**MEDICIÓN DE RPM FILTRADAS USANDO PIO**

*(Publicado el 2021-04-17 por Lucky Resistor)*

<https://luckyresistor.me/2021/04/17/filtered-rpm-measurement-using-pio/>

El chip Raspberry Pi RP2040 es un microcontrolador extraordinario. El bloque IO programable (PIO) es sólo una de las muchas características excelentes de este chip. En esta publicación, le muestro un método sobre cómo utilizar PIO para capturar la velocidad de un ventilador de PC conectado con el filtro de software incluido.

La mayoría de los ventiladores tienen una señal de retroalimentación, donde se puede medir la velocidad real del ventilador. Esta señal de retroalimentación generalmente se implementa como un colector abierto. Por lo tanto, puede conectar este directorio de líneas a un microcontrolador y utilizar el pull-up incluido para leer la señal.

A menudo, la señal devuelta por el ventilador tiene algo de ruido. A veces se obtienen picos que superan el umbral lógico y, por lo tanto, es necesario filtrarlos.

* Implementación
* El código PIO
* La clase contenedora
* Conclusión

**Implementación**

El siguiente código implementa la medición de RPM como algoritmo PIO y lo envuelve en una clase.

##XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX##

## Copyright (c) 2021 by Lucky Resistor. <https://luckyresistor.me/> ##

## See LICENSE file for details. ##

##XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX##

##XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX##

## Herramienta para capturar la entrada de RPM de un VENTILADOR ##

##XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX##

import rp2

from micropython import const

from machine import Pin

#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#

# La frecuencia de la medición #

#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#

FREQ = const(1\_000\_000)

#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#

# El valor inicial del contador #

#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#

COUNTER\_START = const(0x10000)

@rp2.asm\_pio(in\_shiftdir=rp2.PIO.SHIFT\_LEFT, autopull=False, autopush=False)

def measure\_rpm():

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

## Inicialice X con 0x10000 para valores de 16 bits ##

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

label("switch\_to\_low")

set(x, 1)

mov(isr, x)

in\_(null, 16)

mov(x, isr)

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

## Inicializar el ISR con 0x00000000 ##

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

set(y, 0)

mov(isr, y)

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

## Esperar nivel sea high estable ##

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

label("wait\_for\_high")

in\_(pins, 1)

set(isr, 0)

mov(y, invert(isr))

jmp(not\_y, "switch\_to\_high")

jmp(x\_dec, "wait\_for\_high")

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

## Nos encontramos con un tiempo de espera, ##

## enviamos cero, seguimos esperando ##

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

set(y, 0)

mov(isr, y)

push(noblock)

jmp("switch\_to\_low")

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

## Inicialice la ISR con el 0xFFFFFFFFF ##

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

label("switch\_to\_high")

set(y, 0)

mov(isr, invert(y))

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

## Espere un low estable ##

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

label("wait\_for\_low")

in\_(pins, 1)

set(isr, 0)

mov(y, isr)

jmp(not\_y, "send\_data")

jmp(x\_dec, "wait\_for\_low")

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

## Nos encontramos con un tiempo de espera, enviamos ##

## cero, reiniciamos. ##

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

set(y, 0)

mov(isr, y)

push(noblock)

jmp("switch\_to\_low")

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

## Enviar el contador en este punto ##

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##

label("send\_data")

mov(isr, x)

push(noblock)

##\*\*\*\*\*\*\*\*\*\*\*##

## Repetir ##

##\*\*\*\*\*\*\*\*\*\*\*##

jmp("switch\_to\_low")

class RpmInput:

####XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX####

#### Clase para medir una señal de RPM de un ventilador ####

####XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX####

def \_\_init\_\_(self, sm\_id: int, pin: int):

####XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX####

#### Crear una nueva instancia de entrada RPM. ####

#### ####

#### :param sm\_id: El identificador de la máquina de ####

#### estado que se utilizará. ####

#### :param pin: El número de PIN utilizado para ####

#### la entrada. ####

####XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX####

self.\_sm = rp2.StateMachine(sm\_id, measure\_rpm,freq=FREQ,

in\_base=Pin(pin,mode=Pin.IN,

pull=Pin.PULL\_UP))

self.\_sm.active(1)

self.\_value = 0

def poll(self):

####XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX####

#### Encuesta la interfaz (Poll the interface). ####

#### ####

#### Llame a este método desde el bucle principal ####

#### tantas veces como desee. ####

####XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX####

self.\_value = self.\_sm.get()

def read\_rpm(self):

####XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX####

#### Convierta el último valor leído en un valor ####

#### de RPM. ####

#### ####

#### :return: Un valor de RPM, o cero si no se ####

#### recibe señal. ####

####XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX####

if self.\_value == 0:

return 0

counts = COUNTER\_START - self.\_value

###XOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXO###

### signals\_per\_rotation = 2 (señales\_por\_rotación) ###

### La fórmula es: ###

### 1 / ( counts \* 5 \* 1/frequency ) / signals\_per\_rotation \* 60 ###

### ###

### Resuelve a: ###

### 12 \* frequency / counts ###

###XOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXO###

return 6 \* FREQ // counts

**El Código PIO**

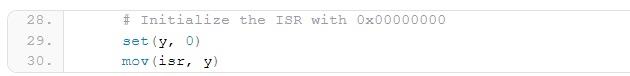
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

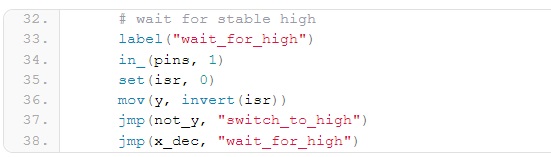
El código comienza en la línea 19 con instrucciones para desactivar la función de extracción automática y agrega un recordatorio para la dirección del cambio (se deja de forma predeterminada).

Después del inicio, inicializo X con 0x10000. El comando set está limitado a cinco bits, por lo que cargo el ISR con uno (línea 23, 24) y cambio este bit a la ubicación correcta (línea 25), luego lo copio nuevamente a X.

Ahora, X está cargado con 0x10000. Al cambiar el número de pasos de cambio, es fácil ajustar la precisión de este contador.



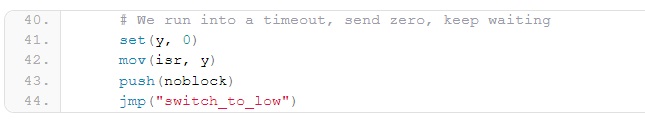
A continuación, inicializo el registro ISR con cero. El registro se utiliza para actuar como filtro.



La entrada se lee y su valor se traslada al registro ISR (línea 34). A continuación, reinicio el contador de lectura del ISR, que es necesario para mantener los bits en movimiento (línea 35).

Ahora copio el valor actual del registro en Y, pero invierto todos los bits (línea 36). Si hubo 32 lecturas posteriores de 1, el registro Y es, por tanto, cero.

Si el registro Y es cero salto a la detección de la siguiente fase (línea 37), si no, disminuyo X y si X es mayor que cero repito este bloque (línea 38).



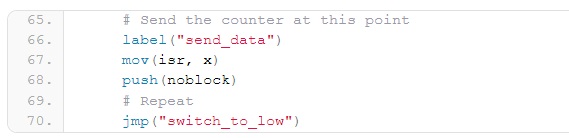
El bloque anterior se ejecuta en caso de que se agote el tiempo de espera. Aquí introduzco un cero en el FIFO y vuelvo al inicio del código. Esto seguirá esperando un nivel alto en la línea y seguirá sacando ceros del FIFO para indicar un tiempo de espera.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Este es casi el mismo código que antes, pero en este caso inicializo el ISR con unos (línea 49). Además, como pruebo los ceros, copio el contenido del ISR sin invertirlo (línea 55).

Si la transición de alto a bajo se detecta con éxito, el código continúa enviando el valor del contador al FIFO (línea 56).



El recuento restante en X se copia al ISR, que ahora se utiliza para enviar los datos al FIFO, donde se pueden leer desde la aplicación.

Este código simple permite mediciones de RPM estables, incluso si hay picos cortos y otros problemas menores en la señal del ventilador.

Se agotará el tiempo de espera si se pierde la señal o si solo hay ruido en la línea.

**La Clase Contenedora (The Wrapper Class)**

La clase contenedora simplifica el uso del código PIO y convierte la salida en una aproximación de un valor de RPM.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Se crea una nueva instancia con el identificador de la máquina de estado y el número de pin. Luego, el código crea la máquina de estados con la frecuencia correcta y también inicializa el pin con la configuración correcta.

Al final, se activa la máquina de estados y el búfer de valores se inicializa con cero.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Debido al FIFO, es necesario leer continuamente los valores del búfer para obtener las mediciones actuales. Por eso agregué un método de encuesta. Este método debe llamarse en el bucle principal.

El método read\_rpm simplemente usa el último valor leído y lo convierte en un valor de RPM. Esta conversión no es perfecta, porque el cálculo ignora los ciclos para enviar datos e inicializar registros. Está lo suficientemente cerca para monitorear la velocidad de un ventilador y mantener los cálculos al mínimo.

**Conclusión**

Espero que hayas encontrado útil este código. 😄 Déjame saber si tienes ideas para mejorar o si encuentras nuevos usos para el código de filtro.

Si tiene alguna pregunta, información perdida o simplemente desea brindar comentarios, no dude en comentar a continuación.