**Universidad Autónoma de Guadalajara**

Ingeniería Electrónica Biomédica

Architecture of Microcontrollers

*“*Practice 4. LM35 measurement on LCD using cortex M0+”

Andrea Alejandra Mondragón Olivos

Jesús Arnoldo Zerecero Núñez

2915351

2885993

Practice 4. LM35 measurement on LCD using cortex M0+

**Introduction**

For this practice, a measurement taken from an LM35 temperature sensor was processed through the ADC module in order to show the value measured in a 16x2 LCD screen.

For the code, an ADC, LCD and PIT driver were implemented. The PIT driver will be used to create a countdown which activates a flag every 100 ms. This countdown is needed to delay each of the functional parts of this practice: Taking an ADC value from the LM35 each 200ms, toggling the microcontroller’s blue LED each 1 second and refreshing the ADC value show in the LCD each 2 seconds.

Other auxiliary modules were also used:

* Nested Vector Interrupt Contoller (NVIC): To utilize interrupts with the PIT countdown terminated flag (each 200 ms, this flag will set).
* General Purpose Input Output (GPIO): In order to activate and utilize the external ports, for the LM35 input, and the LED and LCD outputs.
* **LM35 sensor**

This is a precision integrated-circuit temperature device with an output voltage linearly-proportional to the centigrade temperature (for every °C measured, 10mv are sent). This device does not require any external calibration or trimming to provide accurate measurements from -55°C to 150°C.

* **Liquid Crystal Display (LCD)**

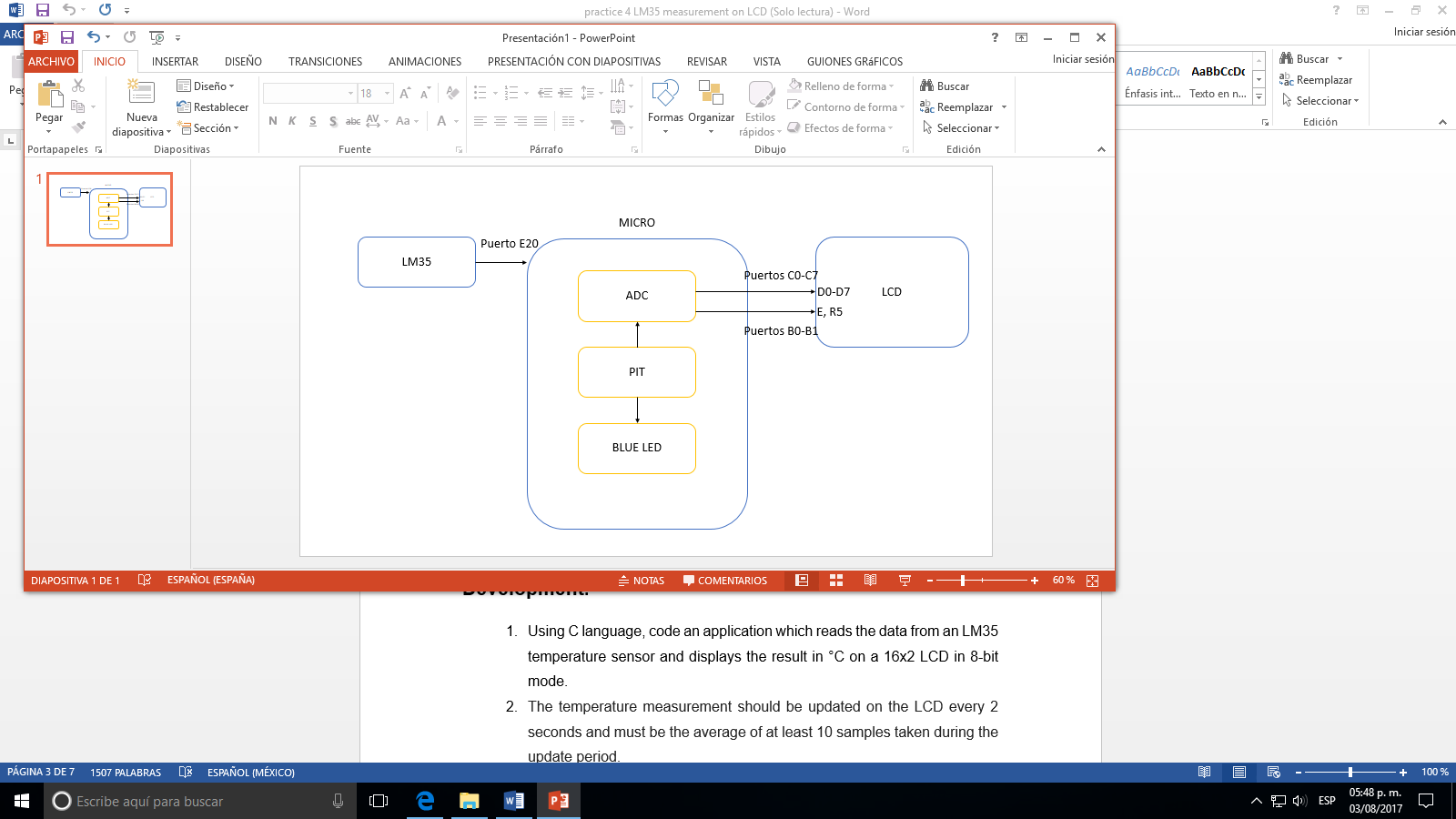
This device is a driver/controller matrix liquid crystal display, it has two lines, each able to of show up to 16 characters. It is ideal for displaying information in micro controllable digital systems. This is a LSI (Large Scale Integration) component, it offers the possibility of being controlled by 4 or 8 bits. In this case, the 8-bit mode was selected.

* **General Purpose Input Output (GPIO)**

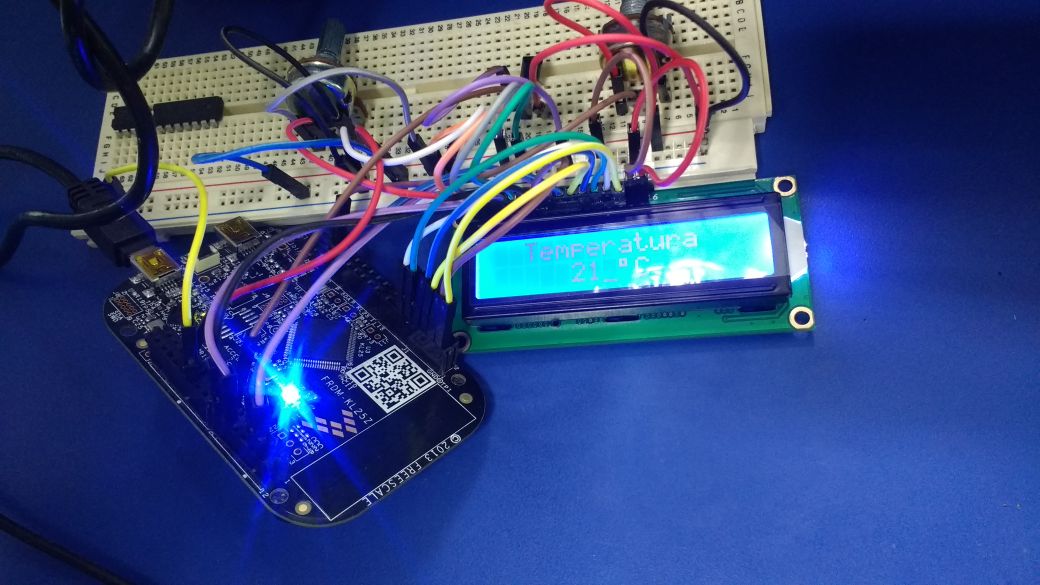
In most embedded systems, a processor will be ultimately responsible for sensing the state of any devices which translate external signals (like the LM35 sensor), to digital level logic voltages (the ADC module’s purpose), and controlling devices which translate logic level directly into action. This is the function of the GPIO

* **Periodic Interrupt Timer (PIT)**

The interrupts are program parts which are reacting to the events taking place in the microcontrollers. They are usually used for quick response to an event, but they can also be used for completing several parallel processes, precisely timed action and saving power. For example, it is possible to make a LED blinking using interruptions, so that blinking frequency does not depend on what is happening in the program at the moment. When the interrupt occurs, then main program execution is halted and the interrupt’s source is attended. In most cases, whenever the interrupt flag is set, a callback function is called, so the user can do whatever he wants inside that function. When the callback’s function finishes its execution, the program continues on the state where it was left off.



*Figure 1.Circuit’s Block Diagram*



*Figure 2. Circuit’s physical assembly.*

**Development.**

1. Using C language, code an application which reads the data from an LM35 temperature sensor and displays the result in °C on a 16x2 LCD in 8-bit mode.
2. The temperature measurement should be updated on the LCD every 2 seconds and must be the average of at least 10 samples taken during the update period.
3. The microcontroller’s blue LED must be toggled each second for show the application running.

**LCD driver:**

1. Initialize the function for the activation of ports and sending of the first commands:

* 1. Clocks for Port B (for the RS and E) and C (for the data from D0 to D7) are activated and set as outputs.
  2. The 8 bits, 2 lines mode command is sent 3 times so that the LCD captures the command.
  3. The display is cleared.

1. Create a function which sends a single letter or command to the LCD.
2. Create a function which sends a whole word or phrase, calling the previous function multiple times.
3. Create a function which sends the exact coordinates to the LCD, for it to place the cursor and start writing from there.

**ADC driver:**

1. Declare the function of ADC initialization.
   1. We activate the clock for port E, activating port pin 20 for the LM35 input.
   2. Activate the clock for the ADC0.
   3. The structure’s parameters are read and the respective register masking takes place.
2. A calibration is declared but not used, as calibration is not needed for the 8 bit resolution mode.
3. Finally, an ADC reading function is created, where the ADC channel is selected.
   1. A do nothing cycle is created in order to wait for the COCO flag (polling) to set (conversion completed).
   2. The result register value is saved in a return variable and the ADC is turned off.

**PIT driver:**

1. We declare the static pointer for the PIT callback function.
2. Start the initialization function of the given PIT code.
   1. Activate the PIT clock.
   2. Load the timer count to the pit load value.
   3. Enable the interrupt module.
3. Initiate the counter.

**Main:**

1. Include the .h files that we need to include the peripheral declarations, the PIT counter, the header of stdint for the C code, and the ADC and LCD initializations.
2. Defines the macros used in main.c.
3. Create a structure for the ADC‘s initial configuration in which the user chooses the parameters to send the ADC initialization function.
4. Then we declare the function for the PIT callback for the counters’ increasing.
5. We write and declare all the variables that we will use in this section, for the ADC, LED, LCD, the LM35 measure, the ADC channel, the ADC accumulator, the index, and the BCD data to show in the LCD.
6. Inside the main function:
   1. The PIT is initialized and its Counter begins.
   2. The ADC is initialized.
   3. Port D’s clk is enabled, as the LED’s output depends on it.
   4. Pin 1 of the D port will be used to toggle the LED
   5. The LCD is initialized, and the word "Temperature" in the LCD coordinates ​​of y = 0, x = 2, “°C” is written on the lower line of the LCD.
   6. The function enters an infinite cycle, where the counter increased by the PIT’s callback function will be evaluated. Each counter increases every 100ms.
      1. Whenever the ADC counter reaches 2 (every 200ms), a sample is read.
         1. Whenever 10 samples are read, an average is taken, offseted and converted to BCD.
      2. The BCD conversion results are written (and refreshed) on the LCD (units, tens and thousands).
      3. Whenever the LED counter reaches 10 (every second), the blue LED will toggle.

This will give us a continuous functionality which will constantly show the current room temperature in the LCD screen, while the LED toggle each second.

**Conclusion:**

**Arnoldo:**

In conclusion, the usage of the microcontroller’s internal modules and external inputs/outputs allows us to elaborate many different applications, according to what’s needed. Almost anything is possible. A good thing to do is create drivers for each module, and just include them in your application whenever you want to use them. This drivers must be neat and must allow you to send every existing parameter and option depending on how the user wants the module to work. The difficulties we had while coding this application were knowing how the ADC’s and GPIO’s modules and registers work. Also, programming the LCD was no small task, as a certain delay had to me made between each character or command sending. This delay is uncertain, so multiple tests had to be made. Also, another difficult job was adjusting the value read by the ADC module to the real value, for which an exact rule of three function had to be calculated.

**Alejandra:**

With this practice we were able to study and practice the use of peripherals with the activation of these ports in the initialization as indicated in the reference manual of the microcontroller, as well as the application of UART to set the communication with the hardware that we use in the realization of this practice at the LCD level. We did the writing of reusable code (drivers) for certain functions of our application, avoiding having to make the activations in each Project. Then we could see the application of the UART communication protocol for the use of our peripherals and the process of data obtained by the sensor sent by this communication to the LCD, taking into consideration the code processes of timers and flags necessary for the proper functioning of the Program in when to synchronization and the handling of the data after the conversions and calculations that were necessary to do.

**References:**

* Wiring Pi, wiringpi.com/the-gpio-utility/.The GPIO utility | Wiring PiCommand line utility to manipulate the Raspberry Pi's GPIO pins.wiringpi.com
* “LM35 (ACTIVE).” LM35 ±0.5°C Temperature Sensor with Analog Output with 30V Capability | TI.com, www.ti.com/product/LM35.
* “Embedded Microcontroller Interfacing for M-COR ® Systems.” Google Books, [books.google.com.mx/books?id=oT7V4awRnewC&pg=PA291&lpg=PA291&dq=PIT%2Bmicrocontroller&source=bl&ots=qbahJw\_AGe&sig=iHlY33I6TaLms\_nSPHkC4nBsHU4&hl=es-419&sa=X&ved=0ahUKEwim89rDg7rVAhWDQSYKHa86AS8Q6AEIRDAE#v=onepage&q=PIT%20microcontroller&f=false](https://l.facebook.com/l.php?u=http%3A%2F%2Fbooks.google.com.mx%2Fbooks%3Fid%3DoT7V4awRnewC%26pg%3DPA291%26lpg%3DPA291%26dq%3DPIT%252Bmicrocontroller%26source%3Dbl%26ots%3DqbahJw_AGe%26sig%3DiHlY33I6TaLms_nSPHkC4nBsHU4%26hl%3Des-419%26sa%3DX%26ved%3D0ahUKEwim89rDg7rVAhWDQSYKHa86AS8Q6AEIRDAE%23v%3Donepage%26q%3DPIT%2520microcontroller%26f%3Dfalse&h=ATPPjzJpAu9BKuwjADmS3ymb5-V8STnScNBtr5aGTGxqrwCHaClIgI7kNPQUEgFC23BCqE_fOeceqeXW_t-c_KuIoJRXjke1pQ2URDjhggaCI-ed8RrBzR_ARjxCWFNuPNCnVlOkv20j87g).
* KL-25 Sub Family Reference Manual.