T2 Control de un Shaft Encoder y Velocidad en un Motor Sistemas Empotrados 1

Héctor Lacueva Sacristán 869637

17/02/2025

Índice

Objetivos **1**

Problema **1**
Pulsos por vuelta . . . 1
Pulsos máximos detectables por segundo . . . 1
Revoluciones por segundo máximas . . . 1
Velocidad angular máxima (rad/s) . . . 2
Velocidad en RPM . . . 2
Tiempo de cómputo máximo de la RSI . . . 2

Decisiones **2**

Grafo máquina de estados **2**

Variables de estado **3**

Pseudocódigo **3**

Comentarios **5**

Objetivos

- Control de rebotes de un shaft encoder.
- Implementar un sistema de control de velocidad utilizando un shaft encoder en un motor LEGO.

Problema

Contamos con un motor del cual queremos conocer su velocidad angular y su sentido de giro, este cuenta con un shaft encoder con 360 agujeros por vuelta, por lo que cada grado que se mueva el motor se genera un pulso.

Para las mediciones contamos con dos sensores, el sensor A y el sensor B, que gracias a un desfase permiten calcular el sentido de giro del motor.

Para realizar el muestreo contamos con un reloj a 1KHz, puede tomar hasta 1000 muestras por segundo.

Pulsos por vuelta

$$P_{vuelta} = 360 \text{ Uno por cada agujero.}$$

Pulsos máximos detectables por segundo

$$N_{máx} = 1000\text{pulsos/s}$$

Revoluciones por segundo máximas

$$rps_{max} = \frac{N_{máx}}{P_{vuelta}} \approx 2,78\text{rev/s}$$

Velocidad angular máxima (rad/s)

$$\omega_{máx} = rps_{máx} \times 2\pi \approx 17,45 \text{rad/s}$$

Velocidad en RPM

$$RPM_{máx} = rps_{máx} \times 60 \approx 166,67 \text{RPM}$$

Tiempo de cómputo máximo de la RSI

Pese a no encontrar ninguna referencia, está claro que no puede ser mayor que el tiempo de muestreo. Si además tenemos en cuenta que se deben realizar otras operaciones, pongamos que como máximo un 25% del tiempo entre distintos muestreos.

$$Tisr_{máx} = 0,25 \times 1ms = 250\mu s$$

Decisiones

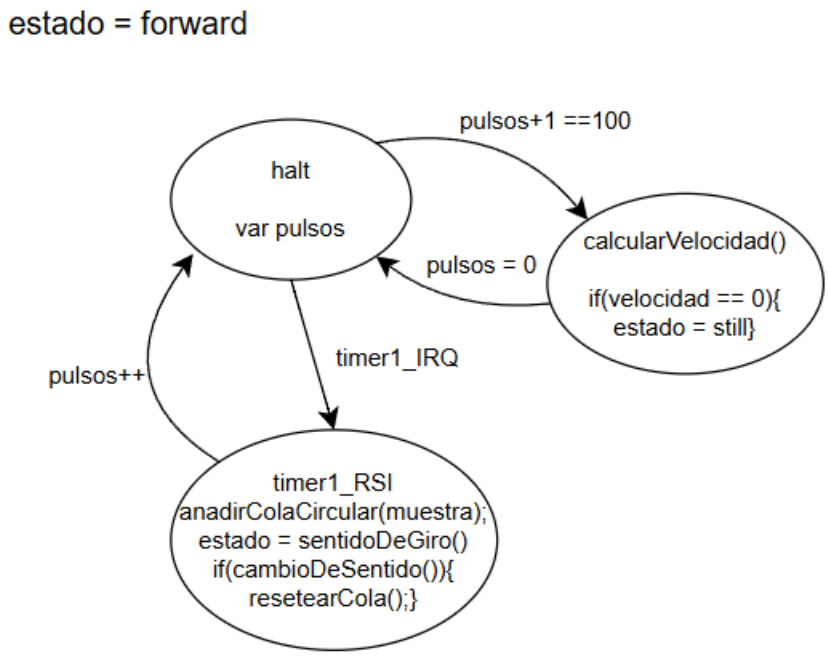
- Cada 100 ms se calculará la velocidad (10 veces por segundo aprox), realmente se realizará cuando se hayan ejecutado 100 muestreos.
- Cada 1 ms saltará una RSI se calculará el estado en la situación actual con respecto a la situación anterior.
- Cuando se produce un cambio en el sentido de giro del motor, se resetea la cola.
- **Cuando se está calculando la velocidad se desactivan las interrupciones del timer para evitar cualquier problema y al acabar se re-activan**.
- Cuando se muestrean los sensores, si se produce un error como los que se muestran en la siguiente imágen se inserta en la cola el valor del último estado.

Tabla con posibles variantes entre dos muestreos

Previous sample pair(n - 1)		Current samples pair(n)		SAMPLE register	ACC operation	ACCDBL operation	Description
A	B	A	B				
0	0	0	0	0	No change	No change	No movement
0	0	0	1	1	Increment	No change	Movement in positive direction
0	0	1	0	-1	Decrement	No change	Movement in negative direction
0	0	1	1	2	No change	Increment	Error: Double transition
0	1	0	0	-1	Decrement	No change	Movement in negative direction
0	1	0	1	0	No change	No change	No movement
0	1	1	0	2	No change	Increment	Error: Double transition
0	1	1	1	1	Increment	No change	Movement in positive direction
1	0	0	0	1	Increment	No change	Movement in positive direction
1	0	0	1	2	No change	Increment	Error: Double transition
1	0	1	0	0	No change	No change	No movement
1	0	1	1	-1	Decrement	No change	Movement in negative direction
1	1	0	0	2	No change	Increment	Error: Double transition
1	1	0	1	-1	Decrement	No change	Movement in negative direction
1	1	1	0	1	Increment	No change	Movement in positive direction
1	1	1	1	0	No change	No change	No movement

Grafo máquina de estados

Grafo de la máquina de estados



Variables de estado

Las variables de estado que se considerarán serán las siguientes.

```
struct{
    siguiente = 0;
    numPulsos = 0;
    int pulsos[100];
}colaCircular;

ColaCircular colacircular;

enum{
    forward = 1,
    still = 0,
    backward = -1
}Estado;

Estado estado = forward, old_estado = forward;

int muestra, viejoA = 0, viejoB = 0, A, B;

float velocidadAngular = 0;
```

Pseudocódigo

```
void resetColaCircular(){
    numPulsos = 0;
    siguiente = 0;
}

void anadirColaCircular(int muestra){
    pulsos[siguiente] = muestra;
```

```
    numPulsos = (numPulsos == 100) ? 100 : numPulsos+1;
    siguiente = (siguiente + 1)%100;
}

void calcularVelocidadAngular(){
    int sumaPulsos = 0;
    for(int i = 0; i < numPulsos; i++){
        int indice = (siguiente - numPulsos + i + 100) % 100;
        sumaPulsos += pulsos[indice];
    }

    // Calcular velocidad angular
    float deltaTheta = (2 * pi / numPulsos) * sumaPulsos;
    float deltaTime = numPulsos * T;

    if (deltaTime > 0) {
        velocidadAngular = deltaTheta / deltaTime;
    } else {
        velocidadAngular = 0;
        estado = still;
    }
}


int leeA(){
    // Función que devuelve el estado del PIN de Entrada al que esté conectado el sensorA.
    return state_A;
}

int leeB(){
    // Función que devuelve el estado del PIN de Entrada al que esté conectado el sensorB.
    return state_B;
}

// Se ejecuta cada 1ms
void timer1_RSI(){
    // Guardo los valores de la última muestra
    old_estado = estado;
    viejoA = A;
    viejoB = B;

    // Leo los nuevos valores
    A = leeA();
    B = leeB();

    cambios = viejoA != A;
    cambios += viejoB != B;

    // Calculo el el movimiento con respecto al anterior
    muestra = 0;

    switch(cambios){
        case 1:
            muestra = ((viejoA == A && A != B) || (viejoB == B && A == B)) ? 1 : -1;
            estado = muestra;
            break;
        case 2:
            // El valor de la muestra será el previo
            muestra = estado;
            break;
    }
```

```
    // Si se ha modificado el sentido, se resetea la cola
    if(estado != old_estado && estado != still && old_estado != still){
        resetColaCircular();
    }

    // Se el nuevo rastreo
    anadirColaCircular(muestra);
}

int main(){
    int muestreos = 0;
    while(1){
        // Espera a una interrupción
        wfi();
        // Tras la interrupción, si ya se han realizado 100
        // muestreos, se calcula la velocidadActual
        if(muestreos+1 == 100){
            calcularVelocidadAngular();
        }
        muestreos = (muestreos + 1)%100

    }
}
```

Comentarios

Es el diseño más simple que se me ocurre y no se han considerado temas de concurrencia en el pseudocódigo. Está claro que la RSI del timer no es bueno que realice tantas tareas, debería almacenar los nuevos valores, guardar los viejos y calcular aparte el resto de cosas pero conllevaba más tiempo y no lo estimé oportuno.