



UNIVERSITAT POLITÈCNICA DE CATALUNYA

BARCELONATECH

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa



Grau en enginyeria en tecnologies aeroespacials

Laboratori de circuits electrònics

Memòria del projecte:
Control de la inclinació d'un braç giratori

Autors: Alexis Leon Delgado, Juan Garrido Moreno

Professorat: Víctor Manuel Sunyé Socias

Curs 2019-2020. Quadrimestre de tardor

Equip: Taula 6, braç giratori 5

Índex

1 Introducció	3
2 Primera fase	4
2.1 Tasca primera	4
2.2 Tasca segona	4
2.3 Tasca tercera	5
2.4 Tasca quarta	6
2.4.1 Control operatiu del motor	6
2.4.2 Control mitjançant introducció manual de dades	7
3 Segona fase	9
3.1 Cinquena tasca	9
3.1.1 Introducció i tractament de les dades introduïdes	9
3.1.2 Implementació de l'acció de control	9
3.1.3 Modificacions en el <i>setup</i>	9
3.2 Sisena tasca	11
3.3 Setena tasca	11
3.4 Vuitena tasca	13
4 Tercera fase	15
4.1 Novena tasca	15
4.1.1 Descripció d'un controlador PID	15
4.1.2 Implementació del control PID	16
4.2 Desena tasca	17
4.2.1 Consigna d'inclinació: 25%	19
4.2.2 Consigna d'inclinació: 50%	19
4.2.3 Consigna d'inclinació: 75%	20
4.3 Onzena tasca	21
5 Conclusions	23
5.1 Primera fase	23
5.2 Segona fase	23
5.3 Tercera fase	23
5.4 Possibles millores	23

1 Introducció

La finalitat última d'aquest projecte consisteix en la implementació d'un controlador d'inclinació d'un braç giratori fent ús d'una placa Arduino UNO. El sistema disposa d'una hèlix accionada gràcies a un motor, component que s'haurà de controlar mitjançant software per tal d'aconseguir el comportament del braç desitjat.

Amb l'objectiu d'assolir un correcte control amb realimentació del comportament de l'aparell, també es farà ús del sensor d'inclinació. Aquest, està conformat per un potenciòmetre integrat al braç i s'encarrega de proporcionar un voltatge variable en funció de l'angle adoptat pel mateix. L'esquema del sistema complet es pot observar a la següent figura:

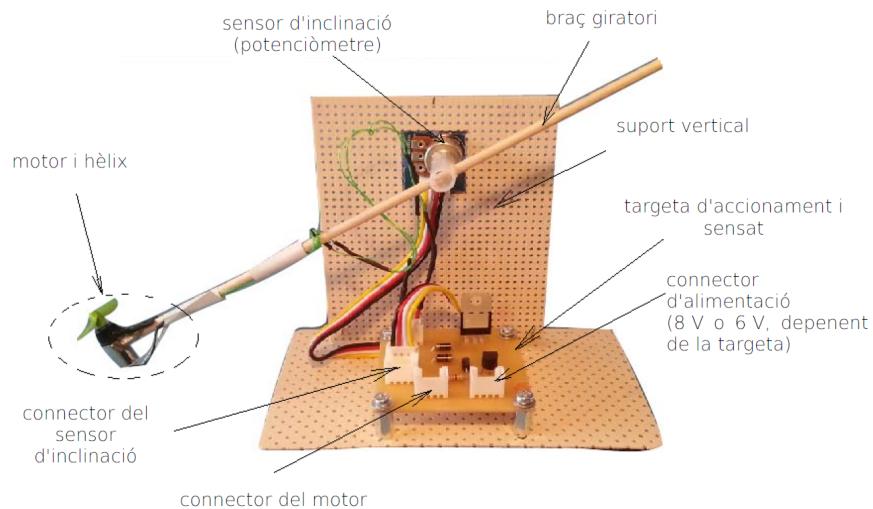


Figura 1: Esquema del braç giratori

2 Primera fase

2.1 Tasca primera

Mostrar en el monitor sèrie de l'entorn de desenvolupament de la targeta Arduino UNO (d'ara en endavant, monitor sèrie) l'enter que resulta de digitalitzar, usant el convertidor A/D incorporat a la targeta, la tensió v_{sens} (vegeu les figures 3 i 4) quan la inclinació del braç giratori és mínima i quan és màxima.

Abans d'iniciar la redacció del codi associat a les operacions demanades, és essencial ajustar una sèrie de paràmetres auxiliars. Al `void setup` s'inclouran aquelles sentències que hagin de ser executades un únic cop, a l'inici de l'execució del programa. L'ordre `Serial.begin` indica al sistema el nombre de bits per segon que s'estableixen en la connexió PC-placa Arduino. És un valor de 9600 que s'ha deixat fix per a tot el projecte.

La definició del tipus i nom de les variables manipulades en el posterior procediment es realitza mitjançant l'eina `#define`. Això suposa un avantatge important en programes d'elevada complexitat degut a que permet controlar totes les variables des d'un únic punt i de forma ordenada.

El convertidor A/D de la placa Arduino transforma l'entrada analògica: la tensió del potenciòmetre, en aquest cas, a una cadena de bits del tipus enter (definida com `int`). L'eina de codi que permet completar aquest subprocés és `analogRead`.

Per acabar, a l'interior del `loop`, sinònim de procés repetit en bucle, s'inclouen les comandes `analogRead`, que permet obté la senyal d'un cert pin de la placa (POT_PIN, en aquest cas) i l'emmagatzema en la variable especificada, anomenada `pot`; i `Serial.print`, que imprimeix el text i les variables indicades al Monitor Sèrie.

```

1 #define POT_PIN A0           //Definim una macro donat que aporta més comoditat
2 int pot;                   //Definim variable pot com enter
3
4 void setup() {
5   Serial.begin(9600);
6 }
7
8 void loop() {
9   pot = analogRead (POT_PIN); //Llegeix l'entrada POT_PIN (és a dir , A0) i la
                           //retorna com a variable int
10  Serial.print("Voltatge (V)= ");
11  Serial.println(pot);
12 }
```

Codi 1: Codi associat a la tasca 1 de la primera fase.

A continuació s'exposen lectures mínima i màxima, associades a una posició del braç d'inclinació mínima i màxima, respectivament.

$$V_{mín} = \boxed{9} \quad V_{máx} = \boxed{336}$$

És necessari esmentar que els valors previs oscil·laven amb una incertesa de ± 2 . Degut als posteriors càlculs que succeeixen aquesta tasca i la necessitat d'evitar qualsevol resultat negatiu que desvirtuï el sentit físic ha estat forçós d'emprar tant el màxim com el mínim absoluts observats. En addició, aquests valors mímmis i màxims no seran necessàriament els considerats per a les futures tasques, degut a que les condicions pròpies del sistema poden variar d'una sessió a l'altra.

2.2 Tasca segona

Mostrar en el monitor sèrie la inclinació del braç giratori expressada en tant per cent del canvi d'inclinació màxim. No es controlarà la freqüència de mostreig.

El percentatge és una variable numèrica que es definirà amb `float` que, a diferència de `int`, permet operar amb nombres decimals. No obstant, com l'operació per determinar aquest es realitza a partir d'enters, el sistema per inèrcia dona la resposta com a enter també. Aquest lleuger inconvenient es soluciona amb l'indicatiu `float` ubicat entre parèntesi precedent una de les variables emprades per al càlcul pertinent (veure línia 14 del codi 2), indicant així que les operacions es realitzin considerant les `int` com a `float`.

A continuació s'adjunta de forma clara el càlcul del percentatge:

$$\% = \frac{pot - V_{\min}}{V_{\max} - V_{\min}} \cdot 100$$

On *pot* es la lectura de la tensió del potenciòmetre convertida a digital. El denominador de la fracció serà sempre positiu, ja que $v_{\max} > v_{\min}$, però en cas que es donin fluctuacions en *pot*, és convenient d'assegurar que el numerador en cap moment esdevingui negatiu. Queda justificada la decisió presa vers els valors màxim i mínim a la tasca prèvia a aquesta.

```

1 #define POT_PIN A0
2 int pot;
3
4 float percentatge;
5 int Vmax=336; //Definim els límits determinats a l'apartat anterior
6 int Vmin=9;
7
8 void setup() {
9   Serial.begin(9600);
10 }
11
12 void loop() {
13   pot = analogRead(POT_PIN);
14   percentatge=((float)pot-Vmin)*100/(Vmax-Vmin); //Obliga a avaluar l'operació amb
15   //variable float (amb decimals)
16   Serial.print("Percentatge = ");
17   Serial.println(percentatge);
}

```

Codi 2: Codi associat a la tasca 2 de la primera fase.

2.3 Tasca tercera

Mostrar en el monitor sèrie la inclinació del braç giratori expressada en tant per cent del canvi d'inclinació màxim. La freqüència de mostreig haurà de ser $\approx 500\text{ Hz}$.

Empíricament es va observar que la freqüència de mostreig del senyal era elevada, tot i no haver estat definida en cap moment. Per controlar aquest procediment es requereix programar un temporitzador.

Les eines de control de temps no es troben incloses en la llibreria de funcions de sèrie de l'entorn de programació Arduino, raó per la qual s'han d'instal·lar addicionalment. La biblioteca emprada rep el nom de TimerOne, la qual s'ha de descarregar i instal·lar seguint les instruccions d'Atenea.

La biblioteca es carrega al codi a través de l'eina `#include` mitjançant la sintaxi que es mostra al Codi 3, línia 1.

L'enunciat específica com la freqüència de mostreig ha de ser de 500 Hz , i en conseqüència, el període és de $T = 0,002\text{ s}$. Nogensmenys, el processos informàtics es donen en intervals de temps ínfims, raó per la qual la biblioteca ha estat programada per manipular variables en dimensions de microsegons μs ; així, $T = 2000\text{ }\mu\text{s}$.

Un canvi implementat addicionalment és el qualificatiu *volatile* (veure Codi 3, línies 5 i 6), que ordena a la placa no optimitzar el senyal rebut a partir de la transmissió directa d'informació desde la RAM en comptes de la Flash [1]. Es recomana l'ús d'aquesta ordre quan el valor de la variable pot alterar-se sobtadament per raons alienes al procediment de treball usual. El procediment implementat esdevindrà un d'aquests casos al incloure un *Interrupt Service Routine* (ISR) que interromp l'acció quan el temporitzador s'atura. L'eina *volatile* protegeix així el valor de la variable.

La funció ISR compleix la particularitat de tractar-se d'una funció de no-retorn (no té entrades ni sortides) (veure Codi 3, línies 10-13) que s'executarà immediatament quan el cronòmetre arriba a T . Aquesta funció comprendrà la presa de dades i el càlcul del percentatge. En addició, es recomana que aquesta funció sigui el més curta possible per evitar qualsevol tipus de solapament amb l'inici d'altres processos.

Al *setup* inicial s'inclouen dues noves sentències. Primerament, amb *Timer1.initialize* es marca l'inici d'un cronòmetre amb el T definit prèviament. Aquest cronòmetre es va repetint de forma periòdica i automàtica.

Seguidament, amb *attachInterrupt* s'indica al programa que aturi qualsevol procediment per executar l'acció que se li indica: la funció de no-retorn prèviament definida. La combinació d'ambdues sentències provoca la repetició de la lectura del pin i el posterior càclul un total de 500 vegades per segon.

Per acabar, i com a detall, resulta curiós comentar com l'execució de la funció ISR és cíclica, tot i trobar-se fora del *loop*.

És necessari aclarir que, si bé el sistema realitza 500 lectures i càlculs de percentatge per segon, l'exposició per pantalla no es duu a terme simultàniament. Les ordres de *Serial.print* es troben al *loop*, el qual s'executa de manera independent de la funció ISR. Testimoni experimental és que, quan es va executar el programa al laboratori, la freqüència d'aparició dels resultats al Monitor Sèrie no s'apreciava com pròxima a 500 Hz.

```

1 #include <TimerOne.h>
2 #define POT_PIN A0
3 #define T 2000 //En microsegons
4
5 volatile int pot;
6 volatile float percentatge;
7 int Vmax=336;
8 int Vmin=9;
9
10 void ISR_llegeix_pot (void){                                //Funció de mostreig d'inclinació
11     pot = analogRead (POT_PIN);
12     percentatge=((float)pot-Vmin)*100/(Vmax-Vmin);
13 }
14
15 void setup() {
16     Serial.begin(9600);
17     Timer1.initialize (T);
18     Timer1.attachInterrupt (ISR_llegeix_pot);                  //Executem la funció d'interrupció
19 }
20
21 void loop() {
22     Serial.print("Percentatge = ");
23     Serial.println(percentatge);
24 }
```

Codi 3: Codi associat a la tasca 3 de la primera fase.

2.4 Tasca quarta

Generar la tensió v_{cont} (vegeu les figures 3, 4 i 5) i accionar el motor amb velocitat variable. El cicle de treball s'introduirà mitjançant el monitor sèrie en forma d'enter comprès entre 0 (correspondrà a $D = 0$) i 100 (correspondrà a $D = 1$).

El control del motor es realitza manualment a través d'una entrada numèrica al *Serial Monitor*. En conseqüència, és necessari d'incloure al codi les comandes per llegir entrades introduïdes per l'usuari, així com el control operatiu del motor. A fi de proporcionar una explicació més fidel al procediment experimental seguit, s'estudiaran separadament ambdós conceptes a implementar.

2.4.1 Control operatiu del motor

El motor s'alimenta amb una senyal de control v_{cont} . Aquesta és quadrada: només pot prendre valor 1 o 0; és d'amplitud unitària i pot ser periòdica o no en funció de la variació del paràmetre D , determinat en aquest cas per l'usuari.

En aquest cas és necessari l'ús de funció *Timer1.pwm* on les sigles PWM són l'acrònim de *Pulse-Width Modulation*. A aquesta sentència se l'ha d'especificar el pin de sortida (9 en aquest cas) i el paràmetre anomenat *Duty*, que és un enter positiu de 10 bits, sinònim de que el seu valor podrà variar entre 0 i 1023.

Per comprovar el correcte funcionament, al laboratori s'ha elaborat primerament el codi 4, que es limita a activar el motor a partir d'una variable de valor ja definit al propi codi. Per a valors de *Duty* entorn a 200 ~ 300, s'ha observat com el rotor s'activava, tot i que la força generada per l'hèlix era insuficient per a que el braç

recorregués un arc important. Per assolir aquest fenomen ha estat necessari un increment substancial del valor d'entrada fins a $800 \sim 900$. Tot seguit d'adjunta el codi emprat per realitzar les esmentades proves prèvies.

```

1 #include <TimerOne.h>
2 #define MOTOR_PIN 9
3 #define T 2000 //En microsegons
4
5 void setup() {
6   Serial.begin(9600);
7   Timer1.initialize(T);
8   Timer1.pwm(MOTOR_PIN, 800);
9 }
10
11 void loop() {
12 }
```

Codi 4: Codi associat a les proves prèvies referents a la tasca 4 de la primera fase.

2.4.2 Control mitjançant introducció manual de dades

Un cop realitzades les proves prèvies pertinents, es procedeix a escriure el codi que comprendrà la tasca 4. En aquest cas, s'ha partit precisament d'aquest Codi 4 ja existent de manera que s'han dut a terme petites modificacions per assolir específicament els requeriments demanats en aquesta tasca.

En primer lloc, s'ha afegit la funció *entrada de dades* (Veure Codi 5, línia 7), que no disposa de cap variable de sortida ni entrada. La finalitat de l'esmentada funció es llegir una variable d'entrada introduïda mitjançant el serial monitor i, en conseqüència, accionar el motor. Concretament, aquesta *input* és la velocitat de gir del motor en valor percentual i indicada per l'usuari.

Primerament, s'imposa la condició *Serial.available()>0*, que interroga el sistema per veure si existeix alguna entrada pendent al serial monitor. Així mateix, si existeix alguna, envia la resposta corresponent. Per possibilitar l'entrada de l'acció de control manual, s'introduceix dins de l'esmentada condició la comanda *Serial.parseInt()*, que converteix l'entrada numèrica en format ASCII rebuda per l'Arduino a format decimal. El resultat d'aquesta sentència es guarda en la variable D i s'imposa una condició de saturació, que limita l'entrada a un rang de 0-100. Tot seguit s'empra la comanda *Timer1.setPwmDuty*, que imposa un nou PWM amb valor $10.23*D$ per complir amb l'interval vàlid de *Duty* entre 0 i 1023. Per acabar es mostra per pantalla la velocitat de gir percentual seleccionada.

Pel que fa al *setup*, seguint el procés emprat al Codi 3, s'ha afegit la sentència *Timer1.pwm* (Veure Codi 5, línia 21), la qual s'avalua en *MOTOR_PIN* i amb el valor 0 com a entrada per tal de configurar inicialment el pin.

Finalment, es porta a terme la crida de la funció *entrada_dades* a la secció del *loop*. El codi resultant s'adjunta seguidament.

```

1 #include <TimerOne.h>
2 #define T 2000 //En microsegons
3 #define MOTOR_PIN 9
4
5 int D;
6
7 void entrada_dades (void){                                //Funció d'entrada manual dades
8   if (Serial.available(>0)){                            //Llegeix l'enter D
9     D = Serial.parseInt();                           //Corregeix el valor introduït en
10    if (D>100) D=100;                                 cas que calgui
11    if (D<0) D=0;                                   
12    Timer1.setPwmDuty (MOTOR_PIN, 10.23*D);          //Imprimeix la velocitat de gir
13    Serial.print("Velocitat de gir seleccionada (%): ");
14    Serial.println(D);                                //Aquesta comanda imprimeix el valor de D
15  }
16}
17
18 void setup() {
```

```
19 Serial.begin(9600);
20 Timer1.initialize(T);
21 Timer1_pwm(MOTOR_PIN,0);
22 }
23
24 void loop() {
25     entrada_dades();
26 }
```

Codi 5: Codi referent a la tasca 4 de la primera fase.

3 Segona fase

Durant la fase primera s'ha realitzat un estudi general de les diferents possibilitats de modificació que ofereix el sistema que s'analitza. Un cop ja es dominen aquestes eines, és possible fer un pas més enllà i introduir conceptes de control automàtic, com és el cas de la realimentació, amb el fi d'aconseguir que el sistema presenti cert comportament sense cap més acció de l'usuari que la posada en marxa.

3.1 Cinquena tasca

Implementar un control proporcional de la inclinació del braç. La freqüència amb què es mostrejarà la inclinació i s'actualitzarà el cicle de treball del senyal v_{cont} serà de $\approx 500\text{ Hz}$. La inclinació de referència o consigna s'introduirà a través del monitor sèrie en forma d'enter comprès entre 0 (0% d'inclinació) i 100 (100% d'inclinació).

3.1.1 Introducció i tractament de les dades introduïdes

Seguint la metodologia de la quarta tasca, s'incorpora novament la introducció de dades a través del *Serial Monitor*.

Per efectuar una correcta comparació, és necessari que les variables es trobin en la mateixa escala: la lectura del pin del potenciómetre es troba entre V_{min} i V_{max} , i en conseqüència l'entrada aplicada per l'usuari compresa de 0 a 100, s'haurà de transformar a un nombre enter comprés dins el primer interval. La transformació es realitza dins la funció *entrada_dades*, concretament amb la sentència ubicada a la línia 17 del Codi 6. En cas que l'entrada es trobin fora del marge mencionat, les línies 15 i 16 del Codi 6 fiten l'entrada al valor màxim o mínim, segons convingui.

Cal esmentar com els valors V_{max} i V_{min} difereixen respecte de la fase anterior degut a la variació del sistema d'una sessió a l'altra. Per a aquesta fase,

$$V_{min} = \boxed{1} \quad V_{max} = \boxed{340}$$

Al Codi 6 no ha existit una definició directa d'aquests perquè ja s'han tingut en compte a l'operació que transforma l'angle introduït manualment a l'escala del potenciómetre.

3.1.2 Implementació de l'acció de control

Es defineix acció de control com aquell resultat que es reimplementat al sistema a partir de la comparació de la sortida del sistema amb la consigna introduïda, paràmetre anomenat error.

L'acció de control proporcional és la més senzilla i consisteix en el producte de l'error per una constant de proporcionalitat K_p .

$$P(t) = K_p \cdot e(t)$$

On $P(t)$ és l'acció de control i $e(t)$ l'error, ambdós en funció del temps. Observi's com aquesta acció de control es basa únicament en l'estat instantani del sistema.

Amb la finalitat de portar a terme el control proporcional, es defineix la funció de no-retorn *control_P*, la qual esdevindrà rutina d'interrupció. Dins d'aquesta funció es troba la comanda *analogRead* (línia 24 del Codi 6), per tal d'obtenir el valor puntual d'angle mesurat pel potenciómetre i, a continuació, comparar-lo amb la consigna introduïda per l'usuari (ja convertida a l'escala convenient). El fruit de la comparació és l'error, el qual es multiplica per la constant arbitrària K_p per obtenir la corresponent acció de control.

Per tal d'abordar la possibilitat de l'obtenció d'un valor de control proporcional P superior a 1023 o inferior a 0, s'afegeixen dues condicions de saturació. En el cas que $P \geq 1023$, es retorna $P = 1023$. En canvi, si $P \leq 0$ la P obtinguda és tal que $P = 0$.

3.1.3 Modificacions en el *setup*

En quant al *setup* es refereix, s'ha emprat la mateixa estructura de codi que la utilitzada al codi 5. Novament l'entrada que rep per primer cop el motor mitjançant la sentència *Timer1.pwm* és 0, inicialitzant així el programa amb el motor apagat.

Per tal d'implementar la realimentació del sistema, la comanda `Timer1.attachInterrupt` interromp el `loop` i executa la funció `control_P`. D'aquesta manera, cada $2000\ \mu s$ es recalcula el valor de P amb les dades d'entrada corresponents a aquest nou estat del sistema. Aquest procés es realitza amb la finalitat d'actualitzar el valor de P, aconseguint així l'esmentada realimentació.

Per acabar, al `loop` només hi apareix la funció `entrada_dades`, que tal i com es troba definida només s'executarà completament quan l'usuari introdueixi dades i premi la tecla `Enter`.

A continuació es mostra el codi corresponent a la tasca 5. Es necessari esmentar la variació dels paràmetres V_{max} i V_{min} , reajustats a fi que el percentatge estigui en tot moment entre 0 i 100. Es recorda que els límits poden variar lleugerament d'una sessió experimental a l'altra.

```

1 #include <TimerOne.h>
2 #define POT_PIN A0
3 #define MOTOR_PIN 9
4 #define T 2000 //En microsegons
5 #define Kp 2500
6
7 volatile int pot;                                //Definides com a volatile per evitar optimitzacions
8 volatile float P;                               //Funció d'entrada manual de dades
9 volatile float error;                           //Llegeix l'enter anglePercentual
10 int angle;                                     //Corregeix el valor introduït (en
11                                         cas que calgui)
12 void entrada_dades (void){                      //Conversió de angle entre 0-100 a
13     if (Serial.available()>0){                   bits equiv.
14         int anglePercentual = Serial.parseInt();   //Funció de l'acció de control P
15         if (anglePercentual>100) anglePercentual=100; //Es calcula l'error entre l'angle
16         if (anglePercentual<0) anglePercentual=0;    introduït (ja convertit a format bits) i pot, que és la mesura real feta pel
17         angle=3.39*anglePercentual+1;               potenciómetre
18         Serial.print("Angle seleccionat (%): ");    //Es determina el valor de P
19         Serial.println(anglePercentual);             mitjançant la multiplicació de la constant Kp per l'error en aquest instant
20     }
21 }
22
23 void control_P (void){                          //S'estableix una condició de
24     pot = analogRead (POT_PIN);                  saturació superior per P
25     error=angle-pot;                            //S'estableix una condició de
26     angle=pot*Kp;                             saturació inferior per P
27     if(P>1023) P=1023;                         Timer1.setPwmDuty (MOTOR_PIN, P);
28     if(P<0) P=0;                               //S'interromp el loop per executar
29     Timer1.setPwmDuty (MOTOR_PIN, P);           la funció control_P, que actualitza P
30 }
31
32 void setup() {
33     Serial.begin(9600);
34     Timer1.initialize (T);
35     Timer1.pwm (MOTOR_PIN,0);
36     Timer1.attachInterrupt(control_P);          //S'interromp el loop per executar
37 }                                              la funció control_P, que actualitza P
38
39 void loop() {
40     entrada_dades();
41 }
```

Codi 6: Codi referent a la tasca 5 de la segona fase.

3.2 Sisena tasca

Determinar el valor mínim K_u del guany proporcional que provoca que el sistema oscil·li permanentment amb una consigna d'inclinació del 50%.

Aquesta tasca coincideix amb el primer pas del mètode de Ziegler-Nichols, on es demana trobar l'anomenada constant de proporcionalitat crítica per a la qual la resposta del sistema és una oscil·lació estable d'amplitud constant.

Per obtenir el valor demandat, progressivament s'incrementa el valor de K_p fins a observar el comportament prèviament descrit. Per a aquest sistema el valor de **constant de proporcionalitat crítica** ha estat el següent:

$$K_{p, \text{crítica}} \equiv K_u \approx 3500$$

Aquest valor ha estat trobat a partir d'un augment progressiu de K_p fins a trobar un valor que proporcionés una oscil·lació constant i mantinguda. Com a resum del comportament observat durant el procés d'augment progressiu de K_p , inicialment, el sistema presenta una oscil·lació esmoreïda fins a un valor molt superior a $K_p = 1000$. Així mateix, per a $K_p \gtrsim 2500$ el moviment periòdic és observable per a un temps major. El resultat s'ha intentat acotar en ordres de 100 fins a arribar al valor de $K_{p, \text{crítica}}$ anteriorment exposat.

Cal esmentar, doncs, que es tracta d'un valor aproximat, degut a que seria necessari un llarg temps d'observació i un minuciós ajust a fi de trobar el primer punt que assegura un esmoreïment totalment nul.

3.3 Setena tasca

Obtenir una representació gràfica en funció del temps de la inclinació del braç amb una consigna d'inclinació del 50% i a partir d'ella, mesurar el valor T_u del període d'oscil·lació corresponent.

El codi emprat per a aquesta segona tasca presenta una estructura molt similar al Codi 6 prèviament exposat. Per aquesta raó únicament s'especificarà quines modificacions s'han introduït i sobre quins mòduls s'han aplicat.

En primer lloc, pel que fa a definició de variables, s'ha fixat la constant K_p al valor crític determinat a l'apartat anterior. També s'ha definit la variable *percentatge* en format *volatile float* per tal d'evitar la seva optimització. A més, s'ha inicialitzat la variable *t*, que posteriorment s'emprarà per la representació gràfica de la resposta.

Pel que fa a les funcions implementades, en primer lloc, s'ha eliminat la funció *entrada_dades()*. Això és degut a que no és necessària la introducció de la consigna d'inclinació pel serial monitor ja que és un valor constant específic per aquest apartat, concretament del 50%. És per aquest motiu que, per tal d'establir el valor d'aquesta consigna, es defineixen les línies de codi 14 i 15, on es té en compte els valors de *Vmax* i *Vmin* per calcular la variable *angle* en format bits.

En segon lloc, en quant a la funció *control_P()*, s'ha addicionat la variable *percentatge* a la línia 19, que calcula la inclinació real del braç en format percentual i ja explicada prèviament al Codi 3. A més, s'ha definit la variable temporal *t*, la qual augmenta progressivament de valor a la línia 25 del Codi 7. La finalitat d'aquesta expressió és definir el vector temps que utilitzarà Matlab per dur a terme la representació temporal corresponent tot tenint en compte el temps de mostreig de 2 ms.

Finalment, l'última modificació realitzada ha sigut la del *loop()*. En aquest cas, s'han afegit els *Serial.print* de les línies 36-40 per tal d'obtenir les ternes (*t*, *anglePercentual*, *percentatge*) necessàries per graficar la resposta temporal del braç mitjançant Matlab.

A continuació, es mostra el codi resultant:

```

1 #include <TimerOne.h>
2 #define POT_PIN A0
3 #define MOTOR_PIN 9
4 #define T 2000 //En microsegons
5 #define Kp 3500
6
7 volatile int pot;
8 volatile float percentatge;
9 int Vmax=340;
```

```

10 int Vmin=1;
11 volatile float P;
12 volatile float error;
13 float t=0; // Inicialitzem el temps a 0
14 int anglePercentual=50; //S'escull el valor de la consigna
15 int angle=3.39*anglePercentual+1; //Es transforma la consigna a bits
16
17 void control_P (void){ //Funció de l'acció de control P
18   pot = analogRead (POT_PIN);
19   percentatge=((float)pot-Vmin)*100/(Vmax-Vmin);
20   error=angle-pot;
21   P=Kp*error;
22   if (P>1023) P=1023;
23   if (P<0) P=0;
24   Timer1.setPwmDuty (MOTOR_PIN, P); //S'incrementa el comptador de temps
25   t=t+0.002;
26 }
27
28 void setup() {
29   Serial.begin(9600);
30   Timer1.initialize (T);
31   Timer1.pwm (MOTOR_PIN,0);
32   Timer1.attachInterrupt(control_P);
33 }
34
35 void loop(){
36   Serial.print(t);
37   Serial.print(" , ");
38   Serial.print(anglePercentual);
39   Serial.print(" , ");
40   Serial.println((int)percentatge); //Imprimim el percentatge com a enter
41 }

```

Codi 7: Codi referent a la tasca 7 de la segona fase.

Posteriorment, emprant l'arxiu de Matlab *llegeix_temps_percentatges.m*, proporcionat pel personal docent, s'obté la representació temporal mostrada a la Figura 2, on ambdues gràfiques il·lustren el mateix experiment, però en escales temporals diferents a fi d'ofrir una perspectiva més clara.

Observi's com la gràfica esquerra, si bé presenta una amplitud lleugerament superior a l'inici, en $\approx 4\text{ s}$ assoleix un estat estacionari oscil·lant sense variacions massa significatives en l'amplitud, aportant una demostració gràfica de que l'estat assolit és el demanat.

En base a la gràfica dreta, és possible determinar el període d'oscil·lació comparant diversos parells de pics consecutius, realitzant així una mitjana per assegurar un valor fiable de T_u . En aquest cas, el valor mitjà del **període d'oscil·lació crític** ha estat de:

$$T_u \approx [0, 6\text{ s}]$$

D'altra banda, s'observa també com la resposta no oscil·la entorn el punt d'equilibri, sinó d'un valor proper però inferior. Per quantificar l'error, es trobarà el valor mig de la resposta a partir de dividir entre 2 la suma del punt d'oscil·lació superior i de l'inferior. Aquest valors són 51 i 44%, respectivament, i el valor resultant:

$$\bar{R} = \frac{51 + 44}{2} = 47,5\%$$

On \bar{R} representa el valor mig de la resposta.

L'error relatiu respecte de la consigna és:

$$\varepsilon_{r, 50, crit} = \frac{50 - 47,5}{50} \cdot 100 = 5\%$$

És una desviació petita que podrà ser corregida amb la implementació d'una component de control integral, com es veurà en futurs apartats.

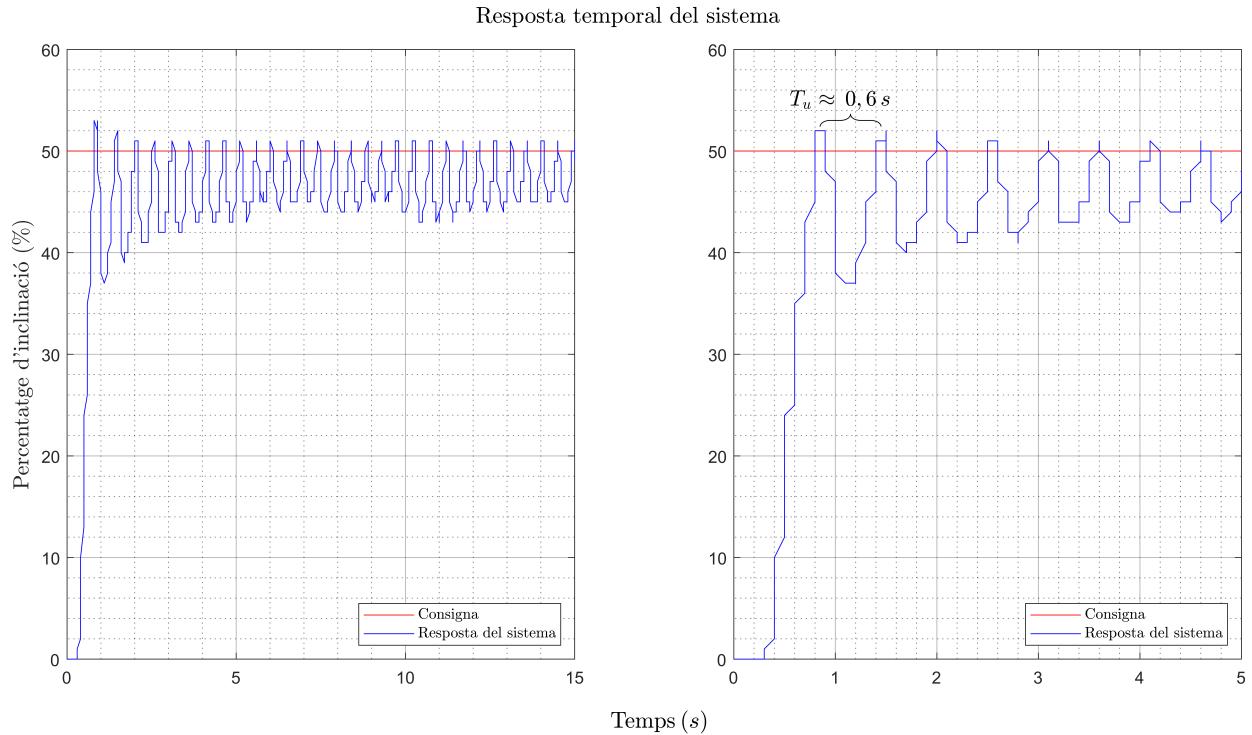


Figura 2: Gràfic de la resposta temporal del sistema vers la consigna del 50%.

3.4 Vuitena tasca

Ajustar el valor del guany proporcional usant el mètode de Ziegler-Nichols i comprovar experimentalment el funcionament del control ajustat. Obtenir una representació gràfica en funció del temps de la inclinació del braç amb una consigna d'inclinació del 50%.

Un cop determinats K_u i T_u , el mètode Ziegler-Nichols estableix unes expressions de referència (veure Taula 1) per determinar els valors que s'han d'aplicar al controlador, que a partir d'ara rebran el nom d'*ideals*.

Controlador	K_p	T_i	T_d	K_i	K_d
P	$0,5K_u$	-	-	-	-
PI	$0,45K_u$	$T_u/1,2$	-	$0,54K_u/T_u$	-
PID	$0,6K_u$	$0,5T_u$	$0,125T_u$	$1,2K_u/T_u$	$3K_uT_u/40$

Taula 1: Relacions entre paràmetres del controlador i característiques de l'oscil·lació mantinguda.

Aquestes expressions varien en funció del tipus de controlador que es desitja implementar al sistema. Observi's com els paràmetres associats als termes integral i derivatiu s'expressen de dues formes distintes en dependència de la funció de transferència considerada per al controlador:

$$G_{PID}(s) = K_p + \frac{K_i}{s} + K_d s = K_p \left(1 + \frac{1}{T_i s} + T_d s \right)$$

Cal detallar que la forma considerada per a l'estudi realitzat ha estat la primera, treballant únicament amb constants K . En aquesta tasca es demana un control exclusivament proporcional, de forma que la **constant de control proporcional ideal** es trobarà amb l'expressió següent:

$$K_{p,ideal} = 0,5 \cdot K_u = 1750$$

Aquest valor ha estat implementat al sistema. Si la K_u trobada anteriorment és correcta, el fenomen que s'observarà degut a aquesta $K_{p,ideal}$ serà una oscil·lació esmorteïda amb una tendència progressiva a un valor final no necessàriament igual a la consigna.

La inclinació demanada del braç és també del 50%. Amb la finalitat d'obtenir la representació gràfica associada, es segueix la mateixa metodologia de la tasca anterior tot emprant el Codi 7 amb el corresponent valor ideal de K_p . La representació temporal resultant es mostra a la Figura 3.

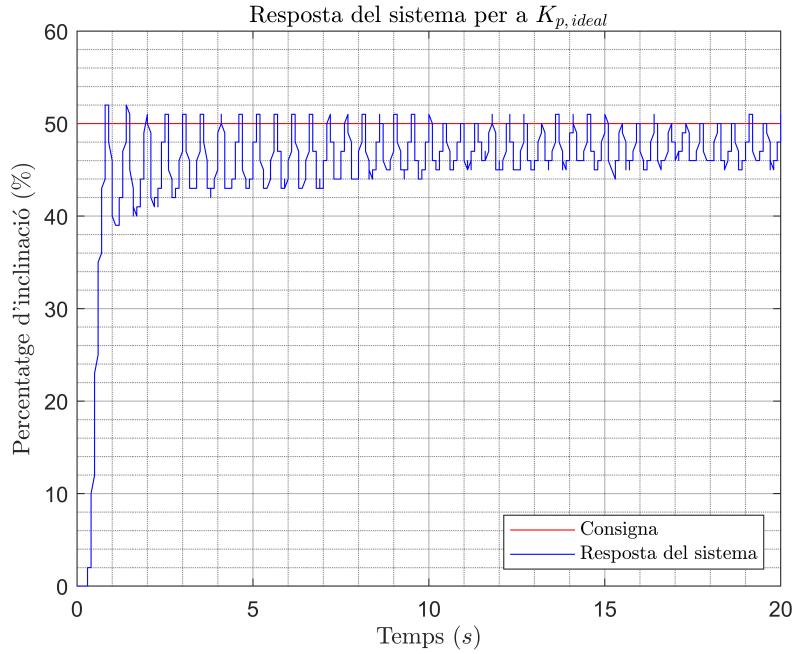


Figura 3: Gràfic de la resposta temporal del sistema amb un control proporcional amb $K_p = K_{p,ideal}$.

Observi's com la resposta, si bé qualitativament similar a la Figura 2, mostra un molt lleuger però progressiu esmoreïment amb el temps. Cal mencionar que, en aquest cas, el valor ideal de K_p calculat mitjançant Ziegler-Nichols proporciona un comportament suficientment satisfactori encara que no excel·lent donat la relativa lentitud de la resposta a esmoreir-se. Una altra conclusió és que es corrobora així com el valor crític de K_u és superior al valor ideal de K_p aplicat sobre el sistema. D'altra banda, es demostra la dificultat per localitzar aquest valor, ja que experimentalment és difícil apreciar quan s'assoleix l'estat crític demanat. Ha estat aquesta raó la que ha portat a imposar un temps d'enregistrament major, en aquest cas de 20 s, aconseguint així una millor perspectiva de l'evolució.

Un fenomen ja observat a la setena tasca és la desviació permanent respecte de la consigna a l'estat estacionari. En aquest cas, el valor d'inclinació al voltant del qual es produeix l'oscillació és:

$$\bar{R} = \frac{50 + 44}{2} = 47\%$$

Valor lleugerament inferior al de la tasca anterior.

L'error relatiu respecte de la consigna introduïda és:

$$\varepsilon_{r, 50, crit} = \frac{50 - 47}{50} \cdot 100 = 6\%$$

En base a aquest resultat i al de la tasca anterior, es demostra quantitativament com la incorporació d'un control únicament proporcional no és eficaç per a l'obtenció d'un error estacionari nul, tot i que per a certes aplicacions de baixa precisió podria ésser vàlid.

4 Tercera fase

4.1 Novena tasca

Implementar un control proporcional-integral-derivatiu de la inclinació del braç. La freqüència amb què es mostrejarà la inclinació i s'actualitzarà el cicle de treball del senyal v_{cont} serà de $\approx 500\text{ Hz}$. La inclinació de referència o consigna s'introduirà a través del monitor sèrie en forma d'enter comprès entre 0 (0% d'inclinació) i 100 (100% d'inclinació).

4.1.1 Descripció d'un controlador PID

Un controlador PID emet una acció de control basada en la suma de tres transformacions independents efectuades sobre l'error mesurat.

La primera contribució és la proporcionalitat, ja descrita a l'apartat 3.1.2 *Implementació de l'acció de control*.

Acció de control integral

Integrar la progressió temporal de l'error permet tenir en consideració els estats anterior pels quals ha passat l'error fins a arribar al valor actual i elaborar una resposta en conseqüència. D'aquesta forma es complementa l'acció proporcional, que només considera exclusivament l'estat actual del sistema. A nivell pràctic, l'acció integradora opera correctament quan la sortida acaba tendint a la consigna sense oscil·lar.

La seva definició formal és mostra a continuació:

$$I(t) = K_i \cdot \int_0^{t_0} e(t) dt$$

On K_i és la constant de l'acció integral.

La seva representació com a successió de termes discrets és la següent:

$$I(t) = K_i T \sum_{i=1}^n e_i \quad (1)$$

Serà aquesta última expressió la que s'implementarà en el codi Arduino, degut a que no es disposa de cap biblioteca que executi integrals numèriques de forma automàtica.

En aquest cas, T farà referència al diferencial de temps, emprant en aquest cas el valor corresponent al temps de mostreig.

Acció de control derivativa

Introduir l'acció derivativa sobre l'error permet avaluar la tendència d'aquest, possibilitant així efectuar una predicción en el marge de la realitat de l'estat futur del sistema, i emetent una resposta per redirigir el sistema amb més precisió a la consigna.

La definició matemàtica original és la següent:

$$D(t) = K_d \cdot \frac{d}{dt} e(t)$$

On K_d és la constant de l'acció integral.

Novament existeix una expressió discretitzada que ofereix una implementació més senzilla:

$$D(t) = K_d \frac{e_n - e_{n-1}}{T} \quad (2)$$

On T s'identifica un cop més amb el diferencial de temps, virtualment infinitesimal però realísticament discret; e_n és l'error instantani entre la consigna i la resposta puntual i e_{n-1} és l'error associat a l'execució immediatament anterior.

Cal destacar però, que aquesta implementació presenta una gran sensibilitat vers el soroll, de forma que la constant K_d no pot prendre valors massa elevat ja que dividint per T ja s'assoleixen magnituds d'ordre elevat.

4.1.2 Implementació del control PID

Pel que fa al codi programat per l'actual tasca, s'ha emprat com a base el Codi 6, de manera que s'han implementat les accions de control integral i derivativa a la ja existent proporcional.

Per tal de programar l'acció integral, s'ha definit la variable *error*, que representa l'error acumulat pel sistema amb el transcurs del temps. Aquest valor s'aconsegueix realitzant un sumatori de l'error instantani del sistema, *error*, i l'error acumulat en els instants immediatament anteriors, *error*. Aquest últim valor es multiplica per *TSEC* (en segons) i per la constant K_i , d'acord amb l'expressió (1).

Cal esmentar que per tal d'implementar un sistema *anti-windup*¹, s'ha d'establir una condició de saturació per l'acció de control integral. És així que s'ha pres $I = 1000$ com a límit superior i $I = 0$ com a l'inferior. En addició, la raó perquè el límit superior no arribi a 1023 és deixar un lleuger marge per a les altres components del control.

L'acció derivativa s'ha implementat seguint l'expressió (2). Per assolir una correcta implementació és imprescindible la definició d'una nova variable anomenada *error_prev* (veure línia 48 del codi 8). Aquesta variable veu configurat el seu valor just al final de la funció de control, i emmagatzema l'error instantani per a comparar-lo amb l'error d'una execució posterior, tal com exigeix l'expressió discretitzada de la derivada. Posteriorment, aquest resultat es divideix per *TSEC*, que vol representar el diferencial de temps, i es multiplica per la constant K_d .

Anteriorment s'ha comentat com la component derivativa és molt sensible al soroll. Per aquesta raó, amb les línies 42-43 del Codi 8 s'estableixen límits superior i inferior per a aquest valor, així com també es fa per les altres contribucions a les línies de codi anteriors.

D'altra banda, i degut a que l'enunciat de la guia no ho demana, s'ha eliminat la variable *percentatge* i, en conseqüència la sentència per al seu càlcul, així com els límits d'inclinació màxima i mínima *Vmax* i *Vmin*.

```

1 #include <TimerOne.h>
2 #define POT_PIN A0
3 #define MOTOR_PIN 9
4 #define T 2000 //En microsegons
5 #define TSEC 0.002 //En segons
6 #define Kp 0
7 #define Ki 0
8 #define Kd 0
9
10 volatile int pot;
11 volatile float P;
12 volatile float I;
13 volatile float D;
14 int angle;
15 volatile float error;
16 volatile float error_prev=0;           //S'inicialitza l'error de la mesura prèvia
17 float e=0;                           //S'inicialitza l'error acumulat
18 float u;                            //Es defineix l'acció de control
19
20 void entrada_dades (void){           //Funció d'entrada manual de dades
21   if (Serial.available()>0){
22     int anglePercentual = Serial.parseInt();
23     if (anglePercentual>100) anglePercentual=100;
24     if (anglePercentual<0) anglePercentual=0;
25     angle=3.39*anglePercentual+1;
26     Serial.print("Angle seleccionat (%): ");
27     Serial.println(anglePercentual);
28   }
29 }
30
31 void control_PID (void){           //Funció de l'acció de control PID
32   pot = analogRead (POT_PIN);
33   error=angle-pot;
34   P=Kp*error;

```

¹Sistema que evita que l'acció integradora continuï integrant en instants posteriors a la saturació de l'entrada. Aquesta compensació es pot aconseguir mitjançant l'acotació de la resposta per sota dels límits de saturació

```

35 if (P>1023) I=1023;                                // Condició de saturació superior per I
36 if (P<0) I=0;                                     // Condició de saturació inferior per I
37 e=e+error;
38 I=Ki*e*TSEC;
39 if (I>1023) I=1000;                                // Condició de saturació superior per I
40 if (I<0) I=0;                                     // Condició de saturació inferior per I
41 D=Kd*(error-error_prev)/TSEC;
42 if (D>1023) D=1023;                                // Condició de saturació superior per I
43 if (D<0) D=0;                                     // Condició de saturació inferior per I
44 u=P+I+D;
45 if (u>1023) u=1023;                                // Condició de saturació superior per u
46 if (u<0) u=0;                                     // Condició de saturació inferior per u
47 Timer1.setPwmDuty (MOTOR_PIN, u);                //S'emmagatzema l'error
48 error_prev=error;
49 }
50
51 void setup() {
52   Serial.begin(9600);
53   Timer1.initialize (T);
54   Timer1.pwm (MOTOR_PIN,0);
55   Timer1.attachInterrupt(control_PID);
56 }
57
58 void loop() {
59   entrada_dades();
60 }
```

Codi 8: Codi referent a la tasca 9 de la segona fase.

4.2 Desena tasca

Ajustar el valor dels guanys usant el mètode de Ziegler-Nichols (enllaç Wikipedia) i comprovar experimentalment el funcionament del control ajustat. Obtenir una representació gràfica en funció del temps de la inclinació del braç amb tres consignes d'inclinació: 25%, 50%, 75%.

Les modificacions aplicades sobre el codi de la tasca anterior són, a efectes pràctics, les mateixes que s'han aplicat del Codi 6 al 7.

En aquest cas, els valors de les constants del controlador són els ideals determinats amb Ziegler-Nichols, tot i que podrien ser arbitràriament modificats per l'usuari. Així mateix, la consigna d'inclinació apareix fixada al 50%, tot i que també podria ser modificada. Es defineixen novament les variables *percentatge* i *t* (veure línies 11 i 21 del Codi 9), així com les expressions corresponents per calcular-les (línies 28 i 45, respectivament).

D'altra banda, es suprimeix la funció *entrada_dades()* degut a que la consigna és fixa i al *loop* únicament s'inclouen sentències per representar pel *serial monitor* les dades que el Matlab ha d'enregistrar.

```

1 #include <TimerOne.h>
2 #define POT_PIN A0
3 #define MOTOR_PIN 9
4 #define T 2000 //En microsegons
5 #define TSEC 0.002 //En segons
6 #define Kp 2100
7 #define Ki 7000
8 #define Kd 158
9
10 volatile int pot;
11 volatile float percentatge;
12 int Vmax=340;
13 int Vmin=1;
14 volatile float P;
15 volatile float I;
16 volatile float D;
17 volatile float error;
18 volatile float error_prev=0;                      //S'inicialitza l'error de la mesura
19 float e=0;                                         //S'inicialitza l'error acumulat
20 float u;                                           //Es defineix l'acció de control
```

```

21 float t=0;                                // Inicialitzem el temps a 0
22 int anglePercentual=50;                   // S'escull el valor de la consigna
23 int angle=3.39*anglePercentual+1;          // Es transforma la consigna a bits
24
25
26 void control_PID (void){                  // Funció de l'acció de control PID
27     pot = analogRead (POT_PIN);
28     percentatge=((float)pot-Vmin)*100/(Vmax-Vmin);
29     error=angle-pot;
30     P=Kp*error;
31     if (P>1023) I=1023;
32     if (P<0) I=0;
33     e=e+error;
34     I=Ki*e*TSEC;
35     if (I>1023) I=1000;
36     if (I<0) I=0;
37     D=Kd*(error-error_prev)/TSEC;
38     if (D>1023) I=1023;
39     if (D<0) D=0;
40     u=P+I+D;
41     if (u>1023) u=1023;
42     if (u<0) u=0;
43     Timer1.setPwmDuty (MOTOR_PIN, u);
44     error_prev=error;
45     t=t+0.002;                            // S'incrementa el comptador de temps
46 }
47
48 void setup() {
49     Serial.begin(9600);
50     Timer1.initialize (T);
51     Timer1.pwm (MOTOR_PIN,0);
52     Timer1.attachInterrupt(control_PID);
53 }
54
55 void loop() {
56     Serial.print(t);
57     Serial.print(" , ");
58     Serial.print(anglePercentual);
59     Serial.print(" , ");
60     Serial.println((int)percentatge);      // Imprimim el percentatge com a enter
61 }
```

Codi 9: Codi referent a la tasca 10 de la segona fase.

Els valors del controlador que apareixen al Codi 9 es justifiquen fent ús de la tercera fila de la Taula 1, es determinen els valors ideal de K_i , K_i i K_d segons el mètode Ziegler-Nichols pel cas d'un controlador PID.

En primer lloc, la **constant de control proporcional** es determina mitjançant l'expressió que segueix:

$$K_{p,ideal} = 0,6K_u = 0,6 \cdot 3500 = \boxed{2100}$$

Observi's com és un valor diferent al cas del control proporcional pur.

D'altra banda, la **constant de control integral** s'obté com:

$$K_{i,ideal} = 1,2 \frac{K_u}{T_u} = 1,2 \cdot \frac{3500}{0,60} = 7000 = \boxed{7 \cdot 10^3}$$

Finalment, la **constant de control derivatiu** es determina amb l'expressió a continuació:

$$K_{d,ideal} = \frac{3}{40} K_u T_u = \frac{3}{40} \cdot 3500 \cdot 0,60 = 157,5 \approx \boxed{158}$$

Un cop obtinguts els valors ideals, es procedeix a testejar el braç amb els esmentats valors de control per diferents consignes d'inclinació: 25%, 50% i 75%.

4.2.1 Consigna d'inclinació: 25%

Pel que a la primera consigna respecta, el comportament del braç es mostra a la Figura 4.

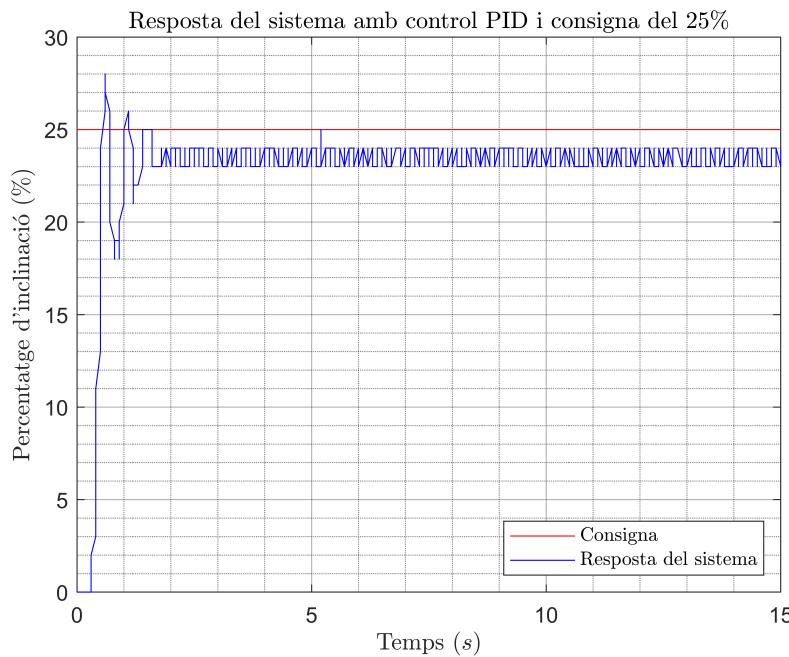


Figura 4: Gràfic de la resposta temporal del sistema amb un control PID i consigna d'inclinació del 25%.

Tal i com s'esperava amb la inclusió de l'acció derivadora i la integral, la resposta s'esmorteix ràpidament, concretament en menys de 2 s, però presentant dos sobrepics inicials: un que sobrepassa la consigna en 3 unitats (diferència del 12%) i altre que la sobrepassa en 1 (diferència del 4%).

Es coneix que l'acció integral actua en part correctament degut a que la resposta del sistema s'estabilitza al voltant d'un valor i sense presentar oscil·lació. Nogensmenys, la resposta presenta un error estacionari vers la consigna, o dit d'una altra forma, la funció de transferència presenta un guany estàtic inferior a 1, de forma que l'acció integral no compleix totalment la seva funció. Aquest valor final no és totalment exacte, sinó que varia irregularment entre el 23 i 24% d'inclinació. Prengent com a valor intermitg 23,5%, es pot afirmar que l'error relatiu respecte la consigna és:

$$\varepsilon_{r, 25} = \frac{25 - 23,5}{25} \cdot 100 = 6\%$$

Tot i que el controlador aplicat sobre el sistema és del tipus PID, la configuració de les constants ha estat dissenyada per a operar amb una consigna del 50%. És la imposició d'una entrada diferent a l'esperada la que condueix a l'aparició de les anomalies esmentades.

No obstant, els valors de sobrepic i error estacionari no són significativament alts per afirmar que la implementació del control amb les constants no óptimes segons Ziegler i Nichols ocasioni un control inocu o inestable; al contrari, el resultat és acceptable.

4.2.2 Consigna d'inclinació: 50%

En aquest apartat s'estudia la resposta del sistema amb control PID per a la consigna de disseny. La representació temporal es mostra a la Figura 5.

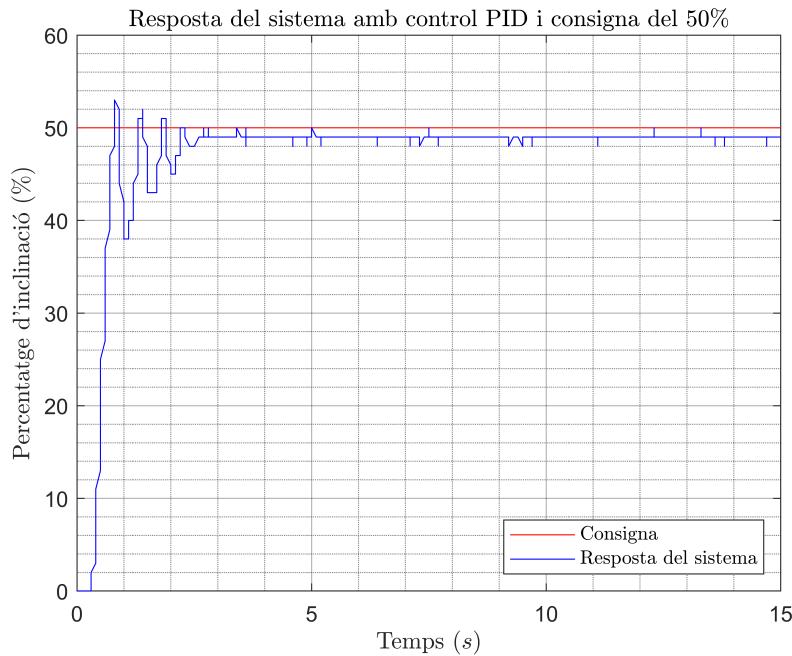


Figura 5: Gràfic de la resposta temporal del sistema amb un control PID i consigna d'inclinació del 50%.

A simple vista s'identifica l'operació correcta de l'acció integral degut a la tendència de la resposta a la consigna i sense cap mostra d'oscil·lació estacionària. Per la gràfica s'intueix que l'error estacionari serà inferior al cas anterior. Quantificant el seu valor, s'obté:

$$\varepsilon_{r, 50} = \frac{50 - 49}{50} \cdot 100 = 2 \%$$

Reiterant així el que s'ha comentat prèviament.

El temps d'estabilització és d'aproximadament 2,5 s, i durant l'estat transitori es visualitzen tres sobrepics: el primer supera la consigna per 3 unitats, el segon per 2 i el tercer per 1, essent els valors en percentatge vers la consigna, 6, 4 i 2%, respectivament. Observi's com el primer valor (el de major rellevància) és inferior al primer del cas anterior.

L'aplicació del control PID dissenyat amb el mètode de Ziegler i Nichols aporta unes prestacions correctes: baix error estacionari i sobrepic poc pronunciat, tot i que el temps d'estacionament és millorable. A l'onzena tasca s'intentarà millorar la resposta de forma heurística prenent com a referència els valors ideals.

4.2.3 Consigna d'inclinació: 75%

Finalment s'escull una consigna tal que la inclinació es situa a un 75%, de manera que la gràfica de la resposta del sistema es mostra a continuació.

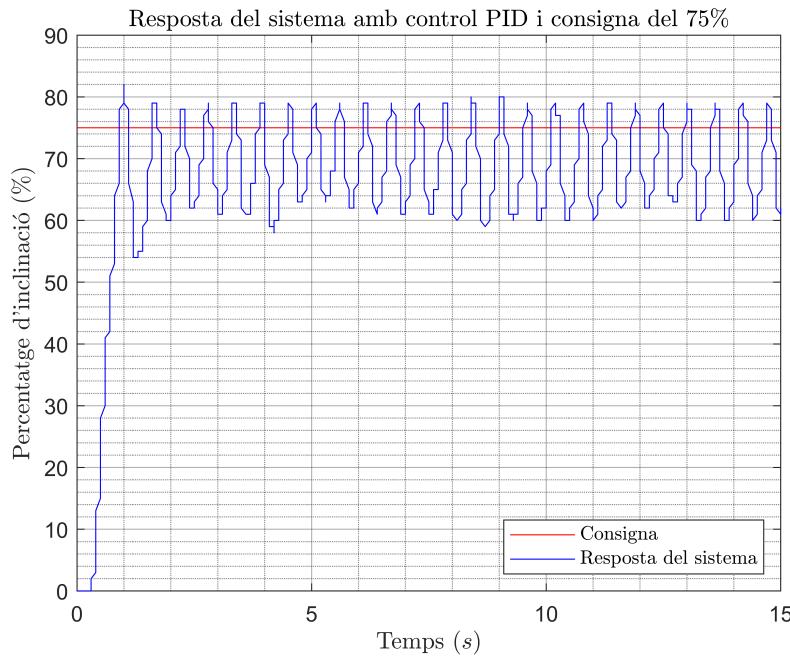


Figura 6: Gràfic de la resposta temporal del sistema amb un control PID i consigna d'inclinació del 75%.

Inicialment es produeix un petit sobrepic de 82 % d'inclinació, superant en un 7 % a la consigna. En aquest cas, donat que el braç es troba pròxim a l'angle d'equilibri inestable, corresponent a la posició vertical, és summament complicat obtenir una resposta coherent per part del sistema. És així que el sistema mostra una elevada presència d'oscil·lació amb una amplitud de 10 unitats.

El valor d'inclinació al voltant del qual es produeix l'oscil·lació és la mitjana entre el màxim i mínim absolut, considerant una oscil·lació constant:

$$\bar{R} = \frac{79 + 60}{2} = 69,5\%$$

L'error relatiu respecte de la consigna introduïda és:

$$\varepsilon_{r,75} = \frac{75 - 69,5}{75} \cdot 100 = 7,33\%$$

Obtenint així la desviació relativa més gran, i en conseqüència el major error estacionari.

Pel que fa al temps d'estabilització del sistema, no és un factor difícil de calcular donada la irregular i oscil·latòria progressió que segueix el braç, però es pot estimar entorn del 2 s.

Tenint en compte els factors exposats, es pot concloure que el disseny de controlador obtingut amb Ziegler-Nichols per a una inclinació del 50% no resulta efectiu per valors elevats de la consigna d'inclinació.

4.3 Onzena tasca

En el cas (probable) que el funcionament no sigui bo, modificar heurísticament els guanys. Obtenir una representació gràfica en funció del temps de la inclinació del braç amb tres consignes d'inclinació: 25%, 50%, 75%.

Tot i que els resultats obtingut mitjançant Ziegler-Nichols són notablement satisfactoris, s'ha optat per dur a terme heurísticament alguns petits ajustos de les constants de les accions de control. Per tant, els valors d'aquestes són tal que:

$$K_p = 2500$$

$$K_i = 100000$$

$$K_d = 35$$

Un increment tant notori de l'acció integral suposa un augment de rapidesa però també de sobrepic. De forma paral·lela, i per compensar aquest fet s'ha incrementat considerablement la constant derivativa, incrementant la robustesa del sistema i reduint l'amplitud dels possibles pics.

Tot seguit, es mostra la representació de la resposta del sistema.

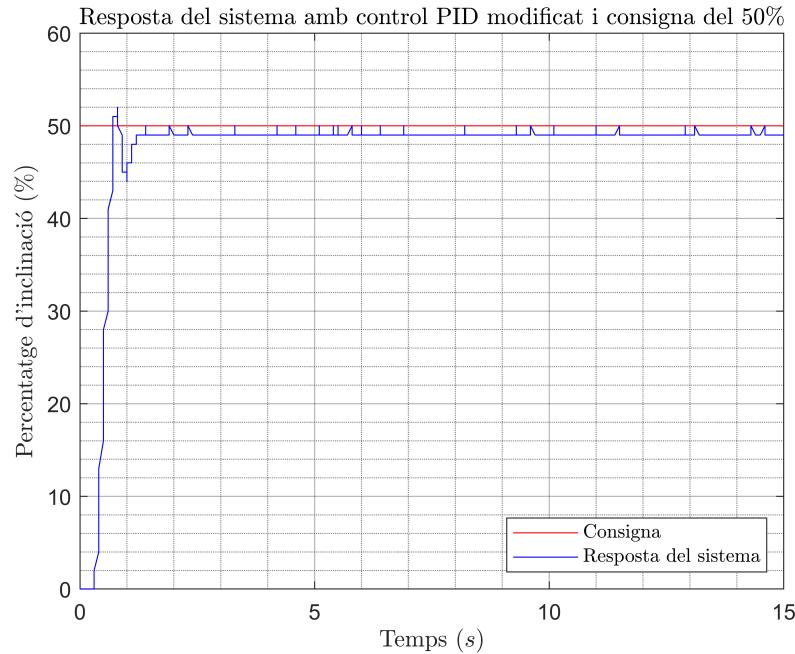


Figura 7: Gràfic de la resposta temporal del sistema amb un control PID modificat i consigna d'inclinació del 50%.

En primera instància, es produeix un únic sobrepic d'aproximadament 52 % d'inclinació per $t = 0,8\text{ s}$, superant així en un 2 % al valor de la consigna, i per a $t = 1,2\text{ s}$, aproximadament, el senyal ja s'ha estabilitzat. Aquest valor és un 60 % inferior a l'obtingut segons Ziegler-Nichols, assolint una millora important.

Pel que fa a la presència d'oscil·lacions, és evident que és relativament baixa tant en amplitud com en duració donada l'existència d'un esmoreïment considerable.

L'error present a la resposta estacionària respecte la consigna és del 2%, coincident amb l'obtingut mitjançant Z-N.

$$\varepsilon_{r, 50, \text{heur}} = \frac{50 - 49}{50} \cdot 100 = 2\%$$

La modificació heurística basada sobre els paràmetres òptims segons Ziegler-Nichols ha conduit a una millora substancial del temps d'establiment així com de l'amplitud de sobrepic. No obstant, l'error estacionari s'ha mantingut invariable.

5 Conclusions

5.1 Primera fase

La primera fase ha consistit en una introducció a les eines de programació que s'han aplicat a les fases posteriors. Els trets més representatius han estat els següents:

- Una funció ISR s'executa de forma periòdica és de no-retorn: no admet ni entrades ni sortides. En addició, per a freqüències molt elevades, aquesta funció haurà de ser el més senzilla possible, evitant la crida d'altres funcions al seu interior per optimitzar la velocitat d'execució i evitar un solapament amb una nova crida de la funció.
- Un funció *pwm* modula el *Duty* (temps d'estat encès), permetent accionar el rotor, en aquest cas. Com a detall, no totes les maquetes del laboratori reaccionen igual davant el mateix *Duty*. El sistema estudiat va requerir un valor considerablement pròxim al límit.

5.2 Segona fase

A la segona fase s'ha aprofundit sobre el disseny d'un programa que apliqui un control en llaç tancat amb realimentació negativa. Es destaquen els següents aspectes:

- La funció de control pot ser invocada com a funció d'interrupció, assolint la revisió i correcció de l'estat actual de forma gairebé constant.
- Un control proporcional, si bé pot arribar a proporcionar una resposta acceptable, és insuficient per garantir un error estacionari totalment nul.

5.3 Tercera fase

La tercera fase gira entorn a l'aplicació del controlador PID. Les conclusions a les que s'ha arribat són les següents:

- En cas de no disposar de biblioteques que permetin realitzar operacions matemàtiques d'alt nivell com derivades i integrals numèriques, és possible implementar una operació discretitzada que s'aproxima més a la teoria quant menor és el període T .
- La inclusió d'una component integral, sempre i quan configurada d'acord a un estat específic del sistema, permet assolir un error estacionari que tendeix a 0, així com una oscil·lació estacionària nul·la.
- El disseny de controlador segons Ziegler-Nichols ofereix uns resultats acceptables per a un estat del sistema determinat. Per a una consigna diferent i de menor valor, la dinàmica del sistema possibilita assolir una resposta acceptable. En canvi, per a una consigna de major valor, el sistema es troba proper a un estat d'equilibri inestable i el controlador deixa d'oferir un resultat vàlid.
- El disseny de controlador de Ziegler-Nichols ofereix uns paràmetres base a partir dels quals, aplicant modificacions heurístiques, és possible assolir un comportament més satisfactori, en cas que es desitgi una millora.

5.4 Possibles millores

Un cop es van realitzar les sessions experimentals, mentre es duia a terme l'estudi de resultats es va observar com algunes xifres, com el percentatge d'inclinació, havien estat enregistrades com a enters. Altres, com el comptador de temps t , havia vist reduït el seu nombre de xifres significatives de 4 a 2. És possible que incrementant la precisió del programa en MatLab que llegia les dades del monitor sèrie es solucionés aquest problema, possibilitant així un estudi amb més profunditat en l'àmbit quantitatiu.

S'ha observat també com, en cap dels casos estudiats s'ha assolit un error estacionari totalment nul. La raó pot ésser deguda a la precisió oferida del potenciómetre, la qual presenta uns límits superior i inferior no totalment fixos, així com una apreciació en la mesura relativament baixa. No obstant, per assolir aquesta millora caldria modificar físicament el sistema, acció no necessària per a realitzar un estudi eminentment qualitatiu.

Referències

- [1] Volatile [Variable Scope & Qualifiers]. Arduino Reference. Consultat el 13 de novembre de 2019.
<https://www.arduino.cc/reference/tr/language/variables/variable-scope--qualifiers/volatile/>
- [2] TimerOne & TimerThree Libraries. PJRC. Consultat el 7 de desembre de 2019.
https://www.pjrc.com/teensy/td_libs_TimerOne.html