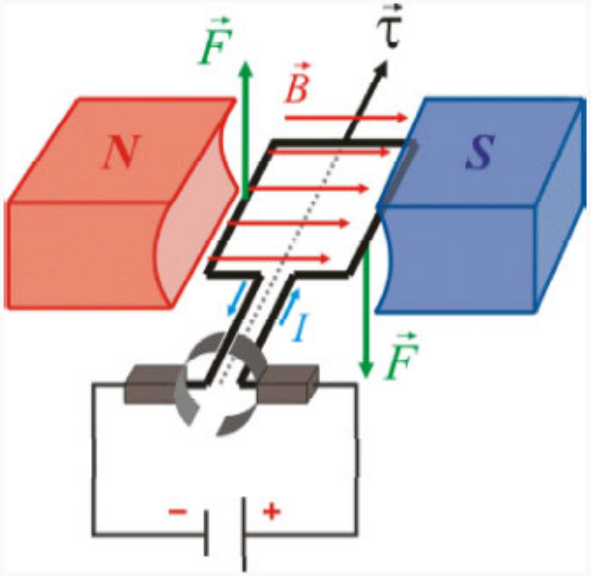
# Implementación del controlador

# Índice

- Motor de DC
- Encoder
- Modulación PWM
- Driver del motor
- Arquitectura SW del controlador

# Motor DC




$$\vec{F} = I \cdot (\vec{L} \times \vec{B})$$

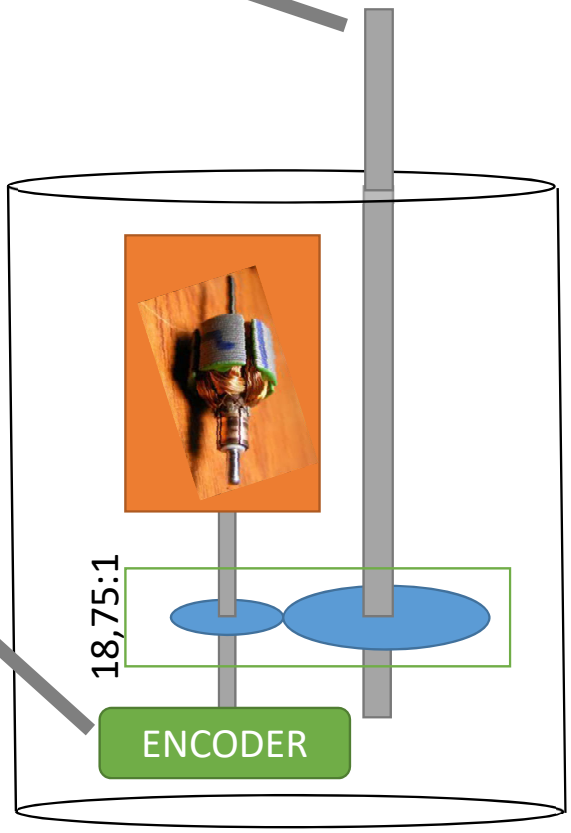
F: fueza inducida
I: corriente
L: longitud de la bobina
$\tau$: par fuerza (torque)

# Motor DC

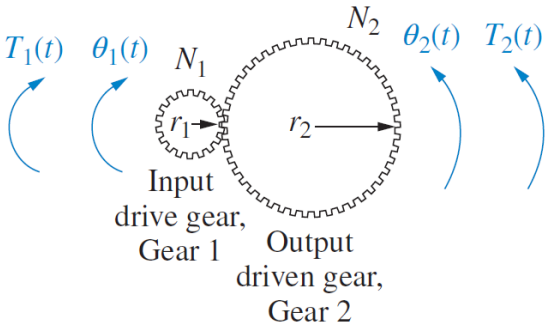
**Pololu 4741**

- 12 V
- Reductora 18,75:1
- Encoder resolución 64
- Velocidad máxima sin carga 8.8 rps





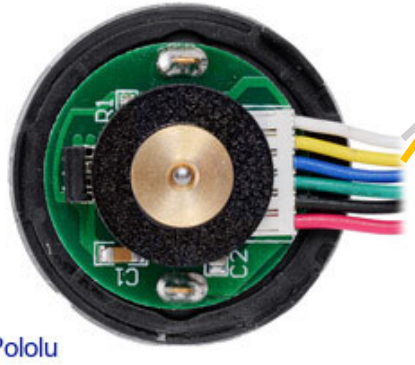

## Engranajes (reductora)

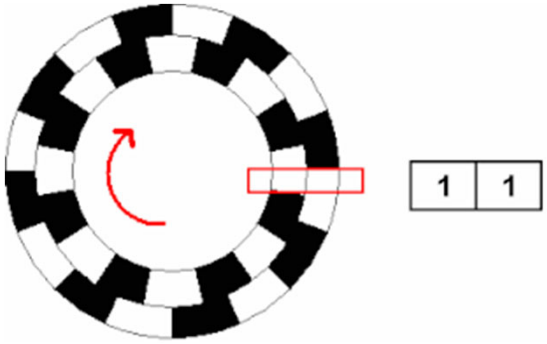


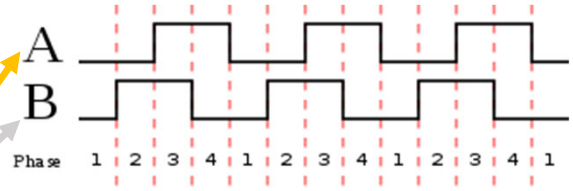

$$\frac{T_1}{T_2} = \frac{\theta_2}{\theta_1} = \frac{r_1}{r_2} = \frac{N_1}{N_2}$$

# Quadrature Encoder

- Disco magnético acoplado al eje
- 2 sensores de efecto Hall
- Resolución 64 cuentas por revolución del eje del motor


Sentido de giro

123412341234

432143214321

Posibilita la realización de medidas
- de fase (relativas)
- o de velocidad de giro

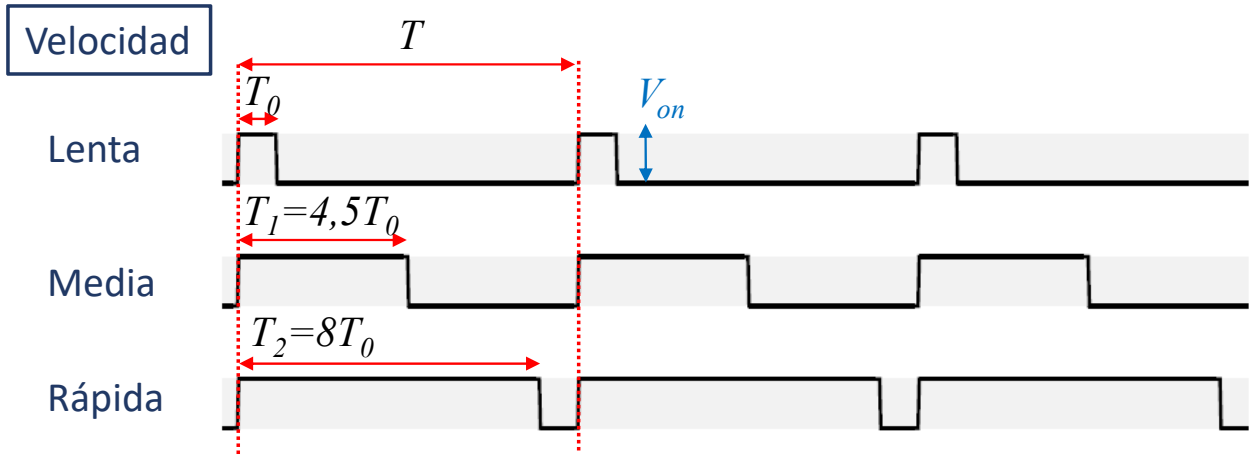
## Implementación

- Detectar cambios en A o B
- Incrementar/decrementar un contador teniendo en cuenta la fase actual y la anterior
- #pasos_1_vuelta = 64*factor_reductora
- Fase = #pasos x 2π/#pasos_1_vuelta (rad)
- Velocidad ang = Fase_en_Tm/Tm (rad/s)

# PWM

## PWM: Modulación de anchura de pulsos

- Genera trenes de pulsos de frecuencia constante y ancho variable
- Aplicaciones: regulación de velocidad en motores o luminosidad en LEDs




Valor medio

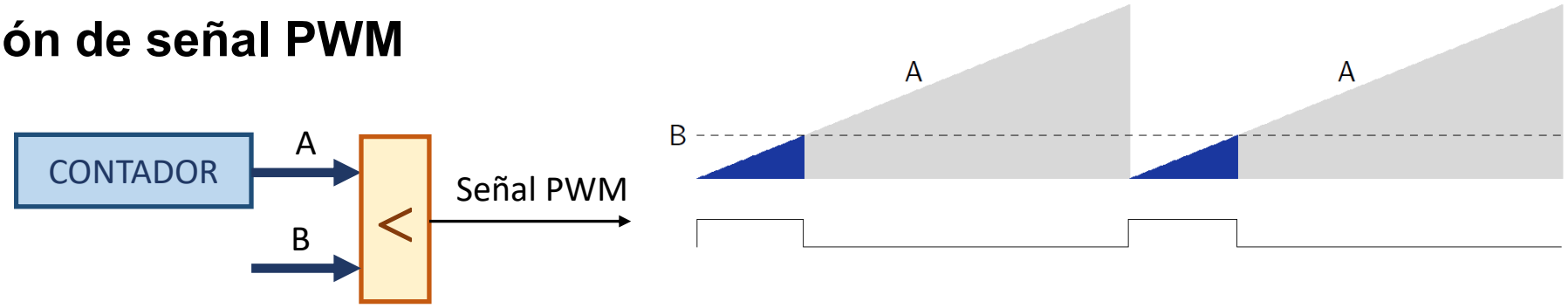
$$V_{m\_o} = V_{on}\frac{T_0}{T}$$

$$V_{m\_1} = V_{on}\frac{T_1}{T} = V_{on}\frac{4{,}5\cdot T_0}{T} = 4{,}5\cdot V_{m\_0}$$

$$V_{m\_2} = V_{on}\frac{T_2}{T} = V_{on}\frac{8\cdot T_0}{T} = 8\cdot V_{m\_0}$$

# PWM

## Generación de señal PWM




**ESP32 contiene 16 controladores PWM independientes (denominados LED_PWM)**

**Programación con el entorno Arduino:**

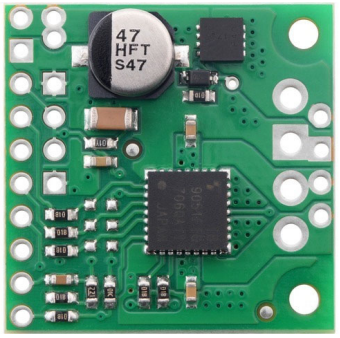
```
// Configurar LED_PWM
ledcSetup(pwmChannel, pwmfreq, pwmresolution);

// Anexar el GPIO al canal PWM
ledcAttachPin(PWM_Pin, pwmChannel);

// Excitacion del motor con PWM
ledcWrite(pwmChannel, Valor_pwm);
```

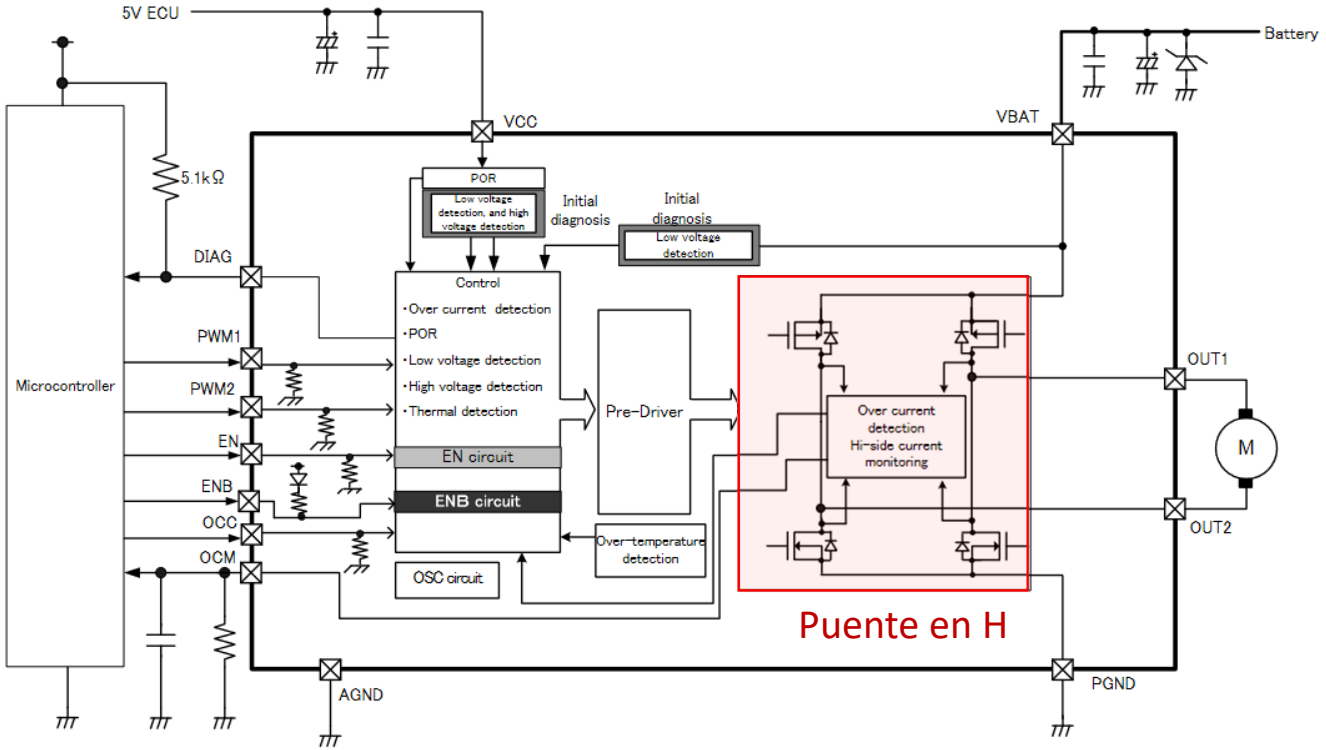
pwmChannel: #canal de 0 a 15
Pwmfreq: frecuencia de operación en Hz
Pwmresolution: #bits del contador

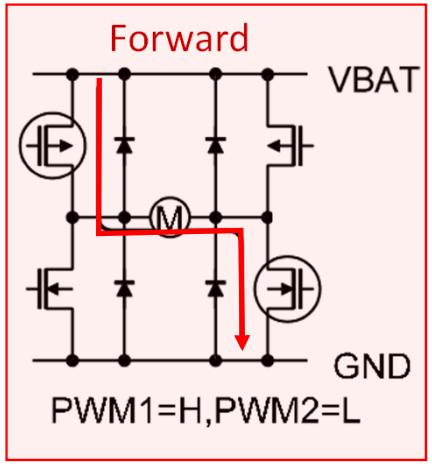
→ Valor_pwm: valor de B (entre 0 y 2^ Pwmresolution-1)

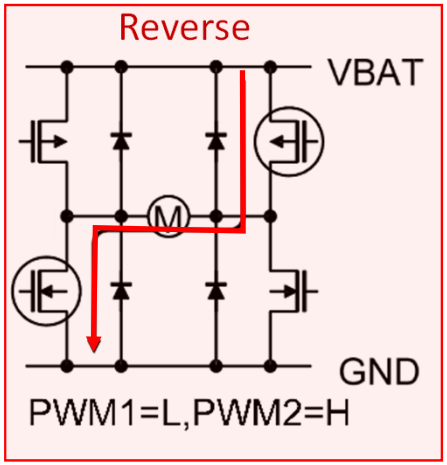
# Driver del motor DC

**TB9051FTG**



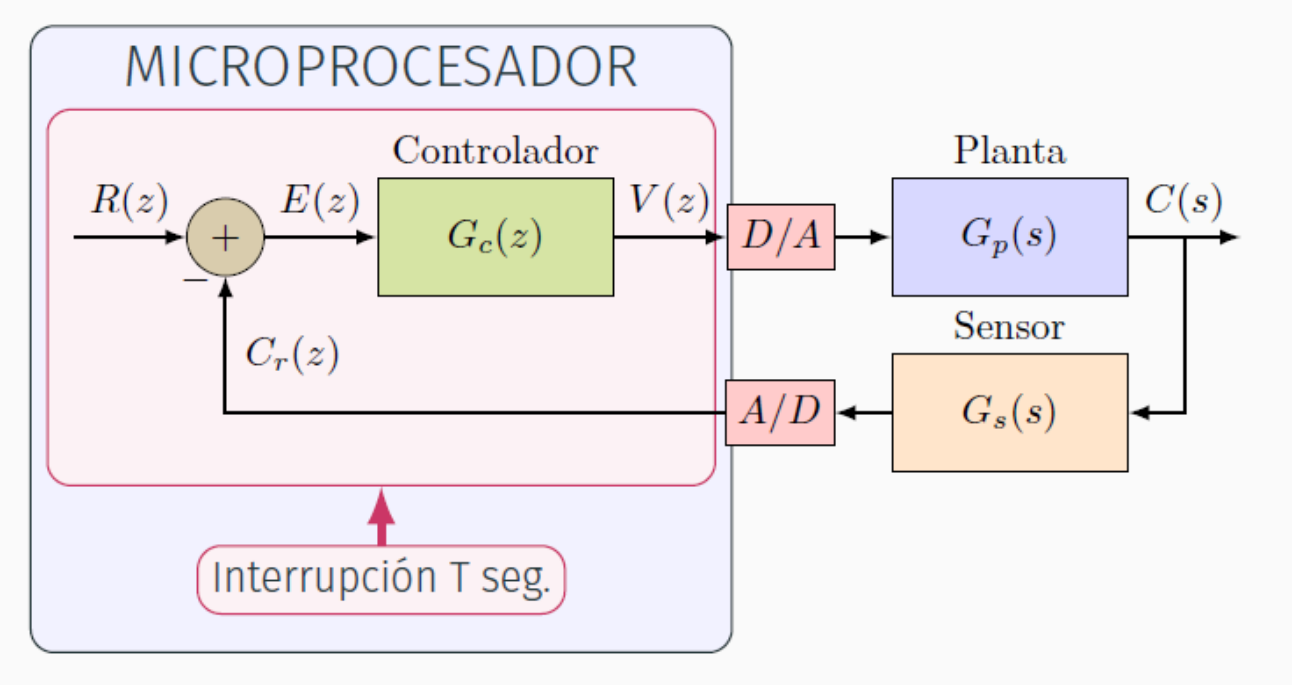



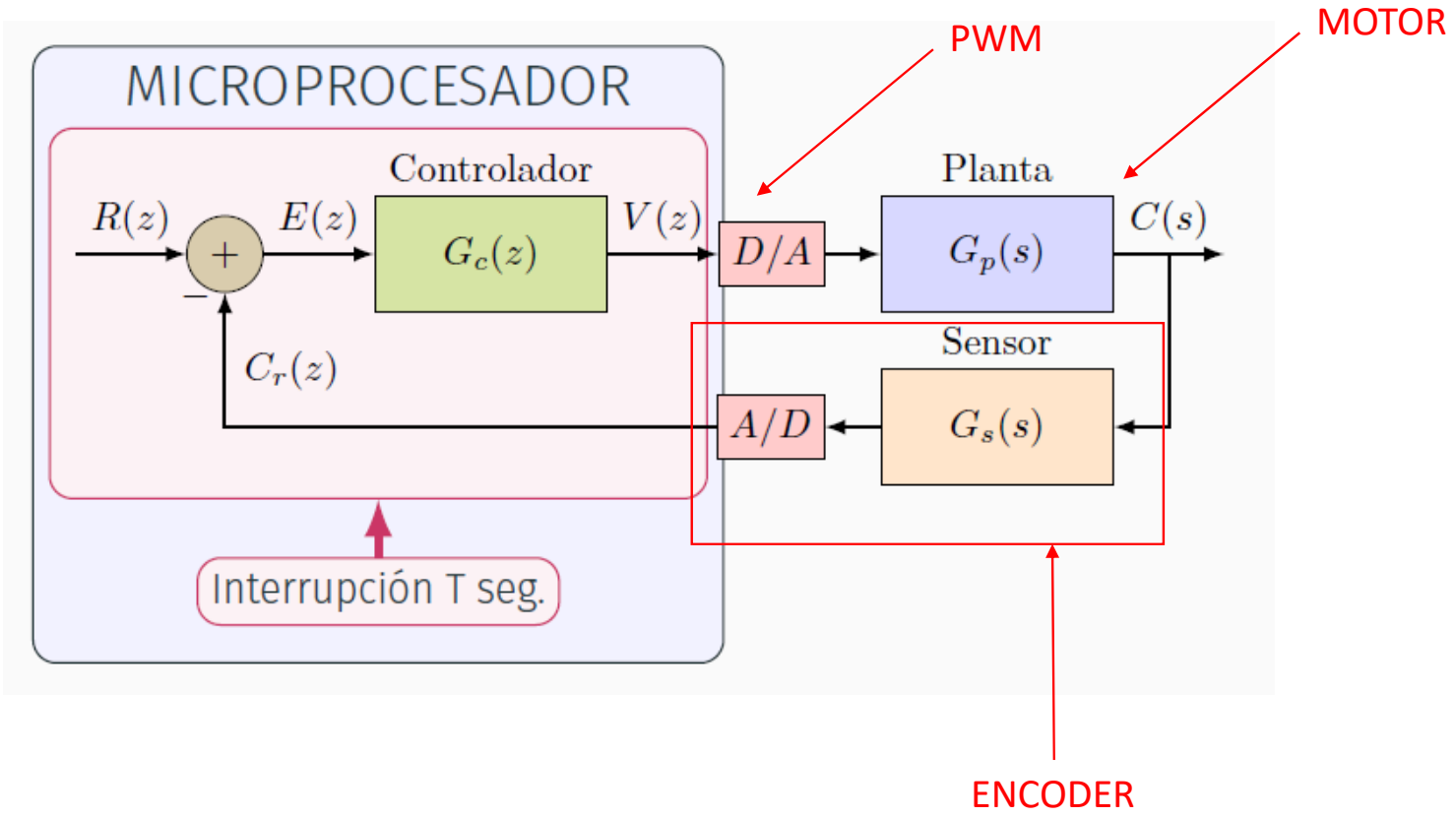




| TB9051FTG simplified truth table (PWM1 + PWM2 + EN) | | | | | | |
|---|---|---|---|---|---|---|
| Inputs | | | | Outputs | | Operation |
| EN | ENB | PWM1 | PWM2 | OUT1 | OUT2 | |
| PWM | 0 | 1 | 0 | PWM (H/Z) | PWM (L/Z) | forward/coast at speed *PWM %* |
| | | 0 | 1 | PWM (L/Z) | PWM (H/Z) | reverse/coast at speed *PWM %* |
| 0 | X | X | X | Z | Z | coast (outputs floating/disconnected) |
| X | 1 | X | X | Z | Z | |

# Arquitectura SW del controlador



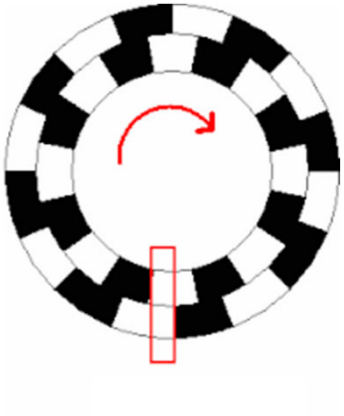
# Arquitectura SW del controlador

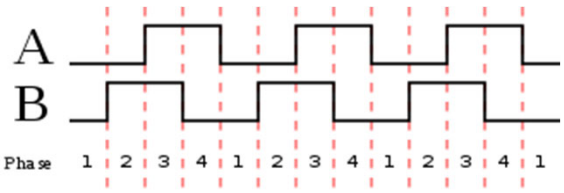


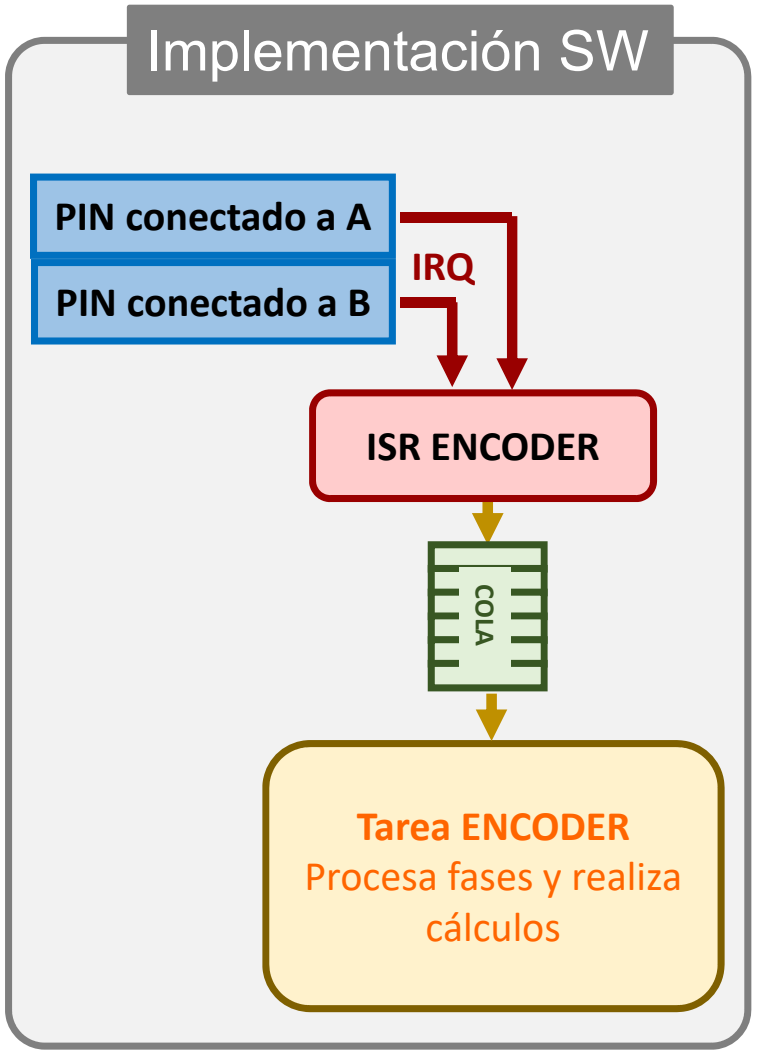
# Arquitectura SW del controlador

## Lectura del encoder

- Genera interrupciones en cada cambio en A o B
- La ISR ENCODER solo lee los valores de Ay B y los envía a una cola
- Posteriormente, la Tarea ENCODER procesa dichos valores
  - Calcula el incremento/decremento de pasos
  - El resultado lo deja en una variable global



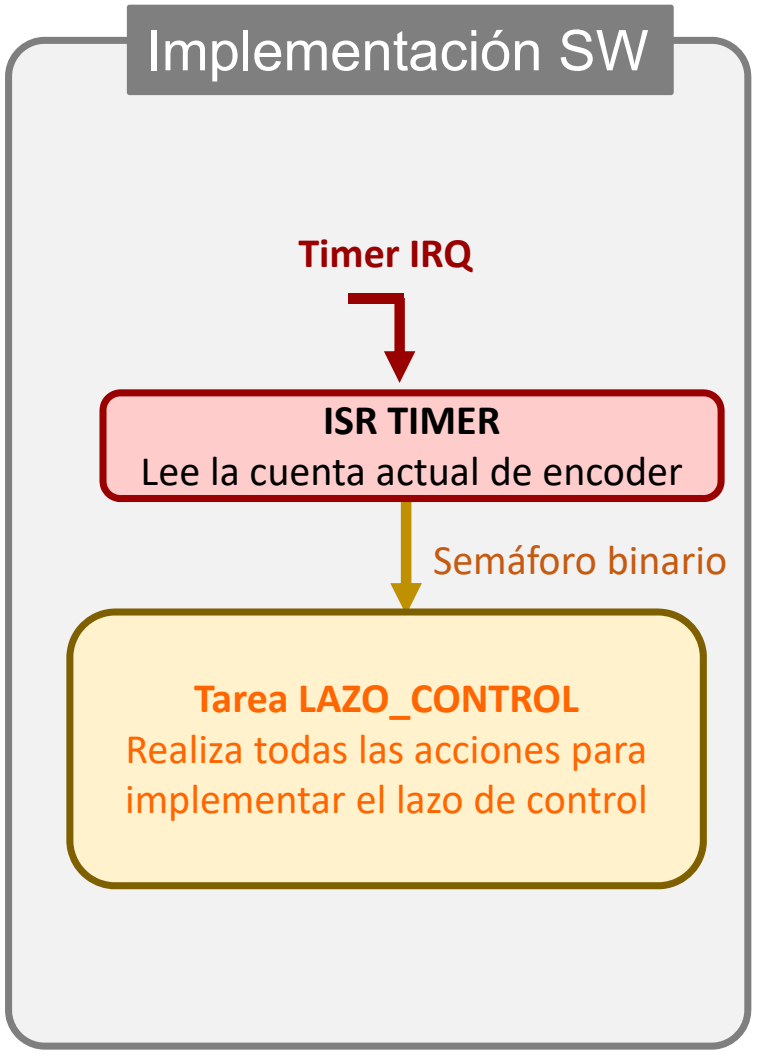


# Arquitectura SW del controlador

## Lazo de control

- Un TIMER genera interrupciones periódicas cada Tm seg
- La ISR del TIMER solo lee la cuenta del encoder y activa un semáforo binario
- La Tarea LAZO_CONTROL
  - Calcula el ángulo/velocidad
  - Lee señal de referencia
  - Calcula la señal de error
  - Procesa acción del controlador
  - Aplica mediante PWM la acción del controlador



# Arquitectura SW del controlador

**Cálculo del ángulo y velocidad del motor**

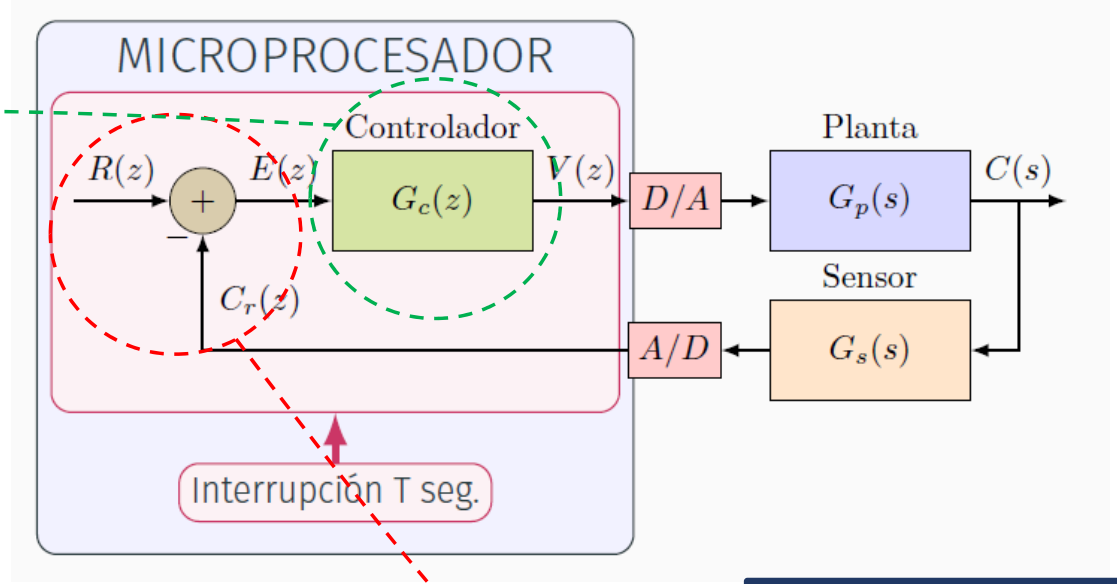
## Ángulo

- #pasos_en_1_vuelta = factor_reducción_engranajes x factor_mult_encoder = 18,75 x 64 = 1200 pasos
- Ángulo_por_paso = $2\pi$/ #pasos_en_1_vuelta = 5,235e-3 rad (0,3º)

## Velocidad de giro

- Periodo de muestreo $T_m$ = 0.01 seg ($f_m$ = 100Hz)
- Velocidad_de_giro = (#pasos_en_$T_m$ / $T_m$ ) / #pasos_en_1_vuelta rps

# Arquitectura SW del controlador




**E** Implementación de un control PI

Func. transf: $G_C(z)=\frac{V(z)}{E(z)}=\frac{10(z-0.9)}{z-1}$

Ec. dif: $v[n]=v[n-1]+10e[n]-9e[n-1]$

```
% pseudocode
c_r = lee_valor_velocidad;
r = lee_señal_referencia;
e = r – c_r;
v = v_1 + 10*e – e_1;
v_1 = v;
e_1 = e;
aplica_acción_contro_con_PWM(v);
```