to control these RGB colors.
4. **pot (*start/stop*) (*period (ms*)):** Commences or stops a continuous measurement of the voltage of an external potentiometer. The user may also choose how often this measurement will be refreshed.
5. **help:** Prints the full command list.
6. **print:** Opens the printer interface in order to physically print the terminal’s text.
7. **color (*color number*):** Changes the terminal’s font color, from 8 different options.
8. **temperature:** Prints the current temperature measured by an external LM35.
9. **ledexterno (on/off):** Toggles an external led.
10. **buzzer (on/off):** Toggles an external buzzer.

These are the new modules and concepts implemented in this practice:

* **TPM/PWM (Timer/Pulse Width Modulation)**

This module is similar to the PIT module, whereas a value is set and a counter will increase or decrease until reaching this value and will activate a flag, for something to be done.

A PWM is a rectangular wave pulse regulated by a period and a duty cycle. The period represents how long a wave cycle lasts, while the duty cycle represents how much time this cycle will yield a logical one. The duty cycle is often represented as a percentage of the period, where 100% duty cycle represents a continuous 0, while 0% duty cycle represents a continuous 1.

* **LEDs RGB (PWM)**

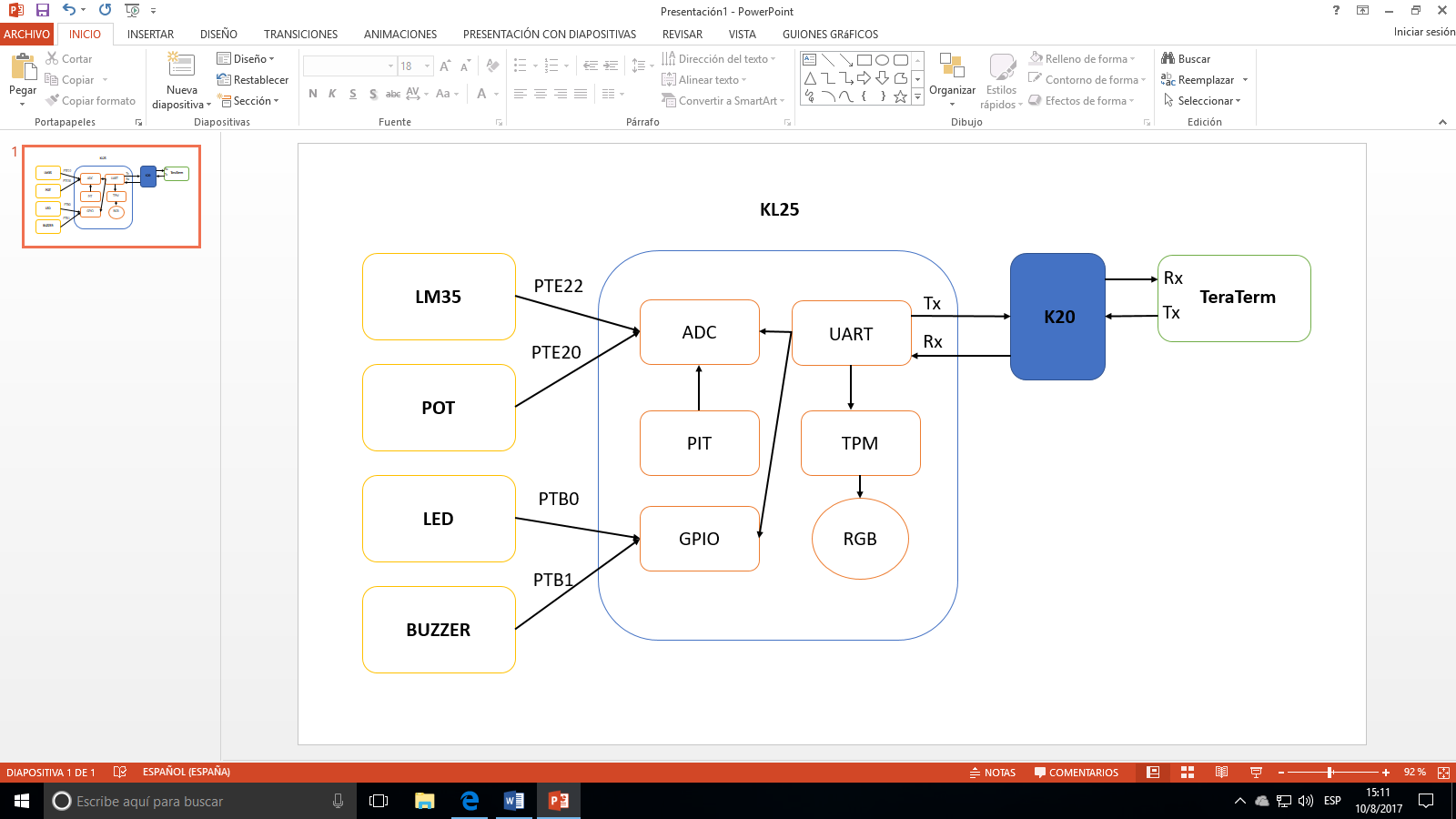
The RGB leds can also be controlled utilizing the TPM/PWM module, apart from the GPIO module. Using PWM, a certain duty cycle can be assigned to control how much of a certain RGB color will be on during its period. This allows the user to make up a great variety of color combinations, using just the base red, green and blue colors.

* **UART (Universal Asynchronous Receiver/Transmitter)**

UART is a serial communication protocol, in which data is sent byte per byte from one device to another. As the UART is asynchronous, a baud rate must be assigned for both devices in order to synchronize one’s transmission with the other’s reception. The baud rate represents the speed the bytes are sent. The UART’s data bus is composed by a start bit, 8 or 9 data bits, one or two stop bits and one, two or no parity bits. Parity bits are used to make sure the data was received correctly.

* **VT100 standard (Tera Term terminal)**

This terminal emulator utilizes a set of commands and ASCII characters which allow for screen clearing, backspacing, changing the background color, and many others. These commands are accessed just by typing a certain set of characters (ex. <ESC>c will reset the terminal). Some of these commands were implemented in this practice, where, whenever the user types an available command name, the microcontroller will be the one typing these set of characters to the terminal, enabling the chosen command.



*Figure 1. Circuit´s Block Diagram*

**Development.**

The following steps were followed in order to code this practice:

1. Create a UART driver, able to send and receive data (byte per byte) using interruptions.
   1. The UART data register can be written on or read from.
      1. Whenever it’s written on, it means the microcontroller will serve as the transmitter. The transmitter flag will activate and data will be sent byte per byte until there’s no more data to send.
      2. Whenever it’s read from, the microcontroller will serve as the receiver. The receiver flag will activate, the data will be caught and the UART callback function will be called.
         1. The callback function enables the application to read the data received, separate it into 3 different buffers (command name, parameter1 and parameter2) and begin comparing the strings in order to know which command the user typed and wants to activate.
2. Create a TPM/PWM driver, able to modulate the duty cycle of each of the RGB leds according to the parameters it receives.
3. Include the drivers created from previous practices: GPIO, ADC and PIT.
4. In the main source, an array of structures will be initialized. There will exist a structure per available command.
   1. Each structure contains a constant string, representing the exact word the user must type in order to access that command’s function. A pointer to that function is also included in the structure.
5. Every time a space (‘ ‘) is typed, the rest of the characters received will be stored in one of the two parameter buffers. Whenever an enter (‘\r’) is typed, the command is read, the parameters are sent to the respective command function and the buffers are cleaned.
6. If a command was typed correctly, the respective function will be called and a certain action will occur. If a command name or a parameter was wrongly typed, an error message will be printed on the screen.

The hardest tasks we encountered while coding this practice were understanding the functionality of the UART and PWM drivers, as coding them was what took the most time. We encountered many errors constantly on the drivers, but at the end, everything flowed smoothly. Another big problem encountered was how some commands transmitted from the KL25 to the terminal weren’t being received properly, and some data was distorted. At the end, we noticed this problem was caused by the great speed the data was being sent at, as the receiver wasn’t able to store it in time before new data was coming in. This was fixed by simply adding a delay every time a command string was sent.

**Conclusion:**

**Arnoldo:**

In conclusion, having developed a command terminal using just the KL25 microcontroller made me understand the true depth and variety of tasks we can achieved using a relatively simple microcontroller. The possibilities seem endless. Using a neat, user-friendly command terminal makes controlling some of this tasks and possibilities in an even easier way, as the user won’t even need to interact with the code anymore. At first, I thought programming and using microcontrollers was an extremely complicated task, but as I am developing and understanding more and more drivers for its modules, it seems like a kid’s game now. I feel capable of accomplishing many things now in a much easier way, by just using the KL25.

**Alejandra:**

We can conclude that we applied most of the knowledge we previously acquired about peripheral use, led control and communication protocols necessary to send and receive the commands. We utilized commands which interacted with the PWM, ADC, PIT and even GPIO modules. We were able to execute many different actions, including controlling external electronical components, such as the LED, the potentiometer, the LM35 and the buzzer.

**References:**

* VT100 User Guide. Digital Equipment Corp., 1981, print.
* Other manuals: VT100 Series Technical Manual (EK-VT100-TM), serial printer interface VT1XX-AC User Guide (EK-VT1AC-UG)
* Adam Osborne, An Introduction to Microcomputers Volume 1: Basic Concepts, Osborne-McGraw Hill Berkeley California USA, 1980 ISBN 0-931988-34-9 pp. 116-126
* Du, Ruoyang; Robertson, Paul (2017). "Cost Effective Grid-Connected Inverter for a Micro Combined Heat and Power System". IEEE Transactions on Industrial Electronics. ISSN 0278-0046. doi:10.1109/TIE.2017.2677340.