UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

**Grau en Enginyeria Electrònica Industrial i Automàtica**

# CONTROL D'UNA PLATAFORMA *BALL IN TUBE* MITJANÇANT UN DISPOSITIU MÒBIL

## Memòria i Annexos


**Autor:** Guillem Ruiz Solé
**Director:** Sebastián Tornil Sin

**Convocatòria:** Octubre 2018


UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est
UPC

# ÍNDEX

# Resum

En aquest treball de fi de grau, emprant els coneixements adquirits a l'especialitat d'enginyeria electrònica industrial i automàtica, es proposa realitzar un control d'una plataforma *Ball in Tube* mitjançant un dispositiu mòbil, a través d'una app creada per a realitzar aquesta funció d'una manera senzilla i intuïtiva.

Aquest control de la plataforma s'establirà en base a dos tipus de controls treballats en assignatures del grau, i dels quals es tenen suficients coneixements com per treballar amb ells.

L'app creada permetrà la manipulació de tots els paràmetres que puguin afectar al control de la plataforma sense suposar una gran dificultat per a l'usuari d'aquesta app.

# Resumen

En este trabajo de fin de grado, usando los conocimientos adquiridos en la especialidad de ingeniería electrónica industrial y automática, se propone realizar un control de una plataforma *Ball in Tube* mediante un dispositivo móvil, a través de una app creada para realizar dicha función de una forma sencilla e intuitiva.

Este control de la plataforma se establecerá en base a dos tipos de control trabajados en asignaturas del grado, y de los cuales se tienen suficientes conocimientos como para trabajar con ellos.

La app creada permitirá la manipulación de todos los parámetros que puedan afectar al control de la plataforma sin suponer una gran dificultad para el usuario de esta app.

# Abstract

In this final degree Project, using the knowledge gained in the industry and automatic electronic engineering, it's proposed to control a *Ball in Tube* platform through a mobile device, through an app created to do this function ina simple and intuitive way.

This platform control will be established based on two types of control worked on subjects of the degree, which we have enough knowledge to work with.

The created app will allow the manipulation of all the parameters which could affect the platform control without being a big deal to the app user.

# Agraïments

En primer lloc, m'agradaria començar donant les gràcies a la meva família que m'ha donat tot el seu suport i ajuda, no només durant la realització d'aquest projecte, sinó també durant tota la carrera, ja fossin moments alegres o moments difícils. També m'agradaria agrair a tots els meus amics el suport que m'han ofert i els moments de desconnexió necessaris quan sorgien dificultats.

Per altra banda, vull expressar la meva gratitud al professor i tutor d'aquest projecte Sebastián Tornil Sin degut a tot el suport, resolució de dubtes i bons consells que m'ha anat donant al llarg de l'evolució d'aquest projecte; oferint-me sempre el temps necessari per reunir-nos i resoldre totes les dificultats que han anat apareixent al llarg d'aquest camí.

Per últim, m'agradaria agrair als companys de feina on he realitzat les meves pràctiques externes, Lubrizol Advanced Materials Sant Cugat; els quals en tot moment s'han preocupat per l'evolució d'aquest projecte i m'han donat el seu suport per seguir endavant en la realització d'aquest.

# 1. Introducció

## 1.1. Motivació

Actualment, la tecnologia evoluciona cap al món de l'automatització, permetent així que molts aspectes de la indústria, el comerç i, fins i tot, de la vida quotidiana siguin controlats de forma remota, ja sigui a través d'ordinadors o des de dispositius mòbils. Aquest canvi que es produeix a dia d'avui a resultat ser la motivació d'aquest treball, és a dir, el fet de facilitar la manipulació de certs objectes o màquines o d'alterar algunes de les seves característiques.

## 1.2. Objectius

En aquest treball apareix la necessitat de controlar el funcionament d'una plataforma *Ball in Tube* de forma remota. La plataforma *Ball in Tube* és un sistema o estructura format per una base amb un tub de metacrilat just a sobre, a dins del qual hi ha una pilota de ping-pong; aquest sistema també compta amb una placa electrònica que s'encarrega de posar en marxa els actuadors i de controlar tot el funcionament de la plataforma. A la base del tub s'hi troba un ventilador que s'encarrega d'elevar la pilota dins del tub, el qual està controlat per la placa, i a la part superior del tub hi ha un sensor de ultrasònic que s'encarrega de prendre lectures de la distància que hi ha entre el propi sensor i la pilota i d'enviar aquests valors a la placa. El funcionament d'aquesta plataforma consisteix en mantenir suspesa la pilota en una posició concreta dins del tub d'aire.

L'objectiu principal d'aquest treball consisteix en poder realitzar aquest control, amb els seus respectius canvis de posició i altres paràmetres, de forma remota a través d'un dispositiu mòbil; evitant, així, la necessitat d'haver-se de connectar a la placa que controla tot el sistema cada vegada que es desitja realitzar un canvi.

Tenint en compte que la base d'aquest treball és la realització del control de la plataforma mitjançant un dispositiu mòbil, caldrà demostrar el correcte funcionament a través de la implementació de diferents tipus de controls, i així també poder veure com influencia el tipus de control en el comportament de la plataforma. Per tant, s'optarà per la implementació de dos controls simples. Aquest control també ha de permetre la realització de canvis o millores en els seus paràmetres per tal d'així poder variar el funcionament del control segons les característiques que l'usuari desitgi tenir en l'ús de la plataforma.

Fent referència al control remot de la plataforma, cal marcar com a objectiu l'ús de la millor comunicació que permeti un control remot per a aquesta aplicació; tenint en compte les plataformes de desenvolupament i la compatibilitat entre elles, a més a més caldrà tenir present l'eficiència del mètode de comunicació entre l'aplicació i la placa que controla la plataforma *Ball in Tube*.

En resum, els objectius que pretén afrontar aquest treball són els següents:

- Implementar un control remot a la plataforma *Ball in Tube*.
- Implementació de dos controls simples.
- Emprar el mètode de comunicació més eficient entre l'aplicació i la plataforma.

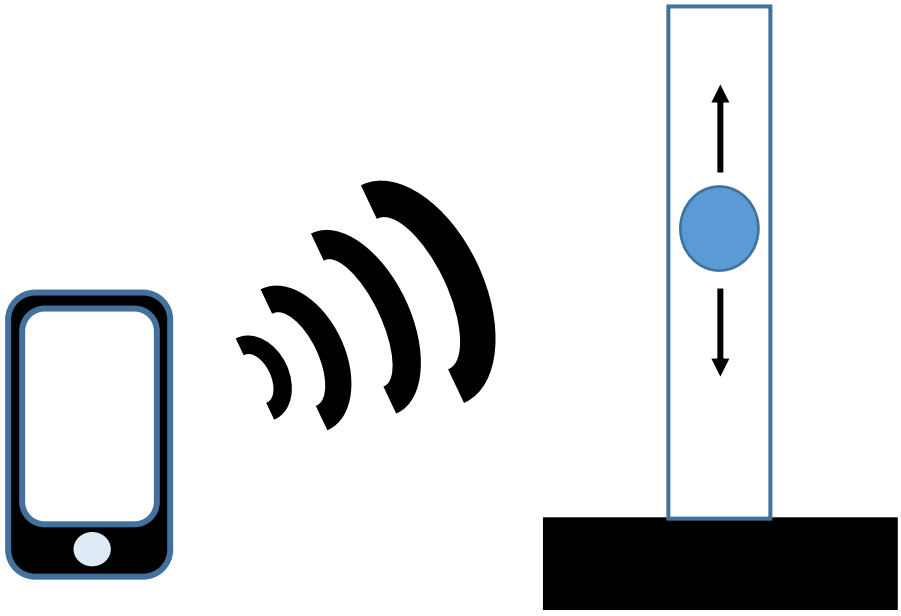


*Figura 1. Esquema representatiu de l'objectiu del projecte*

# 2. La plataforma *Ball in Tube*

La plataforma *Ball in Tube* és una estructura que, controlada per una placa electrònica, s'encarrega de mantenir una pilota de ping-pong suspesa a l'aire a una alçada concreta a partir de les mesures de distància que pren un sensor ubicat a la part superior. Aquesta plataforma està formada pels següents components:

- Una base, on es recolza tota l'estructura i tots els components que la formen.
- Un tub de metacrilat, que s'aixeca per sobre de la base i a dins del qual es realitza tot el procés que caracteritza aquesta plataforma.
- Un ventilador que s'encarrega de proporcionar l'aire necessari per mantenir la pilota suspesa.
- Un sensor d'ultrasons que s'encarregarà de prendre el valor de la distància que hi ha entre la pilota i la part superior del tub.
- Una placa Arduino UNO encarregada de realitzar tot el control de la plataforma.
- Una pilota de ping-pong, la qual és l'objecte de tot el control.

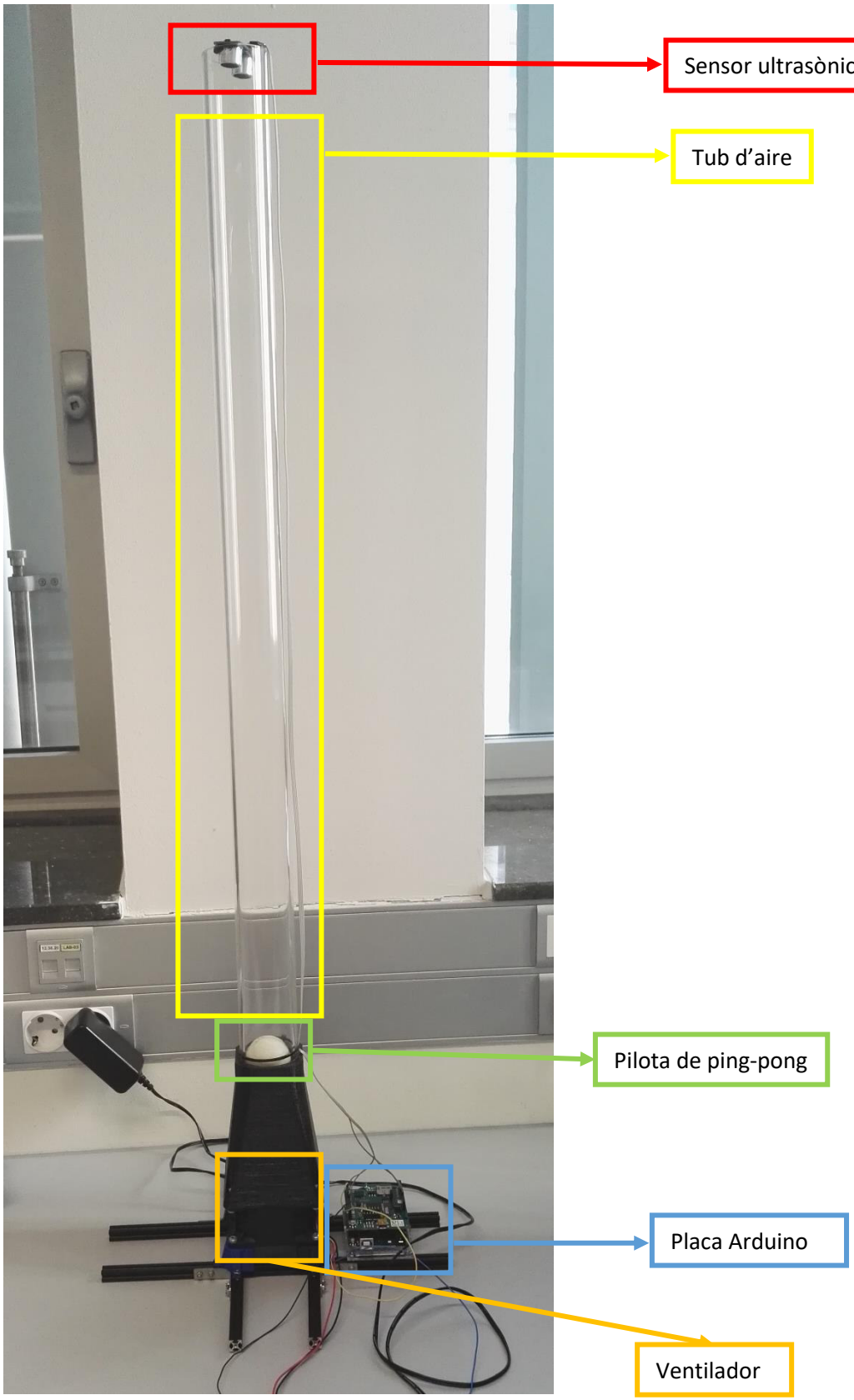



*Figura 2. Plataforma Ball in Tube i les seves parts*

Tal com s'ha esmentat anteriorment, aquesta plataforma pretén mantenir la pilota suspesa dins del tub d'aire a una alçada concreta a partir de les mesures de distància que va prenent el sensor ultrasònic; però també caldrà tenir en compte que l'efectivitat d'aquesta plataforma vindrà determinada pel tipus de control que s'apliqui, és a dir, segons el control que es demani a la placa Arduino que porti a terme. Per tant, caldrà pressuposar que el sistema no serà estable en un principi, i que la dificultat es trobarà en poder realitzar un control on la pilota oscil·li quantes menys vegades millor i a una distància el més petita possible respecte l'alçada que s'estableixi (o *Setpoint*).

## 2.1. La placa Arduino UNO

La placa Arduino és un dels components més importants de tota l'estructura, ja que s'encarrega de tot el control del funcionament de la plataforma. A aquesta placa hi van connectats a tots els components electrònics i mecànics, com el sensor i el ventilador.

La placa Arduino és una placa de circuit imprès simple basada en un microcontrolador de font oberta, el qual permet fer més simple i accessible el disseny de circuits electrònics i microcontroladors. Arduino es pot utilitzar per desenvolupar objectes interactius autònoms o pot ser connectat a programari de l'ordinador.

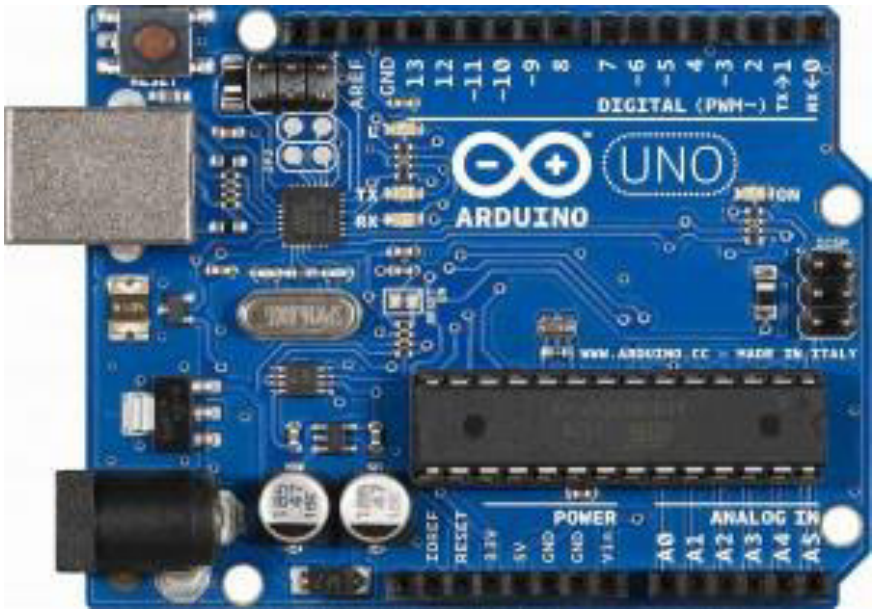



*Figura 3. Placa Arduino UNO*

Existeixen molts models de placa Arduino, en funció de les característiques i les funcions que poden arribar a tenir. En aquest cas s'ha utilitzat el model Arduino UNO, el qual correspon al model més bàsic i amb el que és més fàcil de treballar. Aquest model només compta amb les característiques bàsiques d'una placa Arduino, facilitant així la seva programació i les connexions dels components que s'hi fan.

La placa consta de 14 entrades digitals configurables, ja sigui com a entrada o com a sortida, que operen a 5 V. Cada pin pot proporcionar o rebre com a màxim 40 mA. Els pins 3, 5, 6, 9, 10 i 11 poden proporcionar una sortida PWM (modulació per amplada de polsos). També compta amb 6 entrades analògiques que proporcionen una resolució de 10 bits i que per defecte mesuren de 0 a 5 V.

L'entorn de programació que utilitza la placa Arduino utilitza és un entorn de desenvolupament anomenat IDE Arduino, entorn de desenvolupament integrat ( sigles en anglès de *Integrated Development Environment*). És un programa informàtic format per un conjunt d'eines de programació.

L'IDE d'Arduino és un entorn de programació que consisteix en un editor de codi, un compilador, un depurador (*debugger*) i un constructor d'interfície gràfica (GUI). A més a més , incorpora les eines per carregar el programa ja compilat a la memòria flash del *hardware*. El llenguatge que s'empra per programar la placa és molt semblant a C++, anomenat Processing de Wiring; és un llenguatge de nivell mig, que tracta amb objectes bàsics caràcters, números, bits i direccions de memòria, entre d'altres.

A l'inici del codi del programa que es vulgui redactar, és necessari declarar totes les variables i de quin tipus són (*integer, float, double,* etc.); i també es poden inicialitzar aquestes variables, però no és obligatori fer-ho en aquest punt del codi.

```
int SETPOINT = 40;
int HISTERESIS=5;
float DISTANCIA;
float DURACION;
int TRIG=10;
int ECO=9;
int VENTILADOR=3;
int PWM;
int Input;
```

*Figura 4. Captura de codi encarregat de declarar variables*

A l'hora de redactar el codi per al programa, l'IDE d'Arduino es caracteritza per tenir dues funcions bàsiques:

- *Setup:* aquesta funció només s'executa una vegada, just després de carregar el programa a la placa. S'utilitza principalment per inicialitzar variables i per definir si els components connectats a la placa corresponen a entrades o sortides.

- *Loop:* aquesta funció s'executa de forma iterada, és a dir, quan la placa executa l'última acció o línia del codi, automàticament la placa torna a l'inici d'aquesta funció i torna a executar-la. S'utilitza principalment per dur a terme les accions que s'han de repetir constantment en bucle, com la lectura d'un sensor o l'activació d'un actuador.

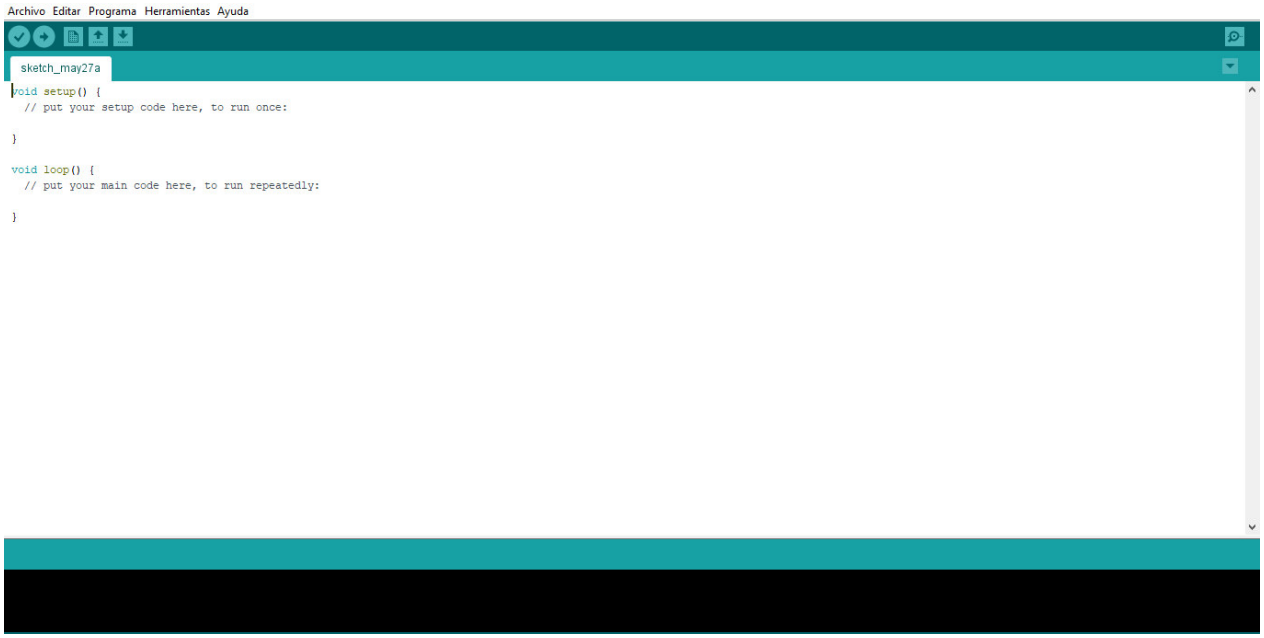



*Figura 5. Pantalla del IDE d'Arduino*

Fent referència a la definició dels components connectats a la placa, cal fer conèixer a la placa si aquests components correspondran a entrades o sortides de la placa, és a dir, si d'aquests components obtindrem valors o si n'hi haurem de proporcionar. Per fer aquesta definició, la qual és realitza dins de la funció de *Setup*, cal donar a conèixer a quin pin de la placa està connectat el component i si serà una entrada (*INPUT*) o una sortida (*OUTPUT*). També és pot declarar una variable amb el nom del component i donar-li el valor del pin al que està connectat, tal i com es veu a la següent imatge.

```
pinMode(TRIG, OUTPUT);
pinMode(ECO, INPUT);
pinMode(VENTILADOR, OUTPUT);
```

*Figura 6. Codi de definició de variables com a entrades o sortides.*

L'IDE d'Arduino compta amb dues eines molt útils que permeten veure els valors de les variables en temps real. Aquestes eines són el *Monitor serie* i el *Serial plotter*. La primera permet veure el valor numèric de la variable a cada iteració de la funció *loop*. La segona permet veure una gràfica contínua del valor de la variable i com aquest va canviant al llarg del temps.

Per poder emprar aquestes dues eines cal activar la comunicació sèrie entre la placa i els components que estan connectats. Per fer això, s'ha d'incloure la línia de codi *Serial.begin()* a la funció *Setup* del codi.

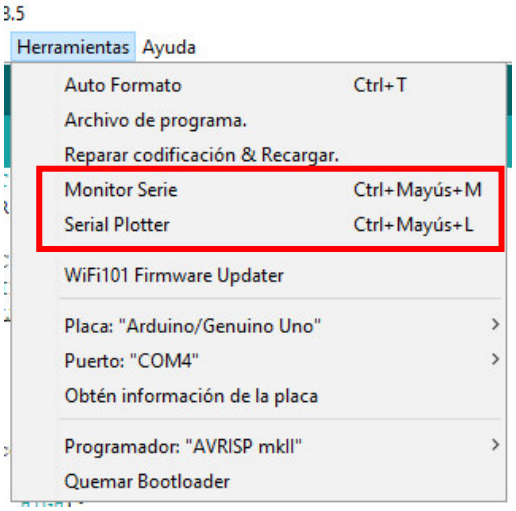



*Figura 7. Ubicació Monitor Serie i Serial Plotter.*

## 2.2. El ventilador QFR0812DE-F00

El ventilador és el component encarregat de proporcionar la quantitat d'aire necessària per fer pujar la pilota de ping-pong fins al l'alçada establerta en el control i també de mantenir-la en aquell punt. Per poder dur a terme aquesta acció, el control s'encarregarà d'anar canviant la velocitat de gir del ventilador per poder així fer que la pilota pugi o baixi segons si està per sobre o per sota del *Setpoint*. Aquest funcionament s'explicarà amb més detall a l'apartat sobre el control.

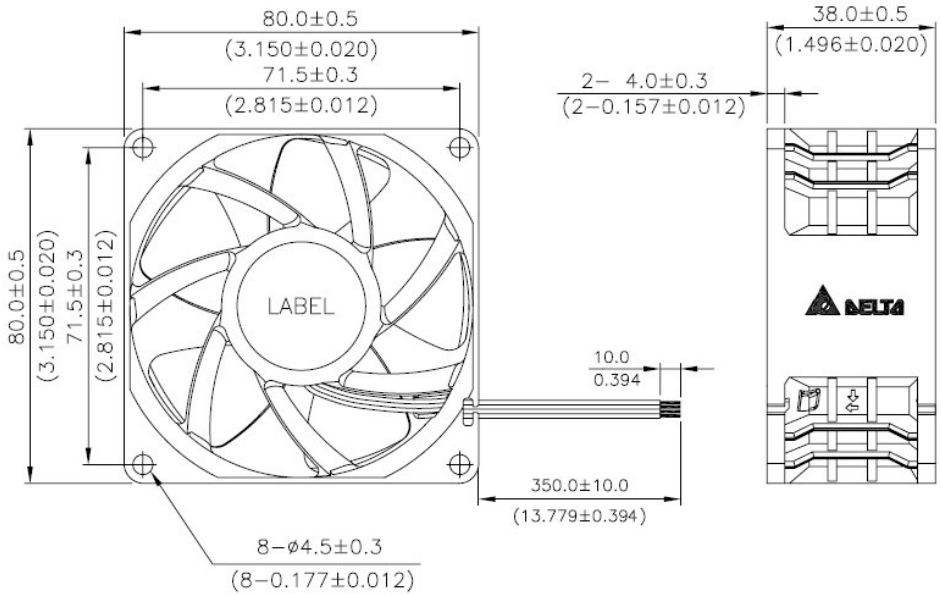



*Figura 8. Esquema del ventilador QFR0812DE-F00.*

El model de ventilador emprat és el QFR0812DE-F00, el qual s'alimenta a 12 V i té una velocitat màxima de 9000 rpm.

| ITEM | DESCRIPTION |
|---|---|
| RATED VOLTAGE | 12 VDC |
| OPERATION VOLTAGE | 10.8 – 13.2 VDC |
| INPUT CURRENT | 1.15 ( 1.70 MAX.) A |
| INPUT POWER | 13.8 ( 20.40 MAX. ) W |
| SPEED | 9000R.P.M. (REF.) |
| MAX. AIR FLOW (AT ZERO STATIC PRESSURE) | 2.991 ( MIN. 2.692) $M^3/MIN$<br>105.21 ( MIN. 94.69) CFM |
| MAX.AIR PRESSURE (AT ZERO AIR FLOW) | 33.91 ( MIN. 27.47 )$mmH_2O$<br>1.335 ( MIN. 1.081 )$inchH_2O$ |
| ACOUSTICAL NOISE (AVG.) | 60.0 (MAX 64.0 ) dB-A |
| INSULATION TYPE | UL: CLASS A |

*Figura 9. Taula de característiques ventilador QFR0812DE-F00.*

Aquest ventilador té 4 connexions diferents, cada una de les quals correspon a:

- Cable vermell: correspon al pol positiu de l'alimentació
- Cable negre: correspon al pol negatiu de l'alimentació.
- Cable groc: correspon a l'entrada que alimenta al ventilador a través del mètode de modulació per amplada de polsos (PWM).
- Cable blau: correspon al sensor de velocitat del ventilador (F00).

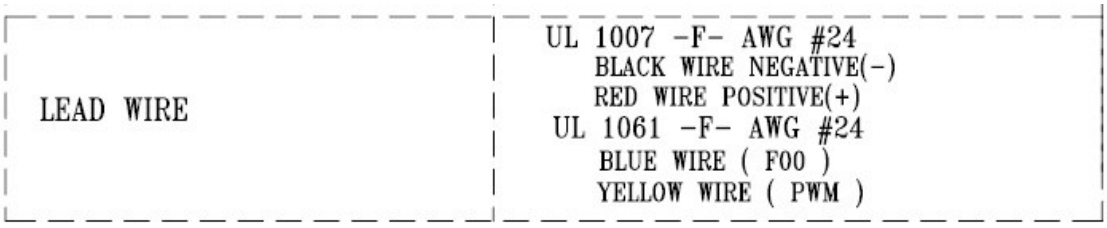



*Figura 10. Taula sobre color dels cables del ventilador QFR0812DE-F00.*

### 2.2.1. Modulació per amplada de polsos

La modulació per amplada de polsos (PWM) consisteix en la utilització d'una ona de polsos rectangulars, l'amplada dels quals és modulada amb el senyal d'entrada. El valor de la sortida vindrà directament relacionat amb el cicle de funcionament (*Duty Cycle*) del senyal d'entrada.

El seu principal ús és la de controlar quantitats d'energia que s'envia a una càrrega o la de transmetre informació a través d'un canal de comunicacions.

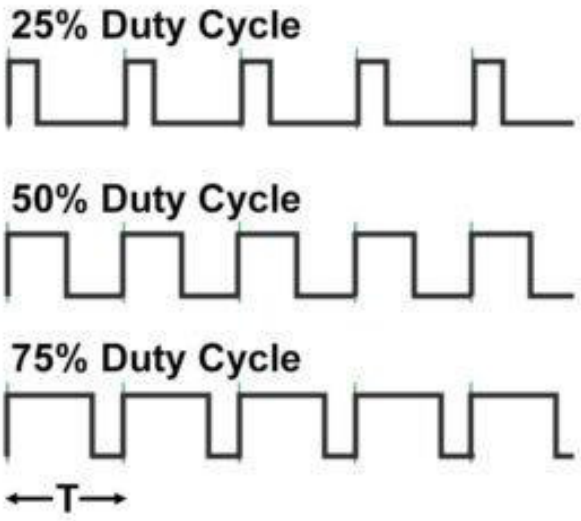



*Figura 11. Funcionament modulació amplada de polsos (PWM).*

Seguint l'explicació sobre el funcionament de la modulació per amplada de polsos, el ventilador rebrà una alimentació de 12 V com a màxim, quan treballi al 100% del seu cicle de treball. Per trobar la velocitat de gir adequada perquè la pilota pugi i es mantingui al *Setpoint*, caldrà definir el cicle de treball més apropiat; la qual cosa es farà a l'apartat de control.

A l'hora d'aplicar aquest funcionament a la placa Arduino, cal saber que la placa compta amb una sortida digital PWM, és a dir, que la placa pot alimentar un component a través de la modulació per amplada de polsos. Aquesta sortida PWM de la placa té unes característiques molt concretes. Al tractar-se d'una sortida digital té un rang de valors entre 0 i 255, els quals corresponen al 0 i al 100% del Duty Cycle del component que s'hi connecti. Per alimentar el ventilador, el qual estarà connectat a la sortida PWM ja que es treballa amb el mode de funcionament PWM que conté aquest model, caldrà utilitzar la seqüència *analogWrite*. Aquesta línia de codi s'utilitza per alimentar components a través dels pins digitals PWM. En aquesta seqüència s'ha d'incloure el pin on està connectat el component i el valor de la PWM en el rang de 0 a 255. El valor de la PWM ha de ser sempre un nombre enter ja que aquesta seqüència només és capaç de llegir nombres enters.

```
analogWrite(VENTILADOR, 87);
```

*Figura 12. Codi per donar tensió al ventilador mitjançant un valor de PWM.*

### 2.2.2. Anàlisi del funcionament del ventilador

Degut a que el model del ventilador estava predeterminat des de l'inici del projecte a causa de que l'estructura ja estava muntada, s'ha optat per analitzar el funcionament del ventilador per així poder determinar si l'aparició de problemes o dades incoherents es poden deure al funcionament d'aquest model concret de ventilador.

Per realitzar aquest anàlisi del funcionament es realitza un programa per a l'Arduino per poder veure el temps de reacció que té el ventilador al canviar de velocitat de gir, és a dir, quants segons pot trigar en estabilitzar-se a una velocitat i després a una altra. El canvi de velocitat de gir es realitzarà mitjançant el canvi de PWM, i així es podrà veure com canvia el detector de rotació (esmentat al datasheet com a *rotation detect*).

Per observar aquest canvi en el detector de rotació, s'ha creat un programa d'Arduino que va alternant el valor de la PWM subministrada al ventilador entre 0 i 95 per veure com canvia la rotació en el moment de l'arrancada i la parada del ventilador i mantenint cada un d'aquests valors durant un cert temps per veure l'estabilitat que ofereix la rotació del ventilador.

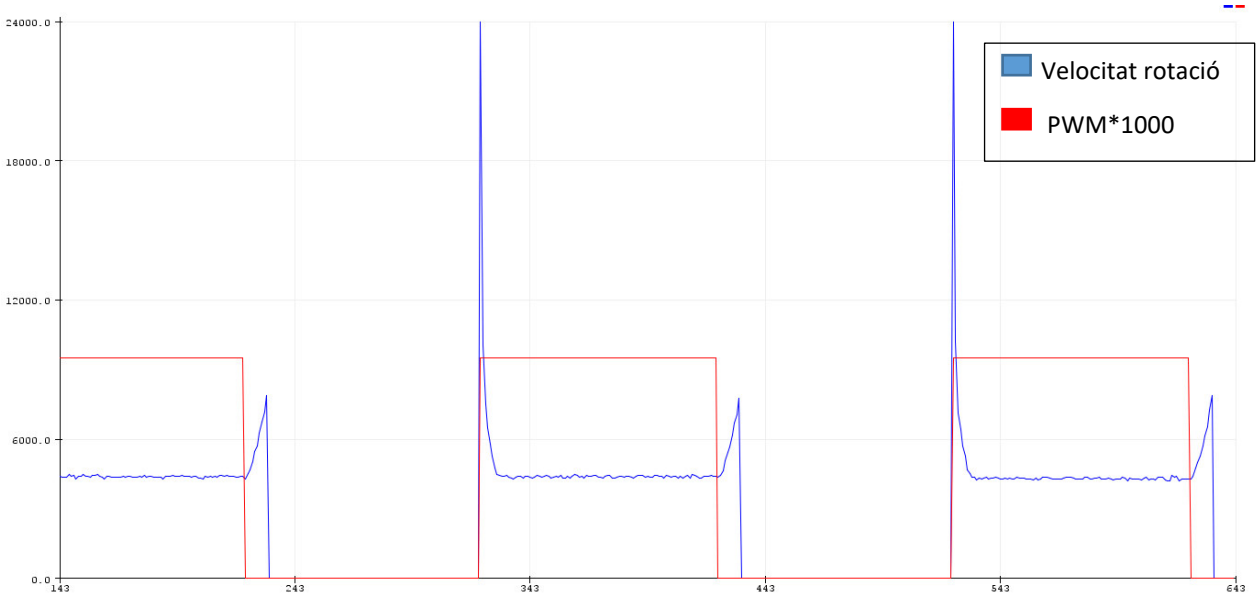



*Figura 13. Resultats proves funcionament del ventilador.*

A la Figura 13 es pot veure la rotació del ventilador en funció de la PWM que se li dóna al ventilador per posar-lo en marxa. Els valors de PWM que apareixen a la gràfica s'han multiplicat per 1000 per així poder veure ambdós paràmetres (PWM i rotació) en un rang similar.

La velocitat màxima de rotació que pot assolir el ventilador és 9000 rpm quan al ventilador se li dóna una PWM de 255 (valor màxim de PWM), però en aquest assaig la PWM tenia un valor de 95 per evitar que la pilota pugés amb massa força. Degut a aquest valor de PWM el valor de la rotació quan el ventilador està en marxa és inferior a les 4000 rpm.

Parlant del funcionament durant aquest assaig, es pot veure com en el moment en que el ventilador es posa en marxa, la velocitat de rotació adquireix un valor molt elevat i després s'estabilitza al valor que li pertoca. Mentre el ventilador segueix en marxa, l'estabilitat de la velocitat de rotació és molt bona ja que en cap moment es desestabilitza ni mostra cap pic. En el moment en que el ventilador para, la velocitat de rotació mostra un petit pic, un petit augment en el valor d'aquesta, per després aturar-se totalment.

D'aquest assaig es pot assumir que el funcionament del ventilador mostra una bona estabilitat mentre es mantingui en un valor constant. En els moments de

canvi de velocitat del ventilador, presenta certs pics que poden suposar una petita desestabilització o un cert retard a l'hora de modificar el seu funcionament. En cas que aquests fets es donin durant els assajos del projecte, se sabrà que el motiu és el funcionament del ventilador.

## 2.3. El sensor ultrasònic HC-SR04

El sensor ultrasònic serà el component encarregat de mesurar la distància que hi ha entre el sensor, que es troba a la part superior del tub d'aire, i la pilota, la qual s'anirà movent en funció de la velocitat de gir del ventilador.


*Figura 14. Sensor ultrasònic HC-SR04.*

El model escollit de sensor és el HC-SR04 el qual utilitza un mètode semblant a un sonar per detectar objectes. Aquest sensor està alimentat a 5 V i pot detectar objectes que es troben entre 2 i 400 centímetres i té una resolució de 0,3 centímetres, la qual permet obtenir una precisió bastant alta de la distància de l'objecte.

**Features:**

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

*Figura 15. Característiques del sensor HC-SR04.*

El sensor conté 4 pins de connexions, amb les següents equivalències:

- Pin Vcc: pin corresponent a l'alimentació (5 V).
- Pin GND: pin corresponent a la connexió a massa.
- Pin Trig: pin corresponent a l'entrada de *Trigger*.
- Pin Echo: pin corresponent a la sortida Echo del sensor.

El funcionament d'aquest sensor és el següent: quan l'entrada de *Trigger* rep un pols 10 μS, comença a enviar ultrasons i quan aquestes ones topen amb un objecte, reboten i són rebudes pel sensor un altre cop. Aquest procés fa que la sortida Echo doni un voltatge de 5 V i que esperi a un retràs (*delay*) el qual donarà el valor de la distància que hi ha fins a l'objecte. El valor real que mesura el sensor és aquest retràs entre l'ona que s'envia i la que es rep, per tant, fa una mesura de temps; però, a través d'un factor de conversió, permet calcular la distància.

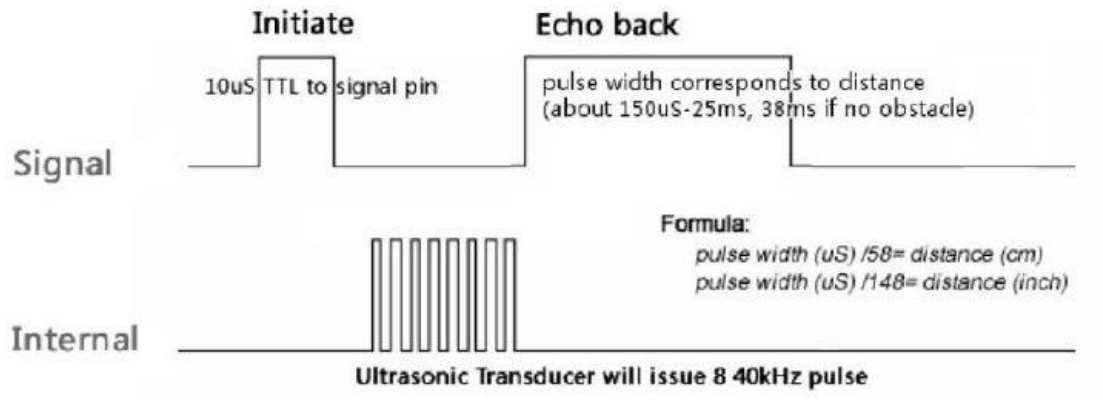



*Figura 16. Funcionament sensor HC-SR04.*

Tal i com s'aprecia a la figura anterior, l'amplada del pols del senyal *Echo* correspon al temps que mesura el sensor entre que envia el senyal ultrasònic i el rep; per tant, la distància obtinguda serà proporcional a la duració del pols.

Per poder dur a terme aquest funcionament del sensor a través de la placa Arduino caldrà seguir uns passos concrets. Primer caldrà definir els pins Trig i Echo com a sortida i entrada respectivament. Tal com s'ha esmentat anteriorment, el sensor comença a funcionar quan el pin Trig rep un pols de 10 μS; acció que es realitza a través de la seqüència de codi *digitalWrite*. Aquesta seqüència serveix per donar valor alt o baix (1 o 0) a un component connectat a la placa. Per fer que el sensor rebi una entrada de polsos cada 10 μS, s'utilitzarà la línia de codi *digitalWrite* alternant els valors *High* i *Low* per donar els 1 i 0 al sensor. Entre cada línia d'aquestes esmentades s'introduirà un *delay* de 10 μS per així poder fer l'entrada de polsos.

Un cop el sensor es posa en marxa, emet les ones que han de topar amb l'objecte. La lectura d'aquestes ones que reboten en l'objecte es fa a través del pin d'entrada Echo. Per fer aquesta lectura s'utilitza la seqüència *pulseIn*, a la qual se li assigna la variable *DURACION*. Això és degut a que el sensor llegeix quan rep un valor alt a l'entrada Echo i dóna un valor de temps que equival al temps de duració entre el moment en que s'emet l'ona i el moment en que es rep. Degut a que el valor que es vol obtenir es una distància en centímetres, el

fabricant del sensor estableix que per fer la transformació de microsegons a centímetres cal fer una divisió de la duració entre 58.

L'ús del número 58 per poder passar la duració del pols a la distància a la que es troba l'objecte, la pilota de ping-pong en aquest cas, es deu a la relació que hi ha entre la distància i el temps a través de la velocitat. Tenint en compte que la relació entre la distància i el temps és la següent:

$$velocitat = \frac{distància}{temps} \rightarrow v = \frac{d}{t} \quad \text{(Eq. 1)}$$

Es pot determinar la fórmula resultant que permetrà calcular la distància:

$$d = v * t \quad \text{(Eq. 2)}$$

Tal com es comenta al *datasheet* del sensor, es dóna per suposat que la velocitat del so és de 340 m/s. A partir de la relació entre distància i temps i el valor de la velocitat del so que proporciona el *datsheet*, aquest mateix document estableix que la relació entre la distància i el temps és:

$$d\ [en\ cm] = \frac{t\ [en\ \mu s]}{58} \quad \text{(Eq. 3)}$$

```
digitalWrite(TRIG, LOW);
delay(0.01);
digitalWrite(TRIG, HIGH);
delay(0.01);
digitalWrite(TRIG, LOW);
DURACION = pulseIn(ECO, HIGH);
distancia = DURACION / 58;
```

*Figura 17. Codi de posta en marxa i càlcul de temps i distància del sensor HC-SR04.*

### 2.3.1. Anàlisi del funcionament del sensor

Degut a que el model del sensor ja estava predeterminat des del principi de la realització del projecte, s'ha decidit dur a terme una comprovació o verificació del funcionament d'aquest sensor. Cal tenir en compte que aquest anàlisi del funcionament del sensor es realitza a l'entorn de funcionament d'aquest projecte, és a dir, a la plataforma *Ball in Tube*; per tant, el resultat d'aquest anàlisi només

és vàlid per a les condicions d'aquest projecte o algun altre que sigui semblant. El que es pretén veure a través d'aquest anàlisi del funcionament del sensor és la seva capacitat de captar dades correctes i quin error a l'hora de prendre mesures pot crear, podent així preveure o entendre de millor manera els resultats que s'obtindran durant la realització del projecte.

Per realitzar aquest anàlisi, s'ha optat per prendre mesures de la pilota quan aquesta es troba a la part inferior del tub sense moure's, és a dir sense que es posi en marxa el ventilador, durant un minut.

A continuació es mostren les dades obtingudes en forma de gràfica.

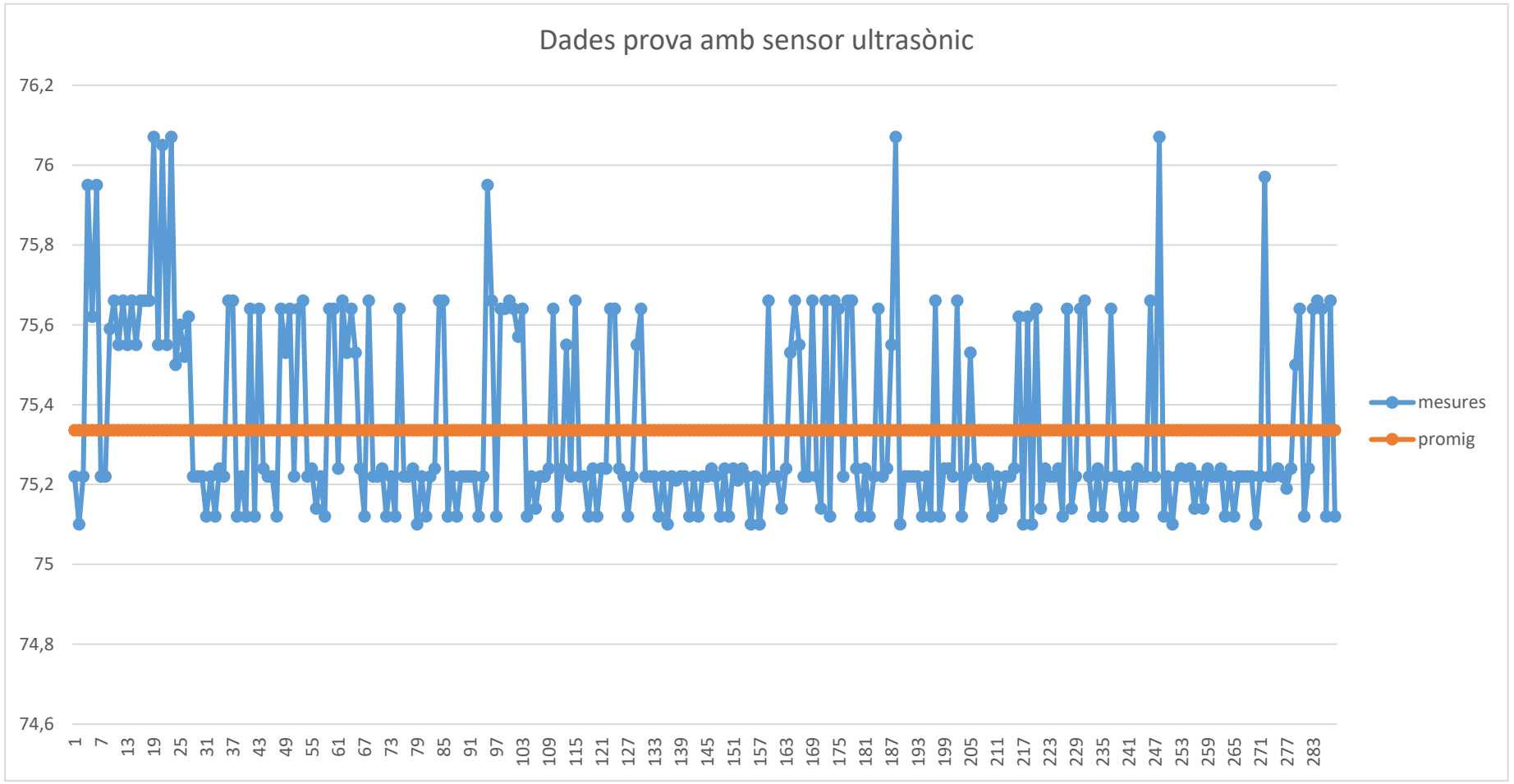



*Figura 18. Resultats anàlisi funcionament sensor HC-SR04.*

Com es pot veure a la gràfica anterior, i a les dades que es troben a l'Annex, el funcionament del sensor és bastant precís ja que els errors a les mesures són bastant petits, els quals es troben entre 0,3 i -0,8 respecte al promig obtingut de totes les mesures. El promig obtingut de totes les mesures és de 75,34 cm; i comparant-ho amb la distància real entre el sensor i la pilota, la qual és de 75,5 cm, es pot dir que el promig de les mesures realitzades és molt proper al valor real d'aquesta distància. Amb aquestes dades obtingudes es pot determinar que el funcionament del sensor no serà la causa d'errors o possibles mals funcionaments de la plataforma ja que amb aquesta prova s'ha determinat que les lectures del sensor són correctes.

Caldria afegir que s'ha realitzat una altra prova on el sensor prengués mesures quan la pilota es troba a dalt de tot del tub, és a dir, enganxada pràcticament al sensor. Respecte a aquesta prova cal dir que cap de les lectures és correcta i això es deu a que al estar la pilota tan a prop del sensor, les ones ultrasòniques que envia el sensor no reboten contra la pilota sinó que passen pel seu costat i arriben fins a la part inferior del tub.

Tenint en compte aquestes dues proves realitzades i les proves aplicant els diferents tipus de control, els quals s'explicaran al proper apartat; es pot suposar que el sensor funcionarà correctament i prendrà mesures bastant properes al valor real quan la pilota es trobi a la zona intermitja del tub.

D'aquest anàlisi es pot assumir que el funcionament del sensor és adequat sempre i quan la pilota es mantingui a una certa distància del sensor, a uns 5 cm aproximadament; en cas que no sigui així, les lectures del sensor són errònies i comportarien un mal funcionament de la plataforma.

# 3. Control

Per garantir un correcte funcionament de la plataforma, s'ha d'establir unes regles o algoritme que siguin capaços de controlar el funcionament del sistema de tal manera que aquest realitzi les seves accions corresponents d'una manera òptima.

Tenint en compte com funciona la plataforma, s'han triat dos tipus de control diferents a aplicar. A partir d'aquests dos controls, es podrà observar com varia el funcionament de la plataforma segons el tipus de control que s'apliqui i també permetrà decidir quin dels dos és més òptim i adequat per a la plataforma tenint en compte diferents aspectes a observar, els quals més endavant es comentaran.

La funció principal de l'algoritme de control serà la de mantenir la pilota suspesa a una alçada concreta, definida com a *Setpoint*, intentant que realitzi el menor nombre d'oscil·lacions possibles, i que aquestes no siguin d'una amplada massa gran; és a dir, que la plataforma sigui capaç de mantenir la pilota a l'alçada establerta de la forma més eficient possible.

L'algoritme de control s'establirà (escriurà) a la placa Arduino, ja que és el component encarregat de fer funcionar la plataforma i, per tant, de realitzar el control d'aquesta. Aquest algoritme serà programat a la placa a través de l'entorn de programació IDE Arduino, del qual se n'ha parlat a l'apartat anterior, per tal de dur a terme les accions que posaran en marxa tots els components, els quals estan connectats a la placa, i fer-los funcionar aplicant el control.

Els dos controls que s'han decidit aplicar a la plataforma són el control amb histèresi i el control PID, els quals es veuran explicats amb detall en els propers apartats, juntament amb el codi per a la placa Arduino que s'ha emprat per poder dur a terme aquests controls.

## 3.1. Control amb histèresi

El control amb histèresi està basat en el control On-Off, el qual consisteix en un algoritme molt simple que conté dues posicions fixes: connectat o desconnectat. El control On-Off té un funcionament molt simple, el qual consisteix en activar l'accionador quan la variable controlada es troba per sota del valor desitjat (*Setpoint*) i en desactivar l'accionador quan la variable controlada es troba per sobre del *Setpoint*.

La diferència que hi ha entre el control amb histèresi respecte al control On-Off és que el control amb histèresi presenta el mateix funcionament que el control On-Off però amb un petit retard en el canvi del senyal per disminuir la quantitat de commutacions, prolongant així la vida útil dels components. Aquest retard o histèresi, en el cas de la plataforma i del seu algoritme de control, consistirà en uns centímetres de més o de menys respecte al *Setpoint*. És a dir, al control amb histèresi aplicat a la plataforma, el senyal commutarà quan la pilota estigui un número de centímetres concret per sobre o per sota del valor del *Setpoint*, fent que així el sistema no pateixi tantes commutacions i que aquestes siguin més controlades i precises.

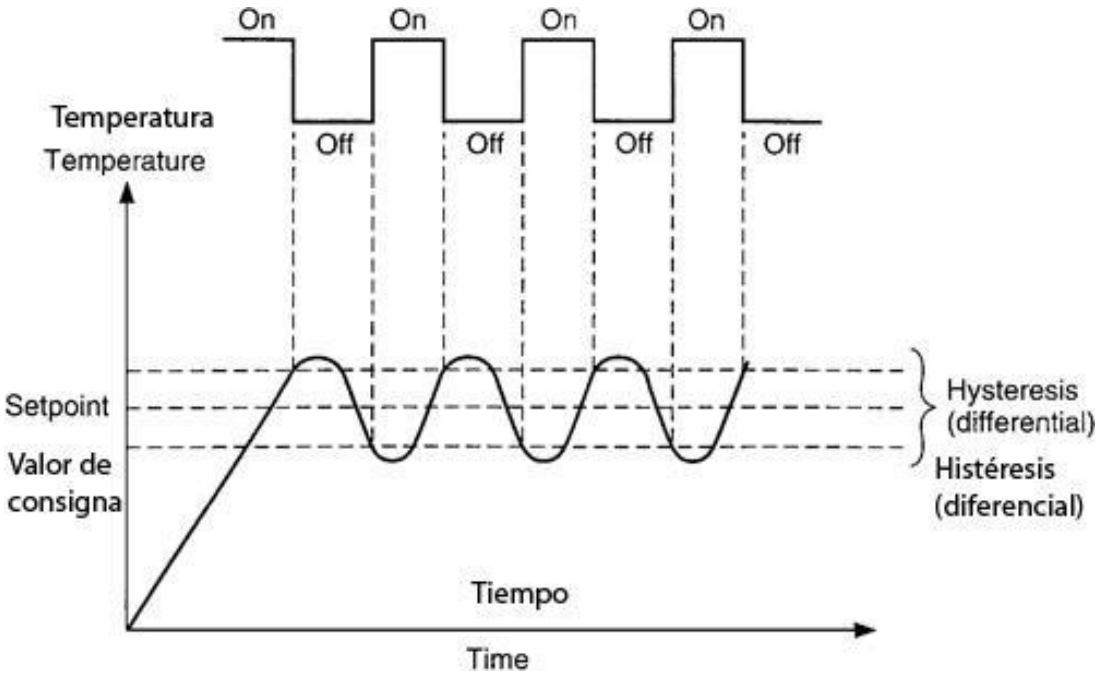



*Figura 19. Funcionament control tot-o-res amb histèresi.*

En base a aquest funcionament, es van començar a realitzar proves amb la plataforma, aplicant-li el codi adequat, el qual s'explicarà en detall més endavant.

La primera prova que es va realitzar va ser utilitzant el ventilador en màxima potència quan la pilota es trobés per sota del *Setpoint* menys la histèresi i desconnectant-lo totalment quan la pilota es trobés per sobre del *Setpoint* més la histèresi. Al dur a terme aquesta prova, es va poder comprovar que al alimentar el ventilador al màxim, l'ascens de la pilota era massa ràpid i, degut al temps de reacció de la placa per commutar el senyal després de passar pel *Setpoint* més la histèresi, la pilota pujava massa amunt. El mateix fet passava en el moment dels descens, el qual es produïa massa ràpid degut a que el ventilador estava totalment desconnectat.

Arrel d'aquesta prova i dels resultats obtinguts, es va optar per no alimentar el ventilador al 100% i no desconnectar-lo totalment en quant es produïssin les respectives condicions de commutació.

Per poder aplicar aquests canvis pel que fa a l'alimentació del ventilador, cal comprendre que aquest està alimentat a través de la sortida PWM de la placa Arduino, la qual correspon a la modulació per amplada de polsos. La PWM, tal com s'ha explicat anteriorment, permet variar el valor de la sortida en funció del *Duty Cycle* que se li estableixi. A més a més, cal tenir en compte que la sortida PWM de la placa és digital, per tant, els seus valors de sortida es troben en un rang de 0 a 255, on 255 correspon a un Duty Cycle del 100%.

### 3.1.1. Programa

Per poder realitzar aquest tipus de control a la placa Arduino caldrà aplicar el funcionament del control amb histèresi mitjançant la funció *if*. Aquesta funció consisteix en la dur a terme unes accions o línies de codi quan es dóna una condició concreta. En el cas del control que es vol aplicar, l'acció a aplicar serà la de alimentar o apagar el ventilador i la condició serà el valor de la lectura del sensor respecte el *Setpoint* establert en el propi codi.

Aquesta funció *if* serà l'última part del codi del control amb histèresi, el qual contindrà prèviament tota la declaració de variables i components, la definició d'entrades i sortides, la posta en marxa del sensor i la transformació de la lectura de segons a centímetres; és a dir, tots els aspectes esmentats en apartats anteriors referents a la placa Arduino.

```
if (DISTANCIA > SETPOINT + HISTERESIS){

  // encender ALIMENTACION
  analogWrite(VENTILADOR, 87);
  }

if (DISTANCIA < SETPOINT - HISTERESIS){

  //cortar ALIMENTACION
  analogWrite(VENTILADOR, 83);
  }
```

*Figura 20. Codi funcionament control amb histèresi.*

En el codi del control se li ha afegit unes línies de codi que permeten veure alguns valors concrets. Això s'aconsegueix a la seqüència *Serial.println* la qual permet agafar el valor d'una variable i anar-lo mostrant de forma contínua. Aquest valor es pot mostrar com un valor numèric, a través de l'eina *Monitor sèrie*; o en forma de gràfic, a través de l'eina *Serial plotter*.

### 3.1.2. Assajos i resultats

Per determinar el valor de la sortida PWM que rebrà el ventilador, es van realitzar diferents assajos amb diferents valors d'aquesta, tant per el moment de l'ascens com per el moment del descens.

**Valors de PWM 0 i 255.**

Tenint en compte que els valors PWM poden anar de 0 a 255, els quals equivalen al 0 i al 100% de revolucions a les que pot arribar la placa, es va començar realitzant un assaig amb aquests dos valors. Aquest assaig equivaldria a un control amb histèresi tot-o-res, el qual consisteix en alimentar al valor màxim i

treure-li l'alimentació a l'aparell segons si es troba per sobre o per sota de la histèresi.

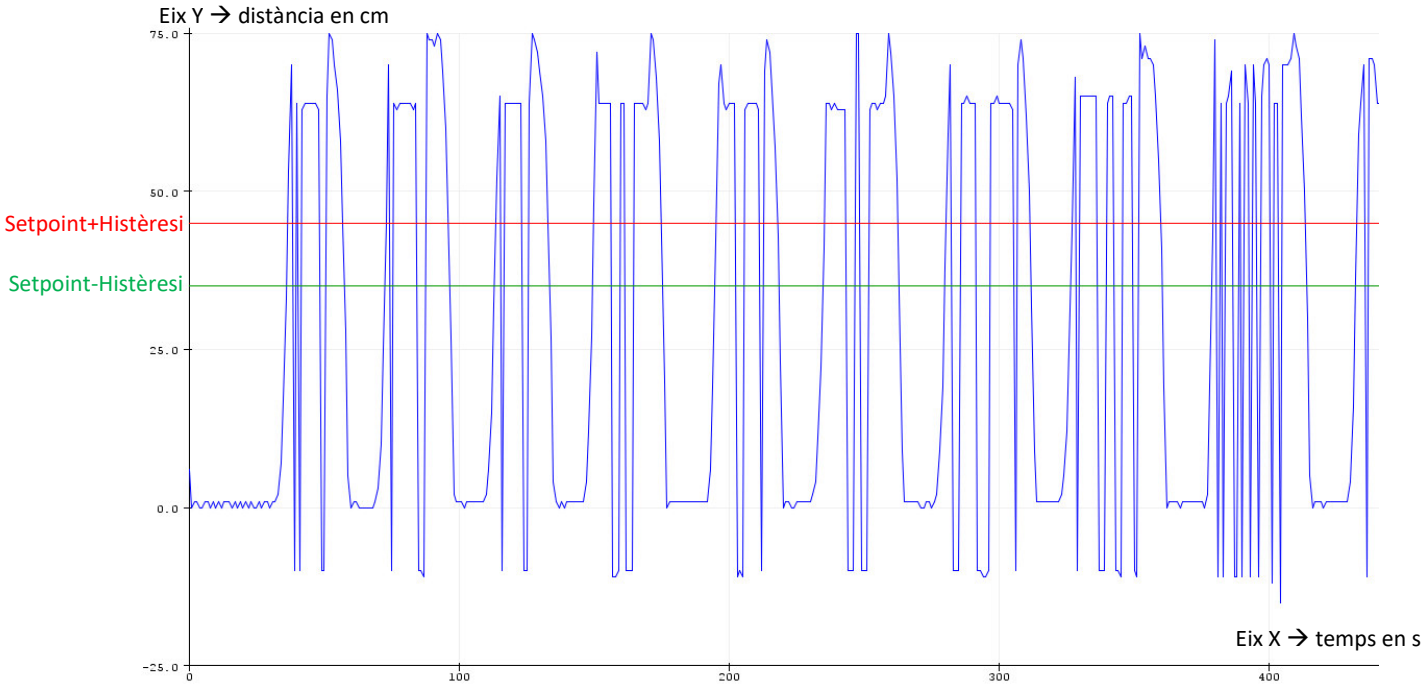



*Figura 21. Resultats assaig PWM=0 i PWM=255.*

Després de realitzar l'assaig amb aquests valors i observant les dades obtingudes (Figura 21) s'ha pogut comprovar que aquests dos valors de PWM, 0 i 255, no permeten que es realitzi un bon control del funcionament de la plataforma. Això es deu a que al proporcionar un valor de 255 a la PWM, les revolucions del ventilador són massa elevades i provoquen un ascens massa ràpid de la pilota, fent que pugi fins a la part superior del tub tot i sobrepassar el valor del *Setpoint* més la histèresi ja que al ascendir tant ràpid no dóna temps a parar la pilota quan està just per sobre de la histèresi. El mateix passa en el descens, ja que el ventilador s'atura totalment i la pilota cau sense res que la freni, de tal manera que el descens també es produeix de manera ràpida i la pilota arriba fins a la part inferior del tub tot i molt abans passar el punt del *Setpoint* menys la histèresi.

Sobre aquest assaig, cal comentar que a la gràfica apareixen certs valors negatius de la distància entre la pilota i la base del tub. Aquests valors són

provocats quan la pilota es troba massa a prop del sensor, provocant que el sensor prengui lectures errònies. Aquest fet també s'esmenta a l'apartat 2.3.1.

A partir d'aquests resultats, s'ha pogut comprovar que un control tot-o-res no és aplicable en aquest cas degut a la distància en la qual es mou la pilota i també degut al temps de reacció que té el ventilador per canviar d'una velocitat a una altra.

**Valors de PWM 50 i 100.**

En base als resultats del primer assaig, s'ha optat per acotar els valors de PWM que s'aplicaran, per poder així controlar millor l'ascens i el descens de la pilota i aconseguir acostar-se al funcionament més semblant possible d'un control amb histèresi. En aquest segon assaig s'ha provat d'utilitzar els valors de PWM de 50 i 100. Aquests dos valors han estat escollits per evitar tant l'ascens massa ràpid produït pel valor de 255, com el descens també ràpid; aquest últim aplicant una resistència provocada per una baixa velocitat en el gir del ventilador.

Els resultats d'aquest nou assaig es poden veure a la següent figura (Figura 22):

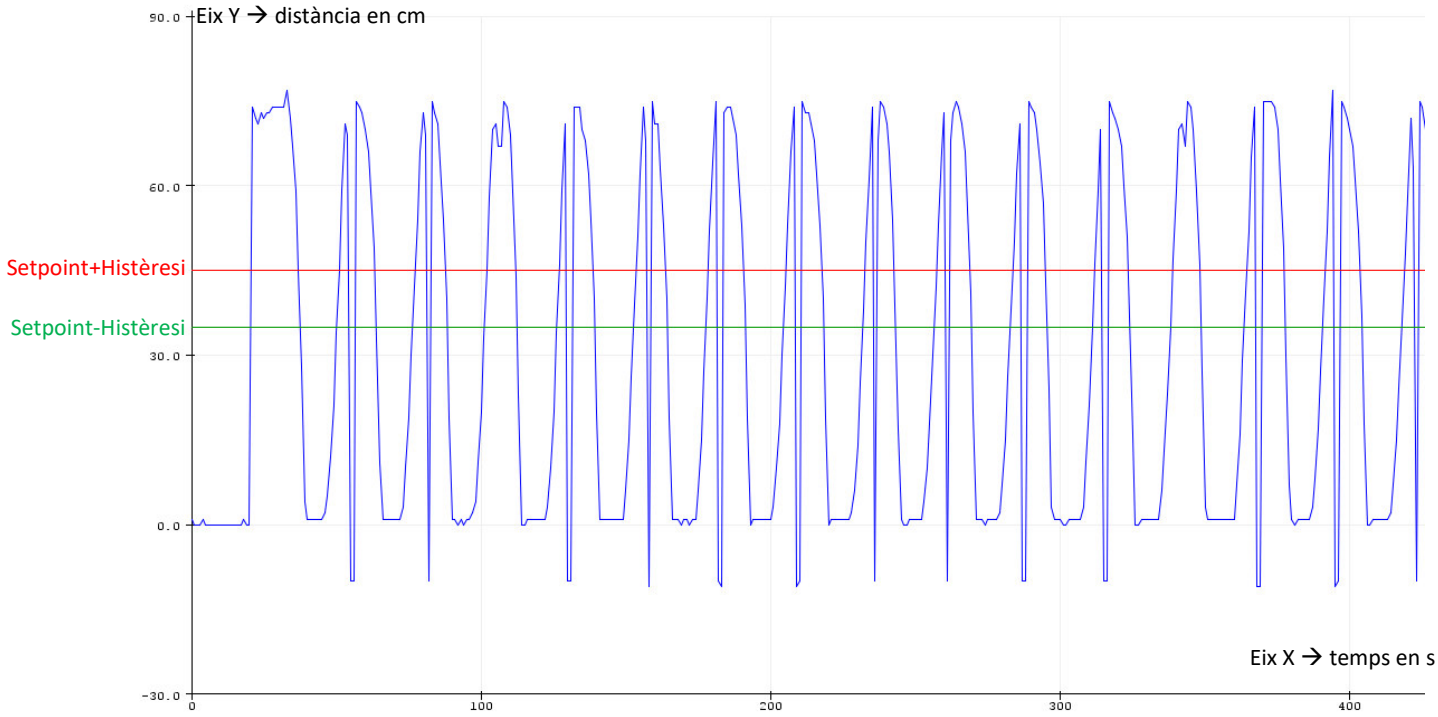



*Figura 22. Resultats assaig PWM=50 i PWM=100.*

Observant la gràfica resultant d'aquesta assaig, es pot veure que tant l'ascens com el descens no són tan ràpids com en el primer assaig. Tot i aquesta millora, ambdues velocitats, d'ascens i de descens, segueixen sent massa elevades i fan que la pilota arribi a la part superior i a la part inferior del tub; és a dir, no dóna temps a realitzar el canvi de velocitat de gir just quan sobrepassen el valor de la histèresi provocant així un mal control sobre la pilota.

En resum, els valors de PWM 50 i 100 tampoc són idonis per realitzar un bon control ja que provoquen el mateix resultat que en el control tot-o-res però a unes velocitats més baixes.

**Valors de PWM definitius.**

Després de la realització d'aquests assajos, es van poder acotar encara més els valors de PWM, de tal manera que es van realitzar més assajos amb valors entre 80 i 90 per a PWM. En base a aquests assajos en un rang més acotat, es van poder concretar els valors més òptims per a la sortida PWM de la placa Arduino, els quals són: 85 quan la pilota es troba per sota del *Setpoint* menys la histèresi, i 82 quan la pilota es troba per sobre del *Setpoint* més la histèresi. Aquests dos valors es troben en el rang de 0 a 255 que estableix la sortida PWM de la placa Arduino i són els dos valors que permeten un funcionament òptim de la plataforma utilitzant aquest tipus de control.

Després de definir els dos valors de PWM més òptims, s'ha comprovat que en el funcionament de la plataforma amb el control amb histèresi no és ideal, és a dir, que quan la pilota passa per sobre del valor del *Setpoint* més la histèresi la reacció no és instantània i la pilota es passa uns centímetres d'aquest valor; i el mateix passa quan es troba per sota del *Setpoint* menys la histèresi. A partir de diferents assajos on l'objectiu era definir de forma precisa aquest error, és va obtenir que l'error era entre 2 i 8 centímetres.

Per comprovar aquest error s'ha utilitzat l'eina *Serial plotter*, la qual permet realitzar una gràfica contínua al llarg del temps de diferents variables controlades per la placa. La gràfica següent és el resultat de l'ús d'aquesta eina per veure

la variació de la lectura del sensor respecte el valor del *Setpoint*, tenint en compte el valor de la histèresi.

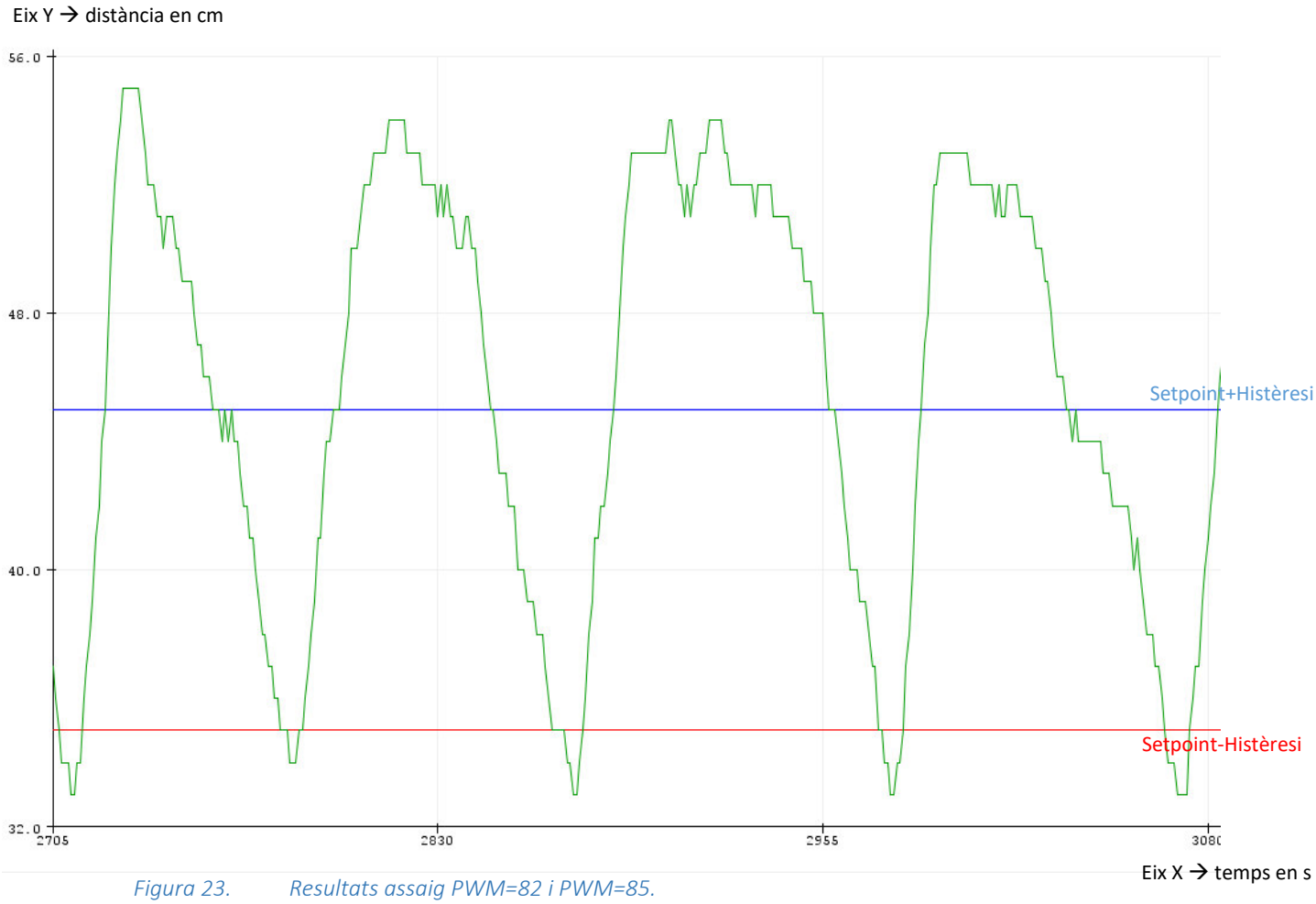



*Figura 23. Resultats assaig PWM=82 i PWM=85.*

Observant aquesta gràfica, la qual mostra el comportament de la lectura del sensor després d'haver-se estabilitzat, es pot veure com el valor de les lectures del sensor va oscil·lant entre els valors del *Setpoint* ± histèresi amb l'error que es comentava anteriorment. Aquestes oscil·lacions són poques i amb una amplada gran, fent així que el sistema mantingui certa estabilitat. Es pot veure un error més gran respecte el punt del *Setpoint* més la histèresi, i això es pot deure a que l'ascens de la pilota és més ràpid en comparació al seu descens, és a dir, que la força que provoca l'aire del ventilador fa que la pilota pugi més ràpid del que baixa.

Provant amb valors inferiors de la PWM per al descens de la pilota només provocava que l'error fos més gran, ja que el descens era massa ràpid; per això s'ha decidit mantenir els valors de PWM que han donat com a resultat aquesta gràfica.

Els valors de la sortida PWM que han proporcionat aquest resultat han estat:

- PWM d'ascens de la pilota = 85
- PWM de descens de la pilota = 82
- Setpoint = 40 cm
- Histèresi = 5 cm

Per verificar el correcte funcionament d'aquests valors de PWM també s'ha realitzat algun assaig canviant el valor de la histèresi, provant així quin efecte pot tenir el canvi en el valor d'aquesta distància.

L'assaig del qual es veuran els resultats a la propera figura (Figura 24) ha estat realitzat amb una histèresi de 15 cm.

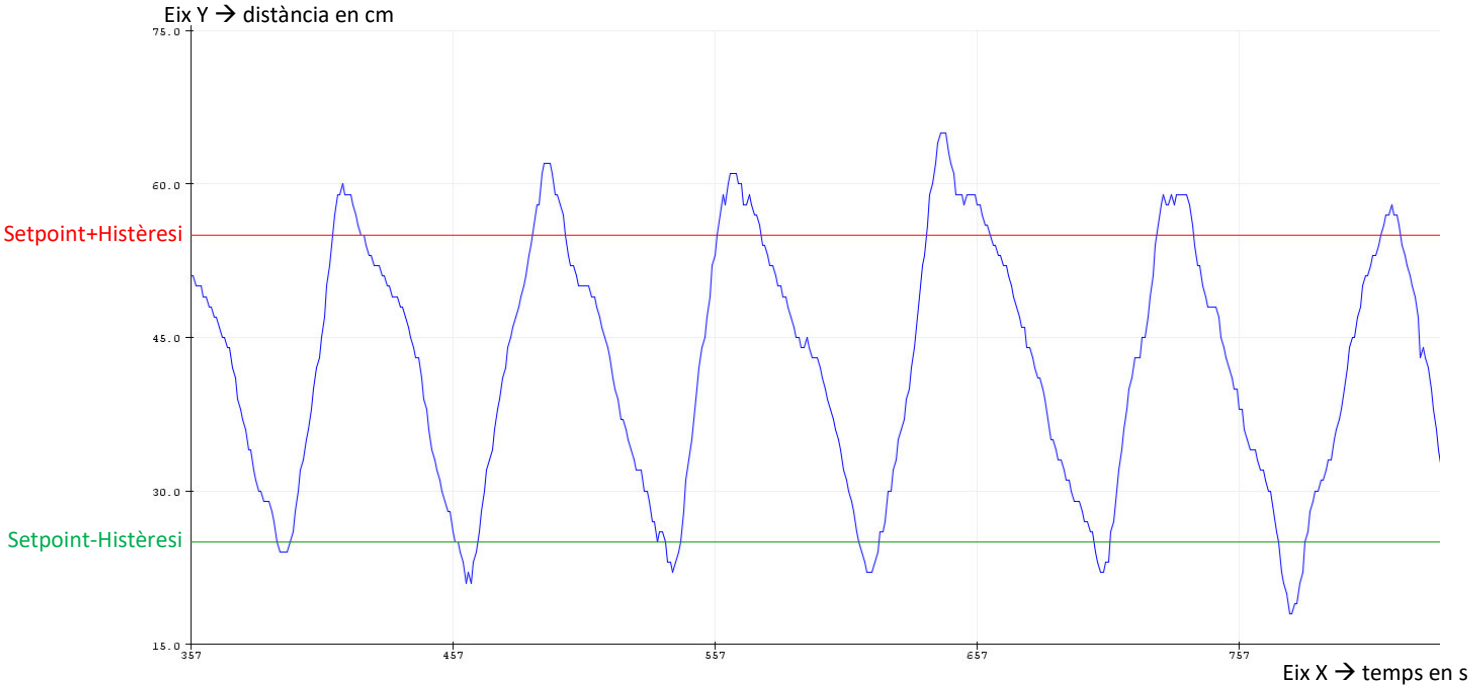



*Figura 24. Resultats assaig valors PWM definitius i histèresi=15 cm.*

Tal i com es veu a la figura, l'error que es produeix segueix sent d'entre 2 i 8 cm, per tant, el valor de la histèresi no implica canvis en els resultats del funcionament de la plataforma (fent referència a l'efectivitat del control).

## 3.2. Control PID

El control PID és un mecanisme de control genèric sobre una realimentació de bucle tancat, utilitzat sobretot a la indústria per a control de sistemes; per tant, és un mètode de control més complex que el control amb histèresi. El PID és un sistema al que li entra el valor de l'error calculat a partir de la sortida desitjada menys la sortida obtinguda, i aquesta sortida és utilitzada com a entrada al sistema que es vol controlar. El controlador pretén minimitzar l'error ajustant l'entrada del sistema.

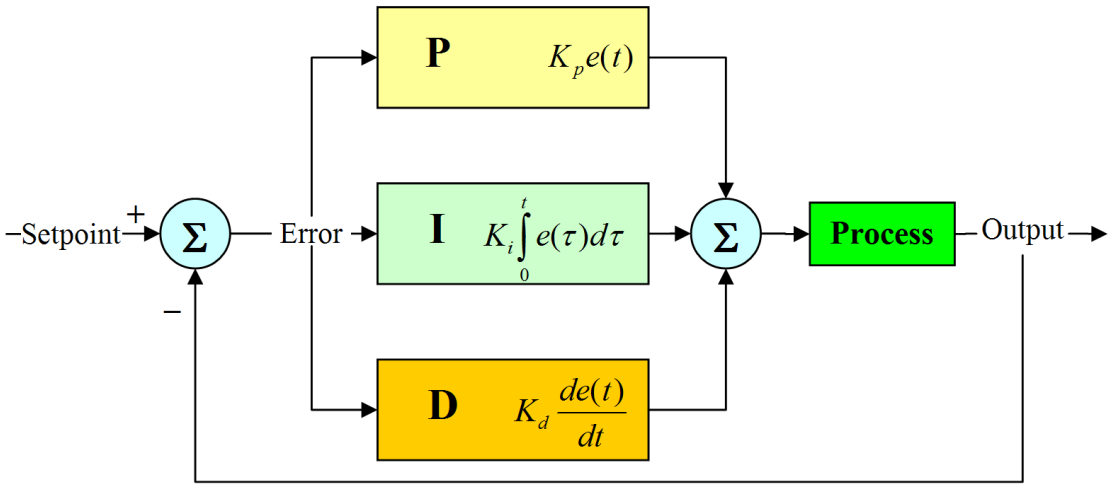



*Figura 25. Diagrama de blocs del control PID.*

El controlador PID ve determinat per tres paràmetres:

- Proporcional: provoca que el canvi present a la sortida del controlador sigui un múltiple del percentatge del canvi en la mesura. Aquest múltiple s'anomena guany.
- Integral: dóna una resposta proporcional a la integral de l'error. Aquesta acció elimina l'error en règim estacionari, provocat per l'acció proporcional. Per contra, provoca que s'obtingui un major temps

d'establiment, una resposta més lenta i un període d'oscil·lació major que en el cas de l'acció proporcional.

- Derivativa: dóna una resposta proporcional a la derivada de l'error. Aquesta acció disminueix l'excés de sobreoscil·lacions.

$$u(t) = K_p * e(t) + \frac{K_p}{T_i} * \int_0^r e(t)dt + K_p * T_d * \frac{de(t)}{dt} \qquad \text{(Eq. 4)}$$

Cada un d'aquests paràmetres influeix en major mesura sobre alguna característica de la resposta o sortida, però també influeixen sobre les demés, fet que provoca que no es trobi un controlador PID ideal.

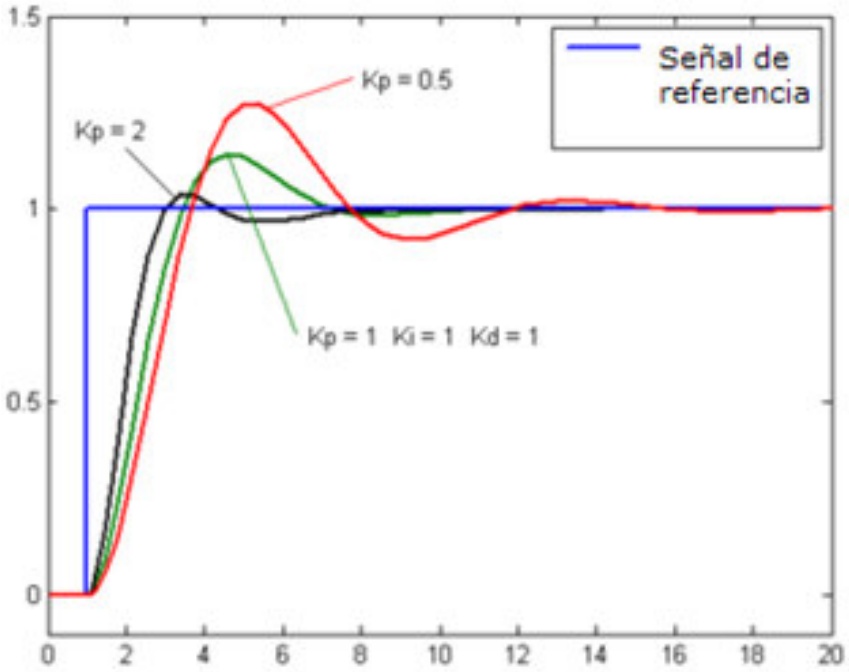



*Figura 26. Funcionament del control PID.*

Per començar, cal entendre el funcionament del control PID, el qual s'ha explicat al principi d'aquest mateix apartat; així que se sap que el valor de la sortida PWM s'anirà ajustant segons el valor que rebi del sensor, ja sigui augmentant el valor de la sortida quan l'entrada del sensor doni un valor inferior al *Setpoint* o disminuint el valor de la sortida quan l'entrada del sensor doni un valor superior al *Setpoint*.

També serà important conèixer el funcionament de la llibreria PID que s'ha emprat en el codi per a la placa Arduino, la qual s'explicarà més endavant, a l'apartat d'explicació del programa realitzat per a aquest control.

### 3.2.1. Programa

Per poder aplicar el funcionament d'un controlador PID al programa del control, ha estat necessari importar una llibreria que simulés el funcionament d'aquest tipus de controlador. Per incloure una llibreria al codi cal utilitzar la seqüència *inlcude* juntament amb el nom de la llibreria, la qual ha d'estar desada a la mateixa carpeta que la resta de llibreries que portava incloses el software d'Arduino.

```
control_pid
#include <PID_v1.h>
```

*Figura 27. Codi per incloure la llibreria PID.*

Un cop importada la llibreria la codi, cal entendre el seu funcionament i com funcionen els paràmetres que utilitza. La característica particular d'aquesta llibreria és que el valor de la sortida (*Output*) que calcula no és un càlcul directe respecte al valor de l'entrada (*Input*) i de l'error, sinó que el valor de la sortida que dóna és o un valor alt (High o 255 en PWM) o un valor baix (Low o 0 en PWM) en funció de si el valor de l'entrada està per sobre o per sota del valor del *Setpoint* que es defineixi.

Un cop conegut el funcionament de la llibreria, cal saber quines seqüències de codi cal posar al programa de control per poder aconseguir que la placa executi aquest tipus de control.

Per començar, cal definir les variables *Input, Output* i *Setpoint* com a variables de tipus *double*, la qual permet tenir el doble de precisió. També caldrà definir els paràmetres Kp, Ki i Kd com a variables del mateix tipus i donar-los un valor concret, en cas de que es vulgui donar valor en aquest punt del codi. Després es defineixen aquests paràmetres com a part de la llibreria PID.

```
double Setpoint;  //
double Input;   //sensor
double Output;  //ventilador

//PID parameters
double Kp=2, Ki=0.01, Kd=1.5;

PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
```

*Figura 28. Codi per definir els paràmetres de la llibreria PID.*

Degut al funcionament de la sortida del controlador PID, s'ha optat per limitar els valors de la sortida i així poder ajustar-la als valors adequats de la sortida PWM que permeten un bon funcionament de la plataforma. Per realitzar aquesta limitació, es fa a través de la seqüència de codi *SetOutputLimits*, la qual permet establir els límits de la sortida entre -255 i 255. Tenint en compte els valors de la PWM extrets del control amb histèresi, s'ha optat per limitar els valors entre -4 i 4 després de realitzar diferents assajos. Per fer que la plataforma treballi amb valors de la sortida PWM adequats, el valor de la sortida es suma a un valor de PWM de 84, el qual resulta ser un punt intermig dels punts utilitzats en el control amb histèresi.

També és necessari definir el mode de funcionament del controlador PID que simula la llibreria, el qual s'ha establert a *Automatic*.

```
myPID.SetOutputLimits(-4,4);
myPID.SetMode(AUTOMATIC);
```

*Figura 29. Definició del mode de funcionament i OutputLimits.*

Per últim, cal executar el funcionament de la llibreria PID amb la seqüència *Compute*, la qual s'encarrega de realitzar la funció que faria el controlador PID. Aquesta seqüència és la que dóna forma al control, i ha d'anar precedida de totes les línies de codi que inicialitzen i defineixen el funcionament dels components connectats a la placa (sensor i ventilador).

```
myPID.Compute();
```

*Figura 30. Crida de la llibreria PID.*

### 3.2.2. Assajos i resultats

A partir de totes les dades recopilades en l'ús del control amb histèresi, ja se sap que el control que s'apliqui no pot tenir uns valors qualsevols de la sortida PWM; per tant, a partir d'aquestes limitacions es podrà fer el control PID evitant tot el procés de prova i error amb el valor de la sortida PWM pel que s'ha passat al utilitzar el control amb histèresi.

En aquest tipus de control hi ha diferents paràmetres per ajustar, els quals fan que el funcionament de la plataforma segueixi una tendència o una altra. Els paràmetres que s'han anat ajustant fins a trobar la configuració més apropiada són les tres accions del controlador PID: Kp, Ki i Kd; i el límit dels valors de la sortida. Tant els paràmetres de control com el límit de la sortida provocaven variacions en l'amplada de les oscil·lacions i també en la quantitat que es produïa d'aquestes.

La primera prova que es va fer vas ser utilitzant només l'acció proporcional, és a dir, donant valor de 1 a la Kp i un valor de 0 a la Ki i a la Kd. Es va escollir Kp=1 com a primera prova per veure si l'acció proporcional funcionava correctament, és a dir, que el valor de la sortida fos el mateix que el de l'entrada degut a que:

$$Output = Kp * Error \quad \text{(Eq. 5)}$$

L'equació del controlador PID queda d'aquesta manera degut als valors de Ki i Kd que s'han definit en aquesta primera prova.

En aquesta prova s'ha vist que la resposta no era l'esperada, és a dir, el valor de la sortida no era el mateix que el de l'entrada a l'acció proporcional degut a que el funcionament del codi corresponent al controlador PID no funcionava de la manera que s'esperava.

Analitzant aquest problema i realitzant diferents assajos es va concloure que la sortida del codi del controlador PID, el qual s'executava a través de la importació d'una llibreria ja feta a la placa Arduino, mostrava un valor alt (High o 255) quan la pilota es trobava per sobre del *Setpoint* i un valor baix (Low o 0) quan aquesta es trobava per sota d'aquest punt.

Tenint en compte aquest funcionament i a partir de l'anàlisi dels valors òptims per a la sortida PWM que s'ha fet a l'apartat del control amb histèresi, es va decidir limitar els valors de la sortida del controlador PID per així poder aconseguir un correcte funcionament de l'acció proporcional.

**Kp=10, Ki=0 i Kd=0.**

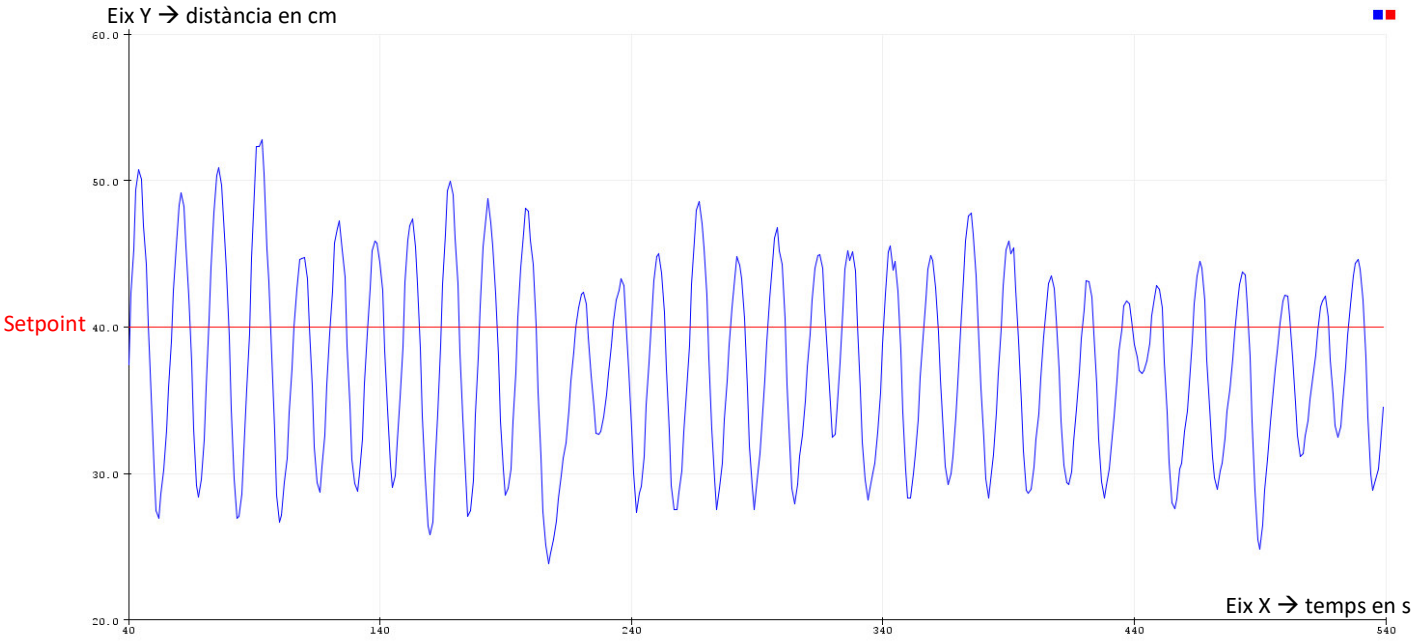



*Figura 31. Resultats assaig Kp=10, Ki=0 i Kd=0.*

Tal com s'ha comentat anteriorment, després d'aplicar la limitació de la sortida de la funció PID, s'ha realitzat un assaig aplicant només l'acció proporcional amb valor de 1. A partir d'aquest assaig es pot determinar que l'acció proporcional per si sola no és capaç d'estabilitzar el comportament de la pilota. Com es pot veure a la imatge anterior (Figura 31), el comportament de la pilota és bastant regular, mantenint unes oscil·lacions d'una amplitud gairebé igual a tota la gràfica, però en cap moment arribant a fer que la pilota es mantingui a una distància propera al valor establert com a *setpoint*.

Després de comprovar el correcte funcionament de l'acció proporcional, s'han aplicat valors a l'acció integral i derivativa per veure com canviava el funcionament del sistema al aplicar tots els paràmetres del control PID.

Primer s'ha optat per introduir l'acció integral a l'acció proporcional que ja es tenia, sense aplicar l'acció derivativa; tot això per observar quins canvis podia provocar en el funcionament de la plataforma.

**Kp=10, Ki=1 i Kd=0.**

Al aplicar l'acció integral en aquest assaig, s'ha observat que el temps d'estabilització ha estat més elevat que a l'assaig anterior, demostrant així la presència de l'acció integral.

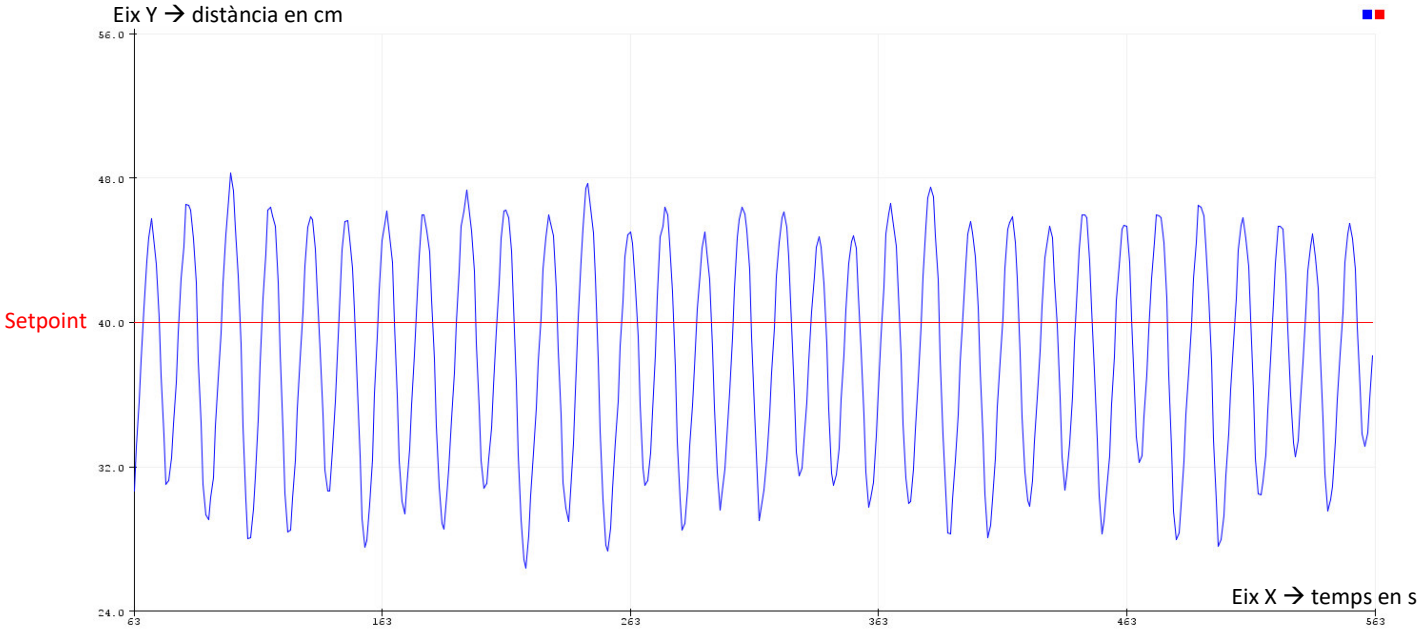



*Figura 32. Resultats assaig Kp=10, Ki=1 i Kd=0.*

Tal com es pot veure a la Figura 32, l'acció integral no ha afectat en l'amplitud de les oscil·lacions ja que no s'ha alterat el valor de l'acció proporcional; però aquesta acció integral si que ha provocat que la quantitat d'oscil·lacions fos més elevada. Aquesta conseqüència implica que el valor aplicat a l'acció integral és massa elevat per a les necessitats d'aquest projecte.

El següent pas ha estat aplicar l'acció derivativa, mantenint els valors de les accions que ja s'han aplicat anteriorment, per poder observar quin efecte té cada acció en el comportament de la plataforma i de quina manera es podrà millorar.

**Kp=10, Ki=1 i Kd=5.**

L'ús de l'acció derivativa, juntament amb les accions ja emprades anteriorment, ha provocat en el comportament de la plataforma un cert canvi. Tal com diu la teoria, ha quedat demostrat que l'acció derivativa pot implicar un augment en l'amplitud de les oscil·lacions; però també ha provocat que la pilota, tot i moure's de forma estable, tingués dificultats per arribar a sobrepassar el *setpoint*, és a dir, les oscil·lacions pròpies d'aquest control amb els valors utilitzats es produïen però per sota del *setpoint*, no utilitzant aquest com a referència o punt mig en les oscil·lacions. Aquest comportament es pot veure a la següent figura (Figura 33):

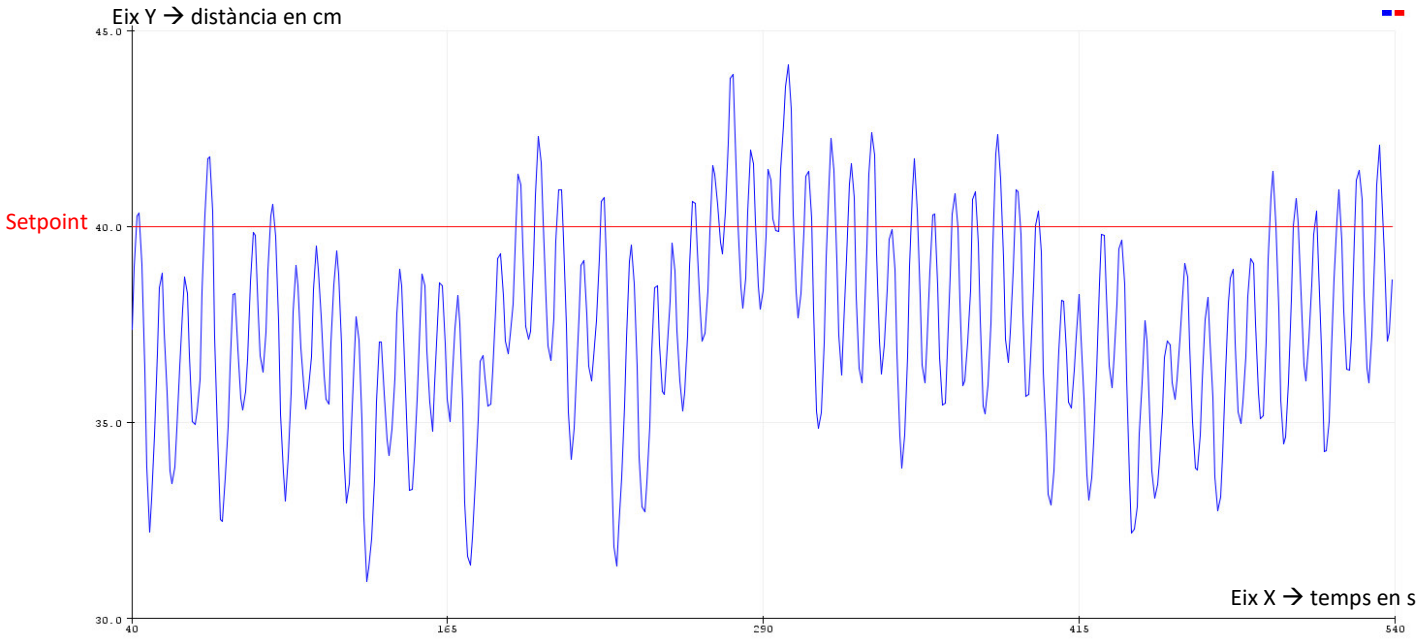



*Figura 33. Resultats assaig Kp=10, Ki=1 i Kd=5.*

Les característiques esmentades en els assajos anteriors, com l'elevada quantitat d'oscil·lacions i l'amplitud de d'aquestes, es manté en aquest assaig ja que els valors de les accions proporcional i integral no s'han alterat des del primer assaig.

Un cop realitzats aquests primers assajos aplicant per primer cop cada tipus d'acció pròpia del control PID, els propers assajos s'han emprat per anar alterant i acotant els valors de les tres accions (proporcional, integral i derivativa) per aconseguir un funcionament més estable i adequat per al projecte.

Per començar, s'han disminuït els valors de Kp, Ki i Kd ja que els valors utilitzats en els assajos anteriors eren massa elevats i no permetien realitzar un control adequat sobre la plataforma.

**Kp=5, Ki=0,5 i Kd=2,5.**

En aquest assaig s'han reduït a la meitat els valors de cada acció, permetent així veure com les parts negatives que provoca cada acció no són tant notòries com en els primers assajos.

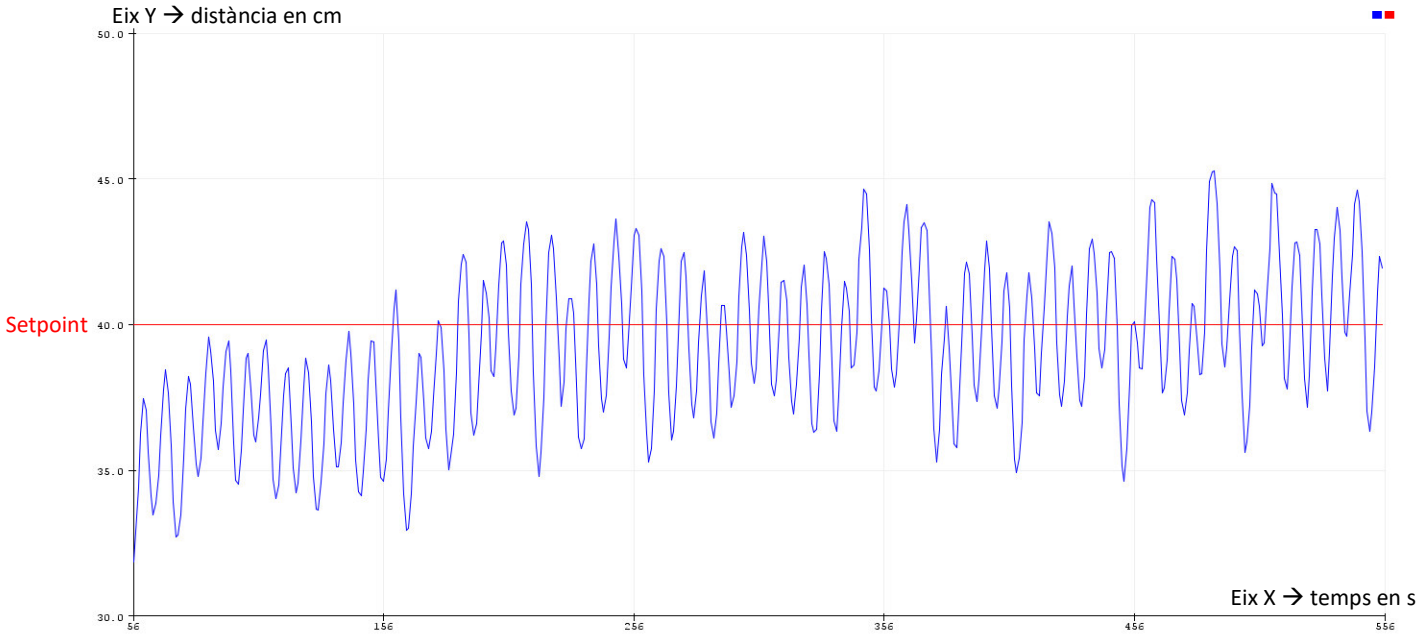



*Figura 34. Resultats assaig Kp=5, Ki=0,5 i Kd=2,5.*

Observant la Figura 34, corresponent a les dades d'aquest assaig, es pot veure com l'amplitud de les oscil·lacions ha disminuït, passant d'uns 20 cm en els assajos anteriors a uns 10 cm aproximadament. Aquest canvi es deu al canvi de valor de l'acció derivativa a un valor inferior, en aquest cas passant de 5 a 2,5. També es pot apreciar la reducció en el nombre d'oscil·lacions que es produeixen durant el funcionament, un canvi provocat per passar d'un valor de l'acció integral de 1 a 0,5.

Amb aquests canvis, queda demostrat que al reduir els valors de Kp, Ki i Kd el control sobre la plataforma ha millorat respecte els primers assajos; però aquest

control encara es pot millorar i per això s'ha prosseguit amb els assajos per acotar encara més els valors de les accions del control.

Mantenint la mateixa dinàmica que en aquest experiment anterior, els valors de les accions del control s'han tornat a reduir a uns valors encara més petits.

**Kp=2, Ki=0,1 i Kd=2.**

Aplicant aquests nous valors al control, l'assaig realitzat mostra com les característiques que en l'assaig anterior han millorat ho tornen a fer en aquest cas.

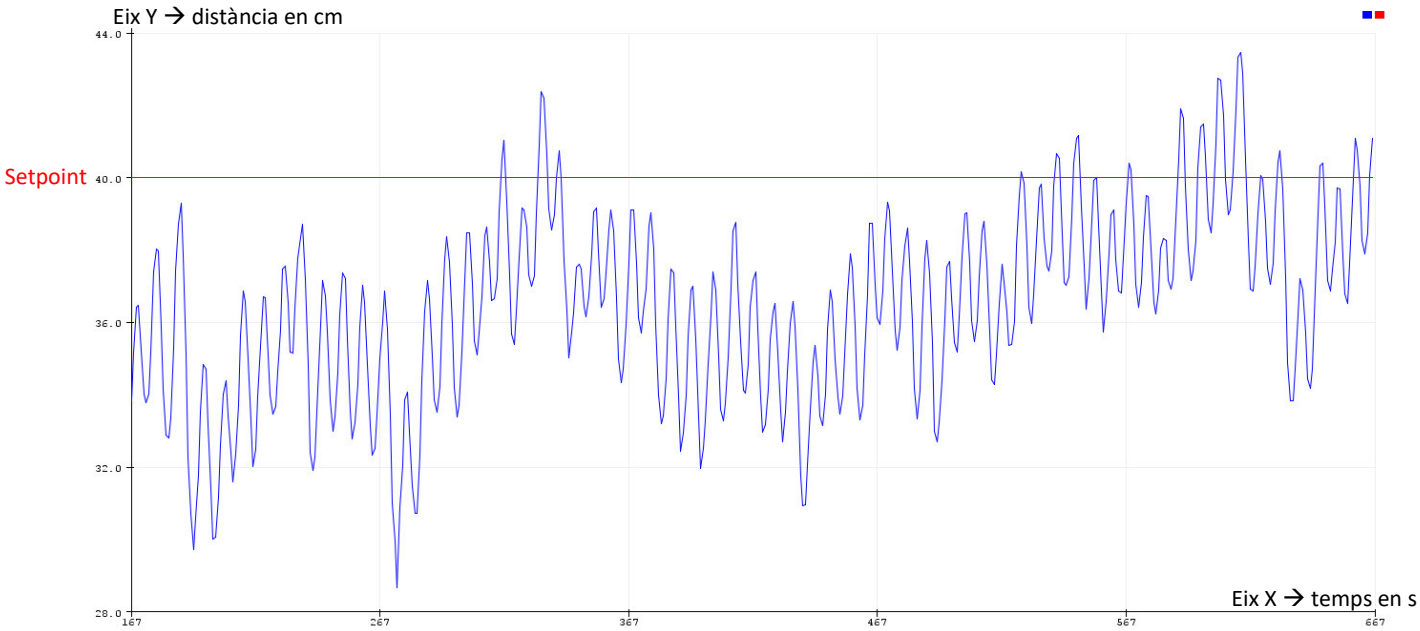



*Figura 35. Resultats assaig Kp=2, Ki=0,1 i Kd=2.*

Observant l'anterior imatge (Figura 35), queda reflectit com l'amplitud de les oscil·lacions ha tornat a disminuir respecte l'últim assaig realitzat. Respecte al nombre d'oscil·lacions, cal esmentar que no s'ha produït un a gran diferència respecte a l'anterior assaig. Cal destacar d'aquest assaig una diferència amb l'anterior, i és que l'estabilització ha estat més lenta. Tal com es veu a la gràfica, durant uns quants segons, les oscil·lacions s'han produït per sota del valor de *setpoint*, tal com ha passat a l'assaig en que s'ha introduït l'acció derivativa.

D'aquestes dades es pot extreure que encara es pot millorar el control, sobretot ajustant l'acció integral i l'acció derivativa ja que l'acció proporcional, tot i que no es mostri a les gràfiques, no està comportant problemes al control.

**Kp=2, Ki=0,05 i Kd=1,5.**

Per seguir millorant aquest control, s'ha optat per reduir només els valors de Ki i Kd.

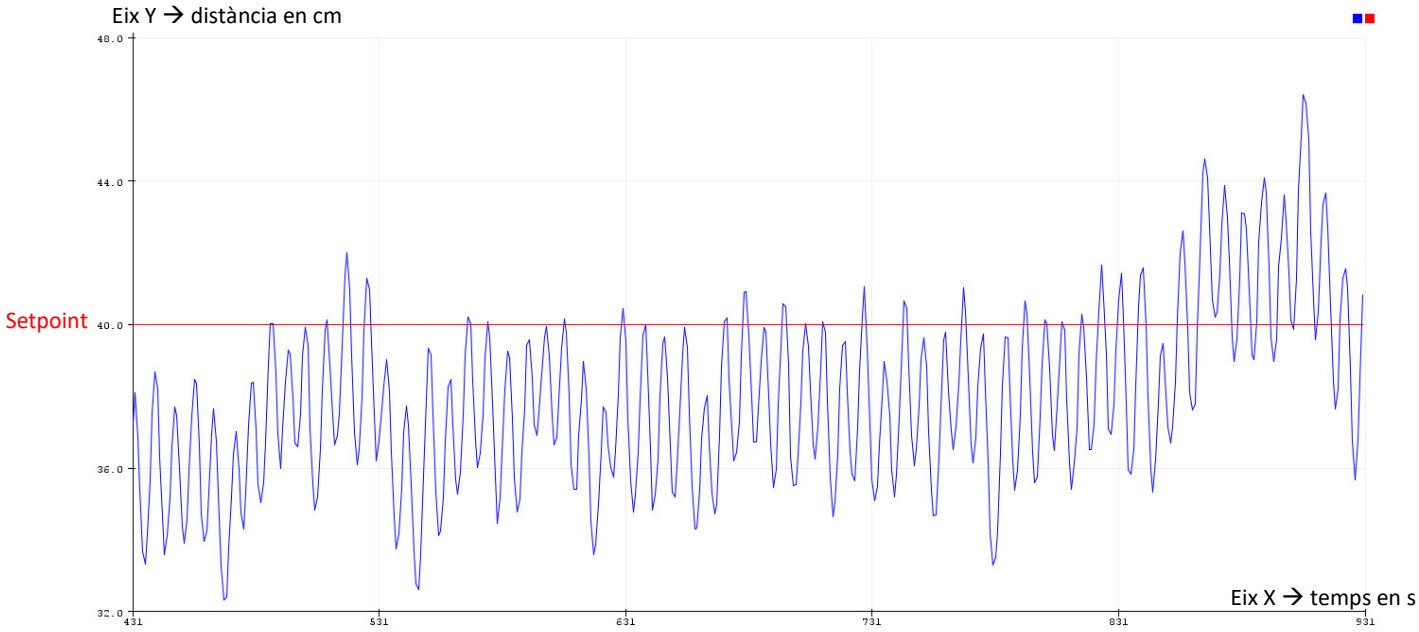



*Figura 36. Resultats assaig Kp=2, Ki=0,05 i Kd=1,5.*

Tal i com es veu a la Figura 36, amb aquests valors de les accions del control PID també es produeix una estabilització més lenta que en altres assajos, però en benefici del funcionament de la plataforma cal esmentar la reducció en l'amplitud de les oscil·lacions, les qual estan al voltant del valor d'uns 6 cm, fet que implicaria que la pilota sobrepasses, ja sigui per sota o per sobre del *setpoint*, uns 3 cm de mitja aproximadament. Observant totes les oscil·lacions, es pot veure com l'error de la mesura respecte al *setpoint* es troba entre 2 i 6 cm.

També cal fer referència a un menor valor en el nombre d'oscil·lacions, representant una major estabilitat en el control que en els casos anteriors. Amb aquestes dades es pot assumir que aquest assaig és el que representa un millor

control respecte als anteriors. Però també cal contemplar l'opció de seguir reduint els valors dels paràmetres Kp, Ki i Kd per veure si es pot obtenir un millor resultat.

**Kp=2, Ki=0,01 i Kd=1,5.**

En aquest següent assaig només s'ha alterat el valor del paràmetre Ki, per veure si és possible reduir encara més la quantitat d'oscil·lacions durant el funcionament. També s'ha canviat el valor del *setpoint* de 40 cm, valor utilitzat en els altres assajos, a 50 cm per veure també com podia afectar.

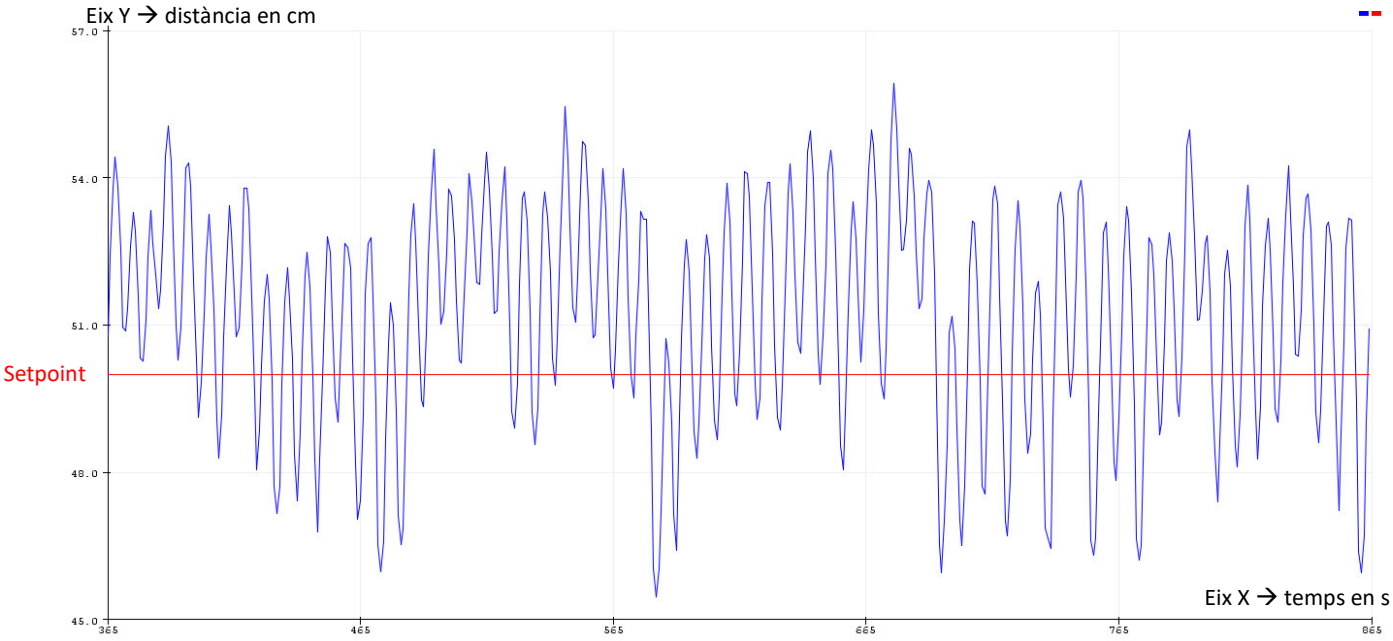



*Figura 37. Resultats assaig Kp=2, Ki=0,01 i Kd=1,5.*

Al fer més petit el valor de Ki, les oscil·lacions han tornat a augmentar en nombre, demostrant així que un valor massa petit del paràmetre Ki també pot causar que el control no sigui adequat. Respecte al canvi en el valor del *setpoint* no s'aprecia cap canvi respecte als altres assajos, els quals tenien un valor diferent de *setpoint*.

Veient aquest últim assaig, per comprovar que uns valors inferiors als del millor assaig realitzat poden millorar o no el control de la plataforma, s'ha realitzat un últim assaig canviant el valor de tots els paràmetres.

**Kp=3, Ki=0,01 i Kd=1.**

Amb aquest últim assaig es pretén establir els valors que provoquen un control més adequat per a la plataforma, ja siguin els valors emprats en aquest assaig obtenint uns millors resultats o amb els valors de l'assaig que ha proporcionat un millor resultat fins ara.

Amb aquests nous valors, s'han obtingut uns resultats bastant semblants al millor assaig obtingut, els quals es poden veure a la següent imatge (Figura 38).

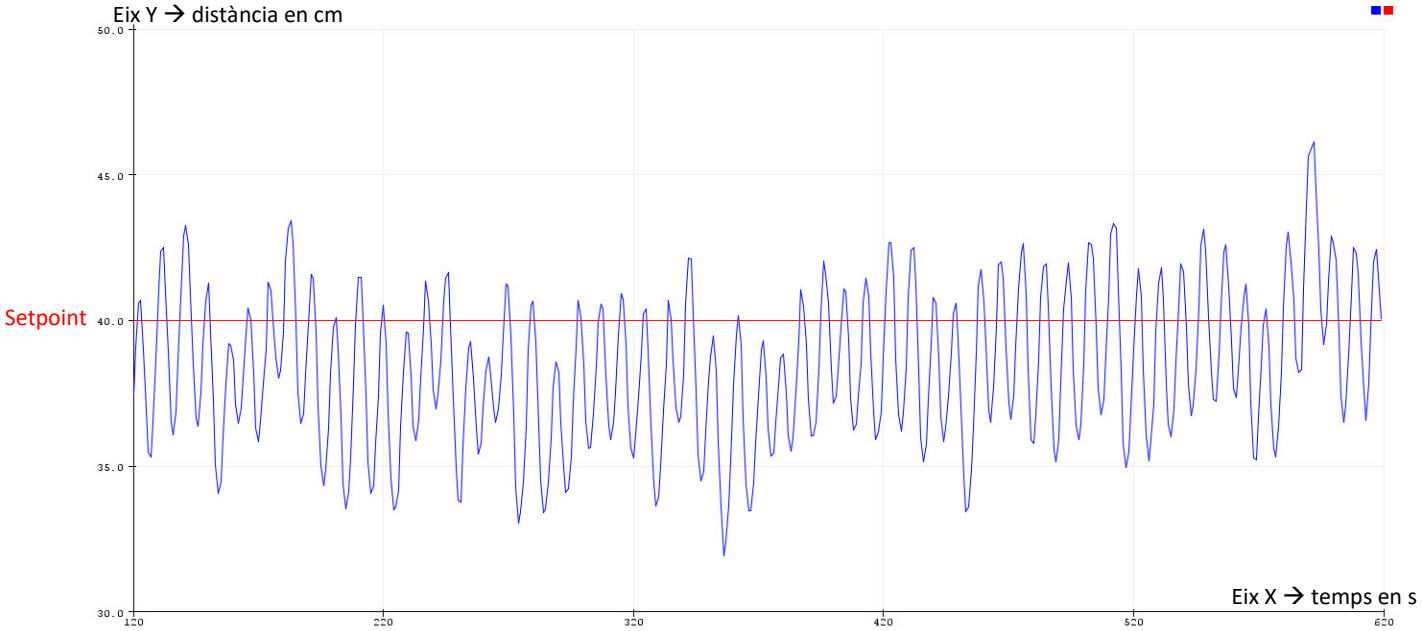



*Figura 38. Resultats assaig Kp=3, Ki=0,01 i Kd=1.*

Observant les dades del funcionament d'aquest assaig, es pot veure com el temps d'estabilització del control és bastant semblant a l'assaig amb millor resultat (Kp=2, Ki=0,05 i Kd=1,5), tot i que s'aprecia que es inferior; per tant, en aquesta característica del control no hi ha una gran diferència entre ambdós assajos. Si que es pot apreciar una diferència una mica major pel que fa a la quantitat d'oscil·lacions produïdes i a l'amplitud d'aquestes. En aquest assaig aquestes dues característiques no han millorat respecte a l'altre assaig esmentat anteriorment, sinó que són mínimament pitjors. És a dir, aquest assaig conté una major quantitat d'oscil·lacions i aquestes tenen una amplitud més elevada.

A partir dels diferents assajos realitzats amb diferents valors dels tres paràmetres, Kp, Ki i Kd, s'ha arribat a la solució més estable en les condicions que permet la plataforma.

Tal com s'ha esmentat anteriorment, dos dels assajos realitzats han mostrar comportaments molt semblants, amb unes petites diferències que han permès escollir el valor dels paràmetres que ofereix un millor resultat en les condicions en les que es realitza aquest projecte. L'assaig que ha ofert un millor funcionament ha estat el que ha treballat amb els següents valors per als paràmetres Kp, Ki i Kd:

- Kp = 2.
- Ki = 0,05.
- Kd = 1,5.
- OutputLimits = (-4, 4).

Observant aquest assaig, es pot veure que les oscil·lacions arriben a uns valors de ±6 cm respecte el *Setpoint*; per tant, el sistema no arriba a estabilitzar-se completament però ofereix una millor estabilitat respecte els altres assajos i unes millors característiques de funcionament. Aquests valors són els que proporcionen una resposta amb una amplada d'oscil·lacions menor, però té l'inconvenient que les oscil·lacions es produeixen amb certa rapidesa, fet que dificulta l'estabilització del sistema.

## 3.3. Comparativa entre els dos controls

Després de realitzar d'executar, ajustar i analitzar els dos controls que s'han implementat, es pot concloure que ambdós controls tenen avantatges i inconvenients pel que fa al funcionament de la plataforma.

En primer lloc, fixant-se en la quantitat d'oscil·lacions, el control amb histèresi té un millor funcionament que el control PID. Això es pot deure a que els valors de PWM que s'utilitzen en el control amb histèresi són dos valors fixes que varien

entre ells en funció de la posició de la pilota, donant així una major estabilitat pel que fa a aquesta característica de la resposta.

Observant ara una altra característica de la resposta, com l'amplada de les oscil·lacions, és a dir, l'error respecte al punt en el qual haurien de canviar de tendència; es pot veure que el control PID proporciona una resposta amb un error més petit respecte al control amb histèresi, el qual dóna una amplada d'oscil·lacions més alta sense tenir en compte el propi valor d'histèresi que es proporciona a aquest control.

A la següent taula es pot veure la comparació entre aquestes característiques en els dos controls:

| | Control amb histèresi | Control PID |
|---|---|---|
| Quantitat d'oscil·lacions/període de les oscil·lacions | Poca quantitat d'oscil·lacions, període alt. | Molta quantitat d'oscil·lacions, període petit. |
| Amplada de les oscil·lacions/error respecte punt de canvi | Error entre 1 i 7 cm respecte punt de canvi de funcionament. | Error entre 2 i 6 cm respecte punt de canvi de funcionament. |
| Estabilitat | Bona estabilitat al llarg d'un cert període de temps, tendència es manté. | Bona estabilitat al llarg d'un cert període de temps, tendència es manté. |

Taula 1. Comparativa de resultats dels controls.

Es pot concloure que els dos controls aconsegueixen mantenir una certa estabilitat dins del seu funcionament, els quals s'han explicat anteriorment amb les seves característiques i les seves diferències.

# 4. Connexió remota

Tal com s'ha esmentat en els objectius d'aquest treball, la finalitat del projecte és la realització d'un control de la plataforma *Ball in Tube* de forma remota mitjançant un dispositiu mòbil. Per aconseguir realitzar aquest control a distància caldrà establir una comunicació entre la placa Arduino i el dispositiu mòbil de tal manera que es puguin rebre i enviar dades per així poder realitzar l'esmentat control remot.

Quan es pensa en una comunicació remota entre diversos components, principalment es pensa en connexions *WiFi* o *Bluetooth*, ja que ambdues són molt conegudes i efectives a l'hora d'enviar o rebre dades a distància. A partir d'aquests dos tipus de connexions, s'avaluarà quina de les dues s'ha escollit coma solució per al control remot de la plataforma

## 4.1. Justificació i elecció

Tant la connexió via *WiFi* com la connexió via Bluetooth són mètodes per a realitzar connexions sense fils que permeten que alguns aparells es connectin amb altres aparells. Tot i així, la forma en la que aquests dos protocols treballen són molt diferents. La *WiFi* és una xarxa sense fils que permet als aparells, com ordinadors o *smartphones*, connectar-se a la xarxa a través d'un punt d'accés, com un *router*. En canvi, el *Bluetooth* és un estàndard que es va desenvolupar principalment per a telèfons per transferir dades a altres telèfons o a auriculars.

La tecnologia *Bluetooth* és molt útil quan es tracta de transferir informació entre dos o més aparells que estan a prop entre ells quan la velocitat de transmissió no és un problema. El *Bluettoth* és més adequat per a aplicacions amb una amplada de banda baixa, com transferir dades de so amb els telèfons.

La *WiFi* és més adequada per a xarxes que operen a gran escala perquè permet una connexió més ràpida, un millor rang des de l'estació base i una millor seguretat sense fils.

A continuació es pot observar una taula comparativa entre les característiques principals de les connexions *WiFi* i *Bluetooth*.

| Edit | Bluetooth | Wi-Fi |
|---|---|---|
| Frequency | 2.4 GHz | 2.4, 3.6, 5 GHz |
| Cost | Low | High |
| Bandwidth | Low ( 800 Kbps ) | High (11 Mbps ) |
| Specifications authority | Bluetooth SIG | IEEE, WECA |
| Security | It is less secure | Security issues are already being debated. |
| Year of development | 1994 | 1991 |
| Primary Devices | Mobile phones, mouse, keyboards, office and industrial automation devices. Activity trackers, such as Fitbit and Jawbone. | Notebook computers, desktop computers, servers, TV, Latest mobiles. |
| Hardware requirement | Bluetooth adaptor on all the devices connecting with each other | Wireless adaptors on all the devices of the network, a wireless router and/or wireless access points |
| Range | 5-30 meters | With 802.11b/g the typical range is 32 meters indoors and 95 meters (300 ft) outdoors. 802.11n has greater range. 2.5GHz Wi-Fi communication has greater range than 5GHz. Antennas can also increase range. |
| Power Consumption | Low | High |
| Ease of Use | Fairly simple to use. Can be used to connect upto seven devices at a time. It is easy to switch between devices or find and connect to any device. | It is more complex and requires configuration of hardware and software. |
| Latency | 200ms | 150ms |
| Bit-rate | 2.1Mbps | 600 Mbps |

*Figura 39. Taula comparativa entre Bluetooth i WiFi. Font: Bluetooth vs WiFi [en línia]. Disponible a: https://www.diffen.com/difference/Bluetooth_vs_Wifi*

Però a l'hora de triar una de les dues connexions no només cal fixar-se en les característiques principals, també cal centrar-se en les necessitats del projecte o utilitat que se li vulgui donar.

Analitzant l'ús que se li vol donar a la plataforma i al control remot, cal tenir en compte que el control a través del dispositiu mòbil es farà sempre des d'una distància propera a la plataforma per poder així verificar el correcte funcionament a l'hora d'enviar i rebre dades. Tenint en compte aquest fet, és més apropiat aplicar una connexió via *Bluetooth* ja que no serà necessari connectar-se a una xarxa ni estar a una gran distància. A més a més, no es garanteix que el punt on estigui situada la plataforma tingui una bona connexió *WiFi*, per tant, s'optarà per la connexió *Bluetooth* ja que assegura una bona connexió sempre que el dispositiu mòbil es trobi a prop de la plataforma, la qual contindrà el component que permet la connexió d'aquest tipus.

Tenint en compte que la plataforma es modifiqués d'ubicació, la connexió via *WiFi* suposaria un problema, ja que quan es realitza el programa que connecta la placa amb el dispositiu mòbil, es fa establint una adreça IP concreta; és a dir, que només permet que es realitzi la connexió *WiFi* només en una zona que tingui accés a aquella adreça IP. Per tant, si la plataforma es desplacés a un altre punt amb una altra adreça IP (una altra connexió *WiFi*), implicaria el fet d'haver de canviar el codi de la placa i de l'aplicació que realitza el control remot per poder tornar a establir una connexió *WiFi* entre la plataforma i el dispositiu mòbil. En canvi, amb una connexió *Bluetooth* això no suposaria un problema degut a que si es treballa amb aquest tipus de connexió, el codi obliga a realitzar la connexió entre la plataforma i el dispositiu cada vegada que es posa en marxa l'aplicació de control; és a dir, que qualsevol dispositiu que contingui l'aplicació de control pot posar en marxa la plataforma només posant en marxa l'aplicació i connectant-se via *Bluetooth* amb la placa.

Cal comentar que la connexió via *WiFi* seria molt útil en cas de tractar-se d'una plataforma que s'hagués de controlar i observar des d'una distància gran i que també permetria una gran velocitat de connexió i de transmissió de dades, sempre i quan la connexió fos bona. Però tenint en compte les condicions amb

les que es treballarà en aquesta plataforma, s'ha optat per implementar una connexió via *Bluetooth*.

## 4.2. Component encarregat de la comunicació sense fils

Per realitzar la connexió remota entre la plataforma i el dispositiu mòbil via *Bluetooth*, tal i com s'ha definit a l'apartat anterior, caldrà utilitzar algun mòdul o component que es connecti a la placa Arduino i permeti realitzar aquesta connexió sense fils.

Tot i que al final s'ha optat per establir una connexió via *Bluetooth*, inicialment s'havia pensat d'utilitzar una connexió *WiFi* a través del component *WiFi Shield* d'Arduino. Aquest component és un mòdul que es connecta a la part superior de la placa Arduino i través de la connexió dels seus pins habilita la connexió *WiFi* entre la placa i algun altre component o aparell.

Però degut las aspectes comentats a l'apartat anterior i alguna dificultat que ha sorgit durant la realització del projecte, s'ha optat per utilitzar una connexió *Bluetooth*. Aquesta connexió es pot fer a través de mòduls que es connecten a la placa Arduino. Hi ha certa varietat entre els mòduls de connexió *Bluetooth* per a Arduino, i ara s'explicarà quin s'ha escollit i la justificació d'aquesta decisió.

Després de fer una recerca, s'ha trobat que els mòduls més utilitzats per a establir aquest tipus de connexió entre una placa i un aparell són els models HC-04, HC-05 i HC-06. Aquests tres models tenen unes característiques molt semblants i pràcticament funcionen de la mateixa manera, però tenen certes diferències que han permès decantar la balança cap a l'elecció d'un dels tres per sobre dels altres dos.

Sobre aquests mòduls, cal començar dient que els dispositius *Bluetooth* poden actuar com a *Masters* o com a *Slaves* (Amos o esclaus). La diferència que hi ha entre aquests dos tipus és que un *Slave* només es pot connectar a un *Master* i a ningú més, en canvi un *Master* es pot connectar a diversos *Slaves* o permetre

que ells es connectin i rebre i sol·licitar informació de tots ells, arbitrant les transferències d'informació.

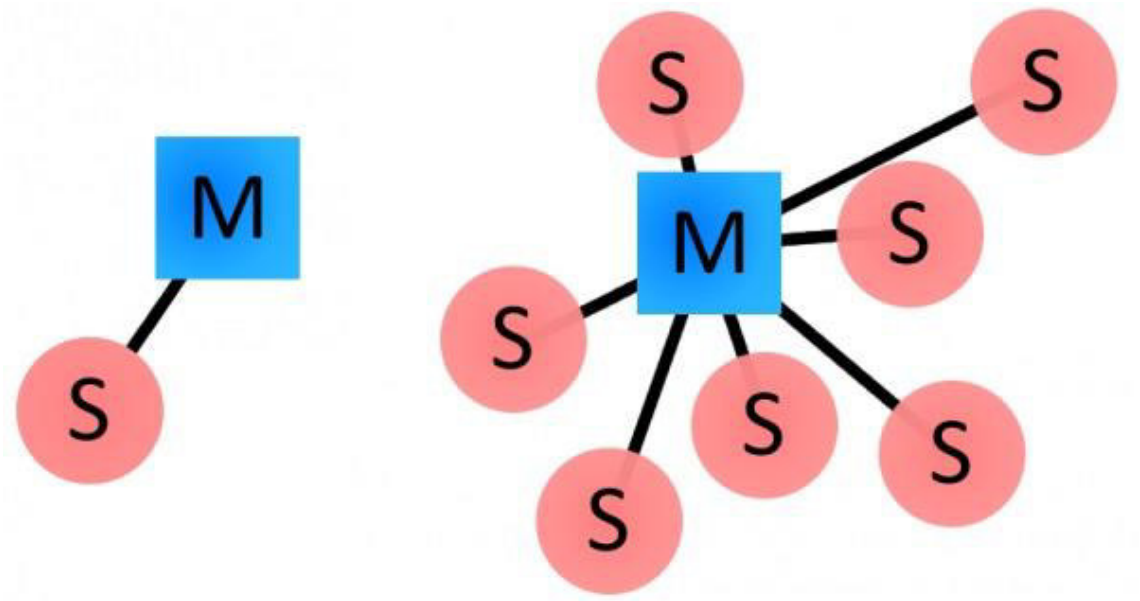



*Figura 40. Esquema Master i Slave.*

Per entendre com funciona la connexió Bluetooth entre dos dispositius, cal esmentar que els dos dispositius es vinculen entre ells i quan això passa s'inicia un procés en el que ells s'identifiquen per un nom i una direcció interna (tots els dispositius que es poden connectar mitjançant Bluetooth tenen una direcció pròpia) i se sol·liciten una clau PIN per autoritzar la connexió. Si l'emparellament es realitza exitosament, ambdós nodes acostumen a guardar la identificació de l'altre i quan es troben a prop es tornen a vincular sense necessitat d'intervenció manual.

Respecte als mòduls esmentats anteriorment, caldria comentar les diferències entre ells per així justificar l'elecció del mòdul escollit per al projecte.

La principal diferència, la qual ha estat determinant en l'elecció del mòdul, és el comportament de cada mòdul. El mòdul HC-06 actua sempre com a *Slave*, permetent que l'altre dispositiu actuï com *Master*; en canvi, el mòdul HC-05 pot actuar tan com a *Master* com a *Slave*, obligant així a l'usuari a haver-lo de configurar primer per fer que actuï d'una manera o de l'altra.

Una altra diferència que hi ha entre tots dos mòduls és que el model HC-05 admet un major nombre d'ordres de configuració, dotant així aquest mòdul d'una complexitat més elevada; en canvi, el model HC-06 admet un menor nombre d'ordres de configuració, fent que aquest model sigui més simple tan a l'hora de programar-lo com d'utilitzar-lo.

Finalment, s'ha optat per l'elecció del mòdul HC-06 degut a la seva senzillesa i al seu fàcil funcionament. Aquesta elecció es deu a que, degut a la seva configuració, el mòdul HC-06 permetrà que el dispositiu mòbil sempre actuï com a *Master*, ja que el mòdul sempre actuarà com a *Slave*. A més a més, degut a que el tipus de projecte no requereix una elevada complexitat en el tipus de connexió Bluetooth, és més viable utilitzar el mòdul HC-06 ja que al ser més simple implicarà menys problemàtic a la hora de configurar-lo per fer-lo funcionar.

Aquest mòdul està format per 6 pins dels quals només s'utilitzaran 4, i són els següents:

- Pin d'alimentació, senyalat com a 5V.
- Pin de terra, senyalat com a GND.
- Pin transmissor de dades, senyalat com a Tx.
- Pin receptor de dades, senyalat com a Rx.

Com ve indiquen els pins d'alimentació i de terra, aquests s'han de connectar als pins de la placa Arduino UNO corresponents a l'alimentació de 5 volts (5V) i a terra (GND).

Aquests quatre pins esmentats es connectaran a la placa Arduino UNO de la següent manera (Figura 41):

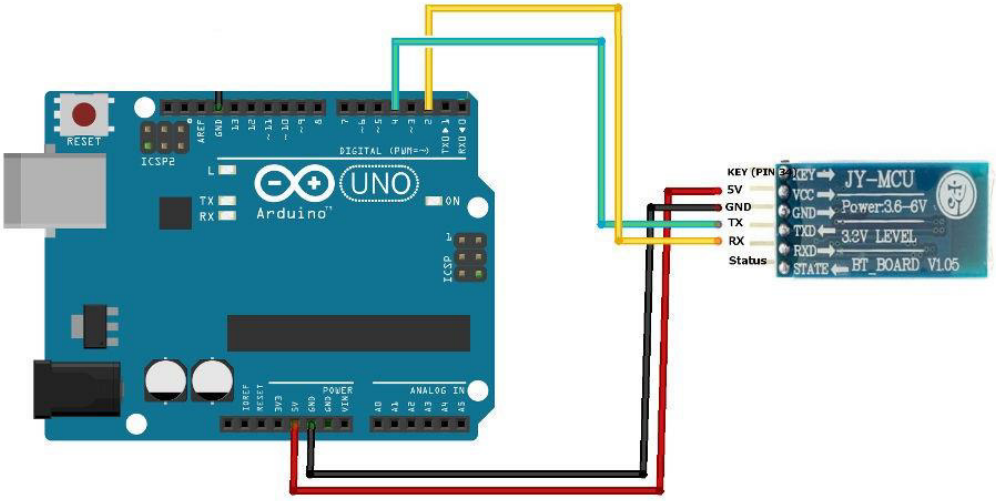



*Figura 41. Esquema connexió mòdul HC-06 amb placa Arduino UNO.*

La placa Arduino UNO també compta amb dos pins Rx i Tx que actuen com a receptors i transmissors de les dades i la informació entre la placa i qualsevol altre component que s'hi connecti; però degut a que aquests pins realitzen aquest procés amb el port sèrie (connexió amb l'ordinador), s'ha optat per connectar-los als pins digitals 2 i 4 de la placa. Al realitzar les connexions d'aquesta manera, és a dir, tenint en compte que aquests dos pins no estan configurats com a transmissors i receptors de dades, caldrà programar la placa de tal manera que siguin capaços de realitzar aquesta funció. Aquest procediment s'explicarà detalladament al següent apartat.

## 4.3. Informació transmesa entre el mòdul i la placa

Un cop entès com funciona la transmissió d'informació via Bluetooth, és necessari saber quin tipus d'informació es transmetrà entre el mòdul Bluetooth escollit i la placa Arduino UNO.

Tenint en compte que el mòdul Bluetooth s'emprarà per connectar-se amb la placa des d'un app per dispositius mòbils, i des de la qual es modificaran els paràmetres del control; la primera idea que es contempla és que l'única informació que es transmetrà entre la placa i el mòdul (o l'app) serà de tipus

numèrica, ja sigui entera o decimal. Tal com s'acaba de comentar, a través de l'app es realitzarà una modificació dels paràmetres del control, els quals són els següents: la histèresi, l'acció proporcional (Kp), l'acció integral (Ki) i l'acció derivativa (Kd).

Però, per poder realitzar aquestes modificacions, caldrà que el codi de la placa compti amb funcions que permetin realitzar aquestes accions; i per activar o accedir a aquestes funcions serà necessari que l'app enviï algun tipus d'informació, la qual pot ser numèrica o de text. Per tant, ja no només es transmetrà informació numèrica entre la placa i el mòdul, sinó que també entra en joc informació en format text.

A l'apartat on es parla de l'app s'especificarà quin tipus d'informació es transmet en cada cas concret.

Tot i haver concretat que el tipus d'informació que es transmet és de tipus numèric o de text, cal tenir en compte que aquestes dades enviades ho fan en un format molt concret.

Quan s'envien dades de l'app a la placa, les dades s'envien caràcter a caràcter, és a dir, no s'envia tota la informació de cop i que es pugui guardar en una sola variable, sinó que s'envia cada caràcter del missatge, ja sigui text o numèric, individualment i de forma consecutiva. Tots aquests caràcters enviats ho fan en format de codi ASCII, el qual és diferent als valors decimals amb els que s'acostuma a treballar. El codi ASCII és el Codi Estatunidenc Estàndard per a l'Intercanvi d'Informació (sigles de *American Standard Code for Information Interchange*) i correspon a un joc de caràcters que assigna valors numèrics a les lletres, xifres i signes de puntuació.

Aquest mètode de transmissió de dades pot dificultar l'ús de les dades al rebre-les, però per poder treballar amb les dades rebudes per la placa s'han creat funcions que permetin agrupar el missatge sencer i poder guardar-lo en una sola variable. El codi que permet realitzar aquesta tasca es troba a la següent figura (Figura 42):

```
String GetLine(){
  String S = "";
  while (BT1.available()){
    var = BT1.read();
    S = S + var;
    delay(25);
  }
  return(S);
}
```

*Figura 42. Codi de recopilació de caràcters en un sol text.*

Aquesta funció s'encarrega de llegir la informació que rep de l'app i mentre hi hagi informació rebuda, aquesta la va guardant tota en una variable de text que després permetrà treballar amb el missatge complet.

Degut a que algunes dades enviades haurien de ser numèriques, serà necessari transformar aquestes dades de tipus text a un valor numèric. Aquest procediment també es realitzarà mitjançant una funció escrita a la placa, la qual es pot veure a la Figura 43 i a la Figura 44:

```
float StrToFloat(String Texto){
  int j=0;
  float numf=0.0;
  while ((Texto[j] != '.') && (Texto[j] != '\n')){
    numf = 10*numf + (Texto[j]-48);
    j=j+1;
  }
  if (Texto[j] == '.'){
    float peso=0.1;
    j=j+1;
    while (Texto[j] != '\n'){
      numf = numf + ((Texto[j]-48)*peso);
      peso = peso*0.1;
      j=j+1;
    }
  }
  return numf;
}
```

*Figura 43. Codi per convertir de text a Float.*

```
double StrToDouble(String Texto){
  int k=0;
  double numd=0.0;
  while ((Texto[k] != '.') && (Texto[k] != '\n')){
    numd = 10*numd + (Texto[k]-48);
    k=k+1;
  }
  if (Texto[k] == '.'){
    float peso=0.1;
    k=k+1;
    while (Texto[k] != '\n'){
      numd = numd + ((Texto[k]-48)*peso);
      peso = peso*0.1;
      k=k+1;
    }
  }
  return numd;
}
```

*Figura 44.* *Codi per convertir de text a Double.*

Aquestes línies de codi s'encarreguen d'anar llegint el text corresponent al missatge rebut des del mòdul i anar-lo convertint en un nombre real pas a pas. Comença mirant el primer caràcter del text i si no és ni un punt (corresponent a la coma decimal) ni el delimitador (\n) el suma a la variable numèrica que inicialment és 0. Després llegeix el següent caràcter i, si es compleix el mateix que amb el primer, aquest també el suma a la variable, la qual ha estat multiplicada per 10 (així es defineixen les centenes, desenes i unitats). En cas que llegeixi un punt, salta al següent caràcter i, mentre aquest no sigui el final del missatge, se sumarà a la variable però multiplicat per 0,1.

En el cas que les dades enviades de l'app a la placa corresponguin a un valor numèric d'una sola xifra, la programació de l'app permet enviar-les com un valor numèric. Aquest fet implica que el format d'aquestes dades enviades serà diferent al que s'ha explicat anteriorment. Al tractar-se de valors numèrics enviats com a tals, es podran rebre com a valors numèrics també i no suposarà cap problema treballar amb ells; però en el cas de voler visualitzar-los, aquests es mostraran en codi ASCII, ja que com s'ha comentat anteriorment els caràcters s'envien en codi ASCII. Si es volguessin visualitzar correctament, seria necessari

buscar l'equivalència entre el codi ASCII i el decimal per poder aplicar aquesta relació al caràcter rebut.

## 4.4. Programa

Tal i com s'ha comentat prèviament, el tipus de connexió realitzat entre el mòdul HC-06 i la placa Arduino UNO requereix d'una programació que permeti enviar i rebre dades entre la placa i el mòdul Bluetooth tenint en compte que els pins de la placa emprats no estan configurats per realitzar aquesta funció.

Per poder utilitzar el mòdul Bluetooth d'aquesta manera, és necessari importar una llibreria que habiliti la comunicació sèrie amb altres pins que no siguin els Rx i Tx de la placa. Una llibreria que pot dur a terme aquesta funció és la llibreria *Software Serial*.

Aquesta llibreria va inclosa de sèrie al IDE d'Arduino, el programa que s'utilitza per programar la placa. Per importar la llibreria al programa s'utilitza les línies de codi #*include* juntament amb el nom de la llibreria; i després s'ha de crear un nou objecte sèrie, en aquest cas anomenat BT1, connectat als pins 2 i 4.

```
#include <SoftwareSerial.h>
SoftwareSerial BT1(4,2);
```

*Figura 45. Codi per incloure la llibreria de comunicació sèrie.*

Sabent que tant la placa com el mòdul Bluetooth poden enviar i rebre informació, caldrà tenir en compte que segons el procediment que es vulgui seguir s'hauran d'aplicar programes o línies de codi diferents, tot i que seguiran un mateix patró.

Per realitzar la comunicació entre la placa i el mòdul HC-06 de tal manera que el mòdul sigui el transmissor (Tx) i la placa el receptor (Rx), cal escriure un fragment de codi o programa que habiliti la comunicació sèrie a través de l'objecte BT1 i que la comunicació sèrie pròpia entre la placa i l'ordinador s'encarregui d'escriure a la pantalla les dades enviades.

```
if (BT1.available()) {     //comprovar que el HC-06 puede enviar datos
  var = BT1.read();       //leer los datos enviados desde el HC-06
  var=var+48;
  Serial.write(var);     //mostrar en pantalla los datos enviados
```

*Figura 46. Codi lectura de dades i passar de codi ASCII a decimal.*

Tal i com es veu a la imatge del codi i dels seus comentaris, aquest és el mètode que s'ha emprat per enviar dades des del mòdul HC-06 cap a la placa. Cal destacar la línia de codi *var=var+48*, ja que aquesta línia de codi s'ha utilitzat per comprovar que les dades envaïdes des del mòdul i les que es mostraven a la pantalla de l'ordinador eren les mateixes. S'ha utilitzat aquest codi degut a que les dades que envia el mòdul es troben en codi ASCII, el qual és diferent al que es mostra per la pantalla de l'ordinador.

El perquè d'aplicar una suma de 48 a la variable llegida es troba en que les proves per verificar el correcte funcionament es van realitzar enviant valors de 0 i 1 des del mòdul a la placa. Seguint una taula d'equivalències entre el codi ASCII i els caràcters amb els que es treballen, els números 0 i 1 corresponen a 48 i 49 en codi ASCII.


*Figura 47. Taula codi ASCII.*

A la taula es veuen totes les equivalències entre els caràcters i el codi ASCII corresponent. A partir de la informació d'aquesta taula s'ha determinat que per poder veure els valors 0 i 1 a la pantalla de l'ordinador era necessari sumar 48 al valor enviat.

Tal i com es poden enviar dades des del mòdul a la placa, també es poden enviar dades des de la placa al mòdul Bluetooth seguint el mateix procés però de manera inversa. En aquest cas primer no cal comprovar si es poden enviar dades des de la placa al mòdul HC-06 sinó que directament s'indica que es mostrin al mòdul (o a la pantalla que està connectada al mòdul). El codi corresponent a aquest procés és el que es mostra a la Figura 48.

```
//########## enviar datos al módulo ##################
    String stringOne = String(Input, 2);
    BT1.print('#' + stringOne + '\n');
```

*Figura 48. Enviar dades de la placa al mòdul.*

Degut a que la informació que es vol enviar al mòdul es per realitzar una gràfica i que aquesta només és capaç de llegir missatges de text com a informació rebuda, és necessari transformar el valor numèric enviat a una variable de tipus *String* (text) per poder-la llegir des del mòdul.

A l'hora d'enviar aquesta informació, s'ha afegit uns delimitadors, per al davant i el darrere del valor numèric, per poder diferenciar entre els valors rebuts quan aquests són llegits pel mòdul i afegits a la gràfica. Aquest procés s'explicarà amb més detall a l'apartat de comunicació entre l'app i la placa.

# 5. L'app

## 5.1. Software de desenvolupament de l'app

Un cop establert el mètode que s'ha emprat per realitzar la connexió sense fils que permetrà controlar la plataforma, també cal determinar quina plataforma o software de desenvolupament d'aplicacions s'ha utilitzat per crear l'aplicació que permetrà realitzar aquest control remot.

Una plataforma o software de desenvolupament d'aplicacions consisteix en un programa o lloc web que permet crear aplicacions per a dispositius mòbils a partir de la creació de la seva estructura o interfície i la programació del seu funcionament.

La primera opció que es va plantejar com a software de desenvolupament d'aplicacions va ser Unity, un entorn enfocat sobretot al disseny de jocs per a ordinador i telèfons mòbils. Es va pensar en aquesta opció degut a que ja s'havia treballat anteriorment i es tenien uns coneixements bàsics d'aquesta plataforma. Però a l'hora d'establir la comunicació Bluetooth entre el mòdul i la placa es van produir molts problemes que van dificultar l'avenç del projecte, motiu pel qual es va optar per un canvi de software de desenvolupament.

L'opció que es va triar definitivament com a software de desenvolupament d'aplicacions va ser el software MIT App Inventor 2. Aquest software és un entorn integrat de desenvolupament que permet crear aplicacions per a dispositius mòbils que continguin el sistema operatiu Android. Aquest consta d'un ús molt visual i intuïtiu i a partir d'unes eines bàsiques, l'usuari va enllaçant un conjunt de blocs per crear una aplicació.

Es tracta d'una eina destacada per la seva simplicitat, és a dir, el seu ús és molt senzill però alhora està bastant limitat degut a aquesta senzillesa. Tot i així, permet cobrir un gran nombre de necessitats bàsiques en un dispositiu mòbil. D'aquest software cal destacar que és una eina *cloud-based*, és a dir, que tots els projectes realitzats es guarden automàticament al servidor de l'eina, el qual

pertany al domini de Google, i s'hi pot accedir des de qualsevol dispositiu que tingui connexió a internet. És una eina que funciona a través del núvol.

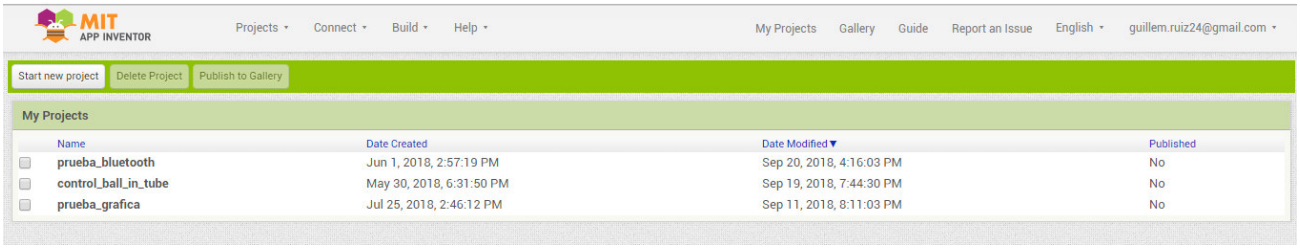



*Figura 49. Pàgina inici MIT App Inventor 2.*

Aquesta eina està formada per dues parts o pantalles diferents, una correspon al disseny de l'interfície de l'app i l'altre a la programació del funcionament d'aquesta.

La primera part, corresponent a la part gràfica i estètica, consisteix en la selecció dels components que formaran part de l'app i la ubicació d'aquests al sector de la pantalla de l'eina que simula la pantalla del dispositiu mòbil. Els components que es poden incloure a l'app poden ser de diferents tipus, diferenciant-se en la utilitat que poden aportar al disseny de l'app.

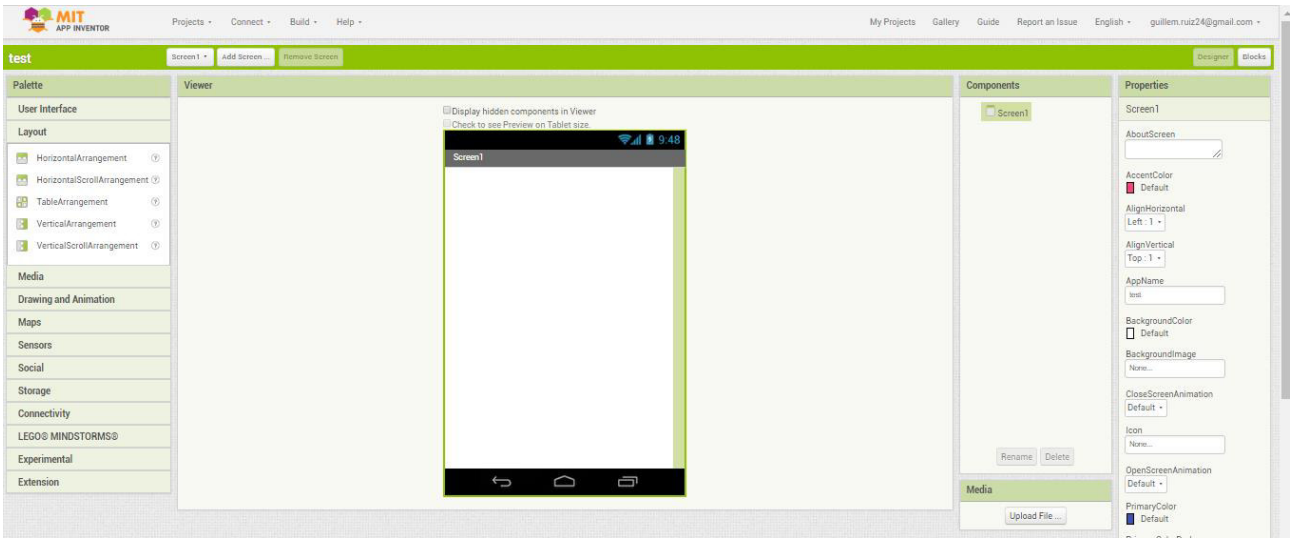



*Figura 50. Pantalla creació de la interfície d'una app.*

Com es pot veure a la Figura 50, a l'esquerra de la pantalla es troben tots els tipus de components que es poden incloure a l'app, a la part central apareix la representació de la pantalla del dispositiu mòbil i a l'esquerra apareixen les característiques que es poden editar de cada component.

Els diferents tipus de components estan estructurats de la següent forma:

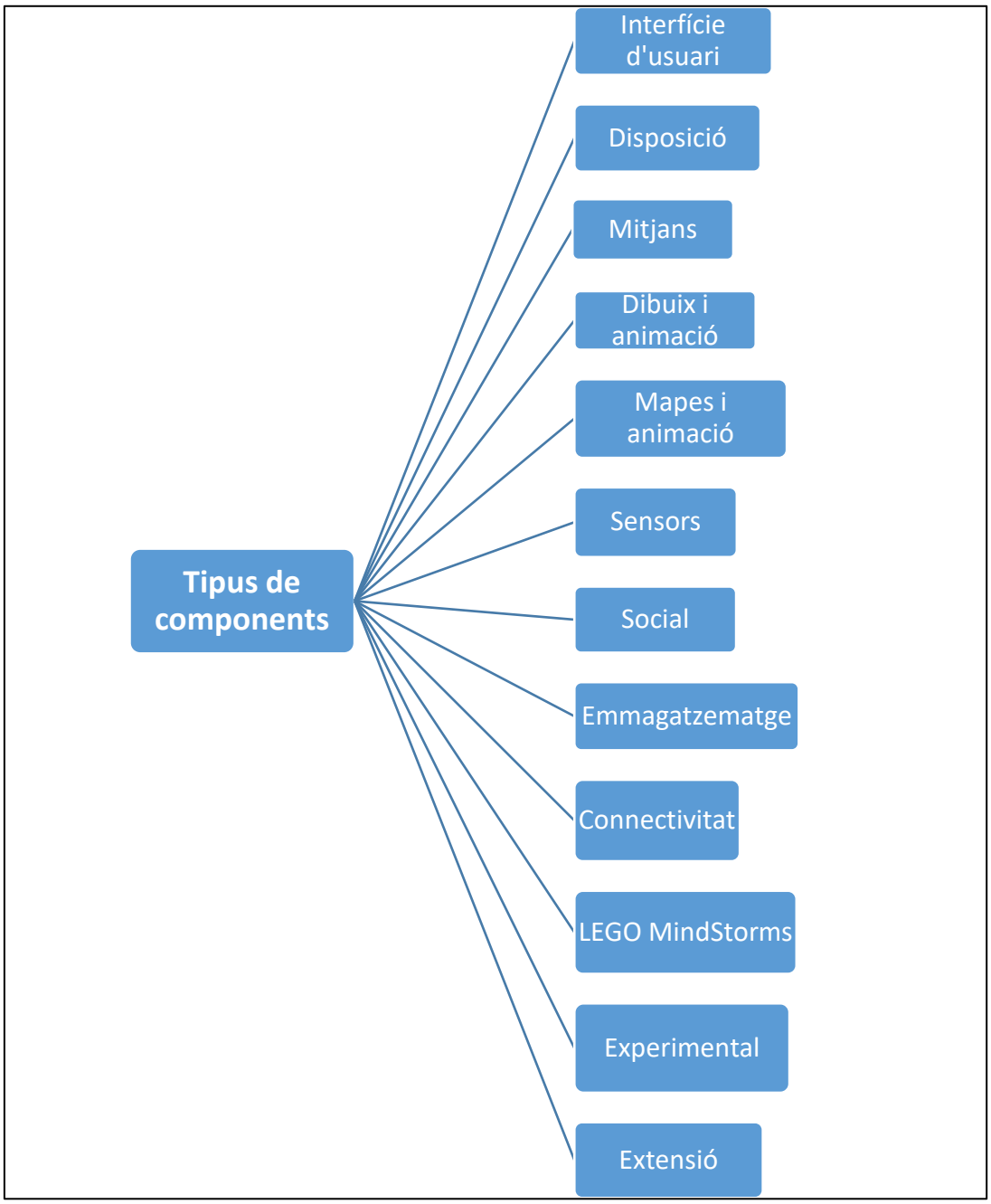



*Figura 51. Esquema components per incloure en una app.*

Aquest esquema (Figura 51) representa les diferents categories de components que es poden incloure al crear una app. Dins de cada categoria hi ha diferents components que pertanyen a aquella categoria degut a la funció que executen dins de l'app. Per exemple, els components que formen part de la categoria Interfície d'usuari són components que permeten que l'usuari interactuï amb l'app, com ho serien els botons, els camps de text, les barres desplaçables, les imatges, caselles de verificació, selectors de llista, etc.

En canvi, la categoria de Disposició està formada per elements que permeten estructurar els components de l'app de diferents maneres, ja sigui en disposicions horitzontals, verticals o ambdues amb un *scroll*.

Com es pot veure, el nom de cada categoria indica de forma bastant clara la funció dels components que en formen part. Més endavant, s'especificaran els components utilitzats per crear l'app d'aquest projecte i a quina categoria pertany cada un d'ells.

Un cop coneguda la primera part de l'eina, cal entrar en la segona part o pantalla que s'utilitza per crear una app. Aquesta segona part és la que s'empra per programar el funcionament de l'app. A aquesta pantalla es pot accedir al clicar el botó de la part superior dreta de la pantalla on posa Blocks.

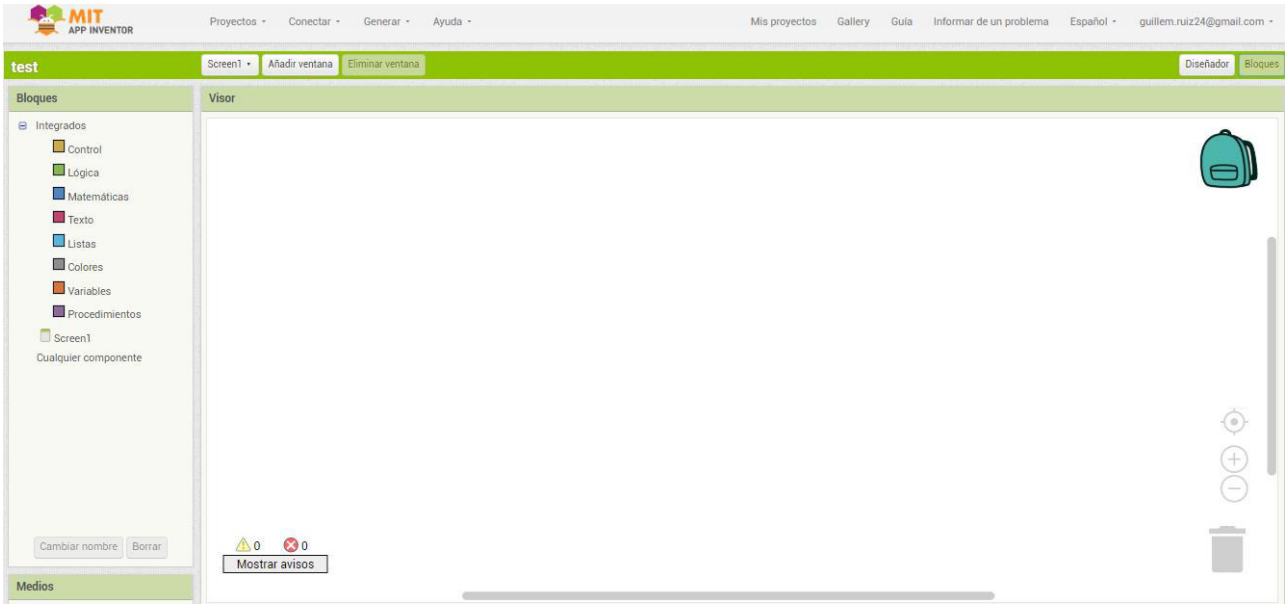



*Figura 52. Pantalla creació de blocs per programar una app.*

Aquesta pantalla està formada per un espai en blanc a la part central, on es crearan i ubicaran els blocs de cada funció; i la part dreta on es poden seleccionar les diferents funcions a realitzar, ja sigui de les que existeixen de manera predeterminada (control, text, lògica, variables, etc.) o de les que formen part dels components que s'utilitzin a l'app que es crearà.

Amb els blocs corresponents a les funcions es pot definir la funcionalitat de l'aplicació i de totes les seves parts, és a dir, el que pot passar el clicar un botó, canvis de color quan passa alguna acció predeterminada, actualització de valors, canvis de característiques, etc.

Mitjançant la unió de diferents blocs, establint condicions i altres característiques, la funcionalitat de l'app queda definida; i els blocs, tot i estar units, queden ben diferenciats entre ells ja que segons la funció que duguin a terme queden definits amb colors diferents.

Com qualsevol altre projecte, els canvis que es van realitzant en la creació de l'app s'han d'anar guardant per evitar perdre'ls. Per guardar un projecte, es realitza a través de la pestanya superior de *Projects*, la qual es desplega al fer-hi un clic, i un cop desplegada es pot guardar el projecte.

Al tractar-se d'una eina de creació d'apps per a dispositius mòbils, una característica important amb la que compta és el mètode per executar l'app en un dispositiu mòbil. Per ser més exactes, no compta amb un sol mètode, sinó que ho pot fer a través de diferents vies. La primera és la generació d'un arxiu amb extensió "*.apk"*. Aquest arxiu és el que permet instal·lar l'aplicació al dispositiu, és a dir, seria com un contenidor de l'aplicació. Si s'opta per utilitzar aquest mètode, aquest arxiu s'ha d'enviar al dispositiu i després instal·lar-lo en aquest mateix. En relació a aquest mètode, existeix una petita variant d'aquest, el qual genera un codi QR que, un cop escanejat o llegit, permet descarregar l'app.

Un altre mètode, el qual difereix dels dos anteriors, que es pot emprar és realitzant una connexió entre l'ordinador i el dispositiu mòbil. Per realitzar aquesta connexió primer és necessari instal·lar l'app de MIT App Inventor 2 al

dispositiu mòbil. Un cop instal·lada l'app al dispositiu, es realitzarà la connexió entre l'ordinador i el dispositiu. Per fer aquesta connexió es farà clic a la pestanya superior *Connect*, i al desplegar-se es clicarà l'opció *AI Companion*.


*Figura 53. Connectar PC amb el mòbil.*

Un cop fet aquest pas, apareixerà un codi QR que s'haurà d'escanejar amb l'app de MIT App Inventor 2 per poder realitzar aquesta connexió. Un cop escanejat el codi QR, a la pantalla de l'app es mostrarà el projecte que s'estigui editant a l'ordinador.

Sobre aquest procés, és important que l'ordinador i el dispositiu estiguin connectats a la mateixa xarxa d'internet.

## 5.2. Estructura/interfície de l'app

Un cop conegudes les parts de l'eina que s'utilitzarà, ja es pot parlar de l'interfície de l'app que s'ha dissenyat i els components que la formen.

Per començar, és necessari saber el què es vol veure en aquesta app, és a dir, quines característiques, paràmetres, dades,... de la plataforma o projecte es volen visualitzar mentre es realitza el control d'aquest.

Degut a que la funció principal de l'app és la de controlar la plataforma i el seu funcionament, serà necessari visualitzar els paràmetres que es poden modificar, és a dir, la histèresi, el *setpoint* i els valors de les accions del control PID: Kp, Ki i Kd. Tenint en compte que també serà possible definir el control que s'executa

sobre la plataforma (amb histèresi o PID), una part de la interfície estarà dedicada a la selecció d'aquest control.

Una altra part important que ha de contenir l'app és la capacitat de connectar-se al mòdul Bluetooth, ja que l'objectiu és que mitjançant l'app es digui a terme un control a distància de la plataforma. Per tant, la interfície de l'aplicació ha de mostrar algun component que permeti realitzar aquesta connexió.

Per últim, tot i que el funcionament de la plataforma i el comportament de la pilota de ping-pong es pot veure en directe observant la plataforma, afegir una gràfica mostrant numèricament el valor de les dades recollides pel sensor ultrasònic aportaria un valor afegit a l'app, ja que mostraria amb un valor exacte a quina distància es troba la pilota.

Un cop definides les parts que es volen mostrar a la interfície de l'app, caldrà escollir quins components s'empraran i la seva distribució a la pantalla que els mostrarà.

Per començar, s'ubicarà primer la gràfica, ja que és un component que pot ocupar bastant espai si es vol veure una gràfica prou gran. La gràfica és un component anomenat *Canvas*, el qual serveix per mostrar valors, variables, etc. en una gràfica, i forma part de la categoria *Drawing and Animation*. En el disseny de l'app en desenvolupament, s'ha decidit ubicar la gràfica a la part dreta de la pantalla, ocupant la meitat de la pantalla per poder veure la gràfica que es dibuixi a una bona mida. A la següent figura (Figura 54) es pot veure la seva ubicació remarcada en vermell.

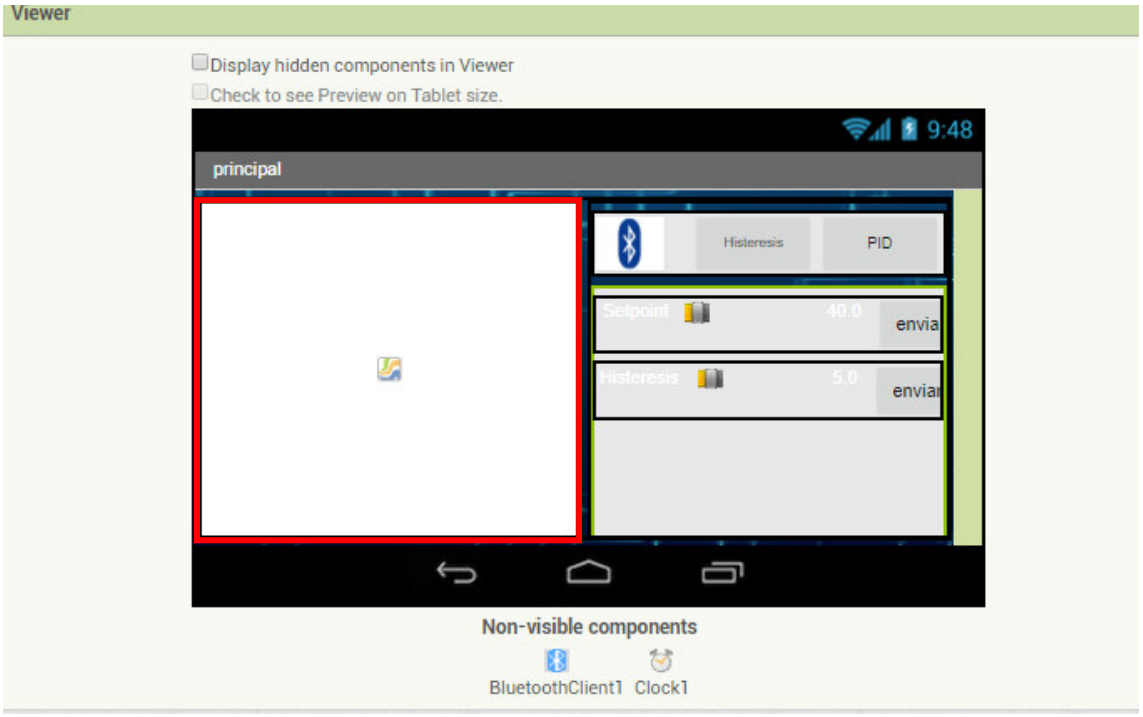



*Figura 54. Ubicació de la gràfica a la pantalla de l'app.*

Un cop decidit l'espai que ocupa l'element més gran de l'app, es poden anar introduint els altres components de l'app a l'espai que queda disponible.

Una característica important d'aquesta eina, és que si es volen distribuir els elements de forma ordenada, ja sigui en una disposició vertical o horitzontal, és necessari introduir els components de la categoria *Disposició* (*Layout*); ja que sense aquets tipus d'ordre l'eina ubica, per defecte, els components en disposició vertical, és a dir, cada element nou introduït quedaria ubicat just a sota de l'últim element existent.

Per evitar aquest problema de distribució, s'introdueix un *HorizonalArrengement*, dins del qual s'ubicarà la gràfica i tots els altres components, per poder situar la gràfica al costat esquerre i la resta de components al costat dret.

Per situar la resta de components a la dreta de la gràfica, s'introduirà un *Vertical Arrengement* que permetrà col·locar els components en disposició vertical.

El següent component a ubicar serà el que permeti establir la connexió Bluetooth amb el mòdul HC-06. Aquest anirà ubicat a la part superior, a la dreta de la gràfica, i el component serà un selector llista, que pertany a la categoria de

*Interfície d'usuari*. S'ha escollit aquest component perquè mostra els dispositius Bluetooth que estan connectats al dispositiu mòbil i permet escollir-ne un. Per identificar de forma clara aquest element, s'ha li ha proporcionat una imatge de fons representant el logotip del Bluetooth, tal i com es veu a la Figura 55 marcat amb un quadre vermell.

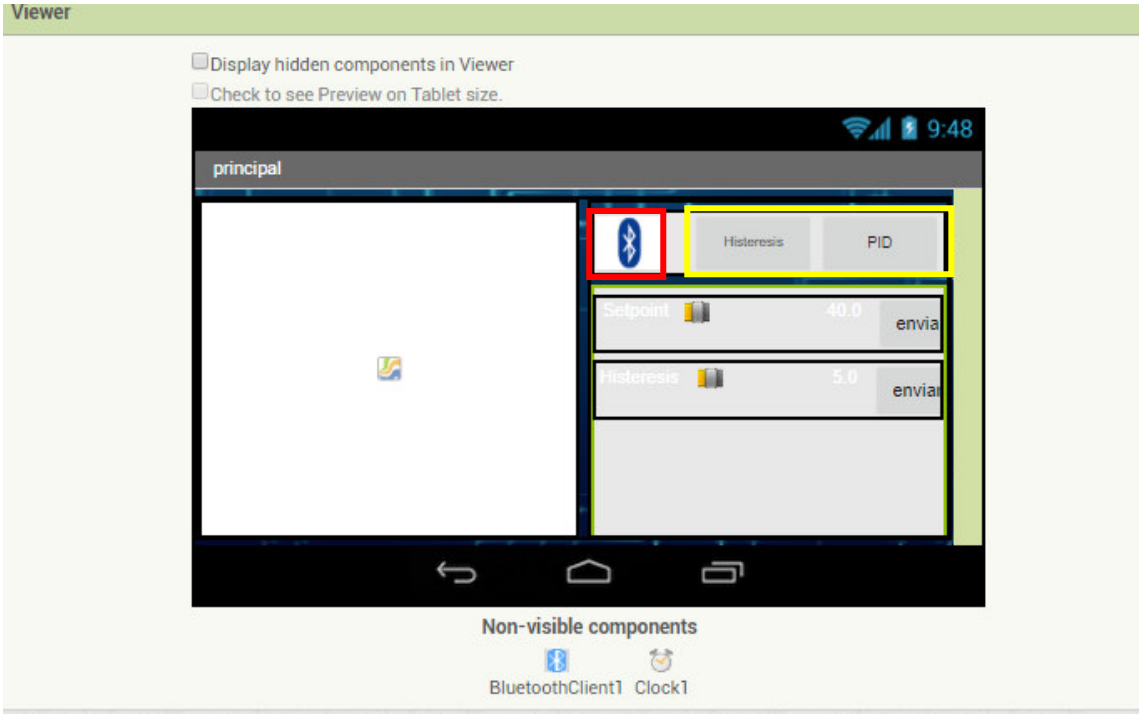



*Figura 55. Ubicació del selector de llista Bluetooth i botons de control a la pantalla de l'app.*

Seguint aquest ordre d'ús durant el funcionament de la plataforma, els següent elements escollits són dos botons que permeten seleccionar el control que s'aplica . Aquests dos botons es troben dins d'un *HorizontalArrangement* que permet que estiguin un al costat de l'altre. Cada botó té un text a dins fent referència a quina acció estan relacionats: control amb histèresi o control PID. A la Figura 55, es poden veure els dos botons marcats amb un quadre groc.

Els components que queden per situar a la interfície són els paràmetres a modificar: la histèresi, el *setpoint*, Kp, Ki i Kd. Al ser cinc paràmetres a modificar, l'espai queda bastant limitat i per evitar posar components amb una mida massa petita, s'ha optat per una opció més eficient i complexa. Es tracta de crear els paràmetres de cada acció de modificació i fer que només es mostrin quan aquella

acció s'està executant, és a dir, quan estigui seleccionat el control amb histèresi només es mostraran els components que permetin modificar la histèresi; i de la mateixa manera per al control PID. Els paràmetres per realitzar el canvi de *setpoint* es mostraran sempre ja que és un paràmetre que es pot modificar en ambdós controls. Per fer que un element sigui o no visible només cal seleccionar o deseleccionar la casella *Visible* d'aquell component.

Per dur a terme la modificació dels paràmetres, a diferència dels altres components, s'utilitzarà més d'un sol component per a una única acció. Cada grup de components encarregat de modificar un paràmetre estarà format per un text indicant el paràmetre, una barra desplaçable que modifiqui el valor del paràmetre, un altre text que mostri el valor del paràmetre que adopta la posició de la barra desplaçable i un botó per enviar el valor a la placa; tots en aquest mateix ordre. Tots aquests components es troben a la categoria de *Interfície d'usuari*. En les següents dues imatges es podrà veure el grup de components que modifiquen la histèresi i els paràmetres del control PID, mantenint l'ordre que s'acaba d'esmentar.

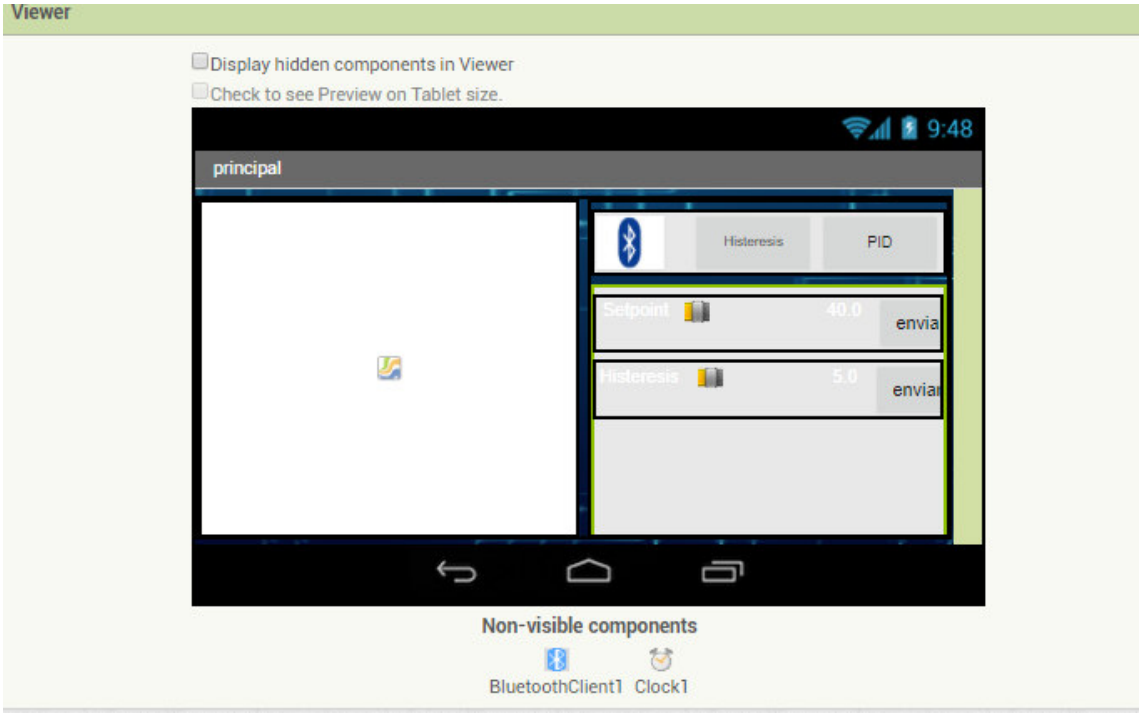



*Figura 56. Ubicació dels components per modificar la histèresi.*

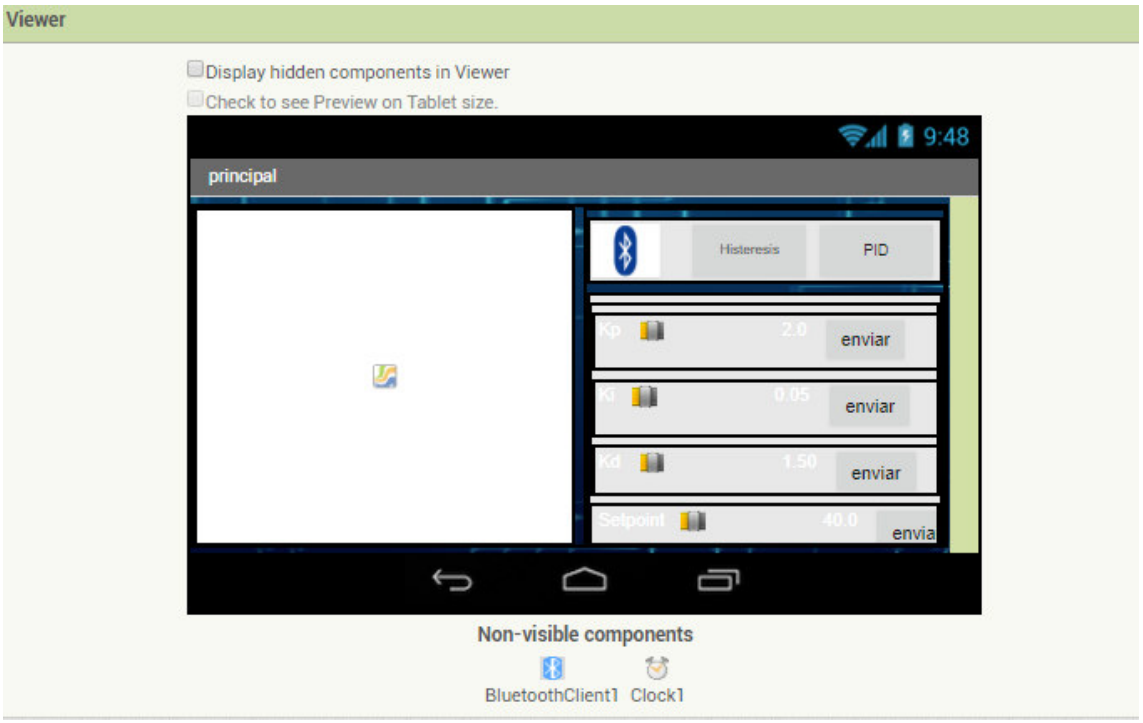



*Figura 57. Ubicació dels components per modificar el setpoint.*

Un cop parlat de tots els elements visibles de la interfície, cal comentar que també existeixen components no-visibles que cal afegir a l'app. Els elements no visibles són aquells amb les que no interactua l'usuari però són necessaris perquè certes funcions es puguin dur a terme. Exemples d'aquests components són el *BluetoothClient* o el *Clock*. El primer d'aquests dos és el que defineix l'app com a *Slave* en la comunicació Bluetooth. El segon permet realitzar comptes de temps per poder dibuixar o mostrar valors a la gràfica. Tal com s'esmenta al principi del paràgraf, aquests dos elements no apareixeran a la pantalla de l'app, però a l'hora de dissenyar aquesta app, apareixen per sota de la simulació de la pantalla del mòbil, tal i com es veu a la Figura 58.

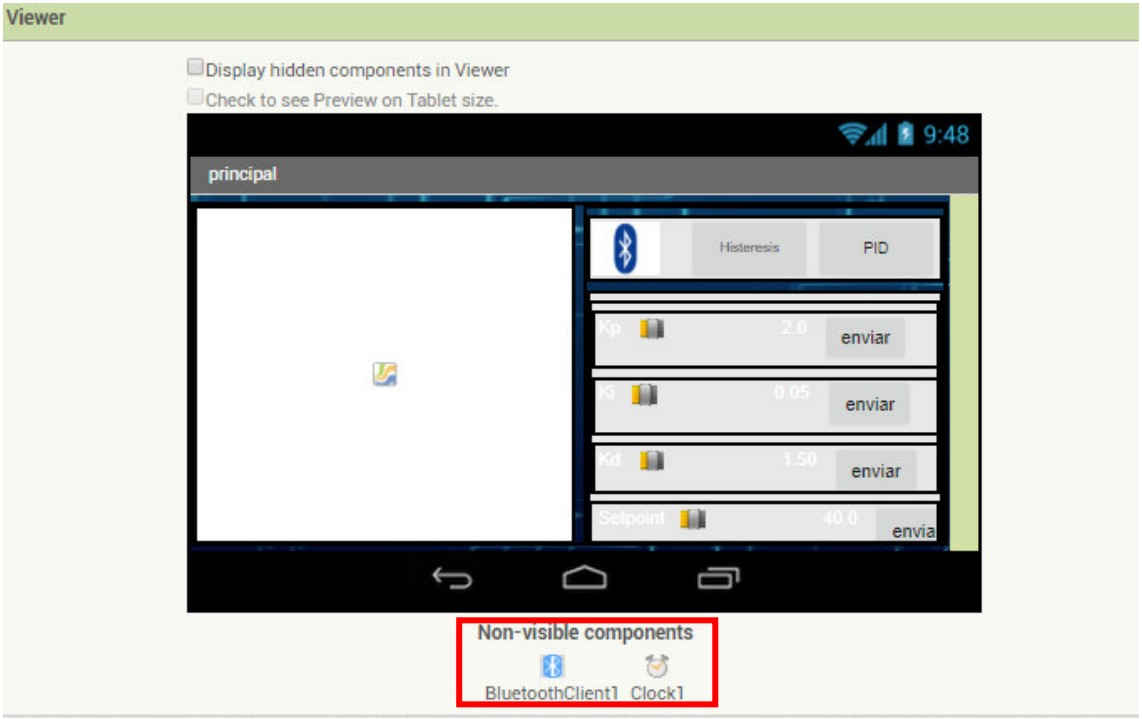



*Figura 58. Components no-visibles de l'app.*

Aquests dos components s'han de configurar de tal manera que les seves característiques siguin les mateixes que les que es defineixin al programa de la placa Arduino UNO.

Per afegir una certa complexitat i més ordre a l'interfície de l'app, s'ha creat una altra pantalla, la qual actuarà com a pantalla principal de l'app. Aquest pantalla estarà formada pel títol de l'app i del projecte, un botó per iniciar l'app i accedir a la pantalla de control i un botó per tancar l'app. Aquesta pantalla es trobarà en una orientació vertical, a diferència de la pantalla de control que es troba en orientació horitzontal.

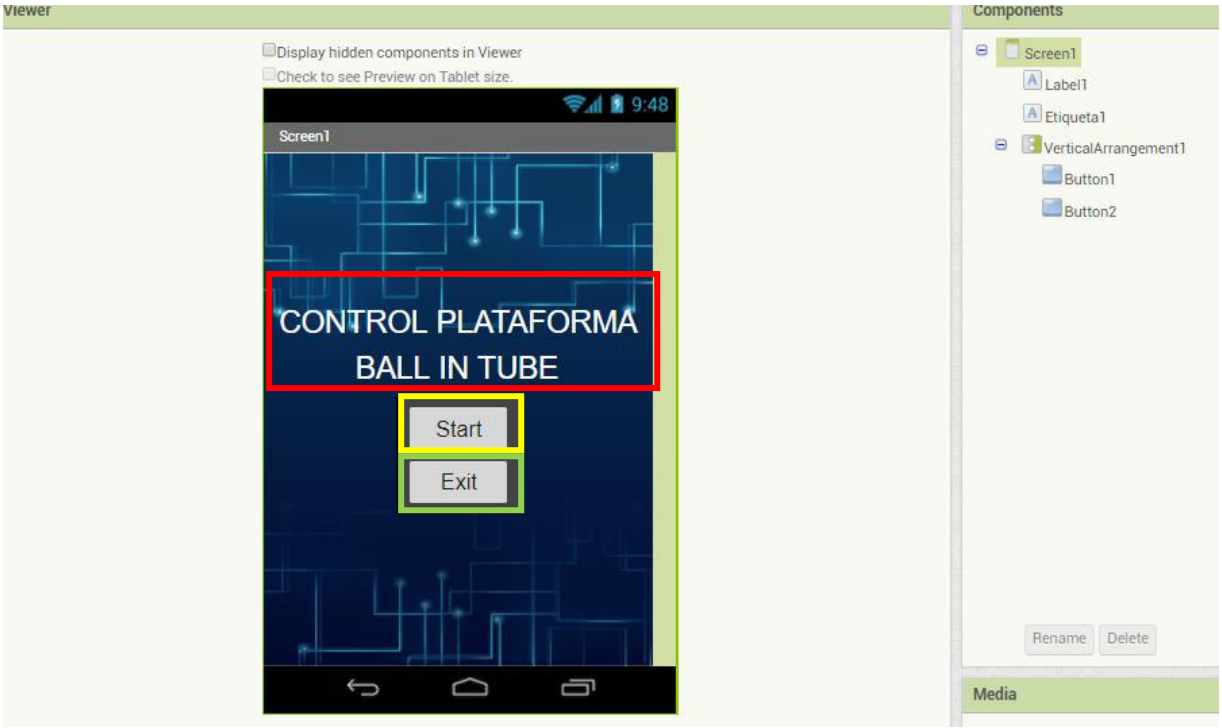



*Figura 59. Pantalla inicial de l'app i els seus components.*

A la imatge es poden veure els components esmentats anteriorment, el títol marcat en vermell, el botó d'inici marcat en groc i el botó per tancar l'app marcat en verd.

## 5.3. Funcionalitat i funcions de l'app

Un cop definida la interfície de l'app, és necessari establir la funcionalitat de tots els components, o almenys la d'aquells que realment duen a terme una acció quan l'usuari interactua amb ells. Quan es parla de funcionalitat es fa referència a l'acció o conseqüència que implica el fet d'interactuar o utilitzar un dels components integrats a l'app.

Per entendre la com funciona l'app des de la pantalla inicial, és necessari explicar la funcionalitat dels components de la primera pantalla. Aquesta primera pantalla és l'última part que s'esmenta a l'apartat anterior, i és la que conté el títol de l'app i dos botons: un d'inici i un de sortida. La funcionalitat d'aquests dos botons és la següent:

- **Botó d'inici:** la seva funció és la d'obrir la pantalla principal de l'app quan és clicat per l'usuari.
- **Botó de sortida:** la seva funció és la de tancar l'app quan l'usuari hi fa clic.

Un cop l'app obre la seva pantalla principal, es poden observar els components que s'han esmentat a l'apartat anterior; però no tots ja que alguns només es mostren quan es realitzen certes accions.

Per començar, el component que més destaca entre tots és la gràfica ja que ocupa la meitat de la pantalla del dispositiu. La gràfica és dels pocs elements amb els quals s'interactua. La seva funció o funcionalitat és la de mostrar el valor de la distància que hi ha entre la pilota i la part més baixa del tub; les dades de la qual les capta el sensor ultrasònic i es mostren en temps real. Per tant, la gràfica només és un component a observar.

Seguint el mateix ordre de components que s'ha utilitzat a l'apartat anterior, el següent element del qual parlar és el selector de llista que permetrà dur a terme la connexió Bluetooth entre el mòdul i la placa. La funció pròpia d'un selector de llista és la de mostrar una llista quan aquest component és clicat. En aquest cas, al fer clic a sobre d'aquest component es mostrarà una llista dels components Bluetooth que estan sincronitzats amb el dispositiu mòbil. Un pas previ important serà el de sincronitzar el mòdul Bluetooth amb el dispositiu mòbil.

Per sincronitzar el mòdul HC-06 amb el mòbil s'han de seguir uns senzills passos:

1. Activar el Bluetooth del telèfon i ubicar a prop el mòdul HC-06, el qual ha d'estar en marxa.
2. Buscar dispositius Bluetooth disponibles amb el mòbil.
3. Un cop apareix el mòdul HC-06 fer clic a sobre.
4. Per vincular ambdós dispositius, demanarà un codi o contrasenya per fer-ho efectiu.

5. Aquests mòduls poden tenir dues possibles contrasenyes de 4 números: 1234 o 0000. Per tant, caldrà provar quina de les dues és la del mòdul. En aquest cas és 1234.
6. Un cop introduïda la contrasenya, els dispositius queden vinculats.

Tornant al selector de llista, al aparèixer els dispositius vinculats haurà de seleccionar-se el mòdul HC-06 i llavors s'establirà la connexió entre el mòdul i la placa.

El següent component que es veu a la pantalla de l'app és el grup de botons que permeten seleccionar el control que s'executarà a la plataforma. Aquests botons no realitzen una sola funció sinó que al clicar-los realitzen varies accions:

- **Botó histèresi:** activa el control amb histèresi a la plataforma. Al activar aquest control, implica mostrar el paràmetre que es pot ajustar en aquest control, és a dir, la histèresi; per tant, quan es selecciona aquest control es mostren els components que poden canviar el valor de la histèresi. També fa que qualsevol component d'algun altre control deixi de ser visible i desactiva les accions dels altres dos botons si estan actives.
- **Botó PID:** aquest botó activa el control PID a la plataforma i desactiva les accions dels altres botons que estiguin executant-se, tal com s'ha esmentat a l'anterior botó. Al activar aquest botó es faran visibles els paràmetres Kp, Ki i Kd per així poder-los modificar, ja que són els paràmetres que intervenen en el control PID.

Tal com s'ha esmentat a l'apartat anterior, els grups de components de canvi d'histèresi i de paràmetres del control PID estan formats pels mateixos elements, per tant, tenen les mateixes funcions excepte que el destinatari de l'acció és diferent l'un de l'altre. Però per explicar la seva funcionalitat només caldrà posar-ne un d'exemple i no explicar-los tots dos per separat.

Aquests grups de components, fent recordatori de l'explicació de l'apartat anterior, estan formats per un text que descriu el paràmetre, un barra desplaçable, un text que mostra el valor de la posició de la barra desplaçable i

un botó per enviar el valor a la placa. Cada un d'aquests components té una funcionalitat pròpia:

- **Text del paràmetre:** com el seu nom indica, és un text que identifica el paràmetre que es modifica, fent referència a que tots els components situats a la seva dreta s'empren per modificar el paràmetre al qual fa referència.
- **Barra desplaçable:** la seva funció és modificar el valor del paràmetre segons la posició del dit de l'usuari sobre la barra. Cada paràmetre té un rang de valors predeterminat per realitzar el canvi de valor entre uns límits acotats.
- **Text del valor de la barra:** aquest component mostra numèricament el valor que adopta el paràmetre mentre l'usuari desplaça el dit per la barra desplaçable; podent així veure el valor exacte que va adoptant el paràmetre.
- **Botó enviar:** aquest component s'encarrega d'enviar el valor del paràmetre modificat a la placa per poder aplicar aquest nou valor a la plataforma. Cada paràmetre té el seu propi botó.

### 5.3.1. Programa/codi

Un cop explicada la funcionalitat dels components de l'app, és important entendre com es realitzen les funcions de cada components i per aconseguir-ho serà necessari entrar a la programació de l'app.

Per programar el funcionament de l'app, cal recordar que es fa a través d'un sistema format per blocs, on cada bloc correspon a una funció; i mitjançant la unió de diferents blocs es van creant les funcions desitjades fins a arribar a crear l'app.

Respecte a la creació dels blocs, cal esmentar que a la pantalla on es creen no és important l'ordre en que es disposen els grups de blocs corresponents a diferents funcions, és a dir, es pot posar primer un grup de blocs que executin les funcions d'un botó i després el grup de blocs de la connexió Bluetooth. Això

es pot fer degut a que aquest tipus de programació no segueix un ordre seqüencial.

Com es va comentar a l'explicació de l'eina MIT App Inventor 2, cada bloc té un color diferent en funció de l'acció que realitzi. Els blocs de color groc són blocs de control, els de color verd són blocs lògics o digitals (*true/false, and/or/not*), els de color blau marí fan referència a processos matemàtics, els de color magenta a accions amb textos, els de color blau cel a llistes, els de color gris a als colors que poden adoptar els components, els de color taronja a les variables que es creïn i els de color morat a processos creats a la pròpia app. També cal esmentar que cada component té els seus blocs, en els quals hi poden haver-hi d'alguns dels colors esmentats anteriorment, però sobretot es caracteritzen per tenir blocs de dos tonalitats de verd diferent. El ver clar fa referència a una característica del component i el verd fosc és per canviar la característica del component.

Un cop coneguts els blocs que es poden emprar, és moment d'indagar en els blocs de cada pantalla. De la mateixa manera que es creen pantalles per a l'app, hi haurà tantes pantalles de bocs com pantalles tingui l'app. En aquest cas hi haurà dos pantalles de blocs: la de la pantalla inicial i la de la pantalla principal.

**Pantalla inicial**

Aquesta pantalla està formada per pocs blocs ja que no compta amb gaires components i, per tant, no gaires accions a realitzar. Els blocs que la formen es poden veure a les següents figures (Figura 60 i Figura 61):

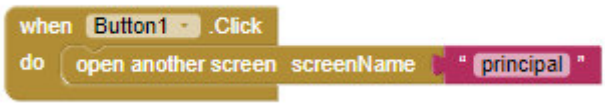



*Figura 60. Blocs botó iniciar app.*

Aquest bloc correspon al botó d'inici, que tal com es veu a la figura, al fer-hi clic obre la pantalla principal.

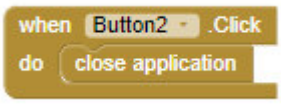



*Figura 61. Blocs botó tancar app.*

Aquest bloc és el que s'encarrega de de tancar l'app al fer-hi clic.

**Pantalla principal**

Aquesta pantalla és la que conté tots els blocs que controlaran la plataforma *Ball in Tube*. Per fer més fàcil d'entendre la programació d'aquesta part, es separaran els blocs en diferents grups o categories segons a quina part del control de la plataforma estiguin relacionats.

*Connexió Bluetooth*

Els blocs que formen la connexió Bluetooth entre el mòdul i la placa són dos, tot i que més endavant es veurà que gairebé tots els grups de blocs tenen algun bloc que fa referència al Bluetooth. Però els blocs d'aquesta categoria són els que intervenen en establir la connexió. Aquesta categoria està formada per dos blocs els quals fan referència al selector de llista esmentat en apartats anteriors. Els blocs són els següents:

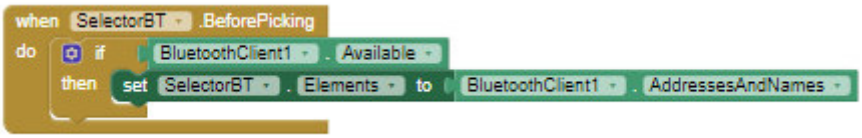



*Figura 62. Blocs selector de llista mostrar adreces Bluetooth.*

Aquest primer bloc és el que estableix que, abans de clicar el selector de llista, si hi ha disponibilitat per connectar-se el mòdul Bluetooth els elements del selector de llista el formaran tots els dispositius Bluetooth vinculats al telèfon.

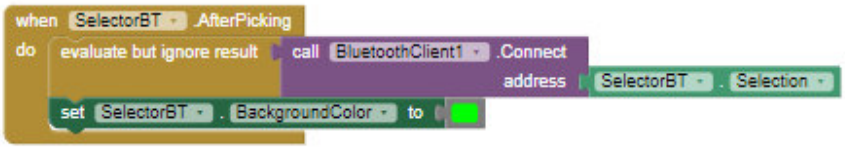



*Figura 63. Blocs connectar adreça Bluetooth seleccionada.*

Després de clicar el selector de llista i triar un dels dispositius o mòduls vinculats, el client Bluetooth o *slave*, és a dir, el mòdul es connecta a l'adreça seleccionada. Alhora, també posarà el fons del selector de llista de color verd per donar a entendre que existeix una connexió.

*Grup de botons de control*

Aquesta categoria està formada pels dos botons que permeten definir el control de la plataforma. A la pantalla, aquests botons es troben al costat del selector de llista esmentat anteriorment. Cada botó té el seu conjunt de blocs, els quals es poden veure a la Figura 64 i la Figura 65.


*Figura 64. Blocs accions produïdes al prémer el botó histèresi.*

Aquest conjunt de blocs correspon a les accions produïdes quan es fa clic al botó *Histèresi*. Quan aquesta acció per part de l'usuari es produeix, el conjunt de blocs comença determinant si aquest botó ja estava activat o no, fet que se sap pel color de fons del botó que passa a ser verd quan està en marxa. En cas que aquest botó no estigui seleccionat, el que fa és posar-lo amb fons verd, enviar el número 1 a la placa, que servirà per activar el control amb histèresi (al proper apartat s'explicarà amb més detall), i fa visibles els components per canviar la

histèresi. A més a més, en cas que l'altre botó estigui de color verd degut a que s'ha premut anteriorment, li canvia el color a gris per determinar que ja no està activat.

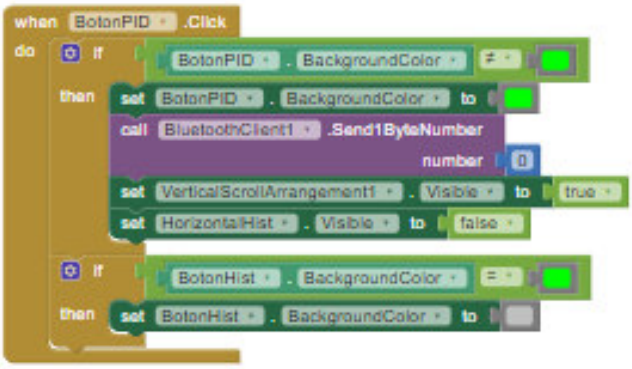



*Figura 65. Blocs accions produïdes al prémer el botó PID.*

Aquest conjunt de blocs correspon a les accions realitzades al prémer el botó *PID*. El funcionament d'aquest conjunt de blocs és el mateix que l'anterior amb l'única diferència que enlloc d'enviar un 1 a la placa envia el número 0.

*Grup de canvi de paràmetres.*

Aquesta categoria està formada pels grups de components que permeten canviar els valors de la histèresi, del *setpoint* i dels paràmetres del control PID.

Els conjunts de blocs de canvi de paràmetres realitzaran les mateixes accions per tots els paràmetres, amb la diferència que estaran adreçats a diferents paràmetres.

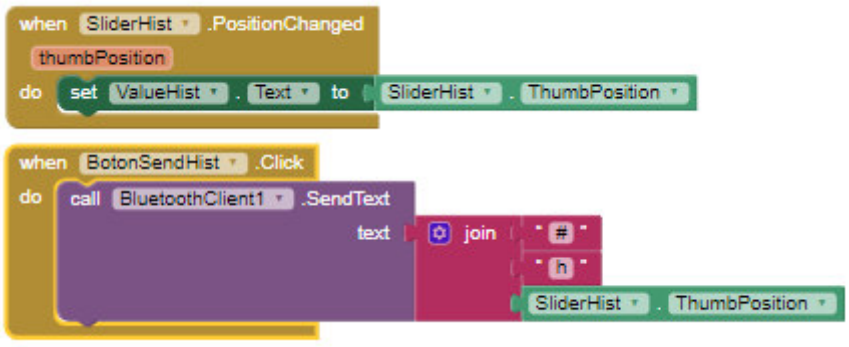



*Figura 66. Blocs components canvi de histèresi.*

Aquesta primera figura (Figura 66) correspon als conjunts de blocs que corresponen al canvi de valor de la histèresi. El primer conjunt, relacionat amb el desplaçament de la barra, provoca que el text que mostra el valor del paràmetre canviï d'acord a la posició de la barra desplaçable. El segon conjunt de blocs s'encarrega d'enviar el valor de la barra desplaçable quan el botó *enviar* és premut.

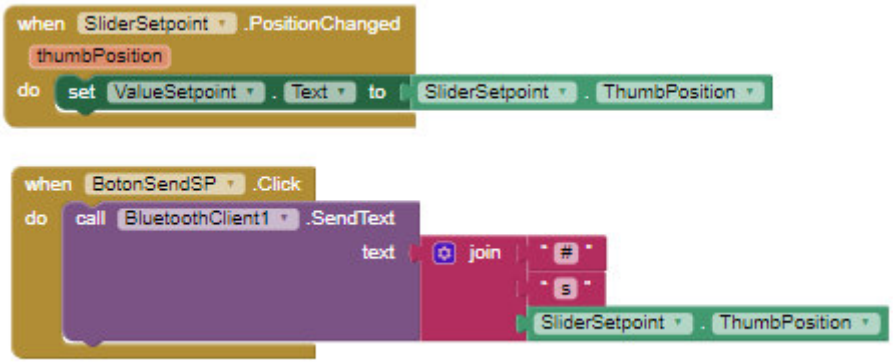



*Figura 67. Conjunt de blocs per modificar el valor del setpoint.*

La Figura 67 representa els blocs que s'encarreguen del canvi de valor del *setpoint*; i aquests duen a terme les mateixes accions que els blocs del canvi de valor de la histèresi amb la diferència que aquests ho fan adreçats al *setpoint*.

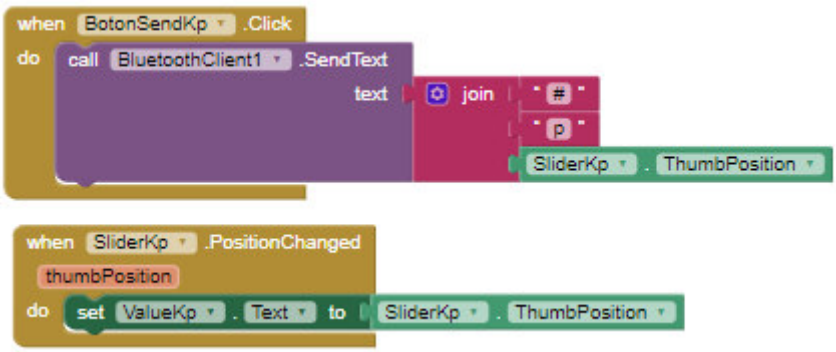



*Figura 68. Conjunt de blocs per modificar el valor de Kp.*

A la Figura 68 es mostra el conjunt de blocs encarregats de la modificació del paràmetre Kp; format pels mateixos elements i realitzant les mateixes accions que els dos blocs anteriors, però destinades a la modificació de Kp.

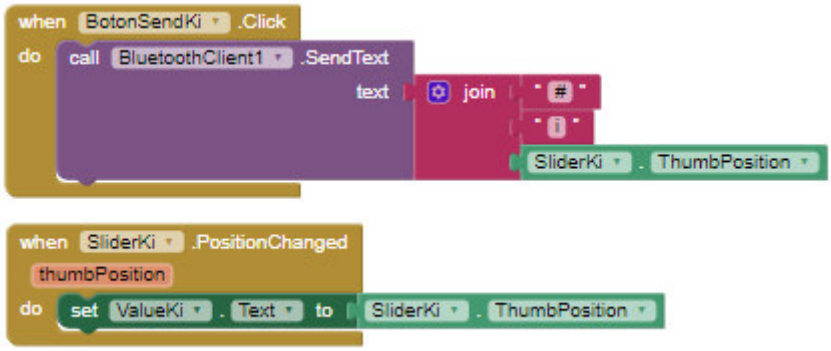



*Figura 69. Conjunt de blocs per modificar el paràmetre Ki.*

A la Figura 69 es pot veure el conjunt de blocs encarregats de modificar el paràmetre Ki. Aquest conjunt està format pels mateixos blocs que els de la Figura 68, però amb la diferència que les seves accions afecten al paràmetre Ki.

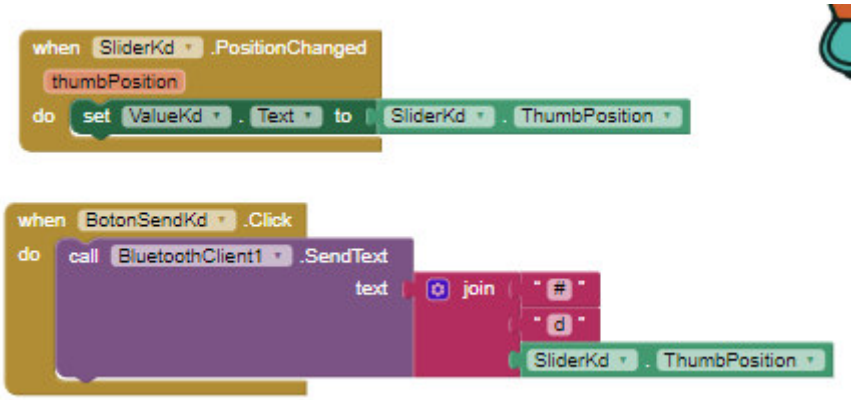



*Figura 70. Conjunt de blocs per modificar el paràmetre Kd.*

A la Figura 70 es mostren els blocs que s'ocupen de poder modificar el valor del paràmetre Kd. Igual que amb els blocs del paràmetre Ki, aquests també realitzen les mateixes accions que els del paràmetre Kp, però adreçats a la modificació del paràmetre Kd.

*La gràfica*

Aquesta categoria és la que es fa càrrec dels blocs relacionats amb la gràfica i la representació dels valors enviats des de la placa en aquesta. Aquesta categoria està formada per diferents grups de blocs, ja que l'acció de mostrar uns valors en una gràfica no només consta d'aquesta representació, sinó que

també cal definir o configurar com es durà a terme aquesta representació de punts.

Primer de tot, és necessari inicialitzar certes variables que s'utilitzaran per poder realitzar aquesta gràfica. Les variables a inicialitzar es poden veure a Figura 71.


*Figura 71. Inicialització de variables de la gràfica.*

Aquestes variables són:

- **Factor d'escala:** s'utilitza perquè la representació de punts de la gràfica s'ajusti a la mida de l'espai de la gràfica. S'inicialitza amb valor 0, però més endavant es modifica el seu valor.
- **xAnt:** correspon al valor anterior de la x. S'utilitza per anar guardant el valor de x a cada cop que va avançant la gràfica.
- **yAnt:** correspon al valor anterior de y. La funció és la mateixa que la de xAnt però en referència al valor y.

Un cop inicialitzades les variables, caldrà determinar com funcionarà el procés de mostrar els punts a la gràfica, el qual es pot veure a Figura 72.

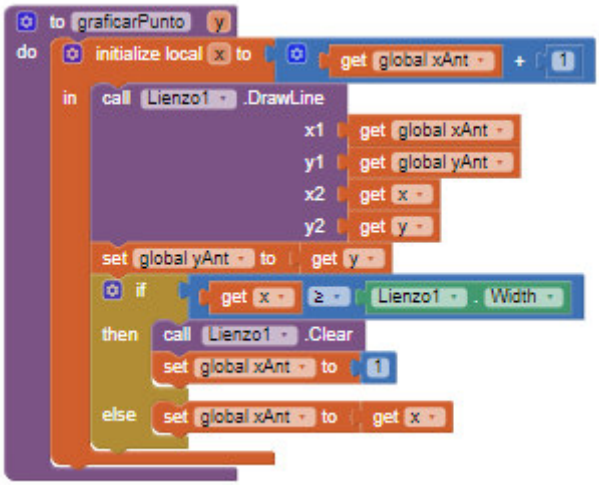



*Figura 72. Procediment sobre com es mostraran els valors a la gràfica.*

Aquest procediment indica com els valors de y, corresponents a les dades enviades des de la placa, es mostraran a la gràfica. Per començar, s'estableix que el valor x serà igual al valor xAnt +1. El valor x correspon a l'eix horitzontal de la gràfica i cada volta del bucle anirà sumant 1. Un cop definida la variable x, es determina que els valors a la gràfica serà mostraran com una línia, definida per quatre punts: punt d'inici (xAnt, yAnt) i punt final (x, y).

Just després de dibuixar aquesta línia a la gràfica, el següent pas és el de passar els valors finals a valors inicials per poder introduir nous valors. El valor de yAnt passarà a ser el valor que tenia y, per així poder donar pas al nou valor que enviï la placa. Pel que fa al valor de xAnt, passarà a ser el valor que tenia x quan no s'arribi a l'amplada de l'espai de la gràfica; en cas que això sí que passi, el valor de xAnt passarà a ser 1 un altre cop i es farà una neteja del que s'havia mostrat a la gràfica, permetent així que es puguin seguir mostrant valors.

Per últim, cal definir el valor de y que es mostrarà quan el component *Clock* faci de *Timer*, fent que y canviï de valor cada cop que passi el temps establert com a *Timer*. A la Figura 73 es poden veure els blocs encarregats d'aquest procés.

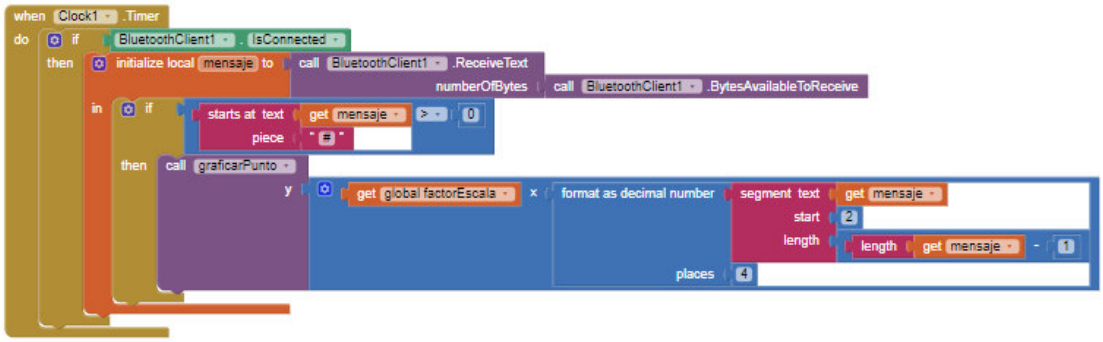



*Figura 73. Valor de y per a la gràfica.*

Aquest procés només s'executa quan hi ha connexió Bluetooth establerta, ja que sinó apareixeria un error abans de connectar-se al mòdul Bluetooth. Un cop es determina que hi ha connexió, es crea una variable local (només per a aquesta acció) que s'encarrega de llegir el missatge que li arriba de la placa. El missatge que s'envia des de la placa amb el valor de la distància de la pilota és de tipus text i conté, a part de números, altres caràcters per poder identificar quan

comença i acaba cada missatge. Un cop rebut el missatge, la següent acció a dur a terme és la de passar-lo a un valor numèric. Per poder fer-ho, caldrà agafar els caràcters que només siguin números i descartar els que no ho siguin (al següent apartat s'explicarà amb més detall aquest procés). Aquests valors numèrics s'hauran de multiplicar pel factor d'escala per així ajustar-los a l'espai de la gràfica. Tot aquest procés es repeteix cada vegada que el *Timer* del component *Clock* es posa a 0 i es torna a reiniciar.

*Configuració inicial*

Per últim, alguns components o variables s'han d'inicialitzar just en el moment en que s'obre la pantalla principal ja que aquesta tot i just arrancar l'app, hi ha certes parts que ja estaran en marxa.


*Figura 74. Inicialització per a la configuració inicial.*

Com es pot veure a la Figura 74, al obrir la pantalla principal de l'app, es defineix el valor que adoptarà el factor d'escala, el qual és l'amplada de l'espai de la gràfica dividit entre el valor màxim que pot adoptar el valor que s'envia des de la placa. Un altre component que s'inicialitza en aquest moment és el byte delimitador del Bluetooth, el qual només s'utilitza al enviar missatges a la placa i del qual se'n parlarà al proper apartat.

Una altra inicialització important és la del color del botó *histèresi*, el qual s'inicialitza com a color verd ja que, per defecte, la plataforma comença funcionant amb el control amb histèresi; per tant, al obrir la pantalla principal de l'app, apareixen els components relacionats amb el control amb histèresi.

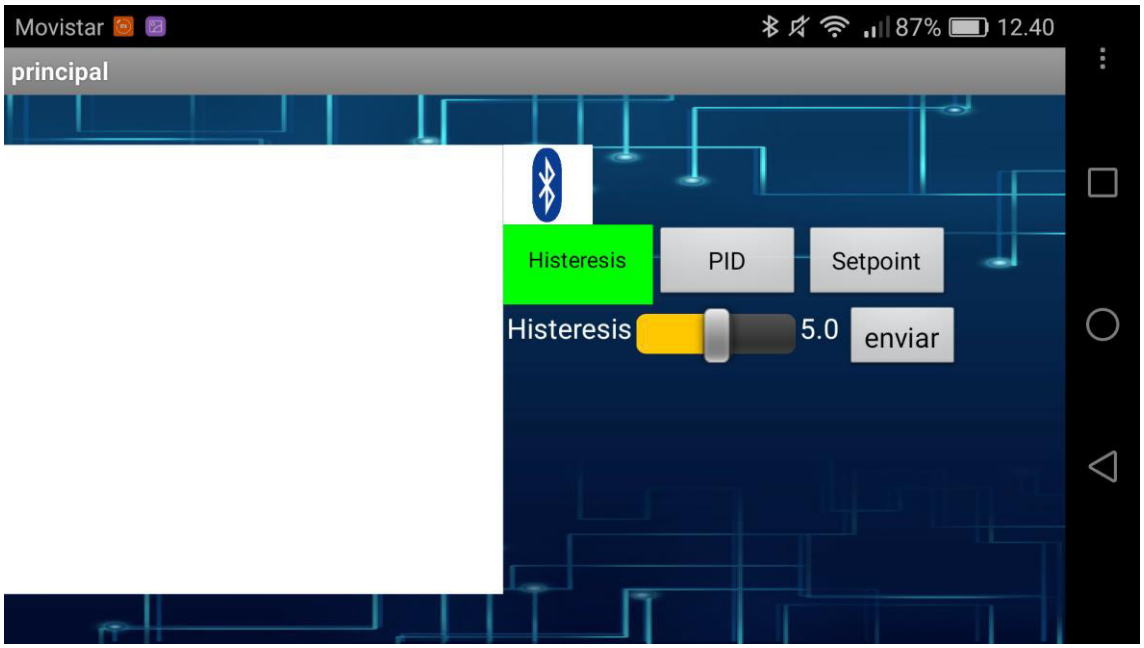



*Figura 75. Configuració inicial pantalla principal.*

## 5.4. Comunicacions amb la placa

Un cop coneguda la programació de l'app i dels seus components, cal fer èmfasi a les comunicacions que s'estableixen entre l'app i la placa Arduino UNO. Tal com s'ha esmentat a l'apartat anterior, els blocs relacionats amb la comunicació Bluetooth s'explicaran amb més detall en aquest apartat, per així poder relacionar aquells blocs amb el codi Arduino amb el qual està directament lligat.

Fent referència al que es comenta a l'apartat 4.3., les comunicacions entre l'app (mòdul Bluetooth) i la placa s'utilitzen per transmetre informació d'un a l'altre, ja sigui amb missatges amb format de text o numèric. Per entendre bé aquesta comunicació, es parlarà de cada exemple on es transmet informació entre l'app i la placa.

El primer cas són els botons *histèresi i PID*, encarregats de gestionar el tipus de control que s'aplica a la plataforma. Als blocs d'aquests dos botons es pot veure com cada un d'ells envia un missatge diferent (a la Figura 64 i Figura 65): el botó *histèresi* envia el número 1 i el botó *PID* envia el número 0. Aquesta diferència al enviar números o text es deu a que al enviar números enters no es produeix cap problema, per això no s'envia tot com a text; a diferència dels propers casos.

Per a cada botó s'envia un missatge diferent perquè al codi aplicat a la placa s'executarà una funció o una altra segons el missatge que la placa rebi de l'app.

```
//######### rebre dades del mòdul ##################
    if (BT1.available()) {
      var = BT1.read();
      //Serial.write(var);

      if (var == 1){
        Serial.println("CONTROL HISTERESIS");
        flag_control=1;
      }

      if (var == 0) {
        Serial.println("CONTROL PID");
        flag_control=0;
      }
```

*Figura 76. Codi lectura missatges enviats per botons Histerèsi i PID.*

A la Figura 76 es veu el codi d'Arduino que s'encarrega d'aquesta primera lectura de la informació que s'envia des dels dos botons de l'app esmentats anteriorment.

La metodologia que s'ha utilitzat ha estat la següent: s'ha creat una variable a la placa Arduino que canvia segons el control que es vulgui posar en marxa (*flag_control*). Quan la placa rep el valor 1 (control amb histèresi), el valor de la variable *flag_control* passa a ser 1, i aquesta variable s'emprarà més endavant per activar totes les accions del control amb histèresi. Quan la placa rep el valor 0 (control PID), el valor de la variable *flag_control* passa a ser 0, fet que provocarà l'activació del control PID més endavant.

Com ja se sap per apartats anteriors, segons el botó que es premi s'activaran uns components o uns altres; i aquests components també transmetran informació a la placa.

En cas de prémer el botó *histèresi*, es mostren els components relacionats amb el canvi de paràmetre. D'aquests componnets, l'únic que estableix una connexió amb la placa és el botó *enviar* (Figura 66). Com es pot veure, el missatge que s'envia no només està format pel nou valor del paràmetre, sinó que va precedit

d'un indicador que marca l'inici del missatge (#), i el qual no és rebut per la placa, i d'una lletra que identifica el paràmetre que es modifica, en aquest cas la *h*.

Quan es prem aquest botó, s'envia el valor en funció de la posició de la barra desplaçable. Aquest valor s'envia com un missatge en format text ja que és un nombre real (conté decimals) i enviar-lo com a valor numèric implica molts problemes per llegir-lo.

Al prémer aquest botó, s'envia el nou valor del paràmetre en un missatge de text, el qual es rebut i llegit per la placa quan s'executa el codi d'Arduino de la Figura 77.

```
    if (flag_control==1){
        String S = GetLine();
        if (S!=""){
          Serial.print(S);
          H = S +'\n';
          if (H[0] == 'h'){
            HISTERESIS = StrToFloat(H);                    // passar string a float
            Serial.println(HISTERESIS);
            Serial.println("histeresis cambiada");
          }
          if (H[0] == 's'){
            SETPOINT = StrToFloat(H);
            Serial.println(SETPOINT);
            Serial.println("setpoint cambiada");
        }
      }
// #### funcionament control amb histèresi ####
      if (Input < SETPOINT - HISTERESIS){
        // encender ALIMENTACION
        PWM = 85;
        analogWrite(VENTILADOR, PWM);
      }

      if (Input > SETPOINT + HISTERESIS){
        //cortar ALIMENTACION
        PWM = 82;
        analogWrite(VENTILADOR, PWM);
      }
    }
```

*Figura 77. Codi control histèresi executat després de prémer el botó histèresi.*

El codi d'aquesta imatge es posa en marxa després de prémer el botó *histèresi* de l'app, és a dir, quan la variable *flag_control* adopta el valor 1. Quan es dóna aquest cas, el codi s'encarrega de crear una variable de text que llegirà i ajuntarà tot el missatge enviat des de la placa, el qual correspon al nou valor de la *histèresi* però en format de text. Aquesta variable de text per ajuntar tot el missatge s'utilitza perquè la forma d'enviar missatges de l'app a la placa és caràcter a caràcter, per tant, és necessari recopilar tots els caràcters rebuts com un sol text.

Un cop recopilat tot el missatge, el codi s'encarrega de transformar-lo en un nombre real (double en aquest cas) i després l'assigna a la variable del paràmetre a canviar, en aquest cas la *histèresi*. Per als altres paràmetres que siguin nombres reals, el mètode de lectura del missatge i de passar-lo a un nombre real serà el mateix.

Degut a que el *setpoint* es manté actiu en qualsevol dels dos controls, quan està activat el control amb histèresi també es pot modificar el valor d'aquest paràmetre. Aprofitant la lectura de dades i la recopilació en un missatge de text que es realitza quan s'activa el control amb histèresi, s'ha introduït un bucle if que permet llegir el primer caràcter del missatge (Figura 77), el qual correspon a la lletra identificadora del paràmetre a modificar. De la mateixa manera que quan s'enviava el missatge del canvi de valor de la histèresi, quan s'envia el canvi de valor del *setpoint* (Figura 67) aquest va precedit de la lletra *s* per fer saber a la placa que el paràmetre a modificar és el *setpoint*.

Tornant a observar la Figura 77, després de recopilar el missatge en un sol text; en funció de la lletra que precedeixi el valor, aquest missatge es transformarà en un nombre real i es guardarà a la variable corresponent al paràmetre modificat.

Pel que fa a canvi de paràmetres, quan es prem el botó *PID* i la variable *flag_control* passa a tenir el valor 0, passarà a executar-se el codi de les següents imatges.

```
// #### control PID activat ####
    if (flag_control==0){

// #### funcionament control PID ####
      myPID.Compute();

      PWM = Output+84;
      analogWrite(VENTILADOR, PWM);

// #### lectura del paràmetre a modificar ####

        String T = GetLine();
        if (T!=""){
          Serial.print(T);
          K = T + '\n';
          if (K[0] == 'p'){
            Kp=StrToFloat(K);                    //passar de string a double
            Serial.println(Kp);
            Serial.println("kp cambiada");
          }
          if (K[0] == 'i'){
            Ki=StrToFloat(K);
            Serial.println(Ki);
            Serial.println("ki cambiada");
```

*Figura 78. Codi posta en marxa control PID (part 1).*

```
        if (K[0] == 'd'){
          Kd=StrToFloat(K);                    //passar de string a double
          Serial.println(Kd);
          Serial.println("kd cambiada");
        }
        if (K[0] == 's'){
          SETPOINT=StrToFloat(K);
          Serial.println(SETPOINT);
          Serial.println("setpoint cambiada");
        }
      }
  }
```

*Figura 79. Codi posta en marxa control PID (part 2).*

Aquest codi s'encarrega de posar en marxa el control PID i també de tornar a fer una lectura de les dades que envia l'app per emmagatzemar-les en unmissatge de text, com en el cas del control amb histèresi. A diferència de l'altre control, en aquest hi ha més possibles paràmetres a modificar, per tant, això vol dir que hi haurà més possibles lletres que identifiquin el paràmetre que li arriba a la placa.

En cas de modificar-se el paràmetre Kp, el missatge anirà precedit d'una *p*. Si el que es vol modificar és la Ki, el primer caràcter del missatge serà una *i*. I per últim, en cas de modificar la Kd es rebrà un missatge que comenci per la lletra *d*.

En aquest control també es pot modificar el *setpoint*, per tant, tot el que s'ha esmentat de la modificació d'aquest paràmetre es repeteix en aquest control.

L'últim cas on es produeix una comunicació entre l'app i la placa és en els valors que es mostren a la gràfica. En aquest cas, la transmissió d'informació va de la placa a l'app, no com en els casos anteriors. Aquesta informació que s'enviarà correspon al valor de la distància que hi ha entre la pilota i la part inferior del tub; valor extret de les lectures del sensor ultrasònic i una operació matemàtica. El missatge que s'enviarà a la placa també ha de ser en format text, per tant, primer de tot és necessari passar el valor numèric de la distància a un text.

```
//########## enviar datos al módulo ##################
    String stringOne = String(Input, 2);
    BT1.print('#' + stringOne + '\n');
```

*Figura 80.* *Codi enviar dades a l'app.*

Tal com es veu a la Figura 80, primer s'executa una funció per canviar el valor numèric a text; i després s'envia el missatge a l'app, però afegint un caràcter a l'inici del missatge (#) i un altre al final (\n). Aquests dos caràcters s'afegeixen per diferenciar cada missatge quan aquest és llegit per l'app.

Quan el missatge arriba a l'app, els blocs encarregats de determinar el valor de y llegeixen el missatge rebut des de la placa i separen els números dels caràcters delimitadors del missatge (Figura 73).

# 6. Estudi econòmic

Per a realitzar l'estudi econòmic s'han tingut en compte dos tipus de despeses, les materials i les d'enginyeria; aquestes últimes són les que corresponen a les hores de treball de l'enginyer com a tal.

Pel que fa als costos materials, no són gaire elevats degut a que la plataforma ja estava muntada i només ha estat necessari comprar alguns components per completar la plataforma. A la següent taula es poden veure aquests costos.

| COSTOS MATERIALS | | | |
|---|---|---|---|
| DESCRIPCIÓ | UNITATS | PREU UNITARI (€) | PREU TOTAL (€) |
| Mòdul Bluetooth HC-06 | 1 | 9,66 | 9,66 |
| Placa Protoboard ROHS BB-301 (1/2 Protoboard) | 1 | 2,0933 | 2,09 |
| Conjunt de cables multicolors | 1 | 1,55 | 1,55 |
| TOTAL | | | 13,30 |

Taula 2. Costos materials.

Pel que fa als costos d'enginyeria, corresponen a les hores invertides en dissenyar l'app, en realitzar assajos, en programar, etc.

| COSTOS D'ENGINYERIA | | | |
|---|---|---|---|
| DESCRIPCIÓ | PREU UNITARI (€) | UNITATS (h) | PREU TOTAL (€) |
| Disseny inicial de l'app | 20 | 10 | 200 |
| Anàlisi del funcionament dels components | 20 | 25 | 500 |
| Programació de la placa | 20 | 200 | 4000 |
| Programació de l'app | 20 | 250 | 5000 |
| Redacció de la memòria | 20 | 300 | 6000 |
| Assajos i proves amb la plataforma | 20 | 75 | 1500 |
| TOTAL | | 860 | 17200 |

Taula 3. Costos d'enginyeria.

A partir d'aquestes dues taules de costos es pot extreure el preu total que ha suposat la realització d'aquest treball, el qual es pot veure a la taula següent.

| COSTOS TOTALS | |
|---|---|
| DESCRIPCIÓ | PREU (€) |
| Costos materials | 13,30 |
| Costos d'enginyeria | 17200 |
| TOTAL | 17213,30 |

Taula 4. Costos totals.

# 7. Anàlisi de l'impacte ambiental

La realització d'aquest projecte no implica cap problema mediambiental degut a que els components utilitzats no realitzen cap tipus d'emissió que sigui perjudicial per al medi ambient.

Degut també a que la plataforma ja estava construïda, no ha pogut aparèixer cap problema que es pogués donar durant la seva construcció i muntatge, evitant així qualsevol situació perjudicial per al medi ambient.

Per acabar, cal comentar que en cas de mal funcionament de la plataforma o de qualsevol dels seus components no hi hauria cap implicació mediambiental, ja que es tracta d'una estructura de petites dimensions i que els components que la formen no poden causar cap dany en cas de mal funcionament.

# 8. Conclusions

L'objectiu del projecte realitzat era establir un control a una plataforma *Ball in Tube* mitjançant un dispositiu mòbil, és a dir, exercir un control sobre la plataforma esmentada a través d'una app. Un cop finalitzat el projecte es poden extreure les següents conclusions:

- Tenint en compte que els components de la plataforma ja estaven inclosos, el funcionament d'aquests és correcte i permet que la plataforma, com a conjunt, funcioni de forma adient i sense problemes.
- S'han definit dos controls diferents a la plataforma, dels quals s'ha extret el millor funcionament possible dins les condicions de funcionament de la pròpia plataforma, tenint en compte les limitacions que oferia l'entorn de treball.
- El control PID, que s'ha determinat com a més eficient entre els dos establerts i dins de l'entorn de treball, presenta un funcionament correcte i amb un error petit durant la seva execució; i, respecte al control amb histèresi, la diferència en el funcionament no és gaire notòria. Per tant, en algunes circumstàncies concretes, el control amb histèresi podria donar millors resultats que el control PID dins de la plataforma del projecte.
- Aquests dos controls definits no són els més òptims degut a que l'objectiu principal del projecte és realitzar un control a través d'un dispositiu mòbil; per tant, aquests dos controls establerts s'han utilitzat com a demostració de la possibilitat de realitzar aquest control a distància ja que ambdós controls s'havien treballat en assignatures del grau.
- La comunicació utilitzada per realitzar aquest control a distància s'ha escollit degut als beneficis i a la simplicitat que aportava al projecte i a les seves condicions de treball.
- L'app creada per realitzar aquest control és capaç de manipular tots els paràmetres que entren en joc en ambdós controls, a més a més, de comptar amb un disseny intuïtiu i senzill d'utilitzar, evitant mostrar massa elements a la pantalla i oferint a l'usuari totes els dades necessàries per realitzar aquest control de la plataforma.

- Les comunicacions establertes entre l'app i la placa s'han definit de tal manera que no apareguin creuaments d'informació, fent així que les dades transmeses d'un lloc a l'altre arribin sense ser modificades.
- Per facilitar l'ús de l'app, aquesta s'ha configurat de tal manera que només es puguin modificar els paràmetres d'un en un, facilitant així a l'usuari l'ús de l'app.

Com a conclusió final, es pot dir que els objectius establerts per a aquest projecte s'han complert en la seva major part. El control a distància de la plataforma obtingut funciona correctament i és de fàcil ús. Els controls definits a la plataforma funcionen correctament en les condicions de treball que ofereix la plataforma, però no són els més òptims ja que aquest no era l'objectiu del projecte.

# Bibliografia

*Arduino: main page*. [en línia]. Accedit el 13 d'abril de 2018. Disponible a: https://www.arduino.cc/

*Arduino: Reference*. [en línia]. Accedit el 18 d'abril de 2018. Disponible a: https://www.arduino.cc/reference/en/

*Módulo Bluetooth HC-06*. [en línia]. Accedit el 15 de juny de 2018. Disponible a: https://www.prometec.net/bt-hc06/

*Conexión Bluetooth mediante modulo HC-06*. [en línia]. 2017, 25 d'octubre. Accedit el 2 de juny de 2018. Disponible a: https://forum.arduino.cc/index.php?topic=507595.0

*MIT App Inventor: Connect your phone or Tablet over WiFi*. [en línia]. Accedit el 22 de setembre de 2018. Disponible a: http://appinventor.mit.edu/explore/ai2/setup-device-wifi.html

*MIT App Inventor: Getting Started with MIT App Inventor 2*. [en línia]. Accedit el 27 de maig de 2018. Disponible a: http://appinventor.mit.edu/explore/get-started.html

*El código ASCII*. [en línia]. Accedit el 6 de setembre de 2018. Disponible a: https://elcodigoascii.com.ar/

*How do you convert a float to string*. [en línia]. Accedit el 4 de setembre de 2018. Disponible a: http://forum.arduino.cc/index.php?topic=243660.0

Villalpando, J. A., *App Inventor 2 en español: Tutorial de iniciación de App Inventor 2.* [en línia]. Accedit el 27 de juliol de 2018. Disponible a: http://kio4.com/appinventor/169DDD_javascript_funciones.htm

González, C., *WiFi vs Bluetooth: diferencias, ventajas e inconvenientes*. [en línia]. 2014, 27 de novembre. Accedit el 1 de juliol de 2018. Disponible a: https://www.adslzone.net/2014/11/27/wifi-vs-bluetooth-diferencias-ventajas-e-inconvenientes/

*Bluetooth vs Wi-Fi*. [en línia]. Accedit el 1 de juliol de 2018. Disponible a: https://www.diffen.com/difference/Bluetooth_vs_Wifi

Beauregard, B., *Improving the Beginner's PID-Introduction* [en línia]. Accedit el 25 d'abril de 2018. Disponible a: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

Moyano, J. E., *Arduino PID – Guía de uso de la librería* [en línia]. Accedit el 25 d'abril de 2018. Disponible a: http://brettbeauregard.com/blog/wp-content/uploads/2012/07/Gu%C3%ADa-de-uso-PID-para-Arduino.pdf

*Controlador PID*. [en línia]. Accedit el 2 de maig de 2018. Disponible a: https://es.wikipedia.org/wiki/Controlador_PID

Golato, M. A., *Acciones de control*. [en línia]. 2016, Universidad Nacional de Tucumán. Accedit el 10 d'abril de 2018. Disponible a: https://catedras.facet.unt.edu.ar/sistemasdecontrol/wp-content/uploads/sites/101/2016/04/5_Acciones-de-control_2016.pdf

# Annexos

## Codi font dels diferents programes i parts del treball

### Codi de la placa Arduino UNO

```
// ####### definició resta de variables #############
float HISTERESIS=5;
float DISTANCIA;
float DURACION;
int TRIG=10;
int ECO=9;
int VENTILADOR=3;
int PWM;
float setpoint_aux;
char var;
int flag_control;
String H;
String K;

// ######### inicialització de variables i paràmetres ##########
void setup()
   {
      SETPOINT = 40;
      pinMode(TRIG, OUTPUT);
      pinMode(ECO, INPUT);
      pinMode(VENTILADOR, OUTPUT);
      Serial.begin(9600);
      BT1.begin(9600);
      myPID.SetOutputLimits(-4,4);
      myPID.SetMode(AUTOMATIC);
      analogWrite(VENTILADOR,90);
      flag_control=1;
   }

// ######### funció de lectura de dades rebudes
// i recopilació en un string #################

String GetLine(){
  String S = "";
  while (BT1.available()){
    var = BT1.read();
    S = S + var;
    delay(25);
  }
  return(S);
}
```

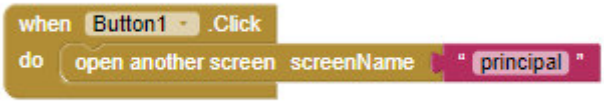
```
// ####### funció transformar string en float ######
float StrToFloat(String Texto){
  int i=1;
  float numf=0.0;
  while ((Texto[i] != '.') && (Texto[i] != '\n')){
    numf = 10*numf + (Texto[i]-48);
    i=i+1;
  }
  if (Texto[i] == '.'){
    float peso=0.1;
    i=i+1;
    while (Texto[i] != '\n'){
      numf = numf + ((Texto[i]-48)*peso);
      peso = peso*0.1;
      i=i+1;
    }
  }
  return numf;
}

// ####### funció transformar string en double ######
double StrToDouble(String Texto2){
  int i=0;
  double numd=0.0;
  while ((Texto2[i] != '.') && (Texto2[i] != '\n')){
    numd = 10*numd + (Texto2[i]-48);
    i=i+1;
  }
  if (Texto2[i] == '.'){
    float peso=0.1;
    i=i+1;
    while (Texto2[i] != '\n'){
      numd = numd + ((Texto2[i]-48)*peso);
      peso = peso*0.1;
      i=i+1;
    }
  }
  return numd;
}
```

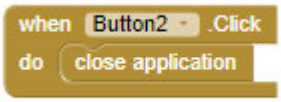
```
void loop(){

// ######## reatlizar lectures amb sensor #############
    digitalWrite(TRIG, LOW);
    delay(0.01);
    digitalWrite(TRIG, HIGH);
    delay(0.01);
    digitalWrite(TRIG, LOW);
    DURACION = pulseIn(ECO, HIGH);
    DISTANCIA = (DURACION / 58);
    Input=78-DISTANCIA;

//########## enviar dades al mòdul ##################
    String stringOne = String(Input, 2);
    BT1.print('#' + stringOne + '\n');

 //######### rebre dades del mòdul #################
    if (BT1.available()) {
      var = BT1.read();
      //Serial.write(var);

      if (var == 1){
        Serial.println("CONTROL HISTERESIS");
        flag_control=1;
      }

      if (var == 0) {
        Serial.println("CONTROL PID");
        flag_control=0;
      }
    }
```

```
//#### control amb histèresi activat ####
    if (flag_control==1){
        String S = GetLine();
        if (S!=""){
          Serial.print(S);
          H = S +'\n';
          if (H[0] == 'h'){
            HISTERESIS = StrToFloat(H);                    // passar string a float
            Serial.println(HISTERESIS);
            Serial.println("histeresis cambiada");
          }
          if (H[0] == 's'){
            SETPOINT = StrToFloat(H);
            Serial.println(SETPOINT);
            Serial.println("setpoint cambiada");
        }
      }
// #### funcionament control amb histèresi ####
      if (Input < SETPOINT - HISTERESIS){
        // encender ALIMENTACION
        PWM = 85;
        analogWrite(VENTILADOR, PWM);
      }
      if (Input > SETPOINT + HISTERESIS){
        //cortar ALIMENTACION
        PWM = 82;
        analogWrite(VENTILADOR, PWM);
      }
    }
```

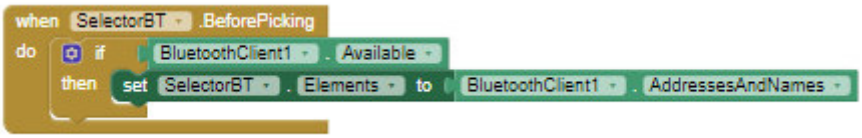
```
// #### control PID activat ####
    if (flag_control==0){

// #### funcionament control PID ####
      myPID.Compute();

      PWM = Output+84;
      analogWrite(VENTILADOR, PWM);

// #### lectura del paràmetre a modificar ####

        String T = GetLine();
        if (T!=""){
          Serial.print(T);
          K = T + '\n';
          if (K[0] == 'p'){
            Kp=StrToFloat(K);                    //passar de string a double
            Serial.println(Kp);
            Serial.println("kp cambiada");
          }
          if (K[0] == 'i'){
            Ki=StrToFloat(K);
            Serial.println(Ki);
            Serial.println("ki cambiada");
          if (K[0] == 'd'){
            Kd=StrToFloat(K);                    //passar de string a double
            Serial.println(Kd);
            Serial.println("kd cambiada");
          }
          if (K[0] == 's'){
            SETPOINT=StrToFloat(K);
            Serial.println(SETPOINT);
            Serial.println("setpoint cambiada");
          }
        }
    }
    delay(200);
}
```

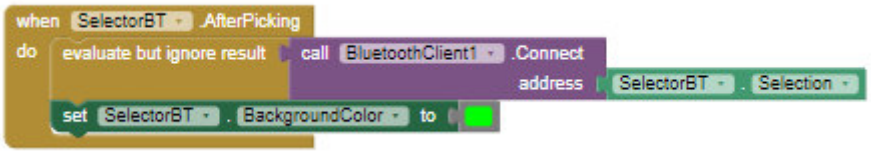
## Codi de l'app (MIT App Inventor

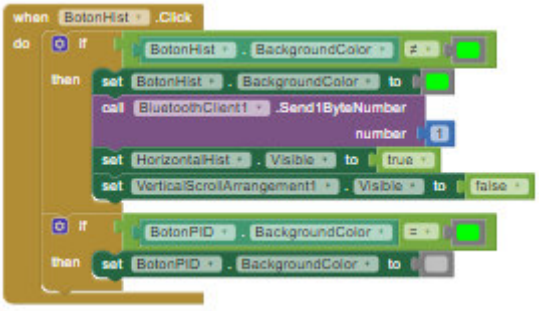


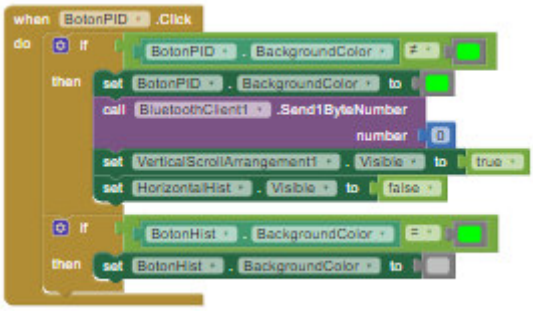





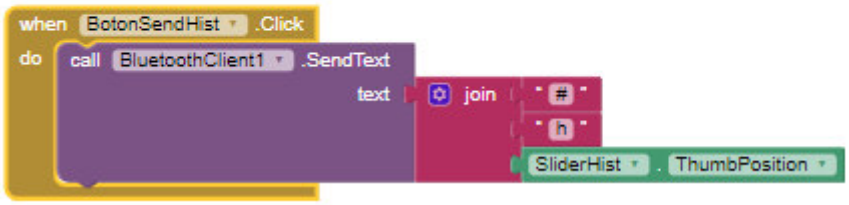



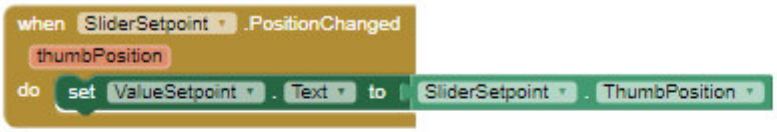



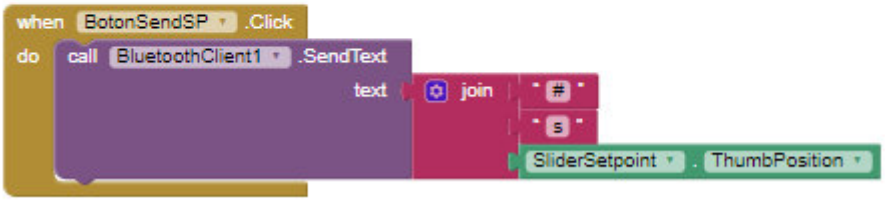



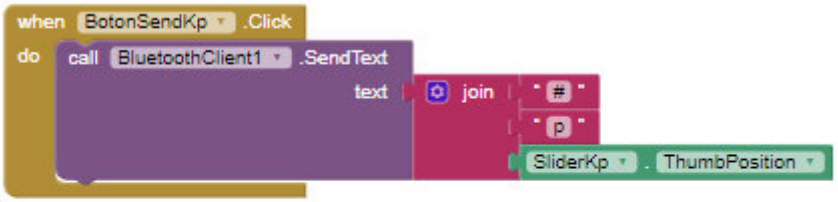

when BotonPID .Click
do if BotonPID . BackgroundColor ≠
then set BotonPID . BackgroundColor to
call BluetoothClient1 .Send1ByteNumber
number 0
set VerticalScrollArrangement1 . Visible to true
set HorizontalHist . Visible to false
if BotonHist . BackgroundColor =
then set BotonHist . BackgroundColor to


when SliderHist .PositionChanged
thumbPosition
do set ValueHist . Text to SliderHist . ThumbPosition

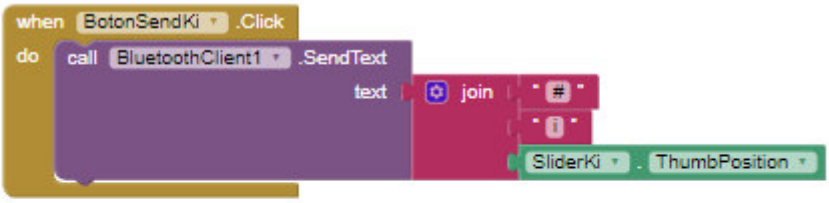

when BotonSendHist .Click
do call BluetoothClient1 .SendText
text join "#"
"h"
SliderHist . ThumbPosition


when SliderSetpoint .PositionChanged
thumbPosition
do set ValueSetpoint . Text to SliderSetpoint . ThumbPosition

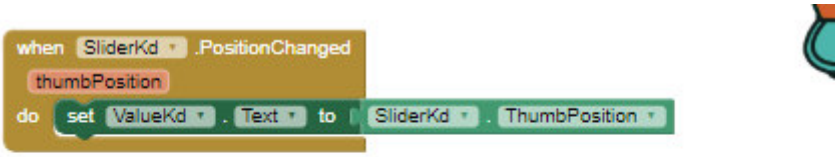

when BotonSendSP .Click
do call BluetoothClient1 .SendText
text join "#"
"s"
SliderSetpoint . ThumbPosition

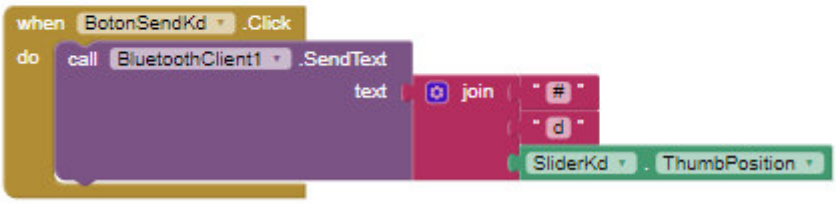

when BotonSendKp .Click
do call BluetoothClient1 .SendText
text join "#"
"p"
SliderKp . ThumbPosition

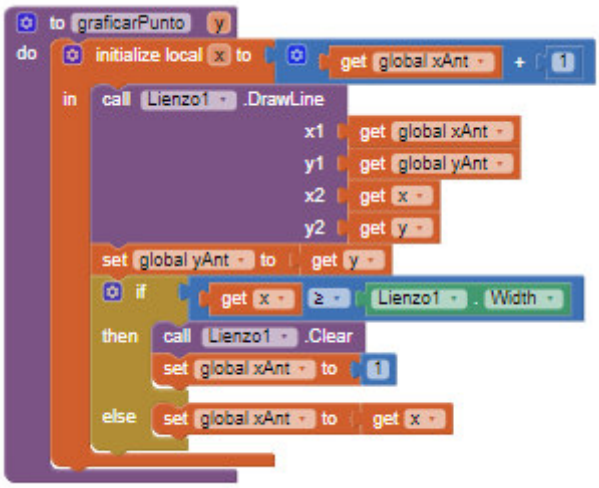

when SliderKp .PositionChanged
thumbPosition
do set ValueKp . Text to SliderKp . ThumbPosition

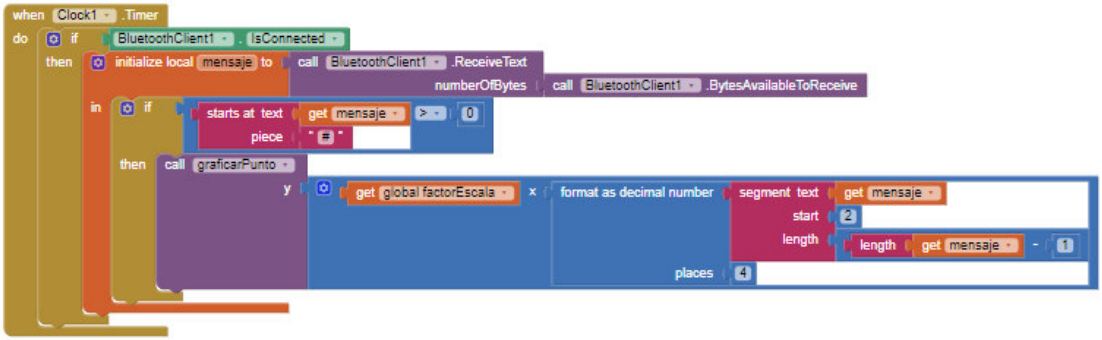

when BotonSendKi .Click
do call BluetoothClient1 .SendText
text join "#"
"i"
SliderKi . ThumbPosition


when SliderKi .PositionChanged
thumbPosition
do set ValueKi . Text to SliderKi . ThumbPosition


when SliderKd .PositionChanged
thumbPosition
do set ValueKd . Text to SliderKd . ThumbPosition


when BotonSendKd .Click
do call BluetoothClient1 .SendText
text join "#"
"d"
SliderKd . ThumbPosition


to graficarPunto y
do initialize local x to get global xAnt + 1
in call Lienzo1 .DrawLine
x1 get global xAnt
y1 get global yAnt
x2 get x
y2 get y
set global yAnt to get y
if get x ≥ Lienzo1 . Width
then call Lienzo1 .Clear
set global xAnt to 1
else set global xAnt to get x


when Clock1 .Timer
do if BluetoothClient1 . IsConnected
then initialize local mensaje to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
in if starts at text get mensaje > 0
piece "#"
then call graficarPunto
y get global factorEscala x format as decimal number segment text get mensaje
start 2
length length get mensaje - 1
places 4

# Datasheets dels components

DELTA ELECTRONICS, INC.
252, SHANG YING ROAD, KUEI SAN
TAOYUAN HSIEN 333, TAIWAN, R. O. C.
TEL : 886-(0)3-3591968
FAX : 886-(0)3-3591991

## SPECIFICATION FOR APPROVAL

Customer:

Description: DC FAN

Customer P/N: REV:

Delta Model NO.: QFR0812DE-F00

Sample Rev: 01 Issue NO:

Sample Issue Date: APR.20.2006. Quantity:

1. SCOPE:

THIS SPECIFICATION DEFINES THE ELECTRICAL AND MECHANICAL CHARACTERISTICS OF THE DC BRUSHLESS AXIAL FLOW FAN. THE FAN MOTOR IS WITH TWO PHASES AND FOUR POLES.

2. CHARACTERS:

| ITEM | DESCRIPTION |
|---|---|
| RATED VOLTAGE | 12 VDC |
| OPERATION VOLTAGE | 10.8 - 13.2 VDC |
| INPUT CURRENT | 1.15 ( 1.70 MAX.) A |
| INPUT POWER | 13.8 ( 20.40 MAX. ) W |
| SPEED | 9000R.P.M. (REF.) |
| MAX. AIR FLOW (AT ZERO STATIC PRESSURE) | 2.991 ( MIN. 2.692) $M^3$/MIN<br>105.21 ( MIN. 94.69) CFM |
| MAX.AIR PRESSURE (AT ZERO AIR FLOW) | 33.91 ( MIN. 27.47 )$mmH_2O$<br>1.335 ( MIN. 1.081 )$inchH_2O$ |
| ACOUSTICAL NOISE (AVG.) | 60.0 (MAX 64.0 ) dB-A |
| INSULATION TYPE | UL: CLASS A |

(continued)

PART NO:

DELTA MODEL: QFR0812DE-F00

| | |
|---|---|
| INSULATION STRENGTH | 10 MEG OHM MIN. AT 500 VDC<br>(BETWEEN FRAME AND (+) TERMINAL) |
| DIELECTRIC STRENGTH | 5 mA MAX. AT 500 VAC 60 Hz<br>ONE MINUTE, (BETWEEN FRAME AND<br>(+) TERMINAL) |
| EXTERNAL COVER | OPEN TYPE |
| LIFE EXPECTANCE | 50,000 HOURS CONTINOUS OPERATION<br>AT 40 °C WITH 15 ~ 65 %RH. |
| ROTATION | CLOCKWISE VIEW<br>FROM NAME PLATE SIDE |
| OVER CURRENT SHUT DOWN | THE CURRENT WILL SHUT DOWN,<br>WHEN LOCKING ROTOR. |
| LEAD WIRE | UL 1007 -F- AWG #24<br>BLACK WIRE NEGATIVE(-)<br>RED WIRE POSITIVE(+)<br>UL 1061 -F- AWG #24<br>BLUE WIRE ( FOO )<br>YELLOW WIRE ( PWM ) |

NOTES: 1. ALL READINGS ARE MEASURED AFTER STABLY WARMING UP THROUGH 10 MINUTES
2. THE VALUES WRITTEN IN PARENS , ( ), ARE LIMITED SPEC.
3. ACOUSTICAL NOISE MEASURING CONDITION:


NOISE IS MEASURED AT RATED VOLTAGE IN FREE AIR IN ANECHOIC CHAMBER WITH B & K SOUND LEVEL METER WITH MICROPHONE AT A DISTANCE OF ONE METER FROM THE FAN INTAKE.

page: 2

A00

PART NO:

DELTA MODEL: QFR0812DE-F00

3. MECHANICAL:

| | |
|---|---|
| 3-1. DIMENSIONS | SEE DIMENSIONS DRAWING |
| 3-2. FRAME | PLASTIC UL: 94V-0 |
| 3-3. IMPELLER | PLASTIC UL: 94V-0 |
| 3-4. BEARING SYSTEM | TWO BALL BEARINGS |
| 3-5. WEIGHT | 160 GRAMS |

4. ENVIRONMENTAL:

| | |
|---|---|
| 4-1. OPERATING TEMPERATURE | -10 TO +60 DEGREE C |
| 4-2. STORAGE TEMPERATURE | -40 TO +70 DEGREE C |
| 4-3. OPERATING HUMIDITY | 5 TO 90 % RH |
| 4-4. STORAGE HUMIDITY | 5 TO 95 % RH |

5. PROTECTION:

5-1. LOCKED ROTOR PROTECTION

IMPEDANCE OF MOTOR WINDING PROTECTS MOTOR FROM FIRE IN 96 HOURS OF LOCKED ROTOR CONDITION AT THE RATED VOLTAGE.

5-2. POLARITY PROTECTION

BE CAPABLE OF WITHSTANDING IF REVERSE CONNECTION FOR POSITIVE AND NEGATIVE LEADS.

6. RE OZONE DEPLETING SUBSTANCES:

6-1. NO CONTAINING PBBs, PBBOs, CFCs, PBBEs, PBDPEs AND HCFCs.

7. PRODUCTION LOCATION

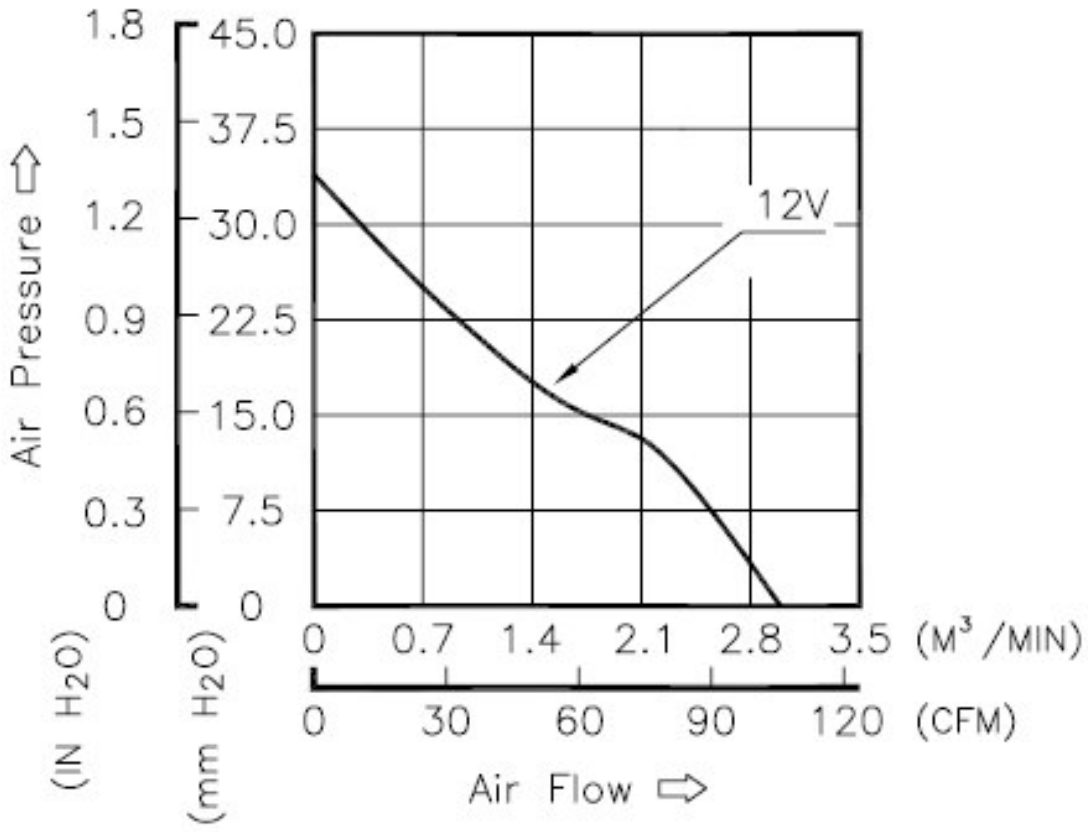
7-1. PRODUCTS WILL BE PRODUCED IN CHINA OR THAILAND OR TAIWAN.

page: 3 A00

PART NO:

DELTA MODEL: QFR0812DE-F00

P & Q CURVE:




* TEST CONDITION: INPUT VOLTAGE ----- OPERATION VOLTAGE
TEMPERATURE ------ ROOM TEMPERATURE
HUMIDITY ---------- 65%RH

page: 4

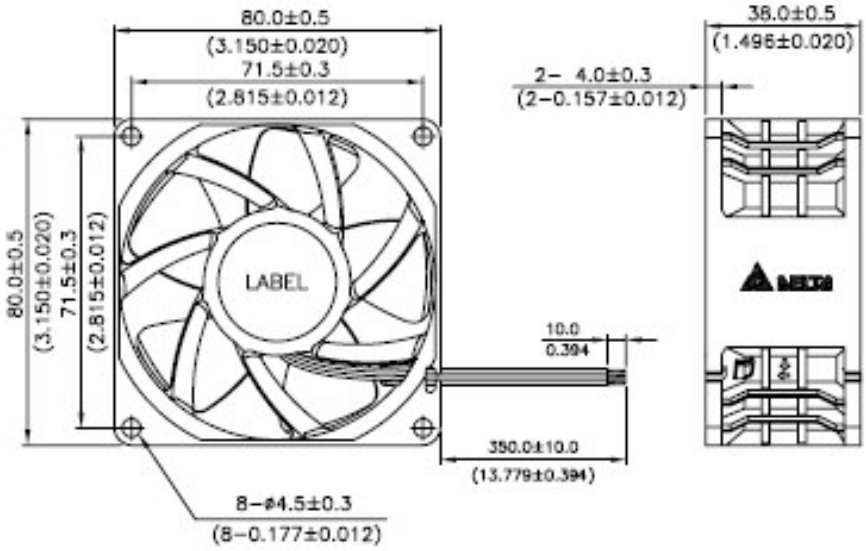
A00

PART NO:

DELTA MODEL: QFR0812DE-F00

Attach: DIMENSIONS DRAWING

LABEL:

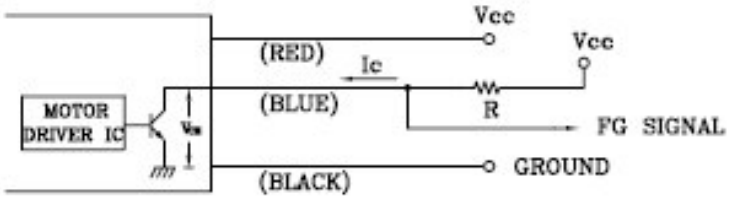



PART NO:

DELTA MODEL: QFR0812DE-F00

ROTATION DETECT (FG) SIGNAL:

1. OUTPUT CIRCUIT - OPEN COLLECTOR MODE:



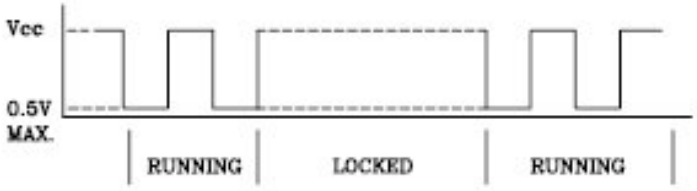

CAUTION: THE FG SINGAL LEAD WIRE MUST BE KEPT AWAY FROM "+" LEAD WIRE & "-" LEAD WIRE.

2. SPECIFICATION:

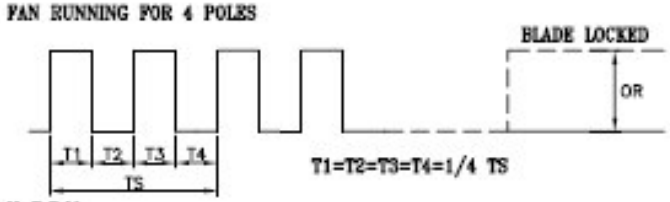
$V_{cc}$ = 13.2 V MAX. $I_c$ = 5mA MAX.

$V_{ce}$ = 0.5V MAX. $R \geq V_{cc}/I_c$

3. FREQUENCY GENERATOR WAVEFORM:




FAN RUNNING FOR 4 POLES




N=R.P.M

TS=60/N(SEC)

*VOLTAGE LEVEL AFTER BLADE LOCKED

*4 POLES

page: 6

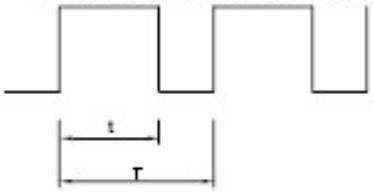
A00

PART NO:

DELTA MODEL: QFR0612DE-F00

11. PWM CONTROL SIGNAL:

SIGNAL VOLTAGE RANGE: 0 ~ 20VDC



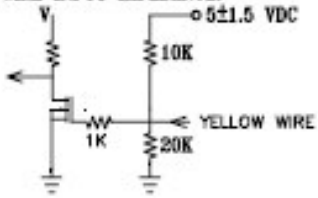

DUTY CYCLE= $\frac{t}{T}$ *100(%)

- THE FREQUENCY FOR CONTROL SINGAL OF THE FAN SHALL BE ABLE TO ACCEPT A 30HZ~300 KHZ.
- THE PREFERRED OPERATING POINT FOR THE FAN IS 25K HZ.
- AT 100% DUTY CYCLE,THE ROTOR WILL SPIN AT MAXIMUM SPEED.
- AT 0% DUTY CYCLE,THE ROTOR WILL STOP.
- WITH CONTROL SIGNAL LEAD DISCONNECTED,THE FAN WILL SPIN AT MAXIMUM SPEED.
- AT RATED VOLTAGE ,25K HZ ,30% DUTY CYCLE ,THE FAN WILL BE ABLE TO START FROM A DEAD STOP .

12. SPEED VS PWM CONTROL SIGNAL: (AT RATED VOLTAGE & PWM FREQUENCY=25KHZ)

| DUTY CYCLE (%) | SPEED R.P.M. (REF.) | CURRENT (A) TYP. |
|---|---|---|
| 100 | 9000±10% | 1.15 |
| 50 | 4600±10% | 0.20 |
| 0 | 0 | 0.01 |

13. PWM CONTROL LEAD WIRE INPUT IMPEDANCE:




14-1. THE FAN SPEED WILL DEFAULT TO MAXIMUM WHEN THE SPEED CONTROLL INPUT IS DISCONNECTED.

page: 7

A00

Descriptions:

1. Delta will not guarantee the performance of the products if the application condition falls outside the parameters set forth in the specification.
2. A written request should be submitted to Delta prior to approval if deviation from this specification is required.
3. Please exercise caution when handling fans. Damage may be caused when pressure is applied to the impeller, if the fans are handled by the lead wires, or if the fans are hard-dropped to the production floor.
4. Except as pertains to some special designs, there is no guarantee that the products will be free from any such safety problems or failures as caused by the introduction of powder, droplets of water or encroachment of insect into the hub.
5. The above-mentioned conditions are representative of some unique examples and viewed as the first point of reference prior to all other information.
6. It is very important to establish the correct polarity before connecting the fan to the power source. Positive (+) and Negative (-). Damage may be caused to the fans if connection is with reverse polarity, as there is no foolproof method to protect against such error.
7. Delta fans are not suitable where any corrosive fluids are introduced to their environment.
8. Please ensure all fans are stored according to the storage temperature limits specified. Do not store fans in a high humidity environment. We highly recommend performance testing is conducted before shipping, if the fans have been stored over 6 months.
9. Not all fans are provided with the Lock Rotor Protection feature. If you impair the rotation of the impeller for the fans that do not have this function, the performance of those fans will lead to failure.
10. Please be cautious when mounting the fan. Incorrect mounting of fans may cause excess resonance, vibration and subsequent noise.
11. It is important to consider safety when testing the fans. A suitable fan guard should be fitted to the fan to guard against any potential for personal injury.
12. Except where specifically stated, all tests are carried out at relative (ambient) temperature and humidity conditions of 25°C, 65%. The test value is only for fan performance itself.
13. Be certain to connect an "over 4.7μF" capacitor to the fan externally when the application calls for using multiple fans in parallel, to avoid any unstable power.

Cytron Technologies

# User's Manual

## V1.0

## May 2013



# Index



## 1.0 INTRODUCTION

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

**Features:**

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

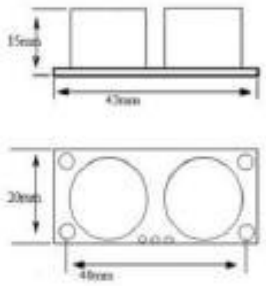
**2.0 PACKING LIST**

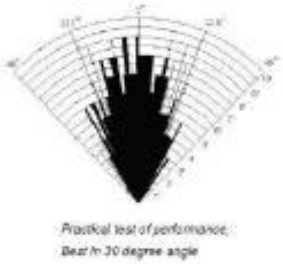

1. 1 x HC-SR04 module

### 3.0 PRODUCT LAYOUT


VCC = +5VDC
Trig = Trigger input of Sensor
Echo = Echo output of Sensor
GND = GND



*Practical test of performance.*
*Best in 30 degree angle*

## 4.0 PRODUCT SPECIFICATION AND LIMITATIONS

| Parameter | Min | Typ. | Max | Unit |
|---|---|---|---|---|
| Operating Voltage | 4.50 | 5.0 | 5.5 | V |
| Quiescent Current | 1.5 | 2 | 2.5 | mA |
| Working Current | 10 | 15 | 20 | mA |
| Ultrasonic Frequency | - | 40 | - | kHz |

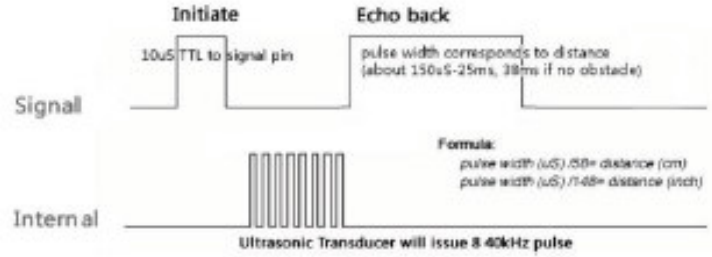
 

## 5.0 OPERATION

The timing diagram of HC-SR04 is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin.

Time = Width of Echo pulse, in uS (micro second)

- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340m/s



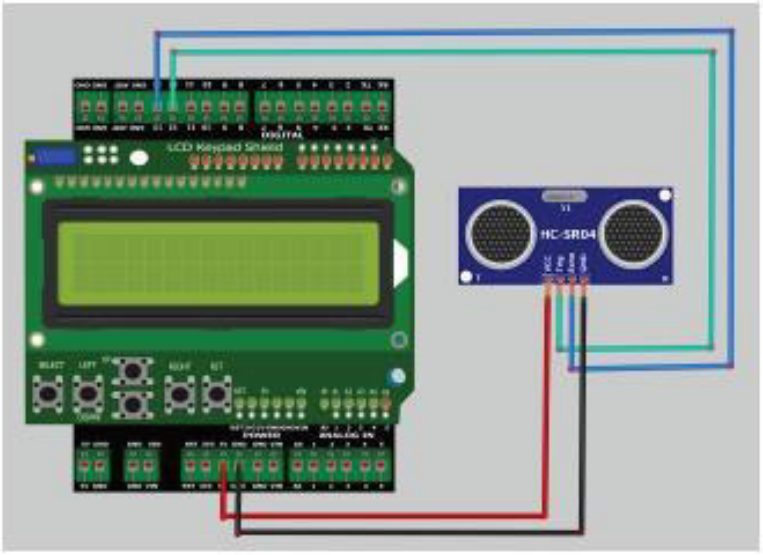

Note:

- Please connect the GND pin first before supplying power to VCC.
- Please make sure the surface of object to be detect should have at least 0.5 meter² for better performance.

## 6.0 HARDWARE INTERFACE

Here is example connection for Ultrasonic Ranging module to Arduino UNO board. It can be interface with any microcontroller with digital input such as PIC, SK40C, SK28A, SKds40A, Arduino series.



## 7.0 EXAMPLE CODE

This is example code Ultrasonic Ranging module. Please download the complete code at the product page.

```
#include "Ultrasonic.h"
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
Ultrasonic ultrasonic(12,13);

void setup() {
lcd.begin(16, 2);
lcd.setCursor(0, 0);
lcd.print("HC-SR04 testing..");
delay(1000);

}

void loop()
{
  //lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print(ultrasonic.Ranging(CM));
  lcd.print("cm  ");

  delay(100);
}
```

**8.0 WARRANTY**

- Product warranty is valid for 6 months.
- Warranty only applies to manufacturing defect.
- Damaged caused by miss-use is not covered under warranty
- Warranty does not cover freight cost for both ways.

*Prepared by*
***Cytron Technologies Sdn. Bhd.***
No. 16, Jalan Industri Ringan Permatang Tinggi 2,
Kawasan Industri Ringan Permatang Tinggi,
14100 Simpang Ampat,
Penang, Malaysia.

*Tel:* *+604 - 504 1878*
*Fax:* *+604 - 504 0138*

*URL:* www.cytron.com.my
*Email:* support@cytron.com.my
sales@cytron.com.my

# Dades dels anàlisis de funcionament

| Lectures del sensor HC-SR04 | | |
|---|---|---|
| valors | promig | error |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,95 | 75,3356597 | -0,61434028 |
| 75,62 | 75,3356597 | -0,28434028 |
| 75,95 | 75,3356597 | -0,61434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,59 | 75,3356597 | -0,25434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,55 | 75,3356597 | -0,21434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,55 | 75,3356597 | -0,21434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,55 | 75,3356597 | -0,21434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 76,07 | 75,3356597 | -0,73434028 |
| 75,55 | 75,3356597 | -0,21434028 |
| 76,05 | 75,3356597 | -0,71434028 |
| 75,55 | 75,3356597 | -0,21434028 |
| 76,07 | 75,3356597 | -0,73434028 |
| 75,5 | 75,3356597 | -0,16434028 |
| 75,6 | 75,3356597 | -0,26434028 |
| 75,52 | 75,3356597 | -0,18434028 |
| 75,62 | 75,3356597 | -0,28434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,66 | 75,3356597 | -0,32434028 |

| | | |
|---|---|---|
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,53 | 75,3356597 | -0,19434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,53 | 75,3356597 | -0,19434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,53 | 75,3356597 | -0,19434028 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,22 | 75,3356597 | 0,11565972 |

| | | |
|---|---|---|
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,95 | 75,3356597 | -0,61434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,57 | 75,3356597 | -0,23434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,55 | 75,3356597 | -0,21434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,64 | 75,3356597 | -0,30434028 |

| | | |
|---|---|---|
| 75,64 | 75,3356597 | -0,30434028 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,55 | 75,3356597 | -0,21434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,21 | 75,3356597 | 0,12565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,21 | 75,3356597 | 0,12565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,21 | 75,3356597 | 0,12565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,53 | 75,3356597 | -0,19434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,55 | 75,3356597 | -0,21434028 |

| | | |
|---|---|---|
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,55 | 75,3356597 | -0,21434028 |
| 76,07 | 75,3356597 | -0,73434028 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,53 | 75,3356597 | -0,19434028 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |

| | | |
|---|---|---|
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,62 | 75,3356597 | -0,28434028 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,62 | 75,3356597 | -0,28434028 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 76,07 | 75,3356597 | -0,73434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,22 | 75,3356597 | 0,11565972 |

| | | |
|---|---|---|
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,14 | 75,3356597 | 0,19565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,1 | 75,3356597 | 0,23565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,97 | 75,3356597 | -0,63434028 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,22 | 75,3356597 | 0,11565972 |
| 75,19 | 75,3356597 | 0,14565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,5 | 75,3356597 | -0,16434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,24 | 75,3356597 | 0,09565972 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,64 | 75,3356597 | -0,30434028 |
| 75,12 | 75,3356597 | 0,21565972 |
| 75,66 | 75,3356597 | -0,32434028 |
| 75,12 | 75,3356597 | 0,21565972 |