Universidad Zaragoza
1542

Trabajo Fin de Grado

LoCoQuad:

Diseño y desarrollo de una plataforma robótica de bajo coste

LoCoQuad:

Design and development of a low-cost robotic platform

Autor

Manuel Bernal Lecina

Director

Javier Civera Sancho

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2020

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Manuel Bernal Lecina ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado (Título del Trabajo)

LoCoQuad: Diseño y desarrollo de una plataforma robótica de bajo coste.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 19 de Junio de 2020

Fdo: D. Manuel Bernal Lecina

AGRADECIMIENTOS

No me olvido de nadie, pues todos me han hecho mejor persona, mejor ingeniero o mejor ser vivo. En especial me acuerdo de los que siempre han estado ahí. De los que me acompañaron durante los primeros años. De los que me ayudaron a seguir adelante. De los que vivieron mis locuras. De los que me sacaron del pozo. De los que se fueron para bien. De los que se quedaron para disfrutar. De los que no aguantaron a mi lado. De los que tuve que dejar de ver. De los que se rieron conmigo. De los que aún siguen aquí. De los que quiero y siempre querré. De los que me conocen casi del todo. De los que me dieron una y dos oportunidades. De los que me dieron muchas más.

Gracias.

LoCoQuad:

Diseño y desarrollo de una plataforma robótica de bajo coste

RESUMEN

En esta memoria se reflejan los pasos llevados a cabo para el diseño, construcción y validación de una plataforma robótica de bajo coste llamada LoCoQuad. Este nombre viene del inglés "Low Cost Quadruped" o Cuadrúpedo de bajo coste. Los cuadrúpedos son una tipología de robot que desde hace unos años se ha desarrollado en todo el mundo, tratando de buscarles un propósito más allá de la investigación en sistemas móviles autónomos. Para llevar a cabo esta tarea se han desarrollado varias plataformas dentro de un rango de precios muy amplio. La única condición para considerar que un robot sea un cuadrúpedo es que posea cuatro patas, pero éstas pueden tener gran cantidad de diferencias. En general las patas se clasifican según su número de ejes y la disposición de los mismos. Siendo las patas más usadas las de 3 ejes, debido a la capacidad de éstas para alcanzar cualquier posición dentro de su campo de trabajo. Las disposiciones más usadas son las conocidas como mamíferas, semejantes a las patas de cualquier mamífero, y arácnidas, semejantes a las de insectos y arañas.

LoCoQuad es un cuadrúpedo diseñado para investigación y docencia, cuyo objetivo es ofrecer una plataforma con varias configuraciones. LoCoQuad se ha desarrollado partiendo de cero en este TFG, y es actualmente un producto terminado. Mediante un diseño cuidadoso, se ha conseguido que sea el cuadrúpedo más asequible del mundo (con un coste aproximado de 150$). Se han liberado el diseño y el código realizado, y se ha hecho público un informe técnico con el objetivo de someterlo a una conferencia científica. El proyecto ha tenido ya un impacto considerable: se ha recibido el primer premio tecnológico en el III Certamen de Jóvenes Creadores Aragoneses y se me contactó para una entrevista en un blog de tecnología internacional con una repercusión considerable en redes sociales.

Índice

Capítulo 1

Introducción

Actualmente, la robótica está cruzando la barrera de la investigación para pasar a formar parte del mundo globalizado en el que vivimos. Los brazos robóticos industriales cruzaron esta línea hace varias décadas, mientras que otros como el robot limpiador Roomba o las plataformas robóticas para logística lo han hecho en los últimos años. En el momento actual los mercados buscan huecos para nuevos productos robóticos y se desarrollan plataformas con mayores capacidades y mayores grados de fiabilidad. En el campo de la investigación también se trabaja para dar nuevas capacidades a los robots o desarrollar nuevos robots con características específicas. Un ejemplo serían los drones, robots aéreos que se distancian significativamente de los brazos robóticos. Los drones tienen otros objetivos de más dificultad técnica, y por ello se enfrentan a nuevos retos. Éstos se resuelven en los grupos de investigación de universidades y empresas tecnológicas de todo el mundo. Cuando los problemas nucleares alcanzan una cierta madurez, pasan a ser las empresas las que continúan el desarrollo y la mejora de estas plataformas.

Una de esas tareas en las que se necesita todavía cierto esfuerzo investigador, antes de poder pasar a ser un producto más, es el aprendizaje automático. Dentro de la inteligencia artificial (AI) existen gran variedad de estrategias y técnicas para alcanzar el funcionamiento autónomo de agentes. En la arista en la que se unen la inteligencia artificial con la robótica, estos algoritmos tienen una gran importancia, ya que permitirán que los robots del mañana puedan aprender a relacionarse con su entorno de manera independiente. En el momento actual dicha percepción e interacción autónoma es todavía un escenario lejano, pero se está trabajando en ideas y líneas de investigación que permitirán, al menos poco a poco, ir alcanzando

ese objetivo.

Una de las estrategias que se engloban dentro de la inteligencia artificial se llama “aprendizaje por refuerzo”, del inglés “reinforcement learning” (RL). Esta técnica se basa en un ciclo de aprendizaje en el que el sistema persigue la mayor recompensa posible por sus acciones. En otras palabras, se define un objetivo para el sistema, se le permiten una serie de acciones y en función del resultado obtenido se le premia con una moneda virtual. Este ciclo se repite hasta que el sistema es capaz de alcanzar el objetivo por si mismo. El sistema recuerda las acciones necesarias para cumplir con ese objetivo, siendo capaz de reproducirlo desde ese momento en cualquier circunstancia.

Este algoritmo puede parecer sencillo, pero su aplicación a la robótica tiene más de un inconveniente. En primer lugar, una pequeña variación en el objetivo implica un nuevo ciclo de entrenamiento, haciendo que el proceso de aprendizaje sea muy largo si queremos que el robot sepa hacer un número elevado de tareas. Por otro lado, existe la posibilidad de que un objetivo sea demasiado complejo y el robot no sea capaz de completar nunca el aprendizaje. Otro inconveniente será la imposibilidad de utilizar las acciones aprendidas en otros robots de similares características, ya que la geometría y la configuración de cada robot puede variar de manera significativa.

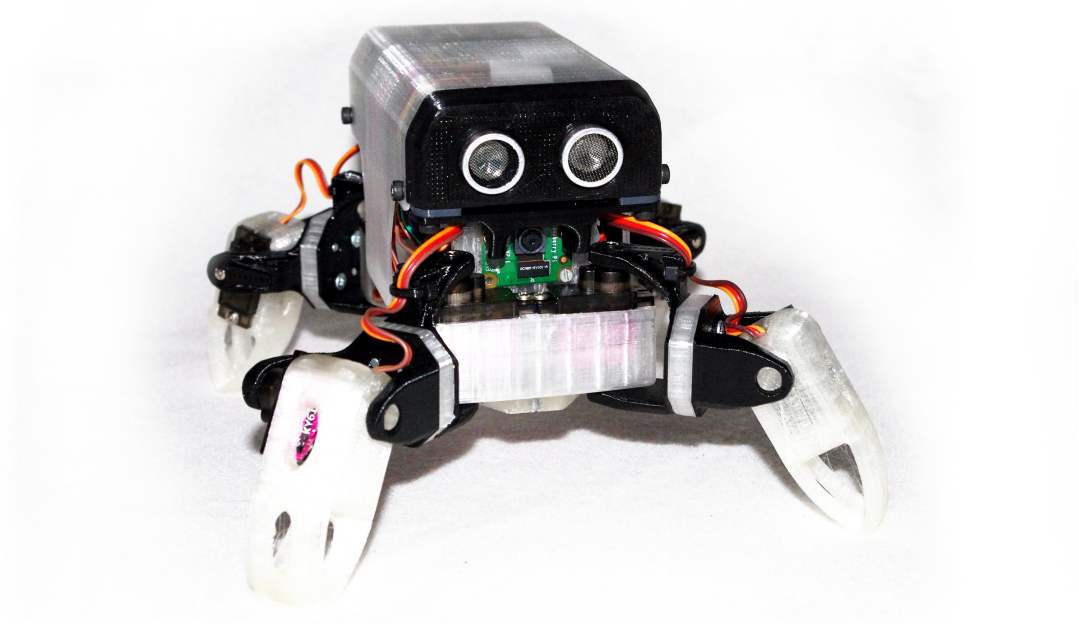


Figura 1.1: Plataforma robótica, LoCoQuad.

Por todo esto, decidimos desarrollar LoCoQuad (Fig.1.1), una plataforma robótica que nos permitirá investigar, desarrollar y, lo más importante, probar

algoritmos de RL a un coste muy bajo y con una flexibilidad muy grande.

LoCoQuad se ha desarrollado desde la primera pieza hasta la útima soldadura con el fin de ser útil para la comunidad investigadora. Mediante un diseño muy cuidadoso, se ha conseguido que LoCoQuad sea la plataforma robótica de cuatro patas de más bajo coste del estado del arte. Esto se consiguió mediante un minucioso proceso de toma de decisiones e iteraciones en el diseño, para garantizar que LoCoQuad tuviera a la vez configurabilidad, capacidades razonables de cálculo y locomoción y el mínimo coste. Se ha escrito un artículo científico[1] describiendo el desarrollo de la plataforma que al hacerlo público ha llevado a este robot a ganar premios (en el III Certamen de Jóvenes Creadores Aragoneses[2]) y ser mencionado por blogs tecnológicos internacionales[3].

El diseño de sus distintos componentes[4], el código desarrollado[5] y toda la documentación asociada han sido liberados para su uso de manera gratuita.

[1] https://arxiv.org/abs/2003.09025

[2] http://www.boa.aragon.es/cgi-bin/EBOA/BRSCGI?CMD=VEROBJ&MLKOB=1099040064343

[3] https://techxplore.com/news/2020-04-locoquad-arachnoid-inspired-robot-purposes.html

[4] https://www.thingiverse.com/thing:3853722

[5] https://github.com/TomBlackroad/LoCoQuad

Capítulo 2

Objetivos

Desde el comienzo de este trabajo, el objetivo principal ha sido el desarrollo de una plataforma robótica de bajo coste, código abierto, diseño abierto y con una gran configurabilidad.

Este objetivo requiere ser dividido en conceptos más concretos que una vez alcanzados permitan ser agrupados y den por satisfecho el objetivo principal.

Empezando por los pilares de cualquier robot, dividiremos los desarrollos en Mecánica, Electrónica y Programación. Además, se realizará un estudio previo, un análisis del punto inicial y una batería de tests. Se documentarán las problemáticas encontradas y se propondrán diferentes proyecciones a futuro para este proyecto.

Ordenados, los objetivos serán:

1. Estudio previo del estado del arte en el campo de la robótica móvil y el aprendizaje autónomo en robots.
2. Consideraciones y análisis del punto de partida antes de comenzar el desarrollo.
3. Diseño mecánico de una plataforma sobre la que instalar los sensores, los actuadores, la unidad de control, el sistema de alimentación y el de comunicaciones.
4. Iteración del diseño teniendo en cuenta los errores que aparezcan de la primera versión.
5. Diseño electrónico compatible con la estructura mecánica diseñada.

6. Integración de la mecánica y la electrónica en un sistema inanimado sobre el que poder empezar a programar.

7. Ajuste y verificación de la integración, programando pequeños programas de testeo.

8. Programación de una estructura de desarrollo básica sobre la que poder programar acciones de alto nivel.

9. Programación de acciones basadas en prueba-error.

10. Validación de la plataforma mediante las acciones programadas.

11. Análisis los problemas encontrados durante los objetivos anteriores.

12. Comentar las líneas de trabajo futuras, la proyección de la plataforma y su participación en otros proyectos.

Capítulo 3

Estudio Previo

Al comenzar este trabajo, lo primero que hacía falta era entender el contexto en el que se iba a trabajar, teniendo en cuenta el objetivo principal y tratando de comprender la relevancia del mismo dentro de una perspectiva lo más global y ambiciosa posible.

Hoy en día, la robótica es un campo de la investigación en el que se trabaja con gran interés para resolver los problemas que van apareciendo con el paso del tiempo. Hace varias décadas, el interés principal de la robótica era la manufacturación. La automatización de procesos industriales conllevó la reducción de los tiempos de trabajo por pieza y esto tuvo un gran impacto tanto en la capacidad de producción como en la reducción de los costes de producción. Ha sido la investigación en robótica la que ha permitido el desarrollo industrial de los últimos años, a la vez que ha permitido expandir las fronteras de este campo fuera del ámbito industrial. Otros campos en desarrollo de la investigación en robótica se están centrando en sistemas móviles autónomos, desarrollando capacidades como la visión o la navegación en entornos reales no controlados como las fábricas. Los nuevos robots ya no son brazos robóticos, sino submarinos, drones, automóviles autónomos, prótesis biónicas, plataformas logísticas, cuadrúpedos o hexápodos entre otros. Este fenómeno expansivo ha traído una gran variedad de nuevas plataformas robóticas sobre las que los investigadores llevan poco tiempo trabajando. Esto supone una gran ventaja, ya que se retoma la capacidad de diseñar, algo abandonada tras ser adoptadas las fisionomías estándar para brazos industriales.

Otro gran avance del siglo XXI ha sido la evolución de la programación, su liberalización y su apertura a la mayor parte de la comunidad investigadora.

Programar ya no es algo exclusivo de programadores, sino que cualquier investigador tiene la posibilidad y a veces la obligación de desarrollar programas propios. Aparecen las comunidades Open-Source o de "código abierto", que permiten compartir, y documentarse en los lenguajes de programación más utilizados. En el campo de la robótica estos lenguajes serían sin lugar a duda: C++, C y Python. Además, no se nos puede olvidar el sistema operativo Open-Source por antonomasia, Linux, en alguna de sus distribuciones y sabores más conocidos como Ubuntu o Debian. En los últimos años la revitalización de Ubuntu ha sido un importante avance para la investigación en robótica, sumándose o, en cierto modo, gracias al lanzamiento de ROS (Robotic Operative System) [1]. Una plataforma capaz de crear la red de comunicaciones dentro de cualquier robot y sobre la que podemos implementar algoritmos en Python y C++, además de establecer comunicaciones multi-robot o con el exterior. ROS ha cobrado un papel protagonista en la investigación robótica gracias a participar de la filosofía Open-Source, permitiendo a todos los investigadores del mundo reunirse alrededor de un entorno común y compartido.

Siguiendo con la filosofía Open-Source, hay que mencionar también el crecimiento de las plataformas electrónicas que se suman a esta idea de compartir y colaborar. Ideas como el proyecto Arduino [2] o la evolución de plataformas como Raspberry Pi [3] o Orange Pi [4] entre otras ha resultado ser toda una revolución en diferentes campos, como la educación, la industria del prototipado o la propia investigación. En el campo de la robótica su impacto ha sido menor, pero en cambio han hecho asequible la creación de sistemas complejos a precios mucho más asequibles que anteriormente, ya que se utilizan soluciones generalistas de bajo coste y plataformas abiertas para implementar comportamientos específicos, reduciendo el desarrollo de una plataforma robótica básicamente a su programación.

Por último, hay que mencionar que, para poder contar con un robot, la programación y la electrónica ya se han cubierto. Queda por tratar la mecánica y el diseño. Estos dos campos son en realidad uno solo y en relación con lo planteado anteriormente, todo pasa por software de diseño 3D Open-Source [5] y tecnologías de impresión 3D [6]. Ambas han evolucionado mucho en los últimos años y actualmente

[1] www.ros.org
[2] www.arduino.cc
[3] www.raspberrypi.org
[4] www.orangepi.org
[5] https://www.autodesk.com/products/fusion-360/overview
[6] www.ultimaker.com/software/ultimaker-cura

los resultados que permiten obtener son de una calidad excepcional.

Para poder tener una referencia de los robots de bajo coste disponibles en el mercado hasta la fecha, se realizó la tabla 3.1. En ella se muestran las plataformas robóticas junto a su precio unitario y la tipología de robot. El precio objetivo de LoCoQuad se pretendía que fuera competitivo y además supusiera una gran diferencia frente a otros robots de su misma tipología. Como se verá más adelante, este objetivo se logró y LoCoQuad es el cuadrúpedo arácnido de sus características más barato del mundo.

Plataforma	**Precio [$]**	**Tipología**
Aracna [1]	1389	Quadrúpedo
E-Puck [2]	280	Ruedas
Phantom X MRIII[7]	1300	Hexapodo
Roomba [3]	200	Ruedas
CotsBots [4]	200	Ruedas
Kilobot [5]	100	Vibración
Khepera IV [6]	2670	Ruedas
PiArm	263	Brazo
NAO power V6 [7]	9000	Humanoide
Cozmo[8]	180	Ruedas
Sphero RVR[9]	250	Ruedas
RoboMaster S1[10]	400	Ruedas
Molecubes [8]	350	Modular
Lynxmotion SQ3U	550	Quadrúpedo
LoCoQuad	**150-165**	Quadrúpedo

Tabla 3.1: Coste de LoCoQuad frente a otras pltaformas

Con lo enunciado anteriormente disponemos de una idea general del estado de la ciencia para desarrollar plataformas robóticas alternativas a los brazos robóticos de finales del siglo XX, manteniendo una filosofía Open-Source y siguiendo así la línea de nuestro objetivo principal. En relación al bajo coste que se persigue, cabe mencionar que las soluciones Open-Source son por lo general más asequibles que los productos cerrados.

[7] `https://www.trossenrobotics.com/phantomx-ax-hexapod.aspx`
[8] `https://anki.com/en-us/cozmo.html`
[9] `https://www.sphero.com/rvr`
[10] `https://www.dji.com/es/robomaster-s1`

Capítulo 4

Mecánica

El proceso comenzó con la selección de los elementos limitantes del diseño de cualquier robot pensado para interactuar con el mundo real: los actuadores. En los robots de patas, éstos pueden ser de muchísimas clases, pero generalmente se trata de motores eléctricos o modificaciones de éstos. En el Anexo B se tratan los actuadores para robótica en mayor profundidad.

Para este proyecto, teniendo en cuenta las condiciones iniciales y los objetivos, precisábamos de un actuador barato, ligero, con una buena relación peso-par, de pequeñas dimensiones, fácil de adquirir y fácil de programar. El resultado tras una extensa búsqueda fue el servo-motor MG90S, que cumple con todas los requerimientos con las siguientes características.

Características	Valores
Voltaje de operación	4.8 V a 6 V
Velocidad de operación	0.1 s/60º (4.8 V), 0.08 s/60º (6 V)
Par detenido	18 Ncm (4.8 V), 22 Ncm (6 V)
Capacidad de rotación	180º aprox. (90º en cada dirección)
Banda muerta	5 us
Peso ligero	13.4 g
Dimensiones compactas	Largo 22.5 mm, Ancho 12 mm, Alto 35.5 mm
Largo del cable	25 cm
Piñonería	Metálica

Tabla 4.1: Características Servo MG90

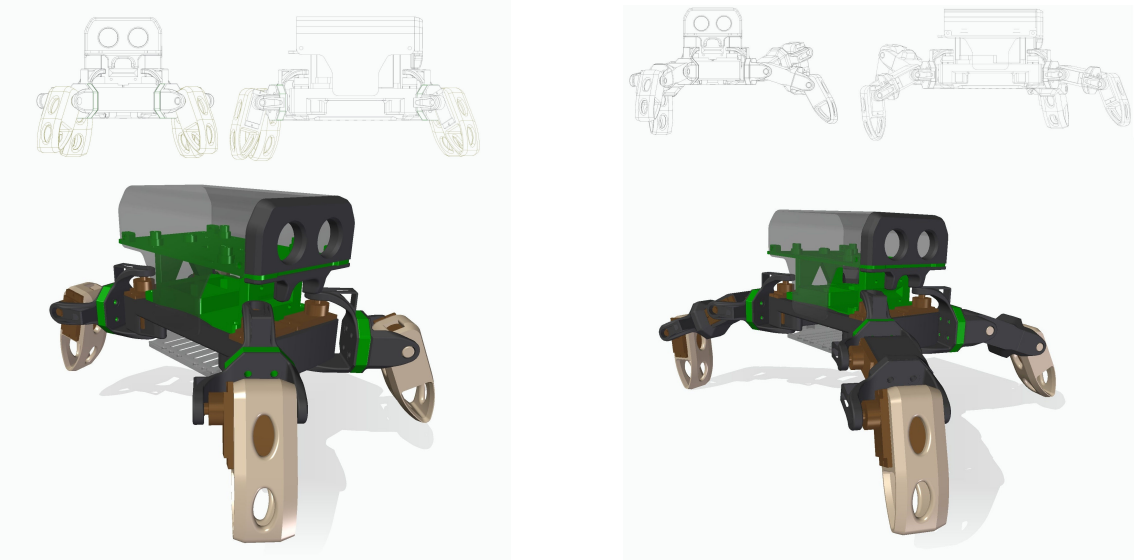
Una vez seleccionados los actuadores, debemos acotar las dimensiones y el peso máximo de nuestro robot para garantizar que los actuadores trabajen dentro de su rango de operación. Para realizar esta tarea, el proceso lógico es un ciclo de

iteración, ya que el peso y las dimensiones del robot van a estar relacionados y, por tanto, la modificación de una de las dos variables alterará el valor de la otra. Se optó por diseñar las patas más compactas que permitieran integrar los servos y verificar posteriormente que los motores fueran capaces de maniobrar el robot resultante.

En este punto, se decidió plantear un cuerpo central por niveles, que albergarían diferentes componentes. En este cuerpo central irían instalados cuatro motores, que harían de primeros ejes para las cuatro patas. Posteriormente y gracias a haber realizado el estudio previo, se detectó una carencia importante en las plataformas robóticas en general: la falta de flexibilidad en la configuración de las mismas. Los robots comerciales tienen objetivos específicos y requieren altos grados de precisión, y su diseño y programación están optimizados para tal propósito. En los robots educativos por el contrario existen algunas opciones en formato kit, que permiten variar los diseños de los robots con el objetivo de que éstos puedan utilizarse con una gran variedad de propósitos. Esta característica resulta realmente interesante para un robot que quiera aprender a moverse autónomamente, y sin que le afecten posibles casuísticas como el bloqueo, la pérdida o la rotura de uno de sus motores. Además, en el campo del aprendizaje autónomo, resulta de gran interés contar con plataformas que con un pequeño ajuste cambien alguna de sus patas o simplemente éstas sean desmontadas. Esto hace que los algoritmos deban estar atentos, se percaten de que algo está ocurriendo y puedan refinar su comportamiento para llegar a alcanzar sus metas u objetivos programados.

Fue esta reflexión la que hizo que se optara por un robot con diferentes configuraciones, agrupadas en torno a dos configuraciones principales. La primera con cuatro patas de dos ejes (2J) y la segunda con cuatro patas de tres ejes (3J). Además de todas las configuraciones intermedias y derivadas de éstas. En la figura 4.1 se pueden ver ambas configuraciones.

El motivo por el que se optó por una disposición tipo araña fue la estabilidad. En general, los robots con disposiciones de patas tipo mamífero (únicamente de 4 patas) son cuerpos que se elevan en mayor medida del suelo, haciéndolos mecánicamente más inestables. Por el contrario, las disposiciones arácnidas tienden a establecer distancias entre el cuerpo del robot y el suelo muy inferiores. Esto resultaba muy interesante para el proyecto ya que no se esperaba conseguir una gran estabilidad con los motores seleccionados y en concreto con la configuración 2J, en la que, debido a la falta del tercer eje, el campo de acción del mismo quedaba limitado a una



(a) LoCoQuad 2J (b) LoCoQuad 3J

Figura 4.1: Modelos de las dos configuraciones básicas de LoCoQuad.

superficie. Teóricamente, dicha superficie sería la piel exterior de medio toroide, con un radio exterior igual a la longitud del primer elemento del brazo y con un radio de revolución igual a la longitud del segundo elemento.

Para aumentar la configurabilidad del sistema, se decidió integrar unos discos multiposición entre los motores. Estos discos contaban con un patrón de agujeros que permitían girar los elementos de las patas sobre el eje principal de sus elementos para alterar sus posiciones iniciales. Estas posiciones se limitaron a cuatro: 0, 45, 90 y 135 grados. En la imagen de la figura 4.2 se muestran algunas de estas configuraciones.

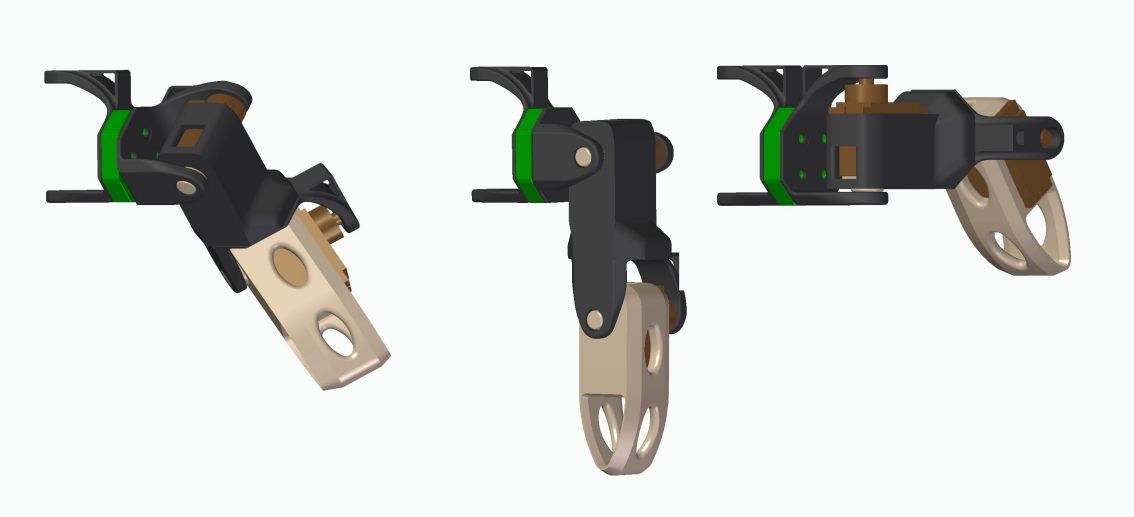


Figura 4.2: Modelos de algunas de las configuraciones de las patas.

Una vez concluido el diseño de las patas se pasó al del cuerpo, para el cual hubo que decidir todos los componentes que debían ser instalados dentro del robot y que se detallan más adelante. Como se comentaba antes, se optó por un diseño apilado, en el que cada nivel se dedicaba un tipo de componente concreto. En la figura 4.3 se ve el diseño final en modo expandido, lo cual nos permite ver las diferentes zonas del cuerpo del robot. En la base se instalan las baterías, protegidas por la tapa inferior. En el primer nivel, se insertan los cuatro motores de la base de las patas y la electrónica de potencia y de control. En el nivel superior, junto con la pared frontal, se fijan los sensores y la placa base.

A la hora de diseñar esta plataforma, siempre se tuvo en mente la incuestionable ventaja que supondría que cualquiera pudiera imprimir sus piezas en una impresora 3D. Por tanto, se limitaron las dimensiones y la complejidad de las piezas para que las impresoras 3D más simples no encontraran problemas a la hora de producir los diseños de LoCoQuad.

Una vez se concluyó el diseño y se validó su correcta impresión, era necesario

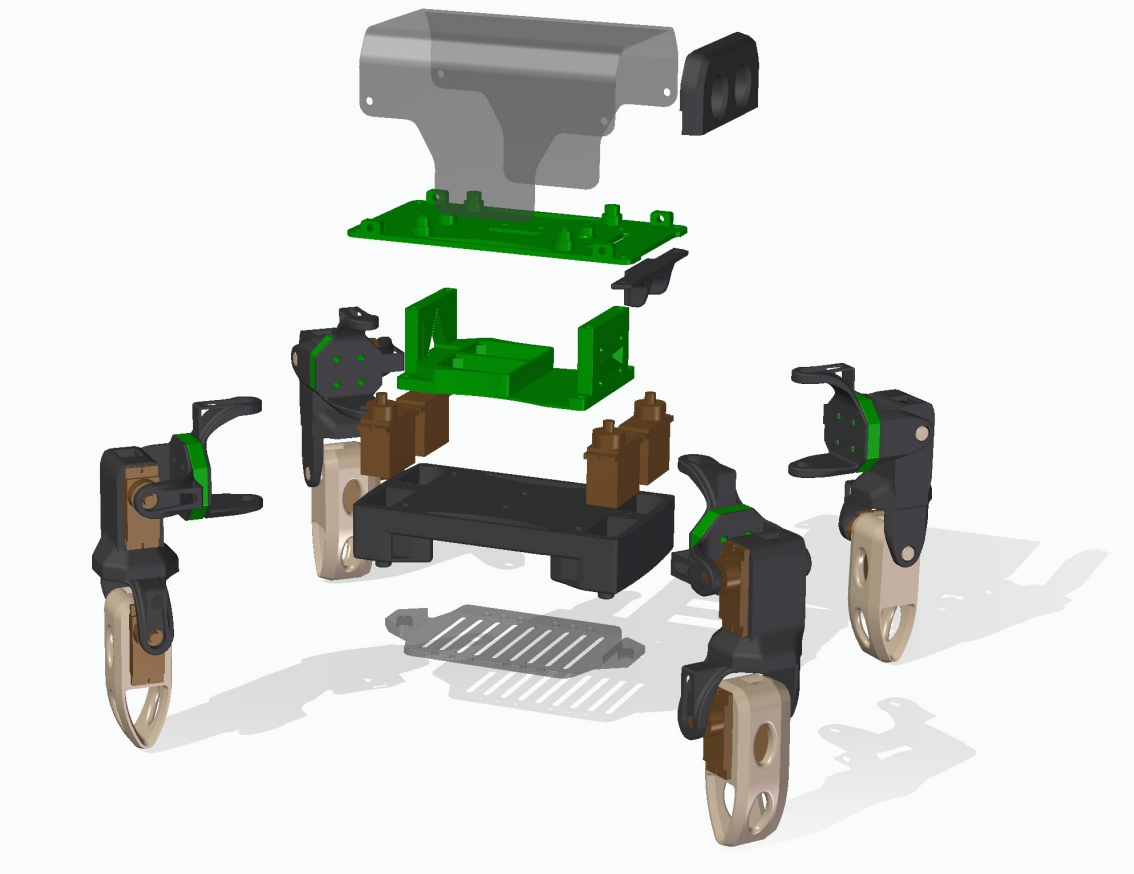


Figura 4.3: Modelo expandido del diseño

validar también que los motores fuesen capaces de maniobrar el diseño terminado. Tras imprimir las piezas, pesar los elementos de los brazos, y montado el cuerpo, ya se podía realizar esta comprobación.

En primer lugar, había que considerar la peor situación en la que se podía encontrar el robot. Claramente se debía realizar este estudio sobre la configuración 3J, asumiendo que el robot estaría apoyado sobre los extremos de sus patas. Se tomó la posición de reposo, que se muestra en la imagen de la figura 4.4 como la más desfavorable, ya que las patas opuestas en diagonal completamente extendidas suponían la distancia más grande entre dos puntos de apoyo. En la imagen también se detallan las distancias necesarias para realizar el equilibrio de momentos respecto del eje más desfavorecido, el más externo.

Para calcular el equilibrio de momentos aislamos una pata y suponemos que está sometida a un cuarto del peso del cuerpo de robot. En la figura 4.5 se muestran las fuerzas y los momentos que actúan sobre la pata. Se hace equilibrio de fuerzas para obtener el valor de la reacción. En dicha figura, BW representa el peso del cuerpo central del robot. W son los pesos de los elementos de las patas. J son las posiciones de las articulaciones. R es la reacción al contacto. Y Γ los pares en las articulaciones.

$$R = \frac{BW}{4} + W_2 + W_3 \tag{4.1}$$

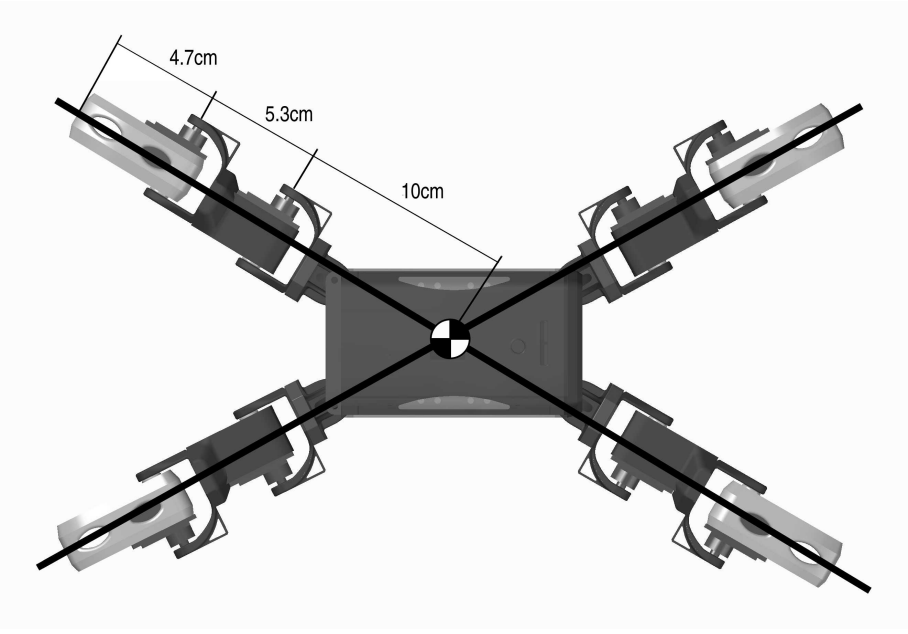


Figura 4.4: Modelos de algunas de las configuraciones de las patas.

A continuación, se equilibran los momentos suponiendo los dos motores actuando a su valor de par nominal,

$$\frac{\Gamma_2 + \Gamma_3}{g} = \frac{BW}{4} d_1 + W_2 d_3 - W_3 d_4 + R d_2 \tag{4.2}$$

Para obtener el peso de los elementos del brazo y teniendo en cuenta que al ser fabricados mediante impresión 3D, la masa puede variar, por lo que se adoptó un valor mayorado de 30 gramos que daba un considerable margen a las medidas.

Resolviendo la ecuación 4.2 obtenemos que el valor máximo para el cuerpo del robot es en torno a 850 gramos. Ambas configuraciones cuentan con pesos inferiores a esta cantidad, siendo para 2J de 560 g y para 3j de 670 g. Queda por tanto validado el diseño mecánico, que cuenta con un margen de carga considerable para ambas configuraciones, el cual se puede emplear para acoplar sensores externos, antenas o baterías extra.

Mientras se diseñaba el robot, también se tuvo presente la parte estética. Esto aportaba valor de cara a promocionar y anunciar el robot al mundo, además de hacerlo más amigable para tareas educativas.

Otra cuestión por mencionar en este apartado es el uso de la versión estudiante del software Solid Edge, que ha agilizado notablemente el proceso de diseño. En

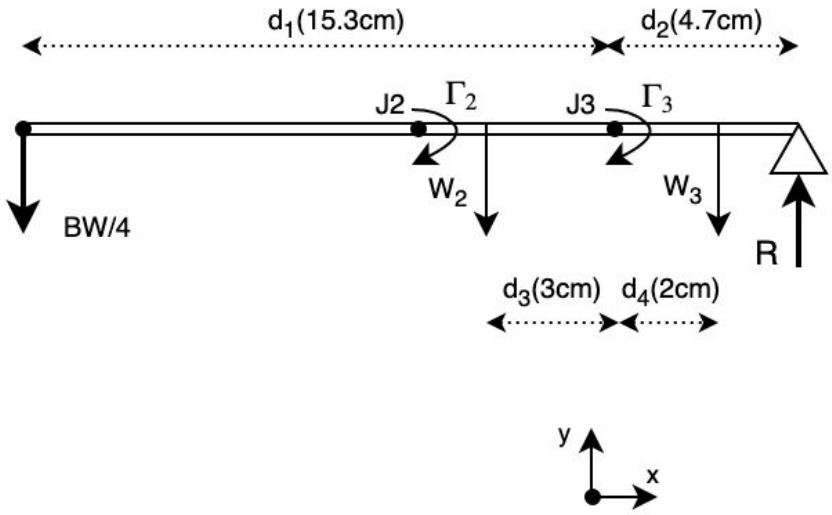


Figura 4.5: Diagrama de fuerzas aplicado a una pata de LoCoQuad 3J.

cuanto a la impresión 3D, se contaba con una Tronxy X3 para imprimir en PLA con diferentes calidades y acabados. En el Anexo C se muestran más imágenes del robot y también de algunos de los diseños preliminares que sirvieron para probar diferentes configuraciones, actuadores y soluciones.

Todos los diseños están disponibles de manera gratuita en `https://www.thingiverse.com/thing:3853722`.

Capítulo 5

Electrónica

En la cuestión electrónica se diferencian cuatro partes: la placa base, los actuadores, los sensores y el sistema de potencia. En la figura 5.1 se muestra el esquemático a alto nivel de las conexiones.

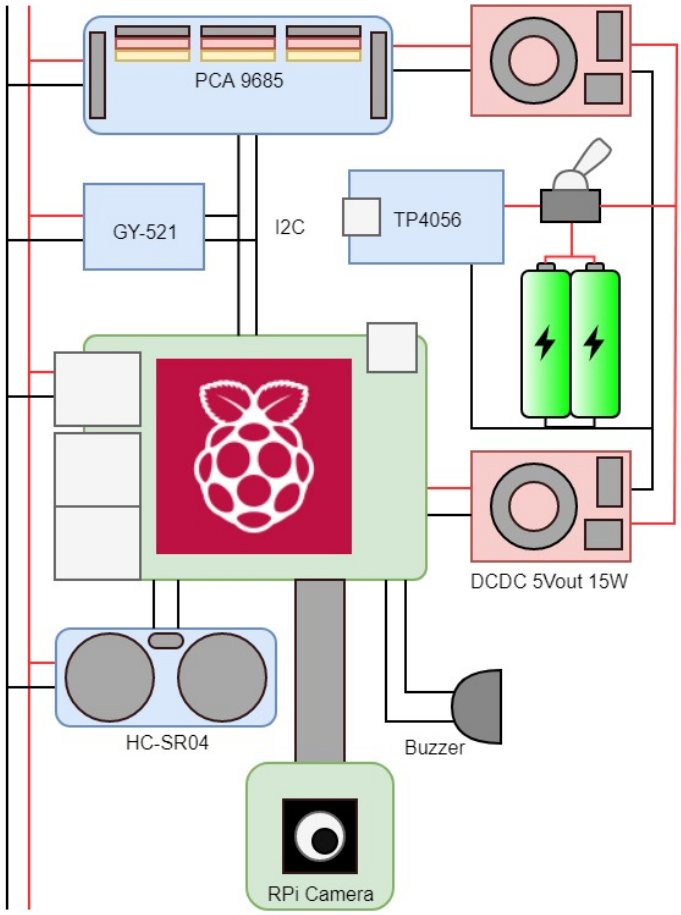


Figura 5.1: Esquema elecrónico de LoCoQuad

Para comprender mejor a LoCoQuad hay que entender su cerebro. Para esta tarea, se seleccionó la conocida placa de desarrollo de código abierto y bajo coste: Raspberry Pi 3 Model B. Este SBC (ordenador en placa simple) está muy extendida en la comunidad robótica, ya que supone dar el salto a un microprocesador sin dar el salto en precio. Tiene una gran flexibilidad a la hora de controlar periféricos, gracias a una serie de puertos específicos como el CSI para conectar un módulo de cámara, un hub de cuatro USBs, o el GPIO (entradas y salidad digitales de propósito general). Además, cuenta con chip de comunicaciones, lo cual lo hace ideal para utilizarlo via WiFi, mediante protocolo SSH como se verá más adelante. Su velocidad de computo es limitada (4cores/1.2Ghz), al igual que su memoria RAM (1Gb). Estas prestaciones inicialmente no suponen ninguna limitación para desarrollar este proyecto, pero se deben estudiar las limitaciones de éstas si se desea ampliar los requerimientos del robot. A nivel de conectividad, la Raspberry presenta un pequeño inconveniente, ya que a pesar de ser alimentada a 5V, y para reducir el consumo de la misma, su lógica es de 3.3V, por lo que habrá que tener esto en cuenta a la hora de conectar sensores o actuadores que operen a 5V.

En la sección anterior se mencionan los actuadores principales a partir de los cueales se construyó el robot. A nivel de control, los motores requieren alimentación y señal de control. Ambas necesidades son cubiertas mediante el uso de un driver I2C (PCA9685 by Adafruit) que se comunica con la placa base y permite actualizar los valores de 16 registros de 12 bits sobre los que se escribe el duty cycle de cada PWM que controla cada motor. Todos registros trabajan a la misma frecuencia lo cual no es un problema, ya que todos los actuadores son iguales. Además de estos motores que permiten el movimiento del robot en el mundo físico, se instaló un buzzer con el propósito de interactuar con el usuario y mandar indicaciones acústicas en caso de ser necesario. Se planteó también el añadir señales lumínicas, pero esto quedó descartado, ya que otros componentes integraban sus propias señales lumínicas, lo cual podría conducir a errores. El buzzer se conecta directamente al GPIO de la placa base.

En cuanto a los sensores, LoCoQuad incorpora una cámara RGB de 5Mpx. Conectada al puerto CSI de la Raspberry Pi, para evitar problemas de compatibilidades y para hacer el diseño más compacto que conectando una cámara USB se optó por el módulo de la misma compañía que produce la placa base (Raspberry Camera Module v1.3). Además, LoCoQuad incorpora un sensor de distancia ultrasónico (HC-SR04) este módulo se diseñó para placas de desarrollo

Arduino y su lógica es de 5V, por lo que se debe conectar un conversor de niveles lógicos o al menos un divisor resistivo bien calibrado para poder conectarlo al GPIO sin riesgo de dañar la placa base. También se instaló una IMU (unidad inercial) (GY-521) que se comunica por I2C con la Raspberry prolongando la conexión de ésta con el driver de los motores. La IMU opera con lógica 3.3V aunque se alimenta a 5V.

Para alimentar todos estos componentes hace falta un sistema de potencia. Las demandas de corriente de los componentes se detallan en la tabla 5.1. Inicialmente se diseñó LoCoQuad para albergar dos baterías LiPo 18650 de 3300 mAh en paralelo con un coeficiente de descarga 5C. Estas baterías proporcionan valores de tensión entre 3.7V y 4.2V. Estos valores son insuficientes para alimentar correctamente los componentes electrónicos que componen LoCoQuad, por ello se añadieron a la salida de la batería un toggle de selección entre dos modos: ON y CARGA. En ON, las baterías se conectan a dos conversores DCDC tipo Boost de 5.2V de tensión fija de salida y máximo 3A de corriente por conversor, lo que podría proporcionar una potencia de hasta 31.2W (15.6W por conversor). El hecho de que se decidiera utilizar dos boost en lugar de un único conversor tiene que ver con la disponibilidad y el coste de éstos. Además, esto permite separar la alimentación de la placa base del resto de componentes con gran consumo de corriente. En el modo CARGA, la batería se conecta a un módulo de carga monocelda (TP4065) que permite cargar la batería con una entrada de 5V y una salida de 1A.

Componente	Encendido	Operación normal
Raspberry Pi	hasta 3A (según periféricos)	$\sim 300mA$
Raspberry Pi	despreciable	250mA
12 Servos	$\sim 120mA$	1.4A hasta 3A en bloqueo
Resto de componentes	despreciable	despreciable
Total	hasta 3A	2A hasta 3.6A

Tabla 5.1: Consumos corriente por componente, en encendido y en operación normal

Usando el consumo máximo puntual como si fuese el nominal, y utilizando la configuración de baterías mencionada se calcula la duración de la batería del robot, con una capacidad de 6600mAh. Teniendo en cuenta la presencia de los conversores, y considerando una eficiencia baja (85 %), la corriente que éstos exigirían a la batería sería algo mayor de 4A. Gracias a la gran capacidad de las baterías de polímero de litio para proporcionar altos ratios de descarga, las baterías seleccionadas pueden suministrar hasta 33A (baterias 10C) de manera segura. Por lo tanto, la vida de la

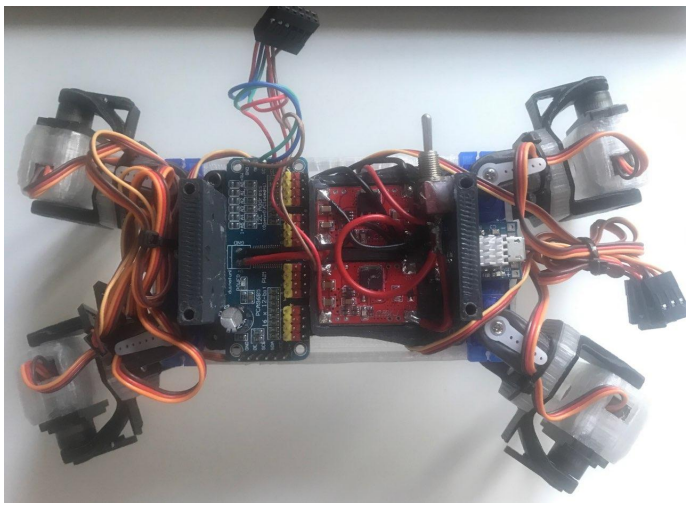


Figura 5.2: Imagen del interior del robot durante el ensamblaje.

batería sería 99 minutos. Esto en las peores condiciones y contando con un margen de seguridad. En la práctica, la batería dura considerablemente más, en el orden de horas, aunque esto depende mucho de la cantidad de movimientos y grabaciones que realice el robot. Hay que pensar que la Raspberry tiene un consumo de corriente bajo una vez que se ha encendido. Y los servos, pocas veces llegan a la posición de bloqueo y muchas menos llegan todos a la vez. Se dio por validada la solución para la electrónica de potencia, que además resultó ser realmente compacta y fácil de soldar y conectar. En la figura 5.2 se aprecia la disposición de alguno de los componentes dentro del nivel central del robot.

Capítulo 6

Programación

En esta sección se tratan todos lo temas relacionados con el sistema operativo, los algoritmos y la estructura del programa principal.

En primer lugar, habiendo elegido la Raspberry Pi como placa base, los sistemas operativos más atractivos resultaban ser Raspbian y Ubuntu Mate. Esta última no contaba con versión sin interfaz gráfica por lo que fue descartada, además, no permitía una conexión remota desde el primer inicio. Raspbian es una versión de Debian optimizada para funcionar en las Raspberries. Funciona básicamente como cualquier sistema Linux básico y al ser una versión sin interfaz gráfica se ahorraban ciclos de cómputo, operando únicamente con terminales. Para acceder a la Raspberry de manera remota se configuró una red WiFi y se activó el protocolo SSH para poder conectarse desde otros sistemas de la misma red.

Otra ventaja de Raspbian es la integración de Python, que viene preinstalado, lo cual simplifica el trabajo de instalación previo. Se decidió usar Python, ya que cubría todo lo necesario para un proyecto como LoCoQuad. Permitía usar OpenCV, interactuar con el GPIO de la placa base, programar el driver I2C y la adquisición de valores de los sensores; también permitía exportar el código a ROS (Robotic Operativ System) en caso de ser necesario. Y, por supuesto, es sencillo y asequible para los que comiencen a utilizar LoCoQuad sin saber de programación.

A nivel de programación, se centró el esfuerzo en proveer una serie de capacidades integradas en una estructura de clases que sirviera de base para el continuo desarrollo del robot. Sobre todo, para que no fuera un limitante a la hora de probar, integrar o instalar nuevas ideas en la plataforma. Para ello se optó por un paradigma

basado en objetos. Se estableció una clase principal donde se inicializa el robot y el resto de las clases a partir de ficheros de configuración y posteriormente se entra en una máquina de estados finitos. Esta estructura permite definir cuantos comportamientos se deseen, y cuantas variantes y alternativas sean necesarias para una tarea concreta. Inicialmente se estableció un ciclo de desplazamiento, reposo, captura de imagen, movimiento acrobático, reposo y, de nuevo, desplazamiento. Posteriormente se establecieron interacciones con el usuario, desde pitidos a modo de llanto para exigir que el usuario levantara al robot del suelo, que cesaban cuando el robot verificaba a través de los datos de la IMU que efectivamente le habían cogido del suelo, hasta protocolos de evitación de obstáculos en los que se utilizaba el sensor de distancia. Con la cámara se trabajó en capturar imágenes y videos, lo cual no suponen ningún problema, incluso mientras el robot se esta desplazando. Sí que es cierto que no se ha desarrollado una manera de hacer que los videos lleguen al usuario de manera satisfactoria. Aunque con más tiempo se trabajará en ello, al igual que incorporar redes neuronales entrenadas para detectar objetos de interés para el robot que alteren su ciclo de comportamiento básico y lo lleven a modos de exploración o incluso curiosidad.

Por último, se decidió integrar el código desarrollado en ROS. Esta decisión es estratégica, ya que en la comunidad investigadora, ROS se ha convertido en un estándar mundialmente usado y su integración en LoCoQuad supondría una gran ventaja a nivel de visualización del proyecto. ROS está limitado a sistemas Debian y Ubuntu, por lo que pudo ser instalado en la Raspberry. Se instaló la versión base, sin paquetes con interfaz gráfica para evitar saturar la Raspberry. Se ejecutó el código desarrollado anteriormente desde ROS sin modificarlo y se está trabajando en una versión de éste que saque partido a las ventajas que ofrece ROS gracias a su estructura de comunicaciones preconfigurada.

Todo el código desarrollado para este proyecto está disponible en `https://github.com/TomBlackroad/LoCoQuad`, al igual que los esquemáticos y las listas de materiales. Con todo esto, cualquiera podría fabricar su propio LoCoQuad y programarlo para propósitos personales, educativos o de investigación.

Capítulo 7

Validación

En cualquier producto, la etapa de validación o test resulta crucial. En esta etapa se comprueba que el trabajo realizado y que ha pasado por varias iteraciones tanto de diseño como de selección de componentes, montaje y posterior programación ha alcanzado los objetivos que se planteaban al comienzo del desarrollo.

Para LoCoQuad, este proceso consiste fundamentalmente en comprobar el correcto funcionamiento de sus subsistemas, tanto sensores, como actuadores, baterías, convertidores, sistema de carga y unidad de procesamiento. Además, el comportamiento ante los comandos, y la ejecución general del sistema también deben ser evaluados.

Empezando por los sensores, tanto la cámara como el sensor de ultrasonidos y la IMU, ofrecen datos en crudo, que la unidad de procesamiento es capaz de manejar sin problema. En concreto la cámara es capaz de grabar y almacenar vídeos a 60 fps, con una resolución de 640x480px sin interferir con la ejecución normal de la locomoción o la adquisición de otros datos.

Los actuadores responden sin problemas a las directrices del driver. Cabe destacar los problemas evidentes de calibración, que impiden conseguir posiciones completamente simétricas en el robot. Además, al construir sucesivos robots, éstos precisan de una máscara con los desvíos respecto de la posición de reposo (en cruz pegado al suelo). Eso es algo que se ajusta al comienzo, cuando se empieza a programar el robot. El buzzer de indicación también interactúa bien con el GPIO, aunque podría optimizarse con el uso de un microcontrolador esclavo.

Las baterías y los sistemas de potencia funcionan correctamente, ofreciendo

Concepto	**Unidades**	**Precio[$]**
Partes 3D	×1	10 (13)
Raspberry Pi 3 Model B	×1	35
Raspberry Pi Camera Module v2.1	×1	30
Tarjeta Micro SD 32GB	×1	10
Placa PCA9685 12 canales PWM I2C	×1	15
Servos MG90S 18Ncm par nominal	×8 (12)	24 (36)
Batería 18650 LiPo 3300mAh 20C	×2	6
Porta-baterías 2P 18650 LiPo	×1	1
Cargador LiPo TP4056 a 1A	×1	1
Sensor ultrasónico HC-SR04	×1	1
Unidad inercial GY-521 3Accel + 3Gyro	×1	4
Buzzer 5V	×1	1
Conversor boost DCDC 3.2-5V a 5V 15W	×2	6
Switch ON-OFF	×1	1
M3 M2.5 M2 kit tornillos	×1	5
TOTAL		**150 (165)**

Tabla 7.1: Coste pormenorizado de LoCoQuad 2J y (3J).

suficiente potencia en todo momento para que los actuadores puedan ser activados concurrentemente. La vida de la batería es mayor a lo estimado en los capítulos anteriores ya que no todos los servos se mueven a la vez, y la mayoría de las veces no tienen que sostener posiciones que los lleven a consumir una gran cantidad de corriente. Existe una potencial mejora para el sistema de carga, y es un indicador de carga, que se planteó durante el desarrollo de la estructura electrónica, pero posteriormente se descartó por no contar la placa base con un conversor analógico a digital.

En cuanto a la unidad de procesamiento, se temía que pudiera quedarse corta para procesar imágenes, y calcular y ejecutar la locomoción. Por el contrario, la carga de los núcleos no bloquea la ejecución de los programas y queda margen para exprimir su capacidad. Además, con nuevas versiones de Raspberry Pi, se gana en frecuencia de procesamiento y también en memoria RAM, aunque esto supondría un aumento del coste de la plataforma.

En relación al coste, se puede observar en la tabla 7.1 los precios de los componentes que conforman LoCoQuad y que efectivamente validan que el precio final de la plataforma es realmente bajo. Haciendo de la plataforma una opción muy asequible para la comunidad investigadora.

Por último, en relación con los algoritmos desarrollados y la estructura en máquina de estados finitos, ha sido todo un acierto; se identifica en cada momento el estado y por tanto se sabe qué debe hacer el robot en cada momento. La ejecución es cíclica, pero permite integrar motores de randomización para que el robot cambie de estado en función de una variable random, los inputs de los sensores y los estados anteriores. Esto haría al robot más interactivo, aunque esto no se planteó como un objetivo inicialmente, por lo que se simplificó la máquina de estados para verificar tanto los actuadores como los sensores, como se ha expuesto en el ejemplo del capítulo de programación.

Para hacer que esta validación y el proyecto en sí cobrara mayor interés se grabaron los diferentes movimientos que el robot tiene programados, además de comportamientos específicos como evitar obstáculos y elevarse sobre dos patas diagonalmente opuestas. En la figura 7.1 se muestra la secuencia de giro, y en la figura 7.2 la de equilibrio en dos patas. Todos los experimentos de validación se agruparon en un video de presentación disponible en `https://www.youtube.com/watch?v=MvRcbdmQJ7U`

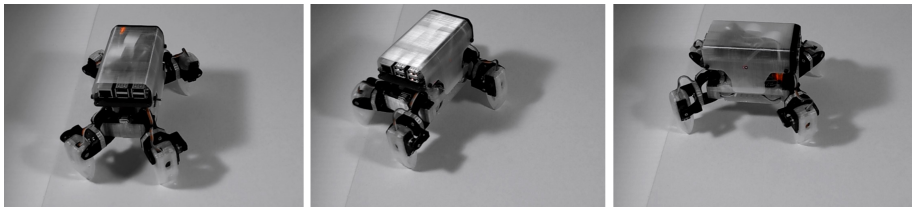


Figura 7.1: Secuencia del movimiento de giro

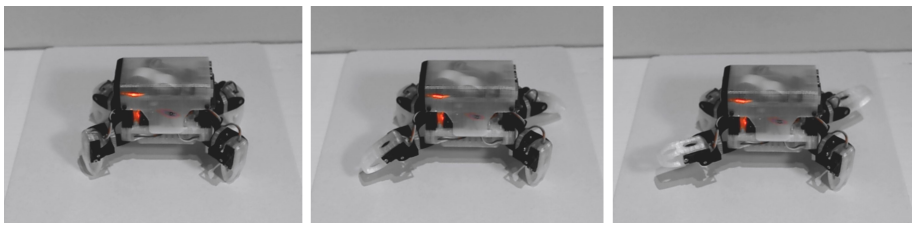


Figura 7.2: Secuencia del movimiento de equilibrio

Para concluir la validación, se comprobó que el robot era capaz de realizar las mismas acciones ejecutando el código en la Raspberry Pi con ROS Kinetic instalado. En este caso, el procesador trabajaba con mayor carga, pero no ocurrió ningún evento que denotara una bajada de rendimiento significativa.

Una vez verificados todos estos aspectos, se siguió utilizando el robot y optimizando el código hasta llegar a dar por concluido el proyecto.

Capítulo 8

Casuísticas

Desde el comienzo del proyecto han ido aconteciendo una serie de problemas, fallos, errores que precisan mención, debido a su relevancia para el desarrollo de LoCoQuad.

A continuación, se detallan aquellos eventos más relevantes:

En primer lugar, la selección de los actuadores supuso un gran reto. Como se ha mencionado anteriormente, y tal y como se expresa en el Anexo B, actualmente el cuello de botella de los robots de bajo coste son los actuadores de bajo coste. Elegir un actuador barato capaz de hacer todo lo que necesitaba el robot, no resultó fácil. Este problema se materializó en forma de varios prototipos que empleaban diferentes actuadores para comprobar su validez. Estos prototipos sirvieron para descartar Servos de menor coste, con trenes de engranajes de plástico en vez de metal. También ayudaron a descartar la opción de los motores paso a paso de 5V, que no eran capaces de realizar movimientos suficientemente rápidos, y además contaban con pares muy bajos en relación con su peso, por lo que exigían extremidades realmente cortas, que impedían su integración en los diseños.

Más adelante, una vez decididos la mayoría de los componentes, se planteó integrar un microcontrolador para realizar las tareas de adquisición y procesamiento de actuadores y sensores. Esto quedó descartado por motivos de compacidad y también en relación con la adquisición del driver multiservo empleado finalmente que solucionaba tanto la alimentación de los actuadores como la comunicación con los mismos. Los sensores podían ser leídos desde el GPIO de la placa base.

El problema más relevante, como se esperaba, fue la calibración del sistema.

Los servos tienen una repetibilidad bastante buena, pero sus orígenes varían notablemente de un motor a otro. Es por tanto difícil ensamblar dos patas con idénticos orígenes. El problema se solucionó aplicando una máscara a los comandos de movimiento, que tiene en cuenta los desfases iniciales respecto de la posición de reposo (en cruz pegado al suelo).

Posteriormente, la orientación de la cámara resultó ser una decisión complicada, ya que se planteaba una posición fija, como finalmente se usó. Esto suponía perder campo de visión y visualizar mucho suelo. Esto podría ser idóneo para una aplicación de siguelíneas, pero no resultaba conveniente para una versión de reconocimiento facial. Finalmente se optó por la posición fija ya que permitía un amplio abanico de aplicaciones, aunque no todas.

El proceso de ensamblaje resultó rápido, eficiente y a pesar de tener que soldar bastantes de los componentes manualmente, no dio ningún problema. Esto se debió a una buena técnica de iteración en el diseño mecánico.

Finalmente, durante la programación aparecieron los últimos problemas. La repetibilidad no era óptima, algo que se esperaba desde el primer momento al seleccionar los actuadores con gran backlash debido a su amplio tren de engranajes. Mecánicamente ya se sabía que un robot con dos ejes por pata no era capaz de desempeñar tareas precisas sin deslizamientos o impactos. Esto se ha mencionado en el capítulo anterior y se explica en el desarrollo mecánico. En cambio, la programación en Python aporta una gran flexibilidad para optimizar la respuesta del robot y su comportamiento final es adecuado, a pesar de ser algo brusco por los impactos que tienen lugar al cambiar el centro de gravedad de un plano de apoyo al siguiente.

Ninguno de los aspectos mencionados anteriormente ha impedido el correcto desarrollo del proyecto. Todo lo contrario. Se han confirmado ideas, posibles problemas y se han despejado dudas. Ayudando a vislumbrar el camino correcto por el que se debía desarrollar LoCoQuad.

Capítulo 9

Conclusiones

Recogiendo todo lo expuesto en esta memoria, LoCoQuad ha resultado ser todo un éxito, tanto por satisfacer los objetivos iniciales, como por la oportunidad de aprendizaje que ha supuesto. En general, la robótica es un campo demasiado extenso como para que una persona desarrolle toda una plataforma robótica. En este caso, al simplificar los problemas iniciales y tener claros los objetivos ha resultado interesante y fructífero el desempeñar las tareas en paralelo.

En relación con la plataforma, no hay duda de que queda mucho camino por delante, pero también hay que decir que tal y como ha quedado, con sus capacidades y sus limitaciones, es un producto de desarrollo listo para ser usado. Faltaría depurar el control, la locomoción y la teleoperación, pero estas tareas no han estado en ningún momento sobre la mesa, ya que supondrían un considerable incremento de la carga de trabajo.

Tras concluir el desarrollo de LoCoQuad, en su primera versión, se presentó a los Premios Jovenes Creadores Aragoneses, en los que obtuvo el primer puesto en la categoría de proyectos tecnológicos.

Con todo lo expuesto en esta memoria, se escribió un artículo que se envió a una conferencia internacional de robótica para su revisión. Además, este mismo artículo se archivó y está disponible en `https://arxiv.org/abs/2003.09025`. Este artículo dio lugar a una entrevista del blog de tecnología Tech Xplore que se puede encontrar en `https://techxplore.com/news/2020-04-locoquad-arachnoid-inspired-robot-purposes.html`.

Todos estos acontecimientos han supuesto un gran refuerzo y motivación para

seguir mejorando las ideas que han surgido durante el desarrollo de LoCoQuad y han supuesto un bonito broche a todo el esfuerzo puesto en el desarrollo de esta plataforma tan especial.

Como reflexión final, la robótica es un campo extenso, en desarrollo y que requiere de una gran implicación por parte de la comunidad investigadora, ya que, a diferencia de otras corrientes o ramas de la investigación, la dependencia de equipos multidisciplinares es altísima. En el futuro, la robótica va a tener un gran impacto en la sociedad y al igual que casi todos tenemos un router en casa, la evolución tecnológica hará que empecemos a tener también robots y otros sistemas inteligentes dispuestos a hacer nuestras vidas más cómodas y sencillas, asistiéndonos cuando lo necesitemos o lo deseemos. Para llegar hasta ese punto, y como ya hemos visto en la ciencia ficción, muchas plataformas y soluciones distintas deberán ser desarrolladas. LoCoQuad no es más que una más de esas soluciones que nos permiten avanzar hacia el futuro, aprendiendo y entendiendo los pasos que damos.

Capítulo 10

Proyección Temporal

A partir del momento en que LoCoQuad quedó ensamblado y podía ejecutar los comandos y los movimientos programados, se cumplió con el objetivo de este proyecto. Sin embargo, durante el proceso de desarrollo se detectaron campos de gran interés en los cuales LoCoQuad podría suponer un gran avance, con un coste de oportunidad realmente bajo, debido a sus características principales y sobre todo a su bajo coste. Entre estos campos, se encuentra la introducción a la programación y la robótica, campo que ya se exploró, al presentar LoCoQuad a los premios CREAR del Gobierno de Aragón, donde ganó el premio a mejor proyecto tecnológico.

Además, en el campo de la investigación, LoCoQuad promete ser una apuesta segura, tanto para estudiantes que se estén iniciando con la robótica, como para investigadores con líneas relacionadas con la robótica de bajo coste y el creciente campo de la inteligencia artificial. Para estos últimos, LoCoQuad podría suponer una manera más asequible de validar algoritmos que únicamente se prueban en simulaciones. Cabe destacar que resulta de gran interés contar con sistemas reales sobre los que poder testear y validar, antes o después de realizar simulaciones.

Como se ha mencionado a lo largo de esta memoria, LoCoQuad presenta una serie de características muy interesantes, que permitirán seguir desarrollando esta plataforma desde diferentes perspectivas y orientarla a diferentes campos. También se han mencionado algunas de las iteraciones y diseños preliminares que no llegaron a hacerse realidad pero que supusieron un gran aprendizaje para el desarrollo del proyecto. Estos diseños se han empezado a revisar para poder reenfocarlos a nuevas plataformas que aprovechen el trabajo realizado con LoCoQuad y aplicarlo a nuevas ideas, y así poder estudiar y entender mejor las limitaciones de este campo tan

apasionante como es la robótica de bajo coste.

Actualmente se ha empezado a trabajar en un proyecto multidisciplinar basado en sistemas de aprendizaje autónomo y visión por computador en el que LoCoQuad es la piedra angular. El objetivo es que sea capaz de aprender por sí mismo a ejecutar trayectorias y posteriormente optimizar este proceso. Para desarrollar este proyecto, LoCoQuad se situa dentro de una arena totalmente definida y delimitada sobre la que mediante códigos bidimensionales se puede obtener la posición del robot relativa a una referencia absoluta. Esto permite realimentar los algoritmos de aprendizage automático y la inteligencia artificial desarrollada para LoCoQuad. Este proyecto ya están dando resultados satisfactorios y tiene gran potencial para formar parte de una linea de investigación innovadora en robótica de bajo coste aplicada a testear la resolución de problemas con inteligencia artificial.

Gracias a continuar usando la plataforma tras la finalización de este proyecto se han detectado potenciales mejoras, adaptaciones y simplificaciones en la versión actual, y con la previsión de continuar usando LoCoQuad en el futuro, se ha comenzado a desarrollar una nueva versión. LoCoQuad promete tener un impacto positivo en el desarrollo robótico de plataformas de bajo coste.

Capítulo 11

Bibliografía

[1] Sara Lohmann, Jason Yosinski, Eric Gold, Jeff Clune, Jeremy Blum, and Hod Lipson. Aracna: An open-source quadruped platform for evolutionary robotics. In *Artificial Life Conference Proceedings 12*, pages 387–392. MIT Press, 2012.

[2] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[3] Ben Tribelhorn and Zachary Dodds. Evaluating the roomba: A low-cost, ubiquitous platform for robotics research and education. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1393–1399. IEEE, 2007.

[4] Sarah Bergbreiter and Kristofer SJ Pister. Cotsbots: An off-the-shelf platform for distributed robotics. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 2, pages 1632–1637. IEEE, 2003.

[5] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298. IEEE, 2012.

[6] Francesco Mondada, Edoardo Franzi, and Andre Guignard. The development of khepera. In *Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop*, pages 7–14, 1999.

[7] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jérôme Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre, and Bruno Maisonnier. Mechatronic design of nao humanoid. In *2009 IEEE International Conference on Robotics and Automation*, pages 769–774. IEEE, 2009.

[8] Victor Zykov, Andrew Chan, and Hod Lipson. Molecubes: An open-source modular robotics kit. In *IROS-2007 Self-Reconfigurable Robotics Workshop*, pages 3–6, 2007.

[9] Gabriel T Sibley, Mohammad H Rahimi, and Gaurav S Sukhatme. Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1143–1148. IEEE, 2002.

[10] Mohammad Ehsanul Karim, Séverin Lemaignan, and Francesco Mondada. A review: Can robots reshape k-12 stem education? In *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 1–8. IEEE, 2015.

[11] Mark Yim, David G Duff, and Kimon D Roufas. Polybot: a modular reconfigurable robot. In *ICRA*, pages 514–520, 2000.

[12] Elena Garcia, Maria Antonia Jimenez, Pablo Gonzalez De Santos, and Manuel Armada. The evolution of robotics research. *IEEE Robotics & Automation Magazine*, 14(1):90–103, 2007.

[13] S. Hirose, Y. Fukuda, K. Yoneda, A. Nagakubo, H. Tsukagoshi, K. Arikawa, G. Endo, T. Doi, and R. Hodoshima. Quadruped walking robots at tokyo institute of technology. *IEEE Robotics Automation Magazine*, 16(2):104–114, June 2009.

[14] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[15] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[16] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[17] Ning Tan, Rajesh Elara Mohan, and Karthikeyan Elangovan. A bio-inspired reconfigurable robot. In *Advances in Reconfigurable Mechanisms and Robots II*, pages 483–493. Springer, 2016.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[19] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[20] Alex Golovinsky, Mark Yim, Ying Zhang, Craig Eldershaw, and David Duff. Polybot and polykinetic/spl trade/system: a modular robotic platform for education. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 2, pages 1381–1386. IEEE, 2004.

[21] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822–10825, 2008.

[22] M Brett McMickell, Bill Goodwine, and Luis Antonio Montestruque. Micabot: A robotic platform for large-scale distributed robotics. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 1600–1605. IEEE, 2003.

[23] James McLurkin, Adam McMullen, Nick Robbins, Golnaz Habibi, Aaron Becker, Alvin Chou, Hao Li, Meagan John, Nnena Okeke, Joshua Rykowski, et al. A robot system design for low-cost multi-robot manipulation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 912–918. IEEE, 2014.

[24] Francisco M López-Rodríguez and Federico Cuesta. Andruino-a1: Low-cost educational mobile robot based on android and arduino. *Journal of Intelligent & Robotic Systems*, 81(1):63–76, 2016.

[25] S Piperidis, L Doitsidis, C Anastasopoulos, and NC Tsourveloudis. A low cost modular robot vehicle design for research and education. In *2007 Mediterranean Conference on Control & Automation*, pages 1–6. IEEE, 2007.

[26] Masahiro Fujita and Hiroaki Kitano. Development of an autonomous quadruped robot for robot entertainment. *Autonomous Robots*, 5(1):7–18, 1998.

[27] Akiya Kamimura, Satoshi Murata, Eiichi Yoshida, Haruhisa Kurokawa, Kohji Tomita, and Shigeru Kokaji. Self-reconfigurable modular robot-experiments on reconfiguration and locomotion. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 606–612. IEEE, 2001.

[28] Dayal C Kar. Design of statically stable walking robot: a review. *Journal of Robotic Systems*, 20(11):671–686, 2003.

[29] Ori Kedar, Christie Capper, Yan-Song Chen, Zhaoyang Chen, Julia Di, Yonah Elzora, Lingjian Kong, Yuanxia Lee, Julian Oks1 Jorge Orbay, Fabian Stute, et al. Spyndra 1.0: An open-source proprioceptive robot for studies in machine self-awareness.

Lista de Figuras

Lista de Tablas

Anexos

Anexos A

Condiciones Iniciales

Al comienzo de este proyecto contábamos con una serie de conocimientos, una idea de a dónde queríamos llegar y la ilusión necesaria para emprender este viaje hacia el descubrimiento de nuevas formas de hacer robots.

Como se ha mencionado anteriormente, el objetivo principal ha sido siempre el diseño y desarrollo de una plataforma robótica. En los instantes iniciales, el propósito final de la misma no quedó definido al cien por cien ya que no contábamos con el conocimiento real del potencial de nuestras ideas. Ahora podemos decir que íbamos bien encaminados en algunas de las premisas iniciales. En primer lugar, el bajo coste y la filosofía Open-Source, que nos parecían la mejor manera de aportar a la comunidad robótica, bien sea por el alto coste de la mayoría de los robots a día de hoy o por la falta de plataformas abiertas que existe en el campo de la robótica. En segundo lugar, ofrecer una plataforma en la que poder probar soluciones desarrolladas por investigadores de todo el mundo, ya que esto implica la participación de la comunidad investigadora, permitiendo recibir feedback y potenciales mejoras para futuras versiones de nuestro robot LoCoQuad.

En relación con los conocimientos previos que nos permitieran afrontar esta tarea con garantías, podemos enumerar los más relevantes, como la experiencia necesaria para diseñar e imprimir en 3D piezas de casi cualquier complejidad. Además, contábamos con equipo para realizar dichas impresiones de manera fiable. Por el lado de la electrónica, se conocían los entornos Open-Source de Raspberry Pi y Arduino además de las tecnologías que respaldan cada una de ellas. A nivel de programación, se tenía experiencia en Python, C y C++ en el desarrollo de pequeños y medianos proyectos. Por último, a nivel de componentes como motores o baterías,

se habían utilizado con anterioridad en otros proyectos, por lo que se tenía una idea de las características básicas de los componentes más comunes necesarios en el desarrollo del proyecto.

Se contaba pues,, con unas condiciones idóneas para evitar atascos inesperados durante el desarrollo de la plataforma robótica.

En los siguientes capítulos se describirán los procesos de desarrollo y diseño desde los tres pilares fundamentales de cualquier plataforma robótica: la mecánica, la electrónica y la programación.

Anexos B

Actuadores para robótica de bajo coste

En el campo de la robótica el mayor inconveniente y la razón por la que no se está alcanzando una tasa de expansión mayor en las soluciones que se ofrecen al mercado global es la oferta de actuadores. Además, hay que tener en cuanta que es un campo en desarrollo y por lo tanto hay pocos especialistas formados en todas las especialidades que atañen al desarrollo robótico. Sin embargo, los actuadores son una barrera económica inicial muy grande que restringe mucho el número de soluciones robóticas capaces de sortearla. Se podría decir que en cuanto la oferta de actuadores de calidad para robótica aumente ocurrirán dos consecuencias directas. En primer lugar, el abaratamiento de los mismos; y, por otro lado, un incremento en la variedad de actuadores, que a día de hoy sigue siendo muy limitada y cara. Generalmente se producen de manera expresa para empresas grandes.

En el campo de la robótica de bajo coste este efecto no tiene un efecto tan agudo, ya que el propio presupuesto elimina gran cantidad de actuadores que individualmente pueden llegar a costar cientos de veces el valor de todo un desarrollo. En este apartado trataremos los principales actuadores que se plantearon utilizar en un primer momento en el desarrollo de LoCoQuad. Se tratarán sus ventajas y desventajas atendiendo al objetivo y los propósitos de este proyecto, por lo que no se entrará a describir en detalle el modo de operación de los mismos.

En primer lugar, y ya desde el comienzo del proyecto, se planteó el uso de servomotores. Estos son producidos a nivel mundial en gran variedad de formatos, precios y prestaciones. Esto permite tener un abanico de opciones potenciales muy

interesante.

Un servomotor de bajo coste se compone de un motor DC al que se le aplica algún tipo de reducción mecánica sobre su eje de salida y se controla la posición de la salida de dicha reducción mediante un microprocesador integrado dentro del encapsulado de dicho motor. Esto presenta algunas ventajas, como pares altos, relaciones par-peso elevadas o sencillez en el control. En cuanto a las desventajas, se debe destacar la dependencia del controlador para obtener buenos resultados y evitar al máximo los errores mecánicos y holguras que puedan aparecer. Además, el control de posición del servo implica una limitación del sistema a un rango concreto de posiciones, normalmente inferior a 360 grados. Si se desea eliminar dicha restricción el control pasa a realizarse en base a la velocidad, impidiendo realizar acciones con altos niveles de repetibilidad. En la figura B.1 se pueden contemplar dos de los modelos de servomotores más vendidos para aplicaciones de bajo coste y de reducido tamaño.

Otra alternativa de bajo coste son los motores paso a paso. Estos motores se han popularizado mucho en los últimos años debido al incremento de soluciones para impresión 3D y máquinas CAD que se han planteado. Su precio ha caido mucho y a día de hoy resulta una gran solución para aplicaciones de bajo para y mucha precisión. Éstas son precisamente sus grandes ventajas y sus grandes inconvenientes. En primer lugar, no son capaces de generar apenas par. Incluso con trenes de reducción altos los pares no pueden compararse con los de un servomotor. Además, sus velocidades de operación son bastante bajas, por lo que al aplicar

(a) SG90 1.5kg (b) DSServo 60kg

Figura B.1: Servos comerciales de distintos pares.

reducciones éstas tienden a ser demasiado lentas. Un punto que no va a favor ni en contra es sus soluciones de control. Éstas pasan por driver integrados de bajo coste o placas de control industriales que se salen del rango de precios para un proyecto con un presupuesto ajustado. El controlador no va integrado dentro del motor por lo que hay más flexibilidad para configurar el sistema. Por otro lado, y enlazando con las ventajas, los controladores permiten obtener divisiones de los pasos de referencia de estos motores, reduciendo el par pero mejorando significativamente las precisiones que pueden alcanzar, mucho mejores que las de un servo. En la figura B.2 se pueden ver los servos de baja tensión para robots pequeños y los motores Nema empleados para impresión 3D y aplicaciones robóticas pesadas y que precisan de mucha precisión para su correcta operación.

Una alternativa natural a los servos son los motores DC con reductoras que se controlan externamente mediante la incorporación de encoders al sistema. En la figura B.3 se muestran dos casos: abiertos y cerrados. Se ha llegado a alcanzar niveles de miniaturización muy elevados y esto permite integrar estos sistemas realimentados mucho más facilmente en proyectos pequeños y de bajo coste. El control depende principalmente de filtros para garantizar la fiabilidad de las mediciones y permitir un control preciso. Gracias a la comunidad desarrolladora existen soluciones de diversa complejidad para el desarrollador que se pueden emplear. Es por esto que estos motores tienen la ventaja de ser más configurables y permiten controles más robustos, incluso implementar soluciones cerradas dentro del propio proyecto específicas para control de los motores. Como desventaja hay que destacar el aumento de la complejidad, el aumento del coste en referencia al par obtenido y el mayor tiempo de desarrollo que implican estas soluciones.

(a) 28BYJ-48 (b) Nema 17

Figura B.2: Motores paso a paso comerciales de distintos tamaños.

(a) Motor DC micro

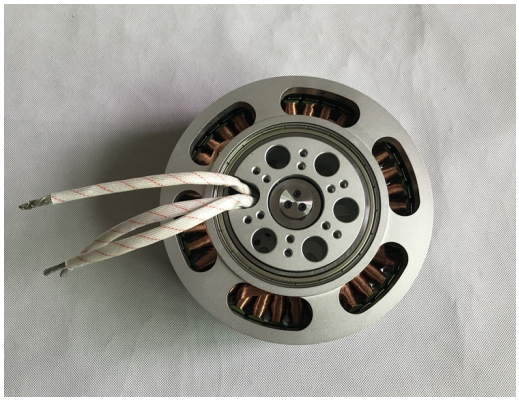
(b) Motor DC

Figura B.3: Motores DC de diferentes tamaños.

Por último, el avance de los drones en los últimos años ha implicado un gran avance en la miniaturización de los motores que más están creciendo en la industria robótica en el último lustro, los motores brushless. Estos motores cuentan con bobinados trifásicos que optimizan las prestaciones de los motores DC de escobillas. Se reduce el mantenimiento y se eliminan fricciones. En la figura B.4 se muestan algunos de los formatos más extendidos de motores brushless para drónica y para robots móviles.

Los motores brushless siguen mejorando a una velocidad elevada por lo que

(a) Motor brushless bala



(b) Motor brushless plato

Figura B.4: Motores DC de diferentes tamaños.

los precios han ido bajando al haber mayor oferta, aunque siguen siendo altos comparativamente. Su control es complejo comparativamente y suele ser necesario recurrir a soluciones de control que encarecen aún más el uso de estos motores. Son motores muy rápidos y con pares en relación a su peso muy elevados, es por esto que han triunfado en los drones, a pesar de requerir más tiempo de desarrollo.

Como conclusión, decir que los servos son la mejor solución para propósitos generales. Los motores paso a paso para proyectos de precisión. Los motores DC para controles más robustos que los de los servos y los motores brushless para proyectos que requieran más potencia y altas velocidades.

Anexos C

Desarrollos previos

Antes de poder diseñar LoCoQuad, se plantearon otros modelos de robot, cada cual con características específicas que permitieron validar y descartar soluciones, ya que los productos de bajo coste no daban garantías de operar según sus características teóricas. Por ello, se comenzó descartando posibles configuraciones, Empezamos con un cuadrúpedo de 8 ejes, que integraba servos dentro de sus patas. Los servos seleccionados eran menos potentes que los de LoCoQuad y las fricciones entre piezas 3D supusieron un gran problema a la hora de evitar que los motores se bloquearan. En la imagen C.1 se puede apreciar dicho diseño.

Posteriormente, se optó por integrar motores paso a paso de baja tensión en una estructura mecánica más compleja que integraba reducciones 3:1 en las propias piezas impresas en 3D. Esta solución sería más optima si las reducciones se hicieran en configuración planetaria, lo cual reduciría considerablemente el gran número de

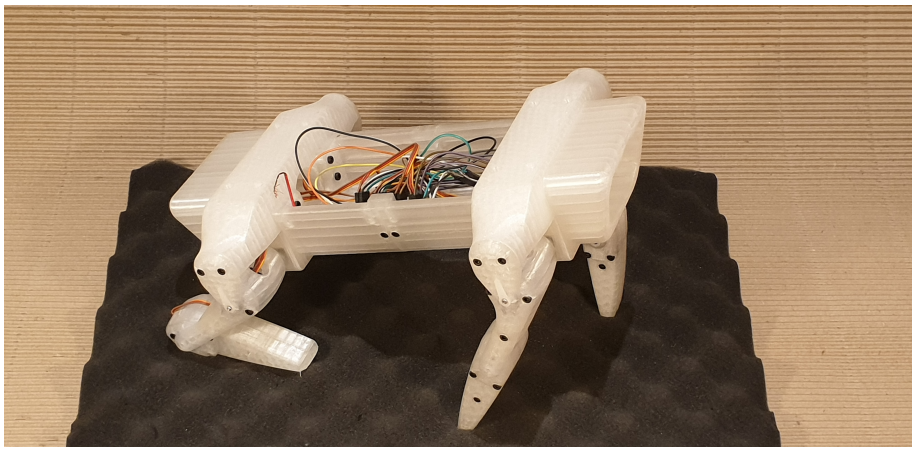


Figura C.1: Primer diseño para el proyecto LoCoQuad

saltos de diente que ocurrían en este prototipo. Además, y debido a la reducción aplicada para aumentar el par de los motores paso bajo, que de normal es bastante inferior a los de un servo, las velocidades que se podían obtener eran realmente bajas, por lo que la envergadura con la que se diseñó el robot hacía completamente inviable este diseño. En las figura C.2 se puede apreciar este prototipo.

Una vez llegados a este punto y como se mencionaba al comienzo de esta memoria, las configuraciones tipo mamífero tienden a ser más inestables y requieren actuadores muy rápidos que permitan realizar controles dinámicos. Es por esto que tras concluir que ninguno de los dos tipos de motores potencialmente viables, tanto por precio como por dimensiones nos permitían manejar las estructuras mecánicas de los robot diseñados se decidió comenzar de cero con robots arácnidos de cuatro patas. Aprovechando los conocimientos adquiridos durante el diseño, fabricación y validación, fallida, de los prototipos mencionados, se decidió descartar los motores paso a paso, y reducir el tamaño del robot al mínimo posible, haciendo las patas mucho más cortas para inferir palancas mucho más asequibles para los servos.

Fue así como se contruyó la primera versión operativa de LoCoQuad, que se muestra en la figura C.3.

Gracias a este desarrollo se detectaron algunos de los problemas que posteriormente y con un poco de trabajo adicional nos llevaron a crear el LoCoQuad que presentamos en este proyecto. En primer lugar, las patas acabadas en punta no ofrecían ninguna ventaja frente a patas acabadas en una arista redondeada. Por otro lado, la posición de la batería marcaba en gran medida el centro de gravedad del robot por lo que se decidió bajar lo máximo posible en la siguiente versión.

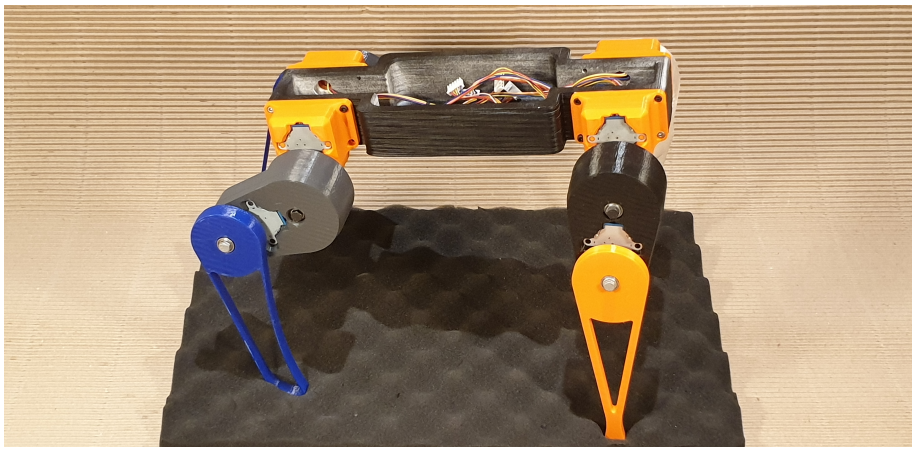


Figura C.2: Segundo diseño para el proyecto LoCoQuad

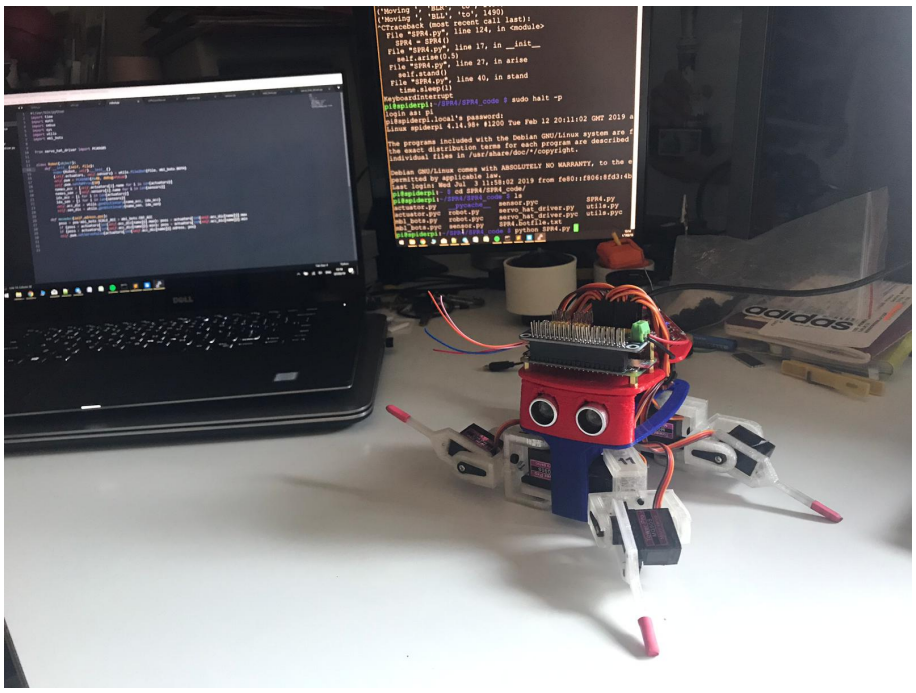


Figura C.3: Versión Beta de LoCoQuad

En cuanto a la electrónica, se comenzó con la versión mini de la Raspberry y posteriormente se pasó a la versión original. Se decidió conservar la apariencia de los ojos mediante el sensor ultrasónico, lo que marcó el diseño de la siguiente versión, que además permitía integrar toda la electrónica dentro del caparazón del robot.

No hizo falta mucho para llegar a la versión definitiva de LoCoQuad que se muestra en la figura C.4.

A partir de esta versión 2J, se amplió a la versión 3J, que se muestra en la figura C.5 en la que se añadió una articulación más a cada una de las patas.

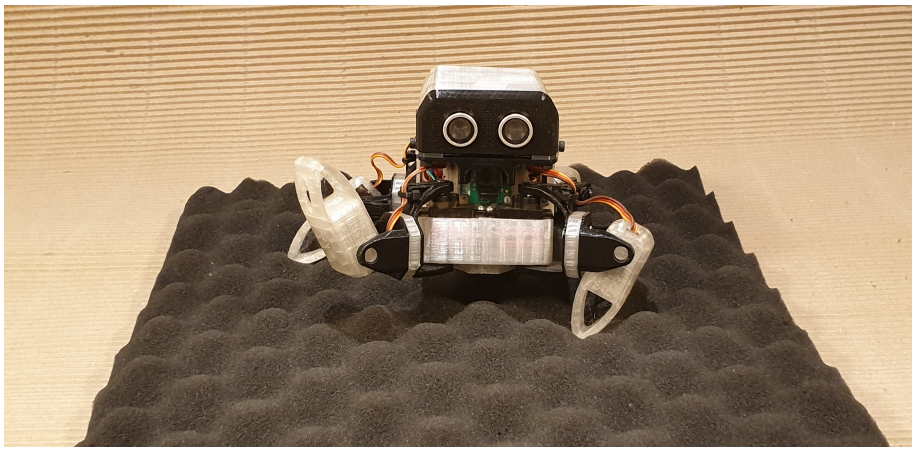


Figura C.4: Versión definitiva de LoCoQuad

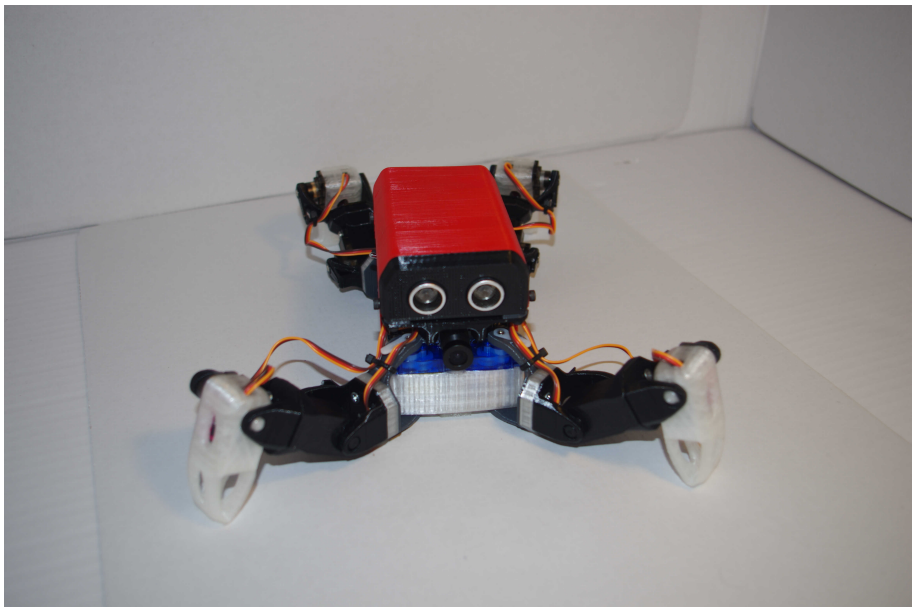


Figura C.5: Versión 3J de LoCoQuad

Figura C.6: Mini Turtle, plataforma de testeo software

Como trabajo posterior y en relación con la integrción de ROS se desarrolló una última plataforma de pruebas para la electrónica y algunas soluciones de software específicas. Este robot se muestra en la figura C.6. Consiste en una placa Orange Pi Zero, para validar su capacidad de gestionar ROS más cómodamente que la Raspberry Pi empleada en LoCoQuad. Además, cuenta con una placa de desarrollo Arduino Uno, para probar las comunicaciones entre procesador y micro. Y una cámara de 5Mpx que permite integrar soluciones con OpenCV en el robot. El sistema de alimentación utiliza los mismos componenetes que se utilizaron en LoCoQuad debido a que éstos ya habían sido validados.

Como conclusión a este apartado se debe hacer referencia a todo el trabajo previo y posterior que ha supuesto el desarrollo de la plataforma robótica LoCoQuad. Todo este conocimiento está sirviendo para mejorar la siguiente versión y llevar nuevas ideas a ver la luz.

Anexos D

ROS, Robotic Operative System

La integración de ROS en el proyecto de LoCoQuad supone un gran avance para la expansión de las capacidades de la plataforma. Esto no contaba como uno de los objetivos del proyecto pero resultaba muy interesante realizar una prueba en la plataforma una vez concluida.

Para instalar ROS en Raspbian, las versiones jugaron un papel fundamental y resultó más sencillo instalar ROS Kinetic en Raspbian Jessie Lite que en Raspbian Strech Lite. Se instaló y se probó su correcto funcionamiento mediante la conexión SSH desde la que se interactuaba con el robot en todo momento.

Las ventajas de ROS en plataformas robóticas son muchas, pero para plataformas de bajo coste con capacidades computacionales significativamente limitadas, la única versión que se encontró viable era sin interfaces gráficas, las cuales son uno de los pilares fundamentales de ROS. La estructura interna de tubería para comunicaciones supone la mayor diferencia a la hora de crear soluciones robóticas de cero. Sin embargo, al tener todo el código realizado antes de integrar ROS, esto no se utilizó, llevando simplemente la ejecución de un nodo que lanzaba a ejecución el main del software desarrollado en Python.

Antes de concluir este apartado hay que mencionar que durante las pruebas realizadas para validar el uso de ROS se probaron otras placas y programaciones, tal y como se mencionaba en el Anexo B, mediante una plataforma alternativa que nos permitió comparar los resultados obtenidos. Sin lugar a dudas, la integración en Armbian resultó mucho más sencilla e implicó muchos menos problemas. Esto se debió a que este sistema operativo cuenta con dos versiones, una de Debian y

otra de Ubuntu, resultando ésta la más cómoda a la hora de realizar las tareas de instalación, ejecución y personalización de paquetes específicos.

En definitiva, el empleo de ROS en una plataforma de bajo coste condiciona la electrónica pero establece una referencia de capacidades mucho más elevada, entre otras cosas por la posibilidad de entrelazar software en C++ y Python.

En la nueva versión de LoCoQuad, ROS ha estado presente en el desarrollo de la misma desde el primer momento, ya que se pretenden integrar soluciones de RL y visión que requieren de la agilidad de ROS para comunicar los sensores y los actuadores entre sí.