VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Facultad de Ciencias

Grado en Ingeniería Informática

Anexo 3: Especificación de diseño

Comedero automático para Mascotas

Autor:

Álvaro Torijano García

Tutor:

Fernando de la prieta Pintado

Fecha de presentación: Enero 2019

1 INTRODUCCIÓN: 4

2 ÁMBITO DEL SOFTWARE: 5

3 DISEÑO PROCEDIMENTAL: 6

3.1 BÚSQUEDA DEL COMANDO INTRODUCIDO 6
3.2 CALCULO DEL NÚMERO DE RACIONES: 8

4 DISEÑO DEL ENSAMBLAJE ELECTRÓNICO: 13

5 IMPRESIÓN 3D 15

6 REFERENCIA CRUZADA DE REQUISITOS: 16

7 PRUEBAS: 17

7.1 PRUEBAS INDIVIDUALES: 17
7.1.1 Pruebas de sintaxis: 17
7.1.2 Pruebas de ejecución controlada: 17
7.1.3 Pruebas de funcionamiento correcto: 17
7.2 PRUEBAS DE INTEGRACIÓN: 17
7.2.1 Pruebas de comunicación bluetooth: 17
7.2.2 Pruebas de comunicación hardware: 17
7.2.3 Pruebas de comunicación entre activities: 17

8 ENTORNO TECNOLÓGICO 18

8.1 ENTORNO LÓGICO: 18
8.1.1 Android: 18
8.1.2 Arduino: 18

9 PLAN DE DESARROLLO E IMPLEMENTACIÓN: 19

9.1 IMPRESIÓN 3D: 20

1 Introducción:

En este anexo se expone el procedimiento para satisfacer los requisitos que se plantearon antes de empezar el proyecto. Estos requisitos se encuentran en la relación detallada en el anexo 2.

Dado que estos requisitos son variados y de distintos ámbitos, este anexo se dividirá en los siguientes apartados:

Ámbito del software

Ámbito del hardware

Ámbito del firmware

Diseño de la API

Diseño de la interfaz

Diseño procedimental

Referencia cruzada a los requisitos

Pruebas

Entorno tecnológico

Plan de desarrollo e implementación

2 Ámbito del software:

En este apartado se explicará brevemente los objetivos de la aplicación Android así como las restricciones de diseño.

Esta aplicación ha sido diseñada para disminuir la complejidad a la hora de interaccionar con el dispositivo y su API y para eliminar la necesidad de modificar el código para poder configurar el dispositivo.

- La aplicación debe permitir al usuario las siguientes tareas:
- Listar y buscar dispositivos bluetooth emparejados o dentro del rango de alcance de la antena.
- Configurar el dispositivo desde una interfaz grafica
- Mostrar el funcionamiento de la API
- Permitir utilizar la API en línea de comandos

La principal restricción de la aplicación era evitar el desarrollo multiplataforma que de haberse abordado habría acarreado un incremento de la complejidad innecesario.

No obstante, en la plataforma Android existen ciertas limitaciones:

- No es posible saber si un dispositivo emparejado está actualmente conectado.
- No es posible interactuar con los métodos de introducción de texto que proporciona el sistema de forma eficaz (en concreto con el teclado virtual)
- El paso de parámetros entre actividades está limitado a números enteros y a cadenas de texto.

3 Diseño procedimental:

En este aprtado se explicarán como se diseñan las funciones mas complejas. Aquellas cuyo funcionamiento sea trivial o claramente intuitivo serán omitidas.

3.1 Búsqueda del comando introducido

comparaOpcion:

Esta función va a buscar una cadena de un conjunto de cadenas dentro de otra cadena ignorando los espacios y los saltos de línea y sin hacer distinciones entre mayúsculas y minúsculas. Después nos devolverá que cadena encontró y cuantos caracteres tuvo que desechar.

ALGORITMO: comparaOpcion

PARAMETROS:

- Opciones: venctor de cadenas
- Comando: cadena en la que buscar
- Cant: tamaño del vector de cadenas
- Longitud: vector de tamaños de las cadenas de opciones

DEVUELVE:

- K: Opción encontrada
- P: Caracteres desechados

VARIABLES:

- I: posición desde la que empezar a leer de la cadena “comando”
- J: posición en la que se busca dentro de la cadena comando
- K: cadena del vector de opciones que se está buscando
- C: cantidad de coincidencias de la cadena

```
INICIO:
        MIENTRAS (comando[i] es <<espacio>> ó comando[i] es <<salto de línea>>)
                i <- i+1
        FIN MIENTRAS
        P <- i
        PARA K desde 0 hasta cant
                C <- 0
                PARA j desde i hasta longitud[k]
                        SI entrada[j] es opciones[k][j-i]  ENTONCES
                                C <- C + 1
                        FIN SI
                        SI C es longitud [k] ENTONCES
                                DEVOLVER K + 1
                        FIN SI
                FIN PARA
        FIN PARA
        DEVOLVER 0
FIN
```

3.2 Calculo del número de raciones:

Se estima que un gato adulto debe consumir entre 15 y 20 gramos de comida diaria por cada kilogramo de masa. Sin embargo a partir de cierto numero de kilogramos de masa esta cantidad varia, así como los primeros kilogramos que tienen asociados una cantidad de comida mas alta. Esta estimación quizá demasiado generalista también varia dependiendo del fabricante de pienso. Para este código se ha tomado el ejemplo de la comida FRISKIS porque comparte la dosificación con otros fabricantes siendo esta como sigue.

De 0 Kg a 1 Kg: 25 gramos

De 1 a 3 Kg: 15 gramos

De 3 a 8 Kg: 10 gramos

De 8 a 14 Kg: 5 gramos

Siendo estas cantidades acumulativas, por tanto, el algoritmo para calcular la cantidad de comida necesaria puede hacerse fácilmente con la instrucción SWITCH como se muestra a continuación:

ALGORITMO: calculaComida

PARAMETROS:

Peso: peso en gramos del animal

Dosis: gramos por ración que dispensa el comedero

DEVUELVE:

N: Numero de raciones

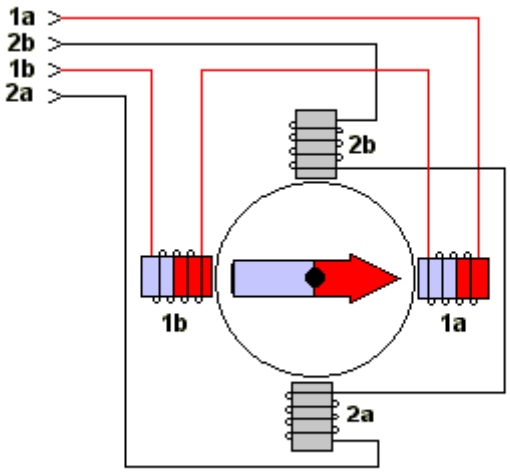
VARIABLES:

Gramos

```
INICIO:
        SEGUN (Peso / 1000) HACER:
                CASO 14
                        Gramos <- Gramos + 5
                CASO 13
                        Gramos <- Gramos + 5
                CASO 12
                        Gramos <- Gramos + 5
                CASO 11
                        Gramos <- Gramos + 5
                CASO 10
                        Gramos <- Gramos + 5
                CASO 9
                        Gramos <- Gramos + 5
                CASO 8
                        Gramos <- Gramos + 5
                CASO 7
                        Gramos <- Gramos + 10
                CASO 6
                        Gramos <- Gramos + 10
                CASO 5
                        Gramos <- Gramos + 10
                CASO 4
                        Gramos <- Gramos + 10
                CASO 3
                        Gramos <- Gramos + 10
                CASO 2
                        Gramos <- Gramos + 15
                CASO 1
                        Gramos <- Gramos + 15
                CASO 0
                        Gramos <- Gramos + 20
        FIN SEGÚN
        N <- Gramos / Dosis
        DEVOLVER N
FIN
```

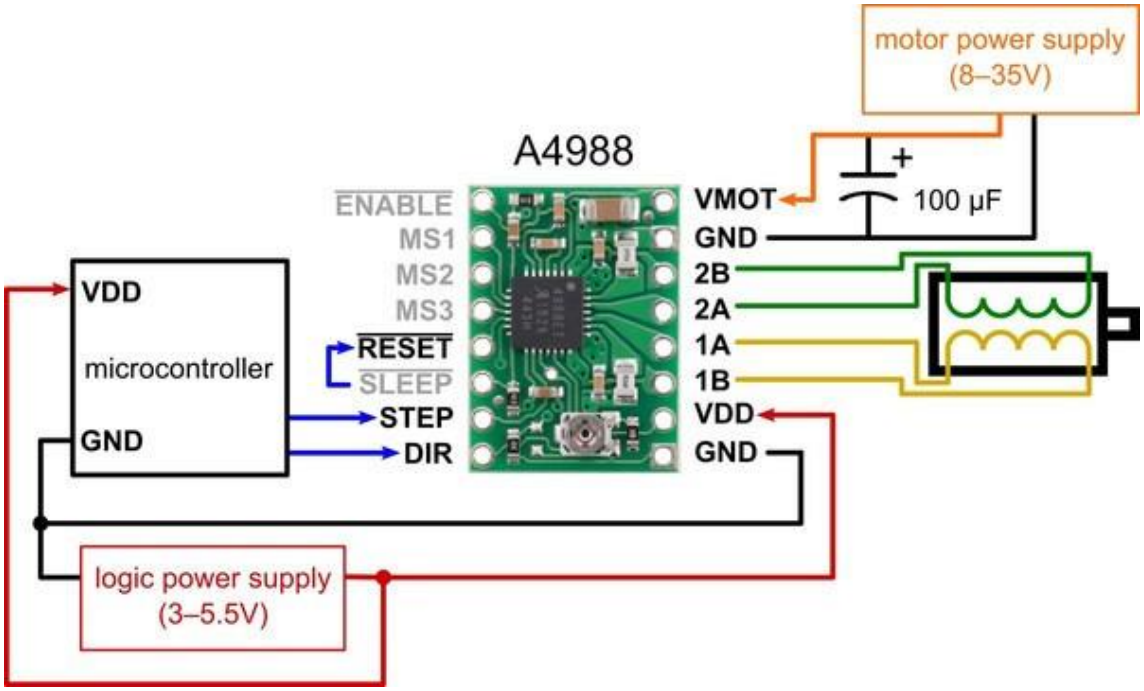
Accionamiento de la tolva:

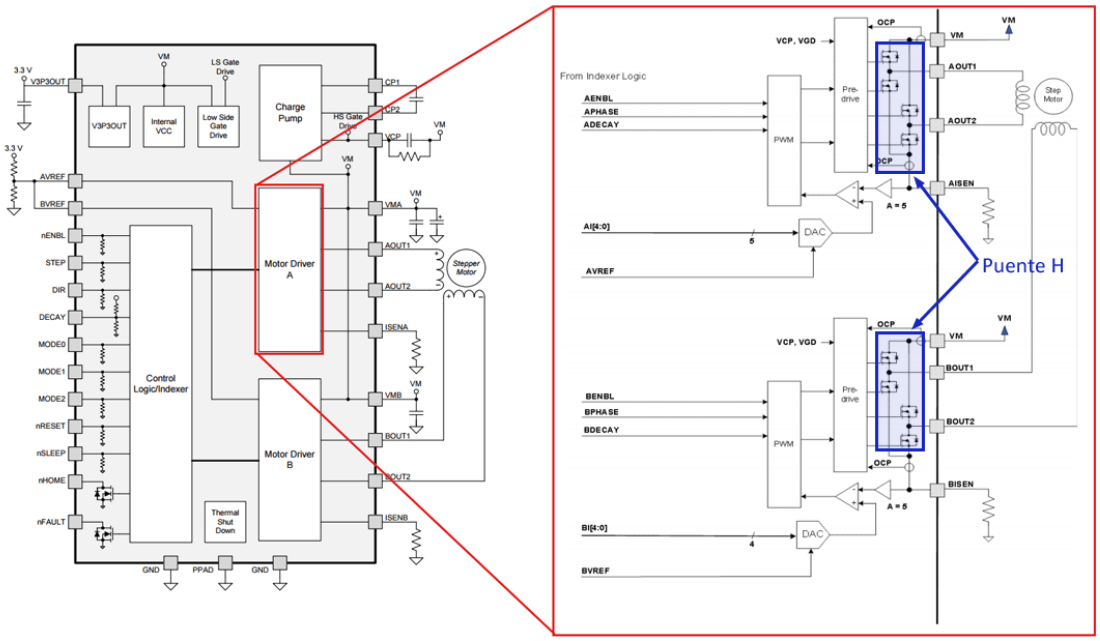
Para accionar la tolva es necesario mover un motor paso a paso. La principal diferencia entre un motor paso a paso y uno convencional de corriente continua con escobillas es que mientras el segundo gira indefinidamente cuando se aplica una diferencia de potencial entre los bornes, para el paso a paso solamente se produce un giro de unos pocos graos. En este caso de 1.8° sexagesimales.



Para controlar este tipo de motores se utilizan habitualmente controladores en puente de H.

En este caso un controlador en doble puente H que además permite posiciones intermedias del rotor cargando parcialmente las bobinas. Concretamente 16 micropasos (lo que equivale a 0.1125° sexagesimales).





ALGORITMO: girarTolva

PARAMETROS:

Vueltas: numero de vueltas que tiene que dar la tolva

Sentido: si la tolva ha de girar en sentido horario o contra horario

DEVUELVE:

Nada

VARIABLES:

I, j: índices para contar pasos y vueltas

```
INICIO:
        SI sentido es Verdadero entonces
                Encender salida(DIR)
        SI NO
                Apagar salida(DIR)
        FIN SI
        encenderMotor
        PARA i desde 0 hasta Vueltas HACER
                PARA j desde 0 hasta 200 HACER
                        Encender Salida(STEP)
                        Esperar (50ms)
                        Apagar Salida(STEP)
                        Esperar (50ms)
                FIN PARA
        FIN PARA
        apagarMotor
FIN
```

4 Diseño del ensamblaje electrónico:

Para este proyecto se ha tratado de utilizar todo lo posible el bus I2C.

El bus I2C es un bus semi síncrono que permite conectar hasta 127 dispositivos a una velocidad de hasta 1.2 Mbps. El bus I2C solamente utiliza 2 hilos, y además en el Arduino estos pines sesituan en las entradas analógicas 4 y 5 (para las líneas SDA y SCL respectivamente).

A este bus se conecta el controlador de la pantalla (LOM1602) para el que se usa la librería New LiquidCrystal de Francisco Malpartida (https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads/)

A este bus se conecta también el modulo de tiempo real DS3231 representado en este esquema por la pieza SO08.

Para comunicarse con este modulo no se utiliza ninguna librería, pero recurriendo al Datasheet del fabricante este es su funcionamiento:

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Los 7 primeros registros nos dan la fecha, hora y día de la semana, y escribiéndolos es posible poner en hora el reloj.

Además este reloj es especialmente preciso porque utiliza un oscilador de cuarzo compensado por temperatura (TCXO: Temperature Compensated Crystal Oscilator) cuya temperatura se puede leer de los registros 0x11h – 0x12

El modulo de bluetooth es un HC-05 que se comunica por interfaz UART.

La interfaz UART es un puerto que utiliza dos hilos, asíncrono, y con un solo dispositivo por bus.

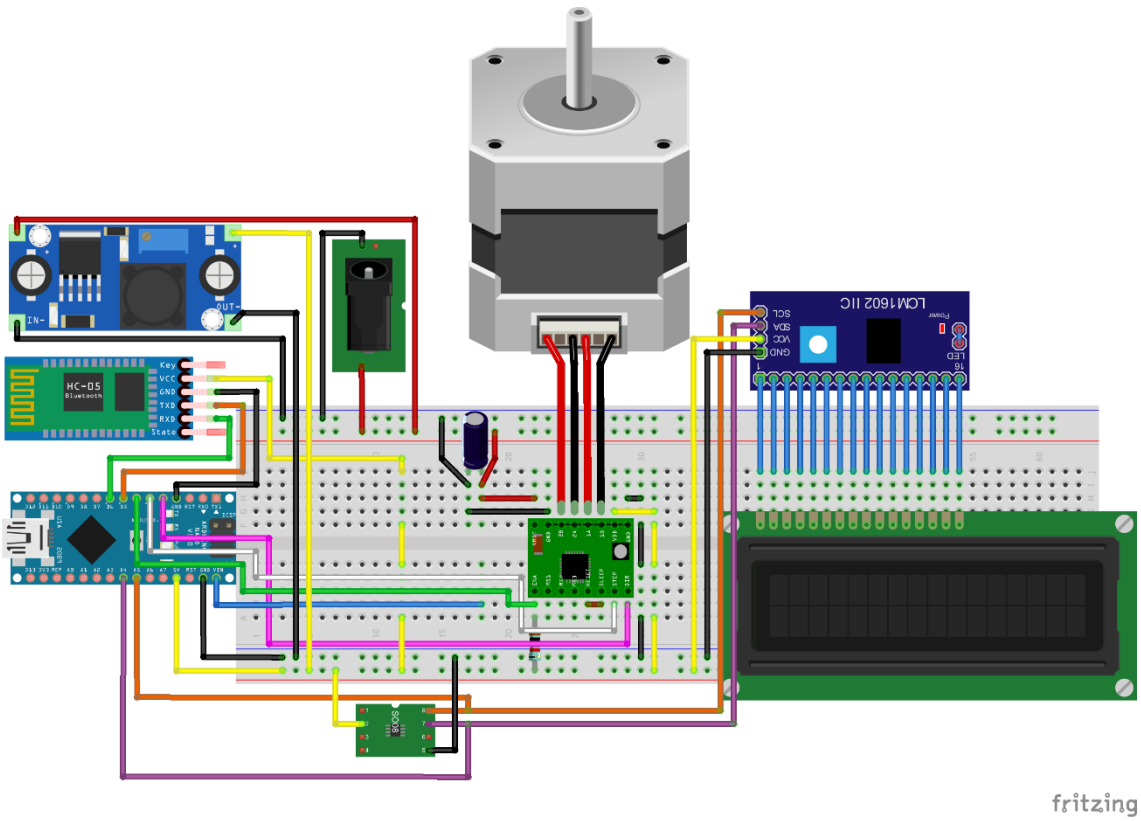
A pesar de que este modulo no necesita una librería especifica, su funcionamiento se puede configurar con la siguiente lista de comandos AT.

Un comando AT es aquel que sigue la forma: “AT+CMD ARG”. Habitualmente son la forma de configurar sobre todo dispositivos MODEM.

- AT : Ceck the connection.
- AT+NAME : See default name
- AT+ADDR : see default address
- AT+VERSION : See version
- AT+UART : See baudrate
- AT+ROLE: See role of bt module(1=master/0=slave)
- AT+RESET : Reset and exit AT mode
- AT+ORGL : Restore factory settings
- AT+PSWD: see default password

La alimentación se realiza con un modulo LM2596 que es un conversor DC-DC y porciona hasta 3 amperios.

Es importante mencionar también la resistencia en configuración PULL-UP que se utiliza en la patilla ENABLE del “pololu” para evitar que el motor se encienda y gire mientras el bootloader del Arduino se carga.



5 Impresión 3d

Para la impresión 3d se han utilizado los siguientes parámetros:

Carcasa: (Makerbot replicator)

Material: PLA

Temperatura: 215 °C

Temperatura de la placa de impresión: Apagada

Superficie de adhesión: Ninguna

Sin soportes

Grosor de capa: 0.1mm

Recuento de líneas de borde: 3

Relleno: 25%

Patron de relleno: Rejilla

Grosor Superior e inferior: 0.6mm

Tornillo y soportes: (Prusa I3 Steel)

Material: ABS

Temperatura: 235 °C

Temperatura de la placa de impresión: 90°C

Superficie de adhesión: Borde

Sin soportes

Grosor de capa: 0.2mm

Recuento de líneas de borde: 3

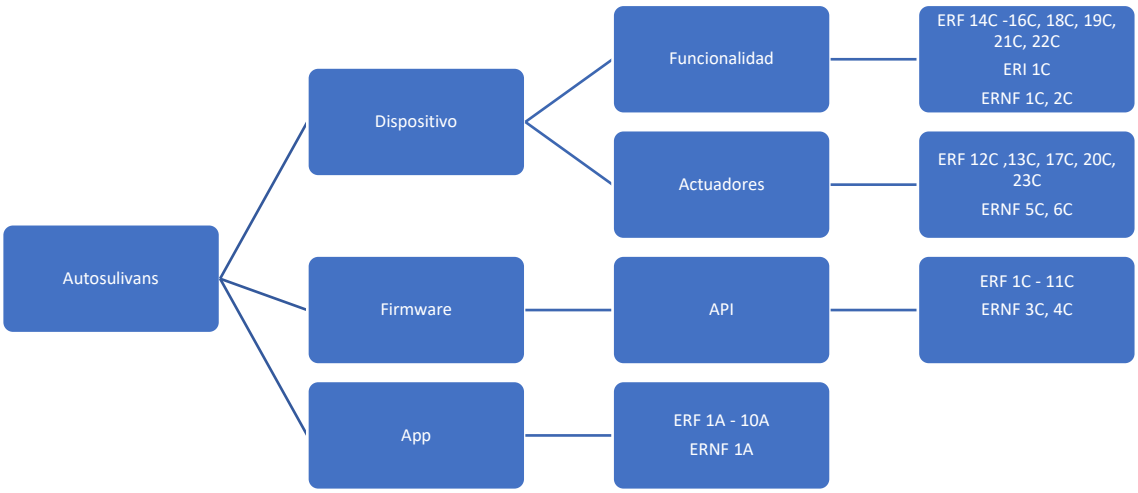
Relleno: 100%

Patron de relleno: Rejilla

Grosor Superior e inferior: 0.6mm

6 Referencia cruzada de requisitos:

Previamente se han mencionado los requisitos necesarios para el éxito del proyecto. Estos requisitos se resumen y clasifican a continuación por categorías:



Analizando este esquema se pude ver fácilmente que todos los requisitos se han satisfecho.

7 Pruebas:

Estas son las pruebas que se realizan para comprobar el correcto funcionamiento del sistema:

7.1 Pruebas Individuales:

7.1.1 Pruebas de sintaxis:

Consiste en garantizar que cada comando y sus argumentos contienen datos validos y que están expresados siguiendo la sintaxis exigida por la API

7.1.2 Pruebas de ejecución controlada:

Este tipo de pruebas consisten en comprobar que no se produzcan errores durante la ejecución que comprometan la estabilidad de la aplicación o el dispositivo.

7.1.3 Pruebas de funcionamiento correcto:

Estas pruebas tratan de comprobar que cada operación hace exactamente lo que se espera de ella.

7.2 Pruebas de integración:

Este tipo de pruebas están destinadas a comprobar que todas las partes del sistema funcionan de forma conjunta sin problemas o conflictos:

7.2.1 Pruebas de comunicación bluetooth:

El objetivo de estas pruebas es determinar si la comunicación se realiza correctamente y realizar el control de errores.

7.2.2 Pruebas de comunicación hardware:

Durante estas pruebas se comprueba que todos los componentes del sistema se comuniquen correctamente con el microcontrolador en primer lugar uno a uno, y posteriormente todos a la vez (montados en un mismo circuito, no simultáneamente).

7.2.3 Pruebas de comunicación entre activities:

El objetivo de estas pruebas es determinar si la comunicación entre las diferentes activities de la aplicación Android se realizan correctamente.

8 Entorno tecnológico

8.1 Entorno Lógico:

8.1.1 Android:

Para que la aplicación se ejecute es necesario contar con un dispositivo Android con al menos la versión 4.3.3 y con conectividad bluetooth. Para instalar esta aplicación es necesario contar con al menos 1.5Mb para el archivo de instalación y 3.5Mb disponibles para la instalación (aunque este ultimo tamaño puede variar entre versiones de Android. Datos obtenidos de Android 9 + MIUI 10.4.2).

8.1.2 Arduino:

Este proyecto puede correr en absolutamente cualquier dispositivo que pertenezca a la familia de microcontroladores ARDUINO siempre y cuando cuenten con los buses y pines necesarios para poder hacer el ensamblaje y cuente con al menos 22 KB de memoria flash, 32Bytes de EEPROM y 2KB de memoria principal.

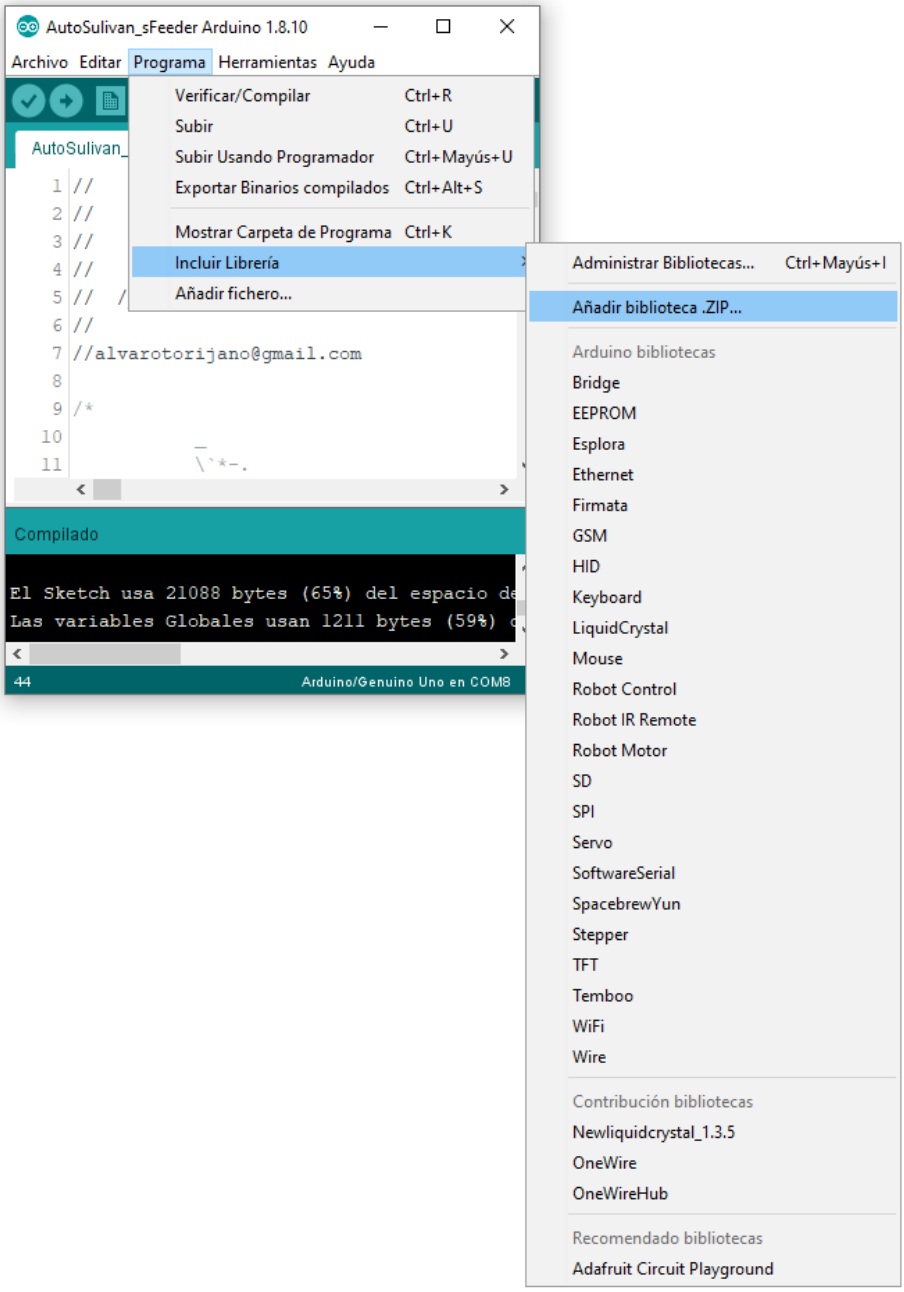
9 Plan de desarrollo e implementación:

Hardware.

Será necesario instalar el IDE de Arduino que se puede descargar desde este enlace.

Una vez instalado será necesario descargar la librería New Liquid Crystal de este enlace adjunta también en la carpeta de Arduino del proyecto.

Una vez descargada es necesario instalar la librería desde el IDE de Arduino. Esto se hace en Programa -> Incluir Librería -> Añadir Biblioteca .ZIP



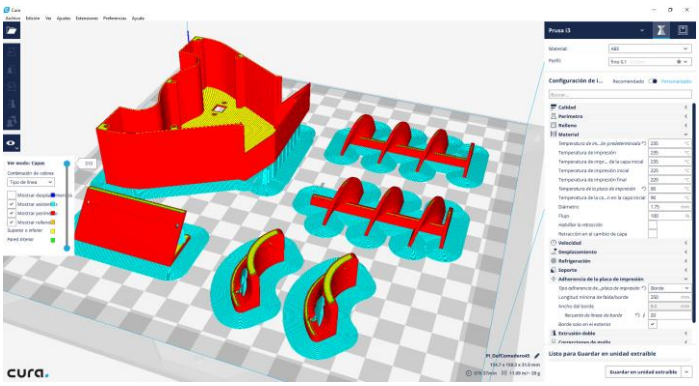
Hecho esto solamente es necesario abrir el archivo AutoSulivan_sFeeder.ino y pulsar en el botón subir o usar la combinación de teclas CRTL + U.

9.1 Impresión 3D:

Para la impresión 3D se ha utilizado la técnica de deposición de material fundido (FDM o FFF).

Para ello en este proyecto se usó CURA y una impresora RepRap, en concreto una prusa I3 steel y una makerbot replicator.

Los ajustes varían ampliamente dependiendo de las preferencias de cada persona, la maquina y el acabado que se quiera obtener. Los ajustes utilizados en este proyecto están descritos arriba, por lo que ahora solo se mostrará una captura de las piezas listas para imprimirse:



Android:

Instalar la aplicación en un dispositivo Android es trivial. Basta con abrir el archivo .apk desde el propio dispositivo para que se lance el instalador del sistema. Explicado detalladamente en el manual de usuario.