



Universitat de les
Illes Balears



Treball Final de Grau

GRAU EN ENGINYERIA ELECTRÒNICA INDUSTRIAL I AUTOMÀTICA

DISSENY I CONSTRUCCIÓ D'UN ESCÀNER 3D A PARTIR D'UN LIDAR BIDIMENSIONAL

JORDI SOLER BUSQUETS

DIRECTOR: GABRIEL OLIVER CODINA
CODIRECTOR: MIQUEL MASSOT CAMPOS

Escola Politècnica Superior
Universitat de les Illes Balears

Palma, 17 de juny de 2014

ÍNDIX

Índex	i
Índex de figures	iii
Índex de taules	v
Resum	vii
1 Introducció i motivació	1
1.1 Introducció	1
1.2 Objectius i motivació	2
2 Requeriments físics i disseny de l'escàner	5
2.1 Aspectes físics	5
2.1.1 Introducció	5
2.1.2 Camp de visió	5
2.1.3 Mecanismes de transmissió de moviment	5
2.1.4 Suport del LIDAR	6
2.1.5 Control de la posició	6
2.1.6 Cablejat i connexions	6
2.1.7 Encapsulat i dispositius addicionals	6
2.2 Tecnologia utilitzada	6
2.2.1 Placa <i>Arduino Duemilanove</i>	6
2.2.2 Motor pas a pas	8
2.2.3 Impressió 3D	8
2.2.4 Sensor reflectiu	9
2.2.5 Elements mecànics addicionals	10
2.3 Disseny elegit	12
2.3.1 Elements mecànics	12
2.3.2 Elements elèctrics i electrònics	15
3 Desenvolupament del programari	19
3.1 Introducció i entorn de desenvolupament	19
3.2 Codi extern	20
3.3 Solució elegida i característiques	20
3.3.1 Funcionament general i justificació	20
3.3.2 Paquet <i>lidar_scan</i>	23

3.3.3	Programa <i>hokuyo_arduino</i>	25
3.3.4	Arxiu <i>lidarscan.launch</i>	26
4	Resultats	29
4.1	Escanejos senzills	29
4.2	Escaneig compost	31
4.3	Escaneig exterior	37
5	Conclusions i treballs futurs	39
A	Codi font	41
A.1	Node de ROS <i>lidar_scan_node</i>	41
A.2	Node <i>pc_snapshotter</i>	43
A.3	Programa <i>hokuyo_arduino</i>	45
A.4	Arxiu <i>lidarscan.launch</i>	48
B	Codi per a l'obtenció de l'escaneig compost	51
B.1	Node <i>compondre</i>	51
B.2	Arxiu <i>reconstruccio.launch</i>	54
C	Fulls de característiques	55
C.1	Hokuyo UTM-30LX-EW	55
C.2	Sensor reflectiu OPB704	63
C.3	Motor pas a pas Nema 17	66
C.4	Circuit integrat DRV8825	68
	Bibliografia	77

ÍNDIX DE FIGURES

1.1	Velodyne HDL-64E.	2
1.2	Hokuyo UTM-30LX-EW. Aspecte extern.	2
1.3	Rang de visió del Hokuyo UTM-30LX-EW.	3
1.4	Esquema de funcionament de l'escàner 3D sobre un robot.	3
2.1	Placa de microcontrolador <i>Arduino Duemilanove</i>	7
2.2	Esquema de les comunicacions de l'escàner 3D.	7
2.3	Pinout de la placa DRV8825 controladora del motor pas a pas.	8
2.4	Principi de funcionament del sensor OPB704.	9
2.5	Circuit acondicionador per al sensor reflectiu.	10
2.6	Dimensions del <i>slip ring</i>	10
2.7	Ensamblatge d'elements mecànics de l'escàner.	12
2.8	Acoblament entre les parts mòbil i fixa.	12
2.9	Plataforma de contacte per al Hokuyo UTM-30LX-EW.	13
2.10	Eix mòbil. Ensamblatge amb el suport	14
2.11	Eix mòbil. Engranatge de l'eix de gir del suport	14
2.12	Engranatge del motor pas a pas.	14
2.13	Placa de PVC.	15
2.14	Layout de la placa de circuit imprès.	16
2.15	Distribució dels elements.	17
3.1	Esquema principal de l'estructura del programari.	20
3.2	Esquema de funcionament de la classe <i>lidarScan</i>	24
3.3	Esquema de funcionament del node <i>pc_snapshotter</i>	25
4.1	Escaneig del laboratori 126.	30
4.2	Diferències entre escanejos.	30
4.3	Niguls de punts basats en diferents rebots del làser.	31
4.4	Punts d'escaneig.	32
4.5	Presa d'escaneigs.	32
4.6	Escaneig compost. Punt 1.	33
4.7	Escaneig compost. Punt 2.	34
4.8	Escaneig compost. Punt 4.	35
4.9	Escaneig compost. Punt 5.	35
4.10	Escaneig compost. Planta del conjunt.	36
4.11	Preparació de l'escaneig exterior.	37
4.12	Escaneig exterior. Vista 1.	38

4.13 Escaneig exterior. Vista 2.	38
--	----

ÍNDIX DE TAULES

2.1	Resolució angular en funció de les entrades M0, M1 i M2.	9
2.2	Característiques del rodament 619-0402.	11
2.3	Entrades i sortides de la placa <i>Arduino</i>	16
3.1	Tòpics usats i tipus associats.	21
3.2	Valors de les sortides de la placa <i>Arduino</i>	26
3.3	Nodes i paràmetres de l'arxiu <i>lidarscan.launch</i>	27
4.1	Error en les relacions entre sistemes de coordenades.	34

RESUM

En aquest treball es desenvolupa un escàner làser 3D a partir del sensor Hokuyo UTM-30LX-EW, un sensor làser dissenyat per escanejar en un pla. L'escaneig 3D s'aconsegueix fent rotar el sensor làser de manera que els seus successius escanejos 2D corresponguin a un pla orientat lleugerament diferent a l'anterior sobre un eix de rotació fins a completar una volta, obtenint com a resultat un núvol de punts de tot l'entorn. L'objectiu de l'escàner és, dins l'àmbit de la robòtica mòbil, proporcionar informació útil sobre l'espai que envolta el robot que l'utilitzi. En el disseny s'han tingut en compte tant aspectes físics com de programari de manera conjunta aconseguint una alta integració de tots els elements que componen la solució al problema.

S'ha dissenyat un suport giratori per al sensor làser permetent a aquest fer un escanament de l'entorn a una velocitat angular controlada a partir d'un motor pas a pas. Aquest disseny s'ha dut a terme mitjançant eines de disseny per computador i s'ha implementat mitjançant impressió 3D. El control dels elements físics de dins l'escàner s'ha fet amb una placa de microcontrolador i circuits addicionals. L'escàner envia les seves dades a un ordinador extern on aquestes són interpretades i processades per part del programari controlador desenvolupat en aquest treball. El programari s'ha desenvolupat dins l'entorn de treball *ROS (Robot Operating System)* que proporciona eines adequades per a processar dades de sensors làser i està orientat a aplicacions de robòtica.

El resultat és un escàner capaç d'escanejar entorns estàtics. Amb ell es poden reconstruir emplaçaments amb precisió suficient per a aplicacions de reconeixement d'obstacles o navegació. Tot i això, el làser vibra durant l'escaneig provocant errors i no s'han explotat diferents modes d'escaneig. Les seves limitacions pel que fa a precisió i funcionalitats fan recomanables treballs futurs abans del seu ús en aplicacions finalistes.

INTRODUCCIÓ I MOTIVACIÓ

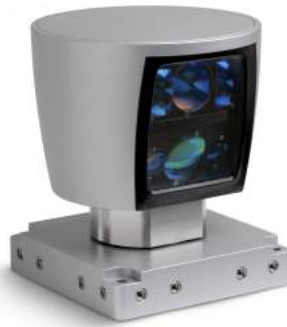
1.1 Introducció

L'ús de dispositius làser dins l'àmbit de la robòtica ha mostrat tenir moltes aplicacions en els darrers anys. La necessitat de conèixer la distància entre dos punts amb molta precisió en àmbits tan diversos com l'agricultura, la geologia, l'automoció o la robòtica, entre d'altres, ha portat al desenvolupament de la tecnologia LIDAR (*Light Detection and Ranging* o *Laser Imaging, Detection and Ranging*). El principi de funcionament d'aquesta tecnologia és il·luminar un objectiu mitjanant un raig làser i determinar la distància a partir de la llum reflectida. Avui en dia, hi ha molts de sensors diferents basats en tecnologia LIDAR en el mercat.

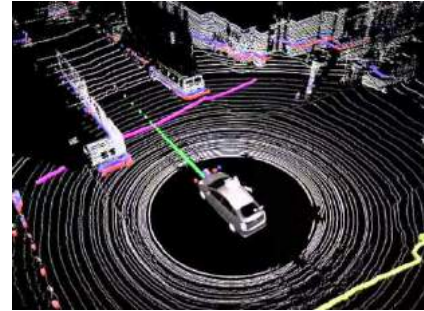
Molts d'aquests sensors tenen per objectiu l'escaneig tridimensional de peces o espais. Utilitzant les lectures de diferents feixos làser s'obté informació sobre un volum. Un exemple n'és el model *FARO Focus 3D* que mitjançant el gir d'un mirall, que determina la direcció de sortida d'un raig làser, es mostregen les distàncies de tot un volum. Aquest producte pot arribar a enregistrar 976.000 punts per segon i està enfocat a aplicacions relacionades amb arquitectura i construcció.

Altres escàners tridimensionals basats en tecnologia LIDAR es fonamenten en fer girar un sensor làser bidimensional per capturar lectures en diferents angles, component els resultats en un escaneig 3D. Aquest és el cas del model *HDL-64E* (figura 1.1a) de *Velodyne*, enfocat en aplicacions d'automoció autònoma, que disposa de 64 parells d'emissors i receptors làsers integrats en una estructura giratòria. En girar es fa un escombrat de l'entorn d'alta definició (figura 1.1b).

En comptes de tenir un sistema d'escaneig 3D integrat, altres escàners 3D, basats també en una plataforma giratòria, fan servir sensors LIDAR bidimensionals comercials. Aquest és el cas del robot PR2 de *Willow Garage* que consta de dos LIDAR planaris Hokuyou UTM-30LX per a obtenir informació sobre el seu entorn. També és una pràctica freqüent en treballs de recerca [1, 2, 3, 5].



(a) Vista exterior.



(b) Escaneig en entorn urbà.

Figura 1.1: Velodyne HDL-64E.

1.2 Objectius i motivació

L'ús d'un LIDAR bidimensional comercial per a implementar un escàner 3D és una alternativa de menor cost, en respecte als escàners 3D integrats, que permet adaptar la configuració de l'escàner a l'aplicació i aconseguir una major integració tant en el maquinari com en el programari. Així, a més del menor cost econòmic, fan que, tot i les limitacions pròpies del LIDAR bidimensional, el seu ús en l'escaneig 3D sigui freqüent en el camp de la robòtica mòbil.

En aquest Treball Final de Grau s'ha dissenyat i construït un escàner 3D a partir d'un LIDAR bidimensional. En concret, s'utilitza un sensor Hokuyo UTM-30LX-EW (figura 1.2). El seu full de característiques es troba a l'annex C.1. Es tracta d'un sensor planari que es basa en el principi de funcionament de mesura de distàncies per temps de vol (ToF de l'anglès *Time of Flight*). Mesurant el temps que tarda un feix de llum des que és emès per l'emissor làser fins que és rebut pel receptor i coneixent la constant de la velocitat de la llum, es mesura l'espai recorregut i, per extensió, la distància a què es troba la superfície que ha reflectit el feix.



Figura 1.2: Hokuyo UTM-30LX-EW. Aspecte extern.

El Hokuyo UTM-30LX-EW consta d'un únic parell emissor-receptor làser de manera que les lectures en el seu pla de visió es fan de manera seqüencial. Es tarden 25ms en fer un escaneig i aquests tenen una resolució angular de 0.25° . El seu abast de fins a 30m el fa adequat per a l'aplicació d'aquest treball. Disposa d'un angle mort de 90° com es mostra a la figura 1.3. Una funcionalitat important del Hokuyo UTM-30LX-EW és l'anomenada *multiecho* que permet obtenir informació de diferents rebots d'un mateix feix làser així com de la intensitat d'aquests rebots.

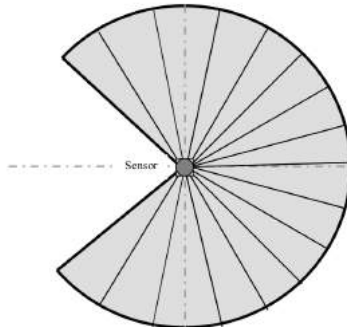


Figura 1.3: Rang de visió del Hokuyo UTM-30LX-EW.

L'escàner 3D construït a partir del LIDAR ha de poder ser emprat en robòtica mòbil, situat a sobre d'un robot per a proporcionar-li les dades dels escaneigs. S'ha de compondre d'una part fixa on s'uneixi amb el robot i una part mòbil, on es situï el LIDAR. A més dels dispositius físics es proporciona tot el programari necessari per tal de controlar l'escàner.

L'eix corresponent a l'angle zero dels feixos del LIDAR ha d'estar orientat verticalment i ha de ser l'eix de gir d'aquest, com mostra la figura 1.4. Així, el resultat final ha de ser un dispositiu capaç d'enregistrar i comunicar un nígul de punts corresponent a l'escaneig 3D de l'espai pròxim al robot.

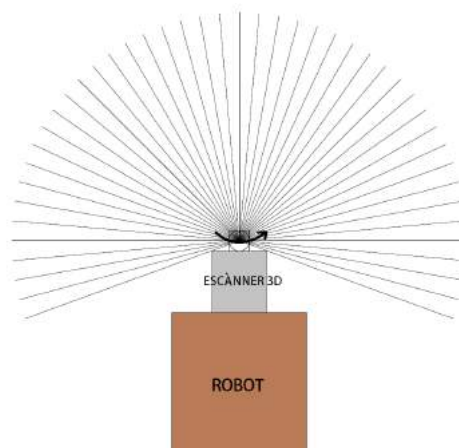


Figura 1.4: Esquema de funcionament de l'escàner 3D sobre un robot.

REQUERIMENTS FÍSICS I DISSENY DE L'ESCÀNER

2.1 Aspectes físics

2.1.1 Introducció

Per a dissenyar un model físic capaç d'habilitar el LIDAR per a satisfer la funcionalitat d'escaneig 3D, és necessari un estudi dels aspectes físics que s'han de complir per tal d'aconseguir-ho. En concret, s'ha de dissenyar un sistema giratori que permeti el gir de l'estructura de suport del LIDAR de manera fluida i controlada. Són també característiques importants la capacitat de conèixer l'orientació del sensor làser i la definició angular dels escaneigs 3D, directament relacionada amb la velocitat de rotació del LIDAR. A més, es vol que el gir es pugui portar a terme de manera indefinida.

Les característiques que es descriuen en els següents apartats resumeixen els principals requeriments a tenir en compte a l'hora de dissenyar el model d'escàner.

2.1.2 Camp de visió

Tenint com a objectiu el reconeixement de l'entorn dins l'àmbit de la robòtica mòbil el camp de visió ha de ser el major possible. Tanmateix, l'escàner estarà situat sobre un robot i la informació que pugui proporcionar en referència a la part del robot, és irrellevant.

2.1.3 Mecanismes de transmissió de moviment

Per a transmetre el moviment del motor pas a pas al LIDAR és necessari algun tipus de transmissió que permeti una distribució compacta dels elements, proporcionant una estructura robusta.

Quant a la relació de transmissió, és necessari que dins el rang de freqüències de funcionament fluid del motor pas a pas, el LIDAR giri a una velocitat adequada. Do-

nada l'aplicació de l'escàner 3D, el seu rang de velocitats d'escaneig hauria d'anar de 5rpm a 50rpm, aproximadament, depenent de si es necessita un escaneig ràpid o d'alta resolució.

2.1.4 Suport del LIDAR

Per a assegurar la subjecció del sensor làser amb la resta de components de l'escàner, és necessari un suport que giri solidàriament amb ell. Ha de permetre suportar el seu pes (370g aproximadament) i evitar oscil·lacions. La seva implementació està estretament lligada a la dels mecanismes de transmissió de moviment.

2.1.5 Control de la posició

Per a posicionar els diferents escaneigs del LIDAR a l'espai és imprescindible conèixer l'orientació d'aquest. El fet d'utilitzar un motor pas a pas facilita aquesta tasca, ja que a causa del seu control es pot deduir l'increment de la posició angular. Tot i així, és necessari poder distingir a través d'algun sensor un angle de referència conegut per tal de poder tractar amb posicions angulars absolutes.

2.1.6 Cablejat i connexions

Tant el sensor làser com els dispositius electrònics addicionals necessiten transmetre informació a dispositius externs com s'explicarà en més detall en el capítol referent al programari. A més, aquests circuits s'han d'alimentar externament. Per altra banda, les necessitats de comunicació entre dispositius electrònics dins el mateix escàner fan que s'hagin de portar a terme connexions internes. Tant les comunicacions el·ectròniques amb l'exterior com les connexions entre els diferents dispositius dins l'escàner s'han d'implementar sense perjudici per a les seves funcionalitats. En concret s'ha de posar especial atenció a les comunicacions elèctriques entre la part mòbil i la part fixa del suport, ja que el gir del rotor no ha d'afectar tals comunicacions.

2.1.7 Encapsulat i dispositius addicionals

Es necessiten alguns dispositius electrònics addicionals per a complir amb tots els requeriments físics ja mencionats. Com es descriuran en la següent secció, el control a baix nivell del sistema es porta a terme per una placa de microcontrolador i circuits amb funcionalitats més específiques. En concret, es fa ús d'una placa controladora per al motor pas a pas i d'un sensor reflectiu per al control de posició. Tots aquests elements s'han de situar dins l'escàner de manera que la distribució final sigui compacta i no impedeixi el bon funcionament dels mateixos. Per tal d'aconseguir que l'escàner sigui portable i de fàcil ús ha d'estar degudament encapsulat.

2.2 Tecnologia utilitzada

2.2.1 Placa Arduino Duemilanove

Dins el mateix escàner, s'han de gestionar els processos físics involucrats en la rutina del dispositiu. Aquests són processos amb necessitats de temps real, que han de ser

tractats de manera periòdica, amb requisits temporals estrictes i amb poca quantitat d'informació a transmetre entre elements del maquinari. Per a donar solució a aquests requisits s'utilitza un sistema basat en microcontrolador.

Concretament, les tasques de control dels elements físics i controladors electrònics seran portades a terme per la placa *Arduino Duemilanove* (figura 2.1). Es tracta d'una placa basada en un microcontrolador de la família ATmega. La versió que s'utilitza per aquest treball és la basada en ATmega168. Aquesta placa permet augmentar la funcionalitat i la facilitat d'ús i programació del microcontrolador. Entre les característiques principals destaquen les seves 14 entrades/sortides digitals, 6 entrades analògiques, un oscil·lador de cristall de 16Mhz i un port USB permetent una comunicació adequada amb un computador extern, com es detalla en la secció 3.

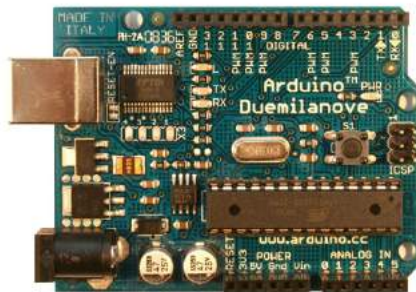


Figura 2.1: Placa de microcontrolador *Arduino Duemilanove*.

D'aquesta manera s'estableix la comunicació entre el computador principal i els dispositius físics que permeten el moviment del LIDAR i en monitoritzen el seu estat. La figura 2.2 il·lustra el paper de la placa.

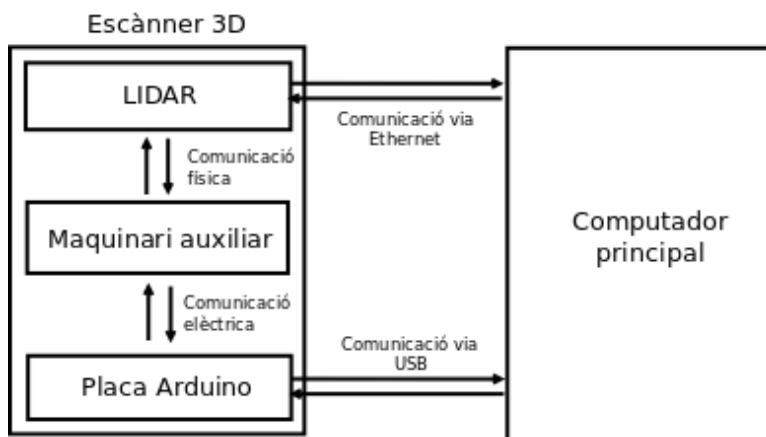


Figura 2.2: Esquema de les comunicacions de l'escàner 3D.

2.2.2 Motor pas a pas

L'element generador de moviment de l'escàner és un motor pas a pas bipolar. En concret, s'utilitza el model Nema 17 de National Instruments. S'en pot trobar una fitxa tècnica a l'annex C.3. Com a característiques principals destaquen un pas de 1.8° , una precisió angular del 3% i una velocitat màxima de 3000rpm.

L'ús d'un motor pas a pas facilita el control de la posició angular del LIDAR. Gràcies al seu sistema de control, es pot conèixer l'increment angular que s'ha produït a partir de les consignes enviades al motor. No és necessari emprar codificadors angulars òptics ni cap altre sistema extern per a monitoritzar els canvis de posició angular. Els mecanismes de control del Nema 17 no només permet tenir informació de la posició sinó que també permet controlar-la de manera senzilla.

Per a portar a terme el control del motor, s'utilitza un controlador, la placa de Pololu basada en el circuit integrat DRV8825, la fitxa tècnica del qual es pot veure a l'annex C.4. Ja que la potència que les sortides de la placa *Arduino* poden subministrar no és suficient per a alimentar el motor, es necessiten circuits auxiliars. Aquest circuit auxiliar consta d'entrades diferents per l'electrònica de control i l'alimentació del motor, com es pot observar a la figura 2.3. S'encarrega d'enviar les consignes adequades al motor segons les comandes rebudes des de la placa *Arduino*. Gràcies a això el control del motor és flexible i senzill.

Una de les seves funcionalitats més importants és el *microstepping* que permet augmentar la resolució angular del motor fins a un factor 32, en el cas del controlador DRV8825. La longitud angular transcorreguda en un sol pas és una fracció del pas complet (1.8), regulable mitjanant els pins M0, M1 i M2 com es mostra a la taula 2.1.

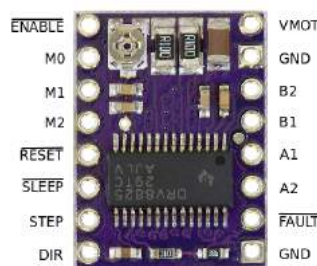


Figura 2.3: Pinout de la placa DRV8825 controladora del motor pas a pas.

A més, el controlador DRV8825 controla la intensitat de sortida cap al motor per tal de protegir-lo enfront de sobreintensitats.

2.2.3 Impressió 3D

Per a implementar el suport del LIDAR i els mecanismes de transmissió s'utilitza la impressió 3D. El seu principi de funcionament és la fabricació per addició, és a dir, les peces es construeixen a partir de la superposició de successives capes de material. Aquesta tecnologia permet crear peces tridimensionals de diferents materials a partir de dissenys desenvolupats a mitjanant eines CAD (*Computer-Aided Design*). Més concretament, el material utilitzat és el plàstic ABS (de l'anglès *Acrylonitrile Butadiene Styrene*) i el programari de disseny utilitzat és *Solidworks*. Tant *Solidworks* com la

M0	M1	M2	Resolució angular
Baix	Baix	Baix	Pas complet
Alt	Baix	Baix	Mig pas
Baix	Alt	Baix	1/4 de pas
Alt	Alt	Baix	1/8 de pas
Baix	Baix	Alt	1/16 de pas
Alt	Baix	Alt	1/32 de pas
Baix	Alt	Alt	1/32 de pas
Alt	Alt	Alt	1/32 de pas

Taula 2.1: Resolució angular en funció de les entrades M0, M1 i M2.

impresora 3D utilitzada satisfan les necessitats de precisió i versatilitat necessàries d'aquest treball fins i tot per al disseny d'engranatges a mida.

2.2.4 Sensor reflectiu

La necessitat de poder diferenciar un punt de referència en el gir del LIDAR porta a emprar un sensor reflectiu capaç de distingir superfícies reflectants. El sensor elegit és un OPB704. Està format per un LED de llum infraroja i d'un fototransistor NPN muntats un al costat de l'altre de manera que en presència d'una superfície reflectant a la proximitat del caire receptor del sensor, la llum del LED activa el fototransistor, com es pot veure en la figura 2.4. Tant el díode com el fototransistor poden dissipar una potència de fins a 100mW.

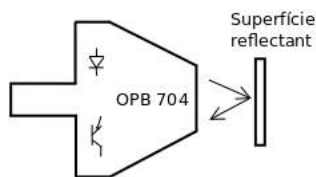


Figura 2.4: Principi de funcionament del sensor OPB704.

La transmissió de la informació del sensor a la placa *Arduino*, es fa mitjançant el circuit de condicionament de la figura 2.5 on el díode i el fototransistor són els que componen internament el sensor reflectiu. La resistència R1 permet polaritzar el LED en intensitat. La resistència R2 es pot considerar una resistència *pull-down* de manera que quan el fototransistor no condueix el voltatge de sortida és el de massa i quan sí que condueix, la tensió de sortida s'apropa a la d'alimentació. D'aquesta manera V_o indica l'estat de conducció del fototransistor permetent detectar objectes reflectants a la proximitat del sensor. Per a mesurar aquesta sortida, es connecta a una entrada analògica de la placa *Arduino*. Els valors elegits per a les resistències són $R1=360\ \Omega$ i $R2=11\ k\Omega$.

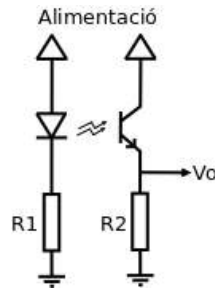


Figura 2.5: Circuit acondicionador per al sensor reflectiu.

2.2.5 Elements mecànics addicionals

Amb l'objectiu de complir amb els requeriments físics, s'utilitzen elements mecànics addicionals, cada un amb una funció específica.

Contacte giratori (*slip ring*)

Entre la part mòbil i la part fixa del suport hi ha d'haver comunicació elèctrica, ja que el LIDAR, situat al rotor necessita transmetre la informació a la base fixa. Per altra banda, una de les funcionalitats a implementar és la possibilitat de gir continu de la part mòbil. Per tal de complir amb els dos requeriments es necessita un contacte capaç de mantenir la connexió siguin quins siguin la posició i el recorregut angular del rotor.

Per a solucionar aquest problema s'utilitza un contacte giratori (*slip ring*) que uneix diferents fils conductors mitjançant dues parts que poden rotar una respecte de l'altra mantenint el contacte elèctric. El model elegit és un SRC022A-6 de sis fils. A la figura 2.6 es mostren les seves dimensions i aspecte extern.

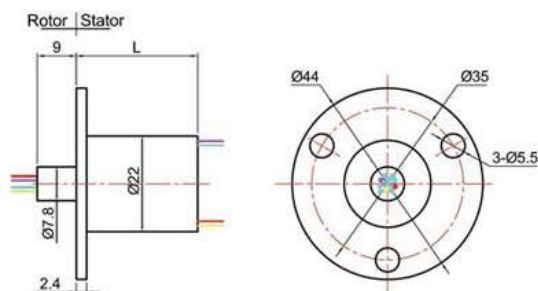


Figura 2.6: Dimensions del *slip ring*.

En el cas del SRC022A-6, $L=19$. Totes les mesures són en *mm*.

Rodament

La unió entre la part mòbil i la part fixa del suport es fa per mitjà d'un rodament. D'aquesta manera el gir s'efectua amb molt baixa fricció. El model utilitzat és 619-0402 de RS. Les seves característiques es mostren en la taula 2.2.

Diàmetre interior mm	Diàmetre exterior mm	Amplada mm	Càrrega dinàmica N
20	32	7	3.600
Càrrega estàtica N	Velocitat màxima (grassa) rpm	Velocitat màxima (oli) rpm	Pes Kg
2.223	18900	22500	0.019

Taula 2.2: Característiques del rodament 619-0402.

Placa de PVC

L'element d'unió entre els elements mecànics és una placa de PVC. La seva cara superior és la superfície de referència de l'escàner.

2.3 Disseny elegit

En resposta a les necessitats funcionals ja descrites, es presenta el disseny explicat a continuació. Conceptualment, s'aborda la solució proposada des d'un enfocament mecànic i electrònic separatament.

2.3.1 Elements mecànics

Funcionament general

La figura 2.7 mostra el disseny mecànic. El LIDAR reposa sobre la part mòbil del suport d'on se subjecta cargolat per la part inferior. El cablejat del LIDAR es comunica amb l'estator mitjançant el contacte giratori *slip ring* passant per l'interior del tub que suporta la plataforma per al LIDAR. L'extrem terminal d'aquest tub està encaixat dins el rodament, que al seu torn està encaixat a pressió dins la placa de PVC. La figura 2.8 mostra aquest acoblament.

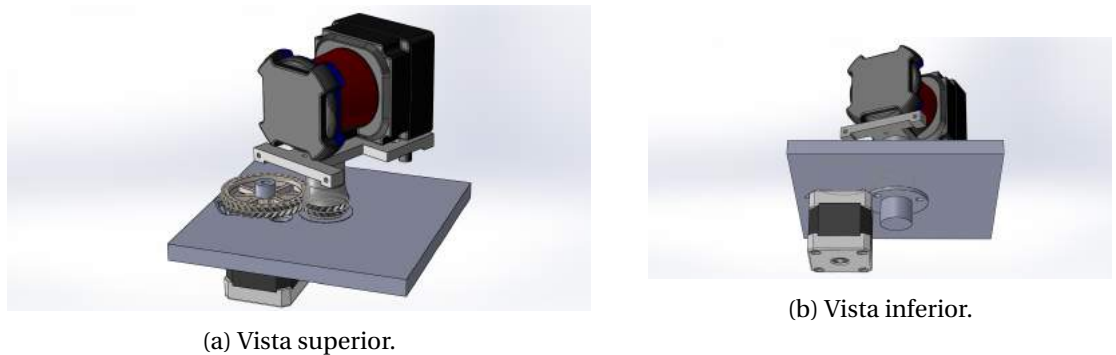


Figura 2.7: Ensamblatge d'elements mecànics de l'escàner.

El motor pas a pas està cargolat a la placa de PVC. La transmissió de moviment es porta a terme a través de dos engranatges solidaris amb l'eix del motor i amb l'eix de rotació de la part mòbil, respectivament. Els detalls de cada part es discuteixen en els següents apartats.

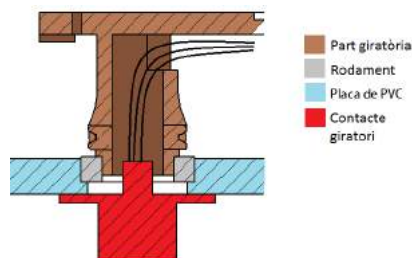


Figura 2.8: Acoblament entre les parts mòbil i fixa.

Plataforma giratòria

És la part de l'escàner en contacte amb el LIDAR. La figura 2.9 mostra el seu disseny i dimensions. Aquesta peça s'ha implementat mitjançant impressió 3D. Està composta per un pla que segueix la figura del LIDAR i un tub que el comunica amb la resta de l'escàner.

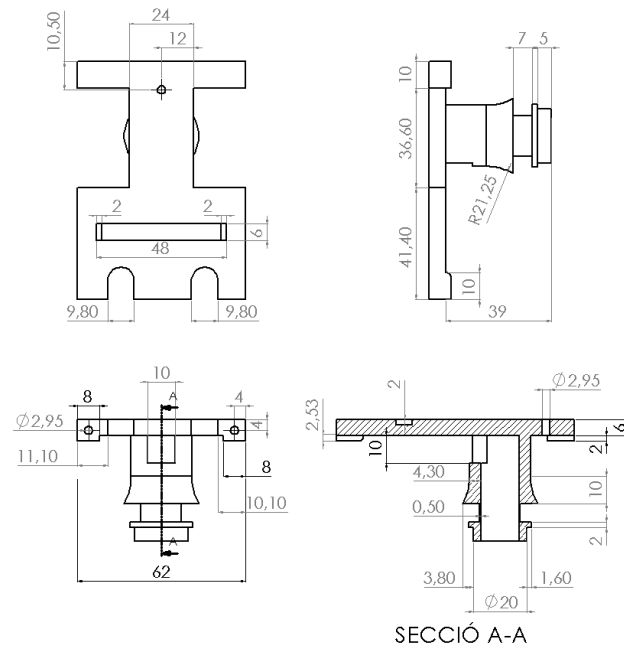


Figura 2.9: Plataforma de contacte per al Hokuyo UTM-30LX-EW.
Totes les mesures en *mm*.

Als extrems superior i inferior hi ha orificis per a cargolar plaques de subjecció per al LIDAR. La placa del costat inferior, també té orificis per a cargolar el sensor làser. Al suport, s'hi ha dissenyat una canaleta perquè es pugui embridar el cablejat del LIDAR. A la part superior hi ha un orifici per a introduir-hi una barra d'ajust de la inclinació del sensor làser.

La longitud del tub és un paràmetre crític, ja que condiciona tant el balanceig del LIDAR com el seu camp de visió. El valor elegit, 39cm, que és un compromís entre els dos aspectes. L'engranatge de l'eix de rotació del LIDAR està unit al tub i imprès conjuntament com es mostra a la figura 2.10. El centre del tub, ja que és l'eix de rotació, està centrat al receptor òptic del LIDAR.

En el tub on es subjecta el LIDAR s'hi ha dibuixat una marca blanca per a que el sensor òptic pugui detectar un punt singular.

Mecanismes de transmissió

Els engranatges encarregats de la transmissió del moviment són de tipus doble heli-coïdal. Estan representats a les figures 2.10, 2.11 i 2.12. El nombre de dents de l'engranatge del motor (Z_1) és 36 i en el cas de l'eix de gir del làser (Z_2) és 18. Per tant, la

2. REQUERIMENTS FÍSICS I DISSENY DE L'ESCÀNER

relació de transmissió (n) és de 2.

$$n = \frac{Z_1}{Z_2} = 2 \quad (2.1)$$

Així, configurant el controlador a un octau de pas, per un rang de velocitats del motor de 40Hz a 800Hz, s'obté un rang de velocitats de gir de 3rpm a 60rpm.

$$\frac{f \text{ pas}_{motor}}{1 \text{ s}} \cdot \frac{1.8/8^\circ_{motor}}{1 \text{ pas}_{motor}} \cdot \frac{1 \text{ rev}_{motor}}{360^\circ_{motor}} \cdot \frac{2 \text{ rev}_{LIDAR}}{1 \text{ rev}_{motor}} \cdot \frac{60 \text{ s}}{1 \text{ min}} = 0.075 \cdot f \text{ rpm} \quad (2.2)$$

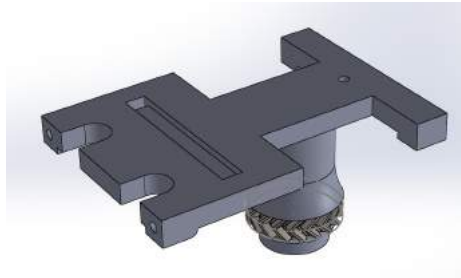


Figura 2.10: Eix mòbil. Ensamblatge amb el suport

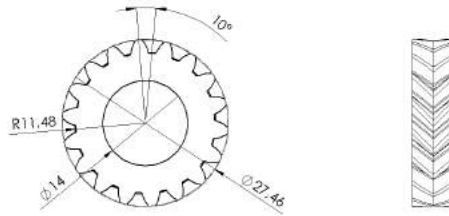


Figura 2.11: Eix mòbil. Engranatge de l'eix de gir del suport

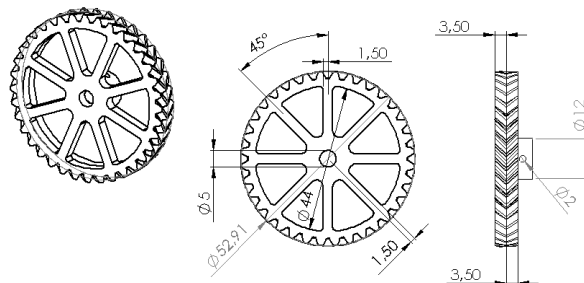


Figura 2.12: Engranatge del motor pas a pas.
Totes les mesures en mm .

Disseny de la placa de PVC

La figura 2.13 mostra la placa de PVC amb els orificis corresponents per a subjectar el rodament i cargolar el motor.

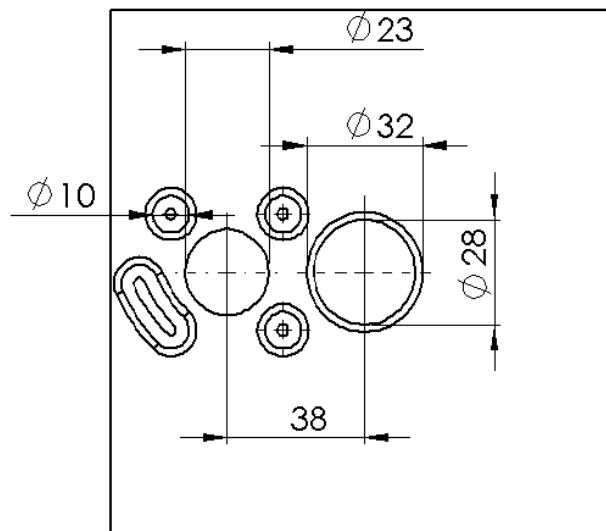


Figura 2.13: Placa de PVC.

Al centre, orifici per al rodament; a l'esquerra, orificis pel motor. Totes les mesures en *mm*.

2.3.2 Elements elèctrics i electrònics

L'alimentació, tant del motor com del LIDAR és una font externa de 12V. L'alimentació de la placa *Arduino* i la transmissió de dades amb el computador principal es fa a través de USB.

Les connexions de la placa *Arduino* elegides són les que es mostren en la taula 2.3. A més, la massa de la placa es connecta amb l'entrada de terra de l'electrònica de control (*gnd*) i amb la massa del circuit de condicionament del sensor reflectiu (figura 2.5).

Per a les connexions entre elements s'ha dissenyat una placa de circuit imprès. El programa utilitzat per al disseny ha estat *Fritzing*. La figura 2.14 en mostra el seu aspecte. El circuit d'acondicionament del sensor reflectiu està integrat dins la placa. S'hi ha fet un pla de terra. El amplada de les pistes és de 42 mil per a les pistes d'alimentació i 32 mil per la resta.

Pin	Connexió	E/S	Pin	Connexió	E/S
13	Dir (Pololu)	S	12	Step (Pololu)	S
11	Sleep (Pololu)	S	10	Reset (Pololu)	S
9	M2 (Pololu)	S	8	M1 (Pololu)	S
7	M0 (Pololu)	S	6	Enable (Pololu)	S
5	Fault (Pololu)	S	4	Alimentació OPB	S
A0	Lectura OPB	E			

Taula 2.3: Entrades i sortides de la placa *Arduino*.

La marca "*Pololu*" és una referència a la placa controladora del motor

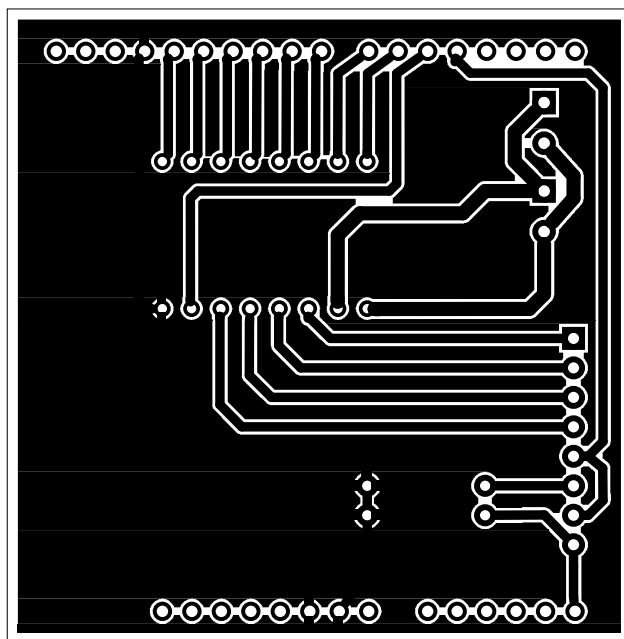


Figura 2.14: Layout de la placa de circuit imprès.

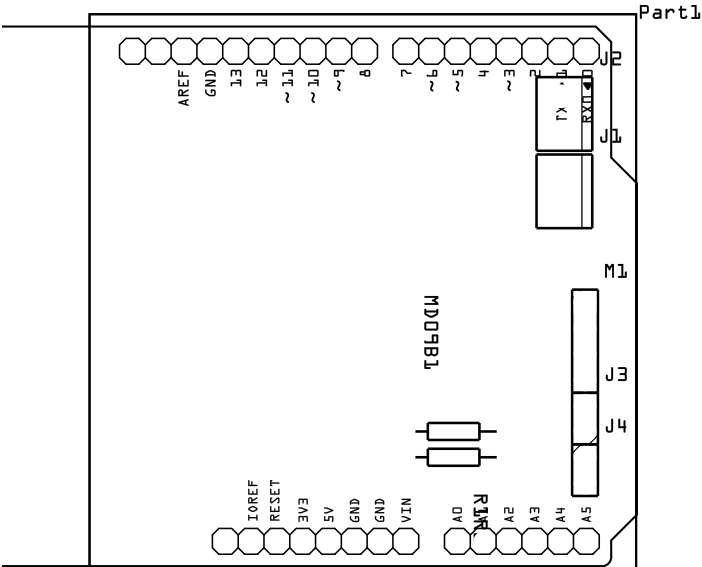


Figura 2.15: Distribució dels elements.

Les files de connectors de la part superior i inferior són les corresponents a la placa *Arduino*. Al centre hi ha les connexions per a la placa DRV8825 i a la dreta, de dalt a baix, l'alimentació externa, la del LIDAR, les connexions amb el motor i el sensor reflectiu.

DESENVOLUPAMENT DEL PROGRAMARI

3.1 Introducció i entorn de desenvolupament

Tot el programari necessari per a portar a terme el treball està basat en *ROS (Robot Operating System)*. Aquest sistema té un suport complet només en plataformes linux. *ROS* és un entorn de treball que proporciona biblioteques i altres eines enfocades al desenvolupament d'aplicacions dins l'àmbit de la robòtica. S'utilitzen les eines de processament d'informació que proporciona *ROS* per a missatges estàndard del tipus escaneig làser (*LaserScan*) i nígul de punts (*PointCloud*) així com protocols propis de comunicació i controladors desenvolupats externament a aquest treball, com s'explica en l'apartat 3.2. Tot el desenvolupament del codi s'ha fet emprant la distribució *ROS Hydro Medusa*.

El codi de les aplicacions de *ROS* s'estructura en paquets que al seu torn poden estar dividits en diferents nodes. El codi font de cada node es pot escriure en diferents llenguatges, en el cas d'aquest treball, s'ha elegit C++.

Per a entendre el codi desenvolupat en aquest treball, és necessari aclarir alguns conceptes bàsics del sistema *ROS*. Cada node es pot comunicar amb la resta mitjançant uns protocols definits pel sistema: missatges i serveis. En la comunicació per missatges s'estableixen uns canals (tòpics) en els quals hi ha nodes publicadors i subscriptors que comparteixen missatges tipus, definits externament al codi dels nodes. Aquests missatges poden ser propis del sistema *ROS* o definits per l'usuari.

La comunicació mitjançant serveis es porta a terme entre dos nodes. Un dels quals és un client del servei i l'altre és qui l'ofereix i dona resposta a la sol·licitud del servei. En aquest treball s'utilitzen serveis però no se'n defineix cap.

ROS dona la possibilitat d'enregistrar els missatges que es publiquen als tòpics en un arxiu *bagfile*. Aquests arxius es poden tornar a executar simulant les comunicacions que hi ha enregistrades.

Per a llançar diferents nodes de manera automàtica *ROS* proporciona la possibilitat d'executar arxius tipus *launch*. Aquests arxius permeten definir els paràmetres que s'han d'utilitzar per a cada node, entre altres funcions.

3.2 Codi extern

En aquest treball s'utilitza a més del propi entorn de treball ROS, codi genèric desenvolupat externament. El controlador del Hokuyo UTM-30LX-EW que s'utilitza és el node *urg_node* del paquet *urg_node*. S'encarrega de comunicar les lectures del LIDAR al sistema ROS. D'acord amb la funcionalitat *multiecho* del sensor, publica, entre altres, tres tòpics: *first*, *last* i *most_intense*. Aquests corresponen a escanejos làser considerant els rebots del feix làser més pròxims, més allunyats i més intensos, respectivament. Els missatges associats a aquests tòpics són del tipus *sensor_msgs/LaserScan*.

La comunicació del sistema ROS amb la placa *Arduino* es fa a través del node *serial_node* del paquet *rosserial_python*. Aquest permet la integració del codi del microcontrolador dins el sistema ROS aconseguint crear un node virtual al microcontrolador, emprant el node *serial_node* com a connector.

Per a acumular diferents lectures del LIDAR i compondre-les en un escaneig 3D s'utilitza el node *laser_scan_assembler* del paquet *laser_assembler*. Aquest node rep missatges de tipus *LaserScan* i *tf* i en retorna del tipus *PointCloud*. Per a permetre sol·licitar les dades acumulades el node proporciona un servei, *AssembleScans*, que retorna un missatge de tipus *sensor_msgs/PointCloud*.

La visualització dels escaneigs es fa gràcies a l'aplicació de visualització RViz integrada en la distribució de ROS en el node *rviz* del paquet *rviz*.

3.3 Solució elegida i característiques

3.3.1 Funcionament general i justificació

L'estructura de la solució elegida es basa en un conjunt de nodes dins el sistema ROS i d'un programa executat sobre la placa *Arduino*, com mostra la figura 3.1. Per a implementar el control de l'escàner s'ha desenvolupat un paquet de ROS (*lidar_scan*) amb dos nodes: *lidar_scan_node* i *pc_snapshotter*. D'aquesta manera es controla la posició angular del làser i es porten a terme les operacions necessàries per a compondre les lectures del LIDAR en un escaneig 3D de l'entorn.

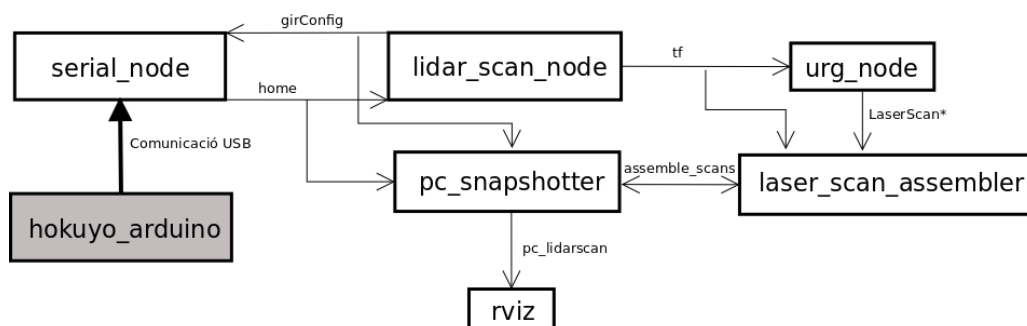


Figura 3.1: Esquema principal de l'estructura del programari.

S'hi mostren els nodes involucrats i les comunicacions entre ells. (*) Els missatges de *LaserScan* es poden comunicar a través de diferents tòpics. En la resta de casos l'etiqueta indica el tòpic o servei.

L'únic servei que s'utilitza és el *assemble_scans* del node *laser_scan_assembler*, sol·licitat pel node *pc_snapshotter*. Les altres comunicacions entre nodes s'efectuen per missatges. La taula 3.1 relaciona cada tòpic amb el missatge a què està associat.

Tòpic	Tipus de missatge
girConfig	lidar_scan/codiGir
home	std_msgs/Empty
tf	tf2_msgs/TfMessage
most_intense	sensor_msgs/LaserScan
first	sensor_msgs/LaserScan
last	sensor_msgs/LaserScan
pc_lidarscan	sensor_msgs/PointCloud

Taula 3.1: Tòpics usats i tipus associats.

Ja que la funcionalitat *multiecho* del LIDAR permet obtenir informació sobre diferents rebots de cada feix làser, es poden emprar tres tòpics diferents per a publicar els missatges del tipus *LaserScan* referents al primer rebot, l'últim i el més intens: *first*, *last* i *most_intense*, respectivament.

A l'hora de repartir les funcionalitats entre el programa *hokuyo_arduino* i el paquet *lidar_scan* s'ha tendit a encarregar tasques al paquet *lidar_scan* executat en un ordinador, a causa de la limitada capacitat de processament del microcontrolador AT-mega168 i a la feblesa de la connexió de la placa Arduino amb el sistema ROS.

El paquet *lidar_scan* s'encarrega de les següents tasques:

- Control de l'estat de l'escàner.
- Comunicació amb nodes auxiliars.
- Publicació dels escaneigs 3D.
- Càlculs geomètrics referents a la posició i publicació de les posicions d'interès de l'escàner.

El programa *hokuyo_arduino* s'encarrega de les següents tasques:

- Donar les consignes adequades als dispositius electrònics de l'escàner.
- Controlar la periodicitat en què s'envia l'ordre d'efectuar un pas al motor pas a pas.
- Rotar el LIDAR fins que aquest arribi a la seva posició inicial.

Tot el programari desenvolupat pot treballar fent servir missatges *LaserScan* de qualsevol dels tres tòpics que proporciona el controlador del Hokuyo UTM-30LX-EW: *first*, *last* i *most_intense*.

Tot i que el fabricant de la placa DRV825, controladora del motor pas a pas, assegura poder controlar passos 32 vegades menors que el pas nominal, els experiments amb el motor i el controlador mostren que l'angle mínim que es pot controlar és un

octau de pas. Aquesta ha estat la configuració que s'ha utilitzat, ja que permet que el motor treballi a freqüències més elevades, amb menys vibracions.

Durant un escaneig planari del LIDAR el motor pas a pas es mou, de manera que es comet un error en la lectura. Aquest error depèn de la freqüència de funcionament del motor, de manera que si el motor es moguéssim contínuament en lloc d'avançar a passos discrets, l'angle recorregut durant un escaneig 2D seria proporcional a la freqüència de funcionament del motor amb un factor 0.01125 si s'expressa en graus. L'angle en qüestió no superaria els 1.5° en escanejos de resolució a partir de 250 escanejos 2D per revolució. Aconseguir escanejos amb resolucions similars fent que per a cada lectura 2D vàlida el LIDAR estès quiet, provocaria que els escanejos duressin un temps inassumible per a entorns dinàmics. En aquest cas hi hauria moltes vibracions a causa del moviment a baixa freqüència del motor.

$$\frac{f \text{ pas}}{1 \text{ s}} \cdot \frac{0.025 \text{ s}}{1 \text{ esc.2D}} \cdot \frac{1.8/8 \text{ }^\circ_{\text{motor}}}{1 \text{ pas}} \cdot \frac{2 \text{ }^\circ_{\text{LIDAR}}}{1 \text{ }^\circ_{\text{motor}}} = 0.01125 \cdot f \frac{\text{ }^\circ_{\text{LIDAR}}}{\text{esc.2D}} \quad (3.1)$$

Un cop que la part mòbil de l'escàner passa pel punt singular i aquesta informació es comunica a través d'un missatge al tòpic *home*, la placa *Arduino* deixa de comunicar-se amb el sistema *ROS*. Durant l'execució de l'escaneig tant els nodes *lidar_scan_node* i *pc_snapshotter* com el programa *hokuyo_arduino* duen un compte del temps de manera independent. Això pot provocar desajustos entre el sistema físic i la representació que té d'ell el sistema *ROS*. Tot i això, errors en l'escaneig deguts a la comunicació entre nodes han fet que la solució passi per un control separat del temps basat en temporitzadors.

Per a calcular el desplaçament angular que s'ha portat a terme els dos nodes del paquet *lidar_scan* assumeixen un model de moviment continu del sensor. Així també s'introdueix un error en considerar que parts fraccionàries de passos del motor impliquen desplaçaments angulars proporcionals envers de tractar amb valors de posició angular discrets. Aquest error, com a màxim pot fer que un escaneig 2D se situï a un pas de diferència d'allà on estaria amb un model discret, això és 0.45° com a màxim. No es tracta d'un error acumulatiu.

En el desenvolupament del treball s'ha intentat que el control del temps fos únic per a tots els nodes, fent que la placa *Arduino* comunicués els passos que s'han ordenat fer al motor. Tanmateix en intentar establir una comunicació suficientment ràpida per tal que a l'arribada de cada escaneig 2D aquest disposi d'informació actualitzada sobre l'orientació a què s'ha d'ubicar, es produeixen, de tant en tant, errors de sincronització entre el node *serial_node* i la placa *Arduino*. A més, un càlcul de posició segons un model teòric, com s'ha portat a terme, permet calcular la posició teòrica del motor pas a pas just a l'instant en què arriba un escaneig 2D, assegurant que, en qualsevol cas, cada escaneig làser s'associarà amb la posició corresponent al seu instant d'arribada segons el model. En cas contrari, en publicar la posició del LIDAR amb independència del temps de publicació dels missatges *LaserScan* pot provocar que diverses lectures consecutives del sensor làser s'associïn a una mateixa orientació.

Els resultats experimentals han mostrat que fent servir tot el rang d'escaneig del LIDAR s'aconsegueixen pitjors resultats que emprant-ne només la meitat, fent que un costat i altre del sensor làser donin resultats lleugerament diferents. Per aquest motiu, en aquest treball només s'empren les dades referents als feixos làsers corresponents al rang d'escaneig comprès entre 0 i 135° del LIDAR. D'aquesta manera el temps neces-

sari per a efectuar un escaneig 3D augmenta al doble per a la mateixa resolució si ho comparem amb un sistema equivalent que utilitzi tot el rang d'escaneig.

El motiu pel qual el paquet *lidar_scan* consta de dos nodes en lloc d'integrar totes les funcions en un de sol és purament empíric. Fer la sol·licitud dels níguls de punts i publicar-los en el node *lidar_scan_node* provoca errors en la publicació de les posicions del LIDAR en el moment en què es publica un nigul de punts. Alguna vegada la publicació del nigul de punts durant l'escaneig, fins i tot fent-se en un node independent sembla provocar errors en la publicació de les posicions. Relegant aquestes funcions a un node independent, el *pc_snapshotter* també permet una reconstrucció dels níguls de punts *offline*, a partir de les dades dels escaneigs 2D enregistrades en un *bagfile* en un altre moment.

3.3.2 Paquet *lidar_scan*

Aquest paquet és l'encarregat de gestionar la informació dels sensors per compondre-la en escanejos 3D.

En el seu node principal, el *lidar_scan_node*, l'únic paràmetre d'entrada es diu *periode* i representa el període en què el motor pas a pas ha d'efectuar un pas. Se subscriu a un únic tòpic, *home*, que indica el moment en què el rotor de l'escàner es troba en el punt de referència per primera vegada.

Els missatges publicats pel node es poden observar al següent llistat de tòpics.

- **tf** (*tf2_msgs/TfMessage*).
Indica la relació de posició i orientació entre un sistema de coordenades situat a la base de l'escàner i un altre situat al receptor òptic del LIDAR.
- **girConfig** (*lidar_scan/codiGir*).
És un missatge propi del paquet. Indica el mode de funcionament que ha de seguir l'escàner. Té dos camps de dades:
 - **header**, de tipus *std_msgs/Header* ¹
 - **per**, de tipus *int64*. Indica el període en què s'han de succeir els passos del motor pas a pas.

Tot i només transmetre informació sobre el període dels passos del motor, s'ha elegit crear un missatge propi, ja que d'aquesta manera es pot modificar per a implementar més funcionalitats com ara definir diferents rangs d'escaneig per a fer escanejos selectius.

Internament, el node *lidar_scan_node* es basa en la classe *lidarScan*, l'esquema de funcionament de la qual es pot veure a la figura 3.2. Controla l'estat de l'escàner i publica la posició del LIDAR. Cada escaneig 2D s'ha de posicionar a un sistema de coordenades corresponent a la posició del LIDAR en el moment de l'escaneig. Per a aconseguir-ho, el node se subscriu als escanejos 2D, captura l'instant de temps en què s'han publicat i publica la posició associada a l'escaneig 2D, calculada a partir de

¹ Es tracta d'un tipus de missatge especial que proporciona funcionalitats addicionals a la publicació de missatges

la posició anterior i l'espai que s'hauria d'haver recorregut en un escaneig del LIDAR (25ms). En el cas de treballar a un octau de pas:

$$25\text{ ms} \cdot \frac{1\text{ pas}}{\text{periode } \mu\text{s}} \cdot \frac{\pi/800\text{ rad}}{1\text{ pas}} \cdot \frac{10^3\text{ } \mu\text{s}}{1\text{ ms}} = 31.25\pi\text{ rad} \quad (3.2)$$

Aquest càlcul de la posició assumeix que els missatges de tipus *LaserScan* provinents del controlador del LIDAR es succeeixen de manera totalment regular i que els passos entre cada escaneig 2D coincideixen amb longitud amb els resultats teòrics i són tants com correspon a un interval de 25ms, essent aquest un nombre no necessàriament enter. El sistema físic real, però, pot estar esbiaixat i presentar un error sistemàtic que es vagi acumulant, ja que no hi ha comunicació entre la placa *Arduino* i el node *lidar_scan_node* ni, per tant, possibilitat de corregir-lo.

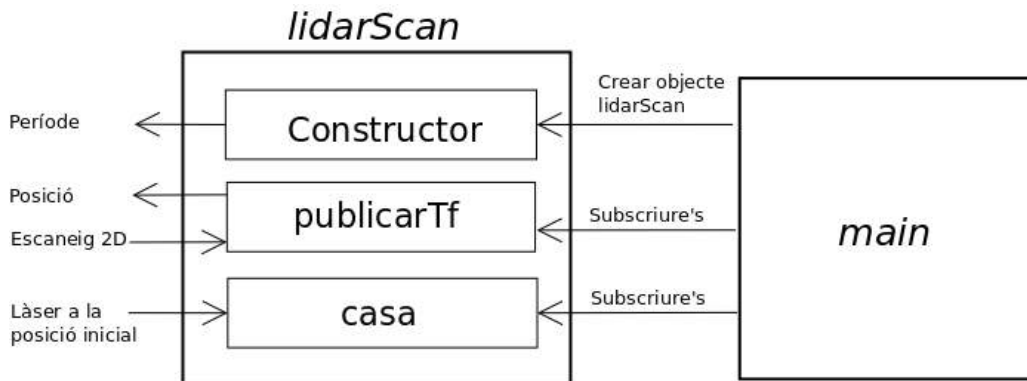


Figura 3.2: Esquema de funcionament de la classe *lidarScan*.
Els blocs dins la classe representen els seus mètodes.

La funció *main* es limita a portar a terme les següents passes:

1. Inicialització el node.
2. Elecció el període dels passos del motor a partir de la lectura del paràmetre *periode*.
3. Creació d'un objecte *lidarScan*.
4. Subscripció als tòpics *home* i el tòpic amb els missatges *LaserScan* a considerar (*first*, *last* o *most_intense*).
5. Esperar al final de l'execució de l'escaneig.

El codi complet del node es pot consultar a l'annex A.1.

El node *pc_snapshotter*, s'encarrega de publicar els escanejos 3D de manera periòdica. L'interval de temps que transcorre entre la publicació de dos nívuls de punts consecutius és el necessari per a efectuar un escaneig complet (360° del LIDAR) segons el model teòric. Treballant a un octau de pas, el període d'escaneig 3D es calcula a partir de la variable *periode* que expressa el període de funcionament del motor pas a pas:

$$\frac{\text{periode } \mu s}{1 \text{ pas}} \cdot \frac{1 \text{ pas}}{\pi/800 \text{ rad}} \cdot \frac{2\pi \text{ rad}}{1 \text{ rev.}_{PAP}} \cdot \frac{1 \text{ rev.}_{PAP}}{2 \text{ rev.}_{LIDAR}} \cdot \frac{1 \text{ s}}{10^6 \mu s} = \frac{\text{periode}}{1250} \text{ s} \quad (3.3)$$

D'aquesta manera el control del període d'escaneig és en llaç obert, independent de l'angle real del LIDAR respecte a la base. A més, els temporitzadors de ROS no poden ser considerats de temps real. Tanmateix, la precisió en els angles d'inici i final d'escaneig no és un paràmetre crític i la implementació d'un control en llaç tancat pot causar errors.

Per a conèixer el període de funcionament del motor, el node *pc_snapshotter* se subscriu al tòpic *gir_config*. La captura de l'instant d'inici d'escaneig 3D es fa a partir de la subscripció al tòpic *home*. Les dades corresponents a un nigul de punts se sol·liciten al node acumulador *laser_scan_assembler* a través del servei *assemble_scans*. Els intervals de temps pels que s'acumulen els escanejos 2D en un nigul de punts estan marcats per les interrupcions del temporitzador. A cada interrupció es reben dades referents al temps comprès entre la interrupció que acaba d'ocórrer i la seva precedent. L'esquema de funcionament del node, basat en la classe *PCSnapshotter*

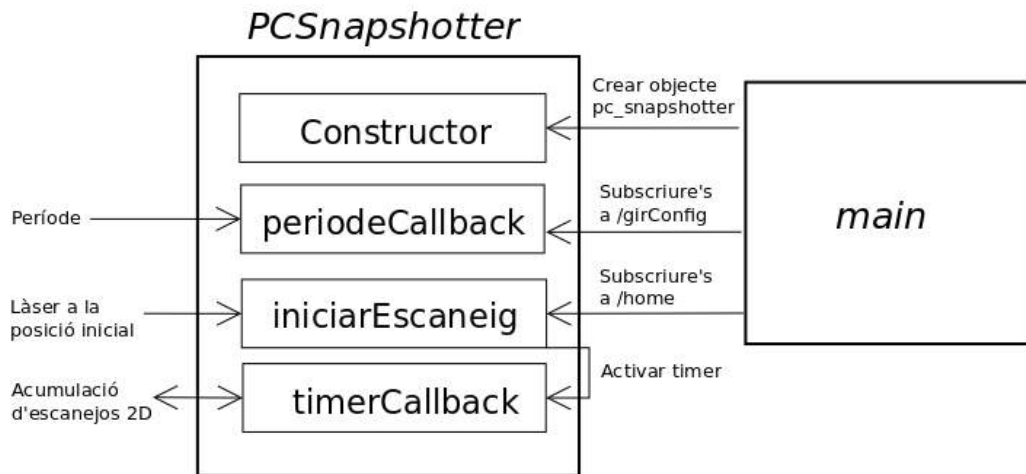


Figura 3.3: Esquema de funcionament del node *pc_snapshotter*.

Basat en la classe *PCSnapshotter*. Els blocs dins la classe representen els seus mètodes.

El codi d'aquest node es pot observar a l'annex A.2

A més del codi font dels nodes, el paquet *lidar_scan* conté la definició del missatge *codiGir*, un arxiu XML de definició del paquet i un arxiu CMakeLists.txt per a construir el paquet adequadament.

3.3.3 Programa *hokuyo_arduino*

El desenvolupament de programari per a les plaques de microcontrolador *Arduino* es porta a terme en el llenguatge C++ utilitzant les biblioteques pròpies de la placa que proporcionen alt control sobre els seus elements. Així mateix, s'utilitzen biblioteques de ROS per a crear un node virtual a la placa mitjançant el node *serial_python*

com a connector real entre el programa executat al microcontrolador i el sistema *ROS*. Aquest node està associat a tòpics ja comentats en l'apartat anterior. Concretament publica al tòpic *home* i està subscrit al tòpic *girConfig*.

La seqüència d'accions que porta a terme el programa en un escaneig 3D en ordre cronològic són les següents:

1. Inicialitzar les entrades i sortides digitals.
2. Inicialitzar el node virtual de *ROS*, subscriure's al tòpic *girConfig* i advertir la publicació al tòpic *home*.
3. Estar a l'espera d'un missatge de configuració pel tòpic *girConfig*.
4. Un cop rebut un missatge de configuració:
 - a) Portar la part mòbil del sensor al punt inicial, fent desplaçaments i lectures de l'entrada analògica associada al sensor reflectiu.
 - b) Activar un temporitzador la rutina de servei de la interrupció del qual s'encarrega d'efectuar els passos. Associar al temporitzador el període de funcionament del motor.
5. Efectuar els passos de manera periòdica.

El valor en què s'inicialitzen les sortides de la placa *Arduino* es mostren a la taula 3.2.

Pin	Connexió	Valor lògic	Pin	Connexió	Valor lògic
13	Dir (Pololu)	Baix	11	Sleep (Pololu)	Alt
10	Reset (Pololu)	Alt	9	M2 (Pololu)	Alt
8	M1 (Pololu)	Alt	7	M0 (Pololu)	Alt
6	Enable (Pololu)	Baix	4	Alimentació OPB	Alt

Taula 3.2: Valors de les sortides de la placa *Arduino*.
Només els valors de sortides constants.

El bucle principal està a l'espera d'interrupcions, que poden provenir del node *lidar_scan_node* indicant el període de funcionament desitjat del motor pas a pas o bé del temporitzador intern del microcontrolador ATmega128.

Les interrupcions internes controlen la freqüència d'enviament de les consignes adequades per tal que el motor realitzi cada pas, de manera que s'assegura la regularitat en la successió dels passos del motor.

Tot el codi del programa es troba en l'annex A.3.

3.3.4 Arxiu *lidarscan.launch*

Tots els nodes que intervenen en l'escaneig 3D són llançats des d'aquest arxiu permetent predefinir els paràmetres de cada node i executar-los amb una única instrucció. Els nodes i paràmetres que intervenen es mostren a la taula 3.3.

El controlador del Hokuyo UTM 30LX-EW es connecta al sensor que té una direcció IP "192.168.0.10". El sistema de coordenades sobre el qual publica els escanejos 2D s'anomena *hokuyo*. L'angle mínim d'escaneig és 0 radians, fent que només es consideri la meitat del rang d'escaneig del sensor (de l'angle 0 fins al màxim, per defecte, $3\pi/4$).

El node *laser_scan_assembler* enregistra un màxim de 1200 escanejos 2D abans de començar a descartar-ne. El sistema de coordenades sobre el qual s'acumulen els escanejos 2D es diu *base_laser*. Es defineix també que el tòpic d'entrada dels escanejos 2D sigui un dels publicats pel controlador del LIDAR.

S'indica al node *serial_node.py* que la connexió a la placa *Arduino* es fa a través del port `"/dev/ttyUSB0"` i que en cas que el node deixés d'executar-se s'hauria de tornar a llançar automàticament. El codi d'aquest arxiu es troba a l'annex A.4.

Paquet	Node	Paràmetre	Valor
laser_assembler	laser_scan_assembler	max_scans	1200
		fixed_frame	base_laser
urg_node	urg_node	ip_address	192.168.0.10
		frame_id	hokuyo
		angle_min	0
roscpp	serial_node.py		
rviz	rviz		
lidar_scan	pc_snapshotter		

Taula 3.3: Nodes i paràmetres de l'arxiu *lidarscan.launch*.

RESULTATS

Per a comprovar el bon funcionament de l'escàner 3D s'han realitzat una sèrie de proves al llarg del seu desenvolupament. En aquest capítol s'exposen els resultats obtinguts amb l'escàner al final d'aquest Treball Final de Grau.

4.1 Escanejos senzills

La figura 4.1 mostra un escaneig 3D amb una resolució de 640 escaneigs 2D per cada nigul de punts. Els colors indiquen la intensitat amb què s'ha rebut cada rebot dels feixos làser. Es pot observar com els objectes sobre els quals es reflecteixen els feixos làser incidents projecten una ombra al interposar-se entre el LIDAR i objectes que tenen al darrere.

A l'hora de detectar l'angle inicial, aquest no és sempre el mateix sinó que sofreix petites variacions.

És destacable l'efecte de la publicació dels missatges *PointCloud* durant l'escaneig. Si aquesta publicació es fa de manera simultània a l'escaneig, *online*, ocorren errors en la publicació de la posició del LIDAR coincidint amb els instants precedents a la publicació del nigul de punts. A més es produeix un error en la posició que es va acumulant, de manera que cada nigul de punts sembla rotat respecte a l'anterior. Per contra, si mentre l'escàner es troba en funcionament, en lloc d'utilitzar les dades per a la reconstrucció 3D simultàniament, es guarda la informació necessària en un arxiu *bagfile* i posteriorment es creen niguls de punts a partir de la informació enregistrada els errors descrits anteriorment no es produeixen.

A baixes freqüències, la part mòbil de l'escàner vibra causant errors perceptibles en l'escaneig resultant. Tant l'afecte de la publicació dels niguls de punts com el de les vibracions a baixa freqüència es poden apreciar a la figura 4.2 observant la silueta de la vista en planta d'un escaneig.

Les diferències entre els escanejos basats en els diferents tòpics dels *LaserScan* són a penes perceptibles en les proves que s'han portat a terme. La figura 4.3 compara els

4. RESULTATS

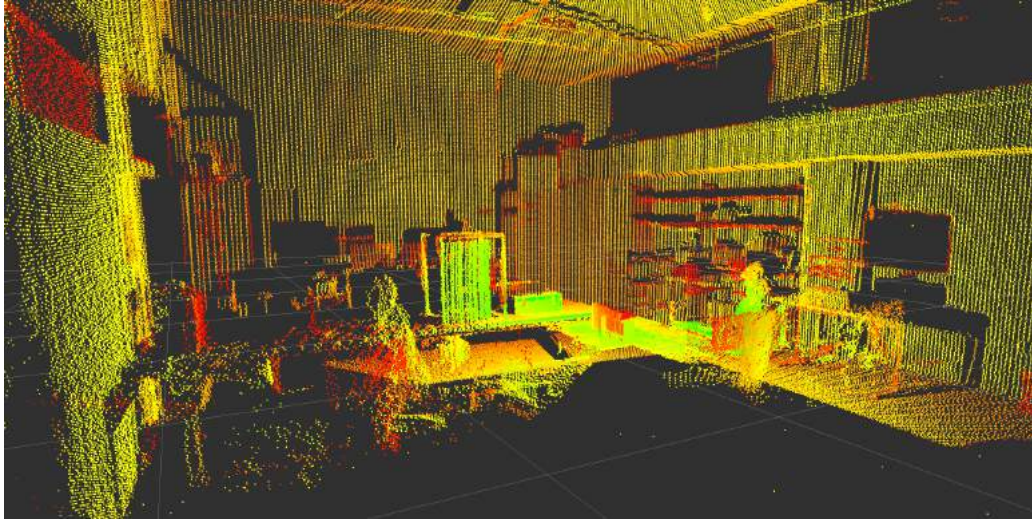
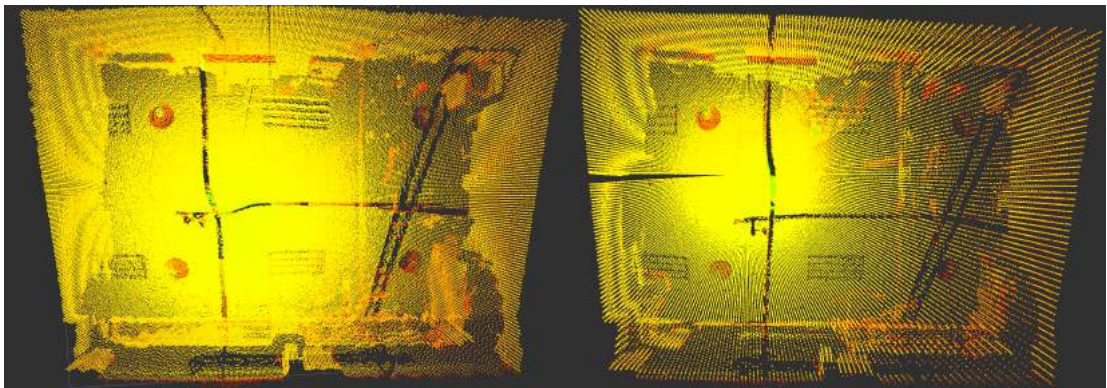


Figura 4.1: Escaneig del laboratori 126.
Realitzat a una freqüència de funcionament del motor pas a pas de 50 Hz.

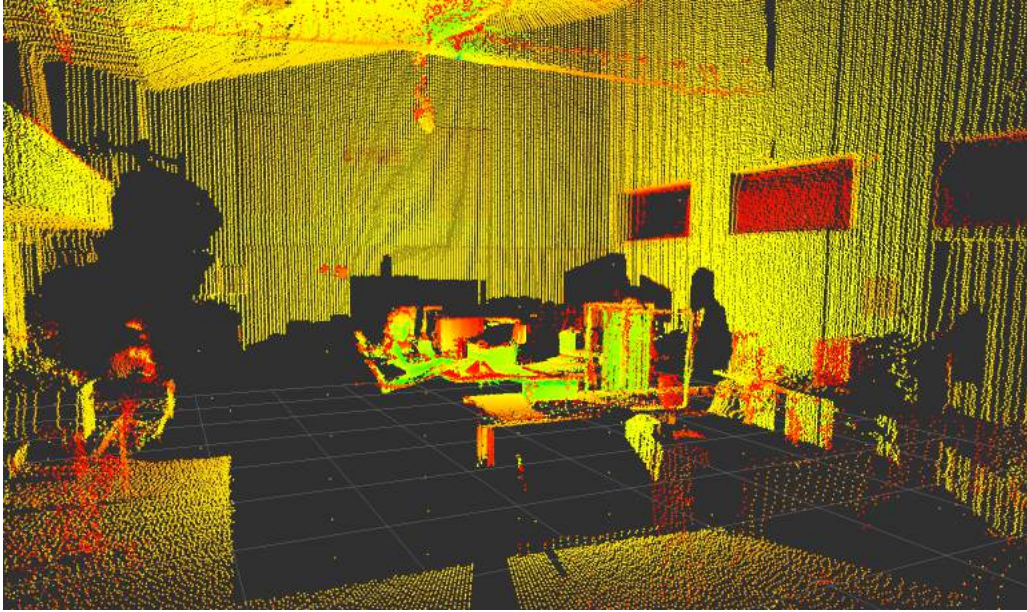


(a) Escaneig *offline* a 50Hz

(b) Escaneig *online* a 100Hz

Figura 4.2: Diferències entre escanejos.

resultats obtinguts per a diferents robots en un mateix escaneig.



(a) Escaneig a partir del tòpic *first*.



(b) Escaneig a partir del tòpic *last*.

Figura 4.3: Niguls de punts basats en diferents rebots del làser.
Les imatges pertanyen al mateix escaneig.

4.2 Escaneig compost

A partir de les dades de diferents niguls de punts, realitzats en moments diferents, es poden reconstruir per a abastar més espai. Seguidament es mostra el resultat d'un escaneig compost de diferents niguls de punts. S'han escanejat 6 punts al vestíbul del primer pis de l'edifici Anselm Turmeda.

Les figures 4.6-4.11 en mostren els resultats.

Tots els niguls de punts han estat generats en el moment de l'escaneig, fet que ha provocat imprecisions i algun rang d'escaneig sense lectures, com es pot apreciar a la

4. RESULTATS

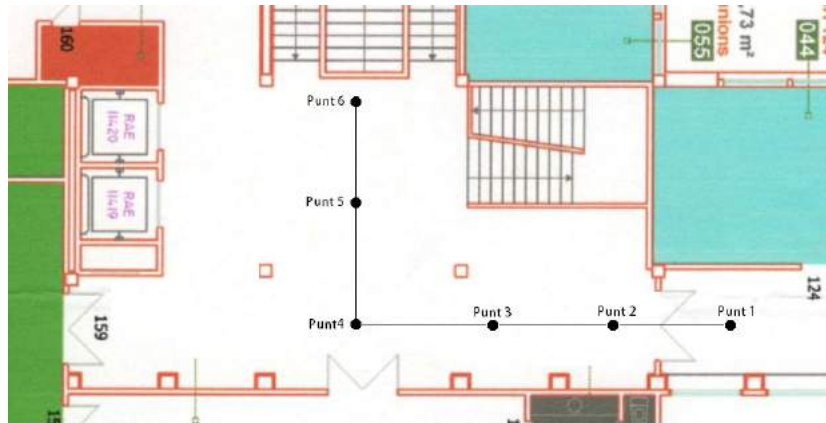


Figura 4.4: Punts d'escaneig.
Ubicació orientativa.



Figura 4.5: Presa d'escaneigs.
Vestíbul del primer pis de l'edifici Anselm Turmeda.

figura 4.9. La freqüència de funcionament del motor pas a pas ha estat en tots els casos de 50 Hz. L'error en els escanejos ha provocat que els níguls de punts no encaixin del tot entre ells. En alguns casos es pot apreciar que el sòl entre escanejos consecutius no formen un únic pla.

La figura 4.11 dóna una visió de conjunt de l'escaneig compost on es pot apreciar els errors en els escanejos, deguts, sobretot, a la publicació del nígl de punts durant l'escaneig.

Per a aconseguir unir els diferents escaneigs, cada un d'ells ha estat guardat en un arxiu *bagfile*. Aquests han estat reproduïts simultàniament al mateix temps que un

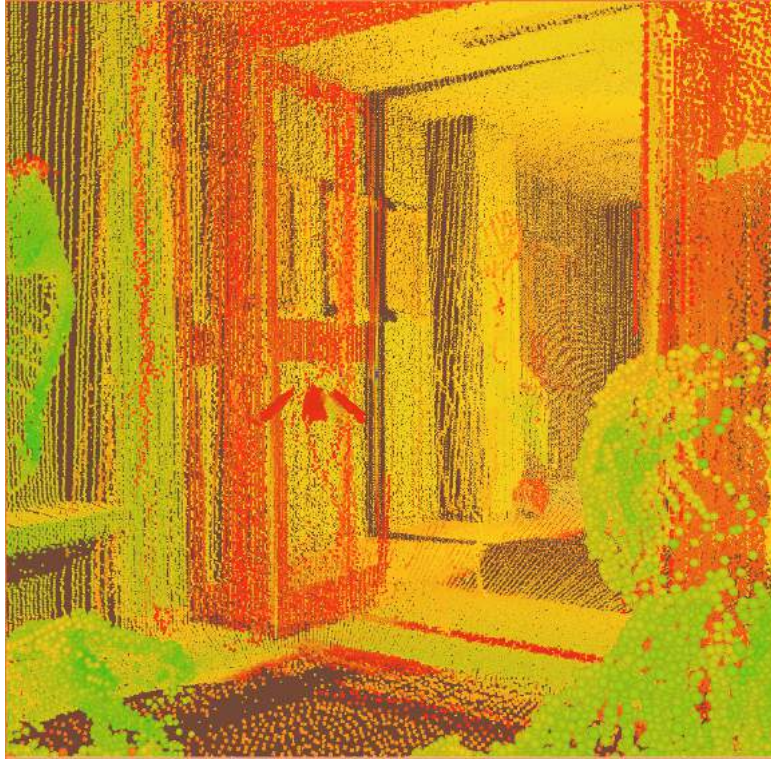


Figura 4.6: Escaneig compost. Punt 1.

node de ros, creat a tal efecte, capturava els missatges tipus *PointCloud* i en canviava el sistema de coordenades associat així com el temps de publicació. També de manera simultània s'han publicat les relacions entre els diferents sistemes de coordenades de cada nou nígul de punts permetent una correcta visualització. Tant el codi del node esmentat com el de l'arxiu *launch* encarregat de llançar tots els nodes adequadament es troben a l'annex B.

Com a conseqüència dels errors en els escanejos, hi ha diferències entre les relacions de posició dels sistemes de coordenades reals i les que s'han establert per tal que els escanejos consecutius encaixin. La taula 4.1 mostra les correccions en els sistemes de coordenades escanejats. Per a relacionar els sistemes de coordenades, tots es relacionen amb el sistema de coordenades 1.

4. RESULTATS

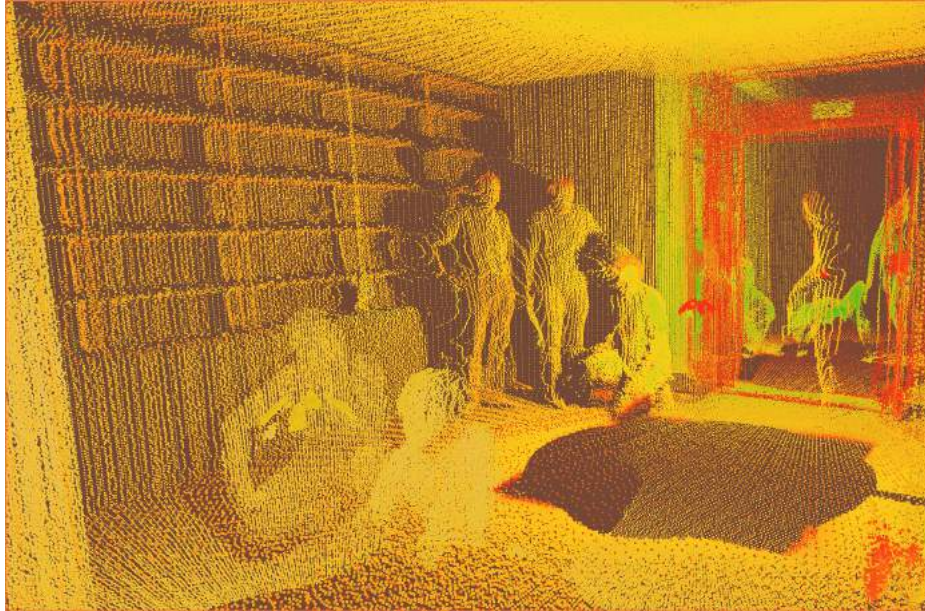


Figura 4.7: Escaneig compost. Punt 2.

SDC	Posició real	Posició corregida	Error acumulat	Error diferencial	Direcció
1	0	0	0	0	Y
2	2.9	3.2	0.3	0.3	Y
3	5.5	6.4	0.9	0.6	Y
4	8.29	9.19	0.9	0	Y
5	2.71	2.71	0	-0.147	X
6	5.42	5.42	0	0	X

Taula 4.1: Error en les relacions entre sistemes de coordenades.

Només es mostren els errors en la direcció del desplaçament real. L'error diferencial correspon a l'error entre sistemes de coordenades consecutius. Totes les mesures estan en metres.

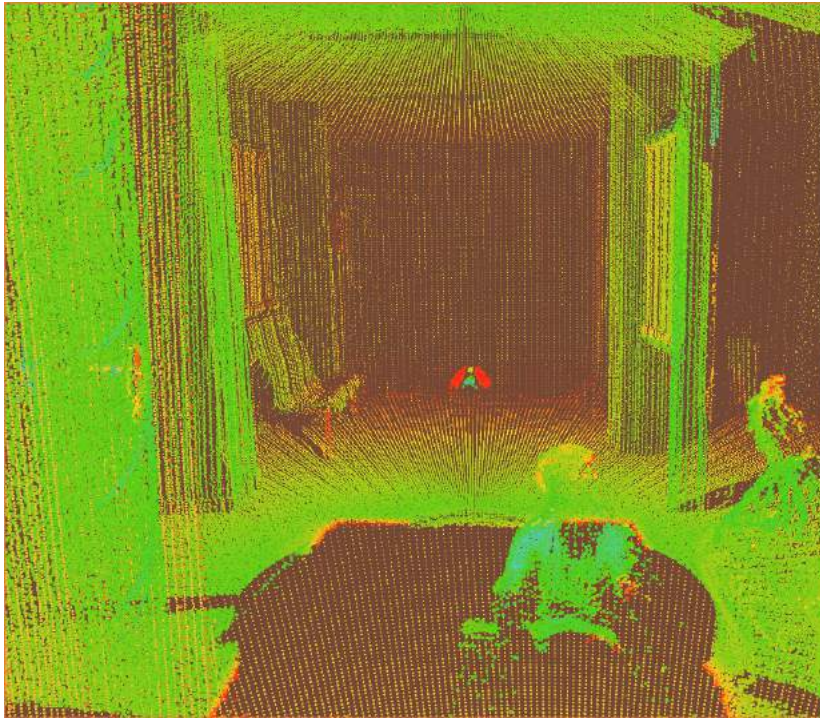


Figura 4.8: Escaneig compost. Punt 4.

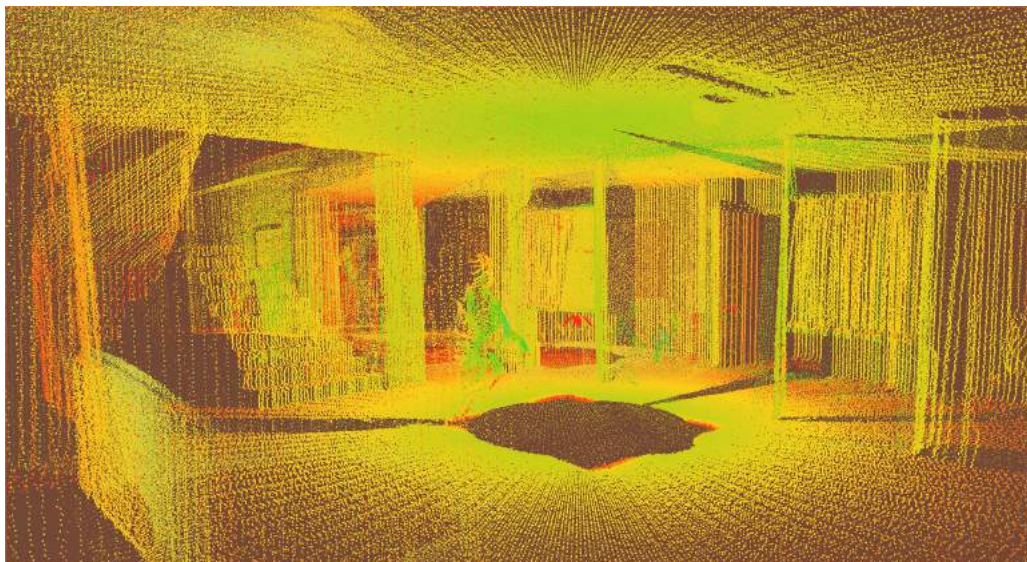


Figura 4.9: Escaneig compost. Punt 5.

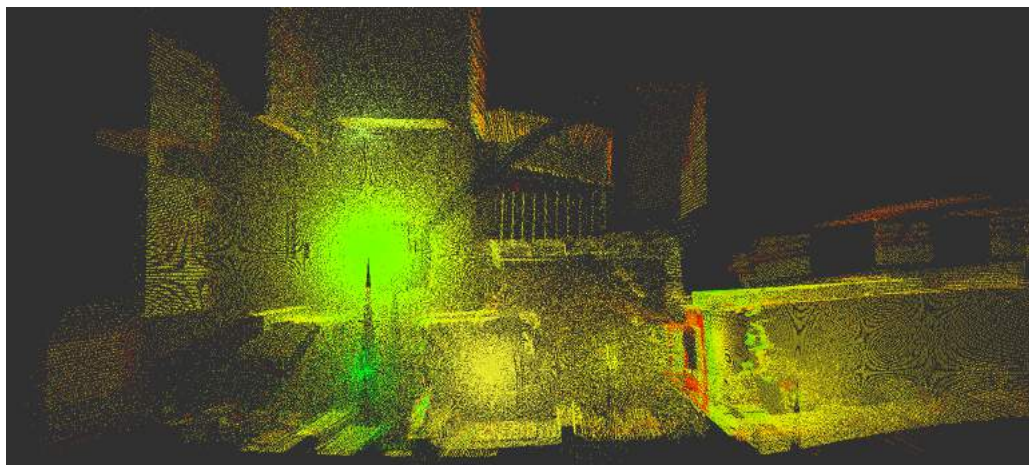


Figura 4.10: Escaneig compost. Planta del conjunt.

4.3 Escaneig exterior

Per a comprovar el comportament de l'escàner a l'exterior s'ha realitzat un escaneig a fora de l'edifici Anselm Turmeda.



Figura 4.11: Preparació de l'escaneig exterior.

Amb el mateix llinar de lectura analògica que a l'interior per al sensor reflectiu, a l'exterior l'escàner no detecta el punt de referència. El resultat de l'escaneig es mostra a les figures 4.12 i 4.13. Es pot comprovar que com més allunyat està un punt de l'escàner, es rep amb menor intensitat el corresponent rebot làser (color més vermell). En els punts més propers el color és entre verd i cel indicant que els rebots s'han rebut amb molta intensitat.

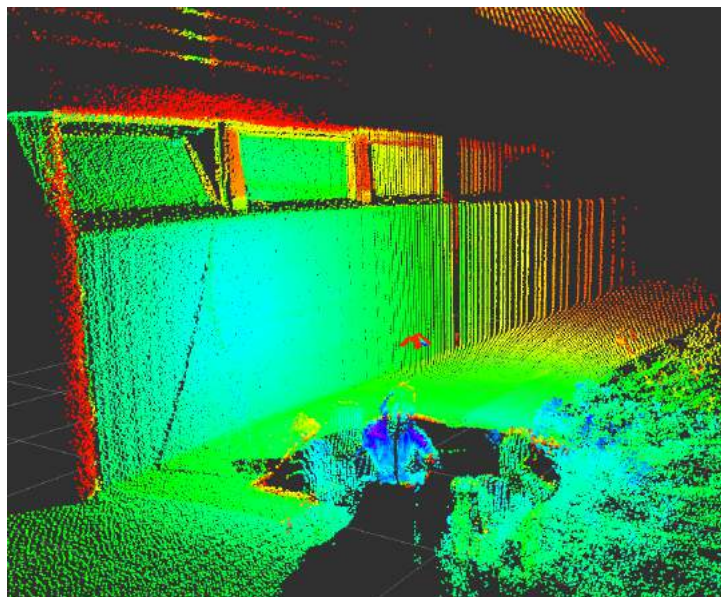


Figura 4.12: Escaneig exterior. Vista 1.

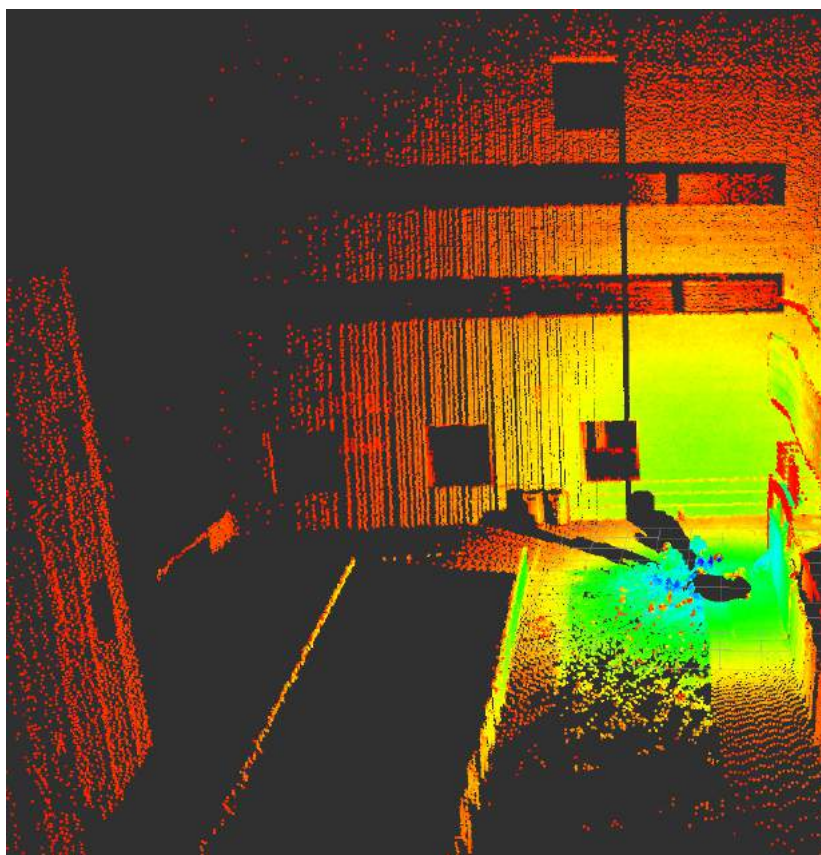


Figura 4.13: Escaneig exterior. Vista 2.

CONCLUSIONS I TREBALLS FUTURS

El disseny elegit per l'escàner 3D ha mostrat ser vàlid per a la reconstrucció d'emplaçaments en ambients poc dinàmics proporcionant uns resultats acurats, aptes per al reconeixement de superfícies a l'espai. En entorns dinàmics s'ha d'arribar a un compromís entre la capacitat d'observar l'entorn a freqüència suficientment alta per a detectar canvis i una resolució adequada. Per això una reconstrucció completa de l'entorn pot no ser viable en entorns dinàmics.

La limitada capacitat de processar dades i de comunicació a través de port sèrie de la placa *Arduino* mitjançant el node *serial_node* dificulta la integració del programa del microcontrolador a la resta de programari en temps d'escaneig. Llevat d'això, la tecnologia utilitzada ha resultat ser adequada per al projecte. Els elements mecànics, el contacte giratori, el coixinet, el sensor òptic i els elements impresos, han complert amb la funció per la qual estaven dissenyats.

Pel que fa al disseny mecànic, l'actual relació de transmissió entre l'eix motor i l'eix d'escaneig és massa alta per a aconseguir resultats precisos. Per a escanejos amb definició mitjana-alta el motor ha de treballar per sota del seu rang de freqüències de funcionament fluid.

S'ha pogut comprovar la utilitat de desenvolupar el programari en un entorn de *ROS*. No només permet utilitzar controladors externs per al LIDAR sinó que el seu enfocament cap als sistemes basats en sensors facilita el desenvolupament del programari proporcionant eines molt específiques.

Per a millorar el comportament de l'escàner i millorar-ne les prestacions següentment es suggereixen aglunes propostes per als treballs futurs.

Es poden portar a terme una bateria de proves que permetin ajustar de manera més acurada els paràmetres emprats al determinar la posició del LIDAR a cada instant de temps com són la relació entre l'origen real dels escaneigs i el punt de referència de l'escàner.

Una opció a estudiar, és la viabilitat de, en lloc d'utilitzar el controlador estàndard, construir un nou controlador per al LIDAR més integrat en el codi propi de l'escàner, aconseguint coordinar la rotació d'aquest amb els escanejos planaris. Altres possibles

avantatges d'un controlador propi com el tractament de les lectures 2D en el mateix controlador s'haurien d'estudiar.

El processament dels escanejos 3D és, en general, una tasca que s'ha de gestionar des de l'aplicació que l'utilitzi. Tot i això, pot ser d'interès aplicar filtres als escanejos 2D del LIDAR abans de convertir-los en un núvol de punts. D'aquesta manera es podrien utilitzar eines de tractament de missatges *LaserScan* com les que proporciona el paquet *laser_filters*.

La possibilitat de fer escanejos selectius pot ser d'interès per a un escàner 3D. Es podria dissenyar un controlador que permetés diferents resolucions per a diferents rangs d'escaneig. Fent canvis en la freqüència de funcionament i el sentit de gir del motor pas a pas es pot aconseguir més versatilitat en l'escaneig, permetent una major adaptació a les necessitats de l'usuari de l'escàner indicades a través de paràmetres. Una altra funcionalitat que podria ser d'interès és la reconfiguració dinàmica dels paràmetres del controlador que es poden portar a terme emprant eines com el paquet *dynamic_reconfigure*.

Per a millorar la feblesa de comunicació entre el microcontrolador i el sistema *ROS* i augmentar l'autonomia de l'escàner, es pot estudiar la possibilitat de portar a terme tant l'execució de *ROS* com la gestió dels controladors electrònics en un únic sistema encastat, aprofitant eines de *ROS* per a sistemes encastats.

Per a reduir les oscil·lacions de la part mòbil es podria disminuir la relació de transmissió de manera que pel mateix rang de velocitats d'escaneig, el motor giri a major freqüència. D'aquesta manera s'aconseguiria també un major parell efectiu a l'eix de rotació del suport.



CODI FONT

A.1 Node de ROS *lidar_scan_node*

```
1  /*****
2  *
3  * Node de control del funcionament de l'escanner 3D.
4  *
5  *****/
6
7  #include <math.h>
8  #include "ros/ros.h"
9  #include "tf/transform_broadcaster.h"
10
11  // Missatges
12  #include "std_msgs/Empty.h"
13  #include "sensor_msgs/LaserScan.h"
14  #include "lidar_scan/CodiGir.h"
15
16  /* Constants de la geometria de l'escanner i relacionades
17   * amb el periode del motor */
18  #define coorx 0.0865
19  #define coory 0.08
20  #define coorz 0.0
21  #define angleInicial 0.0
22  #define periodeDefecte 1500
23  #define maxPeriode 25000
24  #define minPeriode 1250
25
26  using namespace std;
27
28  // Classe encarregada del control de l'escaneig
29  class lidarScan{
30  public:
```

```
31
32 // Inicialitzacio dels atributs
33 lidarScan(int periode) {
34     gir_pub = n_.advertise<lidar_scan::CodiGir>("/girConfig", 10, true);
35
36     angle = angleInicial;
37     pos_coneguda = false;
38     periodeActual = periode;
39     incAngle=125*M_PI/4/periode;
40
41     gir_msg.per = periode;
42     gir_msg.header.stamp = ros::Time::now();
43     gir_pub.publish(gir_msg);
44 }
45
46 // Laser a la posicio inicial
47 void casa(const std_msgs::Empty& msg){
48     if(!pos_coneguda){
49         angle = angleInicial;
50         pos_coneguda=true;
51     }
52 }
53
54
55 // Publicar la transformada entre la base de l'escanner i el sensor
56 optic
57 void publicarTf(const sensor_msgs::LaserScan& scan){
58     if (pos_coneguda){
59         ros::Time temps = scan.header.stamp;
60         angle = angle + incAngle;
61
62         tf::Transform tr;
63         tf::Quaternion q;
64         q.setRPY(-angle*2, -M_PI/2, 0);
65         tr.setRotation(q);
66         tr.setOrigin( tf::Vector3(coorx, coory, coorz) );
67
68         static tf::TransformBroadcaster emisor;
69         emisor.sendTransform(tf::StampedTransform(tr, temps, "base_laser", "hokuyo"));
70     }
71 }
72 private:
73     ros::NodeHandle n_;
74     ros::Publisher gir_pub;
75     lidar_scan::CodiGir gir_msg;
76     double angle, incAngle;
77     bool pos_coneguda;
78     int periodeActual;
79 };
80
81 int main(int argc, char **argv){
```

```

82     int periode;
83
84     ros::init(argc, argv, "lidar_scan_node");
85     ros::NodeHandle n;
86     ros::NodeHandle nh("~");
87
88     if (nh.getParam("periode", periode)){
89         if(periode < minPeriode){
90             periode = minPeriode;
91             nh.setParam("periode", minPeriode);
92             ROS_INFO("El periode de stepping ha de ser, com a minim: %u.
93             Aquest ha estat el que s'ha establert.", minPeriode);
94         }else if(periode > maxPeriode){
95             periode = maxPeriode;
96             nh.setParam("periode", maxPeriode);
97             ROS_INFO("El periode de stepping ha de ser, com a maxim: %u.
98             Aquest ha estat el que s'ha establert.", maxPeriode);
99         }else{
100             ROS_INFO("S'ha inicilitzat l'escaneig al periode: %u", periode);
101         }
102     }else {
103         periode = periodeDefecte;
104         ROS_INFO("S'ha inicilitzat l'escaneig al periode per defecte: %u",
105         periode);
106     }
107
108     lidarScan escanner = lidarScan(periode);
109
110     ros::Subscriber flanc_sub = n.subscribe("/home", 5, &lidarScan::casa, &
111     escanner);
112     ros::Subscriber laser_sub = n.subscribe("/first", 5, &lidarScan::
113     publicarTf, &escanner);
114
115     while(ros::ok()){
116         ros::spinOnce();
117     }
118     return 0;
119 }

```

A.2 Node *pc_snapshotter*

```

1  /*****
2  *
3  * Node per a publicar niguls de putns periodicament
4  *
5  *****/
6
7  #include <cstdio>
8  #include <ros/ros.h>
9

```

```
10 // Serveis
11 #include "laser_assembler/AssembleScans.h"
12
13 // Missatges
14 #include "std_msgs/Empty.h"
15 #include "sensor_msgs/PointCloud.h"
16 #include <lidar_scan/CodiGir.h>
17
18
19 using namespace std;
20
21 class PCSnapshotter{
22
23 public:
24
25     PCSnapshotter(){
26         ROS_INFO("Objecte creat");
27         pub_ = n_.advertise<sensor_msgs::PointCloud> ("pc_lidarScan",
28             1);
29         client_ = n_.serviceClient<laser_assembler::AssembleScans>("
30             assemble_scans");
31         primera_vegada = true;
32         compte=0;
33     }
34
35     void periodeCallback(const lidar_scan::CodiGir periodePas){
36         ROS_INFO("Periode capturat");
37         periodeScan = periodePas.per/1250;
38     }
39
40     void iniciarEscaneig(const std_msgs::Empty& msg){
41         timer_ = n_.createTimer(ros::Duration(periodeScan,0), &
42             PCSnapshotter::timerCallback, this);
43         ROS_INFO("Timer activat amb temps %u", periodeScan);
44     }
45
46     void timerCallback(const ros::TimerEvent& e){
47         if (primera_vegada){
48             ROS_INFO("primera_vegada");
49             primera_vegada = false;
50             return;
51         }
52
53         laser_assembler::AssembleScans srv;
54         srv.request.begin = e.last_real;
55         srv.request.end = e.current_real;
```

```

54     if (client_.call(srv)){
55         compte++;
56         ROS_INFO("S'ha publicat l'escaneig %u amb %u punts", compte
57             , (uint32_t)(srv.response.cloud.points.size())) ;
58         pub_.publish(srv.response.cloud);
59     }else{
60         ROS_ERROR("S'ha produït un error en la crida de l'
61             acumulador\n") ;
62     }
63 }
64
65 private:
66     ros::NodeHandle n_;
67     ros::Publisher pub_;
68     ros::ServiceClient client_;
69     ros::Timer timer_;
70     int compte, periodeScan;
71     bool primera_vegada;
72 } ;
73
74 int main(int argc, char **argv){
75     ros::init(argc, argv, "pc_snapshotter");
76     ros::NodeHandle n;
77
78     ROS_INFO("Esperant servei");
79     ros::service::waitForService("build_cloud");
80     ROS_INFO("Escaneig preparat");
81
82     PCSnapshotter snapshotter;
83     ros::Subscriber config_sub = n.subscribe("/girConfig", 5, &
84         PCSnapshotter::periodeCallback, &snapshotter);
85     ros::Subscriber flanc_sub = n.subscribe("/home", 5, &
86         PCSnapshotter::iniciarEscaneig, &snapshotter);
87
88     while(ros::ok()){
89         ros::spinOnce();
90     }
91     return 0;
92 }

```

A.3 Programa *hokuyo_arduino*

S'ha de notar que en rebre un missatge de tipus *CodiGir* al t pic *gir_config* amb un valor de la variable *per* igual a zero, es deixen d'ordenar passos al motor.

```

1  /*****
2  *
3  * Programa per a controlar els dispositius f sics

```

```
4 * de l'escanner 3D.
5 *
6 *****/
7
8
9 #include <ros.h>
10 #include <ros/time.h>
11 #include <lidar_scan/CodiGir.h>
12 #include <TimerOne.h>
13
14 // Missatges
15 #include <std_msgs/UInt32.h>
16 #include <std_msgs/Empty.h>
17
18 // Durada d'un pols de la sortida 'step' per a efectuar un pas.
19 #define duradaPols 2
20
21 //Representacio simbolica dels pins conectats al driver i al sensor
    optic (obp).
22 #define entradaOpb A5
23 const int p_dir = 13;
24 const int p_bot = 12;
25 const int p_sleep = 11;
26 const int p_reset = 10;
27 const int p_m2 = 9;
28 const int p_m1 = 8;
29 const int p_m0 = 7;
30 const int p_enable = 6;
31 const int p_fault = 5;
32 const int p_obp = 4;
33
34 const int limlectura = 24;
35 int lecturaOpb;
36 bool casa;
37
38 // Dur a terme un pas
39 void performStep(){
40     digitalWrite(p_bot, HIGH);
41     delayMicroseconds(duradaPols);
42     digitalWrite(p_bot, LOW);
43 }
44
45 // Configurar la velocitat d'escaneig i inicialitzacio del
    temporitzador
46 void configurar(const lidar_scan::CodiGir& missatge){
47     Timer1.stop();
48     Timer1.detachInterrupt();
```



```
49
50     if(!casa){
51         homing();
52     }
53     if(missatge.per != 0){
54         Timer1.initialize(missatge.per);
55         Timer1.attachInterrupt(performStep);
56         casa = false;
57     }
58 }
59
60 // Declaracio dels objectes de ROS a utilitzar
61 std_msgs::Empty flanc_msg;
62 ros::NodeHandle nh;
63 ros::Subscriber<lidar_scan::CodiGir> config_sub("girConfig", &
    configurar );
64 ros::Publisher flanc("/home", &flanc_msg);
65
66 // Recerca del punt de referencia
67 void homing(){
68     int i;
69     lecturaOpb=analogRead(entradaOpb);
70     while(lecturaOpb > limlectura && i < 40){
71         performStep();
72         lecturaOpb=analogRead(entradaOpb);
73         delay(5);
74         if(lecturaOpb > limlectura){
75             i++;
76         }
77     }
78     while(lecturaOpb < limlectura){
79         performStep();
80         lecturaOpb=analogRead(entradaOpb);
81         delay(2);
82     }
83
84     casa = true;
85     flanc.publish(&flanc_msg);
86 }
87
88 // Inicialitzacio dels d'entrades i sortides i del node de ROS
89 void setup(){
90     pinMode(p_dir, OUTPUT);
91     pinMode(p_bot, OUTPUT);
92     pinMode(p_sleep, OUTPUT);
93     pinMode(p_reset, OUTPUT);
94     pinMode(p_m2, OUTPUT);
```

```
95     pinMode(p_m1, OUTPUT);
96     pinMode(p_m0, OUTPUT);
97     pinMode(p_enable, OUTPUT);
98     pinMode(p_fault, INPUT);
99     pinMode(p_obp, OUTPUT);
100
101     digitalWrite(p_dir, LOW);
102     digitalWrite(p_sleep, HIGH);
103     digitalWrite(p_reset, HIGH);
104     digitalWrite(p_m2, HIGH);
105     digitalWrite(p_m1, HIGH);
106     digitalWrite(p_m0, HIGH);
107     digitalWrite(p_enable, LOW);
108     digitalWrite(p_obp, HIGH);
109
110     nh.initNode();
111     nh.subscribe(config_sub);
112     nh.advertise(flanc);
113 }
114
115 // Bucle infinit a l'espera d'interrupcions
116 void loop(){
117     nh.spinOnce();
118 }
```

A.4 Arxiu *lidarscan.launch*

La configuració d'aquest arxiu depèn de si es vol dur a terme la publicació dels nignuls de punts en el moment de l'escaneig o no. En cas de publicacions *online* es fa servir el següent codi:

```
1 <launch>
2   <node type="urg_node" pkg="urg_node" name="driver_hokuyo">
3     <param name="ip_address" type="string" value="192.168.0.10"/>
4     <param name="frame_id" type="string" value="hokuyo"/>
5     <param name="angle_min" type="double" value="0"/>
6   </node>
7   <node pkg="rosserial_python" type="serial_node.py" args="/dev/
ttyUSB0" name="serial_node" respawn="true"/>
8   <node pkg="rviz" type="rviz" name="rviz"/>
9   <node type="laser_scan_assembler" pkg="laser_assembler" name="
assembler_most_intense">
10     <remap from="scan" to="most_intense"/>
11     <param name="max_scans" type="int" value="1200" />
12     <param name="fixed_frame" type="string" value="base_laser" />
13   </node>
```

```
14   <node pkg="lidar_scan" type="pc_snapshotter" name="
    pc_snapshotter"/>
15 </launch>
```

Per a escanejos *offline*, en lloc d'executar-se el node *pc_snapshotter* i l'acumulador *laser_scan_assembler* es registrarien els tópics *tf*, *home*, *gir_config* i el tópic de missatges *LaserScan* que es vulgui enregistrar. Així s'haurien de substituir les línies de la 9 a la 14 per una com la següent:

```
<node pkg="roscpp" type="roscpp" name="enregistrador" args="record
/tf /home /gir_config /most_intense /first /last"/>
```

El node *lidar_scan_node* no està inclòs en aquest arxiu per a poder canviar el seu paràmetre *periode* d'una manera més dinàmica. Per a incloure-lo s'hauria d'afegir una línia com la següent:

```
<node pkg="lidar_scan" type="lidar_scan_node" name="lidarscan"
periode="20000"/>
```


CODI PER A L'OBTENCIÓ DE L'ESCANEIG COMPOST

B.1 Node *compondre*

```

1  #include <ros/ros.h>
2  #include "sensor_msgs/PointCloud.h"
3
4  ros::Duration offset(2.0);
5
6  ros::Publisher pub1;
7  ros::Publisher pub2;
8  ros::Publisher pub3;
9  ros::Publisher pub4;
10 ros::Publisher pub5;
11 ros::Publisher pub6;
12
13 sensor_msgs::PointCloud nou1, nou2, nou3, nou4, nou5, nou6;
14 bool b1, b2, b3, b4, b5, b6, espera;
15 ros::Time temps;
16
17
18 void callback1(const sensor_msgs::PointCloud& cloud){
19     if (!b1){
20         ROS_INFO("Rebut pc: 1");
21         nou1=cloud;
22         b1=true;
23     }
24 }
```

```
25
26 void callback2(const sensor_msgs::PointCloud& cloud){
27     if (!b2){
28         ROS_INFO("Rebut pc: 2");
29         nou2=cloud;
30         b2=true;
31     }
32 }
33
34 void callback3(const sensor_msgs::PointCloud& cloud){
35     if (!b3){
36         ROS_INFO("Rebut pc: 3");
37         nou3=cloud;
38         b3=true;
39     }
40 }
41 void callback4(const sensor_msgs::PointCloud& cloud){
42     if (!b4){
43         ROS_INFO("Rebut pc: 4");
44         nou4=cloud;
45         b4=true;
46     }
47 }
48
49 void callback5(const sensor_msgs::PointCloud& cloud){
50     if (!b5){
51         ROS_INFO("Rebut pc: 5");
52         nou5=cloud;
53         b5=true;
54     }
55 }
56
57 void callback6(const sensor_msgs::PointCloud& cloud){
58     if (!b6){
59         ROS_INFO("Rebut pc: 6");
60         nou6=cloud;
61         b6=true;
62     }
63 }
64
65 int main(int argc, char **argv){
66     ros::init(argc, argv, "compondre");
67     ros::NodeHandle n;
68
69     b1=false;
70     b2=false;
71     b3=false;
```

```
72     b4=false;
73     b5=false;
74     b6=false;
75
76     espera=true;
77
78     ros::Subscriber in1_sub = n.subscribe("/pc_lidarScan_i1", 5,
79     callback1);
80     ros::Subscriber in2_sub = n.subscribe("/pc_lidarScan_i2", 5,
81     callback2);
82     ros::Subscriber in3_sub = n.subscribe("/pc_lidarScan_i3", 5,
83     callback3);
84     ros::Subscriber in4_sub = n.subscribe("/pc_lidarScan_i4", 5,
85     callback4);
86     ros::Subscriber in5_sub = n.subscribe("/pc_lidarScan_i5", 5,
87     callback5);
88     ros::Subscriber in6_sub = n.subscribe("/pc_lidarScan_i6", 5,
89     callback6);
90
91     pub1 = n.advertise<sensor_msgs::PointCloud> ("pc1", 1);
92     pub2 = n.advertise<sensor_msgs::PointCloud> ("pc2", 1);
93     pub3 = n.advertise<sensor_msgs::PointCloud> ("pc3", 1);
94     pub4 = n.advertise<sensor_msgs::PointCloud> ("pc4", 1);
95     pub5 = n.advertise<sensor_msgs::PointCloud> ("pc5", 1);
96     pub6 = n.advertise<sensor_msgs::PointCloud> ("pc6", 1);
97
98     while(ros::ok()){
99         if (b1 && b2 && b3 && b4 && b5 && b6 && espera){
100             temps = ros::Time::now();
101             nou1.header.stamp=temps;
102             nou2.header.stamp=temps;
103             nou3.header.stamp=temps;
104             nou4.header.stamp=temps;
105             nou5.header.stamp=temps;
106             nou6.header.stamp=temps;
107
108             nou1.header.frame_id="punt1";
109             nou2.header.frame_id="punt2";
110             nou3.header.frame_id="punt3";
111             nou4.header.frame_id="punt4";
112             nou5.header.frame_id="punt5";
113             nou6.header.frame_id="punt6";
114
115             pub1.publish(nou1);
116             pub2.publish(nou2);
```

```
113         pub3.publish(nou3);
114         pub4.publish(nou4);
115         pub5.publish(nou5);
116         pub6.publish(nou6);
117
118         ROS_INFO("S'han publicat tots els níguls");
119
120         espera=false;
121     }
122     ros::spinOnce();
123 }
124 return 0;
125 }
```

B.2 Arxiu *reconstruccio.launch*

Es pot observar com el node *compondre* s'ha creat al paquet *lidar_scan* de manera provisional.

```
1 <launch>
2   <node pkg="roslab" type="play" name="interior1" output="screen"
3     args="/home/jordi/bagfiles/compost/i1.bag"/>
4   <node pkg="roslab" type="play" name="interior2" output="screen"
5     args="/home/jordi/bagfiles/compost/i2.bag"/>
6   <node pkg="roslab" type="play" name="interior3" output="screen"
7     args="/home/jordi/bagfiles/compost/i3.bag"/>
8   <node pkg="roslab" type="play" name="interior4" output="screen"
9     args="/home/jordi/bagfiles/compost/i4.bag"/>
10  <node pkg="roslab" type="play" name="interior5" output="screen"
11    args="/home/jordi/bagfiles/compost/i5.bag"/>
12  <node pkg="roslab" type="play" name="interior6" output="screen"
13    args="--clock /home/jordi/bagfiles/compost/i6.bag"/>
14
15  <node pkg="lidar_scan" type="compondre" name="reconstruccio"/>
16
17  <node pkg="tf" type="static_transform_publisher" name="
18    static_tf1" args="0 0 0 3.05 0 0 ref punt1 10" />
19  <node pkg="tf" type="static_transform_publisher" name="
20    static_tf2" args="0 2.9 0 1.62 0.03 0 ref punt2 10" />
21  <node pkg="tf" type="static_transform_publisher" name="
22    static_tf3" args="0 5.5 0 1.66 0.03 0 ref punt3 10" />
23  <node pkg="tf" type="static_transform_publisher" name="
24    static_tf4" args="0.147 8.29 -0.1 1.6232 0.06 0 ref punt4 10" />
25  <node pkg="tf" type="static_transform_publisher" name="
26    static_tf5" args="2.71 8.1 -0.1 1.57 0 0.01 ref punt5 10" />
27  <node pkg="tf" type="static_transform_publisher" name="
28    static_tf6" args="5.42 8 -0.15 1.57 0 0 ref punt6 10" />
29 </launch>
```








FULLS DE CARACTERÍSTIQUES

C.1 Hokuyo UTM-30LX-EW

Date: 2012.12.3

Scanning Laser Range Finder UTM-30LX-EW Specification

 × 1		Correction of External Dimension representation		3	2012.12.3	KAMON	RS-0159		
 × 1		The description of the Multiecho function was added.		6,7	2012.11.5	KAMON	RS-0147		
 × 1		Electric cable connection is modified		4	2012.3.28	TAMAKI	RS-0052		
 × 2		Error code table is added.		5,6	2011.11.25	TAMAKI	RS-0006		
Symbol				Amendment Details		Amendment	Date	Amended by	Number
Approved by		Checked by		Drawn by		Designed by		Title	<u>UTM-30LX-EW</u> Specification
KAMITANI		UTSUGI		KAMON		KAMON			
						Drawing No.		C-42-3785	
								1/7	

1. Introduction

1.1 Operation principles

The UTM-30LX-EW uses a laser source ($\lambda=905\text{nm}$) to scan a 270° semicircular field (Figure 1). It measures the distance for each angular step to objects in its range. The measurement data along with its angular step are transmitted via a communication channel. The laser safety is class 1.

2. Diagram of Scanned Area

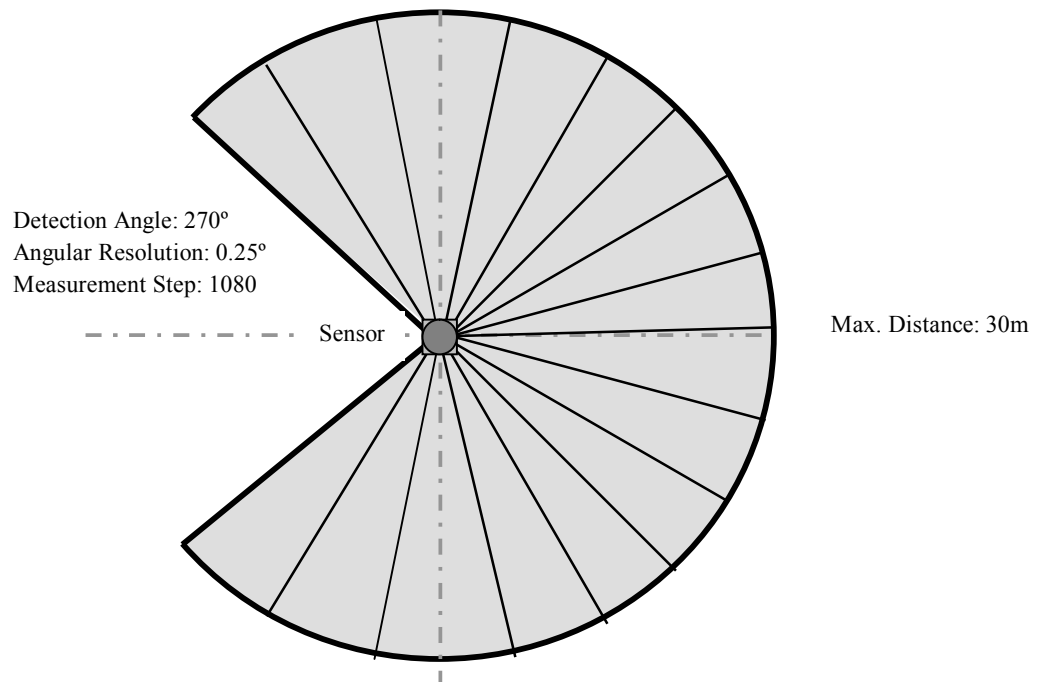



Figure 1

3. Important Notes

- This sensor is not a safety device/tool.
- This sensor is not for use in human detection.
- Hokuyo products are not developed and manufactured for use in weapons, equipment, or related technologies intended for destroying human lives or creating mass destruction. If such possibilities or usages are revealed, the sales of Hokuyo products to those customers might be halted by the laws of Japan such as Foreign Exchange Law, Foreign Trade Law or Export Trade Control Order. In addition, we will export Hokuyo products for the purpose of maintaining the global peace and security in accordance with the above laws of Japan
- Read specifications carefully before use.

Title	UTM-30LX-EW Specification	Drawing No	C-42-3785	2/7
-------	---------------------------	------------	-----------	-----

4. Specifications

Product Name	Scanning Laser Range Finder
Model	UTM-30LX-EW
Light Source	Laser Semiconductor $\lambda = 905\text{nm}$ Laser Class 1
Supply Voltage	12VDC $\pm 10\%$
Supply Current	Max: 1A, Normal : 0.7A
Power Consumption	Less than 8W
Detection Range and Detection Object	Guaranteed Range: 0.1 ~ 30m (White Kent Sheet) * ² Maximum Range : 0.1 ~ 60m Minimum detectable width at 10m : 130mm (Vary with distance)
Accuracy	0.1 – 10m : $\pm 30\text{mm}$, 10 – 30m : $\pm 50\text{mm}$ (White Kent Sheet) * ² Under 3000lx : White Kent Sheet: $\pm 30\text{mm}^{*1}$ (0.1m to 10m) Under 100000lx : White Kent Sheet: $\pm 50\text{mm}^{*1}$ (0.1m to 10m)
Measurement Resolution and Repeated Accuracy	1mm 0.1 – 10m : $\sigma < 10\text{mm}$, 10 – 30m : $\sigma < 30\text{mm}$ (White Kent Sheet) * ² Under 3000lx : $\sigma = 10\text{mm}^{*1}$ (White Kent Sheet up to 10m) Under 100000lx : $\sigma = 30\text{mm}^{*1}$ (White Kent Sheet up to 10m)
Scan Angle	270°
Angular Resolution	0.25° (360°/1440)
Scan Speed	25ms (Motor speed : 2400rpm)
Interface	Ethernet 100BASE-TX(Auto-negotiation)
Output	Synchronous Output 1- Point
LED Display	Green: Power supply. Red: Normal Operation (Continuous), Malfunction (Blink)
Ambient Condition (Temperature, Humidity)	-10°C ~ +50°C Less than 85%RH (Without Dew, Frost)
Storage Temperature	-25~75°C
Environmental Effect	Measured distance will be shorter than the actual distance under rain, snow and direct sunlight* ² .
Vibration Resistance	10 ~ 55Hz Double amplitude 1.5mm in each X, Y, Z axis for 2hrs. 55 ~ 200Hz 98m/s ² sweep of 2min in each X, Y, Z axis for 1hrs.
Impact Resistance	196m/s ² In each X, Y, Z axis 10 times.
Protective Structure	Optics: IP67 (Except Ethernet connector)
Insulation Resistance	10M Ω DC500V Megger
Weight	210g (Without cable)
Case	Polycarbonate
External Dimension (W×D×H)	62mm×62mm×87.5mm  MC-40-3240

*¹ Under Standard Test Condition (Accuracy can not be guaranteed under direct sunlight.)

*² Indoor environment with less than 1000Lx.

Please perform necessary tests with the actual device in the working environment.

Use data filtering techniques to reduce the effect of water droplets when detecting objects under the rain.

5. Quality Reference Value

Vibration resistance during operation	10~150Hz 19.6m/s ² Sweep of 2min in each X,Y,Z axis for 30min
Impact resistance during operation	49m/s ² X, Y,Z axis 10 times
Angular Speed	2 π /s (1Hz)
Angular Acceleration	$\pi/2\text{rad/s}^2$
Life-span	5 Years (Varies with operating conditions)
Noise Level	Less than 25dB at 300 mm
Certification	FDA Approval (21 CFR part 1040.10 and 1040.11)

Title	UTM-30LX-EW Specification	Drawing No	C-42-3785	3/7
-------	---------------------------	------------	-----------	-----

6. Interface

6.1 Robot Cable 4 Pin

Color	Function
Brown	+12 V
Blue	0 V
Green	Synchronous Output

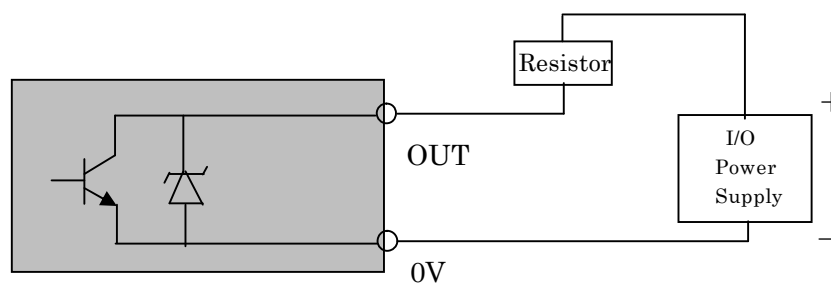
Note: 0 V of the power supply (Blue) and COM Output (0V) (White) are internally connected.

6.2 Ethernet Cable

RJ-45 plug is attached to the cable. (Length: 300mm)

This sensor is compatible with SCIP2.2 communication protocol standard.

6.3 Output Circuit Diagram



Rated power: 30V, 30mA (or less)

Note: Rated resistor should be used for the output.

Figure 2

Title	UTM-30LX-EW Specification	Drawing No	C-42-3785	4/7
-------	---------------------------	------------	-----------	-----

7. Control Signal

Synchronous Output (UTM-30LX)

1 pulse is approximately 1 ms. Output signal Synchronization timing chart is shown below (Figure 3).

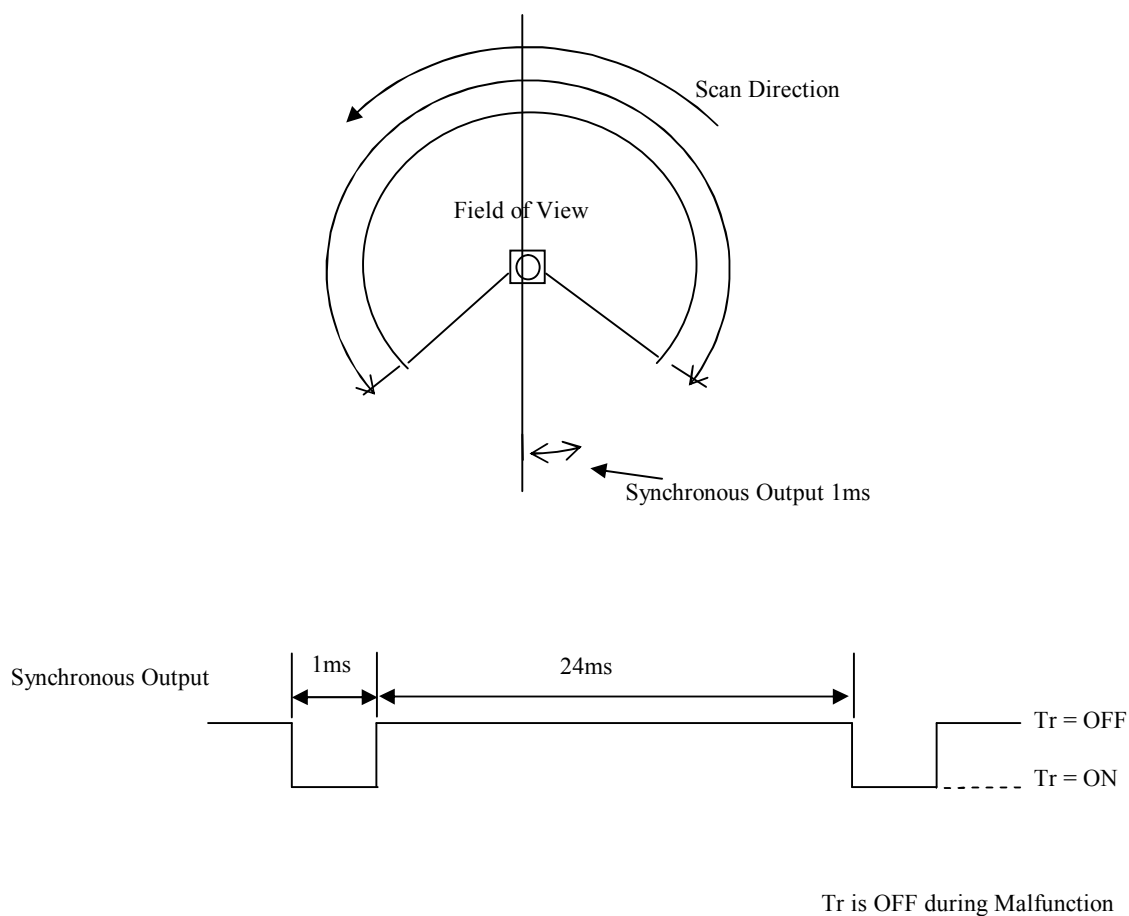


Figure 3

Title	UTM-30LX-EW Specification	Drawing No	C-42-3785	5/7
-------	---------------------------	------------	-----------	-----

8. Malfunction Output:

1. Laser malfunction: When the laser does not emit or exceeds safety class 1.
2. Motor malfunction: When the rotation speed differs from the default value (> 25 ms).

Synchronous/Warning signal will be turned OFF when these malfunctions are detected. The motor and laser will also stop. The details of error can be obtained via communication.

[Error code]

The cause of an error can be acquired from a "STAT" line of the "II" command response of the SCIP communications protocol. An error code and a solution acquired from a "STAT" line are as follows.

ID	Message	Meaning	Solution
000	no error.	Normal	No action is required
050	internal chip access failed.	Abnormal sensor processing system	Sensor has failed and needs to be repaired
100	Internal chip access failed.	Abnormal sensor processing system	
150	internal chip access failed.	Abnormal sensor processing system	
151	internal chip initialize failed.	Sensor processing system failed to initialize	
200	encoder error.	Encoder error	
250	motor startup failed.	Abnormality of the motor	Reduce the vibration and noise to the sensor
251	motor rotation error.	Motor rotation is not stable	
300	laser too high.	Abnormality of the laser light	Reduce the ambient light and noise to the sensor
301	laser too low.	Abnormality of the laser light	
302	laser no echo	Abnormality of the laser light	
303	measurement error.	The control process for measuring distance failed	Reduce the vibration and ambient light and noise to the sensor

[The meaning of the distance value]

The meaning of "x" distance value of each step is as follows.

Distance value "x"	Meaning
$x < 23$	Measurement error. The distance cannot be measured due to light interference or noise.
$23 \leq x < 60000$	Valid distance value [mm]
$60000 \leq x$	Object does not exist or the object has low reflectivity.

9. Multiecho Function

The sensor measures up to three echoes of reflection for each step (direction). Distance and intensity values of every echoes are obtained

Multiple echoes are produced by reflection on surface of transparent objects, reflection on objects' boundary and reflection from small particles such as rain drops, mist, dusts and fog.

This sensor feature of getting distance and intensity values of multiple reflections at the same direction is called Multiecho Function.

※ Two closely positioned objects or low reflectance objects may not produce multiple echoes, so that they are not detectable as separate ones.

Title	UTM-30LX-EW Specification	Drawing No	C-42-3785	6/7
-------	---------------------------	------------	-----------	-----

9. Ethernet Settings

① Initial value

IP address: 192.168.0.10

Port number: 10940

② IP initialization

Remove the rubber cap located at the side of the bottom cover of the sensor. Press and hold the switch inside this hole for more than two seconds in order to start the IP initialization process. Release the switch after the LED flashes in orange color. This indicates the restart of the sensor. Finally, please insert the rubber cap to its original position.

10. Cautions

The heat is generated as the internal circuit of the sensor runs at a very high speed. The generated heat is concentrated at the bottom of the sensor. Please mount a heat sink or any appropriate component to release the heat. An aluminum plate (200mm x 200mm x 2mm) is recommended as the heat sink.

Mutual Interference could occur when two or more identical sensors are mounted at the same detection plane. This is because the sensor could not identify the origin of the received laser pulses. It causes measurement error for one or two steps. Performing data filtering could overcome this problem.

Title	UTM-30LX-EW Specification	Drawing No	C-42-3785	7/7
-------	---------------------------	---------------	-----------	-----



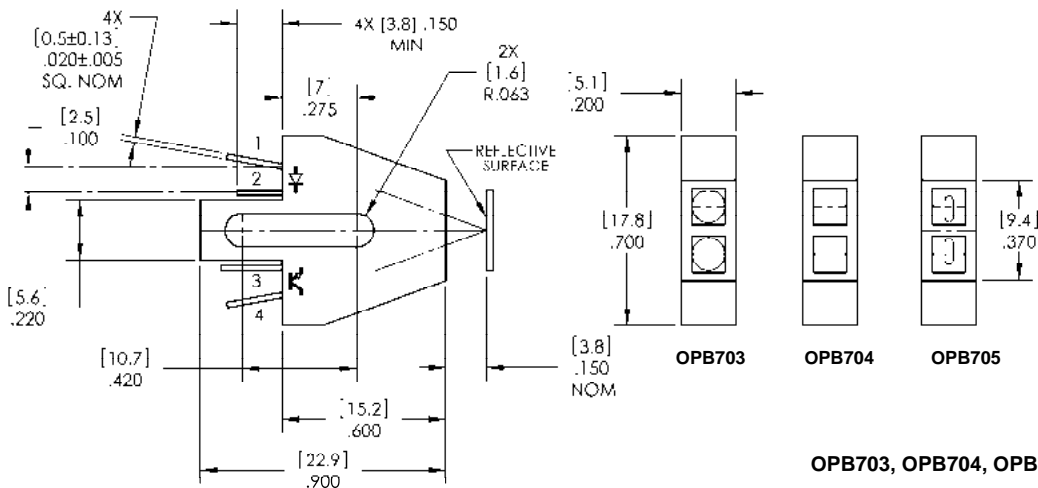
C.2 Sensor reflectiu OPB704

Reflective Object Sensor

OPB703 through OPB705, OPB703WZ through OPB705WZ,
OPB70AWZ through OPB70HWZ

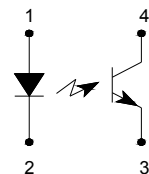


OPB703, OPB704, OPB705

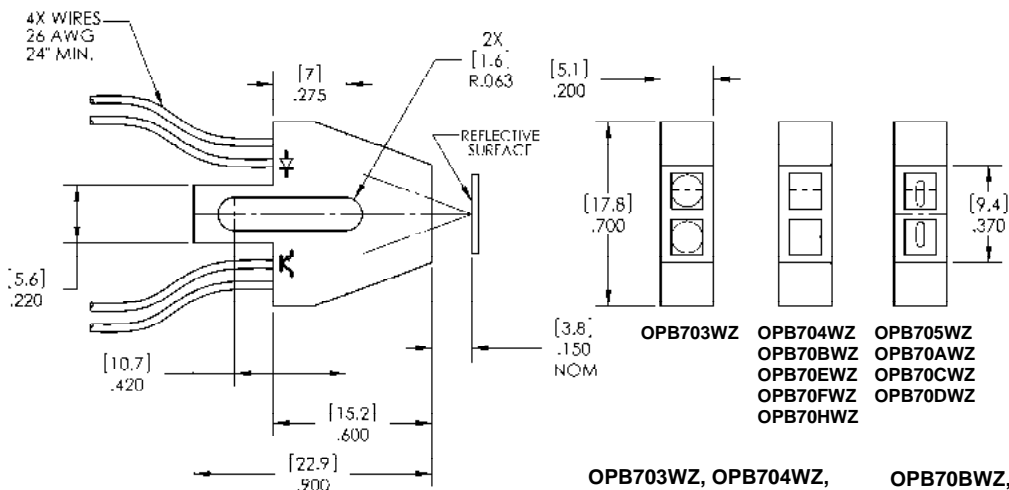


DIMENSIONS ARE IN INCHES (MM)
TOLERANCES ARE .010 UNLESS OTHERWISE SPECIFIED.

OPB703, OPB704, OPB705

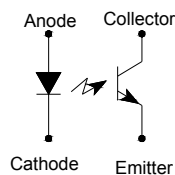


OPB703WZ, OPB704WZ, OPB705WZ, OPB70AWZ, OPB70BWZ, OPB70CWZ, OPB70DWZ

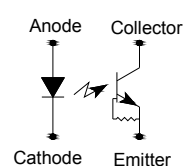


DIMENSIONS ARE IN INCHES (MM)
TOLERANCES ARE .010 UNLESS OTHERWISE SPECIFIED.

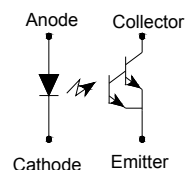
OPB703WZ, OPB704WZ, OPB705WZ, OPB70DWZ OPB70HWZ



OPB70BWZ, OPB70CWZ OPB70EWZ



OPB70AWZ



WIRE COLOR	FUNCTION
ORANGE	ANODE
GREEN	CATHODE
BLUE	EMITTER
WHITE	COLLECTOR

OPTEK reserves the right to make changes at any time in order to improve design and to supply the best product possible.

Reflective Object Sensor

OPB703 through OPB705, OPB703WZ through OPB705WZ,
OPB70AWZ through OPB70HWZ



Absolute Maximum Ratings ($T_A=25^\circ\text{C}$ unless otherwise noted)

Storage Temperature Range	-40°C to $+80^\circ\text{C}$
Lead Soldering Temperature [1/16 inch (1.6 mm) from the case for 5 sec. with soldering iron]	$240^\circ\text{C}^{(1)}$

Input Diode

Forward DC Current	40 mA
Reverse DC Voltage	2 V
Power Dissipation	100 mW ⁽²⁾

Output Photodetector

Collector-Emitter Voltage Phototransistor Photodarlington	30 V 15 V
Emitter-Collector Voltage	5 V
Collector DC Current	25 mA
Power Dissipation	100 mW ⁽²⁾

Electrical Characteristics ($T_A = 25^\circ\text{C}$ unless otherwise noted)

(OPB703, OPB703WZ, OPB704, OPB704WZ, OPB705, OPB705WZ, OPB704G, OPB704GWZ, OPB70HWZ)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
--------	-----------	-----	-----	-----	-------	-----------------

Input Diode (See OP265 for additional information — for reference only)

V_F	Forward Voltage	-	-	1.7	V	$I_F = 40\text{mA}$
I_R	Reverse Current	-	-	100	μA	$V_R = 2\text{ V}$

Output Phototransistor (See OP505 for additional information — for reference only)

$V_{(BR)CEO}$	Collector-Emitter Breakdown Voltage	30	-	-	V	$I_{CE} = 100\text{ }\mu\text{A}$
$V_{(BR)ECO}$	Emitter-Collector Breakdown Voltage	5	-	-	V	$I_{EC} = 100\mu\text{A}$
I_{CEO}	Collector Dark Current	-	-	250	nA	$V_{CE} = 10\text{ V}, I_F = 0, E_E = 0$

Coupled

$I_{C(ON)}$	On-State Collector Current OPB70HWZ OPB703, OPB703WZ OPB704, OPB704WZ OPB705, OPB705WZ	0.60 0.30 0.20 0.15	- - - -	3.5 2.5 2.5 1.0	mA	$V_{CE} = 5\text{ V}, I_F = 40\text{mA}, d = 0.15''^{(3)(7)}$
	OPB704G, OPB704GWZ	0.50	-	6.0		$V_{CE} = 5\text{ V}, I_F = 40\text{mA}, d = 0.20''^{(3)(6)}$
I_{CX}	Crosstalk OPB703, OPB703WZ	-	-	20	μA	$V_{CE} = 5\text{ V}, I_F = 40\text{mA}^{(6)}$
	OPB704, OPB704WZ, OPB70HWZ	-	-	20		
	OPB705, OPB705WZ	-	-	10		

Notes:

- (1) RMA flux is recommended. Duration can be extended to 10 seconds maximum when flow soldering.
- (2) For OPB703, OPB704 and OPB705, derate linearly $1.67\text{ mW}/^\circ\text{C}$ above 25°C .
- (3) For OPB703WZ, OPB704WZ, OPB705WZ, OPB70BWZ, OPB704G, OPB704GWZ and OPB70HWZ derate linearly $1.82\text{ mW}/^\circ\text{C}$ above 25°C .
- (4) The distance from the assembly face to the reflective surface is d.
- (5) Crosstalk (I_{CX}) is the collector current measured with the indicated current in the input diode and with no reflecting surface.
- (6) Measured using Eastman Kodak neutral white test card with 90% diffuse reflectance as a reflecting surface. Reference: Eastman Kodak, Catalog # E 152 7795.

OPTEK reserves the right to make changes at any time in order to improve design and to supply the best product possible.

C.3 Motor pas a pas Nema 17

Stepper Motors and Encoders

Specifications

NEMA 17 Motor

Electrical

Step angle.....	1.8 deg
Steps per revolution.....	200
Angular accuracy	±3%
Phases	2

Industry Standards

Industrial standards	CE, UR
Sealing standard.....	IP40
RoHS compliance.....	Yes

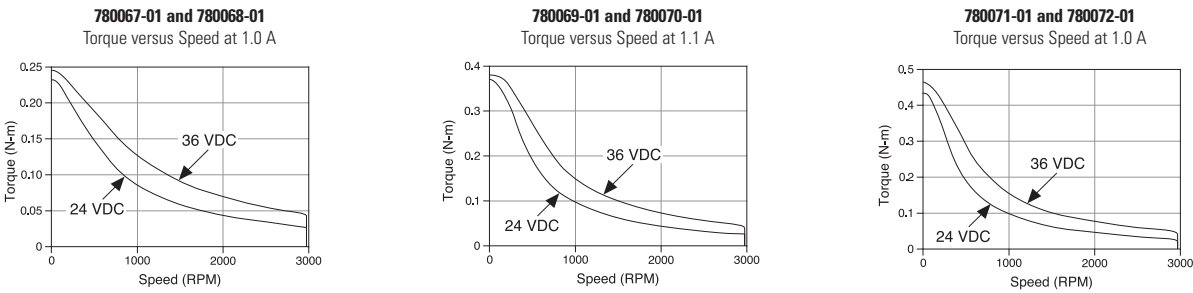
Physical

Operating temperature	-20 to 40 °C
Shaft load (20,000 hours at 1,500 rpm)	
Radial	15 lb (6.8 kg) at shaft center
Axial push	6 lb (2.7 kg)
Axial pull.....	15 lb (6.8 kg)

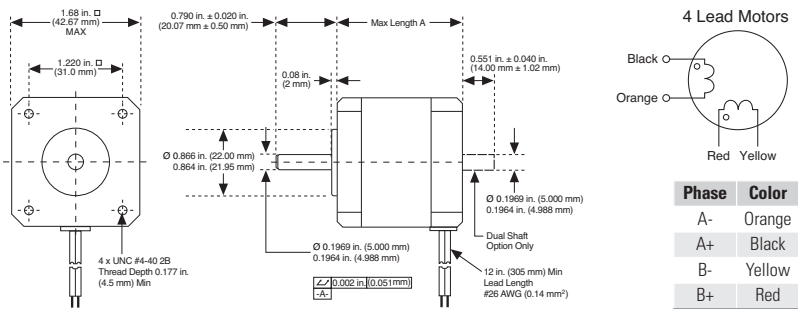
NI Part Number	Manufacturer Part Number	Dual Shaft	Drive	Amps/Phase	Holding Torque oz-in. (N · m)	Rotor Inertia oz-in.-s ² (kg-m ² x10 ⁻³)	Phase Inductance mH	Phase Resistance Ω ±10%	Detent Torque oz-in. (N · m)	Thermal Resistance °C/watt	Max Speed rpm
780067-01	CTP10ELF10MAA00	—	P70530	1.0	43 (0.30)	0.0005 (0.0040)	7.7	5.25	1.98 (0.014)	6.21	3000
780068-01	CTP10ELF10MMA00	✓									
780069-01	CTP11ELF11MAA00	—		1.1	63 (0.44)	0.0008 (0.0050)	11	5.19	2.55 (0.018)	5.44	3000
780070-01	CTP11ELF11MMA00	✓									
780071-01	CTP12ELF10MAA00	—		1.0	80 (0.56)	0.0011 (0.0070)	12	6.51	2.97 (0.021)	4.71	3000
780072-01	CTP12ELF10MAA00	✓									

Rated current is per phase, with the motor mounted, and winding temperature rise ΔT = 90 °C.
Resistance is with winding at 20 °C.

Torque versus Speed



Dimensions and Wiring



NI Part Number	Manufacturer Part Number	Dual Shaft	Max Length A in. (mm)	Net Weight lb (kg)
780067-01	CTP10ELF10MAA00	—	1.37 (34.7)	0.441 (0.200)
780068-01	CTP10ELF10MMA00	✓		
780069-01	CTP11ELF11MAA00	—	1.61 (40.9)	0.573 (0.260)
780070-01	CTP11ELF11MMA00	✓		
780071-01	CTP12ELF10MAA00	—	1.92 (48.8)	0.750 (0.340)
780072-01	CTP12ELF10MAA00	✓		

BUY ONLINE at ni.com or CALL 800 813 3693 (U.S.)

C.4 Circuit integrat DRV8825

STEPPER MOTOR CONTROLLER IC

Check for Samples: [DRV8825](#)

FEATURES

- **PWM Microstepping Motor Driver**
 - **Built-In Microstepping Indexer**
 - **Five-Bit Winding Current Control Allows Up to 32 Current Levels**
 - **Low MOSFET On-Resistance**
- **2.5-A Maximum Drive Current at 24 V, 25°C**
- **Built-In 3.3-V Reference Output**
- **8.2-V to 45-V Operating Supply Voltage Range**
- **Thermally Enhanced Surface Mount Package**

APPLICATIONS

- **Automatic Teller Machines**
- **Money Handling Machines**
- **Video Security Cameras**
- **Printers**
- **Scanners**
- **Office Automation Machines**
- **Gaming Machines**
- **Factory Automation**
- **Robotics**

DESCRIPTION

The DRV8825 provides an integrated motor driver solution for printers, scanners, and other automated equipment applications. The device has two H-bridge drivers, and can drive a bipolar stepper motor or two DC motors. The output driver block for each consists of N-channel power MOSFET's configured as full H-bridges to drive the motor windings. The DRV8825 can supply up to 2.5-A peak or 1.75-A RMS output current (with proper heatsinking at 24 V and 25°C).

A simple step/direction interface allows easy interfacing to controller circuits. Pins allow configuration of the motor in full-step up to 1/32-step modes. Decay mode is programmable.

Internal shutdown functions are provided for overcurrent protection, short circuit protection, undervoltage lockout and overtemperature.

The DRV8825 is available in a 28-pin HTSSOP package with PowerPAD™ (Eco-friendly: RoHS & no Sb/Br).

ORDERING INFORMATION⁽¹⁾

PACKAGE ⁽²⁾		ORDERABLE PART NUMBER	TOP-SIDE MARKING
PowerPAD™ (HTSSOP) - PWP	Reel of 2000	DRV8825PWPR	8825

(1) For the most current packaging and ordering information, see the Package Option Addendum at the end of this document, or see the TI web site at www.ti.com.

(2) Package drawings, thermal data, and symbolization are available at www.ti.com/packaging.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PowerPAD is a trademark of Texas Instruments.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of the Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2010–2011, Texas Instruments Incorporated

DEVICE INFORMATION

Functional Block Diagram

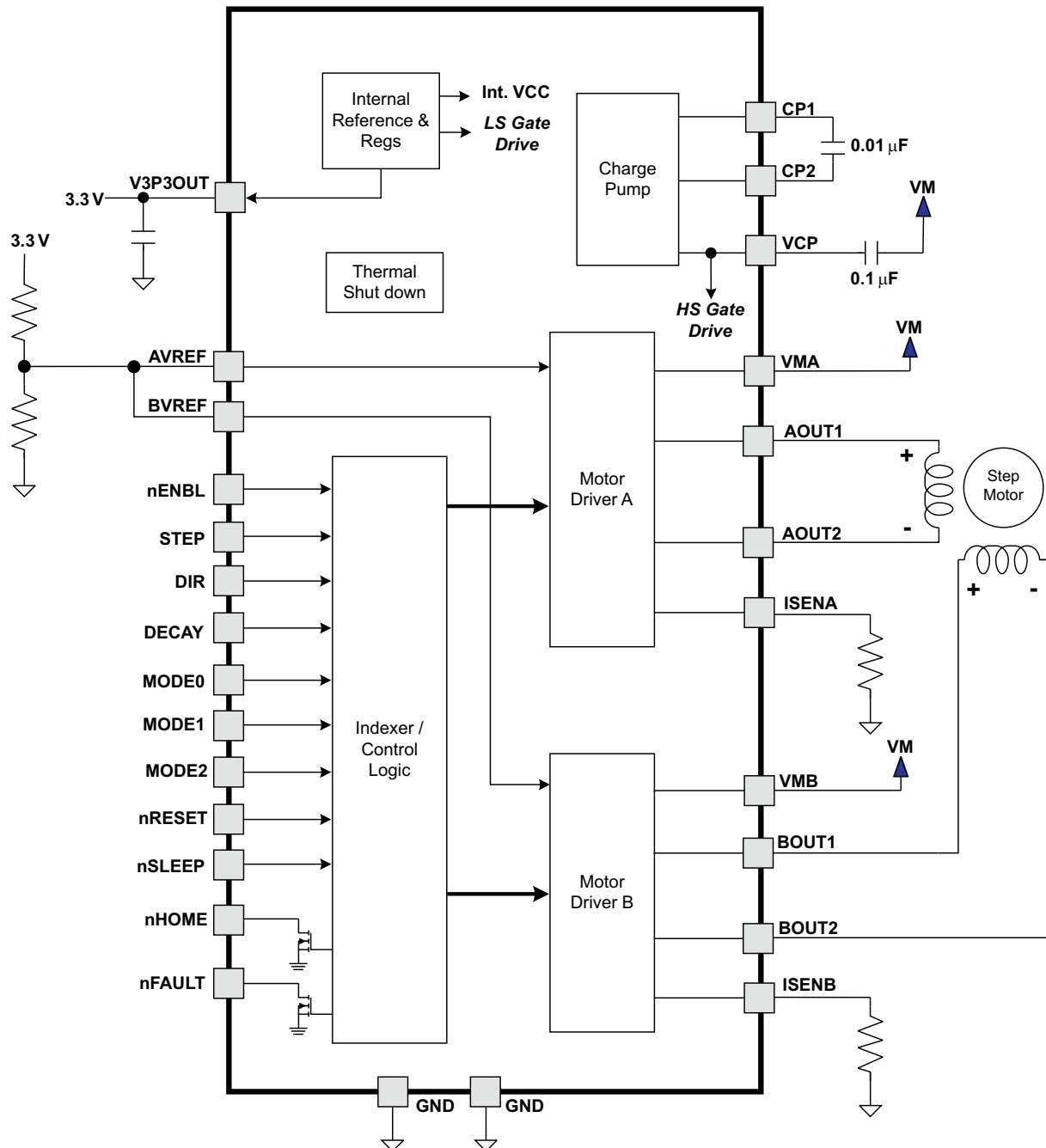
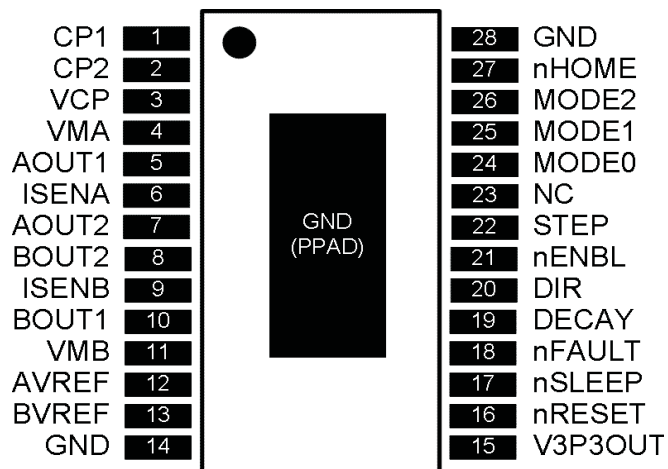


Table 1. TERMINAL FUNCTIONS

NAME	PIN	I/O ⁽¹⁾	DESCRIPTION	EXTERNAL COMPONENTS OR CONNECTIONS
POWER AND GROUND				
GND	14, 28	-	Device ground	
VMA	4	-	Bridge A power supply	Connect to motor supply (8.2 - 45 V). Both pins must be connected to same supply.
VMB	11	-	Bridge B power supply	
V3P3OUT	15	O	3.3-V regulator output	Bypass to GND with a 0.47-μF 6.3-V ceramic capacitor. Can be used to supply VREF.
CP1	1	IO	Charge pump flying capacitor	Connect a 0.01-μF 50-V capacitor between CP1 and CP2.
CP2	2	IO	Charge pump flying capacitor	
VCP	3	IO	High-side gate drive voltage	Connect a 0.1-μF 16-V ceramic capacitor to VM.
CONTROL				
nENBL	21	I	Enable input	Logic high to disable device outputs and indexer operation, logic low to enable. Internal pulldown.
nSLEEP	17	I	Sleep mode input	Logic high to enable device, logic low to enter low-power sleep mode. Internal pulldown.
STEP	22	I	Step input	Rising edge causes the indexer to move one step. Internal pulldown.
DIR	20	I	Direction input	Level sets the direction of stepping. Internal pulldown.
MODE0	24	I	Microstep mode 0	MODE0 - MODE2 set the step mode - full, 1/2, 1/4, 1/8/ 1/16, or 1/32 step. Internal pulldown.
MODE1	25	I	Microstep mode 1	
MODE2	26	I	Microstep mode 2	
DECAY	19	I	Decay mode	Low = slow decay, open = mixed decay, high = fast decay. Internal pulldown and pullup.
nRESET	16	I	Reset input	Active-low reset input initializes the indexer logic and disables the H-bridge outputs. Internal pulldown.
AVREF	12	I	Bridge A current set reference input	Reference voltage for winding current set. Normally AVREF and BVREF are connected to the same voltage. Can be connected to V3P3OUT.
BVREF	13	I	Bridge B current set reference input	
STATUS				
nHOME	27	OD	Home position	Logic low when at home state of step table
nFAULT	18	OD	Fault	Logic low when in fault condition (overtemp, overcurrent)
OUTPUT				
ISENA	6	IO	Bridge A ground / Isense	Connect to current sense resistor for bridge A.
ISENB	9	IO	Bridge B ground / Isense	Connect to current sense resistor for bridge B.
AOUT1	5	O	Bridge A output 1	Connect to bipolar stepper motor winding A. Positive current is AOUT1 → AOUT2
AOUT2	7	O	Bridge A output 2	
BOUT1	10	O	Bridge B output 1	Connect to bipolar stepper motor winding B. Positive current is BOUT1 → BOUT2
BOUT2	8	O	Bridge B output 2	

(1) Directions: I = input, O = output, OZ = tri-state output, OD = open-drain output, IO = input/output



ABSOLUTE MAXIMUM RATINGS⁽¹⁾⁽²⁾

		VALUE	UNIT
VMx	Power supply voltage range	−0.3 to 47	V
	Digital pin voltage range	−0.5 to 7	V
VREF	Input voltage	−0.3 to 4	V
	ISENSEx pin voltage	−0.3 to 0.8	V
Peak motor drive output current, t < 1 μS		Internally limited	A
Continuous motor drive output current ⁽³⁾		2.5	A
ESD rating	HBD (human body model)	2000	V
	CDM (charged device model)	500	
Continuous total power dissipation		See Dissipation Ratings table	
T _J	Operating junction temperature range	−40 to 150	°C
T _{stg}	Storage temperature range	−60 to 150	°C

- (1) Stresses beyond those listed under absolute maximum ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under recommended operating conditions is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.
- (2) All voltage values are with respect to network ground terminal.
- (3) Power dissipation and thermal limits must be observed.

THERMAL INFORMATION

THERMAL METRIC ⁽¹⁾		DRV8825	UNITS
		PWP	
		28 PINS	
θ_{JA}	Junction-to-ambient thermal resistance ⁽²⁾	31.6	°C/W
θ_{JCTop}	Junction-to-case (top) thermal resistance ⁽³⁾	15.9	
θ_{JB}	Junction-to-board thermal resistance ⁽⁴⁾	5.6	
ψ_{JT}	Junction-to-top characterization parameter ⁽⁵⁾	0.2	
ψ_{JB}	Junction-to-board characterization parameter ⁽⁶⁾	5.5	
θ_{JCbot}	Junction-to-case (bottom) thermal resistance ⁽⁷⁾	1.4	

- (1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, [SPRA953](#).
- (2) The junction-to-ambient thermal resistance under natural convection is obtained in a simulation on a JEDEC-standard, high-K board, as specified in JESD51-7, in an environment described in JESD51-2a.
- (3) The junction-to-case (top) thermal resistance is obtained by simulating a cold plate test on the package top. No specific JEDEC-standard test exists, but a close description can be found in the ANSI SEMI standard G30-88.
- (4) The junction-to-board thermal resistance is obtained by simulating in an environment with a ring cold plate fixture to control the PCB temperature, as described in JESD51-8.
- (5) The junction-to-top characterization parameter, ψ_{JT} , estimates the junction temperature of a device in a real system and is extracted from the simulation data for obtaining θ_{JA} , using a procedure described in JESD51-2a (sections 6 and 7).
- (6) The junction-to-board characterization parameter, ψ_{JB} , estimates the junction temperature of a device in a real system and is extracted from the simulation data for obtaining θ_{JA} , using a procedure described in JESD51-2a (sections 6 and 7).
- (7) The junction-to-case (bottom) thermal resistance is obtained by simulating a cold plate test on the exposed (power) pad. No specific JEDEC standard test exists, but a close description can be found in the ANSI SEMI standard G30-88.

RECOMMENDED OPERATING CONDITIONS

		MIN	NOM	MAX	UNIT
V _M	Motor power supply voltage range ⁽¹⁾	8.2		45	V
V _{REF}	VREF input voltage ⁽²⁾	1		3.5	V
I _{V3P3}	V3P3OUT load current	0		1	mA

- (1) All V_M pins must be connected to the same supply voltage.
- (2) Operational at VREF between 0 V and 1 V, but accuracy is degraded.

ELECTRICAL CHARACTERISTICS

over operating free-air temperature range of -40°C to 85°C (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
POWER SUPPLIES						
I_{VM}	VM operating supply current	$V_M = 24\text{ V}$		5	8	mA
I_{VMQ}	VM sleep mode supply current	$V_M = 24\text{ V}$		10	20	μA
V_{UVLO}	VM undervoltage lockout voltage	V_M rising		7.8	8.2	V
V3P3OUT REGULATOR						
V_{3P3}	V3P3OUT voltage	$I_{OUT} = 0$ to 1 mA	3.2	3.3	3.4	V
LOGIC-LEVEL INPUTS						
V_{IL}	Input low voltage			0.6	0.7	V
V_{IH}	Input high voltage		2.2		5.25	V
V_{HYS}	Input hysteresis		0.3	0.45	0.6	V
I_{IL}	Input low current	$V_{IN} = 0$	-20		20	μA
I_{IH}	Input high current	$V_{IN} = 3.3\text{ V}$			100	μA
R_{PD}	Internal pulldown resistance			100		k Ω
nHOME, nFAULT OUTPUTS (OPEN-DRAIN OUTPUTS)						
V_{OL}	Output low voltage	$I_O = 5\text{ mA}$			0.5	V
I_{OH}	Output high leakage current	$V_O = 3.3\text{ V}$			1	μA
DECAY INPUT						
V_{IL}	Input low threshold voltage	For slow decay mode			0.8	V
V_{IH}	Input high threshold voltage	For fast decay mode	2			V
I_{IN}	Input current				± 40	μA
R_{PU}	Internal pullup resistance (up to 3.3 V)			130		k Ω
R_{PD}	Internal pulldown resistance			80		k Ω
H-BRIDGE FETS						
$R_{DS(ON)}$	HS FET on resistance	$V_M = 24\text{ V}, I_O = 1\text{ A}, T_J = 25^\circ\text{C}$		0.2		Ω
		$V_M = 24\text{ V}, I_O = 1\text{ A}, T_J = 85^\circ\text{C}$		0.25	0.32	
	LS FET on resistance	$V_M = 24\text{ V}, I_O = 1\text{ A}, T_J = 25^\circ\text{C}$		0.2		
		$V_M = 24\text{ V}, I_O = 1\text{ A}, T_J = 85^\circ\text{C}$		0.25	0.32	
I_{OFF}	Off-state leakage current		-20		20	μA
MOTOR DRIVER						
f_{PWM}	Internal current control PWM frequency			30		kHz
t_{BLANK}	Current sense blanking time			4		μs
t_R	Rise time		30		200	ns
t_F	Fall time		30		200	ns
PROTECTION CIRCUITS						
I_{OCP}	Overcurrent protection trip level		3			A
t_{OCP}	Overcurrent protection deglitch time			2.5		μs
t_{TSD}	Thermal shutdown temperature	Die temperature	150	160	180	$^\circ\text{C}$
CURRENT CONTROL						
I_{REF}	xVREF input current	$xVREF = 3.3\text{ V}$	-3		3	μA
V_{TRIP}	xISENSE trip voltage	$xVREF = 3.3\text{ V}, 100\%$ current setting	635	660	685	mV
ΔI_{TRIP}	Current trip accuracy (relative to programmed value)	$xVREF = 3.3\text{ V}, 5\%$ current setting	-25		25	%
		$xVREF = 3.3\text{ V}, 10\%-34\%$ current setting	-15		15	
		$xVREF = 3.3\text{ V}, 38\%-67\%$ current setting	-10		10	
		$xVREF = 3.3\text{ V}, 71\%-100\%$ current setting	-5		5	

ELECTRICAL CHARACTERISTICS (continued)

over operating free-air temperature range of -40°C to 85°C (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
A_{SENSE}	Current sense amplifier gain	Reference only		5		V/V

TIMING REQUIREMENTS

			MIN	MAX	UNIT
1	f_{STEP}	Step frequency		250	kHz
2	$t_{\text{WH(STEP)}}$	Pulse duration, STEP high	1.9		μs
3	$t_{\text{WL(STEP)}}$	Pulse duration, STEP low	1.9		μs
4	$t_{\text{SU(STEP)}}$	Setup time, command to STEP rising	650		ns
5	$t_{\text{H(STEP)}}$	Hold time, command to STEP rising	650		ns
6	t_{ENBL}	Enable time, nENBL active to STEP	650		ns
7	t_{WAKE}	Wakeup time, nSLEEP inactive to STEP	1.7		ms

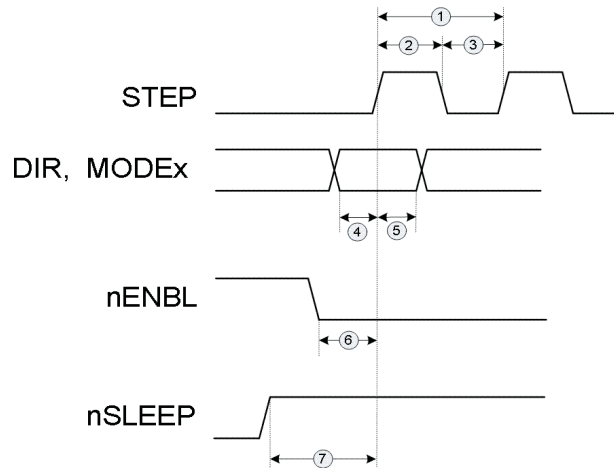


Figure 1. Timing Diagram

BIBLIOGRAFIA

- [1] Hector C. Vargas. Design and fabrication of active tilt-stage for laser rangefinder. Master's thesis, Department of Mechanical Engineering, MIT, 2009. 1.1
- [2] Mehtap Arli Denis Klimentjew and Jianwei Zhang. 3d scene reconstruction based on a moving 2d laser range finder for service-robots. In *Robotics and Biomimetics (ROBIO)*, 2009. 1.1
- [3] Toyokazu Kawahara Kazunori Ohno and Satoshi Tadokoro. Development of 3d laser scanner for measuring uniform and dense 3d shapes of static objects in dynamic environment. *IEEE*, 2008. 1.1
- [4] Aaron Martinez and Enrique Fernández. *Learning ROS for Robotics Programming*. Packt Publishing, September 2013.
- [5] J.S. Chen N.J. Chen. 3d scenes registration using a 2d laser range finder. In *Automation and Logistics (ICAL)*, 2010. 1.1
- [6] Jason M. O'Kane. *A Gentle Introduction to ROS*. Independently published, October 2013. Available at <http://www.cse.sc.edu/~jokane/agitr/>.
- [7] Sarah Price. Laser ranging using time of flight. *Institute for Computer Based Learning*, July 1996.