# Sistemes Encastats

# Memòria de les Pràctiques

#### Realitzat per: Adrià Auguets i Pavel Macutela. 29 de Maig del 2017

# Introducció

En les pràctiques de Sistemes Encastats hem realitzat un detector de DTMF utilitzant l’algoritme de Goertzel. La gràcia d’aquestes pràctiques ha estat experimentar quins són els resultats que podem arribar a obtenir si implementem el DTMF en tres dispositius diferents.

Els dispositius que vam tenir oportunitat de programar, van ser:

- Un microcontrolador Arduino.

- Una FPGA (DE0-Nano).

- Un Sistema Operatiu, el nostre portàtil i una Raspberry Pi 3 Model B.

En tots aquests dispositius hem detectat avantatges i inconvenients en el seu ús, tots els resultats es podran comparar més endavant.

## DTMF - Realitzat amb el Microcontrolador Arduino.

La primera pràctica es va realitzar sobre l’entorn Arduino. Com sempre tota primera tasca té les seves complicacions i aquesta no en va ser l’excepció. Nosaltres abans d’assistir a la primera pràctica vam realitzar un estudi previ per familiaritzar-nos amb el que era el **Dual-Tone Multi-Frequency signaling i l’Algoritme de Goertzel**.

Un cop assolits aquests coneixements i de fer unes primeres proves en Octave, vam començar a realitzar el nostre codi.

**DTMF.c :**

El nostre mòdul principal, nosaltres considerem que el tenim ben estructurat i fàcil d’entendre. Així com, estructurem el codi de manera que en les primeres línies, com es de costum, tenim els *includes* i les declaracions de les variables amb un nom fàcil d’entendre. Tot seguit es poden veure les funcions que usem i després de les funcions hi trobem el main. Al final del programa, sempre hi deixem lloc per la Interrupció.

Els coeficients que fem servir en l’script els hem calculat prèviament amb Octave. Ho podem veure en la Font 1.

Nosaltres, comencem a tractar les mostres dins la interrupció fins que en tenim **205**. Llavors, en **calculem la potencia utilitzant vuit filtres** i directament anem a una màquina d’estats. Aquesta màquina d’estats, primer s’encarrega de detectar quin número o lletra esta sonant o al contrari, no sona res. Després el mostra per pantalla segons toqui segons la màquina d’estats.

Creiem que un apartat que ens diferència de la resta, es la manera que nosaltres detectem el resultat, és a dir, a partir de quines freqüències estan sonant, saber a quin botó fan referencia. Nosaltres només definim *“resultats[16] = {'1','2','3','A','4','5','6','B','7','8','9','C','\*','0','#','D'};”* i per saber a quina posició de la llista resultats fan referencia les dues freqüències nosaltres hem creat la funció, **uint8\_t detectaResultat(void)**, Font 2.

Aquesta funció mira quines són les freqüències baixes que superen el llindar i actualitza la variable *posició* (+= i\*4) i es posa a true un boleà, depenent de quina superi el llindar. Després fem el mateix amb les altes( += i-4) i també es posa a true un boleà. Alhora per a cada freqüència que superi el llindar, es suma 1 a la variable trobat.

Finalment comparem si els dos booleans donen true i que el valor de trobat es igual a dos. Si això es compleix, anem a buscar el valor que ocupa la posició “*posició*” a la llista resultats.

Per acabar, vam comprovar amb l’osciloscopi que l’Arduino tingués temps per realitzar els càlculs.

Que hem aprés en aquesta pràctica d’Arduino?

Com a coneixements que hem assolit realitzant-ho a un microcontrolador, podem destacar que s’ha de tenir sempre en compte els temps d’execució a la hora d’implementar el codi.

També hem aprés a fer servir “càsting” i com tractar amb valors que són decimals, transformant-los en enters, fer càlculs i després tornar a tenir el resultat com si haguéssim fet el càlcul amb floats però amb un temps molt més ràpid.

## DTMF - Realitzat amb una FPGA.

La segona part de la pràctica com hem comentat anteriorment, utilitzem la FPGA (un dispositiu hard amb més recursos que l’arduino), per tal de veure quin seria el resultat obtingut de la mateixa implementació del DTMF.

Per aquesta pràctica utilitzem la FPGA DEO-nano, llenguatge VHDL i Quartus per poder compilar i carregar el programa dintre de la FPGA. També farem servir l’Arduino ja que la FPGA no te ADC.

Per crear el programa de la FPGA utilitzarem tres capes, la primera per calcular l’Algoritme de Goertzel, la segona per descodificar el caràcter i la tercera que serà la màquina d’estats que vam implementar a la pràctica anterior.

Ens proporciones el mòdul «BP\_V0» que conté el algoritme de Goertzel (només detecta la freq. De 697 Hz) el qual hem agut de modificar afegint la la linia :

«generic(a\_int: integer range -33768 to 32767 := 436) »

On 436 és un dels coeficients calculats amb Octave, es veu en la Font 1 y de replicar-lo tantes vegades com filtros necesitem cambiant el valor de la «a\_int» per tal de calcular la potencia de la señal en cada filtre y saber quina a quina freq. pertany el to.

La sortida d’aquests filtres es una senyal (enable) std\_logic (1 ó 0) van a parar a la entrada de un altre mòdul implementat per nosaltres «codificador.vhd» el qual te 9 entrades ( 8 sortides del filtres y un clock) y 2 sortides (un bus de 5 bits y un enable), aquests mòdul simplement agafem les entrades y depenen de la entrada un dels 16 valors que ens interesan el posem a la sortida y activem el enable, es a dir que tindrem un numero que hem detectat.

El tercer mòdul és on implemeten la maquina d’estats, (esperem dada, o esperem silenci) si la entrada enable del mòdul anterior es 1 y estem esperan una dada, si la entrada es un silenci o un valor indeterminat ens quedem en el mateix estat (esperant dada) y desactivem la sortida enable; si es un valor que ens interesa agafem les dades de entrada (de menys pes) y la posem a cada una de les cuatres sortides y activem l’enable y pasem a l’estat de esperm un silenci.

Més resumit ja que necessitem mes conclusions.

Aquestes 5 sortides seran l’entrada de l’Arduino, on sempre està mirant l’estat del pin on esta connectat l’Enable, quan aquesta esta activada, llegim els pins on estan connectats els altres cables, descodifiquem i enviem pel port sèrie el caràcter, on mostrarem pel terminal quin caràcter és.

Què ens diferencia?

Lu del enable que dura poc però es suficient ja que el temps que necessita l’arduino per detectar un senyal es mes petit del que arriba a durar l’enable....

Què hem aprés?

Doncs que l’arduino pot no detectar enables si aquest es molt petit, a utilitzar capes(moduls).

## DTMF - Realitzat amb dos Sistemes Operatius.

La darrera pràctica consisteix en realitzar un DTMF en l’entorn de un Sistema Operatiu. El nostre primer objectiu va ser traduir el codi C que vam fer en la primera pràctica a Python. El nostre codi Python seguia la mateixa estructura i idea que el codi C. L’avantatge que vam tenir, és que, en Python no havíem de declarar variables segons el tipus, o utilitzar casts.

A diferencia d’altres grups nosaltres vam realitzar dos programes Python. Un fet amb bucles i l’altre sense bucles. Això ho vam fer per tal de demostrar que sense bucles el programa Python realitzava més ràpidament els càlculs.

Com a resultat de fer els primers experiments en el Sistema Operatiu Mac, podem veure que en el Python amb bucles tarda uns 0.02459192276 segons en tractar un carácter. En canvi, utilitzant el programa Python sense bucles, veiem que com a resultat en tractar un carácter tarda uns 0,024068832 segons.

En Raspberry Pi 3 Model B, utilitzant el Python amb bucles podem veure que a tractar un carácter tarda uns 0.0280151367188 segons. En canvi sense bucles tarda 0,024008989 s.

A continuació podem veure una taula amb una mitjana de resultats que hem obtingut.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test 1 | | Test 2 | |
|  | bucles.py | NoBucles.py | Bucles.py | NoBucles.py |
| Ordinador | 0,024623856 s | 0,025061801 s | 0,025570512 s | 0,025570667 s |
| RPi3 Model B | 0,026891038 s | 0,026989803 s | 0,025591922 s | 0,025608265 s |

Taula de mitjanes de temps

PER ACABAR DESPRES DE PRESENTAR.

CONCLUSIONS (ELS DOS)

Diferencies entre els 3

Perquè A B o C depenent de que ens demanin.

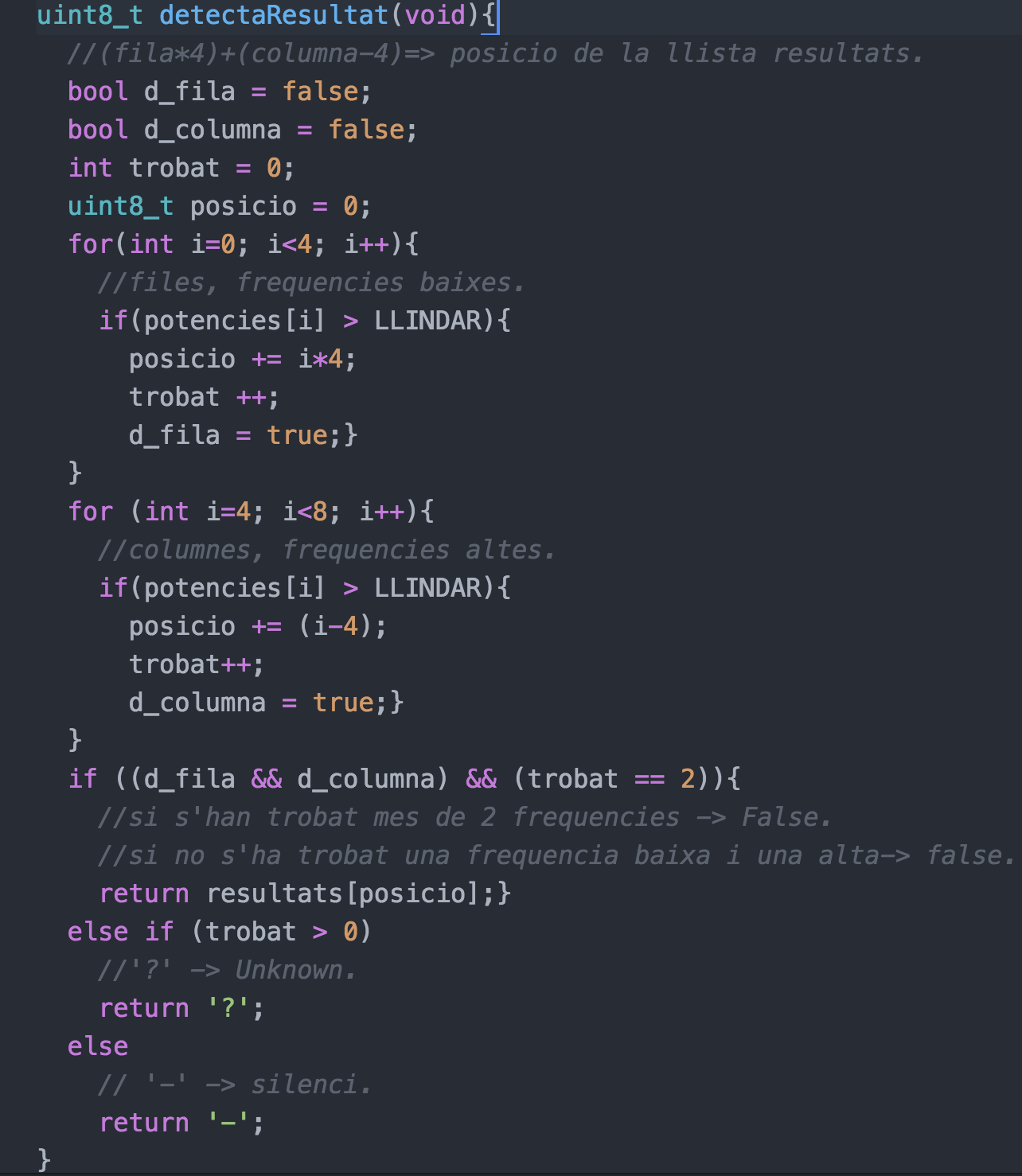
Quin seria el mes fàcil de fer? Optim?

Si vulguessim fer 2 canals quins dispos podríem fer servir? (aquesta no la posem però la pensem)

Imatges:



Font 1: Mostra del càlcul dels Coeficients amb Octave.



Font2: Funció detectaResultat.