Index

Introducció

La idea del projecte és fer un anemòmetre que pugui donar la velocitat del vent de manera econòmica i senzilla.

Per a la lectura d'aquest anemòmetre es farà ús de circuit electrònic per adaptar-lo a la raspberry i que pugui ser interpretat mitjançant codi python.

Donat que els anemòmetres són instruments cars, d'entra 80 i 200 €, s'ha ideat el projecte pensant en fer servir un ventilador de tres cables, els quals, corresponen a l'alimentació i un tercer senyal del tacòmetre.
La intenció és profitant-ne el moviment de les aspes, obtenir un valor o bé en volts generats pel moviment del motor de continua o bé pels pulsos enviats pel tacòmetre (sensor d'efecte hall) del qual disposa alguns d'aquests dispositius.

Materials

El materials principals d'aquest projecte sense tenir en compte els que s'han fet servir per comprovacions puntuals.

- Ventilador 8x8 de 12v EVERCOOL Model EC8010LL12EA.

- Circuit operacional LM324N.

- Convertidor Analògic / Digital ADC0804LCN.

- Placa per soldar components

- Resistències 3 de 10K, 1 d'1K, 1 d'1K1.

- Condensador de 10nF.

- Altres materials per fer proves, potenciòmetre de 10k línial, 8 leds, 8 resistències de 220 ohms. Font d'alimentació 5v.

Descripció del projeccte

Primera etapa circuit ventilador-amplificador operacional:

Degut al l'elevat preu dels dispositius per mesurar la velocitat del vent que hi al mercat s'ha optat per fer servir un ventilador de panel estandard de 12v 0.12A. Amb detector de gir mitjançant detector efecte hall. EVERCOOL Model EC8010LL12EA. Preu aprox: 3,40€.

La idea era captar el senyal del sensor d'efecte hall, agafant els pulsos que entrega en girar a mode de PWM, injectar-los a un circuit RC(resistència-condensador) i obtenir una tensió DC per després convertir-la amb un convertidor A/D a un valor binari que pogués interpretar la Raspberry.
El problema inicial ha estat que degut al circuit intern del ventilador no era possible treballar amb el sensor d'efecte hall sense alimentació i obligava a tenir alimentació als cables de corrent del ventilador generant el gir a través de l'alimentació totalment inacceptable.

En una primera solució havia pensat a tallar cables dels bobinats del motor però estudiant una mica més en profuncitat la situació, es va arribat a la conclusió que no aniria bé captar els pulsos del sensor hall perquè únicament augmenten o disminueixen la freqüència dels pulsos i no el duty cycle dels pulsos (amplada dels pulsos positius). A la pràctica amb un circuit RC i mesurant la tensió amb un polímetre no es veia una resposta prou satisfactoria.

En un segon intent s'ha aprofitat el corrent entregat pel propi motor (cables d'alimentació de motor) quan gira a mode de dinamo. S'ha comprovat que hi havia rangs de 0 a 300mV i fins i tot més en girar-lo manualment.

Degut a la única alimentació disponible de 5V, la idea és treballar amb aquest valor de tensió de referència per al convertidor A/D i aquest sent el rang de lectura del convertidor. Com que el valor entregat pel l'anemòmetre es de 0 a 300mV caldrà augmentar aquest senyal i caldrà per tant amplificar-lo amb un guany de 10.

Per tal d'obtenir un senyal de 0 a 3V, amb possibilitat d'arribar als 5V s'ha fet servir un amplificador operacional LM324N que té un cost baix de 0,55 €.

En càlcul del guany ve donat per les fòrmules següents:

$$Guany = Vout/Vin$$

Sent la tensió Vin donada pel ventilador d'entre 0 i 350mV (fins i tot pics de 500mV) i tenint un rang de lectura al convertidor ADC0804 de 0 a 5V (Vref) farem el següent calcul per obtenir el guany necessari a l'amplificador operacional no inversor:

$$5V/0,5 = 10$$

Per obtenir in guany de 10 a la hora de dissenyar el circuit operacional haurem de tenir en compte el valor de les resistències d'entrada R3 i realimentació R1:

$$Guany = 1 + (R1/R3) \qquad \text{(veure Fig. 1)}$$

$$Guany = 1\ (10K + 1K1)$$

Durant la pràctica s'ha pogut comprovar que amb l'alimentació de l'integrat LM325 a 5v, obtenim un màxim de sortida de 3,5V. Els rangs a mesurar quedaran per sota del 3v o 3,5v, si es volgues tenir un rang de mesura superior no cal alterar el circuit, simplement amb augmentar l'alimentació de LM324N a 12v n'hi ha prou.

Esquema circuit amplificació:

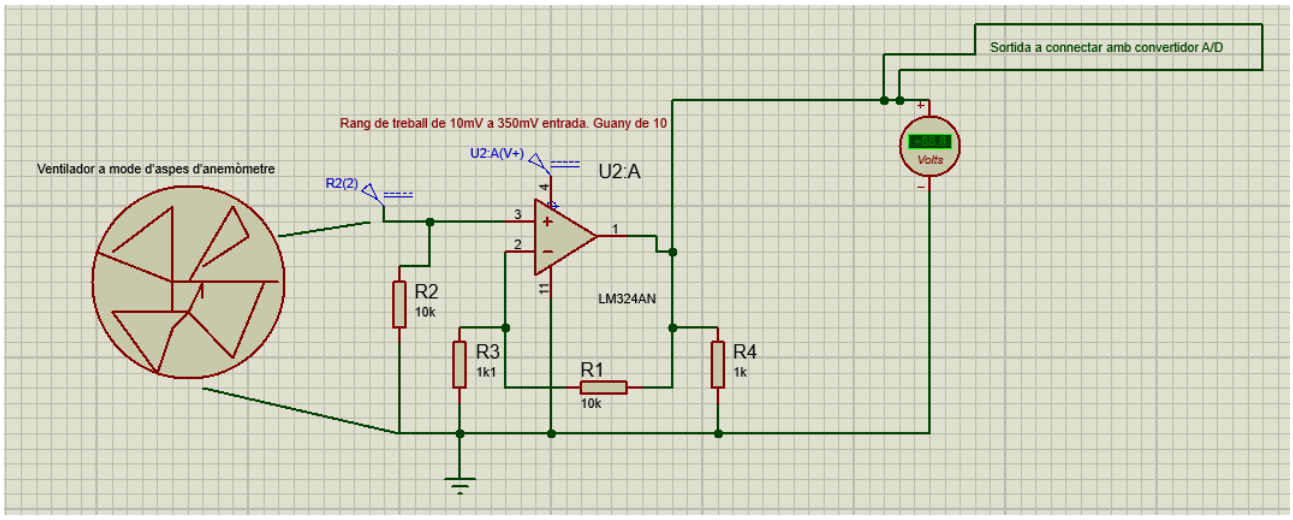

Figura 1.

Les resistències R2 i R4 són per evitar que hi hagi tensión residuals quan el gir estigui aturat.

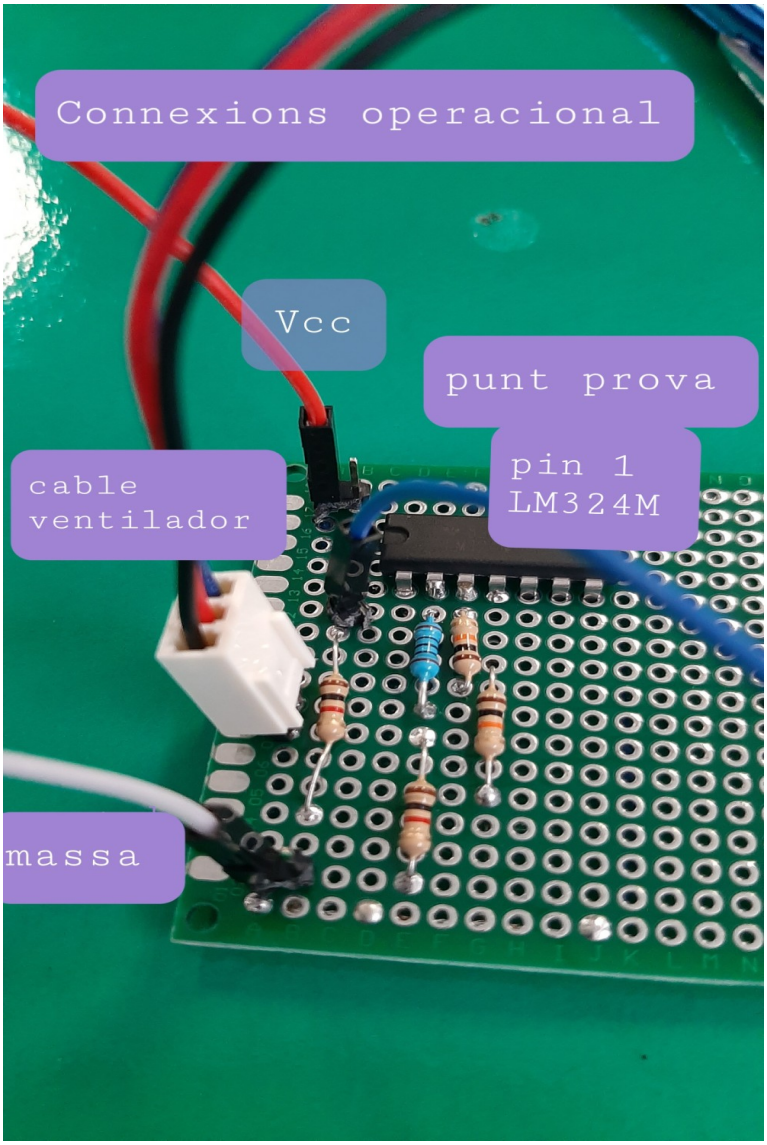

Figura 2

Segona etapa Convertidor-Shift register:

L'objectiu és convertir el senyal entregat en la primera etapa per l'operacional a un valor binary per poder ser interpretat per la Raspberry. La part principal d'aquesta etapa és un convertidor A/D de 8 bits model ADC0804.

Com s'ha comentat en l'apartat anterior, inicialment la idea del projecte era treure el senyal directament del ventilador (sensor d'efecte hall) i llegir-ne els pulsos a través d'una entrada GPIO a mode de PWN en lectura, però per llegir la longitud del duty cycle cal que la raspberry sigui molt precisa en el temps de lectura d'aquesta longidud. Per comentaris a internet s'ha abandonat aquesta via i el motiu ha estat que són necessiten temps molt baixos d'exploració i donat que la raspberry pot estar

ocupada en altres processos poden induir a error en la lectura i les recomanacions en la seva majoria eren fer servir un microcontrolador en aquesta etapa.
En aquest moment em vaig decantar per la opció del convertidor A/D amb sortida sèrie cap a una GPIO, però per preu i disponibilitat a les botigues dels dispositius al final vaig optar per triar un convertidor economic amb sortida de 8 bits paral·lel i convertir-los a sequència en sèrie mitjançant un Shitf register de prop de 50 cts. model 74HC165. Els dos components ADC0804 i 74HC165 no han tingut un preu superior a 6 €.

A l'inici, deixant aparcada la primera etapa amb el ventilador, es va procedir a dissenyar aquesta etapa de conversió, cosa que un cop acabada i représ amb el ventilador es va veure que s'havia de fer servir un operacional per tal de tenir nivells de volts a la entrada del convertidor i no de milivolts com es volia fer en un principi.

Els pins de convertidor ADC0804 en la seva majoria han de posar-se a massa perquè les seves funcions no caldran per a aquest disseny però cal fer esment que el pin 2 RD' ens permet controlar la sortida del component anulant-lo quan el posem a 1(5V). L'entrada que es fa servir per la lectura és Vin+ (pin 6), posant Vin- a massa.
Per tal de donar el rang de lectura tenim Vref (pin 9) de tal manera que, veient que l'entrega de tensió Vin+ que arriba de l'operacional és de 0 a 3,5 fins i tot algun volt més sense arribar als 5V s'ha decidit deixar aquesta referència a 5v, fent una divisió de 255 trams en un valor de 0 a 5v (veure Fig.6 i Pinout Pin 9). Això vol dir que per cada 19,5mV farem un increment d'un bit, sent el valor binari 00000000 – 0V , 00000001 – 20mV , 00000010 – 40mV…..010111001 – 3,5V….etc.

Una vegada obtingut la conversió del valor de tensió d'un llindar de 0 a 5V a binari de 00000000 a 11111111, cal que la raspberry en faci la lectura i la interpretació. Si l'enviéssim directament a les GPIO ho podria fer però ocupariem 8 pins GPIO. També es podrien descartar els bits de més pes i fer-ne servir sols quatre però fent servir un Shift register de baix preu no complicava la situació i comportava un petit repte de fer-lo servir en sentit contrari de com va ser l'ús del Shift register estudiat a la formació amb el 74HC565(complementari del que ens ocupa 75HC165).

Els pins del Shift Register 74HC165 són, 8 pins per entrada de registres, pin 2 rellotge, pin 1 (SH/LH') fet servir com a Latch, pin 7 (QH') sortidad de registres en sèrie. És important notar que aquest integrat entrega el valor d'entrada negat a la sortida, caldrà tenir-ho en compte al crear el codi: octet += str((GPIO.input(QH) +1)%2)
Per tal de donar sincronització entre la sortida de dades i la lectura cal els pulsos del rellotge, perquè surtin les dades al ritme del rellotge cal activar entrada del Lacth.
Els pins 9 i 10 no es fan servir i es poden deixar a l'aire i el pin 15 ha d'estar a massa perquè si no ho fem el resultat de sortida és sempre 00000000.

Per connectar sortida lógica del 74HC165 amb la GPIO cal tenir en compte que quan fem servir la GPIO com a entrada no podem suministrar més de 3,3V. La convinació que surt en sèrie pel 74HC165 són dades de 5V per tant cal posar una limitació amb una resistència de 2k2 entre porta lògica i GPIO i una altra de 3k3 entre GPIO i massa. També es pot fer servir un adaptador bidireccional 5-3,3v.

Esquema circuit Convertidor-Shift register:

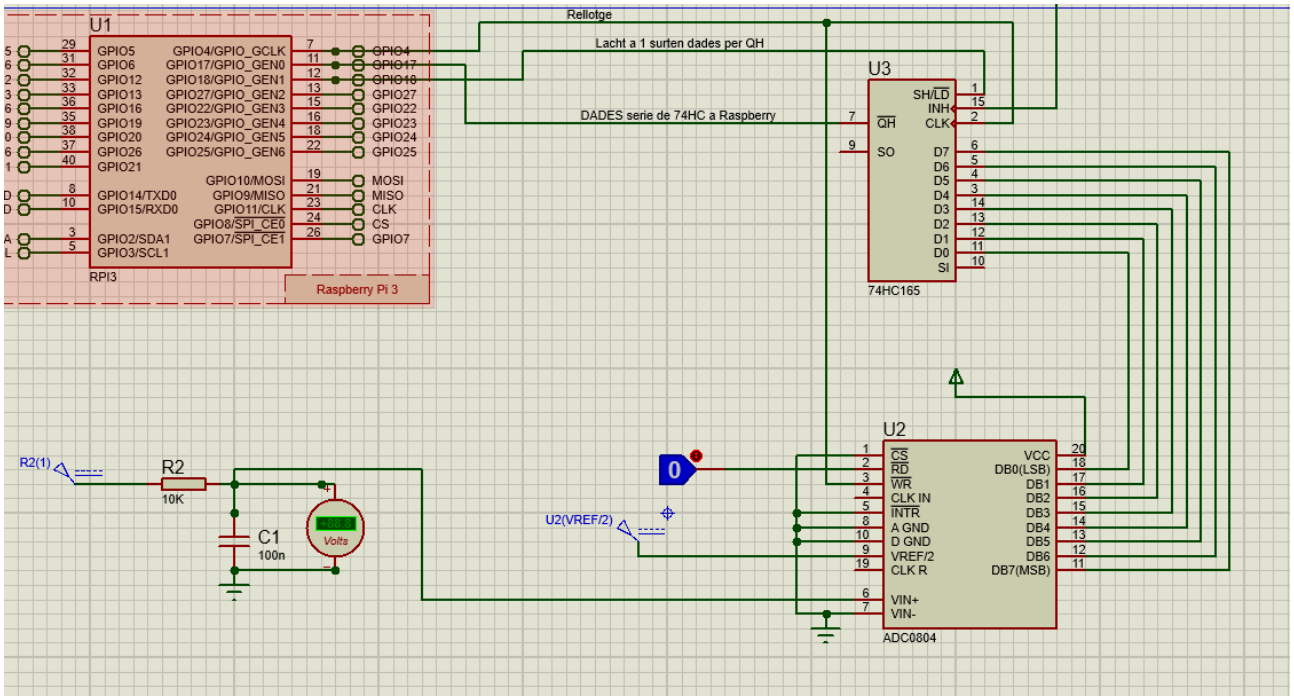

Figura 3

En aquest esquema de simulació de proteus es fa servir un cuit RC (R2 i C1) amb un voltímetre i un generador DC per tal de variar la tensió d'entrada que correspondria a la sortida de l'operacional.

Esquema teòric Proteus amb Operacional, Convertidor i Shift-Register:

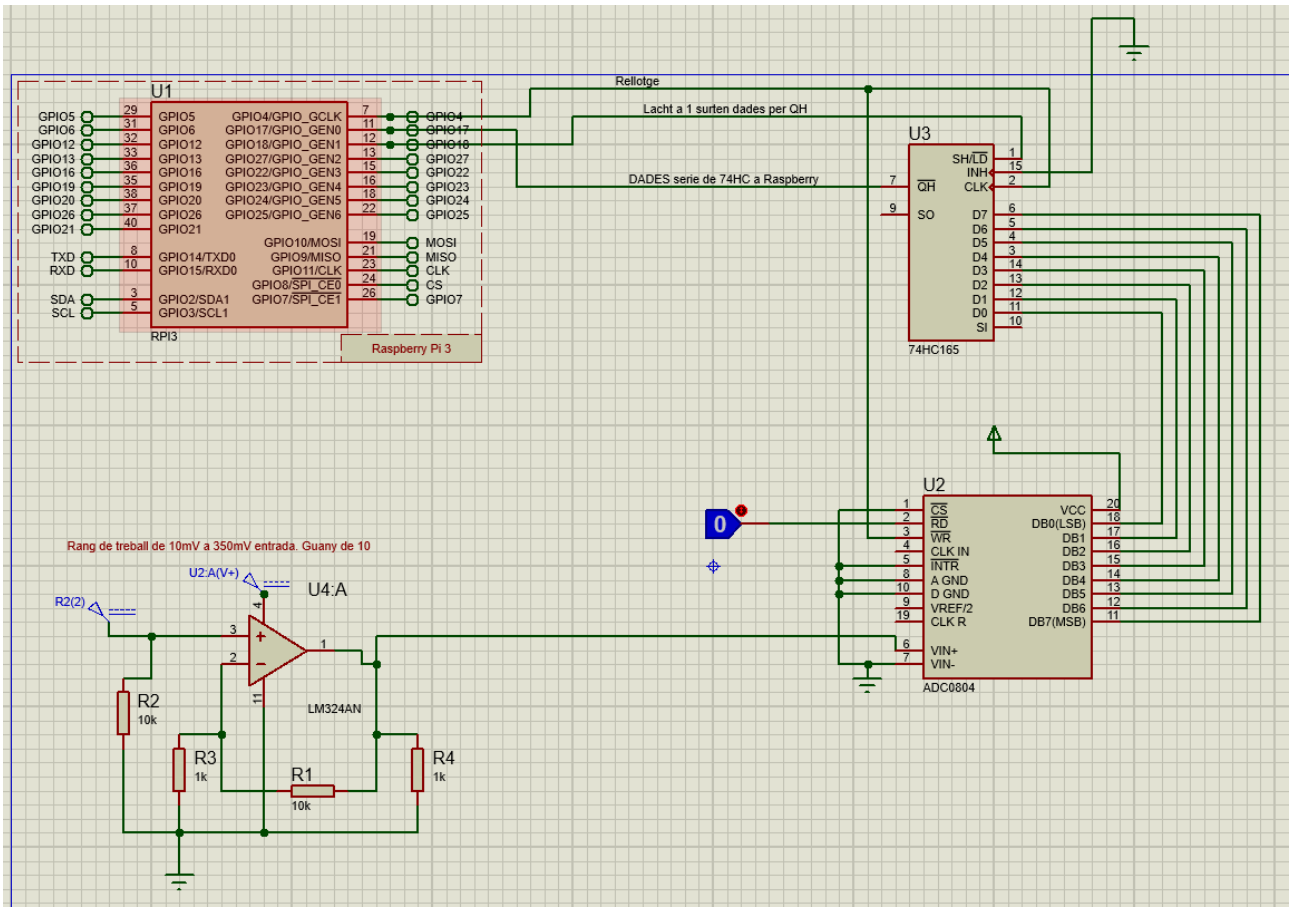

Figura 4

L’entrada 3 de l’operacional LM324N es connectarà directament al positiu del anemòmetre (ventilador).

Circuit pràctic sense el Shift Register fent ús de leds per comprovació:

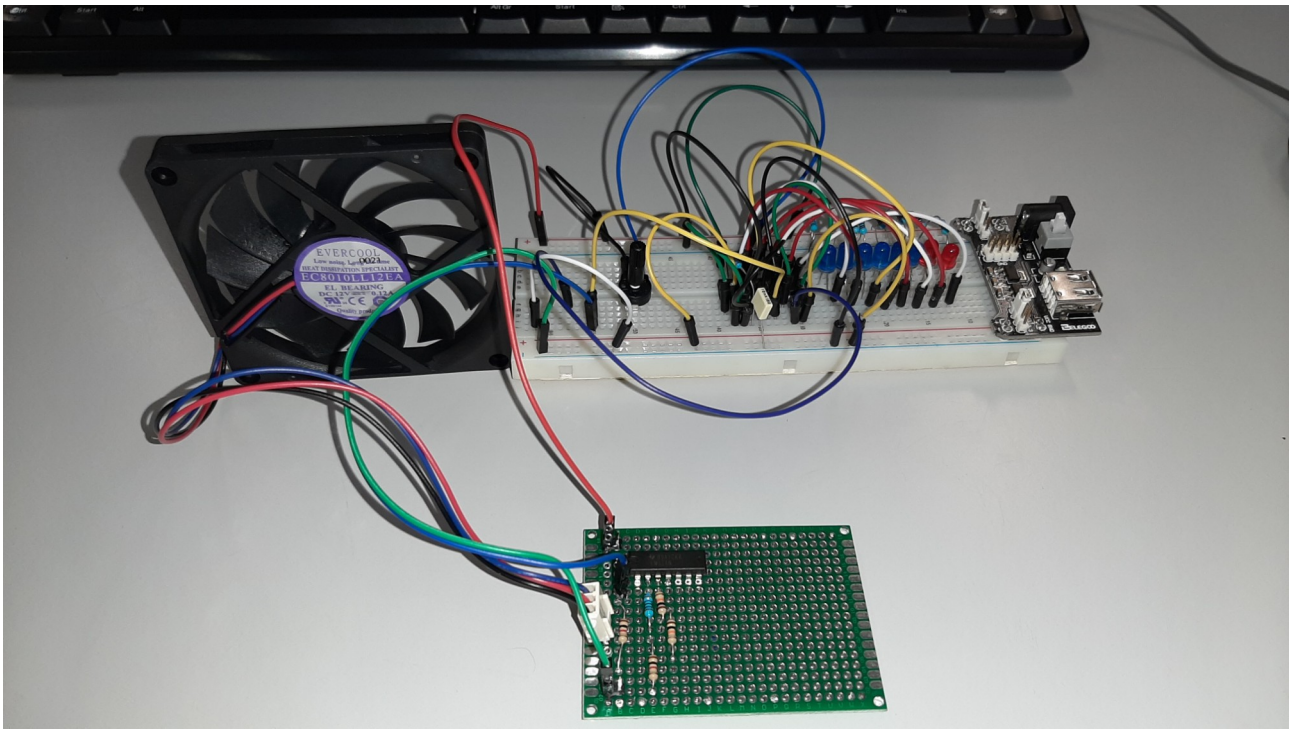

Figura 5

Funcionament ADC0804:

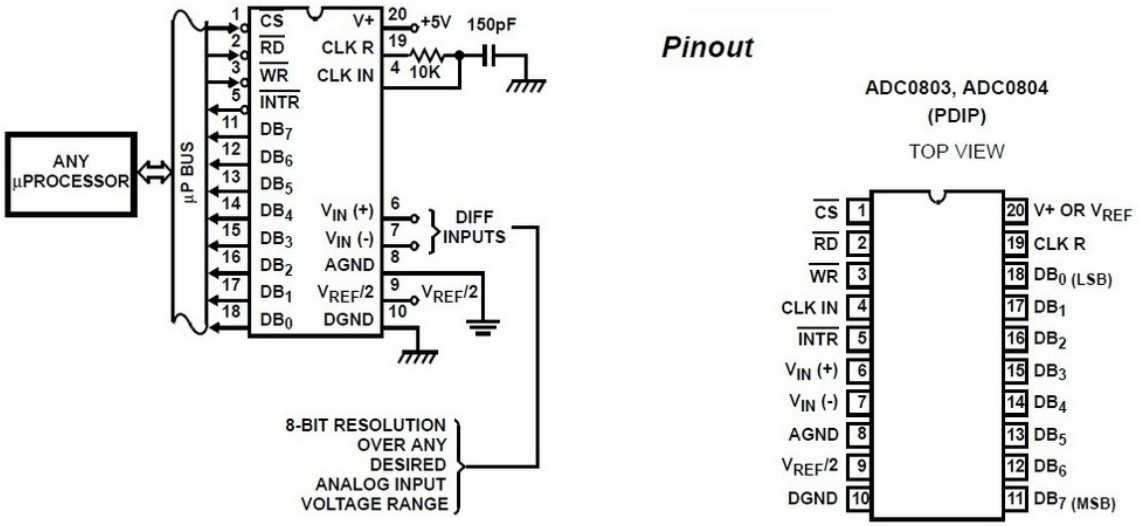

Figura 6

Pin out i control de funcions dispositiu:

Pin 1: **CS** (Chip select) Cal que estigui a 0 per habilitar dispositiu.
Pin 2: **RD** (Read) Posar a 1 i després d'un temps posar a 0. Fent això el valor dels registres interns surten per pins de dades.
Pin 3: **WR** (Write) Passant de 0 a 1 aquest pin comença la conversió d'analògic a digital.
Pin 4: **CLK IN** (Clock In) Per connectar un senyal de rellotge extern.
Pin 5: **INTR** (Interrupt) Aquest pin passa automaticament a 0 quan la conversió està feta.
Pin 6: **Vin**(+) Entrada de lectura analògica del conversor.
Pin 7: **Vin**(-) Aquesta entrada en la majoria de casos caldrà posarla a 0 o massa.
Pin 8: **AnalogGroud** (AGND) Massa analògica.
Pin 9: **Vref/2** La tensió d'aquesta entrada es la que es farà servir de referència per a la divisió dels valors de l'entrades Vin. Sent un convertidor de 8 bits la divisió és de 255 passes. En el projecte el rang és de 0 a 5V equivalent a passes de 19.6mV per bit.

Vref/2 (pin9) (volts)	Input voltage span (volts)	Step size (mV)
Left open	0 – 5	5/255 = 19.6
2	0 – 4	4/255 = 15.69
1.5	0 – 3	3/255 = 11.76
1.28	0 – 2.56	2.56/255 = 10.04
1.0	0 – 2	2/255 = 7.84
0.5	0 – 1	1/255 = 3.92

Figura 6

Pin 10: **DGND** (DigitalGround) Massa digital.
Pin 11 a 18: **DATA**. Sortides de dades digitals.
Pin 19: **CLK R** (Clock R) Es fa servir juntament amb Clock IN per fer ús del rellotge intern.
Pin 20: Alimentació. Vcc a +5V.

La gestió de les entrades de control ADC0804 mitjançant el codi:

Caldrà controlar les entrades WR, INTR, RD. Inicialment caldrà posar l'entrada WR a 0 i després d'una petita espera passar-la a 1. Justament després caldrà tenir present el valor de la sortida INTR, en el moment que es troba en 1 el convertidor està treballant en la lectura de l'entrada Vin i en el moment que passa a 0 aquesta conversió ja estrarà en acabada i a l'espera en els circuits de registres interns. Després cal posar entrada RD de 0 després de l'espera d'un temps curt a 1 per obtenir el resultat de l'operació interna a la sortida de dades.

Problemes trobats durant la pràctica de prova manual del convertidor ADC0804:

Un cop dissenyat el circuit amb el programa Proteus i comprovat que la simulació funcionava correctament (Fig.3) es passa a la seva connexió física fent servir el mateix codi de Proteus a la Raspberry.
A la pràctica s'observa que les lectures obtingudes de la sortida del convertidor no són del coherents, repetint lectures de 00000000 i 11111111 entre lectures de diferents valors, s'intenta variar els time.sleep però no s'obté resultat.

Es pren la decisió de cercar models pràctics de comprovació manual del convertidor sense fer servir codi i després d'haver posat a la pràctica diferents circuits de prova, cap d'ells ha funciona. Alguns d'ells incorporen circuits RC entre els pins CLK IN i CLK R per tal d'aprofitar el senyal de rellotge intern i injectar-lo a l'entrada CLK IN. D'altres a part d'això incorporen un pulsador a massa per entrar un 0 a les entrades WR i INTR que estan unides i en el moment de la pulsació és l'ordre de la lectura i conversió. També n'hi ha que adapten un generador de pulsos de rellotge els introdueixen per WR i posen les entrades CS i RD a 0 (aquest últim no es va poder per la manca del circuit 555, però sí aprofitant el pulsos enviats per la Raspberry a mode de rellotge a l'entrada WR amb el codi del final d'aquesta memòria). Malauradament, aquestes configuracions a la pràctica no han funcionat i per temps s'ha hagut de continuar per acabar la memòria.

Una de les conclusions, en la manca de fruits quan s'ha treballat amb aquest convertidor, és el coneixement que es té del soroll generat a nivell intern d'aquest circuit integrat i que comporta problemes de funcionament per interferència i que cal erradicar a base de condensadors i també tenir ben presents els tempos en la governança de les entrades WR,CLK, INTR i RD per tal de no blocar el funcionament correcte del dispositiu i que a nivell manual sense un dispositiu que el controlés correctament ha estat difícil de superar en el temps establert.

<u>Funcionament del Shift-Register 74HC165:</u>

Pinout:

Pin 1: **SH/LD'** Quan passa d'1 a 0 les dades són carregades al registre intern, quan ho fa de 0 a 1 les dades són enviades a la sortida
Pin 2: **CLK** Entrada de rellotge extern
Pin 3, 4, 5, 6, 11, 12, 13, 14: **$D_{(0-7)}$** en Proteus o **A a H** en els datasheets. Entrades de dades en paral·lel.

Pin 7 i 9: **QH'** i **SO** en proteus o **Q_H'** i **Q_H** en les datasheet. Sortides de dades en sèrie. La sortida 7 és el valor complementari de les entrades paral·leles.
Pin 8: **GND** Connexió a massa.
Pin 10: Entrada sèrie.

El codi de Proteus d'aquest projecte està principalment pensat per llegir les dades entregades en sèrie per aquest dispositiu. Cal una oscil·lació de rellotge que entrega la Raspberry i que fa sincronitzar la les dades que surten una a una des del 74HC165 i fa la funció de no perdre o barrejar cap dels 8 bits que en sortiran. Per a tenir aquesta sincronització el senyal de rellotge provinent de la Raspberry entra al pin 1 CLK.

Per donar l’ordre de sortida de les dades cal un “Latch”. En el moment que el codi dona l’ordre de lectura a l’entrada SH/LD’ i aquesta passa de 0 a 1 per enviar dades i de 1 a 0 per a tornar a carregar l’estat seguent de les entrades. En estat alt, 1, cada dada sortirà en sèrie per la sortida en l’odre de bit de major pes a menor, estat de D7 a D0 al ritme del rellotge.

Per a la lectura del resultat en sèrie s’ha triat la sortida complementaria QH’ per la confusió que ha donat la nomenclatura SO en Proteus, posteriorment en la pràctica s’ha vist que SO és en realitat QH (sortida valor normal).

Codi lectura Proteus:

```
import RPi.GPIO as GPIO
import time

CLK = 4 # Rellotge
SHLD = 18 # Latch
QH = 17 # Dades entrada 74HC165 a Raspberry

def peripheral_setup () :
   GPIO.setmode(GPIO.BCM)
   GPIO.setwarnings(False) # Talalr verbositat
   GPIO.setup(CLK, GPIO.OUT) # PIN CLOCK Rellotge
   GPIO.setup(QH, GPIO.IN) # PIN DATA Dades
   GPIO.setup(SHLD, GPIO.OUT) # PIN LATCH Mostrar dades

def adquisiciodades () :
   for vegades in range(8):
      GPIO.output(SHLD,GPIO.LOW) # cal posar el lach a 0  per entrar dades al 74HC165
      GPIO.output(CLK,GPIO.LOW) # cal posar rellotge a 0 per obtenir oscil·lacio
      GPIO.output(CLK,GPIO.HIGH) # rellotge a 1
      time.sleep(0.1)

def lecturadades ():
   octet =""
   for index in range(8):
      GPIO.output(SHLD,GPIO.HIGH) # cal posar el lach a 1 per passar dades a sortida QH'
      GPIO.output(CLK,GPIO.LOW) # cal posar rellotge a 0 per obtenir oscil·lacio
      GPIO.output(CLK,GPIO.HIGH) # rellotge a 1
      octet += str((GPIO.input(QH)+1)%2)
      time.sleep(0.1)
   print(octet)

def main () :
   peripheral_setup()
   while 1 :
      adquisiciodades()
      lecturadades()
      #print(octet)
if  __name__  ==  '__main__' :
   main()
```

Conclusions

Les pretensions inicials del projecte volien copsar més quantitat de dispositius i adaptar-ne el funcionament amb una formula senzilla i a preu baix, tot i sabent que al mercat es poden trobar ja fets i a un cost molt més elevat i a nivell de codi sols caldria pensar per adapta-los a l'ús que se'ls vogui donar.

La filosofia en el viatge que ha suposat aquest projecte i el fet d'anar a la base de l'electrònica senzilla i amb un codi simple per obtenir uns resultats acceptables ha estat enriquidor i una estada al que es feia en altres èpoques i que a l'actualitat els dispositius utilitzats com els shift-registers i convertidors A/D estan en desús i els que es troben al mercat tenen dates de fabricació antigues, tot i així no han perdut la seva utilitat a la hora d'adaptar dispositius i fer projectes amb recursos limitats.

Els shift registers són útils per qualsevol dispositiu de resultat lògic i han ajudat a estalviar sortides/entrades GPIO de la Raspberry i que per projectes més amplis en podriem anar escasos, i en el cas dels A/D converters demostren la seva amplia adaptabilitat a qualsevol dispositiu que entregui una informació analògica.

Tot i l'intent de simplicitat s'ha trobat pel camí tot un regitzell d'entrebancs que cal anar superant i a vegades les petensión senzilles requereixen d'inventiva per anar adaptant la idea inicial als recursos que es tenen i al temps estipulat.

Joan Sunyer
16 Juliol de 2021