[1]

BASIC MANAGEMENT OF THE INSTRUCTION SET (August 2021)

Juan Camilo Serrano [1], IEEE member

Summary - For this lab, we are trying to implement firmware in an 8-bit Microchip microcontroller in order to experiment with the input / output pins, to know the peripherals and settings of the instruction set. Different exercises are performed such as reproducing acoustic, mechanical and optical variables with the assistance of a microcontroller in order to explore the potential of microcontrollers as embedded systems.

Index of Terms - Microcontroller, PIC16f887, Input and output ports, Instruction set, Delays, Watchdog timer.

I. INTRODUCTION

The laboratory practice of the microcontrollers subject is carried out, which consists of a series of exercises that aim to reinforce all the topics addressed in class.

As the handling of the set of instructions to enable and disable the input and output ports, deactivate internal resistors, perform mathematical and logical operations and store them in memory, creating delays of different times from the microcontroller's clock frequency. All this applied to projects such as managing a stepper motor, use of buttons, sequencing of led lights, data visualization on a 7-segment display and even generation of musical notes.

II. LABORATORY DESCRIPTION

A. Assembly of the hardware necessary for the laboratory activity.

For the development of the practice, the circuit that appears in figure 1 must be built, from this system, different source codes must be elaborated that allow the microcontroller to load the appropriate firmware for each proposed exercise.

In figure 1, the outputs M1, M1, M3 and M4 command an H bridge or a group of transistors connected to the coils of a unipolar stepper motor, it is recommended to use a unipolar motor, but it can also be bipolar (the configuration transistors changes). The number of steps per turn and the supply voltage of the stepper motor coils is free, depending on the availability of elements that each group has. Also, figure 1 shows the diagram of an audio system connected to pin RE2, which allows the microcontroller to generate acoustic indications for the user.

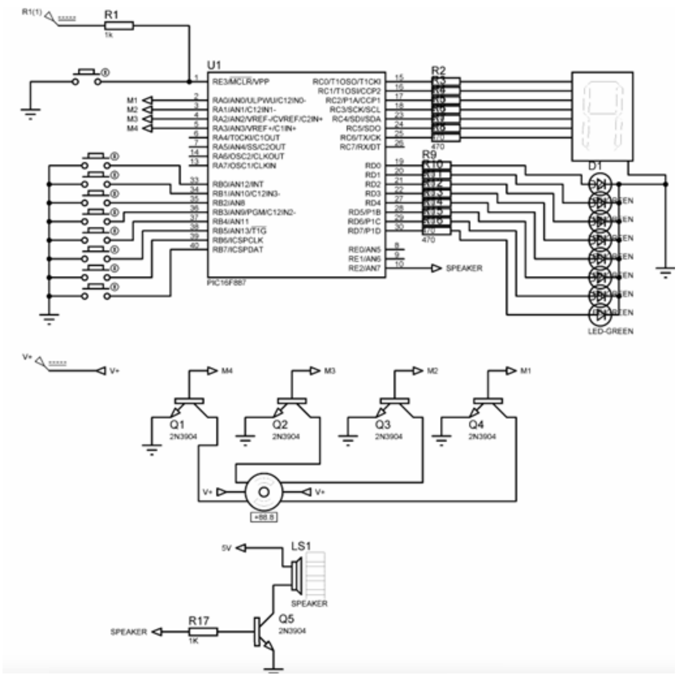


Fig. 1. Hardware necessary for practice.

B. Programmed sequence for stepper motor.

a) Make a firmware that controls the stepper motor, it must be able to rotate clockwise or counterclockwise indefinitely, the variation between clockwise or counterclockwise must be done by means of a button.

b) The display should show in hexadecimal basis, the number of revolutions that the motor has completed

C. Watchdog timer.

For the previous point, activate the Watchdog timer, the default system should prevent the microcontroller from restarting using the CLRWDT [2] [3] instruction. A pushbutton must be configured in such a way that when activated the microcontroller enters an infinite loop and no longer controls the motor, then the microcontroller must be reset by action of the Watchdog timer.

D. Alphanumeric characters on display.

a) Design a non-contact button or push button (the use of an

[1]Pedagogical and Technological University of Colombia (e-mail: juan.serrano01@uptc.edu.co).

infrared emitter and receiver is suggested).

b) It is required that each time the previously designed button is pressed, all the letters corresponding to the student's name appear on the 7-segment display.

E. *Light sequencer.*

A light sequencer of at least 8 light sequences is required, these will be shown on the LEDs and will be exchanged by the 8 buttons, in the way that is considered most convenient, the number of the sequence that is being executed should appear on the display. In addition, you must have push buttons to increase or decrease the speed of the current sequence.

F. *Electric piano.*

Using the audio circuit in figure 2, develop an electronic piano, where each push-button connected to port C is a key, so each time a key is pressed, a different musical note must be obtained through the speaker..

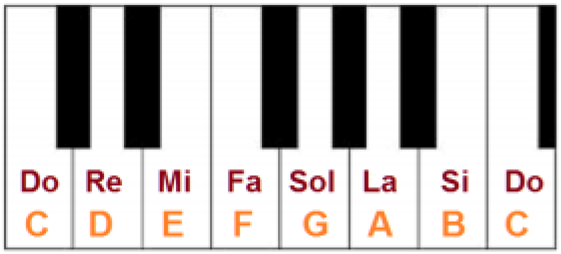


Fig. 2. Tone distribution for each pushbutton.

G. *Application*

Design a firmware for a real system of your choice based on the circuit in figure 1 (stepper motor, acoustic indication, 7 segments, LEDs, pushbuttons).

Only the firmware is required, it is not necessary to build any prototype or model structure, the operating logic will be evaluated. Therefore, if your system requires sensors, it must be simulated using pushbuttons or DIP switches, in the case of an actuator this must be reflected in the stepper motor.

Examples: An elevator, a vending machine, a car turnstile, a digiturn, electronic locks, etc.

III. METHODOLOGY

For the development of the guide, first the circuit of figure 1 was implemented in the proteus simulator, see figure 3, in order to verify the correct operation, before its respective implementation in the protoboard.

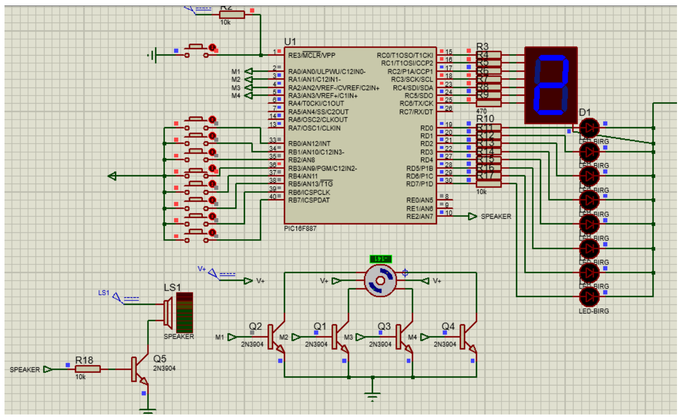


Fig. 3.Implementation of the circuit of figure 1 in proteus.

A. *Assembly of the hardware necessary for the laboratory activity.*

For the assembly of the circuit a double breadboard is required due to the large number of components.

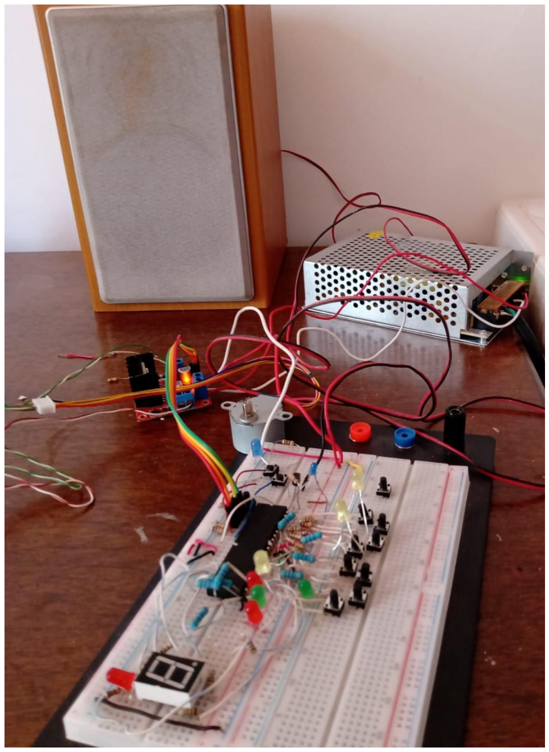
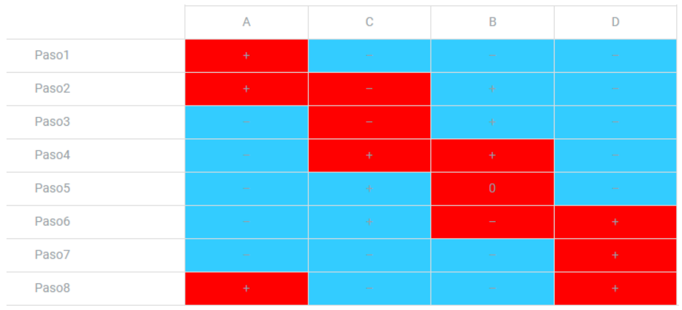


Fig. 4. Physical implementation of the circuit in figure 1.

B. *Programmed sequence for stepper motor.*

The stepper motor consists of 5 electromagnets around the armature, when current circulates through each of the electromagnets a magnetic field is created that attracts the rotor of the motor, if a correct and orderly energization of the electromagnets is carried out, movement is generated with great precision. [2]
The sequence used was the medium-pass sequence, which is illustrated in Figure 5.

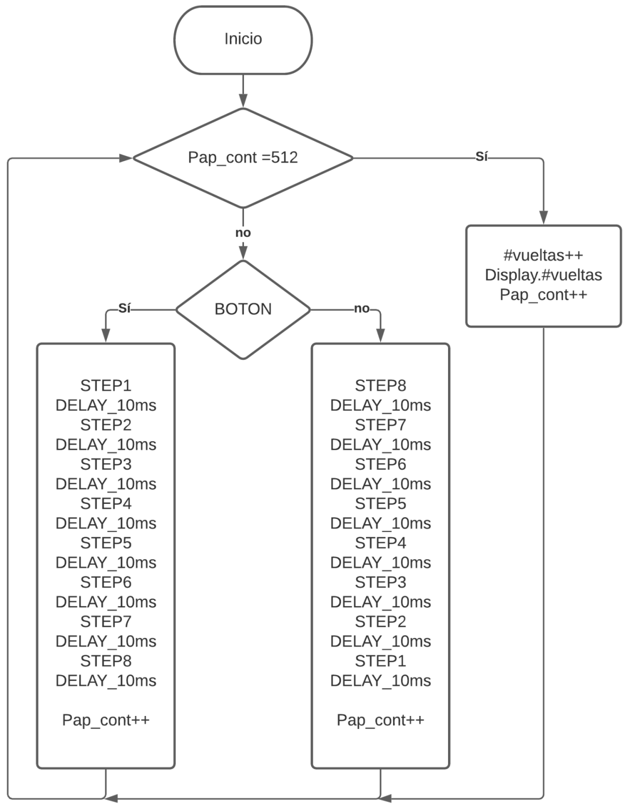


	A	C	B	D
Paso1	+	−	−	−
Paso2	+	−	+	−
Paso3	−	−	+	−
Paso4	−	+	+	−
Paso5	−	+	0	−
Paso6	−	+	−	+
Paso7	−	−	−	+
Paso8	+	−	−	+

Fig. 5.Mid step sequence of a pap motor.[2]

For the PAP motor to complete one revolution, the sequence of 8 steps must be executed 64 times, that is, if each step is executed in intervals of 20 ms, the time it takes for the stepper motor to make a 360 ° rotation would be:

$8 * 8 * 64 * 20ms = 4095 * 20ms = 81.92ms$ (1)

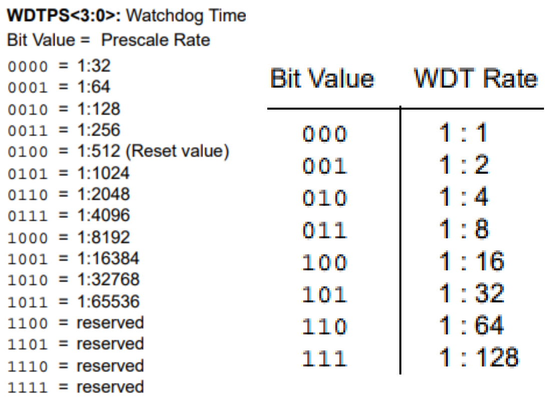


C. *Watchdog timer.*

The watchdog timer is a descending counter that when it reaches zero forces the microcontroller to restart, with the aim that if the system does not work correctly when it restarts, it will recover its functionality. The WDT has the possibility to let the developer choose how often he wants the WDT to be activated, this depending on the application, and in which part of the code he wants this counter to be reset, in this way he makes sure that if the system is WDT working properly does not interfere with this process.

In addition, the WDT has a pre-scale, to expand the counter. In this case, we work with a WDT at 31 Khz, the pre-scale comes by default of 1: 512, that is, 512 are needed

$\frac{1}{31\,KHz} * 512 = 16.51ms$ (2)



WDTPS<3:0>: Watchdog Time
Bit Value = Prescale Rate

0000	= 1:32
0001	= 1:64
0010	= 1:128
0011	= 1:256
0100	= 1:512 (Reset value)
0101	= 1:1024
0110	= 1:2048
0111	= 1:4096
1000	= 1:8192
1001	= 1:16384
1010	= 1:32768
1011	= 1:65536
1100	= reserved
1101	= reserved
1110	= reserved
1111	= reserved

Bit Value	WDT Rate
000	1 : 1
001	1 : 2
010	1 : 4
011	1 : 8
100	1 : 16
101	1 : 32
110	1 : 64
111	1 : 128

Fig. 15. WDTCON and OPTION_REG registers respectively from the WDT prescaler.[5]

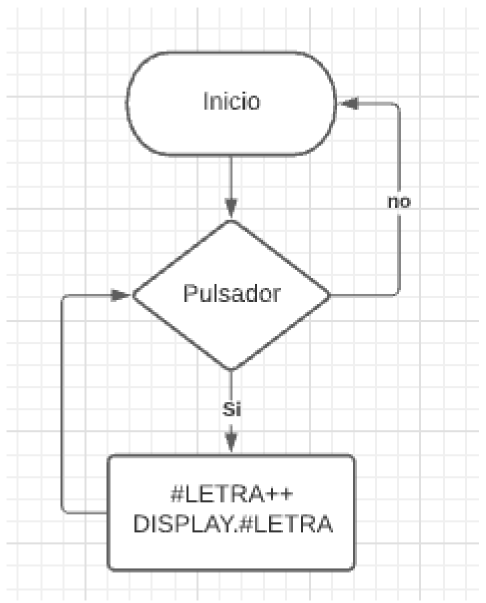
As we are handling delays of at least 20 ms, it is necessary to handle WDT counters higher than this level, otherwise it would imply having to redesign the delays to include the CLRWDT instruction within the loop. This application does not require such short times, so it is chosen to use a count of 66 ms, as a result of having the prescaler of the OPTION_REG 1: 4 and the WDTCON in its default value of 1: 512.

$\frac{1}{31\,KHz} * 512 * 4 = 66\,ms$ (3)

Where it only remains to execute the CLRWDT instruction after each delay.
In any case, it is necessary to test the operation of the WDT. Pin 34 of the microcontroller is available to receive a high logic level to enter an infinite loop, where it is only possible to exit by restarting the program.

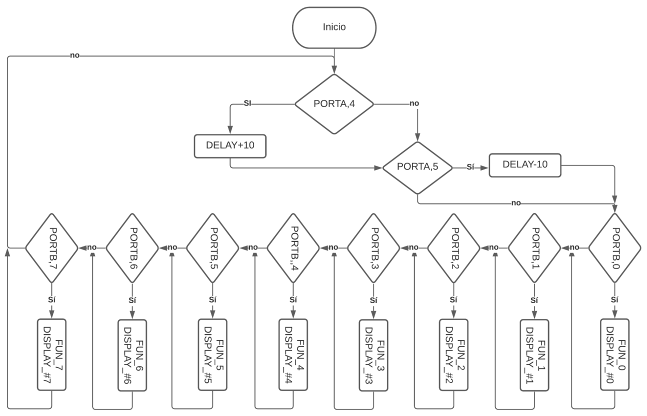
D. *Alphanumeric characters on display.*

For this point, a list was used to store all the information required to display the message JUAN CAMILO SERRANO CORREA, on the 7-segment display, letter by letter each time the contactless button is pressed.



E. *Light sequencer.*

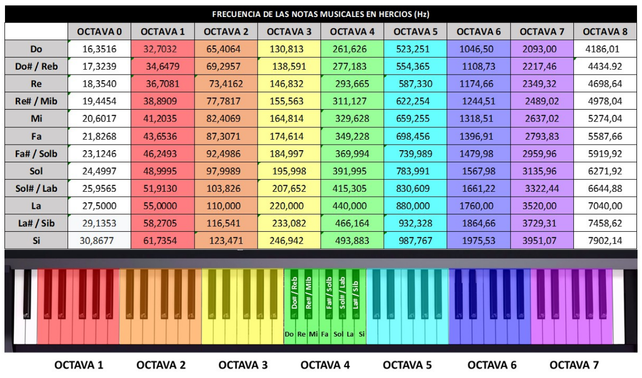
For the development of this point, each of the 8 sequences are in a different function, at the moment that any button of port b is pressed, a specific sequence is executed. Additionally, it has 2 extra buttons located on pins RA4 and RA5 that, when pressed, increase or decrease the sequencing speed of the LEDs, it is worth mentioning that as it was not specified at what amount of interval it had to be increased, this was left to chance. Data also does not specify the number of times that it can increase, that is, the sensitivity, it was left too low, that is, the variable that contains the variable delay increases or decreases, as the case may be, until it overflows and starts again in a similar starting point.



F. *Electric piano.*

For the development of the electronic piano, a square signal is sent where its frequency determines the musical note, for this project we work on octave 4, the onlinemic test software was used as a standard unit, to confirm that the musical note is correctly calibrated.

Below is evidence of the frequency captured by the software based on the sound emitted by the speaker.



FRECUENCIA DE LAS NOTAS MUSICALES EN HERCIOS (Hz)

	OCTAVA 0	OCTAVA 1	OCTAVA 2	OCTAVA 3	OCTAVA 4	OCTAVA 5	OCTAVA 6	OCTAVA 7	OCTAVA 8
Do	16,3516	32,7032	65,4064	130,813	261,626	523,251	1046,50	2093,00	4186,01
Do# / Reb	17,3239	34,6479	69,2957	138,591	277,183	554,365	1108,73	2217,46	4434.92
Re	18,3540	36,7081	73,4162	146,832	293,665	587,330	1174,66	2349,32	4698,64
Re# / Mib	19,4454	38,8909	77,7817	155,563	311,127	622,254	1244,51	2489,02	4978,04
Mi	20,6017	41,2035	82,4069	164,814	329,628	659,255	1318,51	2637,02	5274,04
Fa	21,8268	43,6536	87,3071	174,614	349,228	698,456	1396,91	2793,83	5587,66
Fa# / Solb	23,1246	46,2493	92,4986	184,997	369,994	739,989	1479,98	2959,96	5919,92
Sol	24,4997	48,9995	97,9989	195,998	391,995	783,991	1567,98	3135,96	6271,92
Sol# / Lab	25,9565	51,9130	103,826	207,652	415,305	830,609	1661,22	3322,44	6644,88
La	27,5000	55,0000	110,000	220,000	440,000	880,000	1760,00	3520,00	7040,00
La# / Sib	29,1353	58,2705	116,541	233,082	466,164	932,328	1864,66	3729,31	7458,62
Si	30,8677	61,7354	123,471	246,942	493,883	987,767	1975,53	3951,07	7902,14

Fig. 6.Frequency of musical notes in Hz.[3]

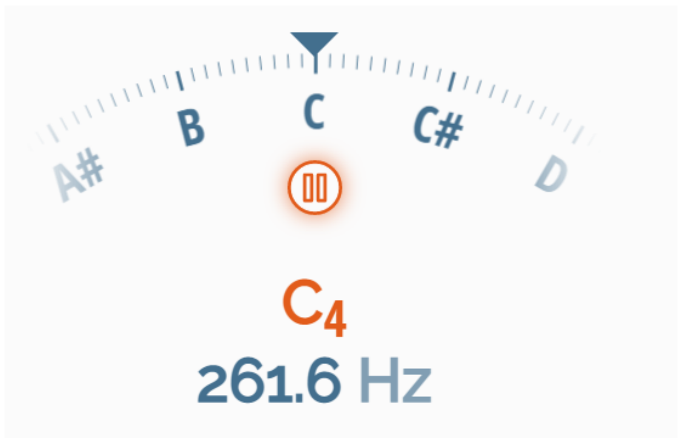


Fig. 7.Musical note C 4 according to onlinemictest.[4]

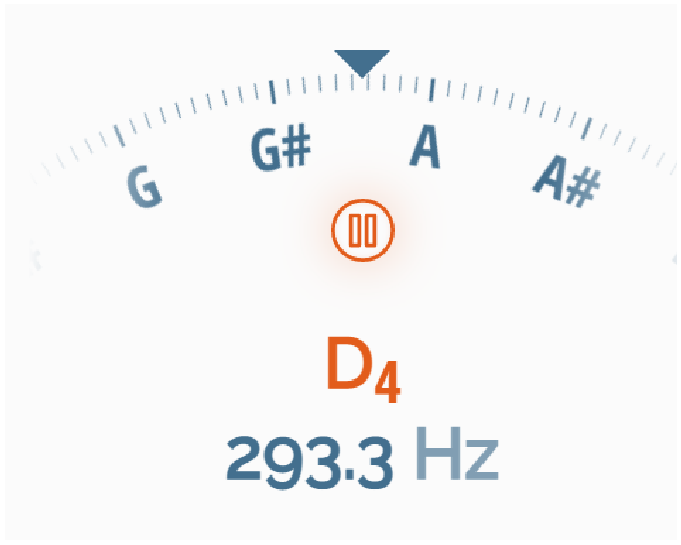


Fig. 8.Musical note D 4 according to onlinemictest.[4]

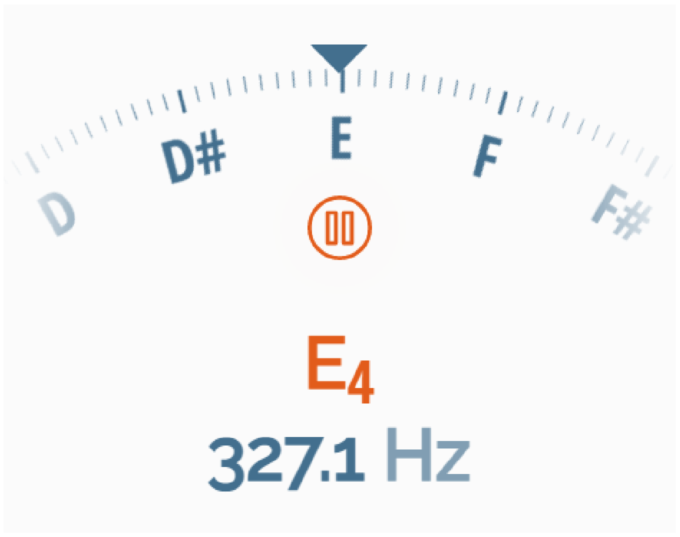


Fig. 9Musical note E 4 according to onlinemictest.[4]

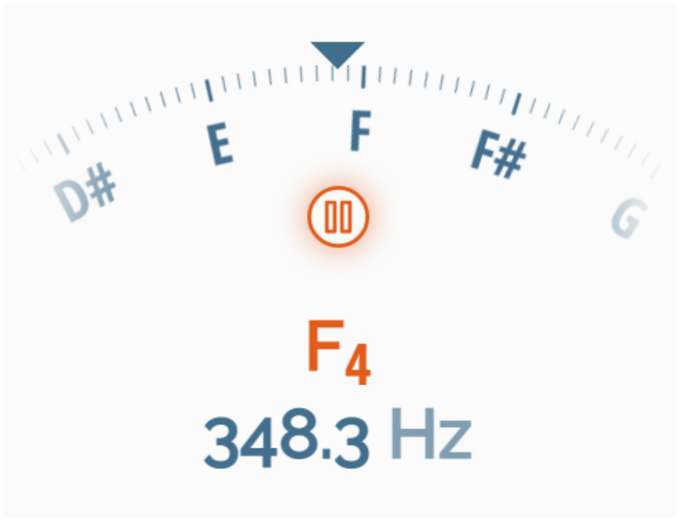


Fig. 10.Musical note F 4 according to onlinemictest.[4]

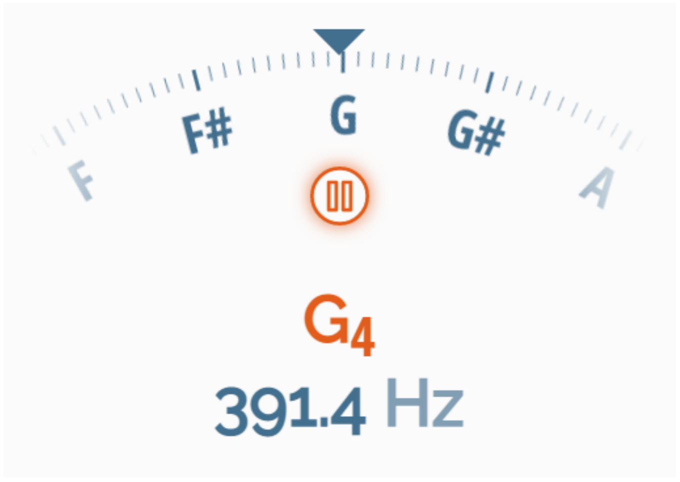


Fig. 11Musical note G 4 according to onlinemictest.[4]

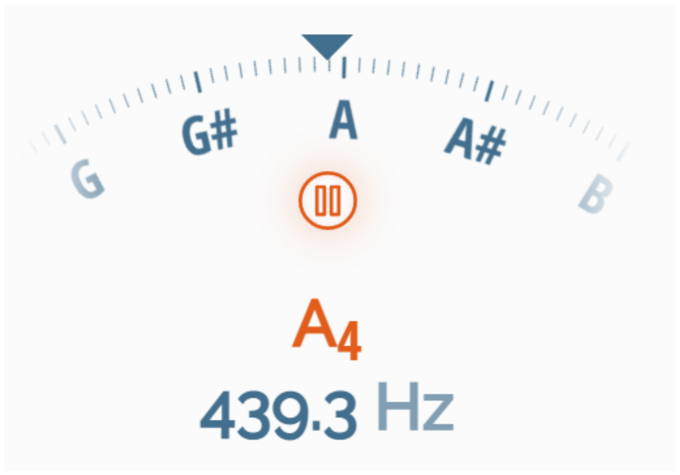


Fig. 12.Musical note A 4 according to onlinemictest.[4]

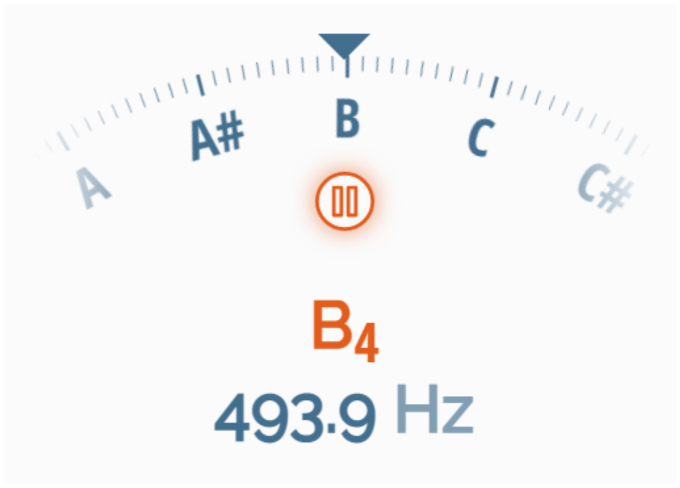


Fig. 13.Musical note B 4 according to onlinemictest.[4]

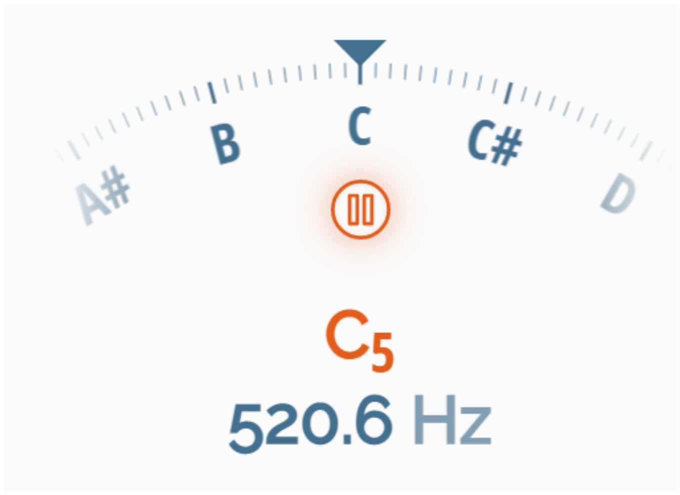
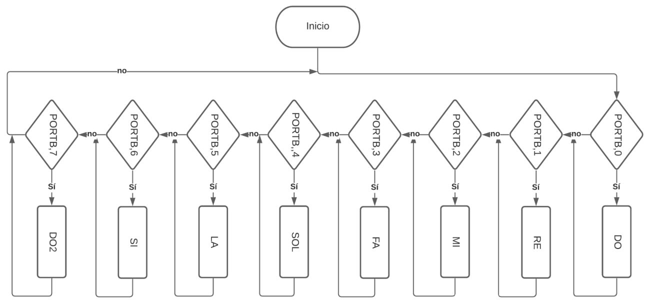


Fig. 14. Musical note C 5 according to onlinemictest.[4]



G. Aplicación.

The pap motor simulates being the motor of an elevator, the LEDs indicate how long it will take to get to the next floor. The 7 segment display shows the floor you are currently on, suddenly the elevator starts to fail and goes up too fast, the passengers panic !!! There is an emergency button, when the elevator is pressed, an electric piano is immediately enabled so that the person inside the elevator can generate an SOS melody. There is a secret piano key that frees people from the elevator, can they find it? If they find her, the elevator doors will open.

IX. CONCLUSION

Although assembly language occupies less memory, the development of some algorithms have an extra complexity, such as the manipulation of data with a higher quantity of bits than the microcontroller, in this case data of more than 255.

It is evident how unreliable the proteus simulator for microcontrollers is, since many errors that occurred in the implementation were never evident in the simulation.

References

[1] https://datasheetspdf.com/pdf/1006817/Kiatronics/28BYJ-48/1.
[2] https://www.diarioelectronicohoy.com/blog/motores-pap-unipolares
[3] https://1.bp.blogspot.com/-NQaGhJX-ix8/XjAEtqBkYpI/AAAAAAAABl8/dcDekwmy2H0wIvs-x5AJvC2RRcecrlbgACLcBGAsYHQ/s1600/FRECUENCIA_NOTAS_MUSICALES.jpg
[4] https://www.onlinemictest.com/es/sintonizadores/pitch-detector.
[5] https://pdf1.alldatasheet.com/datasheet-pdf/view/197543/MICROCHIP/PIC16F887.html