Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi:Grau en Enginyeria Electrònica Industrial i Automàtica

Títol: Plataforma estabilitzadora de càmeres

Document: 1. Memòria

Alumne: Bo Wei Lai

Tutor: Albert Figueras Coma
Departament: Enginyeria elèctrica, electrònica i automàtica
Àrea: Enginyeria de sistemes i automàtica

Convocatòria (mes/any) Setembre/2023

ÍNDEX

1. INTRODUCCIÓ

1.1. Antecedents

Actualment, al mercat es disposa d'una àmplia gamma de sensors i actuadors amb funcions molt diverses. Aquest, juntament amb plaques tipus Arduino, que amb velocitats de rellotge de 16 MHz disposen de velocitats de processament molt elevades i, per tant, permeten implementar multitud d'aplicacions de manera relativament senzilla i econòmica. D'altra banda, la creació de contingut audiovisual és una activitat molt popular avui en dia que requereix eines com els estabilitzadors de càmera, coneguts com a "gimbal".

1.2. Objectiu

L'objecte del treball és dissenyar una plataforma d'estabilització de càmeres portàtil de tres eixos fent servir un microcontrolador Arduino i el repertori de sensors i actuadors disponibles al mercat. Concretament, es faran servir mòduls combinats d'acceleròmetre i giroscopi digital per captar el moviment i tres servomotors que compensin el moviment mesurat en cada eix. D'aquesta manera es vol controlar la posició de la plataforma de tal manera que aquesta sigui capaç de compensar oscil·lacions provocades pel moviment i així aconseguir una imatge més estable.

1.3. Abast

El treball abastarà la selecció dels components, disseny del circuit electrònic i PCB, la programació necessària pel funcionament del dispositiu, així com la carcassa física impresa en 3D d'aquest. Es farà un estudi de l'autonomia del dispositiu.

2. ESTAT DE L'ART

Un gimbal és un dispositiu mecànic que permet mantenir l'orientació d'un objecte, com una càmera, un sensor o qualsevol càrrega útil, de manera estable i constant en relació amb el seu entorn, fins i tot quan el suport o la plataforma on es troba experimenten moviments. Actualment, els gimbals són essencials per a l'estabilització en una varietat d'aplicacions, com la fotografia, la cinematografia, la indústria aeroespacial, la navegació i més.

La història dels gimbals es remunta a l'antiguitat, on s'utilitzaven els primers gimbals mecànics accionats per contrapesos per mantenir l'estabilitat d'instruments de navegació marítima les primeres versions d'aquests dispositius per mantenir instruments nàutics anivellats en vaixells i per orientar dispositius en la navegació astronòmica.

L'aplicació moderna dels gimbals en l'estabilització de càmeres i sensors va començar a desenvolupar-se a mitjans del segle XX a la indústria militar i aeroespacial. A mesura que la tecnologia es miniaturitzava i reduïa el seu preu, aquests dispositius s'han tornat més accessible i ha donat lloc al desenvolupament de gimbals de diferents tipus, més compactes i eficients per a aplicacions comercials i de consum.

Figura 1. Gimbal electrònic de la casa DJI

El màxim exponent d'això són els gimbals electrònics, com el mostrat a la figura anterior. Aquests utilitzen sensors i motors per detectar i corregir moviments no desitjats. Són comuns en càmeres d'acció, de telèfons mòbils i equips de cinema gràcies a les múltiples funcionalitats que ofereixen com ara capacitats avançades d'estabilització que permeten eliminar fins i tot petites oscil·lacions, capacitat de seguiment autònom d'objectes per tal d'enfocar de manera automàtica objectes en moviment i versatilitat d'ús, ja que poden

funcionar amb càmeres de telèfon mòbil, d'acció o fins i tot càmeres tipus reflex, tot amb un dispositiu compacte i portàtil gràcies a la bateria integrada.

En resum, els gimbals electrònics representen l'estat actual de la tecnologia d'estabilització, oferint un control i una estabilitat excepcionals en una àmplia gamma d'aplicacions, des de la producció cinematogràfica professional fins a la creació de contingut persónal amb dispositius mòbils.

3. CIRCUIT ELECTRÒNIC

Per tal de complir amb l'objecte d'aquest projecte, es requereix un circuit electrònic que permeti determinar l'angle girat en cada eix del dispositiu i corregir posteriorment aquesta rotació, tot això alimentat per una bateria recarregable. Es planteja el següent diagrama de blocs pel circuit.

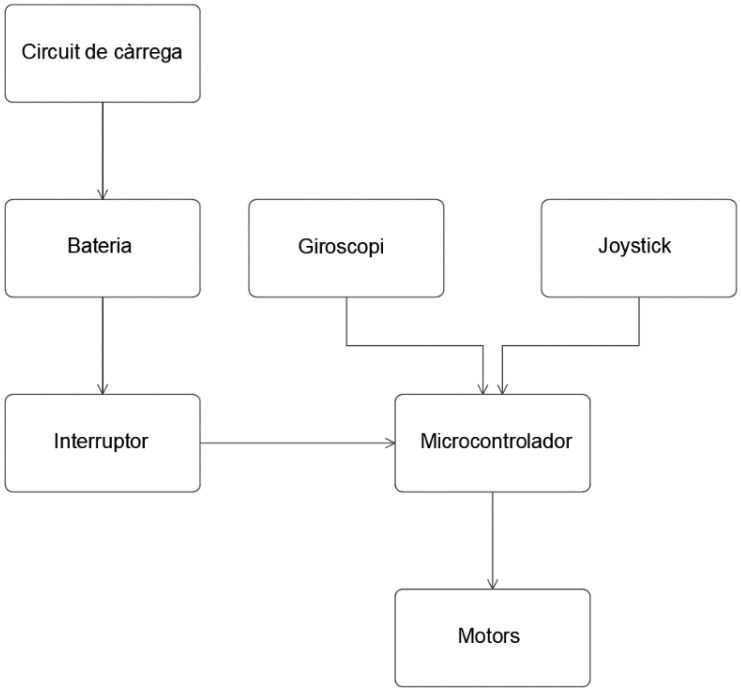

Figura 2. Diagrama de blocs del gimbal.

Resumint la figura anterior, el circuit consta de diferents etapes, alimentant tot el conjunt es disposa d'una bateria que es recarregarà mitjançant un circuit de càrrega de bateries alimentat de manera externa. L'alimentació és aïllada de la resta del circuit via un interruptor. Quan està habilitat, el microcontrolador rep informació del gir a compensar provinent del giroscopi i d'un joystick i envia les instruccions per tal de compensar el gir de cada eix al motor corresponent.

3.1. Actuadors

Una de les parts fonamentals de l'estabilitzador són els actuadors que corregiran el moviment rotatiu del marc del gimbal. Per fer-ho es necessitaran tres motors, un per cada eix de rotació. S'ha optat per la tecnologia de servomotors, que permet controlar de manera precisa i relativament senzilla la posició de l'eix del servomotor.

Concretament, es tracta del model DS3230 PRO, les seves característiques es detallen a la taula següent.

Especificacions servomotor DS3230 Pro	
Tensió de funcionament (V)	4,8-7,2
Rang de moviment (º)	270
Senyal de control	PWM
Ample de banda del senyal (µs)	2000
Senyal per a posició 0º (µs)	500
Velocitat, alimentat a 7,2V (ms/º)	2

Taula 1. Resum de les característiques elèctriques del servomotor DS3230 PRo

Com s'ha remarcat anteriorment, un servomotor permet posicionar de manera precisa el motor, per fer-ho es fa servir un senyal PWM que indica la posició desitjada.

El PWM és una tècnica que permet transmetre un senyal analògic mitjançant un senyal digital. Aquest senyal digital disposa d'un període fix i dins d'aquest període es regula el temps que el senyal roman en estat actiu i baix, variant el temps el qual el senyal roman en estat alt i baix. D'aquesta manera el valor mitjà del senyal pot oscil·lar entre el valor alt i baix del senyal, en funció del temps actiu d'aquest. La següent figura mostra un senyal PWM que al llarg de tres períodes canvia el temps que està en estat alt.

Figura 3. Senyal de control PWM d'un servomotor.

La relació entre el temps en estat alt i el període del senyal es coneix com a duty cycle, mesurat en percentatge, segons la següent fórmula.

$$\text{Duty cycle} = \frac{\text{Temps actiu}}{\text{Període senyal}} \qquad \text{(Eq.1)}$$

Aplicat al control de la posició dels servomotors d'aquest projecte, el període del senyal PWM és de vint milisegons. Dins d'aquest període, l'electrònica del servomotor codifica un senyal entre cinc-cents i dos mil cinc-cents microsegons en estat alt de manera lineal entre zero i dos-cents setanta graus. Per tal de saber la posició de l'eix del motor, aquest està

connectat internament a un potenciòmetre que va canviant el seu desplaçament amb el gir de l'eix. Llavors, la posició de l'eix del servomotor seleccionat en funció de la durada del pols és el descrit a la següent figura.

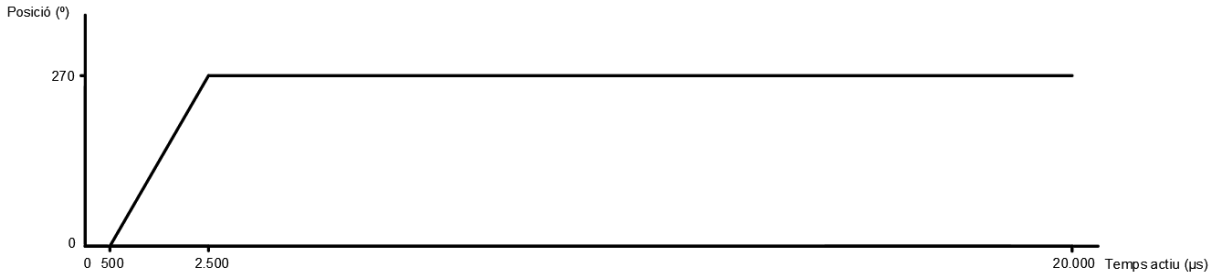

Figura 4.Posició de l'eix del servomotor en funció del puls PWM

La figura anterior descriu el posicionament de l'eix del servomotor en funció del temps actiu del pols PWM enviat. S'observa una zona morta entre els zero i cinc-cents microsegons, l'eix roman a la posició de zero graus. Per polsos superiors a dos mil cinc-cents microsegons l'eix es queda a la posició de dos-cents setanta graus. Entre dos límits a partir dels quals la sensibilitat és zero, hi ha la regió de treball del servomotor, dins d'aquesta regió la posició de l'eix ve determinada per la següent equació.

$$\text{Posició}=0{,}135\cdot\text{Temps actiu}-67{,}5 \qquad \text{(Eq.2)}$$

L'equació anterior correspon a l'equació de la recta a la zona de treball dels servomotors, el valor de 0,135 és definit pel pendent d'aquesta. El temps actiu mesurat en microsegons està limitat al rang permès pels servomotors, cinc-cents i dos mil cinc-cents. Quan aquest val cinc-cents, el valor de la posició dels servomotors és igual a zero, de tal manera que l'ordenada a l'origen ha de valdre 67,5.

En relació amb el projecte, la tria d'aquests tipus de motors i d'aquest model en concret ve determinada per la facilitat de controlar la posició de l'eix, en incloure encoders i drivers de potència interns. També és remarcable la velocitat de la qual disposen, segons el full de característiques, el model triat gira a raó de cinc-cents graus cada segon en el cas de ser alimentat al màxim del rang de tensió tolerat. El servomotor inclou de sèrie un seguit de suports cargolables a l'eix de sortida del servomotor.

3.2. Sensors

Per tal de determinar el gir en cada eix del gimbal, es requereix un sensor que sigui capaç de llegir la velocitat angular de cada eix, és a dir, un giroscopi. Es tria per tal funció un sensor MPU6050, integrat en un mòdul GY-521 que facilita la seva integració posterior a la PCB. La taula següent detalla les característiques generals del mòdul. El MPU6050 és una unitat IMU de tipus MEMS de sis eixos de mesura, tres d'acceleració lineal i tres de velocitat angular, s a dir, combina un giroscopi i un acceleròmetre en el mateix encapsulat.

Un sensor MEMS és un dispositiu que combina components mecànics i elèctrics a una escala molt petita, generalment en el rang de micròmetres a mil·límetres. Aquests dispositius es fabriquen utilitzant tecnologies de microfabricació i permeten la detecció i la mesura de diferents tipus de magnituds físiques, com ara acceleració, pressió, temperatura, moviment angular, entre d'altres.

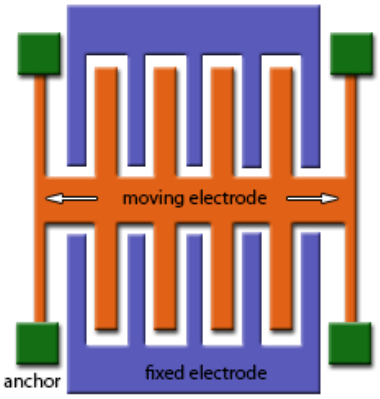

Figura 5. Representació de l'estructura microscòpica d'un sensor MEMS típic.

L'estructura típica dels sensors MEMS consisteix en dos elèctrodes, un fix i un altre mòbil separats per certa distància, generant certa capacitància. En el cas que ens ocupa, quan hi ha un desplaçament, l'elèctrode mòbil es desplaça respecte a la seva posició de repòs, canviant la capacitància que genera conjuntament amb l'elèctrode fixe. Aquest canvi és captat internament pel sensor, amplificat i processat per tal de mesurar quantitativament la magnitud a mesurar. Un sol sensor MEMS és capaç de mesurar una sola magnitud, per exemple, per mesurar l'acceleració lineal en tres eixos són necessaris cel·les com les descrites a la figura 5, una en cada eix de mesura.

Respecte al mòdul GY-521, les seves característiques generals s'especifiquen a la següent taula.

Especificacions mòdul GY-521	
Tensió de funcionament (V)	3,3-5
Protocol de comunicació	I2C
Consum (mA)	3,5
Nº eixos acceleròmetre	3
Sensibilitat de l'acceleròmetre (g)	±2, ±4, ±8, ±16
Nº eixos giroscopi	3
Sensibilitat del giroscopi (º/s)	±250, ±500, ±1000, ±2000
Resolució (bit)	16

Taula 2. Especificacions del mòdul GY-521

Destaquem que es tracta d'un sensor digital que envia la informació mitjançant el bus I2C, aquesta informació es codifica en 2 bytes d'informació per cada eix. És important remarcar també que segons el datasheet del sensor aquest pot tenir cert offset i deriva a la mesura al llarg del temps, cosa que introdueix cert error a les mesures. És, per tant, indispensable realitzar un correcte calibratge del sensor per tal de minimitzar l'offset.

Es contarà també amb un mecanisme per tal de dirigir la càmera a la direcció que es desitgi i poder corregir part de l'error que introdueixi el sensor, per fer-ho s'ha triat implementar un mòdul joystick. Aquest mòdul disposa de dos eixos de control independents i un polsador accionat mitjançant la palanca de control, tot en un conjunt de dimensions reduïdes, prèviament mecanitzat, cosa que facilita la seva integració mecànica, i un preu reduït.

El joystick consisteix en dos potenciòmetres connectats a la tensió de treball del microcontrolador i massa. Els cursors dels potenciòmetres estan connectats elèctricament a dos pins pels quals es transmet el senyal i acoblats mecànicament a una palanca en els eixos x i y, al pivotar aquesta palanca, aquesta desplaça el cursor d'un o dels dos potenciòmetres, en funció de la direcció d'accionament de la palanca. Al canviar el desplaçament del cursor, també canvia la tensió que llegeix el microcontrolador pels pins.

Internament, el senyal del polsador està connectada directament a la massa del mòdul, per tant és necessari per tal que el senyal sigui processat adequadament integrar una resistència de pull-up a la sortida. En aquest sentit el mòdul ja inclou un pad per tal de soldar una resistència SMD que funcioni com a pull-up.

3.3. Microcontrolador

La part central del circuit correspon al microcontrolador, aquest rep les dades del sensor i les processa per donar les ordres de compensació als motors. S'ha triat per a aquesta funció una placa Arduino ProMini, una de les opcions més compactes de la família de plaques Arduino. Com la majoria d'aquestes, es basa en el microcontrolador Atmega328P, s'ha triat la versió que treballa a cinc volts i setze megahertzs de velocitat. La taula següent resumeix les seves característiques principals.

Especificacions Arduino ProMini	
Tensió de funcionament (V)	5,1-12
Protocol de comunicació	I2C,SPI,UART
Velocitat de processament (MHz)	16
Nº pins GPIO	12
Nº pins PWM	6
Nº entrades analògiques	8
Resolució ADC (bit)	10
Consum (mA)	27
Dimensions (mm)	34x17

Taula 3. Característiques del Arduino ProMini

De la taula anterior veiem que la placa incorpora un regulador de tensió que permet alimentar la placa amb tensions fins a dotze volts, el microcontrolador és compatible amb comunicacions I2C, SPI i UART, tot i que no disposa de cap connector USB, cosa que redueix la mida de la placa però obliga a fer servir un adaptador UART TTL a l'hora de carregar el programa. Disposa de dotze pins configurables com a entrades o sortides digitals, dels quals sis es poden configurar com a sortida PWM, a més de vuit pins dedicats a entrades analògiques amb deu bits de resolució. Un dels punts més destacables a l'hora de triar aquesta placa en específic és el seu baix consum i les seves mides molt compactes, juntament amb les característiques mencionades anteriorment la tornen en una opció ideal per aplicacions d'electrònica portàtil com el gimbal.

La taula següent mostra l'ús final de cada pin de la placa Arduino

Pin Arduino	Destí
DTR	NC
TX1	NC
RX1	NC
VCC1	NC
GND1	NC

Taula 4. Llistat de pins del microcontrolador i funcions assignades.

Pin Arduino	Destí
GND2	NC
RAW	7,4V
GND3	GND
RST1	NC
VCC	Joystick_5V,MPU6050_5V
A5	SDA
A4	SCL
A3	NC
A2	NC
A1	Joystick_X
A0	Joystick_Y
13	NC
12	Joystick_Switch
11	ServoX
10	ServoY
9	ServoZ
8	NC
7	NC
6	NC
5	NC
4	NC
3	NC
2	NC
GND4	NC
RST2	NC
RX2	NC
TX2	NC

Taula 5. Llistat de pins del microcontrolador i funcions assignades.

Es veu que majoritàriament els pins no es fan servir, per tant es podrien fer ampliacions de cara a ampliacions futures.

3.4. Alimentació. Estudi d'autonomia

És important triar una font d'alimentació adient per alimentar el conjunt, aquesta ha de ser compacta i amb possibilitat de ser recarregada. La taula següent resumeix els requeriments de tensió i consum de cada element.

Element	Tensió (V)	Consum (mA)
DS3220 PRO	7,4	1500
GY-521	5	3,5
Joystick	5	1
Arduino ProMini	5	27

Taula 6. Requeriments d'alimentació del circuit

Es veu que l'element amb més consum són els motors, alimentats a 7,4 volts i consumint mil cinc-cents miliampers cada un, resultant en un consum combinat de quatre mil cinc-cents miliampers pels motors. La resta d'elements són alimentats a cinc volts i tenen un consum ínfim respecte als motors. Globalment, hi ha un consum de quatre mil cinc-cents trenta i un amb cinc miliampers.

Tenint en compte aquests requisits es tria com a font d'alimentació un bloc de dues bateries electrolítiques tipus 18650 en sèrie. Cada bateria individual té una tensió nominal de 3,7 volts, llavors conjuntament resulta en 7,4 volts necessaris per alimentar els servomotors, l'Arduino disposa d'un regulador intern que permet alimentar-lo amb aquesta tensió sense problemes. Finalment, la part de sensórica es pot alimentar mitjançant els 5 volts que genera l'Arduino.

Les bateries seleccionades poden proporcionar fins a sis ampers d'intensitat, prou per alimentar els tres motors a la vegada, i tenen una capacitat nominal de dos mil cinc-cents miliampers hora. Amb aquesta informació i suposant un consum màxim continu de quatre mil cinc-cents trenta un miliampers, es pot estimar la duració de la bateria segons la següent equació.

$$\text{Temps de funcionament} = \frac{2500\ \text{mAh}}{4531{,}5\ \text{mA}} = 0{,}55\ \text{hores} = 33\ \text{minuts} \qquad \text{(Eq.3)}$$

3.4.1. Circuit de recàrrega

A l'hora de recarregar la bateria, s'implementa un circuit integrat pensat específicament per aquesta tasca, l'integrat en qüestió és el LT1510. El circuit necessari per implementar l'integrat és el següent.

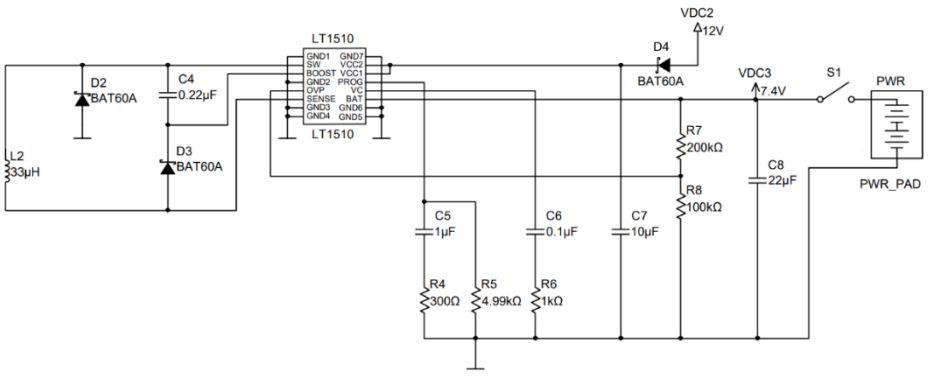

Figura 6. Esquema elèctric del circuit de càrrega de bateries

Aquest integrat permet recarregar diferents tipologies de bateries com ara NiCd, NiMH i de liti. L'integrat s'alimenta entre onze i vint volts i permet programar dos paràmetres canviant el valor de dues resistències del circuit. El primer paràmetre és la intensitat de càrrega de la bateria, la qual determina la velocitat de càrrega de la mateixa. Aquesta pot valdre un màxim de mil cinc-cents miliampers, aquesta intensitat és programable en funció del valor de R5, segons la fórmula del datasheet.

$$I_{bat}=\frac{2,465}{R5}\cdot 2000 \qquad \text{(Eq.4)}$$

Llavors, si fixem la intensitat de càrrega en un amper, el valor de R5 necessari seria:

$$R5=\frac{2,465}{1}\cdot 2000=4,93k\Omega \qquad \text{(Eq.5)}$$

Amb aquest resultat, tirem el valor normalitzat més proper, 4,99 kΩ.

El segon paràmetre a programar és la tensió màxima que es vol assolir. Cal remarcar que el carregador funciona en dues tapes diferenciades, una a corrent constant, en la qual s'injecta corrent de manera constant a les bateries a mesura que aquestes augmenten la seva tensió, i una de tensió constant, en la qual quan s'arriba a un límit de tensió programable, es deixa d'injectar corrent, simplement mantenint la tensió als extrems de la bateria. Aquestes dues etapes es veuen explicades a la figura següent.

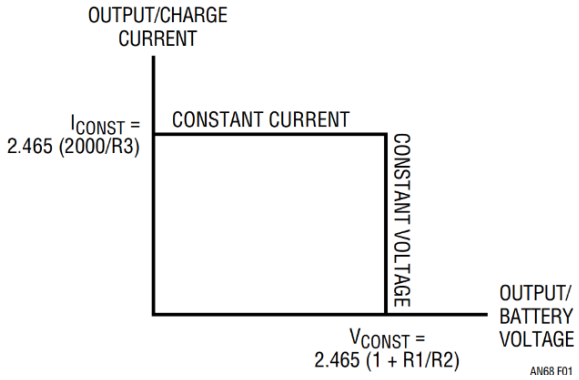

Figura 7. Etapes de càrrega del LT1510

Aquesta tensió llindar depèn de la resistència R7, segons la fórmula:

$$R7=\frac{100000\cdot(V_{bat}-2,465)}{2,465+100000\cdot 0,05\cdot 10^{-6}}=4,93k\Omega \qquad \text{(Eq.6)}$$

Per tant, si fixem la tensió màxima amb un valor igual al del nominal de les dues bateries en sèrie, 7,4 volts, el valor de la resistència R7 seria:

$$R7=\frac{100000\cdot(7,4-2,465)}{2,465+100000\cdot0,05\cdot10^{-6}}=200k\Omega \quad \text{(Eq.7)}$$

El díode D4 protegeix l'alimentació de l'integrat davant de tensions inverses, es tria un díode Schottky BAT60A, capaç de suportar deu volts en tensió inversa, amb una caiguda de tensió de tres-cents milivolts i una intensitat de treball màxima admissible de tres ampers.

La resta de components responen a recomanacions del mateix fabricant per tal d'assegurar l'estabilitat del funcionament.

Amb una capacitat nominal de les bateries de 2500 mAh i una velocitat de càrrega de 1A, podem estimar la velocitat de càrrega de les bateries segons la següent fórmula.

$$\text{Temps de càrrega}=\frac{2500\ mAh}{1000\ mA}=2,5\ \text{hores} \quad \text{(Eq.8)}$$

Durant aquestes dues hores i mitja, la potència consumida pel carregador vindria definida per la següent fórmula.

$$P_{OUT}=V_{BAT}\cdot I_{BAT} \quad \text{(Eq.9)}$$

Per tant, si la tensió de sortida està programada a 7,4 volts i la intensitat de càrrega és d'un amper, la potència consumida serà de 7,4 watts.

3.4.2. Convertidor boost

Com s'ha mencionat abans, el LT1510 opera entre onze i vint volts. Amb això en ment es decideix alimentar-lo a dotze volts. L'opció més convenient seria implementar un connector jack de potència i alimentar-lo mitjançant un carregador extern amb transformador integrat, però per tal d'estandarditzar cablejat i carregadors, s'opta per carregar el dispositiu amb un carregador de cinc volts mitjançant un cable USB tipus C. A conseqüència d'això, cal implementar també un convertidor tipus boost que elevi la tensió de cinc a dotze volts. Per fer-ho es tria un integrat, el LT1070 també de la casa Linear Tecnology. Aquest és un

controlador commutat amb duty cicle variable, pot funcionar per funcions de convertidor buck o boost en funció del circuit implementat, per un convertidor boost el circuit és el següent.

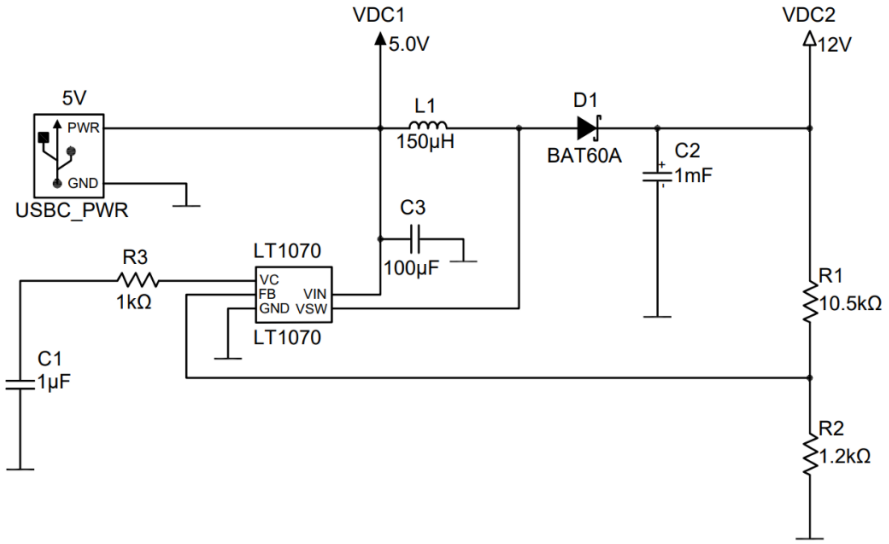

Figura 8. Figura 9. Esquema elèctric del convertidor boost

En aquesta configuració el LT1070 pot treballar entre tres i cinquanta volts, per ajustar la tensió de sortida es fa un divisor de tensió amb R1 i R2, segons la fórmula següent.

$$R1=R2\cdot\left(\frac{V_{OUT}}{1,244}-1\right) \quad \text{(Eq.10)}$$

Fixem R2 en un valor arbitrari de 1,2kΩ. Si volem una tensió de sortida de dotze volts per alimentar el carregador de bateries, el valor de R1 necessari és:

$$R1=1200\cdot\left(\frac{12}{1,244}-1\right)=10,4k\Omega \quad \text{(Eq.11)}$$

Agafem el valor normalitzat més proper, 10,5 kΩ

El següent element a dimensionar és la inductància, el valor de la qual determinarà la potència màxima de sortida i l'arrissat del corrent de sortida. Segons el datasheet el seu valor ve determinat per la següent fórmula

$$L1=\frac{V_{IN}\cdot(V_{OUT}-V_{IN})}{\Delta I\cdot f\cdot V_{OUT}} \quad \text{(Eq.12)}$$

Si la tensió d'entrada és de cinc volts, la sortida de dotze volts, la freqüència de commutació de l'integrat és de quaranta kilohertzs i l'arrissat a la sortida és de mig amper, la inductància ha de prendre un valor marcat per la següent fórmula.

$$L1=\frac{5\cdot(12-5)}{0{,}5\cdot 40\cdot 10^{3}\cdot 12}=146\mu H \qquad \text{(Eq.13)}$$

Normalitzem el valor a 150µH, a partir d'aquest valor es pot calcular segons la fórmula següent, segons el datasheet.

$$P_{MAX}=V_{IN}\cdot\left(I_P-\frac{V_{IN}(V_{OUT}-V_{IN})}{2\cdot L\cdot f\cdot V_{OUT}}\right)\cdot\left(1-\frac{I_P\cdot R}{V_{IN}}+\frac{I_P\cdot R}{V_{OUT}}\right) \qquad \text{(Eq.14)}$$

On el valor de Ip és corrent màxim que admet l'element commutat de l'integrat, cinc ampers per aquest model, i R és la resistència d'aquest element commutat quan està en estat de conducció, dos-cents miliohms, per aquest model. Per tant, la potència màxima de sortida és la següent.

$$P_{MAX}=5\cdot\left(5-\frac{5\cdot(12-5)}{2\cdot 150\cdot 10^{-6}\cdot 40\cdot 10^{3}\cdot 12}\right)\cdot\left(1-\frac{5\cdot 0{,}2}{5}+\frac{5\cdot 0{,}2}{12}\right)=21W \qquad \text{(Eq.15)}$$

Aquesta potència màxima que pot subministrar és major a la que consumeix el carregador de bateries. Tenint en compte que el carregador de bateries consumeix 7,4 watts de potència, això implica un consum de corrent a la sortida del convertidor boost de sis-cents miliampers.

Els condensadors C3 i C2 compensen l'arrissat d'entrada i sortida respectivament. Concretament, el valor de l'arrissat en funció del valor del condensador C2 és el següent.

$$V_{PP}=\frac{V_{OUT}\cdot I_{OUT}}{C2\cdot f\cdot(V_{IN}+V_{OUT})} \qquad \text{(Eq.16)}$$

Fixem el valor del condensador en un milifarad, ja que els valors elevats de capacitància proporcionen valors d'arrissat menors. Llavors l'arrissat de sortida serà:

$$V_{PP}=\frac{5\cdot 0{,}600}{0{,}001\cdot 40\cdot 10^{3}\cdot(5+12)}=4\ mV \qquad \text{(Eq.17)}$$

Finalment, els elements restants són components recomanats pel propi fabricant per assegurar el correcte funcionament del circuit.

3.4.3. Indicador de càrrega

El LT1510 no disposa de cap manera d'indicar visualment la finalització del cicle de càrrega, és a dir, que ha arribat al nivell de tensió màxima programada. És per això que es vol implementar un circuit per donar a l'usuari un avís visual que el dispositiu està carregat. Aquest sistema s'implementarà mitjançant amplificadors operacionals, concretament es tria el ADA4000-4, que es tracta d'un OPAMP amb capacitats dual i single supply, rail-to-rail, amb baix offset. L'ADA4000 ve en diferents encapsulats amb un, dos o quatre operacionals integrats, es tria la versió de quatre operacionals, dels quals se n'aprofiten tres, segons l'esquema següent.

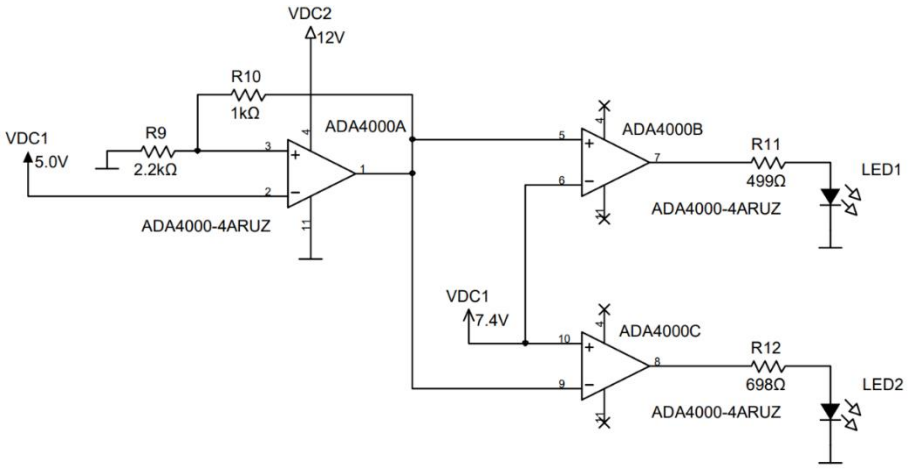

Figura 10. Circuit electrònic del ADA4000-4

Tots els OPAMPs són alimentats mitjançant els dotze volts del LT1070 i massa, funcionant en configuració single supply.

El circuit consisteix en dos comparadors, aquests simplement comparen el valor de tensió de la bateria amb una referència, 7,27 volts. Quan se supera aquest llindar, s'il·lumina un LED blau, i mentre no se superi, un LED vermell roman encès. Remarcar que els dos comparadors són complementaris, és a dir, quan un dona senyal alt, l'altre dona senyal baix, això fa que quan estigui carregant sempre hi ha un LED encès, indicant que la bateria està carregada o no.

La referència de 7,27 volts s'aconsegueix a partir dels cinc volts del carregador tipus C, fent servir un dels operacionals en configuració no inversora. La tensió de sortida de l'amplificador no inversor ve determinada per la següent fórmula.

$$V_{OUT}=V_{IN} \cdot (1+\frac{R10}{R9}) \quad \text{(Eq.18)}$$

Per tant, per valors de Vout de 7,27 volts i Vin de 5 volts, fixem la resistència R9 en 2,2 kiloohms, el valor de la resistència R10 és igual a 1 kiloohms.

3.5. Placa de circuit imprès

Per integrar tots els components descrits anteriorment és necessari d'una PCB, aquesta és necessari que sigui de les mides més compactes possibles, per tal de facilitar la seva posterior integració a la carcassa. Finalment, es col·loquen tots els components en una PCB de fibra de vidre de doble cara de dimensions trenta-cinc per cent mil·límetres.

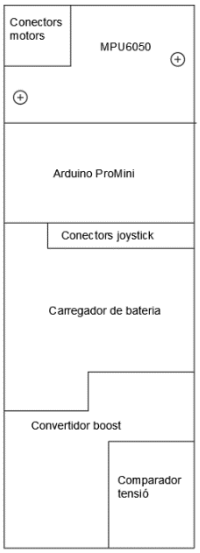

Figura 11. Distribució general dels components sobre la PCB

Tots els components se situen a la cara superior de la placa, seguint la distribució mostrada a la figura anterior, la cara inferior es dedica exclusivament a situar un pla de massa que ajuda a reduir possibles efectes del soroll ambiental i a reduir la temperatura. A part del pla de massa, per la part inferior es passen certes pistes per evitar encreuaments a la part superior. Les pistes tenen un ample de trenta mills, o mil·lèsimes de polzada, per les pistes de senyal i cinquanta mils per pistes de potència.

4. CARCASSA

La carcassa física que albergarà tots els components es dissenya amb el programa Fusion360, propietat d'Autodesk i s'imprimeix en 3D. Consisteix en un total de cinc peces, dos que alberguen la placa, bateries i primer motor, dos més pels dos motors restants i una serveix per acoblar la càmera o objecte a estabilitzar.

La primera peça és la base, aquesta disposa d'un espai reservat pel portapiles amb les dues bateries, per sobre d'aquest espai es situa la PCB, que queda cargolada per via de dos forats. També disposa de quatre forats addicionals per poder acoblar amb la coberta superior.

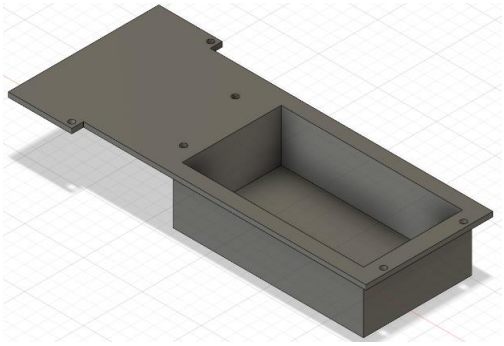

Figura 12. Base de la carcassa.

La carcassa superior disposa d'un orifici en el qual s'incorpora un interruptor que engega el dispositiu, quatre forats per fixar el mòdul joystick i obertures per endollar el carregador USB tipus C i veure l'estat dels LEDs. A la part superior s'hi situa el servomotor que controla l'angle de yaw.

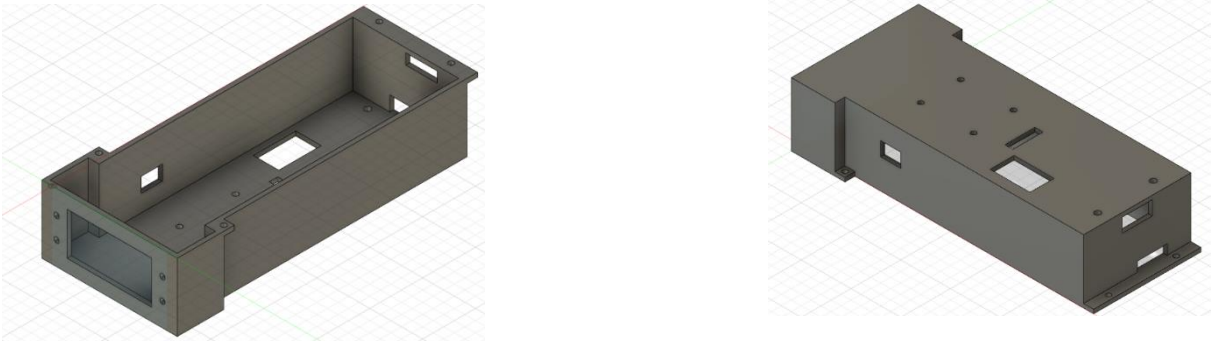

Figura 13. Coberta de la carcassa.

Al motor de roll se li acobla un suport com el de la figura següent, per poder fixar el següent servomotor, que controla l'angle de pitch.

Figura 14. Braç dels servos del roll i pitch.

Alhora aquest segon servomotor també te acoblat un braç idèntic on va situat l'últim motor que controla l'angle de yaw. Aquest últim motor porta un suport circular diferent, detallat a la figura següent.

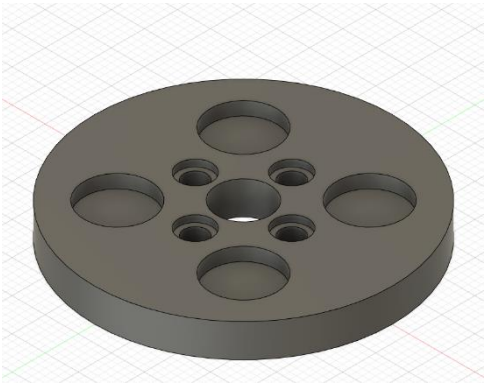

Figura 15.Suport dels imants.

Aquest suport disposa de quatre ranures per integrar imants de neodimi, que juntament amb una placa ferromagnètica adhesiva permetran fixar la càmera al gimbal.

5. DETERMINACIÓ DE L'ORIENTACIÓ

La part fonamental que es necessita per a que el gimbal funcioni adequadament és que es pugui determinar efectivament l'orientació respecte a un sistema de referència, en aquest cas el sistema de referència és sempre la posició inicial del gimbal en el moment de l'engegada, ja que el MPU6050 només captura acceleracions lineals i velocitats angulars, no posicions absolutes. Definim els eixos de gir i els seus sentits segons l'il·lustrat a la figura següent.

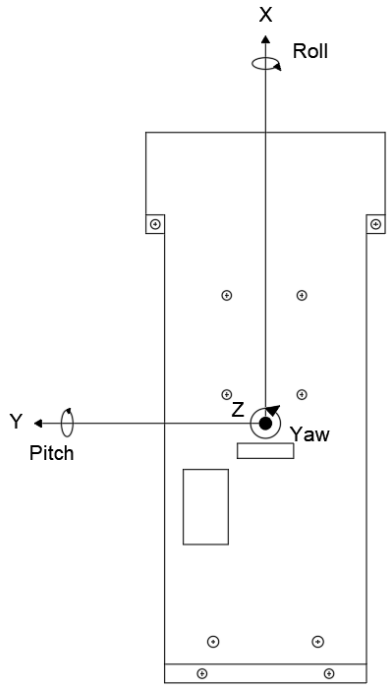

Figura 16. Eixos i sentit de gir del gimbal

Vist frontalment, l'eix x travessa el gimbal longitudinalment, l'angle rotat al voltant d'aquest eix és l'angle de roll, l'eix y travessa lateralment el dispositiu i la rotació respecte a l'eix és l'angle de pitch i finalment l'eix z, que articula l'angle de yaw. Els sentits positius de cada eix linealment són, cap a dalt, cap a l'esquerra i cap a l'usuari, respectivament, i pel que fa a les rotacions, es té en compte la regla de la mà dreta pel sentit positiu de les rotacions.

Un cop fixat el marc de referència, el sensor MPU6050 dona unes mesures de velocitat angular. Amb aquestes dades podem calcular la rotació en cada eix segons la següent fórmula.

$$\sigma = \sigma_{anterior} + \omega \cdot \Delta t \qquad \text{(Eq.19)}$$

D'aquesta manera coneixent la velocitat angular a intervals regular podem saber la rotació de cada eix en un moment determinat.

És per tant indispensable que el sensor proporcioni dades fiables i ajustades a la realitat, d'aquí sorgeix un problema, ja que el giroscopi del MPU6050 presenta a la sortida de dades tant un offset com soroll d'alta freqüència, com es mostra a la següent figura.

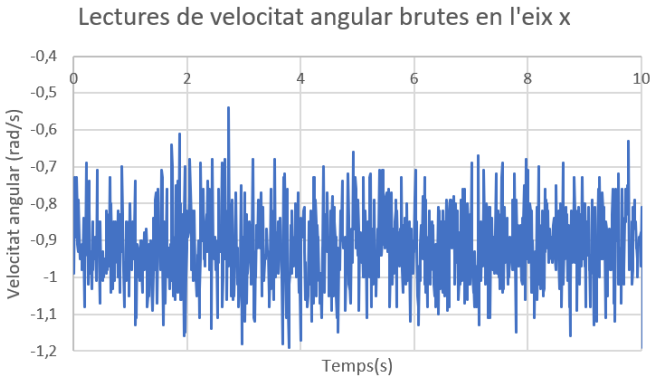

Figura 17. Lectures de velocitat angular en l'eix x.

Les lectures de la figura anterior corresponen a les de l'eix x, preses a intervals de deu milisegons amb el sensor en estàtic. Ressalta la problemàtica comentada anteriorment. L'error causat pel soroll i principalment per l'offset causa que amb el temps la rotació calculada es desviï significativament de la realitat, com s'il·lustra a la figura següent.

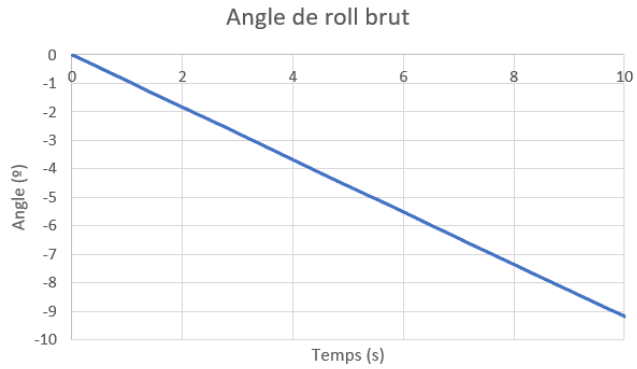

Figura 18. Angle de roll calculat amb el sensor estàtic.

Es veu que en un deu segons l'angle de roll calculat s'ha desviat nou graus del seu valor real. El comportament dels altres eixos de gir és semblant, canviant els valors d'offset.

És per tant necessari corregir aquest comportament, això es pot fer aprofitant eines integrades dins del sensor en el cas del soroll i per programació externa per corregir l'offset.

El sensor MPU6050 integra la possibilitat de filtrar el senyal de sortida tant del giroscopi com de l'acceleròmetre mitjançant un filtre pas-baix d'amplada de banda configurable, això permet, si més no, alleujar el problema del soroll. Si implementem un filtre pas baix d'amplada de banda igual a vint hertzs, el resultat és el següent.

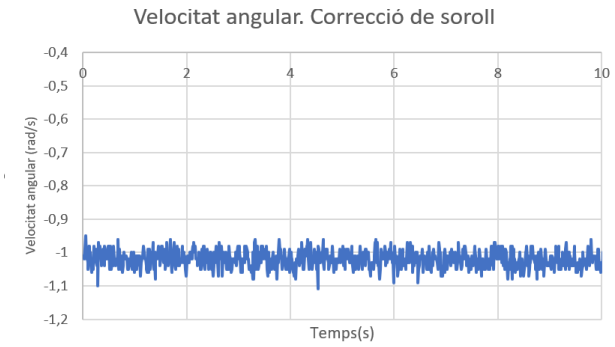

Figura 20. Correcció de soroll aplicant un filtre pas baix a l'eix x.

Observem que l'efecte del soroll és molt més reduït comparativament amb el senyal sense filtrar, ja que passem d'oscil·lacions de 0,6 radians per segons a oscil·lacions de 0,2 radians segons. Això ens permet eliminar la seva interferència a l'hora de calcular l'angle posteriorment i estimar l'offset de manera més precisa en -1 radians per segons. Si apliquem també aquest offset al senyal filtrat, obtenim els següents resultats de velocitat.

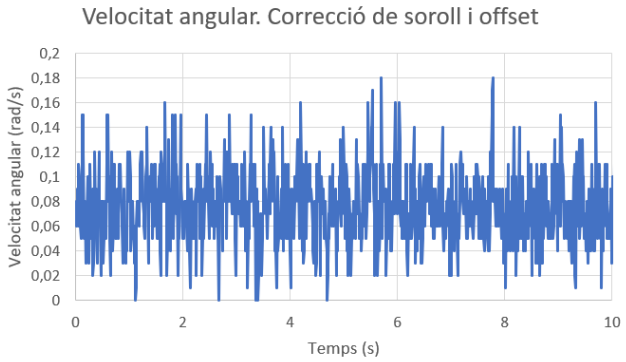

Figura 21. Correcció de soroll i offset a l'eix x

Veiem que tot i haver aplicat l'offset d'u, el senyal no queda centrat realment sobre zero, això ve pel fet que el giroscopi del sensor MPU6050 no té un offset fix e invariable, sinó que aquest paràmetre oscil·la indeterminadament dins un rang de valors que, a més a més pot ser afectat per condicions de temperatura, característiques de l'alimentació del sensor o acceleracions lineals que pateix el mòdul, segons el datasheet. Això fa que sigui difícil trobar un valor d'offset exacte que serveixi per cada vegada que es fa servir el giroscopi, cosa que

provoca que, inevitablement, la rotació calculada es desviï de la realitat, com s'il·lustra a la figura següent.

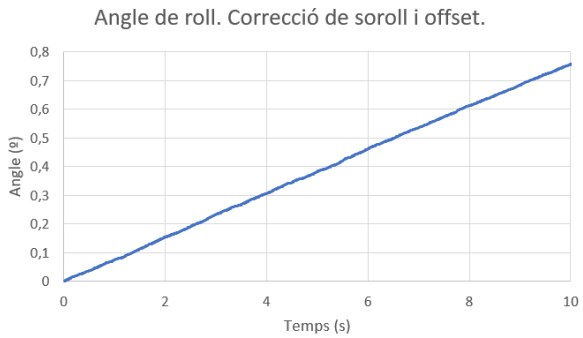

Figura 22. Angle de roll amb lectures de velocitat corregides.

Veiem que la deriva causada per l'error de lectura de les velocitats és menor al de les lectures brutes, però que a llarg termini és capaç de ser prou significatiu. És per això que necessitem algun mecanisme per poder eliminar aquesta deriva. Per fer-ho s'apliquen tècniques de fusió sensorial, és a dir, es combinen dades de dos sensors independents per obtenir informació més fiable.

El segon sensor serà l'acceleròmetre integrat en el mòdul GY-521, el qual té menys presència de soroll i offset i per tant en condicions estàtiques les mesures són molt més precises, com es pot veure a continuació amb l'offset i soroll prèviament corregit.

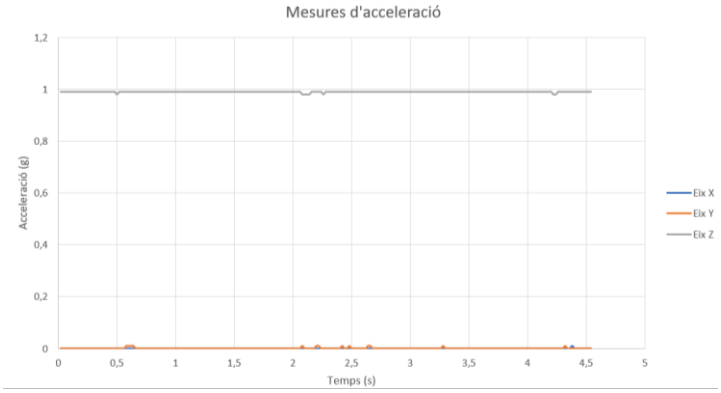

Figura 23. Lectures d'acceleració lineal en els tres eixos.

A partir de les mesures d'acceleració i aprofitant el vector de la gravetat, es pot mesurar l'angle de yaw a partir d'operacions trigonomètriques, segons la fórmula següent:

$$\sigma_z = \tan^{-1}\left(\frac{A_x}{\sqrt{A_y^2+A_z^2}}\right) \qquad \text{(Eq.20)}$$

I de manera similar, l'angle de pitch ve definit per l'equació següent:

$$\sigma_y = \tan^{-1}\left(\frac{A_y}{\sqrt{A_x^2+A_z^2}}\right) \quad \text{(Eq.21)}$$

No és possible determinar l'angle de roll ja que l'eix de gir és paral·lel al vector de la gravetat i, per tant, una rotació sobre l'eix x del dispositiu no provoca cap canvi en les lectures d'acceleració lineal. Per tant a l'hora de fer la fusió de senyals, l'eix x del gimbal no disposarà de mesures auxiliars.

Ara, per combinar els dos senyals, s'implementa un filtre complementari de la següent estructura.

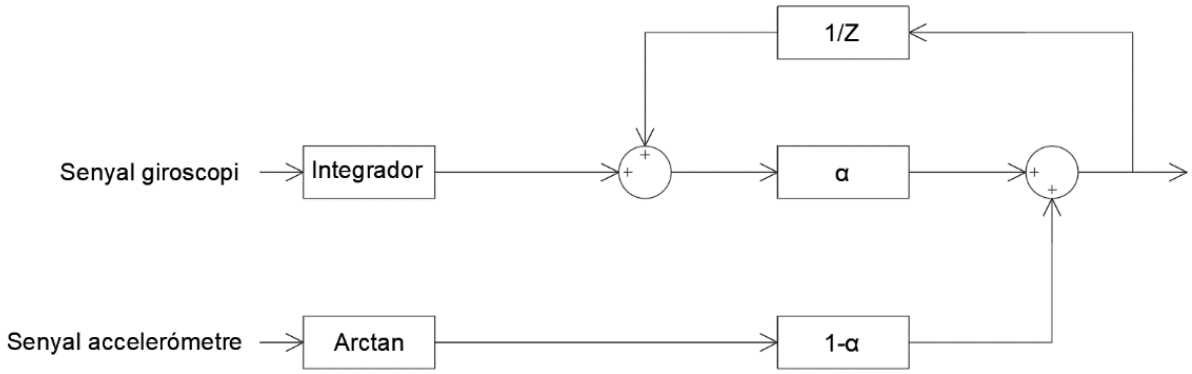

Figura 24.Estructura del filtre complementar dels angles de yaw i pitch.

Aquesta estructura primerament transforma el senyal del giroscopi i acceleròmetre en angles de yaw i pitch, en el cas de l'acceleròmetre ho fa per via de les equacions 20 i 21 descrites anteriorment, i el giroscopi multiplicant el senyal pel temps de mostreig, integrant el senyal. Posteriorment a aquest senyal integrat del giroscopi se li ha de sumar l'angle de sortida calculat amb un període de retard.

A partir d'aquí entra en joc el filtre complementari, aquest es basa en combinar una fracció, α, d'un senyal amb la part complementària, 1 - α, del segon senyal. El giroscopi dona lectures fiables en condicions dinàmiques, mentre que l'acceleròmetre les dona en condicions estàtiques. Per tant, si s'estableix valors d'alfa elevats, es permetrà que els canvis ràpids detectats pel giroscopi siguin processats i tinguin més pes davant de l'acceleròmetre en moments de moviment i en moments en el qual el gimbal estigui en repòs l'acceleròmetre contribuirà marginalment a compensar la deriva provocada pel giroscopi.

La imatge següent mostra els angles de yaw, pitch i roll en estàtic al llarg del temps aplicant un filtre complementari amb alfa igual a 0,9 als angles de yaw i pitch.

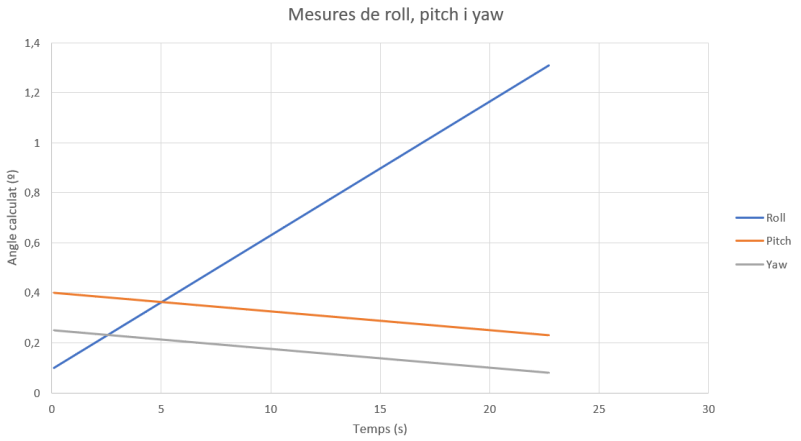

Figura 19.Angles de yaw i pitch filtrats. Angle de roll com a referència prèvia.

S'observa clarament que els eixos que disposen del filtre complementari els efectes provocats per la deriva del giroscopi són molt més atenuats a comparació amb l'eix que no queda filtrat.

6. PROGRAMACIÓ

L'Arduino Pro Mini es programa en llenguatge C. Tot i que Arduino disposa d'un IDE oficial, es fa servir la plataforma Visual Studio Code, un editor de codi molt potent i versàtil, amb l'extensió PlatformIO, una eina multiplataforma que permet programar multitud de sistemes embedded, plaques Arduino incloses. S'ha de remarcar que, com ja s'ha mencionat anteriorment, l'Arduino ProMini no disposa de connexions USB de cap mena, per tant per poder carregar el programa des d'un PC caldrà fer servir un adaptador com per exemple un programador CP2102, també és necessari aquest accessori si es vol fer servir el monitor sèrie.

6.1. Subfuncions

Per tal de no sobrecarregar visualment el codi, es divideix el programa en diferents blocs amb una finalitat específica, resultant en un total de sis funcions auxiliars.

La primera subfunció és PolsadorEstats(), aquesta llegeix les pulsacions del switch del joystick, cada pulsació engloba un flanc de baixada i un de pujada. Partint de què la variable Estat té un valor inicial zero, cada flanc de baixada o pujada incrementa el valor d'aquesta variable en u. Després de la primera pulsació, que engloba el primer flanc de baixada i de pujada, l'estat val dos i la variable ModeManual pren el valor lògic d'u. Una segona pulsació assigna el valor de l'estat a quatre, la variable ModeManual no canvia. Finalment, després de la tercera pulsació, l'estat val sis i la variable ModeManual torna a valor zero i seguidament es reinicien els estats de la funció. El valor de les variables ModeManual i l'estat són llegides posteriorment per les altres subfuncions. La figura següent detalla el flux d'aquest procediment.

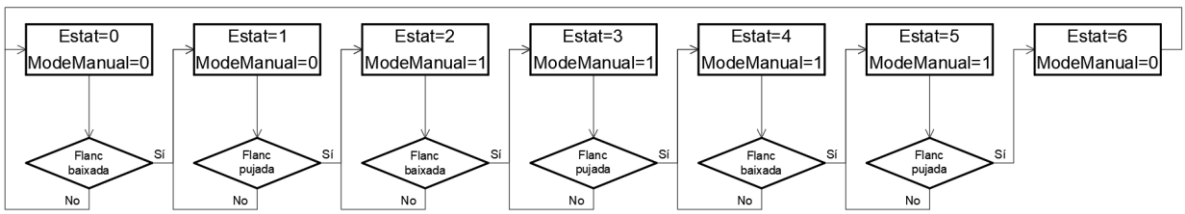

Figura 20. Seqüencia d'execució de la funció PolsadorEstats().

La segona funció auxiliar és AjusatamentManual(), també relacionada amb el joystick, aquesta llegeix el desplaçament del cursor del joystick. Quan la variable Estat val dos i ModeManual és actiu, és a dir, després de la primera pulsació, si el cursor es desplaça fins

al final de la carrera positiva en l'eix horitzontal, la variable RefX augmentarà el seu valor en un, sempre i quan no tingui un valor superior a vuitanta-cinc, si es desplaça fins al final de carrera negatiu, al seu valor se li restarà u, amb un límit inferior de menys vuitanta-cinc. Es segueix la mateixa metodologia per l'eix Y amb la variable RefY. Després d'una segona pulsació, l'estat val quatre i ModeManual continua amb valor u, en aquest cas, el desplaçament horitzontal canvia el valor de la variable RefZ segons el procediment descrit per les altres dues variables, en comptes de RefX. Els valors de RefX, RefY i RefZ es faran servir posteriorment per dirigir l'orientació del gimbal manualment.

La tercera és LecturaAccLineal(), aquesta serveix per obtenir les dades d'acceleració lineal. Per fer-ho, s'inicia la comunicació I2C i cridem al MPU6050 a la seva adreça predeterminada, 0x68. Quan es rep el bit de confirmació de què el sensor està disponible, s'apunta al registre 0x3B, el primer dels sis bytes que contenen les dades d'acceleració lineal, dos bytes per cada eix. Llavors es demana al MPU6050 que enviï un total de sis bytes cap a l'Arduino. Pel propi funcionament del bus I2C, aquests bytes es guarden temporalment en un buffer i són enviades d'un en un seqüencialment al microcontrolador. A mesura que el microcontrolador rep els bytes, els emmagatzema a la seva corresponent variable. Com s'ha dit abans, cada dada d'acceleració consisteix en dos bytes d'informació, cada variable ha d'englobar aquests dos bytes, per fer-ho, quan es rep el primer byte es grava a la variable corresponent, llavors es fa una operació de shift per desplaçar cada bit vuit posicions cap a l'esquerra i llavors se suma el segon byte. Això és possible ja que les variables on s'emmagatzemen els bytes són de tipus int i per tant tenen una mida de dos bytes. Un cop hi ha els dos bytes escrits a la variable, es divideix entre la sensibilitat de l'acceleròmetre, 16384 LSB/g per defecte, perque la informació arriba del MPU6050 en format binari. Aquest procediment es repeteix tres vegades, una per cada eix d'acceleració.

La següent és LecturaVelAngular(), fa el mateix que l'anterior però per velocitats angulars, el procediment és el mateix, inicia el bus I2C, s'adreça al MPU6050 i, en aquest cas, apunta al registre 0x43, demana sis bytes d'informació i codifica dos bytes consecutius en una variable tipus int. Llavors divideix entre la sensibilitat, que per defecte és 131 LSB/rad/s.

La cinquena funció, PosicionamentMotor(), s'encarrega de llegir l'orientació del dispositiu i enviar als servomotors els polsos PWM necessaris per tal de compensar el canvi d'orientació. Per fer-ho, partint de la posició central del servo, se li resta el valor de l'angle girat en un eix específic. El resultat d'aquesta operació és la posició que ha d'adoptar el servo de l'eix en qüestió. Aquest valor llavors és escalat entre els límits del rang de

funcionament dels servos en microsegons, és a dir, cinc-cents o dos mil cinc-cents, escalat el valor entre els límits, la funció writeMicroseconds() envia un senyal PWM amb un temps actiu igual al trobat anteriorment. Aquest procediment es repeteix pels tres servomotors.

La sisena i última funció, TransformacióAcceleració_Orientació(), conté les operacions matemàtiques necessàries per obtenir l'orientació dels tres eixos del dispositiu a partir de les dades d'acceleració lineal i velocitat angular, seguint les equacions 19, 20 i 21 descrites a l'apartat 4. És remarcable que és aquí quan s'afegeix la compensació per offset als càlculs. Aquests offsets es calculen per separat en un codi auxiliar independent del principal.

6.2. Codi principal

El codi principal s'ha estructurat en les diferents subfuncions que s'executen dins de les dues principals, setup() i loop(), seguint el diagrama següent.

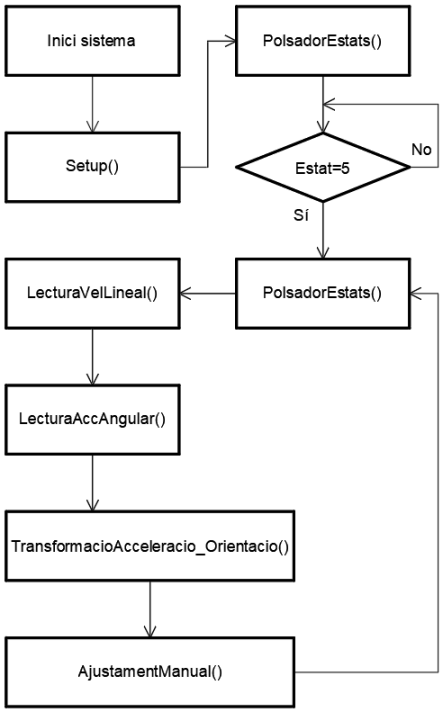

Figura 21. Diagrama de flux del programa principal.

Quan s'inicia el sistema, s'inclouen les llibreries que es faran servir, en aquest cas la que controla la comunicació I2C i els servos. Seguidament, es defineixen les diferents variables i constants que fem servir per emmagatzemar les dades. A part de les variables de tipus bool, int i float que farem servir, també definim tres objectes de classe Servo, corresponents als

tres servomotors que tenim, això permet accedir a les funcions integrades de la llibreria Servo.h.

Declarades les variables i creats els objectes Servo, es defineixen les sis subfuncions auxiliars descrites anteriorment que simplifiquen visualment el flux del programa.

Llavors s'executa la funció setup(), aquí es configura inicialment el pin associat al polsador del joystick com a input, és necessari també habilitar la resistència de pull-up interna de l'Arduino, ja que no s'ha integrat de manera externa al joystick. Seguidament dins el setup s'executa la funció PolsadorEstats() fins que l'estat tingui un valor cinc, és a dir, s'hagin fet tres pulsacions. Aquesta condició actua com a senyal d'engegada pel programa, després de la qual s'inicia la comunicació I2C i es restableix el MPU6050 a configuració de fàbrica, seguidament s'associa cada servomotor a un pin amb capacitat PWM i s'envia un pols de mil cinc-cents microsegons a tots tres servomotors, col·locant-los en posició central de cent trenta-cinc graus.

Seguidament, s'executa el loop, en el qual se segueix executant la funció PolsadorEstats(), es llegeixen les dades d'acceleració lineal i velocitat angular del MPU6050, i es transforma les acceleracions en angles, tenint en compte el temps que tarda a executar un cicle del loop. Finalment, sobre aquest angle girat pel gimbal s'afegeix l'angle introduït manualment pel joysitick i s'envien els polsos PWM corresponents als servomotors.

6.3. Codi de calibratge

Addicionalment, es requereix un codi secundari que es fa servir per tal de trobar el valor d'offset del MPU6050. Per fer-ho es fan dues-centes lectures d'acceleració lineal i velocitat angular, sumant totes les lectures de cada tipus i dividint entre dos-cents, és a dir, la mitjana de les lectures, seguidament es mostra per pantalla els valors d'offset en cada eix del giroscopi i de l'acceleròmetre. Aquests valors són els que s'han d'introduir a la subfunció TransformacióAcceleració_Orientació() del codi principal.

Remarcar que aquest programa s'ha d'executar abans de carregar el programa principal i amb el sensor estàtic.

7. RESUM DEL PRESUPOST

Pel disseny, fabricació i programació del gimbal projectat en aquest document, l'import total ascendeix a mil cent cinquanta-set euros amb noranta-un cèntims d'euro, sense IVA.

8. CONCLUSIONS

S'han assolit els objectius d'aquest projecte, tot i que de manera parcial. El gimbal projectat permet compensar rotacions del dispositiu, funciona per via de bateries recarregables, les quals es recarreguen sense necessitat de treure-les mitjançant un carregador USB tipus C estàndard. Però presenta mancances a l'hora de compensar les oscil·lacions ràpides, a causa de limitacions sobre el control dels servomotors i l'error que introdueix el sensor MPU6050.

Les característiques del gimbal es resumeixen a la següent taula.

Paràmetre	Valor
Tensió de càrrega (V)	5V
Temps de càrrega (minuts)	150
Autonomia (minuts)	33
Rang de moviment (º)	Eix X i Z: ±135; Eix Y: -135~75
Massa màxima acoblable (gr)	300

Taula 7. Característiques del gimbal.

En relació amb possibles millores futures, seria convenient substituir els servomotors per un altra tipologia com ara motors sense escombretes específics per aconseguir un moviment més suau i amb capacitats més àmplies de control de velocitat i posició. La incorporació de sensors més avançats, com ara el MPU9050, possibilita una reducció de l'error causat per deriva i una millor capacitat de determinació de l'orientació.

BO WEI LAI - DNI X4384039D

Firmado digitalmente por BO WEI LAI - DNI X4384039D
Fecha: 2023.09.04 18:41:42 +02'00'

Bo Wei Lai

Graduat en Enginyeria Electrònica Industrial i Automàtica

Celrà, 25 d'agost de 2023

9. RELACIÓ DE DOCUMENTS

El present projecte consta de cinc documents independents, aquests són la memòria, els plànols, el plec de condicions, l'estat d'amidaments i el pressupost.

10. BIBLIOGRAFIA

Analog Devices. Full de característiques LT1070. (https://www.analog.com/media/en/technical-documentation/data-sheets/10701fe.pdf, 15 de juny de 2023)

Analog Devices. Full de característiques LT1510. (https://www.analog.com/media/en/technical-documentation/data-sheets/1510fc.pdf, 9 de juny de 2023)

Analog Devices. LT1070 Design Manual. (https://www.analog.com/media/en/technical-documentation/application-notes/an19fc.pdf, 15 de juny de 2023)

Analog Devices. LT1510 Design Manual. (https://www.analog.com/media/en/technical-documentation/application-notes/an68f.pdf, 9 de juny de 2023)

Analog Devices. Using an Accelerometer for Inclination Sensing. (https://www.analog.com/en/app-notes/an-1057.html, 9 de juny de 2023)

Atmel. Full de característiques de l'AtMega 328P (https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf, 5 de maig de 2023)

Autodesk Instructables. The Making of a DIY Brushless Gimbal With Arduino. (https://www.instructables.com/DIY-Brushless-Gimbal-with-Arduino/, 5 de juliol de 2023)

Douglas, Brian. Drone Control and the Complementary Filter (https://youtu.be/whSw42XddsU, 9 de juny de 2023)

How To Mechatronics. DIY Arduino Gimbal. Self-Stabilizing Platform. (https://howtomechatronics.com/projects/diy-arduino-gimbal-self-stabilizing-platform/?utm_content=cmp-true, 10 d'agost de 2023)

Ivensense. Full de característiques del MPU6050. (https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf, 6 de maig de 2023)

Ivensense. MPU-6050 Register Map and Descriptions. (https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf, 8 d'agost de 2023)

PlatformIO. Arduino Pro Mini Board Documentation (https://docs.platformio.org/en/latest/boards/atmelavr/pro16MHzatmega328.html, 2 d'abril de 2023)

Protobots. Característiques servomotor DS3220. (https://probots.co.in/ds3230-pro-30kg-high-torque-servo-motor-speed-waterproof-metal-gear.html, 3 de maig de 2023)

Sparkfun. Using the Arduino Pro Mini 3.3V. (https://learn.sparkfun.com/tutorials/using-the-arduino-pro-mini-33v#what-it-is-and-isnt, 5 d'abril de 2023)

Trends Topic Journal. How To Get A Reliable Yaw Angle From An MPU6050. (https://www.trendtopicsjournal.com/how-to/get-reliable-yaw-angle-mpu6050/, 10 d'agost de 2023)

UTMEL. Característiques del mòdul GY-521. (https://www.utmel.com/components/mpu6050-module-datasheet-alternative-comparisón?id=414, 6 de maig de 2023)

Wikipedia. Gimbal. (https://en.wikipedia.org/wiki/Gimbal#:~:text=History,-Cardan%20suspension%20in&text=The%20gimbal%20was%20first%20described,holes%20of%20the%20other%20sides., 27 d'abril de 2023)

11. GLOSSARI

ADC: Analogic to Digital Converter

GPIO: General Purpose Input Output

I2C: Inter-Integrated Circuit

IDE: Interactive Development Environment

IMU: Inertial Measurement Unit

LED: Light Emiting Diode

LSB: Least Significant Bit

MEMS: MicroElectroMechanical System

NiCd: Níquel Cadmi

NiMH Níquel Metall Hidròxid

OPAMP: OPerational AMPlifier

PCB: Printed Circuit Board

PWM: Pulse Width Modulation

SMD: Surface Mounted Device

TTL: Transistor to Transistor Logic

UART: Universal Asynchronous Receiver-Transmiter

A. PROGRAMARI

A1. Programa principal

```
#include "I2Cdev.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE

    #include "Wire.h"

#endif

#include <Servo.h>

#define POLSADOR 12

const int MPU = 0x68; //Adreça MPU6050

bool ModeManual;

int RefX=0,RefY=0,RefZ=0,Estat=0;

float AccX, AccY, AccZ; // Variables per guardar el valor d'acceleració lineal dels 3 eixos

float GyroX, GyroY, GyroZ; // Variables per guardar el valor d'acceleració angular dels 3 eixos

float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ; //Variables per guardar el valor de l'angle girat , calculat a partir de acceleracions lineals i angulars

float roll, pitch, yaw; //Valor de angles girats

float TempsTranscorregut, TempsActual, TempsAnterior; //Variables per comptar el temps que dura una acceleració concreta.

Servo ServoX,ServoY,ServoZ;//Creem instàncies de tipus servo

//Subfuncions del programa

void PolsadorEstats() //Funció que permet simular un pulsador amb encalvament. Associat a l'activació del mode manual. També és fa servir per l'inici de sistema.
```

```
{

    //primer estat, a l'espera del flanc de baixada del senyal.

    if(digitalRead(POLSADOR)==1 && Estat==0)

        {Estat=1;

        delay(100);

        }

     //segon estat, a l'espera del flanc de pujada del senyal. A nivell de
simulació, s'ha enclavat el polsador

    else if (digitalRead(POLSADOR)==0 && Estat==1)

        {

            Estat=2;

            ModeManual=1;

            delay(100);

        }

    //tercer estat, a l'espera del flanc de baixada del senyal.

    else if (digitalRead(POLSADOR)==1 && Estat==2)

        {

            Estat=3;

            delay(100);

        }

     //quart estat, a l'espera del flanc de pujada del senyal. A nivell de
simulació, s'ha desenclavat el polsador

    else if (digitalRead(POLSADOR)==0 && Estat==3)
```

```
        {
            delay(100);
            Estat=4;
        }
    else if(digitalRead(POLSADOR)==1 && Estat==4)
        {
        Estat=5;
        delay(100);
        }
     //segon estat, a l'espera del flanc de pujada del senyal. A nivell de simulació, s'ha enclavat el polsador
    else if (digitalRead(POLSADOR)==0 && Estat==5)
        {
            Estat=6;
            delay(100);
        }
    //tercer estat, a l'espera del flanc de baixada del senyal.
    else if (digitalRead(POLSADOR)==1 && Estat==6)
        {
            delay(100);
            Estat=0;
            ModeManual=0;
        }
```

```
}

void AjustamentManual()//Ajustament manual de l'orientació

{

    if (ModeManual==1)

        {

            if (analogRead(A0)>800 && RefX<130 && Estat==3)

                {

                    RefX=RefX+1;

                }

            else if (analogRead(A0)<200 && RefX>-130 && Estat==3)

                {

                    RefX=RefX-1;

                }

            else if (analogRead(A1)>800 && RefY<130 && Estat==3)

                {

                    RefY=RefY+1;

                }

            else if (analogRead(A1)<200 && RefY>-130 && Estat==3)

                {

                    RefY=RefY-1;

                }

            else if (analogRead(A0)>800 && RefZ<130 && Estat==5)
```

```
                {

                    RefZ=RefZ+1;

                }

            else if (analogRead(A0)<200 && RefZ>-130 && Estat==5)

                {

                    RefZ=RefZ-1;

                }

        }

}

void LecturaAccLineal()//Lectura dels valors d'acceleració lineal

{

    Wire.beginTransmission(MPU); //Comença la comunicació I2C

    Wire.write(0x3B); // Apunta al registre 0x3B (ACCEL_XOUT_H)

    Wire.endTransmission(false);

    Wire.requestFrom(MPU, 6, true); // Demana a l'esclau 6 bytes, 2  per cada dada
de acceleració en un sol eix

    //Llegeix cada byte, manipula per tal de transformar la lectura binaria en un
numero decimal

    //Es divideix entre la resolució (16384 pel rang per defecte) per obtenir
acceleracions en m/s^2

    AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // Acceleració lineal en eix
X

    AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // Acceleració lineal en eix
Y

    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // Acceleració lineal en eix
Z
```

```
}

void LecturaVelAngular()//Lectura dels valors d'velocitat angular

{

    Wire.beginTransmission(MPU);

    Wire.write(0x43); // Apunta al registre 0x43

    Wire.endTransmission(false);

     Wire.requestFrom(MPU, 6, true); // Demana al esclau 6 bytes, 2  per cada dada de velocitat en un sol eix

     //Llegeix cada byte, manipula per tal de transformar la lectura binaria en un numero decimal

      //Es divideix entre la resolució (131 pel rang per defecte) per obtenir velocitat en rad/s

    GyroX = (Wire.read() << 8 | Wire.read()) / 131.0;

    GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;

    GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;

}

void TransformacioAcceleracio_Orientacio()//Manipulació matemàtica per obtenir valors d'orientació en funció de les dades mesurades d'acceleració lineal i velocitat angular

{

    // Calcul de l'orientació en funcio de les acceleracions lineals.

    accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - 0.62; // S'afegeix l'offset de l'acceleració lineal en X

    accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) + 1.58; // S'afegeix l'offset de l'acceleració lineal en Y

    // Aplica correccions d'offset a les velocitats angulars dels 3 eixos

    GyroX = GyroX + 1;
```

```
    GyroY = GyroY + 0.985;

    GyroZ = GyroZ + 1.6;

     // càlcul de l'orientació a partir de les velocitats angulars i el temps de
cicle.

    gyroAngleX = gyroAngleX + GyroX * TempsTranscorregut;

    gyroAngleY = gyroAngleY + GyroY * TempsTranscorregut;

    gyroAngleZ = gyroAngleZ + GyroZ * TempsTranscorregut;

     //S'assigna el valor final de yaw pitch i roll, si aquests no superen els
límits mecanics dels motors (90 graus a cada banda)

     if ((0.95 * gyroAngleZ+ 0.05 * accAngleX + RefZ)>=-130 &&(0.95 * gyroAngleZ+
0.05 * accAngleX + RefZ)<=130)

    {

        yaw =  0.95 * gyroAngleZ+ 0.05 * accAngleX + RefZ;

    }

    else if ((0.95 * gyroAngleZ+ 0.05 * accAngleX + RefZ)<-130)

    {

        yaw=-130;

    }

    else if ((0.95 * gyroAngleZ+ 0.05 * accAngleX + RefZ)>130)

    {

        yaw=130;

    }

    if ((gyroAngleX +RefX)>=-130 && (gyroAngleX +RefX)<=130)

    {
```

```
        roll = gyroAngleX +RefX; //S'aplica un filtre complementari

    }

    else if ((gyroAngleX +RefX)<-130)

    {

        roll=-130;

    }

    else if ((gyroAngleX +RefX)>130)

    {

        roll=130;

    }

     if ((0.95 * gyroAngleY + 0.05 * accAngleY+RefY)>=-130 && (0.95 * gyroAngleY +
0.05 * accAngleY+RefY)<=130)

    {

          pitch = 0.95 * gyroAngleY + 0.05 * accAngleY+RefY; //S'aplica un filtre
complementari

    }

    else if ((0.95 * gyroAngleY + 0.05 * accAngleY+RefY)<-130)

    {

        pitch=-130;

    }

    else if ((0.95 * gyroAngleY + 0.05 * accAngleY+RefY)>130)

    {

        pitch=130;
```

```
    }

}

void PosicionamentMotor()//Enviament de polsos PWM als motors

{

     //Partint de la posicio neutral de cada motor, es resta el valor de l'angle girat en cada eix i s'escala el resultat per emetre un puls PWM al motor.

    ServoX.writeMicroseconds(map(135-roll,0,270,500,2500));

    ServoY.writeMicroseconds(map(135+pitch,0,270,500,2500));

    ServoZ.writeMicroseconds(map(135+yaw,0,270,500,2500));

    delay(20);

}

void setup()

{

     pinMode(POLSADOR,INPUT_PULLUP);      //Configurem el pin 12 del switch del joystick com a INPUT, habilitant la resistencia de PULLUP integrada.

     while (true) //Es mante dins el while fins que no es faci un enclavament + desenclavament del polsador del joystick.

        {

            if (Estat==6 && digitalRead(12)==0)

            {

                break;

            }

            PolsadorEstats();

        }
```

```
    Wire.begin();                       // Iniciem el bus I2C

    Wire.beginTransmission(MPU);        // Crida al MPU6050 a la seva adreça.

     Wire.write(0x6B);                      // Apunta al registre 6B del MPU6050,
referent a configuració de mode de funcionament

     Wire.write(0x00);                   //Reseteja el dispositiu a configuració de
fàbrica.

    Wire.endTransmission(true);         //Acaba la transmissió de dades pel bus

    Wire.beginTransmission(MPU);        // Crida al MPU6050 a la seva adreça.

     Wire.write(0x1A);                      // Apunta al registre 1A del MPU6050,
referent a configuració del filtre pas baix de l'acceleròmetre i giroscopi.

    Wire.write(0x06);                   //Estableix un ample de banda de 20 Hz

    Wire.endTransmission(true);         //Acaba la transmissió de dades pel bus

    ModeManual=0;

    //Associem cada servomotor a un pin.

    ServoX.attach(9);

    ServoY.attach(10);

    ServoZ.attach(11);

    delay(10);

    //Posicioar els 3 motors en posició central,90 graus.

    ServoX.writeMicroseconds(1500);

    delay(250);

    ServoY.writeMicroseconds(1500);

    delay(250);

    ServoZ.writeMicroseconds(1500);
```

```
    delay(10);

}



void loop()

{

     PolsadorEstats(); //Exceutem la funció de polsador per determinar si estem en mode manual o no.

    //Lectura dels valors de acceleració angular i lineal

    LecturaAccLineal();

    LecturaVelAngular();

    //Calcul del temps que tarda en executar un cicle de mesura+ajustat de posicio.

    TempsAnterior = TempsActual;

    TempsActual = millis();

       TempsTranscorregut= (TempsActual - TempsAnterior) / 1000.; // Es divideix entre 1000 per obtenir segons

    TransformacioAcceleracio_Orientacio();

    AjustamentManual();

    PosicionamentMotor();

}
```

A2.Programa de calibratge

```
#include "I2Cdev.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
```

```
    #include "Wire.h"

#endif


const int MPU = 0x68; // MPU6050 I2C address

float AccX, AccY, AccZ;

float GyroX, GyroY, GyroZ;

float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;

int c = 0;

void setup()

{

    Serial.begin(19200);

    Wire.begin();                      // Iniciem el bus I2C

    Wire.beginTransmission(MPU);       // Crida al MPU6050 a la seva adreça.

     Wire.write(0x6B);                        // Apunta al registre 6B del MPU6050,
referent a configuració de mode de funcionament

     Wire.write(0x00);                    //Reseteja el dispositiu a configuració de
fàbrica.

    Wire.endTransmission(true);        //Acaba la transmissió de dades pel bus

    delay(20);

    //Llegeix 200 vegades el valor de les acceleracions lineals

     while (c < 200) {

        Wire.beginTransmission(MPU);

        Wire.write(0x3B);

        Wire.endTransmission(false);
```

```
        Wire.requestFrom(MPU, 6, true);

        AccX = (Wire.read() << 8 | Wire.read()) / 16384.0 ;

        AccY = (Wire.read() << 8 | Wire.read()) / 16384.0 ;

        AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0 ;

    //Suma els valors obtinguts cada vegada

         AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ),
2))) * 180 / PI));

           AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) +
pow((AccZ), 2))) * 180 / PI));

        c++;

  }

  //Divideix el valor total entre 200, per obtenir el valor mitjà.

  AccErrorX = AccErrorX / 200;

  AccErrorY = AccErrorY / 200;

  c = 0;

  //Llegeix 200 vegades el valor de les acceleracions angulars

  while (c < 200) {

    Wire.beginTransmission(MPU);

    Wire.write(0x43);

    Wire.endTransmission(false);

    Wire.requestFrom(MPU, 6, true);

    GyroX = Wire.read() << 8 | Wire.read();

    GyroY = Wire.read() << 8 | Wire.read();
```

```
    GyroZ = Wire.read() << 8 | Wire.read();
//Suma els valors obtinguts cada vegada
    GyroErrorX = GyroErrorX + (GyroX / 131.0);
    GyroErrorY = GyroErrorY + (GyroY / 131.0);
    GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
    c++;
  }
 //Divideix el valor total entre 200, per obtenir el valor mitjà.
  GyroErrorX = GyroErrorX / 200;
  GyroErrorY = GyroErrorY / 200;
  GyroErrorZ = GyroErrorZ / 200;
  //Mostra els errors mitjans per pantalla
  Serial.print("AccErrorX: ");
  Serial.println(AccErrorX);
  Serial.print("AccErrorY: ");
  Serial.println(AccErrorY);
  Serial.print("GyroErrorX: ");
  Serial.println(GyroErrorX);
  Serial.print("GyroErrorY: ");
  Serial.println(GyroErrorY);
  Serial.print("GyroErrorZ: ");
  Serial.println(GyroErrorZ);
```

```
}

void loop()

{

}
```

B. INSTRUCCIONS D'ÚS

Especificacions

Paràmetre	Valor
Tensió de càrrega (V)	5V
Temps de càrrega (minuts)	150
Autonomia (minuts)	33
Rang de moviment (º)	Eix X i Z: ±135; Eix Y: -135~75
Massa màxima acoblable (gr)	300

Taula 1. Característiques del gimbal.

Instruccions d'ús

Endollar un carregador USB tipus C al port de càrrega, carregar fins que la llum blava s'engegui.

Agafar el gimbal en posició vertical i enfocant a l'objecte d'interes. Engegar el gimbal.

Clicar tres vegades el joystick, esperar a que els servomotors es posicionin a la posició inicial.

Si es vol dirigir manualment el gimbal, clicar un cop el polsador per controlar l'angle de roll i pitch amb el joystick, un altre cop per ajustar l'angle de yaw i un altre per deshabilitar el joystick.

Recomanacions

Es recomana treballar en un entorn a 25ºC, variacions de les condicions ambientals poden afectar a la precisió de les mesures d'orientació.

No exposar a entorns humits ni amb presencia de agents químics.

Al final de la vida útil del dispositiu, portar-lo a un punt de reciclatge adequat.