



# VNiVERSiDAD D SALAMANCA

---

CAMPUS DE EXCELENCIA INTERNACIONAL

Facultad de Ciencias

Grado en Ingeniería Informática

Comedero automático para mascotas.

Autor:

**Alvaro Torijano Garcia**

Tutor:

**Fernando de la prieta Pintado**



## **Resumen**

En esta memoria muestra el proceso de creación de un dispositivo que permita alimentar a una mascota automáticamente teniendo en cuenta la seguridad del dispositivo y un alta fiabilidad. Este proyecto ha sido diseñado para permitirle al usuario la máxima capacidad de control sobre el dispositivo a la vez que permite a otros desarrolladores integrarlo en sus propias soluciones. Se detalla el proceso de diseño, elección de hardware, planificación e implementación. Este proyecto abarca desde el nivel físico para el diseño y fabricación de las piezas, hasta el diseño y posterior construcción de una aplicación móvil, pasando por la electrónica y el firmware. El proyecto ademas ha sido dotado con una beca TCUE para prototipos orientados a mercado, por lo que este proyecto cuenta ademas con un plan de empresa completo como valor añadido que puede ser consultado en el Anexo 1.

**Abstract** This record shows the creation process for a device that allows autonomous pet feeding keeping in mind the security of the device and high reliability. This project has been designed allowing the user a maximum control capacity over the device at the same time that allows other developers to integrate it into their own solutions. It is detailed the design process, hardware selection, planning, and implementation. The project comprises the hardware level for the design and production of physical parts up to a mobile app crossing through electronics and firmware. This project has also been granted with a TCUE market-oriented prototypes scholarship so in addition, it counts with a complete company planning as an added value that is appended in Anexo 1.<sup>º</sup>

# Índice

<b>1. Introducción</b>	<b>6</b>
1.1. Propuesta . . . . .	6
1.2. Motivación y justificación . . . . .	6
1.3. Descripción del Problema . . . . .	7
1.3.1. Dosificación . . . . .	7
1.3.2. Vida media útil de componentes mecánicos . . . . .	8
1.3.3. Autonomía . . . . .	8
1.3.4. Desarrollo del sistema electrónico . . . . .	8
1.3.5. Posibilidad de configuración . . . . .	9
1.3.6. Algoritmia de los actuadores . . . . .	9
1.4. Objetivos . . . . .	10
1.4.1. Dosificación . . . . .	10
1.4.2. Durabilidad . . . . .	10
1.5. Autonomía . . . . .	10
1.6. Configuración . . . . .	10
1.7. Integración . . . . .	10
1.8. Eficiencia energetica . . . . .	11
1.9. Usabilidad . . . . .	11
1.10. Análisis del estado del arte . . . . .	11
1.10.1. Modelos de comederos disponibles . . . . .	11
IPv6 Cat Feeder . . . . .	11
R 2611 Pet Feeder . . . . .	14
Doug Costlow'w Pet Feeder . . . . .	16
1.10.2. Microcontroladores . . . . .	16
PIC . . . . .	17
Texas Instruments CC . . . . .	18
Raspberri Pi . . . . .	19
Arduino . . . . .	20
1.10.3. Reloj de tiempo real (RTC) . . . . .	21
DS1307 . . . . .	22
PCF2129 . . . . .	23
DS3231 . . . . .	24
1.10.4. Modulo inalambrico . . . . .	25
CC3000 . . . . .	25
NRF24L01 . . . . .	26
HC-06/HC-05 . . . . .	26
1.10.5. Motor . . . . .	27
Motor con escobillas . . . . .	28

Servo . . . . .	29
Brushless . . . . .	29
Paso a paso . . . . .	30
1.11. Pantalla . . . . .	30
7 Segmentos . . . . .	31
LCD 16x2 . . . . .	31
ILI9481 . . . . .	32
<b>2. Diseño</b>	<b>36</b>
<b>3. Planificación</b>	<b>39</b>
<b>4. Técnica y herramientas</b>	<b>39</b>
4.1. Dosificación y fiabilidad mecánica . . . . .	39
4.2. Configuracion . . . . .	45
4.3. Aplicacion android . . . . .	46
<b>5. Aspectos relevantes</b>	<b>50</b>
5.1. Multihilo . . . . .	50
5.2. Herramientas de documentacion . . . . .	51
5.2.1. Doxygen . . . . .	51
5.2.2. fritzing . . . . .	51
5.3. Diseño e impresion 3d . . . . .	52
5.4. Modelo de explotacion segun la TCUE . . . . .	54
<b>6. Conclusiones y Resultados</b>	<b>55</b>
<b>7. Trabajos Futuros</b>	<b>56</b>
7.1. Plugin para domoticz . . . . .	56
7.2. Multiplicidad diaria . . . . .	56
7.3. Deteccion de comida . . . . .	56
7.4. Configuracion via internet . . . . .	56
7.5. Camara . . . . .	56
7.6. Refactorizacion fisica . . . . .	57
7.7. Generacion de logs . . . . .	57
7.8. Nivel de alimento . . . . .	57
7.9. Batería . . . . .	57
7.10. Multiples mascotas . . . . .	57

# **1. Introducción**

## **1.1. Propuesta**

Este proyecto busca producir un dispositivo que permita alimentar correctamente a una mascota en periodos de ausencia del cuidador o en su presencia liberándolo de la tarea de proporcionar comida periódicamente a un animal de compañía.

El dispositivo generado tiene que ser versátil, ofrecer un buen control sobre la rutina de alimentación seriable, seguro y ofrecer un valor añadido a las alternativas comerciales.

También es deseable que debido a que se trata de un proyecto IOT, este se pueda integrar en soluciones de terceros como por ejemplo sistemas domóticos.

Es necesario que el dispositivo siga unas reglas endocrinas que aseguren que la salud del animal se va a mantener en su punto óptimo en lo referente a los hábitos alimenticios y en caso de no ser así o detectarse alguna irregularidad, notificarlo al usuario.

## **1.2. Motivación y justificación**

Este proyecto nace de la necesidad de mantener una mascota correctamente alimentada durante períodos relativamente cortos de tiempo. Cuando una persona necesita por cualquier razón ausentarse de su domicilio y no pueden hacerlo con su mascota también y es necesario encontrar una forma de mantener los cuidados que su mascota necesita. De estos cuidados el más relevante es la alimentación, y a diferencia del agua, que es fácil mantenerla disponible siempre y que ningún animal consume agua de forma descontrolada, la comida debe ser racionada, y además se degrada con mayor facilidad.

Según los datos que publica la FEDIAF (European Pet Food Industry)[5] un 39 % de los hogares españoles cuentan con al menos un perro o un gato como mascota, lo que representa un total de 18 millones de habitantes distribuidos en algo más de 7 millones de hogares según el INE (instituto nacional de estadística) [9]. Esta información se amplia en el anexo 1.

Dada la cantidad de personas que tienen una mascota en su hogar, se hace necesaria una solución que permita alimentar a nuestra mascota durante los períodos de ausencia del cuidador. Dada la necesidad de dispensar el alimento en raciones o dosis se hace evidente la obligación de proveer un mecanismo autónomo que ponga a disposición del animal la cantidad de comida pertinente.

El hecho de disponer de un dispositivo que sea capaz de mantener una

rutina de alimentación saludable para una mascota, permite a los propietarios tener mayor independencia y flexibilizar su rutina diaria, tanto para viajar, como para disponer del tiempo que antes necesitaban invertir en controlar los hábitos alimenticios de su animal.

Una explicación un tanto mas distendida pero amplia, y que aborda el problema desde otras perspectivas puede ser encontrada en el archivo “Autosullivan (candidatura TCUE).pdf” adjunto en el entregable digital de este proyecto.

### **1.3. Descripción del Problema**

En este apartado se analizan los problemas básico que plantea la construcción de un dispositivo de estas características, así como del problema que se necesita resolver. Para eso, el problema se ha seccionado en los apartados de racionamiento, pues es necesario dosificar la comida, vida media útil de los dispositivos mecánicos al hacer la elección del comedero que sera automatizado o algún modelo base a partir del que construir este prototipo. También es necesario analizar la fiabilidad de cada uno de los componentes (mecánico, firmware, y software).

Dado que es un dispositivo que ha de funcionar sin atención humana, es necesario también evaluar la autonomía como un problema. Debido a la diversidad de individuos o preferencias de los cuidadores, es necesario analizar la problemática que plantea la diversidad de casos en el apartado configuración y con esto, el método de comunicación para poder realizar esas tareas.

#### **1.3.1. Dosificación**

Tener la capacidad de racionar la comida que necesita comer un animal de compañía es de fundamental importancia. A pesar de que algunas mascotas son capaces de decidir la cantidad de comida y el momento en el que necesitan alimentarse de una forma coherente y segura para su salud, muchas otras mascotas comen hasta vaciar el comedero, independientemente de la cantidad de comida de la que dispongan. Este problema llega hasta tal punto, que en algunos ejemplares es incluso necesario limitar la velocidad a la que ingieren el alimento utilizando lo que se llaman "cuencos anti atracones" que consiguen frenar al animal al permitir que la comida caiga entre unos pequeños obstáculos que obligan al animal tener que recuperarla para poder ingerirla, evitando así que se atraganten o tengan problemas digestivos.

### **1.3.2. Vida media útil de componentes mecánicos**

Todo aparato que incorpora partes móviles está expuesto al rozamiento y consecuente el desgaste de sus piezas. Esto deriva en un tiempo medio entre fallos que es difícil mantener contenido entre unos límites aceptables. Por otra parte debido a las holguras necesarias para un correcto funcionamiento y a la calidad del proceso de fabricación, todo conjunto de piezas que formen parte de un mecanismo es susceptible de atascarse o encasquillarse en movimiento. Evitar todas estas situaciones de sobra conocidas es un que de hecho es tenido en cuenta a la hora de adoptar una alternativa u otra en el mercado.

A pesar de que este proyecto está únicamente enfocado en la automatización del comedero, es imprescindible elegir un comedero funcional en primer lugar, que ya ofrezca un mínimo estándar de calidad. Por estas es una razones determinante a la hora de escoger un mecanismo que resulte fiable a largo plazo y que minimice el mantenimiento.

### **1.3.3. Autonomía**

El almacenamiento de materia orgánica destinada al consumo como alimento supone garantizar que no estará expuesta a condiciones climáticas que puedan derivar en un deterioro de la calidad por la acción de distintos organismos (bacterias y hongos) que degraden el alimento, la contaminación por agentes externos o simplemente un deterioro mecánico de estos elementos.

Por estas razones es importante tener en cuenta que la comida que el comedero almacena debe estar protegida del aire y los rayos solares para evitar que se deteriore o que su calidad no sea óptima permitiendo que solo quede en contacto con la atmósfera la comida que se va a dispensar de forma inmediata.

### **1.3.4. Desarrollo del sistema electrónico**

Es necesario que el comedero posea un sistema electrónico que le permita tomar decisiones en base al entorno. Para esto el sistema debe poseer una serie de sensores que permitan medir las condiciones del entorno para estimar la calidad del alimento y su grado de degradación, y prevenir de esta manera una posible intoxicación del consumidor, en este caso la mascota.

También es necesario que el comedero cuente con actuadores que le permitan accionar la tolva de manera suficientemente precisa para graduar la cantidad de comida que necesita consumir la mascota. Estos actuadores deben mantener la fiabilidad necesaria que garantice un periodo de vida mínimo usándose diariamente.

El desgaste de los componentes electrónicos también puede ser un problema en el desarrollo de proyecto. Debido a la cantidad de corriente que tenga que manejar, el numero de horas de funcionamiento al día y la cantidad de tareas que tenga que realizar en este tiempo, así como la calidad de los componentes con los que este fabricada, es susceptible a un desgaste prematuro que conlleve a un mal funcionamiento, bien sea por sobrecalentamiento, por deterioro de sus componentes, o por el tiempo que estos componentes pueden estar realizando la misma tarea hasta agotar su periodo de vida útil. Es por tanto muy importante gestionar correctamente el funcionamiento de cada componente con el fin de maximizar su periodo de vida útil.

#### **1.3.5. Posibilidad de configuración**

Es necesario que el usuario pueda modificar el comportamiento del dispositivo debido a que cada mascota tiene distintas necesidades. Por esta razón es necesario que el comedero se pueda configurar y que ademas el usuario pueda también recibir información sobre que parámetros debe configurar, y como afectaran estos al comportamiento del dispositivo.

Debido a que una configuración manual puede ser engorrosa para el usuario final es necesario que la configuración sea una interfaz a través de una aplicación de móvil.

#### **1.3.6. Algoritmia de los actuadores**

Para poder controlar los sistemas electrónicos, es necesario desarrollar los algoritmos de control que permitan calcular la cantidad de comida, y estimar las acciones necesaria para dispensar dicha cantidad.

Es habitual que los algoritmos de interacción entre una maquina y seres vivos no contemplen todos los posibles casos que pueden producirse o que no se comporten correctamente en cada uno de ellos, derivando en comportamientos inesperados, o en la parada completa de la ejecución del programa. La disminución de estos fallos lleva asociado un aumento en el esfuerzo de programación, y en la complejidad del software. Es problemático desarrollar una solución de software que funcione correctamente cuando la cantidad de dispositivos y el número de versiones del sistema operativo es muy alta. Además, dado que este tipo de errores llegan hasta el usuario final y son claramente visibles, harían decrecer rápidamente la valoración que un usuario tiene del producto.

## **1.4. Objetivos**

En esta sección se detallan los objetivos que tienen que cumplir el proyecto y que fueron planteados en la fase inicial.

### **1.4.1. Dosificación**

Es necesario poder controlar la cantidad de comida que recibe el animal, ya no solo para garantizar que ingiere lo necesario para mantenerse saludable, si no tambien para evitar que algunos animales puedan comer en exceso.

### **1.4.2. Durabilidad**

Dado que el comedero va a tener que estar funcionando sin apagarse durante mucho tiempo, es necesario en primer lugar escoger un mecanismo que tenga el menor numero posible de piezas, y dimensionar correctamente los componentes electrónicos evitando que trabajen cerca de su limite, o de hacerlo reduciendo al mínimo el tiempo que los componentes están sometidos a estrés.

## **1.5. Autonomía**

Es una característica deseable que el dispositivo pueda alimentar durante el máximo tiempo posible al animal, por tanto debe mantener un reservorio de alimento que garantice un periodo de al menos dos semanas. Tambien es necesario garantizar que tanto el hardware como el software son capaces de funcionar esa cantidad de tiempo sin sufrir un desgaste acusado que acorte la vida del dispositivo.

## **1.6. Configuración**

La posibilidad de configurar el dispositivo es una necesidad en este proyecto para permitir que su comportamiento sea adaptable sin necesidad de modificar el codigo, y a ser posible que se haga de forma sencilla para el usuario, evitando que tenga que adquirir conocimientos relacionados por ejemplo con la programacion.

## **1.7. Integración**

Este proyecto se diseña tanto para poder ser usado tal se provee, como para poder ser integrado en otros proyectos o soluciones de terceros que impliquen hardware, software o ambas. Es necesario proveer un mecanismo

que permita operar con el comedero sin que sea necesario modificarlo. Es por tanto necesario un protocolo o una API dado que la interfaz grafica queda descartada al no ser posible utilizarla por una maquina o no de forma optima.

## 1.8. Eficiencia energetica

Es una característica deseable que el dispositivo haga un uso optimo de la energía apagando el hardware que no se utiliza, y permitiendo así que pueda ser alimentado desde fuentes de energía cuya capacidad de suministro sea limitada.

## 1.9. Usabilidad

Debido que el dispositivo también puede ser utilizado por personas, es necesario proveer un mecanismo para mostrar información y comunicarse con el dispositivo de forma sencilla. Para esto se necesita un dispositivo de interacción humana

## 1.10. Análisis del estado del arte

En esta sección se ha hecho un análisis de las soluciones disponibles que ofrece la comunidad, el hardware disponible y la cantidad y calidad del software compatible con cada una de las plataformas así como el estado de implantación en el mercado o el tamaño de su comunidad. Este estudio permitirá reducir el esfuerzo en la fase de desarrollo.

### 1.10.1. Modelos de comederos disponibles

A continuación se presentara la evaluación los proyectos que la comunidad ofrece con el fin de evaluar si es posible reutilizar algunas de las soluciones que ya se han dado a este problema, así como evaluar que posibles abordajes existen en este ámbito. Para ello en esta sección se describirán todos los prototipos encontrados y un breve análisis de ellos:

**IPv6 Cat Feeder** El IPv6 Cat Feeder[12] utiliza un diseño de tolva con un dosificador de goma de aletas que interrumpe el flujo del grano actuando como válvula de paso hasta la desembocadura del dosificador como puede verse en la figura 1 que ilustra su construcción. Para mover este mecanismo utiliza un motor de corriente continua con una reductora que es capaz de proporcionar 25Kg/m @4RPM. Para controlar dicho motor utiliza un router WRT54G de código abierto con el que puede controlar el estado de los indicadores

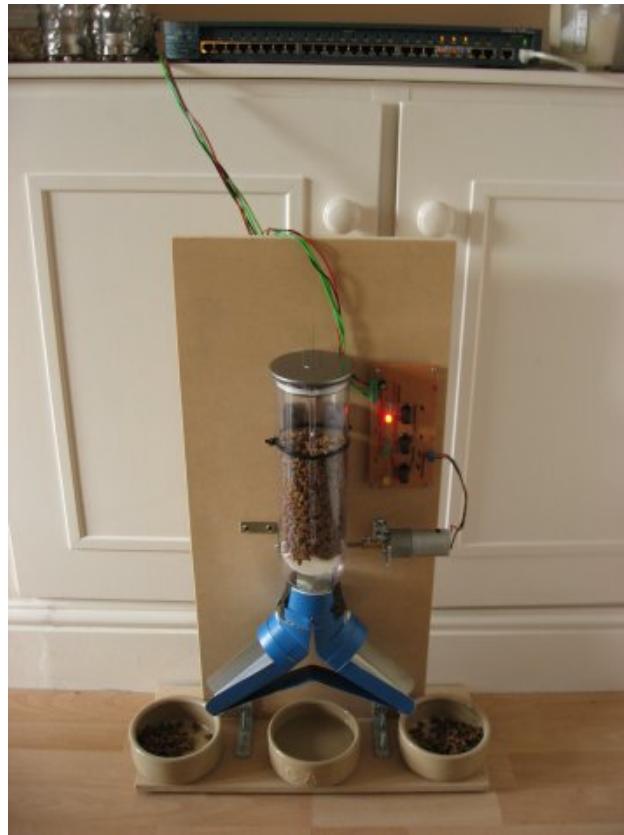


Figura 1: IPv6 Cat feeder

luminosos y así cerrar un relé para encender el motor como puede verse en el esquema electrónico de la siguiente diapositiva.

En el esquema mostrado en la figura 2 se puede ver que el diseño utiliza un primer relé (superior) para abrir o cerrar el suministro eléctrico al motor, un segundo relé (centro) para hacer girar el motor en sentido normal. Y un tercer relé (inferior) para invertir el giro del motor y poder revertir un posible atasco. Este diseño presenta varios problemas:

**Alto costo económico** El motor de corriente continua con reductora tiene un coste económico alto situándose por encima de los 50€. Por otra parte no es posible saber la posición exacta del rotor sin métodos de control añadido, ni tampoco controlar la cantidad de movimiento del motor de una forma precisa. El autor aproxima una solución a este problema utilizando periodos de tiempo fijos en los que puede calcular el numero de revoluciones del motor, pero debido a que la comida solo se dispensa en algunos puntos de la circunferencia, es imposible saber con precisión cuanta comida se dispensa.

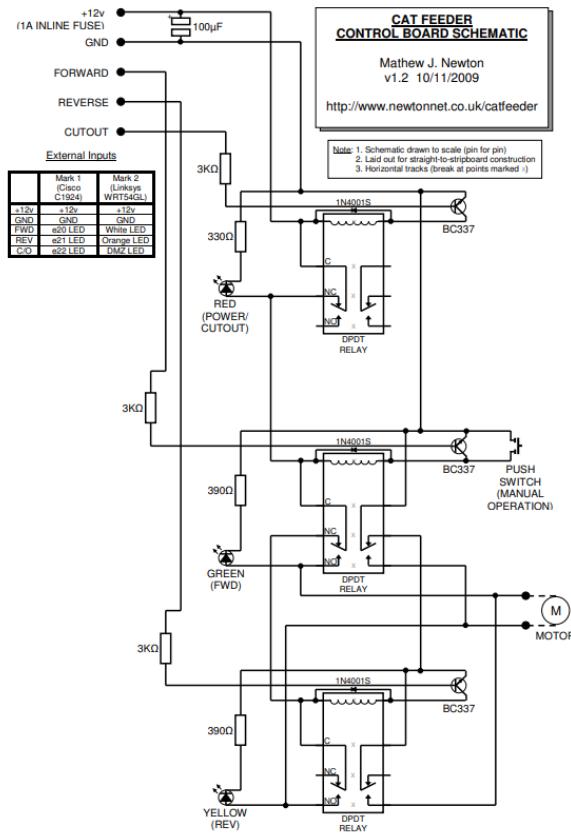


Figura 2: Esquema eléctrico para el control del motor en [12]

**Poca vida útil** El uso de relés tiene asociado un decremento en la vida útil al ser componentes con partes móviles, que además dado que el consumo de un motor se dispara en el arranque, pueden quedarse momentáneamente o permanentemente soldados.

**Elevado consumo eléctrico** Los relés a su vez tienen un consumo asociado alto al tratarse de un electroimán, y además tienen un coste económico elevado.

**Componentes no adaptables** Para el control del motor ha utilizado el router, que pese a ser software abierto no es hardware abierto y por tanto dificulta la adición de nuevos componentes hardware además de complicar la utilización de hardware presente para otros fines distintos a aquellos para los que fue diseñado. Es por tanto un problema a la hora de adaptarlo a las necesidades del presente proyecto.

**R 2611 Pet Feeder** El R 2611 Pet Feeder utiliza un deposito superior con una abertura inferior que sera manejada por un servomotor que hace las veces de obturador del dispensador como puede ser apreciado en la figura 3. El diseño de una compuerta de guillotina es claramente insuficiente para evitar los atascos, los granos de pienso pueden atascarse, evitar que cierre y al no disponer de fuerza suficiente no es capaz de romper los granos que generan el atasco. Por otra parte en este diseño solo es posible modificar el comportamiento reescribiendo el código. Tampoco permite interaccionar con el dispositivo modificando la funcionalidad. Como punto positivo cabe destacar que el micro controlador usado es un Arduino, por lo que modificar el código resulta sencillo para el programador.

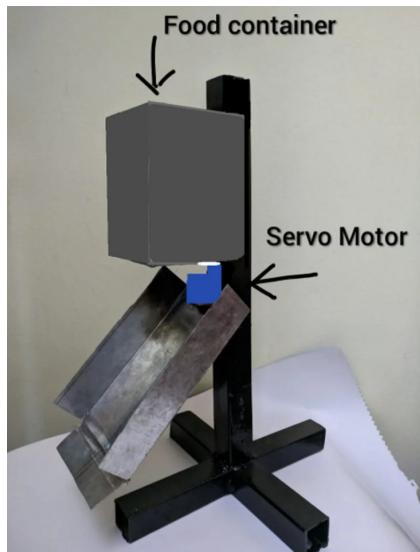


Figura 3: Diagrama de la estructura del comedero

En el esquema que proporciona se puede ver como la comida se coloca en el comedero a través de una rampa, que conduce el grano desde el deposito superior. Esta solución no es capaz de garantizar la posición final de cada grano.

**Yoran Pauw's Automatic Dog Feeder** Para solucionar el problema de dispensar comida el Yoran Pauw's Automatic Dog Feeder [2] utiliza un dispensador de aletas como en caso del IPv6 Cat Feeder al que le añade dos celdas de carga para controlar la cantidad de pienso restante en el deposito y la cantidad de pienso dispensado. Con esta solución el autor es capaz de controlar la cantidad de comida sin tener que conocer la posición del motor. Por otra parte se ha añadido al diseño un sensor laser con una resistencia

dependiente de la luz para controlar la presencia de una bola que la mascota tiene que utilizar para que se dispense la comida. A pesar de no ser necesario, en este caso se utiliza un motor paso a paso para controlar la cantidad de dosis que se dispensan. Como controlador, este modelo utiliza una Raspberri Pi y un conversor analógico-digital para poder hacer una lectura del sensor de luz. Las principales ventajas que esta versión ofrece, es el uso de hardware barato y ampliamente extendido, lo que permite a futuros desarrolladores adaptar el proyecto a sus necesidades. El uso de un dispensador de aletas sigue teniendo los problemas de ser difícil de encontrar por separado, al igual que posee los problemas de que puede atascarse similar al IPv6 Cat Feeder, pese a que en este caso no es necesario usar una reductora debido a que el motor paso a paso ya ofrece par suficiente.



Figura 4: Estructura del comedero diseñado por Yoran Pauw

En la figura 4 se puede ver la configuración de tolva y rampa utilizada junto a una segunda estructura que hace las veces de portería para adiestrar al animal. En la imagen puede verse también como la estructura incorpora un pequeño soporte que evita que el comedero se mueva, asegurando que la comida caiga siempre dentro aunque es imposible saber lo bien que se comporta en este aspecto dado que no se encontró ningún vídeo a priori en el que se vea el funcionamiento del mecanismo. El software para controlar el dispositivo se compone de un portal web que se puede ver en la figura 5 que ejecuta un script en python que hace girar el motor paso a paso. También incorpora una base de datos donde se guarda un histórico de la cantidad de comida en el comedero para poder saber el consumo que hace el animal en

cada momento como puede verse en la figura siguiente.



Figura 5: Aplicacion web del comedero diseñado por Yoran Pauw

**Doug Costlow's Pet Feeder** Este diseño utiliza la estructura de tolva con tornillo sin fin. En este caso todas las piezas están impresas en 3D y cortadas con control numérico. Los archivos necesarios para cada una de las piezas están disponibles de forma publica. En este caso se dispone un servo sin límite de giro en un extremo del tornillo sin fin que es controlado por un Arduino. Este diseño presenta varios problemas graves. Las tolvas necesitan la fuerza suficiente para poder romper el grano de comida, y un servo de prestaciones normales no es capaz de ofrecer el suficiente par. Por otro lado las tolvas son propensas a atascarse cuando todo el tornillo sin fin queda expuesto y el grano es demasiado grande o duro. En este caso solo existe un Arduino que controla a cada cuanto tiempo tiene que girar la tolva y cuantas veces. No hay una aplicación web, ni ninguna otra herramienta que permita al usuario configurar el comportamiento del dispositivo.

### 1.10.2. Microcontroladores

La elección del microcontrolador es la más relevante en un proyecto de automatización donde es necesario desarrollar un prototipo físico, pues sus características y prestaciones serán las que posteriormente permitan o no ampliar las funcionalidades, o determinen la complejidad para implementarlas. Como puntos clave en este punto se revisarán: 1. Nivel de penetración en el mercado 2. Tamaño de la comunidad 3. Lenguajes disponibles 4. simplicidad en el uso.

**PIC** Los micro-controladores PIC son una familia de procesadores RISC con arquitectura Harvard de 8, 16 y 32 (a partir del 2007) bits que se llevan utilizando durante los últimos 40 años para propósitos generales. Su arquitectura se caracteriza por contar con un solo acumulador, tener una pila hardware del sistema y permitir escribir en el contador de programa consiguiendo con esto ultimo saltos indirectos. Todas las posiciones de memoria se pueden utilizar como registros. Sus instrucciones duran típicamente 4 ciclos de reloj y es un procesador segmentado, lo que significa que en realidad puede terminar una instrucción en cada ciclo de reloj como puede verse en la figura 6 proporcionada por el fabricante del dispositivo [10]

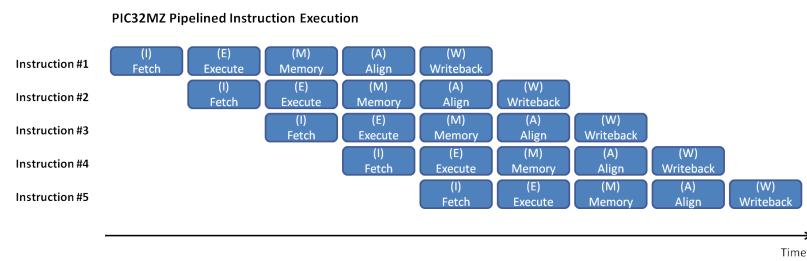


Figura 6: Pipeline de un procesador PIC

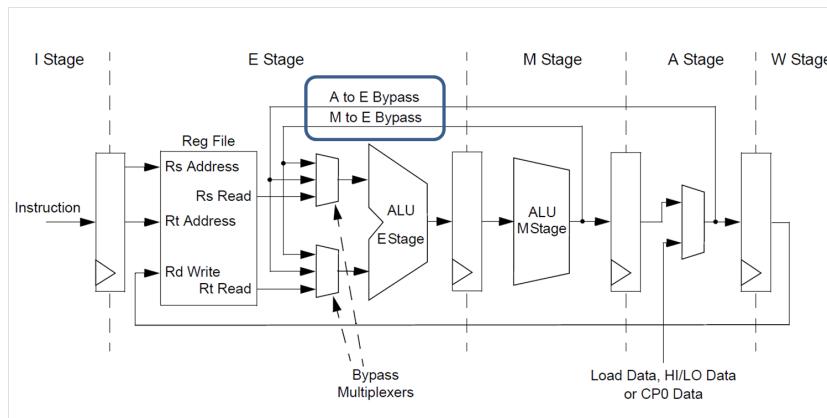


Figura 7: Arquitectura de un procesador PIC

En la figura 7 que se puede encontrar en la documentación técnica del fabricante [10] se pueden observar en detalle las fases de ejecución de este procesador en el que se distinguen las siguientes cinco etapas:

- I Decodificación de instrucciones. El procesador lee la instrucción y compara con su conjunto de instrucciones para saber de qué operación se trata.

- E** Ejecución. En esta etapa la ALU obtiene los datos de memoria y opera con ellos.
- M** Adquisición de memoria. La ALU termina y accede a memoria para almacenar o cargar datos
- A** Alineamiento. En esta etapa los datos son alineados o cargados de nuevo en la ALU para siguientes operaciones.
- W** Escritura. En esta etapa los datos son escritos en un registro o memoria principal.

Ofrecen conexiones USB, UART, I2C, I2S, SPI, CAN, PWM y ADC de hasta 12 bits. Existen modelos con hasta 256KB de ram funcionan típicamente a 5.5V y soportan frecuencias de reloj de hasta 80MHz. Sin embargo es difícil utilizar funciones de red y no cuentan con tecnologías inalámbricas.

Es posible programarlos en Basic, C/C++ ademas de sus correspondientes ensambladores. También es posible encontrar múltiples simuladores y depuradores y programadores. A pesar de su alta expansión hoy en día es difícil encontrar librerías para hardware moderno.

**Texas Instruments CC** La familia de microcontroladores CC de Texas Instruments[3] ofrece alta capacidad de calculo al incorporar núcleos ARM de distintas gamas y en lineas generales las principales ventajas y características serán explicadas a continuación.

Ofrecen conexiones USB, UART, I2C, I2S, SPI, CAN, PWM y ADC de hasta 12 bits y DAC de 8 bits, radio frecuencia sub-1GHz y 2.4GHz e implementan las pilas para protocolos wifi, bluetooth, Lora, ZigBee, SigFox y TIPacket. Incorporan ademas un microprocesador independiente para las funciones de radio. Estos procesadores cuentan ademas con watchdogs, reloj de tiempo real, temporizadores, interfaces para pines táctiles y tienen ademas capacidades criptográficas aceleradas por hardware para RSA1024, ECC512, AES256 y SHA512. Estos procesadores tienen ademas modos de bajo consumo situando la potencia disipada por debajo de los 100 micro amperios en periodos de reposo. Esto los convierte en procesadores notablemente potentes en campo del IOT, e incluso haciendo posible que trabajen con ciertas blockchain que usan criptografía de curva elíptica como es el caso de bitcoin. Sin embargo, pese a que son procesadores populares, no tienen una comunidad activa y es prácticamente imposible encontrar librerías para hardware moderno que sean compatibles con estos procesadores.

Por otra parte el fabricante provee un entorno de desarrollo [4], un kit de desarrollo de software y un sistema operativo de tiempo real, de los que

también es difícil encontrar ejemplos o documentación mas allá de la que el propio fabricante proporciona. Por esto ultimo se vuelven procesadores complejos para el programador, y hacen que una solución basada en estos micro-controladores sea difícil de mantener haciendo innecesariamente complejo para proyectos que no requieren de estas características.

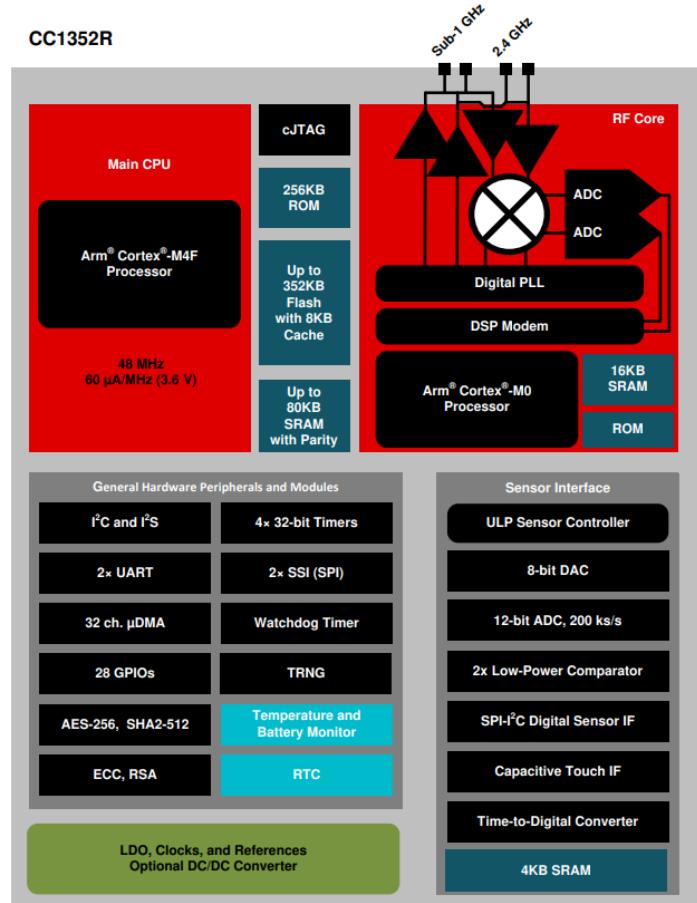


Figure 1-1. CC1352R Block Diagram

Figura 8: Diagrama de bloques funcionales de un CC1352

**Raspberry Pi** La raspberry pi [14] que puede verse en la figura 9 es lo que se define como un SBC (single board computer). Es decir un computador completo desplegado en una sola PCB o placa de circuito impreso. Incorpora puertos USB, HDMI, RJ45, jack de audio, WiFi y Bluetooth (en las ultimas versiones) y el abanico básico de conexiones como es I2C, SPI o UART. Esta dotada de un SoC (system on a chip) Broadcom con entre 1 y 4 núcleos de



Figura 9: SBC Raspberry Pi 4

32 y 64 bits (según versiones) de arquitectura ARMv11 y ARMv8, con un chip de vídeo dedicado integrado y hasta 8GB de ram en su última versión. Como característica destacable cabe mencionar que corre un sistema linux que puede encontrarse en diferentes aromas, el principal de ellos basado en debian conocido como raspbian ahora llamado raspberry pi OS [20]. Estas son las razones que la hacen muy versátil así como un precio relativamente contenido que se sitúa en torno a los 40€ dependiendo del vendedor y la versión. Tiene una altísima penetración en el mercado con 30 millones de unidades vendidas [24] según el fabricante. Mantiene igualmente una comunidad enorme y muy activa, por lo que es fácil comenzar a desarrollar con ella proyectos de pequeña y media envergadura.

**Arduino** Arduino es una plataforma de prototipado rápido multi-propósito abierto, es decir, tanto su hardware como software son abiertos, con la única excepción de algunas cuestiones sobre el microprocesador que el fabricante no hace públicas, pero que no son necesarias de ninguna manera para utilizar el microcontrolador. Arduino es además del hardware, el conjunto que la comunidad de desarrolladores tanto de la marca Arduino, como externa a ella que proporcionan componentes software y hardware compatibles. Es por tanto un ecosistema completo que permite al desarrollador de cualquier nivel llevar a cabo un prototipo rápidamente. Se sabe que la penetración en el mercado es masiva, y que su presencia en proyectos a baja y media escala de desarrollo hardware es hegemónica, pero se desconoce el número de placas vendidas debido a que fabricantes de todo el mundo pueden producir sus propias unidades, y no existe un recuento de cuantas han llegado al mer-

cado. No obstante, la pagina oficial muestra un contador de descargas del IDE oficial que actualmente muestra la cifra de 44 millones de descargas. El modelo mas extendido de placa Arduino es el Arduino Uno R3 que incorpora un microprocesador ATMega 328 de arquitectura AVR (Harvard) y 8 bits, aunque el compilador es capaz de manejar variables de hasta 32, que corre hasta a 20MHz. Tiene 2KB de ram, 32Kb de memoria de programa y 1KB de EEPROM. Cuenta con puertos I2C, SPI, UART, PWM, ADC de 10 bits y pines capacitivos. Esta placa incorpora ademas un regulador de tension, y un segundo microcontrolador para el FTDI, normalmente un atmega 16u2 que ademas puede ser reprogramado aunque en algunas versiones ha sido sustituido por un CH340 que no se puede reprogramar. Todos estos elementos pueden verse relacionados en la imagen inferior.

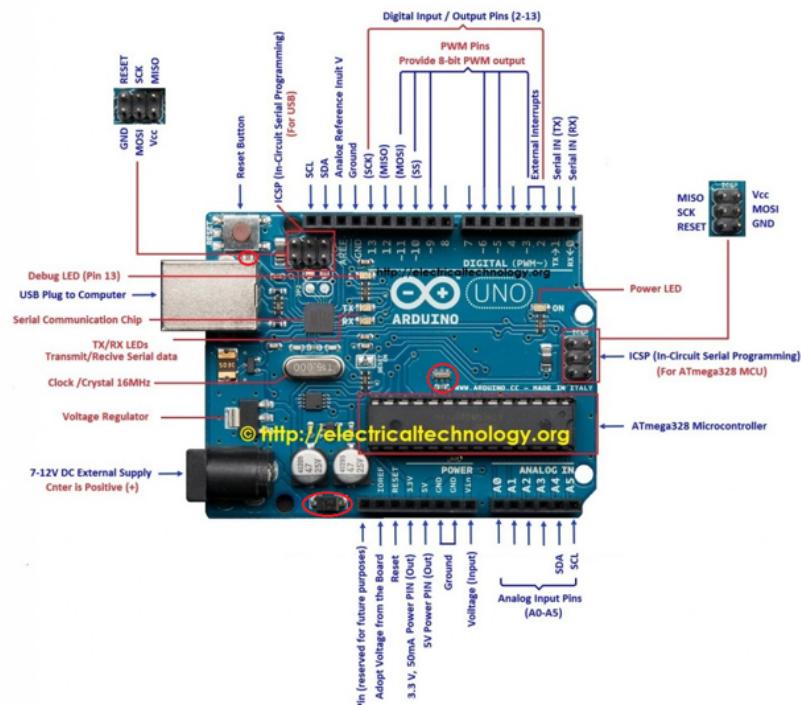


Figura 10: Anatomia de un Arduino UNO R3

### 1.10.3. Reloj de tiempo real (RTC)

El reloj de tiempo real es un componente hardware que indica la hora real (dia, mes, año, horas, minutos y segundos). A pesar de que todos los microcontroladores incorporan un reloj, en algunos de estos no es posible utilizarlo para medir el tiempo, o de serlo, no indican una fecha, solamente el tiempo

que el microcontrolador ha pasado funcionando desde la ultima vez que se encendió o se reinició. Existen algunos microcontroladores que incorporan un reloj de tiempo real en el mismo encapsulado o como parte incluso de la arquitectura del propio microcontrolador, pero no es muy frecuente, y por esa razón es necesario añadir uno al diseño. En este caso se van a analizar las principales alternativas en el mercado entre las que podemos destacar:

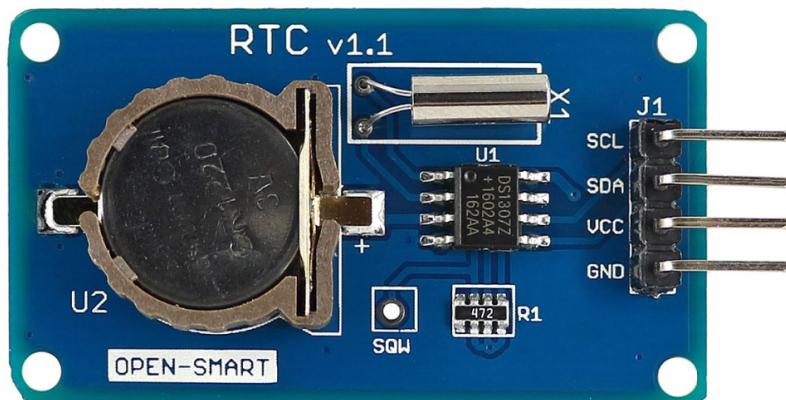


Figura 11: Modulo con reloj de tiempo real DS1307

**DS1307** El reloj DS3231 del fabricante Maxim Integrated es un reloj de 5v regulado por un oscilador de cuarzo común de 32768Hz que cuenta con una interfaz I2C a 100KHz para comunicarse con el microcontrolador y poder fijar y leer la hora. Este reloj cuenta también con una conexión para una batería externa y un generador de pulsos cuadrados de hasta 32 KHz. Puede funcionar a temperaturas extremas de entre -40°C y hasta 85°C sin embargo al estar operado por un cristal común las variaciones de temperatura tendrán indefectiblemente como consecuencia una desviación en la frecuencia del cristal. Esta limitación provoca que este reloj presente una perdida de entorno

a 15 minutos por año lo cual puede ser inaceptable en muchas situaciones. Cada modulo tiene un precio aproximado de en torno a 2.5€ y no incluye la batería de mantenimiento.



Figura 12: Modulo PCF2129 con conexión para Raspberry Pi

**PCF2129** El PCF2129 es una alternativa como reloj de tiempo real de bajo coste que puede funcionar a temperaturas de -40°C a 85°C y 5v. Ofrece también temporizadores de un segundo y un minuto, una salida de pulsos cuadrados configurable de hasta 32Khz dos interfaces I2C a 400Khz y una interfaz SPI. Este reloj tiene también una alarma configurable y un watchdog cuyo comportamiento puede decidir el programador.

Como característica relevante cabe mencionar que opera con un reloj compensado por temperatura (TCXO) integrado en el empaquetado del reloj, lo que disminuye enormemente la desviación de la medición del tiempo a causa de los cambios de frecuencia que sufre el oscilador con las variaciones de la temperatura. Según el fabricante esta precisión es de al menos 3 ppm sin embargo como puede observarse en la gráfica extraída del estudio de Dan Brown [21] en condiciones normales es mucho mas preciso.

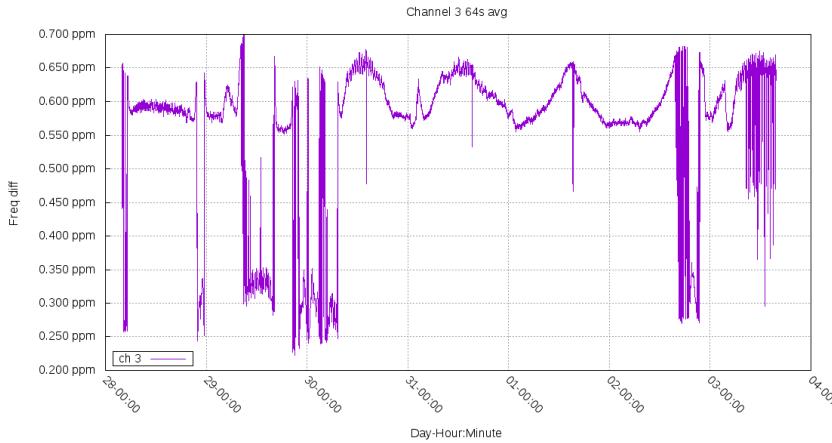


Figura 13: Historico de desviacion de la frecuencia para PCF2129

En esta figura se puede observar el error del cristal del reloj con respecto a un cristal de control que en el estudio se utiliza para medir el error. El reloj utilizado para medir tiene una precisión de 40.8 nano segundos (tomando como referencia la hora GPS) y esta compuesto por un procesador STM32 y un cristal compensado por temperatura como puede verse en la entrada para la placa de desarrollo [22].

**DS3231** El DS3231 aparece como reemplazo para el DS1307 lanzado por el mismo fabricante. Ofrece capacidades similares al PCF2129 es decir: es un reloj de tiempo real a 5v, está regulado por un cristal compensado por temperatura y puede trabajar a las mismas temperaturas de entre -40°C a 85°C

Este reloj ofrece una sola interfaz I2C a 400Khz y dos alarmas configurables. Tiene también una batería de mantenimiento y ofrece una salida de tren de pulsos a 32KHz.

En este caso el fabricante asegura un precisión de al menos 3.5ppm llegando hasta 2ppm en el rango de temperatura que abarca desde los 0°C a los 40°C sin embargo en condiciones normales el margen de error que se puede apreciar es mucho menor, de menos de 0.5ppm llegando a acumular unos 10 segundos anuales de error.

Frente al resto de modelos cabe destacar como característica diferenciadora que este reloj posee un compensador para el envejecimiento del cristal que permite ajustar la desviación que este sufre con el deterioro que provoca el tiempo de funcionamiento o, en caso de producirse, un proceso de soldadura donde el integrado permanece expuesto a excesivo calor.

Este modelo es el mas extendido por su sencillez de uso y su precio que se



Figura 14: Modulo DS3231

sitúa en torno a un euro incluyendo la batería. Ademas el modulo incorpora una memoria EEPROM en el mismo bus I2C que puede resultar útil en usos futuros.

#### 1.10.4. Modulo inalambrico

Para poder comunicar el dispositivo y configurar su funcionamiento es necesario revisar que módulos pueden comunicarse con un dispositivo común (un teléfono móvil, tablet, u ordenador), sin necesidad de cables ni hardware extra en el equipo que va a realizar la configuración del dispensador de comida. Por esta razón solamente van a ser evaluados los módulos inalámbricos que utilicen WiFi o Bluetooth. Por ser las tecnologías que incorporan de forma casi unánime los dispositivos que hay en el mercado.

**CC3000** El CC3000 te Texas instruments es un controlador inalámbrico Wifi compatible con los estándares 802.11b/g. Este chip puede trabajar con redes WEP, WPA y WPA2 y es capaz de manejar hasta 4 conexiones simultáneamente. Se comunica a través de un bus SPI con el procesador es

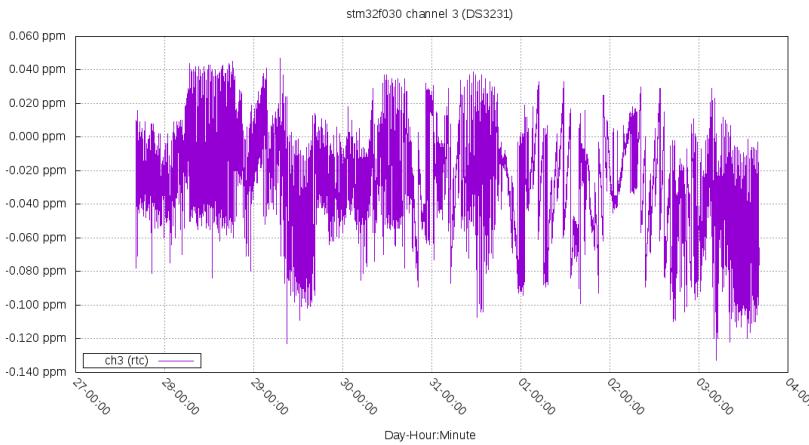


Figura 15: Historico de desviacion de la frecuencia para DS3231

incorpora ademas una tarjeta SD en el mismo bus SPI.

Sin embargo, este modulo no es capaz de funcionar en modo Punto de Acceso, por lo que es necesario configurar la red inalámbrica de otra forma. A pesar de que es posible hacer esta configuración usando el propio modulo que se conecte a una red prefijada que el usuario debe construir bien sea con su teléfono móvil o con cualquier otro dispositivo, esto añade complejidad para el consumidor final, y llevaría asociada la necesidad de tener una conexión a Internet siempre disponible para configurar el dispositivo.

**NRF24L01** El modulo NRF24L01 es un microcontrolador bluetooth que ofrece un vinculo bluetooth 2.0 semi duplex. Este integrado trabaja a 3.3v aunque sus pines pueden funcionar a 5v, tiene una antena integrada con un alcance de entre 10 y 20 metros y un coste aproximado al momento de realizar el prototipo de unos 10€ por unidad. Esto ultimo unido a que necesita un bus SPI para comunicarse con el microcontrolador, hacen que este chip sea caro de utilizar, ya no solo por el precio de salida que tiene, si no por la cantidad de pines que necesita.

**HC-06/HC-05** Los módulos HC-06/05, son en realidad el mismo modulo, y utilizan el mismo chip, pero el fabricante los vende bajo nombres diferentes porque utilizan un firmware diferente, que en el caso del HC-05 le permite funcionar como maestro en la comunicación. Estos módulos ofrecen un modem bluetooth configurable a través de comandos AT que expone una interfaz UART para comunicarse con el microcontrolador a 5v y pueden ser alimentados en el rango de los 3.6v a 6v. Ademas, estos módulos tienen algunos GPIO configurables, y es posible modificar su firmware para actualizarlo, aunque

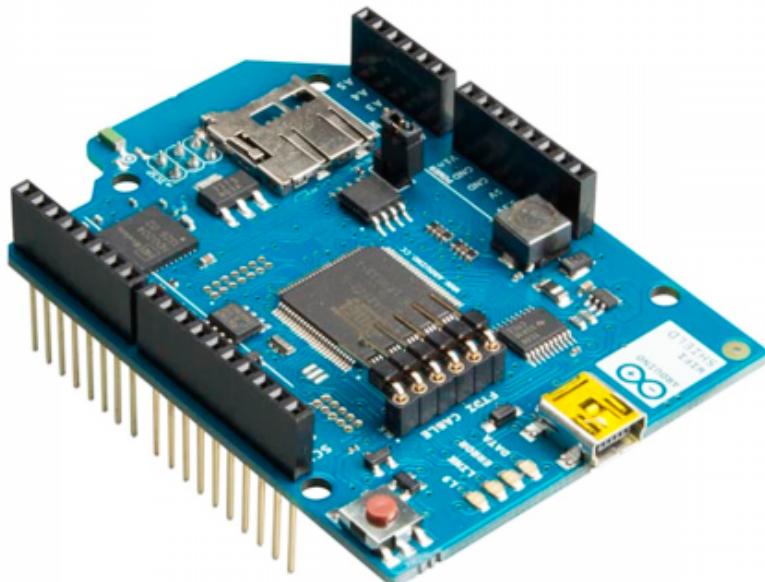


Figura 16: Modulo CC3000 wifi

es un proceso que no compensa la diferencia de precio de unos 5€ para el HC-06 (también llamado Linvor) y unos 8€ para el HC-05. Es importante mencionar que el el firmware del HC-06 es inestable y se producen desconexiones habitualmente, cosa que en el caso del HC-05 no ocurre, al menos con la versión de firmware que en este caso incluían los módulos. Lamentablemente el fabricante no da ningún detalle sobre el software que incorporan los módulos, por lo que adquirir una versión con el firmware apropiado es una cuestión de suerte.

#### 1.10.5. Motor

El motor es un actuador critico en este proyecto. Es el responsable de mover físicamente la tolva, y tiene que permitir controlar con una precisión menor a 5 grados sexagesimales la posición del rotor. También es necesario que pueda girar a bajas revoluciones que pueden variar entre 20 y 180 revoluciones por minuto y debe tener la fuerza necesaria para poder romper

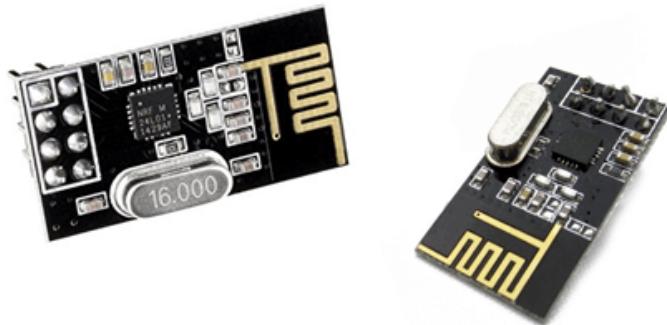


Figura 17: Modulo NRF24L01 Bluetooth

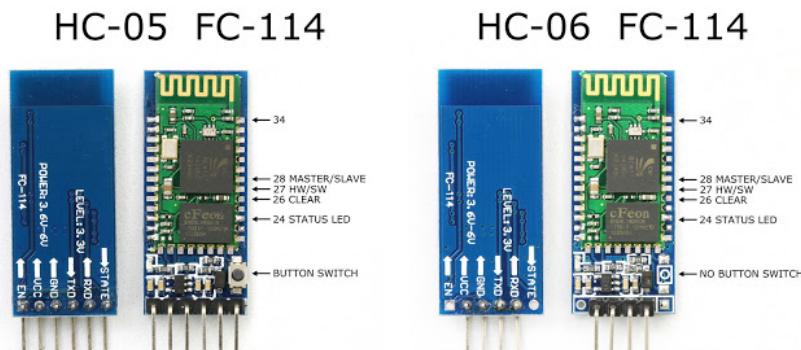
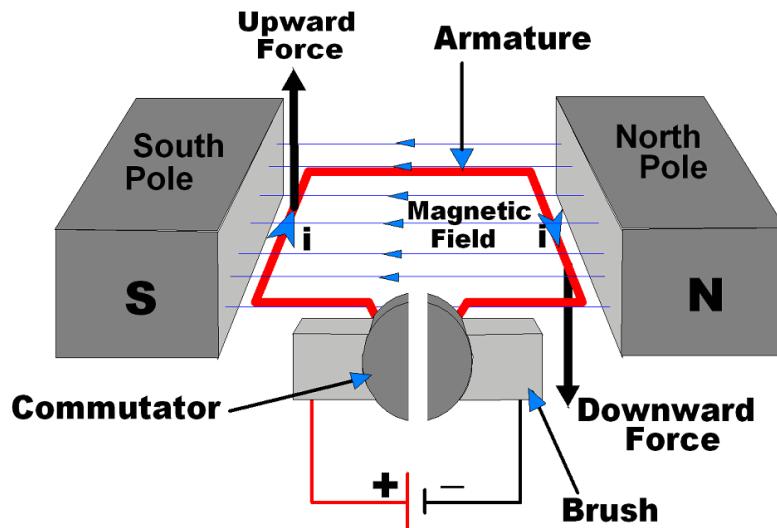


Figura 18: Modulos HC-06/HC-05 Bluetooth

un grano de comida sin que esto suponga un esfuerzo que pueda estropear el bobinado

**Motor con escobillas** Los motores con escobillas se han popularizado gracias a su bajo coste y sencillez para operar con ellos. Su funcionamiento básico consiste en hacer que una bobina colocada en el rotor se cargue eléctricamente para ser atraída por un imán permanente colocado en el exterior. La alternancia en la carga de la bobina se consigue gracias al propio giro del motor, pues al tener los contactos en el eje, y las escobillas fijas al estator, estas van haciendo contacto con las diferentes bobinas cuando el motor gira. Este tipo de motores presentan varios problemas, el primero y más relevante para este proyecto, es la imposibilidad de conocer sin un método de control



**DC Motor Conceptual Diagram**

Figura 19: Diagrama de un motor de corriente continua con escobillas

secundario la posición exacta del rotor, o la velocidad de este. Ademas, existe un punto de equilibrio en el que ninguna cantidad de corriente hará que el motor gire.

**Servo** Un servomotor es un motor de corriente continua al que se le ha añadido un potenciómetro angular que se mueve solidario con el eje de salida, con el fin de poder controlar la posición. Normalmente los servomotores o servos están enfocados a hacer pequeños movimientos y tener una alta repetitividad, por lo que no suelen ofrecer velocidades angulares muy altas, y lo que es mas importante aun, su angulo de giro normalmente no es infinito. Por otra parte tienen el inconveniente de que dado su tamaño, es necesario utilizar un multiplicador de par y esto también eleva su precio.

Como puede verse en la figura superior en la que se muestra un diagrama de un servomotor común en el que se ve como un potenciómetro permite leer la posición del eje de salida midiendo la resistencia del potenciómetro.

**Brushless** Un motor brushless o sin escobillas, resuelve el problema del rozamiento que tenían los motores convencionales, al colocar los imanes en el rotor, y las bobinas en el estator. Para conseguir el giro, se disponen imanes permanentes con las polaridad alterna en el rotor, y una estrella de bobinas

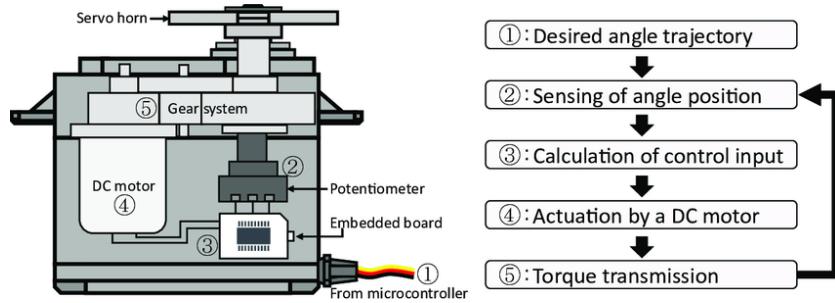


Figura 20: Diagrama de un servomotor

en el estator que se van cargando en orden y con polaridades alternas para hacer girar el rotor como puede verse en la figura 21. Con este modelo ya no se produce el punto de equilibrio, y al reducir el rozamiento y la flotación de las escobillas, pueden girar mas rápido. Aunque es posible controlar el orden de encendido de las bobinas y su velocidad para controlar la velocidad y sentido de giro, la electrónica se vuelve mas compleja por tener que controlar las etapas de potencia. Por las razones antes mencionadas, estos motores están orientados a aplicaciones donde se necesita una velocidad muy alta de giro, que si bien se puede controlar, no es necesario saber la posición del rotor en cada momento.

**Paso a paso** Los motores paso a paso se diferencian de un motor convencional, pero comparten la característica con un motor sin escobillas, que cuando son alimentados, en vez de producir un movimiento continuo, su eje solo gira un angulo determinado, típicamente 1.8 grados. Un motor paso a paso, al igual que un motor brushless, carece de escobillas, y tiene colocadas las bobinas en el estator, salvo, porque sus imanes están extremadamente cerca de las bobinas. Este tipo de motores, al igual que los servos, están enfocados a tareas de alta repetitividad, pero son capaces de ofrecer pares mas altos, y una mejor precisión a costa de una menor velocidad angular y eliminando la necesidad de un sistema de control en bucle cerrado al ser posible controlar eléctricamente la cantidad de movimiento.

### 1.11. Pantalla

La pantalla es el elemento que se utiliza para mostrar información relacionada en este caso con el estado del dispositivo. Dado que es necesario mostrar siempre esta información, pero no siempre es posible utilizar un segundo dispositivo, en esta apartado se van a evaluar algunas de las innumerables alternativas de pantallas que se pueden encontrar.

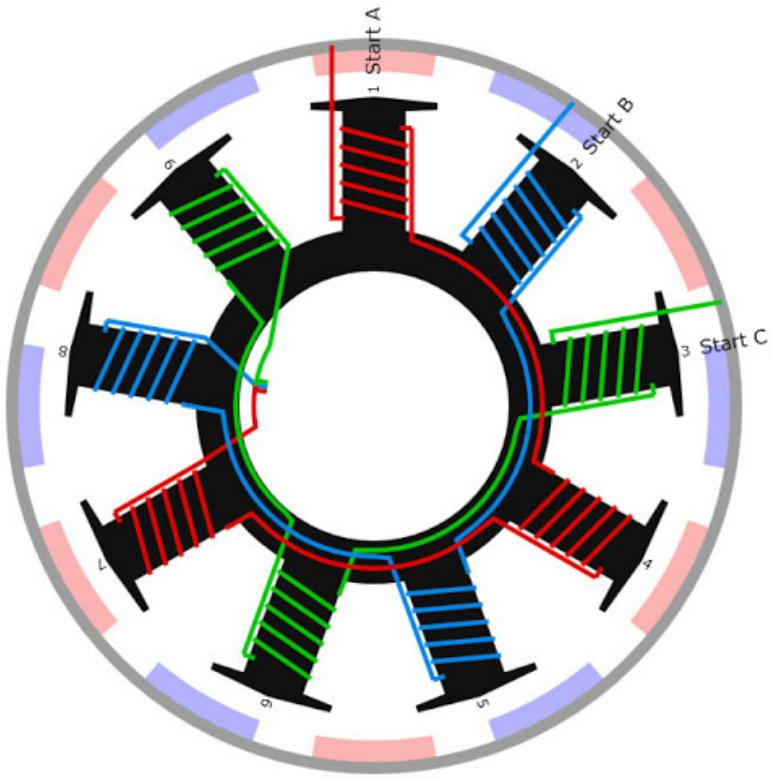


Figura 21: Diagrama de un motor sin escobillas

**7 Segmentos** Se le llama display de 7 segmentos a un tipo concreto de indicador formado por 7 segmentos normalmente rectos y un punto, que forman un ocho, de tal manera, que encendiendo unos u otros es posible representar los diez numeros y algunas letras y figuras. Son baratos y están disponibles en una amplia variedad de formas tamaños y colores (incluso RGB por segmento), pero debido a que la posición de los segmentos es fija, no son capaces de representar de forma reconocible el alfabeto completo. Existen modificaciones de 13, 14 y 16 segmentos que añaden versatilidad a la hora de representar caracteres como puede verse en la figura 23, pero sigue siendo insuficiente para representar el alfabeto latino completo al incorporar este la letra eñe y el acento, y ademas consumen una gran cantidad de pines, o precisan de un controlador dedicado.

**LCD 16x2** Las pantallas 16x2, son en realidad un conjunto de 16x2 matrices de 8x5 píxeles que tienen un alfabeto prefijado, pero que también permiten construir "caracteres" definidos por el programador para suplir aquellos iconos

que por defecto no forman parte del conjunto que el microcontrolador tiene . Estas pantallas son baratas y fáciles de manejar. Normalmente incorporan un segundo micro-controlador (PCF8574) que hace de interfaz I2C a la interfaz propia de la pantalla con que es posible reducir enormemente la cantidad de pines necesarios y la complejidad de la electrónica.

### **ILI9481** .

Esta es una pantalla matricial de 320 x 480 píxeles RGB táctil resistiva que funciona a 5v. Esta pantalla tiene una tasa de refresco de hasta 30 cuadros por segundo y un espacio de 262000 colores con retroiluminación LED. Esta pantalla requiere 15 pines para poder ser controlada si se quiere utilizar también el bus SPI que incorpora para una tarjeta SD. En caso de prescindir de este bus, el numero de pines necesario bajaría a 11. Esta pantalla es compleja de programar, y necesita un procesador que ofrezca la potencia necesaria como para poder actualizar cada cuadro a un ritmo suficiente que haga imperceptible al ojo humano la actualización por lineas que en realidad se produce debido a que el micro controlador carece de un buffer de cuadros como puede verse en la documentación técnica del fabricante [8].

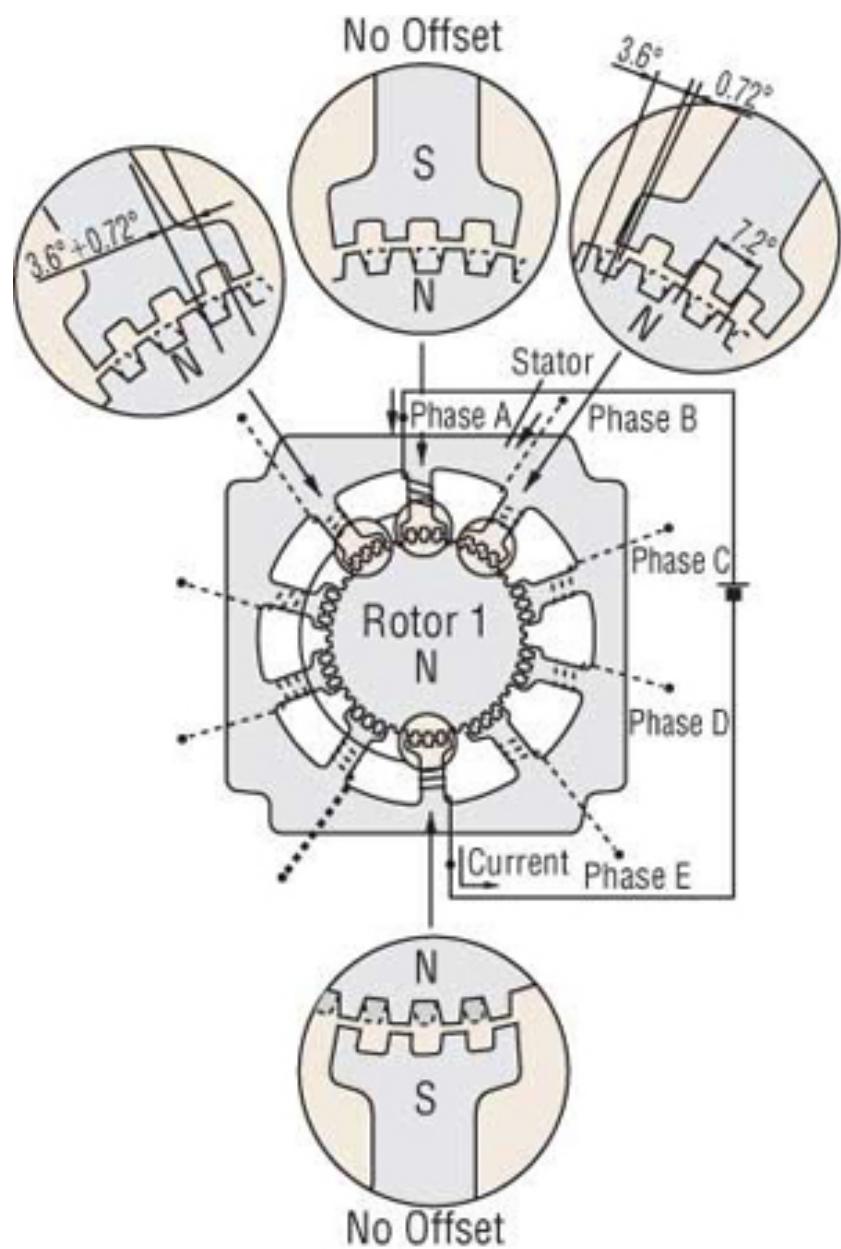


Figura 22: Diagrama de un motor paso a paso

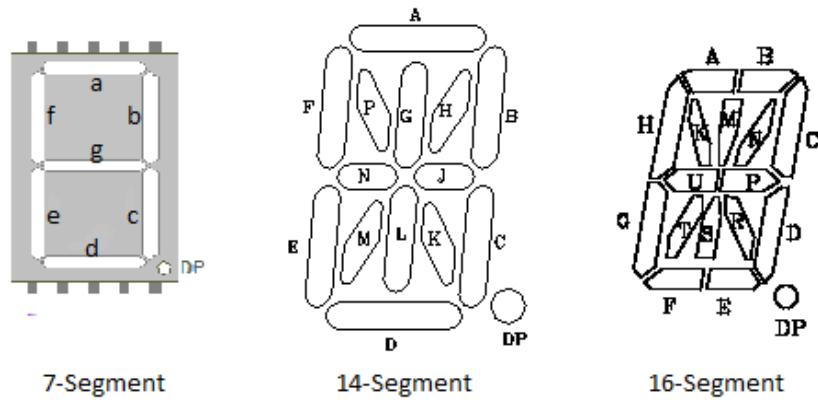


Figura 23: Indicador de 7, 14 y 16 segmentos

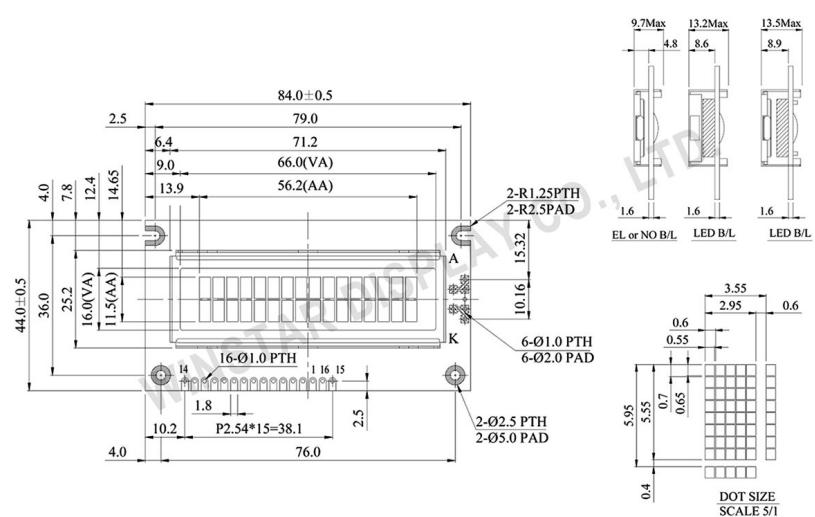


Figura 24: Diagrama de ingeniería de una pantalla 16x2

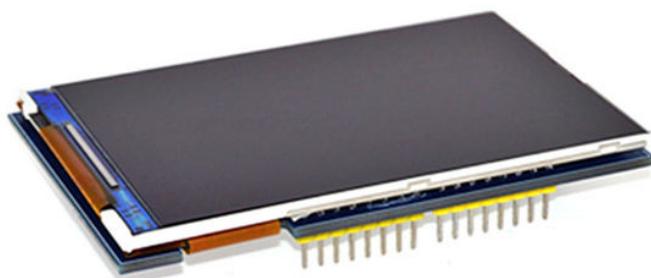


Figura 25: Modulo de pantalla ILI9481.jpg

## 2. Diseño

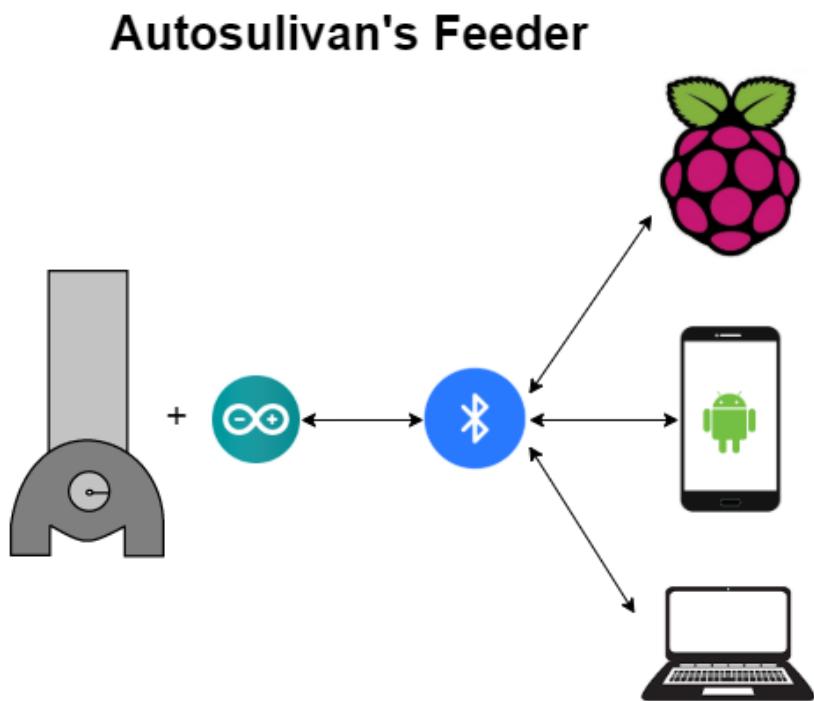


Figura 26: Modulo A4988

Este dispositivo, como puede verse en la figura 27, esta planteado para ser autónomo, pero permitir al usuario configurarlo facilmente desde otro dispositivo, por ellos se ha optado por utilizar un arduino que se comunique a través de bluetooth con un dispositivo móvil android donde se ejecuta una aplicación cliente que haciendo uso de una API en linea de comandos permita configurar el comedero.

Para desarrollar este dispositivo se ha optado por utilizar un microcontrolador ATMega328 incluido en una placa Arduino Nano. Para mostrar información al usuario se ha utilizado una pantalla LCD 16x2 con un modulo I2C integrado manejado por un microcontrolador PCF8574T un reloj DS3231 para obtener la hora un controlador de doble puente H A4988 y un motor paso a paso nema 17 manejado por este controlador. Como modulo bluetooth se ha utilizado un HC-05. Todos estos componentes han sido elegidos por ser aquellos que ostentan el mayor compromiso entre precio y facilidad de uso.

En la figura 27 se puede ver como se ha conectado cada componente hardware con el microcontrolador.

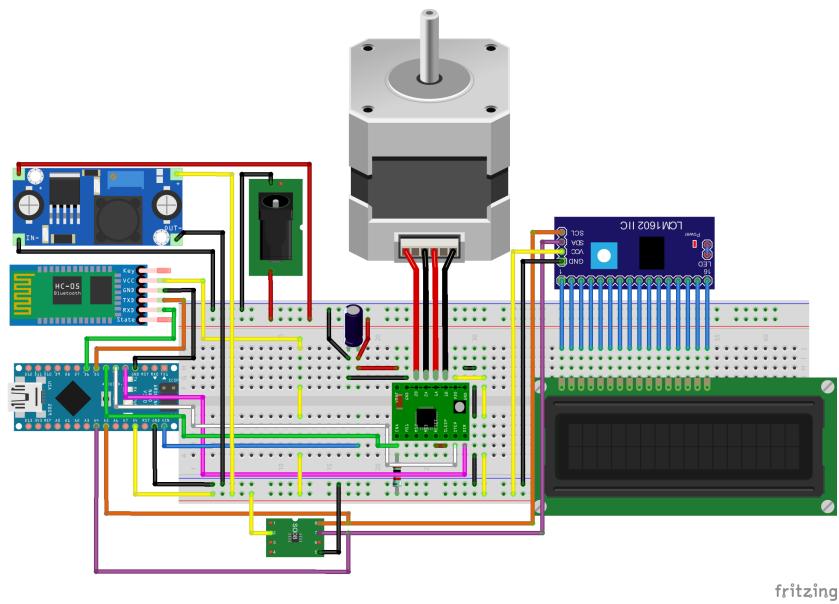


Figura 27: Despligue electronico maquetado en Fritzing

La lista de componentes necesarios y su precio puede consultarse en la tabla 1

Parte	Coste unitario	Cantidad
Arduino Nano	4.60€	1
LCD 16X2 I2C	2.32€	1
Pololu A4988	2.34€	1
Bluetooth HC-05	5.11€	1
Motor NEMA 17 48mm	15.04€	1
Conversor DC-DC Lm2596	1.09€	1
Placa Perforada	8.23€	1
Cable Dupont	1.03€	1
Usb montado en pared	2.86€	1
Enchufe 220 con fusible	17.55€	1
Reloj DS3231	1.69€	1
Fuente de alimentación 12v	5.99€	1
Pla blanco	8€	1
ABS verde	10€	1
Pieza Tubo PVC	4.35€	1
Plancha PVC	2.5€	1
Táper	1.20€	1
Tornillería Acero	5.35€	1
Total	97.25€	

Tabla 1: Tabla de coste de materiales.

### 3. Planificación

Para desarrollar este proyecto se ha utilizado la metodología de programación extrema por estar centrada en los cambios constantes y la aparición de nuevos requisitos a medida que el proyecto se iba desarrollando además de estar centrada en la agilidad y rapidez en el desarrollo [26]. Para esto se implementaba una funcionalidad a la vez que ha sido probada con test unitarios antes de cerrarla y continuar con el desarrollo.

Esto puede verse reflejado en el diagrama de Gant de la figura 28 donde se puede ver que el código se revisaba con frecuencia y se replanteaba varias veces a lo largo del tiempo.

Meses	Mes 1				Mes 2				Mes 3				Mes 4			
Actividades - Semanas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Planificación																
Documentación																
Aprendizaje																
Análisis																
Diseño																
Implementación																
Depuración																
Implantación y pruebas																
Memoria																

Figura 28: Diagrama de Gant del proyecto

### 4. Técnica y herramientas

En esta sección se describen con detalle cada uno de los componentes software y hardware que han sido utilizados para el desarrollo del proyecto. Se ha omitido la suite ofimática.

#### 4.1. Dosificación y fiabilidad mecánica

Para poder dosificar la comida, es necesario poder controlar el movimiento de la tolva. La forma de solucionar este problema hace necesario un sistema de control del giro del motor que en este caso consiste en un sistema de control en lazo abierto, donde el actualizador elegido ha sido un motor NEMA 17 modelo 17HS4401 cuya hoja de datos técnicos [13] muestra un par de 40 N/m que es suficiente para romper un grano de comida que necesita unos 25 N de fuerza hasta romperse (medidos utilizando una bascula de cocina y un gato de mesa). Se considera que al utilizar un motor que entrega una potencia un 60 % superior a lo necesario el sistema está correctamente dimensionado con un margen suficiente que permite que el motor no tenga que trabajar en el

límite máximo de esfuerzo. Esto alarga la vida de las bobinas del motor que están preparadas para permitir el paso de 1.7A, cantidad de corriente que atraviesa el motor cuando se atasca.

Para controlar este motor se utiliza un controlador en doble puente H modelo A4988 que como su hoja de especificaciones muestra [1] es capaz de manejar hasta 2 amperios, lo que es un 17 % mas del máximo que puede atravesar el motor.

Como puede verse en la figura 29 este controlador tiene 16 líneas. Enable: Esta linea habilita el paso de corriente por el motor cuando se pone a tierra. Esto quiere decir, que es posible apagar el motor cuando no se esté usando, ya que es cuando está parado cuando mas corriente lo atraviesa.

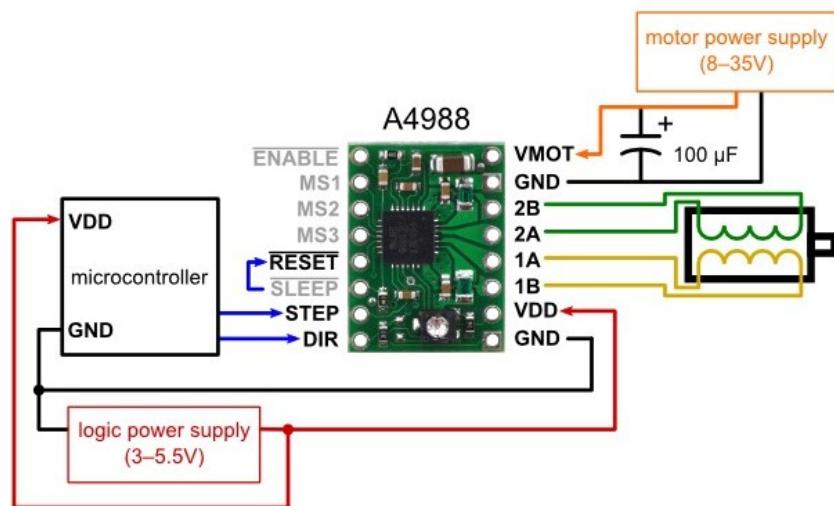


Figura 29: Modulo A4988

MS1-3: Estas líneas lo que hacen es, actuando como un contador binario, elegir el número de micropasos con los que se controla el giro. Esto quiere decir que si el motor utilizado en este caso tiene 200 pasos por revolución, con 3 bits, es posible utilizar hasta 16 micropasos, que el controlador consigue a partir de cargar parcialmente las bobinas del motor en lugar de alimentarlas o

no con toda la corriente disponible. En este caso se pueden dejar sin conectar para no utilizar la división en micropasos ya que no es necesario aumentar la resolución del motor.

Reset-Sleep: Estas líneas permiten apagar el motor y además el controlador para reducir su consumo. En este caso no van a utilizarse, pues la cantidad de energía que disipa el integrado A4988 es demasiado baja como para suponer un cambio significativo al tratarse de unos 50 mAh según se muestra en su datasheet [1].

Step: Esta línea provocará que el controlador cambie el estado de las bobinas para producir un movimiento de un paso en el motor cada vez que esta línea pasa de valor bajo a alto. Este será el pin que es necesario controlar para hacer girar el motor.

Dir: Esta línea cambia el sentido de giro del motor. Este sentido depende también de como se haya conectado el motor al controlador, por lo que el sentido horario o antihorario son arbitrarios, no obstante en este caso el motor ha sido conectado para que el sentido de giro positivo, o que mueve la comida hacia el exterior es antihorario vista la tolva desde la perpendicular al eje de giro y en su salida.

Vmot: Entrada de corriente para el motor.

1A, 1B, 2A, 2B: Estas con las salidas de corriente para las dos bobinas que este controlador es capaz de gestionar. Aquí es donde se debe conectar el motor.

Vdd: Alimentación para el controlador de 5V.

Por tanto el controlador y el motor quedan conectados al microcontrolador como puede verse en la imagen 30

La función que controla el giro del motor tendrá por tanto que poner en bajo la línea Enable, poner a bajo la línea DIR si gira en sentido positivo, y generar un tren de pulsos que hagan girar el motor una vuelta, y tantas vueltas como sea necesario como puede verse en el fragmento de código 1.

Por otra parte y con el fin de orientar a los usuarios con menor conocimiento este comedero cuenta con una función que permite calcular la cantidad de comida necesaria y emitir alertas de sobrepeso en función de la cantidad de masa de la mascota. El objetivo de esta función es liberar al usuario de la necesidad de medir la comida, debido a que esta operación, de hacerse sin instrumentos de medida, es difícil realizarla con precisión, y es habitual proporcionarle cantidades incorrectas de alimento al animal, lo que normalmente suele derivar en un sobrepeso notable en las mascotas.

Para hacer el cálculo se utiliza la tabla nutricional de friskies como ejemplo que puede verse en la figura 31 que puede encontrarse en cualquier supermercado [23].

Para hacer el cálculo se utiliza el fragmento de código 2 en el que se puede

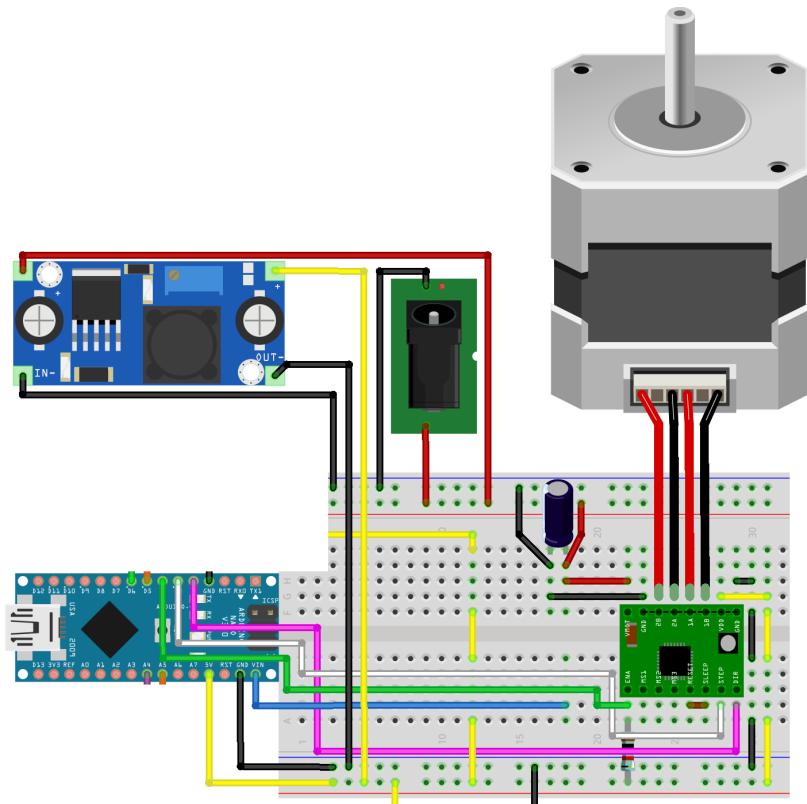


Figura 30: Conexion del controlador A4988

ver y modificar la forma en que el dispositivo calcula la cantidad de pienso que debe suministrar al animal.

```

1 void girarTolva (int vueltas, bool sentido){
2
3     if (sentido){
4         digitalWrite(STEPPER_DIR, LOW);
5     } else{
6         digitalWrite(STEPPER_DIR, HIGH);
7     }
8
9     digitalWrite(LED_PIN, HIGH);
10    digitalWrite(STEPPER_ENABLE, LOW);
11
12    for (int i=0; i<vueltas; i++){
13        for (int j=0; j<PASOS_POR_VUELTA; j++){
14
15            digitalWrite(STEPPER_STEP, HIGH);
16            delay(espera);
17            digitalWrite(STEPPER_STEP, LOW);
18            delay(espera);
19        }
20    }
21    digitalWrite(STEPPER_ENABLE, HIGH);
22    digitalWrite(LED_PIN, LOW);
23}

```

Código 1: código ejemplo.

<p><b>DAILY FEEDING GUIDELINES</b></p> <table border="1"> <thead> <tr> <th rowspan="2">WEIGHT OF CAT</th> <th colspan="4">DAILY FEEDING AMOUNT*</th> </tr> <tr> <th>Dry Food Only</th> <th colspan="3">Dry + Canned</th> </tr> <tr> <th>kg lbs</th> <th>grams</th> <th>cups</th> <th>grams</th> <th>cups / oz.</th> </tr> </thead> <tbody> <tr> <td>2 to 4 5 to 9</td> <td>DRY: 54 to 81</td> <td>1/2 to 3/4</td> <td>DRY: 27 to 36</td> <td>1/4 to 1/3 cup</td> </tr> <tr> <td></td> <td></td> <td></td> <td>WET: 85 to 170</td> <td>3 to 6 oz.</td> </tr> <tr> <td>5 to 6 10 to 14</td> <td>DRY: 81 to 108</td> <td>3/4 to 1</td> <td>DRY: 36 to 71</td> <td>1/3 to 2/3 cup</td> </tr> <tr> <td></td> <td></td> <td></td> <td>WET: 170 to 225</td> <td>6 to 8 oz.</td> </tr> </tbody> </table> <p>*Recommended daily feeding amounts, using a standard 250 mL (8 oz.) measuring cup.</p> <p><b>AFTER EXPLORING, IT'S TIME TO REFUEL.</b> These amounts are averages and your cat's needs may differ. Feeding should be adjusted as necessary to maintain an ideal body condition. Feed your cat PURINA® Friskies® canned in addition to or mixed with PURINA® Friskies® dry. However, since cats enjoy eating small meals throughout the day, keep a bowl of PURINA® Friskies® dry cat food available at all times.</p> <p><b>MAKING THE SWITCH TO PURINA® Friskies®</b> Over the course of 7 to 10 days, add a small portion of PURINA® Friskies® Indoor Delights® dry cat food to your cat's current dry food, and decrease the current food by the same amount. Each day, gradually increase the proportion of PURINA® Friskies® Indoor Delights®, while gradually decreasing the amount of your current food, until the changeover is complete.</p> <p><b>WATER</b> Provide adequate fresh water in a clean container daily. For your pet's health, see your veterinarian regularly.</p> <p><b>STORE IN A COOL, DRY PLACE.</b> Calorie Content (calculated): 3371 kcal/kg • 364 kcal/cup</p>	WEIGHT OF CAT	DAILY FEEDING AMOUNT*				Dry Food Only	Dry + Canned			kg lbs	grams	cups	grams	cups / oz.	2 to 4 5 to 9	DRY: 54 to 81	1/2 to 3/4	DRY: 27 to 36	1/4 to 1/3 cup				WET: 85 to 170	3 to 6 oz.	5 to 6 10 to 14	DRY: 81 to 108	3/4 to 1	DRY: 36 to 71	1/3 to 2/3 cup				WET: 170 to 225	6 to 8 oz.	<p>Les essais d'alimentation effectués conformément aux normes de l'Association of American Feed Control Officials (AAFCO) ont montré que la nourriture Purina Friskies Régal pour chats d'intérieur constitue une alimentation complète et équilibrée pour le maintien des chats adultes.</p> <p><b>RATIONS QUOTIDIENNES</b></p> <table border="1"> <thead> <tr> <th rowspan="2">POIDS DU CHAT</th> <th colspan="4">RATION QUOTIDIENNE*</th> </tr> <tr> <th>Nourriture sèche seulement</th> <th colspan="3">Nourriture sèche et nourriture en boîte</th> </tr> <tr> <th>kg lb</th> <th>grammes</th> <th>tasses</th> <th>grammes</th> <th>tasses/oz</th> </tr> </thead> <tbody> <tr> <td>2 à 4 5 à 9</td> <td>SÈCHE: 54 à 81</td> <td>1/2 à 3/4</td> <td>SÈCHE: 27 à 36</td> <td>1/4 à 1/3 tasse</td> </tr> <tr> <td></td> <td></td> <td></td> <td>HUMIDE: 85 à 170</td> <td>3 à 6 oz</td> </tr> <tr> <td>5 à 6 10 à 14</td> <td>SÈCHE: 81 à 108</td> <td>3/4 à 1</td> <td>SÈCHE: 36 à 71</td> <td>1/3 à 2/3 tasse</td> </tr> <tr> <td></td> <td></td> <td></td> <td>HUMIDE: 170 à 225</td> <td>6 à 8 oz</td> </tr> </tbody> </table> <p>*Rations quotidiennes recommandées, en utilisant une tasse à mesurer standard de 250 mL (8 oz).</p> <p><b>APRÈS AVOIR TANT EXPLORÉ, UN PLEIN D'ÉNERGIE S'IMPOSE.</b> Les quantités recommandées sont des moyennes; il se peut que les besoins de votre chat soient différents. Les rations doivent être ajustées au besoin pour que l'animal maintienne une condition physique idéale. Donnez-lui de la nourriture en boîte PURINA® Friskies® comme ajout à la nourriture sèche PURINA® Friskies® ou donnez-lui une combinaison des deux. Cependant, comme les chats aiment pouvoir manger de petits repas plusieurs fois par jour, laissez un bol de nourriture sèche PURINA® Friskies® à la portée de votre chat en tout temps.</p> <p><b>PASSAGE À PURINA® Friskies®</b> Remplacez une petite quantité de sa nourriture actuelle par la nourriture sèche PURINA® Friskies® Régal pour chats d'intérieur et augmentez graduellement la quantité de nouvelle nourriture en diminuant celle de l'ancienne. La transition complète devrait prendre de 7 à 10 jours. Chaque jour, augmentez graduellement la quantité de nourriture PURINA® Friskies® Régal pour chats d'intérieur et réduisez la quantité de l'ancienne nourriture jusqu'à la fin de la période de transition.</p> <p><b>EAU</b> Laissez toujours de l'eau fraîche dans un bol propre à la portée de votre animal. Préservez la santé de votre animal en consultant régulièrement le vétérinaire.</p> <p><b>GARDEZ DANS UN ENDROIT FRAIS ET SEC.</b> Valeur calorigène (calculée): 3 371 kcal/kg • 364 kcal/tasse</p>	POIDS DU CHAT	RATION QUOTIDIENNE*				Nourriture sèche seulement	Nourriture sèche et nourriture en boîte			kg lb	grammes	tasses	grammes	tasses/oz	2 à 4 5 à 9	SÈCHE: 54 à 81	1/2 à 3/4	SÈCHE: 27 à 36	1/4 à 1/3 tasse				HUMIDE: 85 à 170	3 à 6 oz	5 à 6 10 à 14	SÈCHE: 81 à 108	3/4 à 1	SÈCHE: 36 à 71	1/3 à 2/3 tasse				HUMIDE: 170 à 225	6 à 8 oz
WEIGHT OF CAT		DAILY FEEDING AMOUNT*																																																																			
	Dry Food Only	Dry + Canned																																																																			
kg lbs	grams	cups	grams	cups / oz.																																																																	
2 to 4 5 to 9	DRY: 54 to 81	1/2 to 3/4	DRY: 27 to 36	1/4 to 1/3 cup																																																																	
			WET: 85 to 170	3 to 6 oz.																																																																	
5 to 6 10 to 14	DRY: 81 to 108	3/4 to 1	DRY: 36 to 71	1/3 to 2/3 cup																																																																	
			WET: 170 to 225	6 to 8 oz.																																																																	
POIDS DU CHAT	RATION QUOTIDIENNE*																																																																				
	Nourriture sèche seulement	Nourriture sèche et nourriture en boîte																																																																			
kg lb	grammes	tasses	grammes	tasses/oz																																																																	
2 à 4 5 à 9	SÈCHE: 54 à 81	1/2 à 3/4	SÈCHE: 27 à 36	1/4 à 1/3 tasse																																																																	
			HUMIDE: 85 à 170	3 à 6 oz																																																																	
5 à 6 10 à 14	SÈCHE: 81 à 108	3/4 à 1	SÈCHE: 36 à 71	1/3 à 2/3 tasse																																																																	
			HUMIDE: 170 à 225	6 à 8 oz																																																																	

Figura 31: Informacion nutricional de comida para gatos marca Friskies

```

1  if (h > 25000){
2      mostrarCadena(demasiado_peso);
3      lcd.clear();
4      lcd.print("Demasiado_peso");
5      lcd.setCursor(2,1);
6      lcd.print("para_un_gato");
7      mostrarMensaje();
8      encenderLCD();
9  } else if (h >= 15000){
10     mostrarCadena(sobrepeso);
11     EEPROM.write(ADDR_S_DOSIS,
12                 0);
13     lcd.clear();
14     lcd.setCursor(3,0);
15     lcd.print("Tu_gato");
16     lcd.setCursor(0,1);
17     lcd.print("sufre_sobrepeso");
18     encenderLCD();
19     delay(2000);
20     lcd.clear();
21     lcd.print("Por_favor_");
22     lcd.print("visite");
23     lcd.setCursor(0,1);
24     lcd.print("a_un_");
25     lcd.print("veterinario");
26     for (i=0; i<10; i++){
27         lcd.setBacklight(OFF);
28         delay(500);
29         lcd.setBacklight(ON);
30         delay(500);
31     }
32     lcd.clear();
33     lcd.print("No_se_");
34     lcd.print("dispensara");
35 }

36     else{
37         h = h/1000;
38         j=0;
39         //Esta es la dieta segun
40         su_peso;
41         switch (h){
42             case 14: j += 5;
43             case 13: j += 5;
44             case 12: j += 5;
45             case 11: j += 5;
46             case 10: j += 5;
47             case 9: j += 5;
48             case 8: j += 10;
49             case 7: j += 10;
50             case 6: j += 10;
51             case 5: j += 10;
52             case 4: j += 10;
53             case 3: j += 10;
54             case 2: j += 15;
55             case 1: j += 15;
56             case 0: j += 25;
57             break;
58         }
59         S_dosis = j/S_gram;
60         if ((j % S_gram)>(S_gram/2)) {
61             S_dosis++;
62         }
63         if (S_dosis == 0) {
64             S_dosis++;
65         }
66         EEPROM.write(ADDR_S_DOSIS,
67                     S_dosis);
68         mostrarCadena(suministran);
69 #ifdef DEBUG
70         Serial.print(S_dosis);
71 #else
72         BT.print(S_dosis);
73 #endif
74         mostrarCadena(c_dosis);
75         lcd.clear();
76         lcd.print("Le_");
77         lcd.print("corresponden");
78         lcd.setCursor(4,1);
79         lcd.print("dosis");
80         lcd.print(c_dosis);
81         mostrarMensaje();
82         encenderLCD();
83     }

```

Código 2: código ejemplo.

## 4.2. Configuracion

Debido a que es necesario que el usuario pueda modificar el comportamiento del comedero sin tener que recomilar el codigo, tanto para su comodidad, como para disminuir la complejidad de uso del dispositivo, se ha optado por construir una linea de comandos que permita hacer uso de la API del dispositivo de tal forma que se pueda consumir desde una terminal interactiva o se integrado por una aplicacion de terceros.

La api responde a los siguientes comandos: Ultimo: Muestra cuando fue la ultima vez que se alimento al animal, permitiendo al programador, por ejemplo añadir mas momentos de alimentacion dinamicamente en su aplicacion.

Comida: Este comando permite definir la hora a la que el comedero dispensará la comida Hora: Puede ser necesario cambiar la hora del reloj debido a un cambio de hora, de region, o de la pila del reloj. Este comando permite fijar la fecha hora y dia de la semana del reloj.

Dosis: Es posible que el usuario prefiera establecer personalmente la cantidad de dosis que se suministran en cada racion de comida. Para eso, este comando permite fijarlo manualmente.

Gramos: Debido a que el comedero necesita conocer la cantidad de comida dispensada por vuelta, este comando permite fijar la cantidad de comida media que se dispensa por vuelta, permitiendo que el comedero pueda calcular el numero de vueltas necesario para dispensar una dosis en concreto.

Peso: Este comando especifica el peso de la mascota con el fin de calcular automaticamente la cantidad de comida necesaria para alimentar correctamente el animal.

Reset: Este comando permite reiniciar el dispositivo en caso de que el usuario o el programador lo deseen.

Luz: A veces es deseable mantener la luz de la pantalla siempre encendida, pero en condiciones normales supone un consumo que no es necesario, o sencillamente resulta incomodo al iluminar la estancia en la que se encuentra el dispositivo de noche, por tanto los usuarios pueden elegir si la quieren encendida, o por el contrario que solamente se encienda para mostrar mensajes.

Test: Este comando permite realizar una ciclo de alimentacion completo para comprobar si la cantidad suministrada es la correcta, o para comprobar el funcionamiento del dispositivo.

Man: De usarse solo, este comando muestra la lista de comandos a los que responde la API. Tambien puede usarse junto con otro comando para obtener informacion del uso de cada comando.

Para controlar los comandos introducidos se ha escrito una funcion que permite comprobar si la cadena introducida contiene algun comando de los

que el dispositivo conoce, y en tal caso comprobar el resto de argumentos. Esta función, como puede verse en el código 3, tiene como fin limpiar los espacios iniciales y posteriormente comprobar la coincidencia con la lista de comandos devolviendo el comando encontrado en caso de estar presente.

```

1 byte comparaOpcion(char * entrada, String * opciones, byte *
2   longitud, byte cant, byte * p){
3
4   byte i=0, j, k, c;
5
6   while ((entrada [ i ] == ' ') || (entrada [ i ] == '\n'))
7     i++;
8
9   *p = i;
10  for (k=0; k<cant ; k++){
11    c=0;
12    for (j=i; j<i+longitud [ k ] ; j++){
13      if (toupper(entrada[j]) == opciones[k][j-i])
14        c++;
15      if ( c==longitud [ k ] )
16        return (k+1);
17    }
18  }
19  return 0;
}

```

Código 3: Función que limpia la entrada y comprueba el comando recibido.

### 4.3. Aplicacion android

Para mostrar el funcionamiento de la interacción con la API en línea de comandos, y para que el dispositivo se pueda utilizar por usuarios que no estén familiarizados bien con una línea de comandos o con una API se ha escrito una aplicación Android para este fin. Esta aplicación simplifica la conexión con el dispositivo 32, y ofrece herramientas que permiten utilizar el dispositivo sin necesidad de la línea de comandos, a demás de incluir una para introducir a los desarrolladores en el uso del comedero. Esto puede verse en la figura 33

En la figura 33 se puede ver la pantalla de configuración con los elementos gráficos que interactúan con la API y algunos otros que ofrecen información sobre el estado de la conexión. A continuación se detallan.

1. Campo para introducir el peso del animal
2. Campo para introducir el número de raciones manualmente
3. Botón para encender o apagar el ahorro

de energía 4. Registro de mensajes de la interfaz serie 5. Campo para escribir en la interfaz serie 6. Botón para mostrar el menú de la aplicación 7. Icono para mostrar el estado de la conexión bluetooth 8. Campo para configurar la cantidad de gramos por dosis 9. Configurador para cambiar la hora a la que se da de comer 10. Botón para reiniciar el dispositivo 11. Botón para enviar el texto a la interfaz serie

En el fragmento de código 4 se puede ver un ejemplo de como la aplicación móvil forma y manda el comando al dispositivo. Esto le permite a los desarrolladores integrar este proyecto dentro de los suyos propios o de un tercero, haciendo que esta solución pueda formar parte de por ejemplo un sistema domótico.

```
1 pesoInput.setOnEditorActionListener(new
2     TextView.OnEditorActionListener() {
3         @Override
4         public boolean onEditorAction(TextView v, int actionId,
5             KeyEvent event) {
6             String peso;
7             if (actionId == EditorInfo.IME_ACTION_DONE) {
8                 b.send("peso" + String.format("%05d",
9                     Integer.parseInt(pesoInput.getText().toString())));
10                Toast.makeText(getApplicationContext(),
11                    "Peso establecido en: " +
12                    Integer.parseInt(pesoInput.getText().toString()),
13                    Toast.LENGTH_SHORT).show();
14            }
15            return true;
16        }
17    });
18});
```

Código 4: Función para fijar el peso desde la aplicación android.

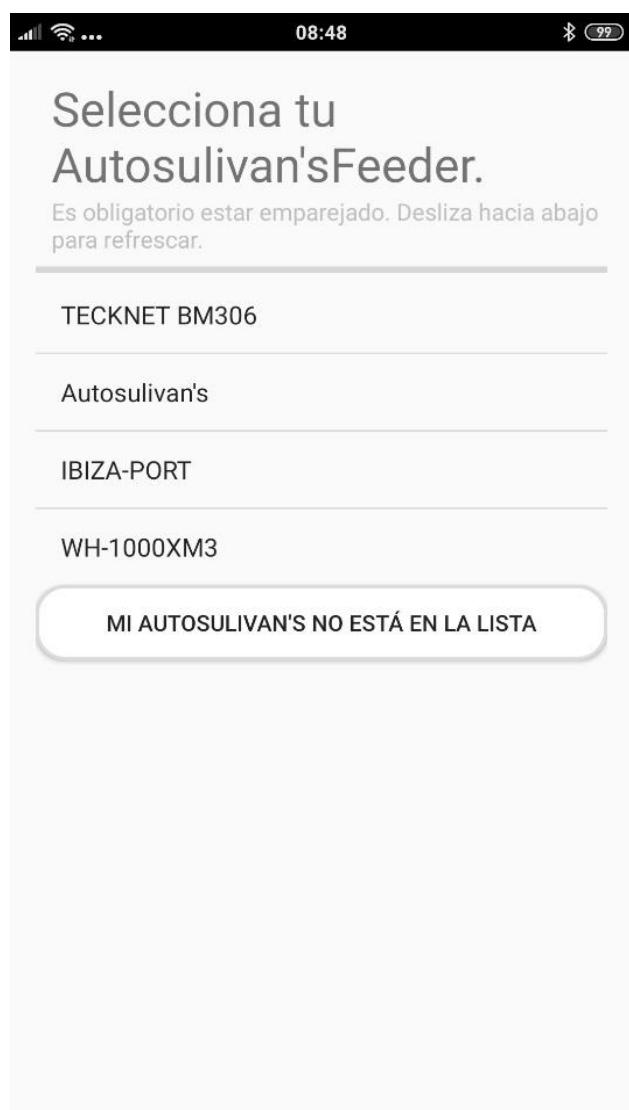


Figura 32: Pantalla de seleccion de dispositivo

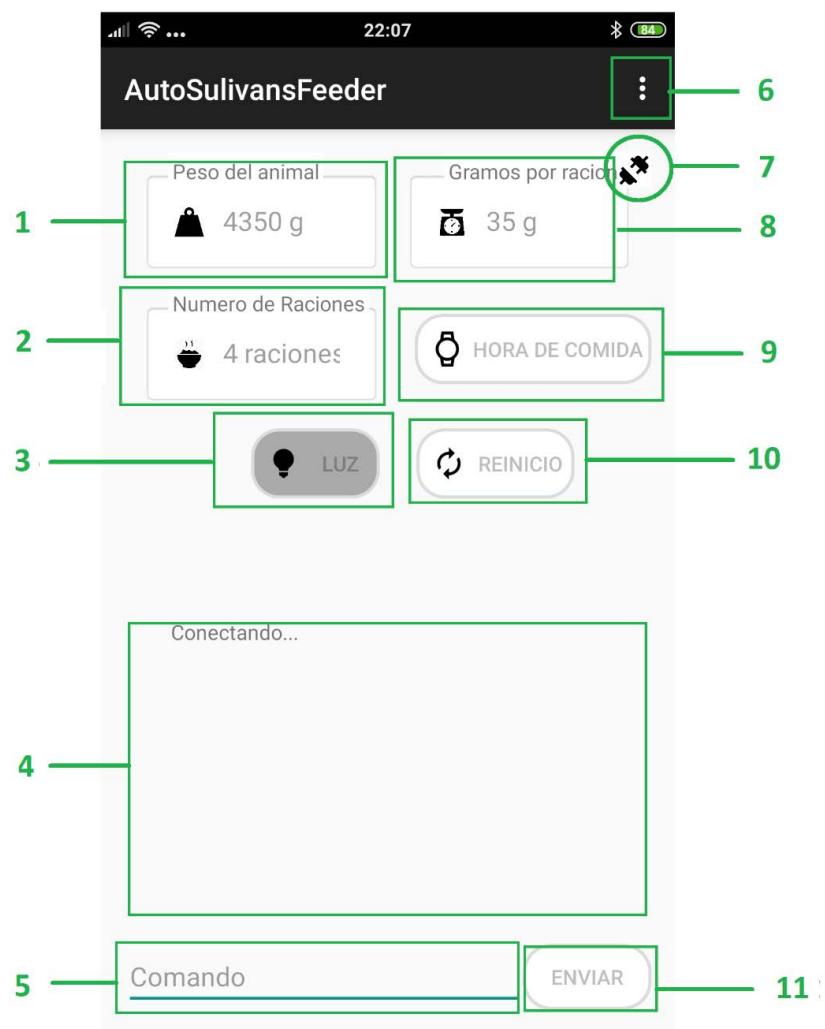


Figura 33: Configurador del comedero

## 5. Aspectos relevantes

En esta sección se describirán aquellos aspectos a destacar del proyecto que por su complejidad, creatividad o por cualquier otra razón resulten de especial interés.

### 5.1. Multihilo

El multihilo o multithreading es una técnica utilizada habitualmente en programación, que permite ejecutar de forma concurrente varios fragmentos de código en el mismo procesador.

Para conseguir esto existen multitud de estrategias, pero básicamente consiste repartir el tiempo de CPU entre las tareas demandantes. Para conseguir esto normalmente es el sistema operativo quien realiza el arbitraje a través de un componente llamado planificador que combinado o no con tecnologías propias del fabricante de hardware (como puede ser el ejemplo del hyperthreading [11] que utiliza una cola de instrucciones doble permitiendo lanzar las instrucciones de una segunda tarea mientras la primera se queda bloqueada esperando a que finalice otra instrucción en ejecución que ocupa una parte diferente del pipeline o que tiene que ser resuelta antes de continuar con la ejecución).

En el caso del microcontrolador elegido para este proyecto, un ATMega328 no existe ninguna estrategia que permita la ejecución multihilo, y por tanto es necesario recurrir a una técnica llamada protothreading.

El protothreading consiste en emular este comportamiento con estrategias definidas por el programador, en lugar de delegar esta tarea en el procesador o el sistema operativo. Para conseguir esto existen muchas estrategias, pero en este caso se ha optado por la temporización de eventos. Esta técnica en concreto se consigue midiendo el tiempo de ejecución del microcontrolador y calculando cuánto tiempo falta para la próxima ejecución de una tarea periódica o programada. En caso de obtenerse un número negativo, significa que esa tarea debe ser ejecutada y el contador de tiempo puesto a cero.

Con esta técnica se consigue la sensación de un multihilo real a pesar de suponer un alto esfuerzo en la programación.

Para implementar esto, en este proyecto se definen los temporizadores y las comprobaciones que pueden verse en el fragmento de código.

Como puede verse en el fragmento de código 5 la dispositivo se reinicia cada 24 horas para evitar que el contador de milisegundos, que es una variable de 32 bits se desborde pasados casi 50 días como puede verse en la documentación oficial [7] y para evitar que un error fatal no previsto pudiera dejar el dispositivo inoperativo.

```

1 #define SPLASH           2.0      //tiempo en segundos
2   que se mostrara el mensaje de bienvenida
3 #define TIEMPO_BRILLO    5.0      //Tiempo en segundos
4   que se mantendra la pantalla encendida al mostrar un
5   mensaje
6 #define TIEMPO_MENSAJE    3.5      //Tiempo que se
7   mantendra en la pantalla el mensaje
8
9 unsigned long millis_actuales = millis();
10
11 if (millis_actuales >= MILLIS_DIARIOS){ //este fragmento evita
12   que el se desborde la variable y ademas soluciona posibles
13   cuelgues restableciendo el funcionamiento una vez al dia
14   resetearComedero();
15 }
16
17 //mostramos la hora en la pantalla si procede
18 if((second != sec_anterior) &&(millis_actuales > mensaje)){...}

```

Código 5: Temporizadores y comprobaciones de tiempo

## 5.2. Herramientas de documentacion

En este apartado se exponen las herramientas utilizadas para generar documentacion y esquemas rapidos enfocados principalmente a un entendimiento rapido pero superficial de como montar y modificar el codigo del proyecto.

### 5.2.1. Doxygen

Doxygen es una herramienta de generacion automatica de documentacion. Esta herramienta puede ser usada tanto para documentar un proyecto, como para servir de ayuda en la lectura del codigo de una aplicacion ya escrita. En la figura se puede ver la pagina de bienvenida de la documentacion generada con esta herramienta para el proyecto de arduino donde cada funcion aparece descrita facilitando el analisis del codigo a otros programadores. En la figura 34 se puede ver una pagina de ejemplo de la documentacion generada con esta herramienta.

### 5.2.2. fritzing

Fritzing [6] es una herramienta (ahora de pago) para describir circuitos electronicos. Es posible construir virtualmente un circuito en una protoboard

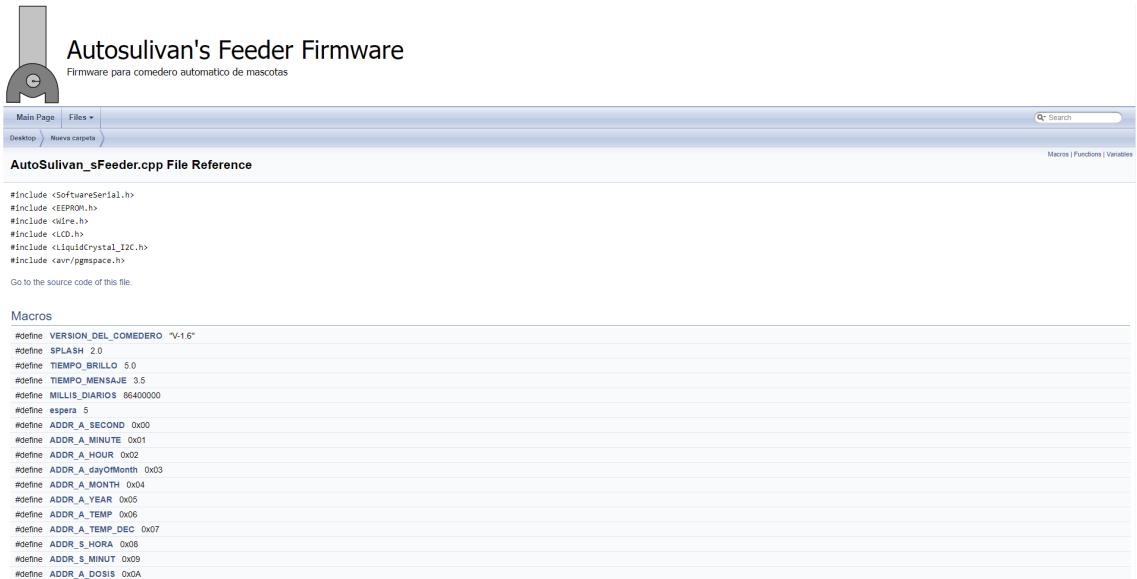


Figura 34: Glosario de funciones generado con doxygen

para despues generar un esquema electrico y una PCB e incluso, dependiendo del microcontrolador, emular el comportamiento. En el caso de arduino, es incluso posible utilizar el hexadecimal generado para ejecutarlo en este entorno virtualizado. Esta es la herramienta utilizada para generar el esquema de la figura 27.

### 5.3. Diseño e impresion 3d

En esta seccion se darán a conocer la cadena de herramientas utilizadas para la obtencion de las piezas fisicas utilizando impresoras 3D de tipo aditivo con tecnologia de deposicion de filamento dundido. Este tipo de impresoras son las mas comunes y por ello tambien las mas baratas. Ademas debido a su popularidad, existe una comunidad amplia y activa que facilita el uso de este tipo de maquinaria. Para este proyecto se ha utilizado una impresora casera derivada de una prusa I3 en este caso de armazón de acero que puede verse en la imagen 35. Este tipo de impresora pertenecen a la comunidad RepRap [16] que provee la ingenieria industrial para el diseño de impresoras 3D quedando al margen de la programacion de estas.

A pesar de que existen multiples firmwares para impresoras 3D, el mas extendido es Marlin [15] originalmente escrito para procesadores AVR AT-Mega 2560 pero actualmente con compatibilidad para ARM de 32 bits. Este firmware destaca por ser el unico capaz de gestionar aceleraciones lineales

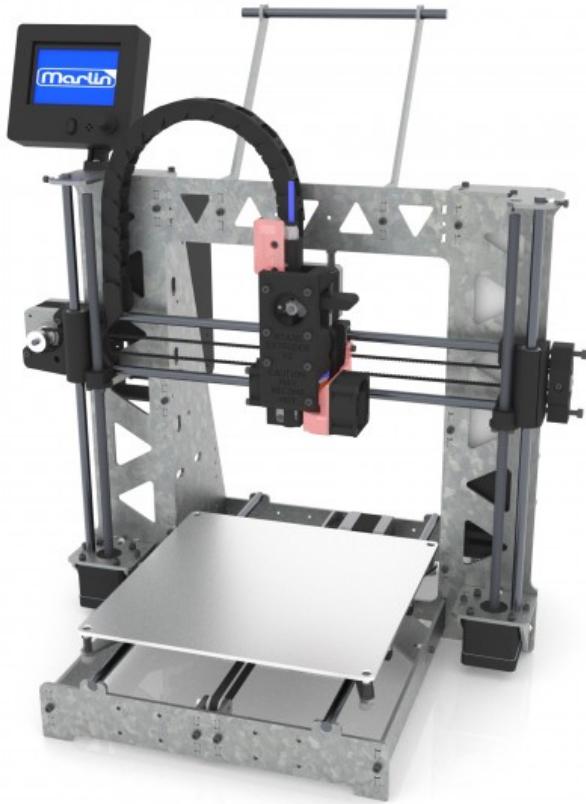


Figura 35: Impresora prusa I3 Steel

y por tener una amplia gama de funcionalidades para diferentes tipos de impresoras y aditamentos.

Para el diseño de las piezas 3D que el diseño original no incluia se ha utilizado inicialmente Catia [17] , el software con mas prestigio actualmente en la industria aeronautica, naval y automotriz. Sin embargo Dasault Systems, el fabricante de este software, lanza a traves de una filial en 1995 un software llamado Solid Works [18] equivalente en prestaciones pero a priori mas sencillo de utilizar aunque enfocado a un mercado diferente donde se requieran mayor numero de partes en un ensamblaje. Su sencillez de uso y su potencia lo han convertido en una de las alternativas principales en la comunidad DIY. Por esta razon, finalmente el desarrollo de piezas en 3D se hace usando este ultimo software CAD-CAM-CAE.

Para la impresion 3D se ha utilizado Cura [19] un software de impresion 3D que puede verse en la imagen con motor de decapado propio, que a pesar de ser privativo, ofrece excelentes posibilidades, permite plugins de la comunidad y es gratuito. Este software esta enfocado a usuarios de cualquier

nivel.

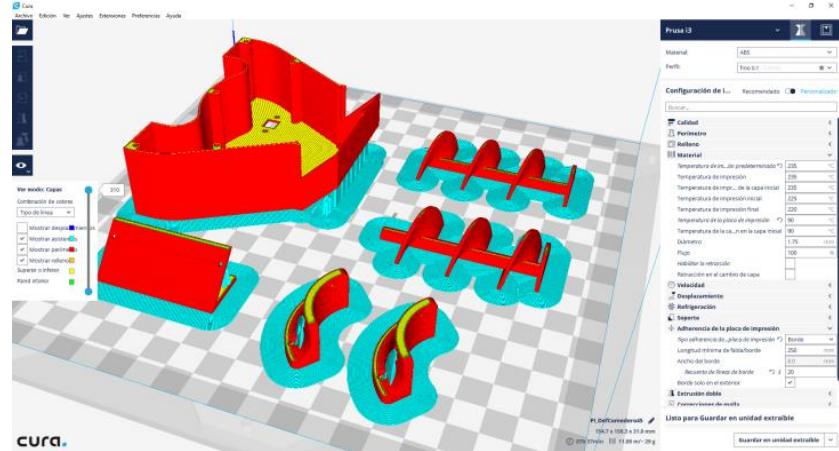


Figura 36: Captura del Ultimaker Cura preparado para imprimir todas las piezas del comedero

#### 5.4. Modelo de explotacion segun la TCUE

Debido a la naturaleza y origen de este proyecto y la competencia con productos fabricados en serie se ha considerado que debe ser distribuido como material abierto bajo la licencia CC-NC-BY. El modelo de negocio se plantea a partir de la venta de kits para montaje, soporte y publicidad en la app android del proyecto. El hecho de vender el producto en kits para montar, permite ahorrar en costes de fabricacion y vender el producto como un juguete educativo, ademas de como un juguete para entusiastas que pueden modificarlo y adaptarlo a sus necesidades. El modelo de explotacion y su correspondiente analisis de mercado cifra el tamaño del mercado en 3,3 millones de niños de entre 8 y 14 años, que son las edades en las que los padres buscan juguetes educativos para sus hijos. Está enfocado a personas con una renta media mensual de entre 1400 y 1600€ lo que hace un total de 2.2 millones de personas de los 3,3 millones anteriormente citados, de los cuales medio millon aproximadamente tiene hijos cifra que ha disminuido a partir de 2018 debido a que la natalidad descendio debido a la crisis economica de 2008. Estos datos y un plan de negocio pormenorizado puede ser consultado en el Anexo 1 de este proyecto.

## 6. Conclusiones y Resultados

Segun los requisitos definidos para este proyecto en el texto punto subsection 1.4 los siguientes objetivos que ha sido pormenorizados en el Anexo 2 serán referenciados y evaluados:

Dosificacion: Este objetivo ha sido definido como OBJ-5 (Posología). Para cumplir este requisito se han descrito los requisitos funcionales ERF16C (Fijar el numero de raciones), ERF18C (Actualizar la cantidad de comida que el comedero dispensa) y ERF19C (calcular automáticamente la cantidad de comida que es necesario suministrar). Estos requisitos han sido cumplidos implementando la funcionalidad descrita.

Durabilidad: Esto se ha conseguido dimensionando correctamente los componentes electrónicos como se explica en el apartado section 4. La durabilidad ha sido probada a lo largo de 60 días de funcionamiento ininterrumpido.

Autonomía: Este requisito ha sido satisfecho con el uso de un contenedor de 2L que puede almacenar comida para, dependiendo del animal, hasta 45 días. El objetivo OBJ-4 ha sido satisfecho con la ERF13C (dar de comer).

Configuración: El objetivo OBJ-7 (Interfaz) ha sido satisfecho utilizando una pantalla y una linea de comandos a través de bluetooth con los requisitos ERF1-11C que detallan el uso de la API.

Integración: El objetivo se ha cumplido con el requisito OBJ-8 (Aplicación gráfica) y con la ERF12C (Introducir comando) que hace posible el uso de la API

Eficiencia energética: Especificado en el objetivo OBJ-9 este objetivo ha sido satisfecho encendiendo y apagando cada componente hardware cuando es necesario utilizarlo.

Usabilidad: Este objetivo ha sido satisfecho con la ERF1-10A que definen las funciones de la aplicación móvil.

No ha sido posible implementar un historial debido a la corta vida de la memoria EEPROM que contiene el procesador, y la cantidad ya moderadamente alta de ciclos de escritura que se realiza y a su limitado tamaño (de solo 1024 bytes) que permitiría como máximo la escritura diaria de 13 bytes para garantizar al menos dos años de funcionamiento, que son insuficientes para mantener un histórico.

Este proyecto ha sido dotado de una beca TCUE para prototipos orientados al mercado. En el marco de esta beca ha sido desarrollado un plan de empresa y una presentación [25] que expone una primera versión de este proyecto.

## **7. Trabajos Futuros**

En esta sección se analizan las posibles líneas de trabajo futuro que podrían complementar el trabajo ya descrito.

### **7.1. Plugin para domoticz**

Domoticz es un software para domótica que permite la integración de nuevos elementos a través de un sistema de plugins escritos en Python. Esta suite puede correr sobre una Raspberry Pi, haciendo que sea barato y sencillo de desplegar, pues es posible obtener una imagen del sistema operativo con la aplicación instalada lista para ser grabada en una memoria SD.

### **7.2. Multiplicidad diaria**

No es estrictamente necesario alimentar a un animal de compañía más de una vez al día, pero hay personas que prefieren hacerlo así, y es posible implementar un sistema que permita añadir tantos eventos de dispensado de comida diarios como se desee, aunque esto haría más complejo el proyecto en cada uno de sus puntos.

### **7.3. Detección de comida**

Este dispositivo no es capaz de saber cuanta comida tiene aun el recipiente, por lo que se podría implementar un segundo dispositivo que mida el peso del comedero y se comunique con este para regular la cantidad de comida que se está suministrando.

### **7.4. Configuración vía internet**

Puede ser deseable hacer cambios en la programación durante los períodos de ausencia y se puede implementar un sistema que permita hacerlo usando un backend y estableciendo un método de comunicación de cada comedero.

### **7.5. Cámara**

Algunos modelos comerciales ofrecen la posibilidad de ver y retransmitir audio y vídeo desde y hasta el comedero con la esperanza para el usuario de poder vigilar e incluso interactuar con su mascota.

## **7.6. Refactorizacion fisica**

Es deseable que el proyecto sea completamente reproducible usando piezas impresas en 3D, pues elimina la dependencia que tienen los usuarios de encontrar piezas compatibles que puedan usar con este proyecto. Seria por tanto importante rediseñar las piezas existentes y crear otras nuevas que eliminan aquellas que no están impresas en 3D o que han tenido que ser cortadas en una maquina de corte con control numérico.

## **7.7. Generacion de logs**

Si fuera posible medir mas variables como por ejemplo la cantidad de comida consumida o agua, seria posible generar un histórico de cuando y cuando come cada mascota y realazar un analisis ulterior de esta información con el fin de obtener conocimiento sobre la conducta del animal que permita optimizar sus hábitos de ser necesario para mejorar su salud.

## **7.8. Nivel de alimento**

Es importante conocer la cantidad de alimento en el deposito del dispositivo, a pesar de que ninguna solución comercial incorpora esta característica.

## **7.9. Batería**

Seria relevante el poder incluir una batería que permita al dispositivo funcionar de forma autónoma durante un tiempo que permita al cuidador del animal volver y que garantice su funcionamiento en caso de perdida del suministro eléctrico.

## **7.10. Multiples mascotas**

Puede ser necesario utilizar el mismo dispositivo para mas de un animal, permitiendo realizar todas las configuraciones para cada uno de los animales que lo utilicen. Para esto se requiere adicionalmente un mecanismo que permita reconocer que animal es el que está haciendo uso de el en ese momento bien sea a través del reconocimiento de imágenes o a con un implante de radiofrecuencia que el animal puede portar bajo la piel.

## Referencias

- [1] A4988 datasheet. [Online]. Available: [https://www.pololu.com/file/0J450/a4988\\_DMOS\\_microstepping\\_driver\\_with\\_translator.pdf](https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf)
- [2] Automatic dog feeder. [Online]. Available: <https://www.instructables.com/id/Automatic-Dog-Feeder-1/>
- [3] Catalogo de integrados para comunicaciones inalambricas de texas. [Online]. Available: <https://www.ti.com/wireless-connectivity/products.html>
- [4] Entorno de desarrollo integrado code composer para procesadores (cc ide). [Online]. Available: <https://www.ti.com/tool/CCSTUDIO>
- [5] Fediaf european statistics. [Online]. Available: <http://www.fediaf.org/who-we-are/european-statistics.html>
- [6] Fritzing landing. [Online]. Available: <https://fritzing.org/>
- [7] Funcion millis() arduino. [Online]. Available: <https://www.arduino.cc/reference/en/language/functions/time/millis/>
- [8] Ili9481 datasheet. [Online]. Available: <https://www.displayfuture.com/Display/datasheet/controller/ILI9481.pdf>
- [9] Ine: Demografía y población. [Online]. Available: [https://www.ine.es/dyngs/INEbase/es/categoria.htm?c=Estadistica\\_P&cid=1254734710984](https://www.ine.es/dyngs/INEbase/es/categoria.htm?c=Estadistica_P&cid=1254734710984)
- [10] Instruction pipeline. [Online]. Available: <https://microchipdeveloper.com/32bit:mz-arch-pipeline>
- [11] Intel hyperthreading. [Online]. Available: <https://www.intel.es/content/www/es/es/architecture-and-technology/hyper-threading/hyper-threading-technology.html>
- [12] Internet-enabled cat ffeeder. [Online]. Available: <http://www.newtonnet.co.uk/catfeeder/>
- [13] Nema 17. [Online]. Available: <http://www.datasheet.es/PDF/928661/17HS4401-pdf.html>
- [14] Pagina de bienvenida de la placa raspberry pi. [Online]. Available: <https://www.raspberrypi.org/>

- [15] Pagina del proyecto marlin firmware. [Online]. Available: <https://marlinfw.org/>
- [16] Pagina del proyecto reprap. [Online]. Available: <https://reprap.org/wiki/RepRap>
- [17] Pagina del software catia. [Online]. Available: <https://www.3ds.com/es/productos-y-servicios/catia/>
- [18] Pagina del software solidworks. [Online]. Available: <https://www.solidworks.com/es>
- [19] Pagina del software ultimaker cura. [Online]. Available: <https://ultimaker.com/es/software/ultimaker-cura>
- [20] Raspbeery pi os. [Online]. Available: <https://www.raspberrypi.org/downloads/raspberry-pi-os/>
- [21] Rtc comparision. [Online]. Available: <https://blog.dan.drown.org/rtc-comparison/>
- [22] stm32f030 devboard with tcxo. [Online]. Available: <https://blog.dan.drown.org/stm32f030-devboard-with-tcxo/>
- [23] Tabla nutricional friskies dry. [Online]. Available: <https://www.walmart.ca/en/ip/friskies-indoor-delights-dry-cat-food/6000071700391>
- [24] Tuit de uno de los creadores de raspberry pi. [Online]. Available: <https://twitter.com/EbenUpton/status/1205906969443393537>
- [25] Video de presentacion del dispositivo en la beca tcue. [Online]. Available: <https://www.youtube.com/watch?v=0UiLpRADHw>
- [26] K. Beck, *Extreme Programming Explained: Embrace Change.* USA: Addison-Wesley Longman Publishing Co., Inc., 1999.