UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales

Sistemas de Computación

TRABAJO PRÁCTICO FINAL

Choquevilca Gustavo

Sosa Ludueña Gabriel

Córdoba, 2018

Índice

1. Introducción

Este proyecto se basa en la creación de un modelo del Automatic Pet Feeder (Alimentador de mascotas automático) para su uso doméstico, utilizando para la creación del prototipo el lenguaje de modelado de alto nivel SysML.

El prototipo del Pet Feeder nos pareció una idea muy interesante ya que se puede llevar su lógica a un gran número de escenarios en donde puede ser aplicable la automatización de recarga de alimentos como cereales, semillas, harinas, azúcar, sal y entre muchos más ejemplos, lo que lo hace un modelo ideal para diferentes temáticas ya sea en lo doméstico, en lo comercial y hasta en lo industrial.

En nuestro caso el prototipo planteado se basa en poder llevar un simple proceso físico (visto en la sección 2.1 del marco teórico del presente informe) a su interacción con la internet para poder realizar un control del dispositivo de manera remota desde cualquier parte del mundo. Con este prototipo tendremos una primer experiencia en la que vinculamos internet con un dispositivo (sistema embebido) contribuyendo al IoT (Internet of Things o el internet de las cosas) de manera que nos contribuye a nuestro aprendizaje y es un modelo que servirá como práctica de laboratorio e investigación para futuras implementaciones.

El lenguaje elegido para realizar el sistema es SysML, que es un lenguaje gráfico de alto nivel que permite modelar el sistema a estudiar a través de diagramas, simplificando la lectura del sistema incluso en sistemas muy complejos. Dispone de un fuerte sistema de reglas para garantizar la legibilidad de los sistemas, a la vez que permite cierta flexibilidad a la hora de modelar.

2. Marco Teórico

2.1. Pet Feeder

El *Pet Feeder*, o alimentador de mascotas automático, es un dispositivo que sirve para recargar el alimento a la mascota automáticamente siempre que el recipiente de carga tenga alimento.
Las partes del pet feeder son:

- Base del alimentador.
- Recipiente de alimento.
- Plato de alimento.

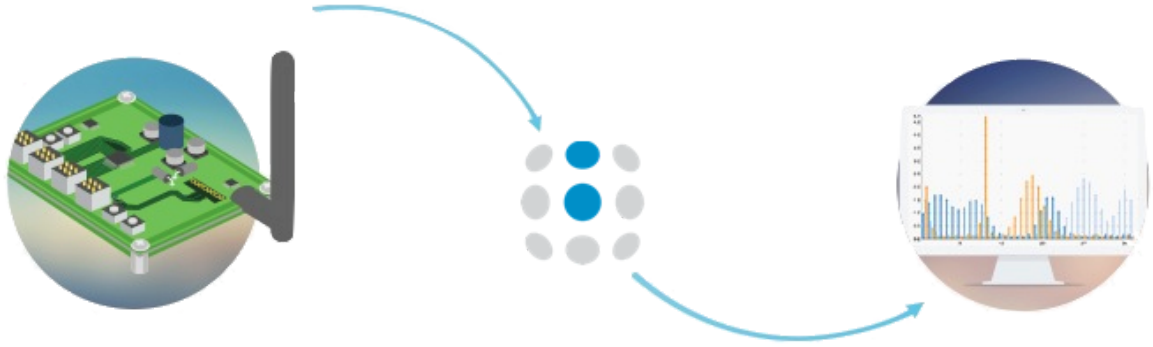
Un ejemplo del producto es como el de la siguiente imagen:

Como se nota el producto es fijo, nuestra idea es automatizar al mismo para que tenga recarga automática y poder visualizar variables de control, para hacer esto hay varias formas con el agregado de uno o más servo motores y vinculando los códigos de programas a internet.

2.2. Internet de las cosas con Ubidots

La facilidad de programación que nos brindan las diferente plataformas de hardware actuales, nos invitan a pensar en una gran cantidad de posibilidades. Por ejemplo, gracias a un dispositivo como el Arduino, es posible medir variables del entorno tales como temperatura, aceleración, posicionamiento, nivel de luz, e incluso cualquier otro sensor que queramos conectar a través de sus puertos análogos.
El desarrollo de éste tipo de proyectos que leen información del ambiente a través de sensores se conoce como el “Internet de las Cosas” y es una tendencia tecnológica que promete crear una nueva era de aplicaciones que revolucionarán la forma en que vivimos. Tal como lo hicieron las aplicaciones web en los años 90s, las redes sociales en la década

del 2000, o las aplicaciones móviles en la presente década, el Internet de las Cosas es una oportunidad para crear innovadoras soluciones en diversos sectores como el comercio, la energía, el transporte, la manufactura, el agro e incluso el hogar.
Para un completo desarrollo de éste nuevo tipo de aplicaciones, debemos trabajar en tres diferentes niveles: la electrónica, las comunicaciones y el software.
Ubidots[1] nos apoyará en el tercer nivel, es decir en la visualización y análisis de los datos.
Ubidots es un servicio en la nube que nos permite almacenar e interpretar información de sensores en tiempo real, haciendo posible la creación de aplicaciones para el Internet de las Cosas de una manera fácil, rápida y divertida.



Gracias a ésta herramienta, podremos ahorrarnos tiempo y dinero al momento de desarrollar aplicaciones como sistemas de telemetría GPS, sistemas para monitoreo de temperatura, aplicaciones para contar vehículos en una calle, etc.
En la siguiente gráfica se ilustra el ahorro en tiempo y esfuerzo al crear una aplicación de Internet de las Cosas con la plataforma Ubidots, o sin ella:

Sin Ubidots	Con Ubidots
Varias líneas de código en cada dispositivo.	Solo necesitamos una línea de código en cada dispositivo.
Desplegar servidores web.	
Definir y programar un protocolo entre los dispositivos y la base de datos.	
Mantenimiento de la base de datos a medida que crece su tamaño.	
Desarrollo de integración con servicios de terceros para envío de SMS/emails.	
Desarrollo y diseño gráfico de interfaz web.	
Desarrollo de dashboard con gráficas, mapas e indicadores.	
Preocuparse por la seguridad de datos.	
Alto costo y tiempo de desarrollo.	

Una gran ventaja de Ubidots es que ofrece un plan gratis, con el cuál podemos realizar prototipos y aplicaciones 100% funcionales.
Para comenzar tendremos que conectar nuestros dispositivos al API de Ubidots, pero antes de ello vamos a repasar algunos conceptos de la plataforma:

- ***Data Source:*** Una fuente de datos se refiere a un dispositivo. Cada Data Source puede tener uno o más sensores o variables. Por ejemplo, en una aplicación de transporte un vehículo sería un Data Source, y sus Variables serían “Velocidad”, “GPS” o “RPM”.
- ***Variable:*** Una variable es un conjunto de datos que cambia en el tiempo. Por ejemplo, las Variables de un Data Source llamado “Refrigerador” serían “Temperatura” y “Humedad”.
- ***Value:*** Es el valor medido por el sensor en un instante de tiempo determinado.
- ***Event:*** Los eventos son acciones que podemos tomar según el valor de nuestras variables. Por ejemplo, podemos configurar un evento para recibir un SMS si la velocidad de un vehículo es mayor que 100 Kmh.

Ubidots permite interactuar con cada uno de éstos elementos de una manera programática, es decir, éstos elementos pueden ser creados, modificados o eliminados a través de programas de hardware o software a través de una API.

3. Requerimientos

A continuación definimos los requerimientos funcionales y no funcionales de nuestro sistema al cual denominamos *APF (Automatic Pet Feeder)*.

3.1. Requerimientos funcionales

Los Requerimientos Funcionales (RF)[2] de un sistema o producto de software y/o hardware se refieren a lo que un sistema debe hacer. Por lo cual tales requerimientos van a depender del tipo de software/hardware que estemos desarrollando, de los usuarios que usaran este sistema o producto y del enfoque, según la experiencia que adaptan las organizaciones (o desarrolladores), cuando se escriben los requerimientos.
Para el desarrollo de nuestro sistema tenemos los siguientes requerimientos funcionales:

Requerimiento	Descripción
RF1	El *APF* debe permitir cargar alimento automáticamente por medio de un pulsador físico y desde una aplicación de software.
RF2	La aplicación de software del *APF* podrá ser accedida desde internet por medio de cualquier dispositivo con acceso a internet.
RF3	El *APF* debe permitir realizar más de una cargas de alimento de forma automática por medio de la aplicación.
RF4	El *APF* debe permitir aplicar cargas fijas cada ciertos intervalos de tiempo. Cuando se encuentre activado este modo se deberá indicar tal situación tanto por hardware como por software.
RF5	El *APF* debe indicar al usuario el porcentaje que resta en su depósito.
RF6	El *APF* debe notificarnos por sms cuando el recipiente de carga contenga solo el 33% o menos de carga.

3.2. Requerimientos no funcionales

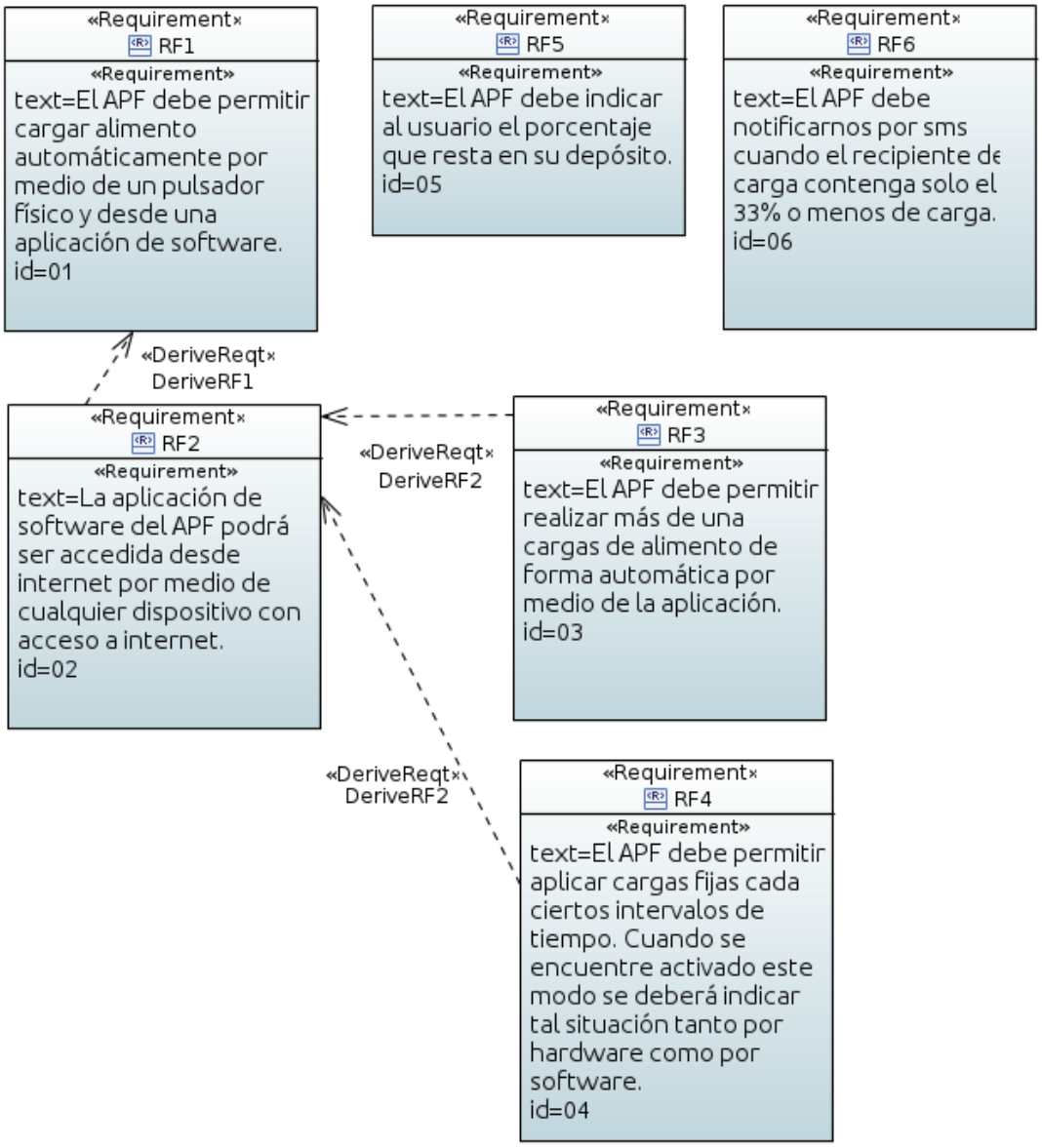
Los Requerimientos No Funcionales (RNF)[2], como indica su nombre, son los requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios. En general son requerimientos que pueden relacionarse con propiedades emergentes del sistema, como la fiabilidad, tiempo de respuesta y uso de almacenamiento. De forma alternativa pueden definir restricciones sobre la implementación del sistema, como las capacidades de los

dispositivos I/O o las representaciones de datos usados en las interfaces con otros sistemas. Por otro lado los requerimientos no funcionales como el rendimiento, la seguridad y la disponibilidad especifican o restringen por lo general características del sistema como un todo. Para la primera versión de nuestro sistema *Pet Feeder* planteamos los siguientes requerimientos no funcionales, que a su vez se clasifican en requerimientos del producto (P), de la organización (O) y externos (E).

REQUERIMIENTO	DESCRIPCIÓN
RNF1 (P)	El *APF* deberá admitir un modo de control físico a través del hardware. Este modo estará siempre habilitado y puede estar en combinación con cualquier otro modo.
RNF2 (P)	El *APF* deberá admitir un modo de control local a través de una computadora por software. Este modo será habilitado desde el software de la computadora y podrá habilitarse en conjunto con el modo físico por hardware.
RNF3 (P)	El *APF* deberá admitir un modo de control remoto a través de internet. Este modo será habilitado desde el software de la computadora y podrá habilitarse en conjunto con el modo físico por hardware.
RNF4 (O)	El hardware del sistema embebido del *APF* deberá controlar el circuito eléctrico mediante una placa arduino uno.
RNF5 (O)	El software del sistema embebido del *APF* se programara por la IDE de arduino.
RNF6 (O)	La aplicación de interfaz entre una computadora y el APF deberá ser programada en python.
RNF7 (O)	La comunicación entre el sistema embebido y la aplicación de la computadora del *APF* será mediante UART.

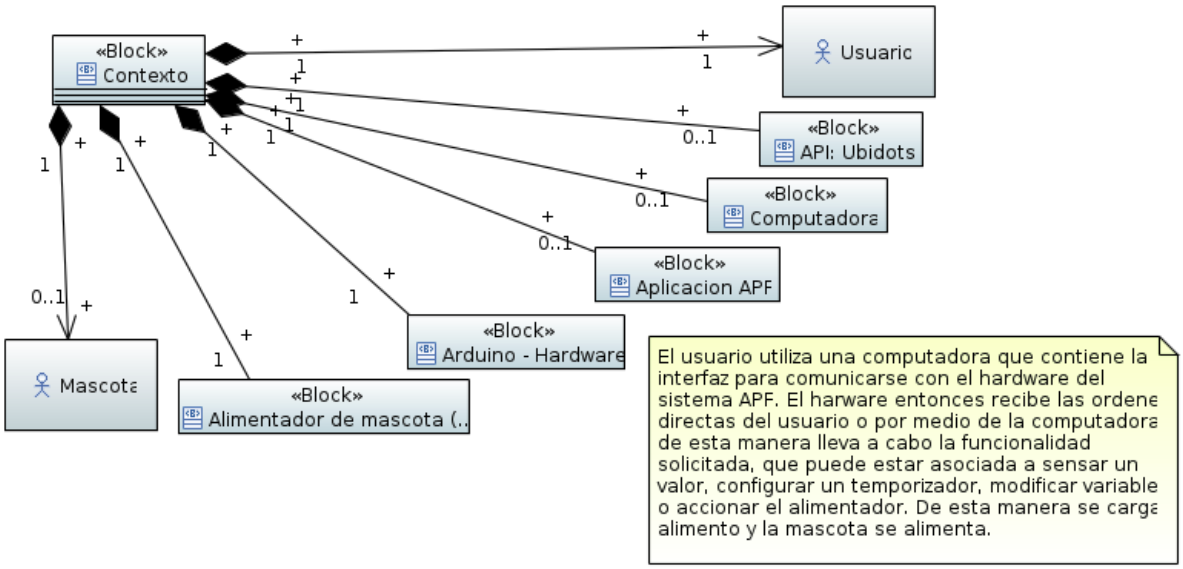
3.3. Diagramas de requerimientos

A continuación podemos observar la relación entre cada uno de nuestros requerimientos funcionales en el siguiente diagrama de requerimientos funcionales.


«Requirement»
RF1
«Requirement»
text=El APF debe permitir cargar alimento automáticamente por medio de un pulsador físico y desde una aplicación de software.
id=01
«Requirement»
RF5
«Requirement»
text=El APF debe indicar al usuario el porcentaje que resta en su depósito.
id=05
«Requirement»
RF6
«Requirement»
text=El APF debe notificarnos por sms cuando el recipiente de carga contenga solo el 33% o menos de carga.
id=06
«DeriveReqt»
DeriveRF1
«Requirement»
RF2
«Requirement»
text=La aplicación de software del APF podrá ser accedida desde internet por medio de cualquier dispositivo con acceso a internet.
id=02
«DeriveReqt»
DeriveRF2
«Requirement»
RF3
«Requirement»
text=El APF debe permitir realizar más de una cargas de alimento de forma automática por medio de la aplicación.
id=03
«DeriveReqt»
DeriveRF2
«Requirement»
RF4
«Requirement»
text=El APF debe permitir aplicar cargas fijas cada ciertos intervalos de tiempo. Cuando se encuentre activado este modo se deberá indicar tal situación tanto por hardware como por software.
id=04

4. Diagrama de bloques y componentes del sistema

El entorno de nuestro sistema se observa en el siguiente diagrama de bloques [3].



Detalles: Para representar el dominio completo de nuestro caso de estudio, creamos un bloque raíz denominado contexto. Sus elementos son:

- Un alimentador de mascotas (sistema estudiado y que es automatizado). Siempre existe ya que es el dispositivo automatizado por el sistema APF.
- Usuario, quien se encarga de manejar y configurar el sistema en cuestión. Siempre que el sistema sea utilizado existe el usuario.
- Mascota, que será quien hará uso final del alimentador de mascota. La mascota es el usuario final que hará o no uso del dispositivo cuando lo requiera.
- Arduino - Hardware: es la unidad de control encargada de interactuar con el alimentador y un usuario. A la vez permite extender sus funcionalidades comunicándose con un computadora. Existe durante todo el periodo de vida en el que se inicia el sistema.
- Aplicación APF: Programa de software encargado de conectar e interpretar una computadora. Es la interfaz necesaria para conectar el sistema embebido (arduino-hardware) con una computadora y de esta forma poder extender sus funcionalidades como permitir acceso a internet. Existe en el contexto si la computadora es utilizada.
- Computadora: es la unidad encargada de extender las funcionalidades del sistema embebido mediante comunicación UART. También sirve como unidad de control ya que desde la misma se puede correr la aplicación APF para configurar y controlar el sistema completo. Esta no siempre está presente a lo largo de la vida del sistema, puede que inicie sin la misma y funcionar con los parámetros por defectos, o bien, conectarse para extender las funcionalidades y configurar adecuadamente el sistema completo.

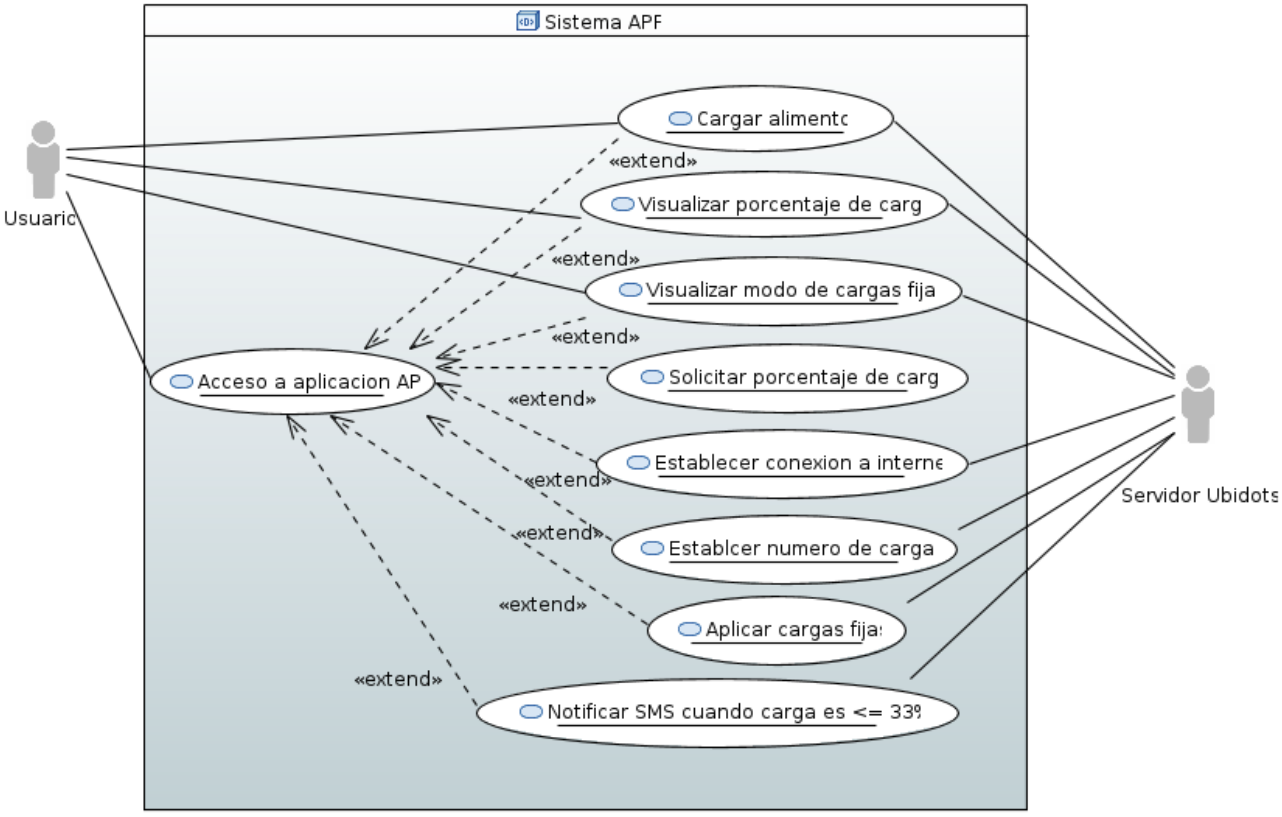
- API Ubidots: es un conjunto de funcionalidades extras que se agregan a la aplicación del APF para, entre otras cosas, permitir acceso a internet y mantener una conexión con servidores.

Nota: decidimos representar el sistema externo mediante bloques mientras que las personas y animales están representadas por actores, de modo que bloques y personas son visualmente diferentes.

5. Modelo de comportamiento del sistema

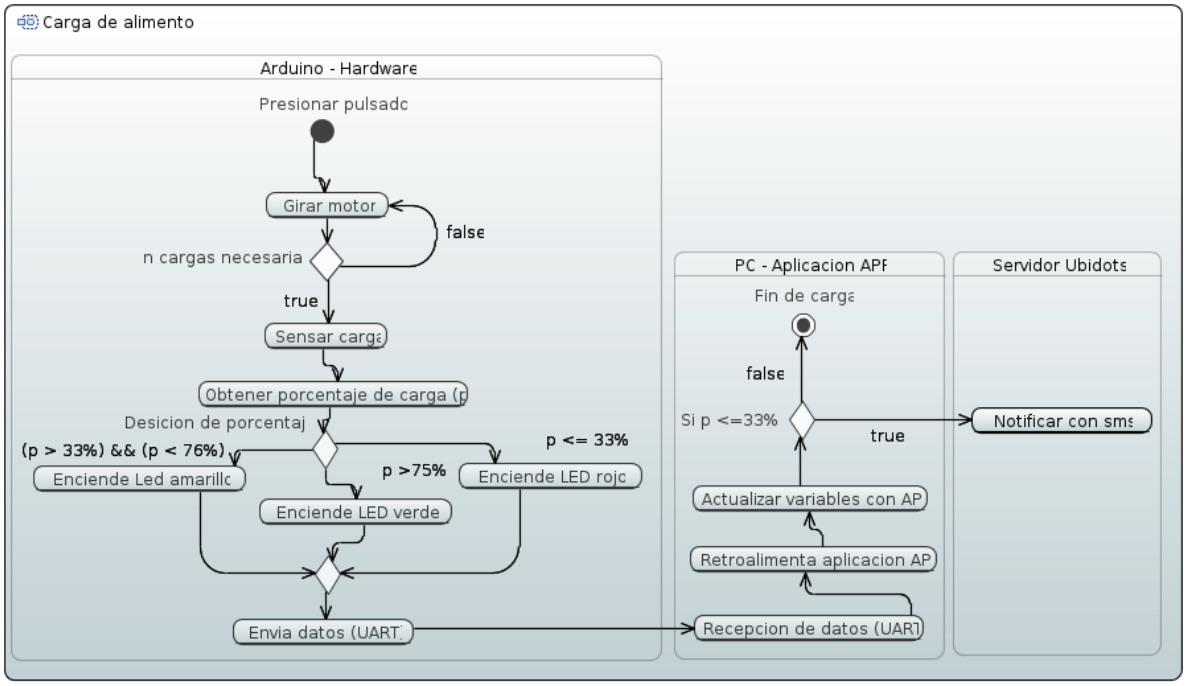
5.1. Casos de uso

Una vez conocidos los requerimientos del sistema se construye un modelo de requerimientos utilizando casos de uso. El sistema APF posee dos actores y nueve caso de usos. Inicialmente podemos observar el comportamiento dinámico de nuestro sistema a través de este diagrama de casos de uso.



5.2. Diagrama de actividades

Otro diagrama que muestra el comportamiento de nuestro sistema es el diagrama de actividad. A continuación mostramos dicho diagrama para nuestro caso de estudio al momento de solicitar una carga de alimento al sistema.

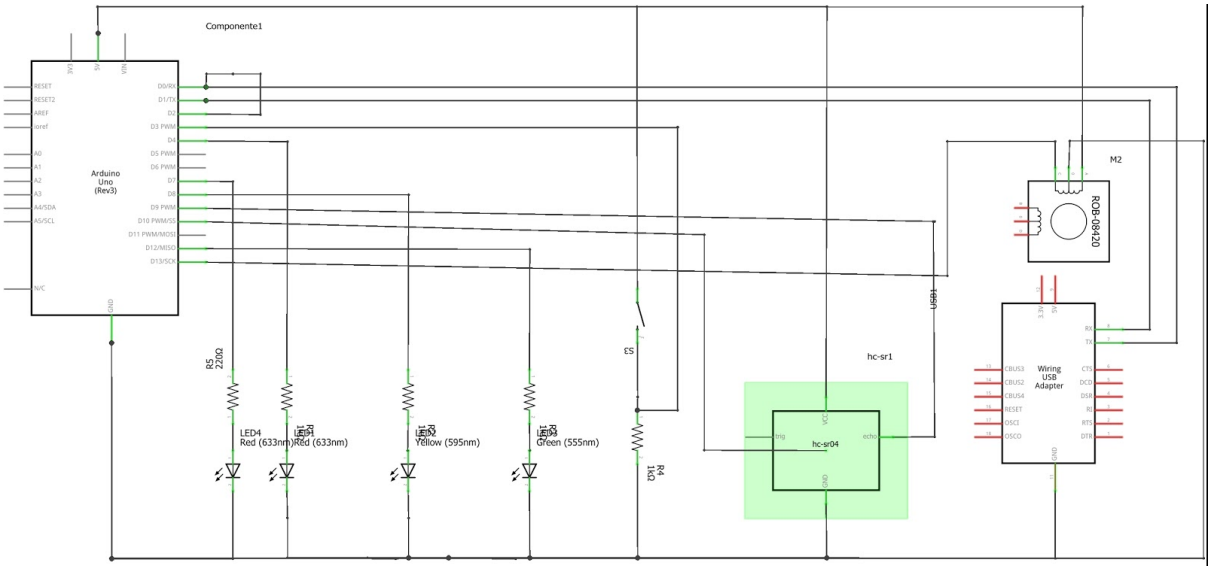

Carga de alimento
Arduino - Hardware
Presionar pulsadc
Girar motor
false
n cargas necesaria
true
Sensar carga
Obtener porcentaje de carga (p
Desicion de porcentaj
(p > 33%) && (p < 76%)
Enciende Led amarillc
p >75%
Enciende LED verde
p <= 33%
Enciende LED rojc
Envia datos (UART
PC - Aplicacion APF
Fin de carga
false
Si p <=33%
true
Actualizar variables con AP
Retroalimenta aplicacion AP
Recepcion de datos (UART
Servidor Ubidots
Notificar con sms

6. Descripción de la solución de hardware

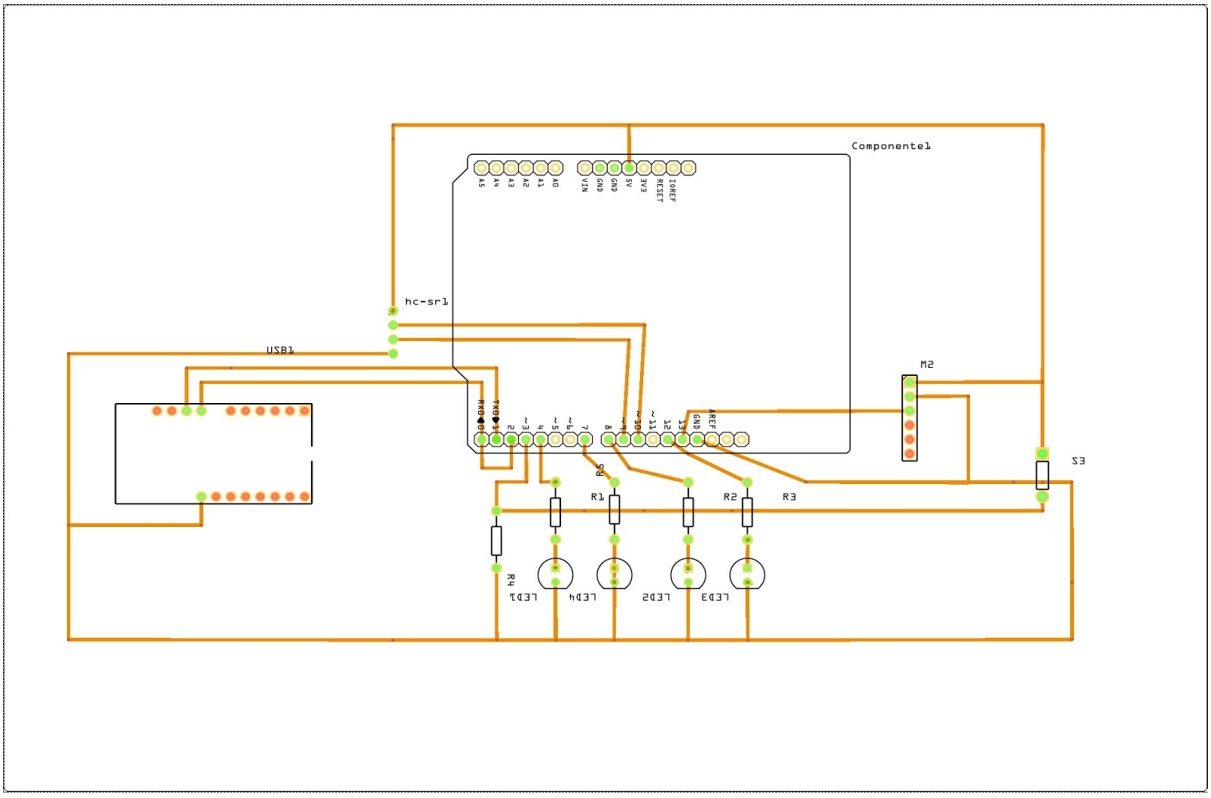
Para implementar el diseño del hardware se utilizaron los siguientes componentes electronicos:

- 5 resistencias de 1kohm
- 4 leds.
- 1 pulsador
- Sensor de distancia HC-SR04
- Servo Motor de DC
- USB serial
- Placa arduino Uno

El diseño de circuito eléctrico utilizado es el siguiente:



El diseño del circuito impreso PCB es el siguiente:


Componente1
hc-sr1
USB1
M2
S3
R1
R2
R3
LED1
LED2
LED3
LED4

7. Descripción de la solución de software

El sistema APF está formado por diversos sistemas de software correspondiente a cada uno de los subsistemas que lo componen. A continuación explicamos cada uno de estos.

7.1. Software del sistema embebido

El sistema APF cuenta, para el manejo del hardware, con una unidad de control compuesta por una placa arduino que en combinación con un circuito electrónico y adaptaciones de elementos electrónicos al alimentador del sistema conforman un sistema embebido encargado de operar sobre este entorno.
La unidad de control es una placa Arduino Uno, la cual es programada en C y que nos permite llevar a cabo nuestras necesidades a través de sus características. Sobre esta placa se utilizan las siguientes características:

- Interrupciones externas: en nuestro desarrollo utilizamos dos interrupciones externas. Una para un pulsador y otra para detectar recepciones por UART.
    - Interrupción externa 1: Esta interrupción es utilizada en combinación con un pulsador por el cual se detecta su accionamiento a través de un flanco de subida. Se adiciona instrucciones de software para eliminar el antirrebote ocasionados sobre estos pines.
    - Interrupción externa 2: Es un interrupcion externa que utilizamos para detectar las recepciones por UART. Como no se puede configurar una interrupción por UART en la placa pero si tiene una unidad de UART para usarla de manera eficiente la combinamos con un pin encargado de detectar un flanco de bajada, necesario para la transmisión UART, en ese momento se sabe que se está recibiendo y esto es muy útil si se requiere utilizar la placa en bajo consumo ya que las interrupciones externas son tenidas en cuenta.
- Interrupciones por timer: Como en toda placa es fundamental el manejo de tiempos, para nuestro caso de estudio también es un requerimiento primordial. Es por lo cual que utilizamos estas interrupciones para un correcto y adecuado manejo de los tiempos. Como la placa arduino tiene varias librerías incorporadas que hacen uso de los timers, hay que tener cuidado con cual timer utilizar. En nuestro caso haremos uso del timer 2 de la placa el mismo es utilizado por una función propia para señales de sonido (función tono()) por lo cual esta función se verá afectada en caso de utilizarse, como nosotros no hacemos uso de esta función entonces manejamos la interrupción por timer 2. La placa proporciona una librería que nos facilita el manejo de esta interrupción indicando cuando activarla y cada que tiempo se requiere llamar a la interrupción, de esta manera ya es configurada y solo debemos marcar la función que la atenderá. En la función por defecto setup() de nuestro código de placa se verá cómo se configura de una manera muy sencilla. Adjuntando la respectiva librería.
- Comunicación UART: para el uso de la comunicación por puerto serial utilizamos una UART configurada a una velocidad de 9600 baudios.
- Soporte a sensor de distancia: utilizamos un sensor de distancia encargado de medir el porcentaje que contiene el depósito de carga. Configuramos un conjunto de pines

para poder llevar a cabo de manera adecuada dicha operación y se utilizan un por de funciones extras para determinar la distancia en centímetros y pasar estos por una escala propia al porcentaje de carga correspondiente.
- Utilización de librería de servo: incluimos la librería de servos a nuestro código para llevar a cabo operaciones relacionadas al manejo de servos. En este caso configuramos y calibramos un conjunto de funciones y parámetros para poder manejar de manera adecuada este tipo de elementos electrónicos.

Para mayor detalle del código recomendamos visitar el mismo a través de nuestro repositorio de github (enlace externo).

7.2. Software de la interfaz de usuario

El sistema APF cuenta con una extensión de sus funcionalidades permitiendo conectar a una computadora. De esta manera una vez conectado existe una interfaz que se comunica con la placa arduino (o el sistema embebido) por UART, esto permite extender sus funcionalidades permitiendole las siguientes:

- Posicionar el de manera adecuada el ángulo inicial, para que luego el ángulo de giro sea el correspondiente al requerido por el dispositivo.
- Configurar más de una carga por accionamiento.
- Configurar cargas fijas menores a 12 hs. Con precisión de segundos.
- Accionar por la computadora el dispositivo.
- Visualizar el porcentaje de resto carga.
- Solicitar porcentaje de resto de carga.

Para mayor detalle del código recomendamos visitar el mismo a través de nuestro repositorio (enlace externo).

7.3. Interfaz de Ubidots

Nuestra interfaz de usuario contiene incluida en su código una API de que nos permite conectarnos a un servidor y de esta manera poder seguir extendiendo nuestras funcionalidades en combinación con el código de interfaz de usuario. De manera resumida, esta API nos permite vincular nuestras variables del código con un servidor de Ubidots que nos indica, por medio de indicadores virtuales en un Tablero Virtual y previamente creado, el comportamiento de nuestras variables en tiempo real. Es decir mientras el código está en ejecución podemos saber cómo están cambiando dichas variables. Además nos permite modificar variables y que dichos cambios impacten directamente sobre nuestro código. De esta manera podemos interactuar con una interfaz virtual, modificando nuestras variables y obtener los resultados sobre nuestro sistema ya que dichos cambios impactan de manera directa si hacemos uso estas variables para controlar nuestro sistema.
En nuestro caso de estudio, para el sistema APF, hicimos uso de esta API para vincularla con una interfaz que pueda ser accedida desde cualquier dispositivo con internet (smartphone, PC, notebook, etc.). También nos permite visualizar el porcentaje de carga que tiene nuestro depósito de alimento y damas tener el control de nuestro sistema desde esta interfaz virtual. Previamente nosotros adaptamos nuestras variables a dicha interfaz

virtual (Tablero virtual proporcionado por el servidor Ubidots) y de esta manera manejamos nuestro sistema a nuestras necesidades.

Nota: **Configuración de Linux para usar nuestro cliente con la API python**

Para poder hacer uso de esta API previamente se deben llevar a cabo los siguientes pasos, sobre nuestro sistema operativo Linux:

Desde el sistema operativo Linux debemos instalar la librería de ubidots en nuestro python.

- Asegurémonos de que su dispositivo esté actualizado para que tenga las últimas herramientas de python (tenga en cuenta que esto llevará un tiempo):

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

- Descargue el instalador de pip e instale la biblioteca de Python de Ubidots

```
$ sudo apt-get install python-setuptools
$ sudo easy_install pip
$ sudo pip install ubidots
```

8. Conclusiones

El presente trabajo nos deja una muy buena experiencia, que nos mostró cómo automatizar un proceso físico y explotar sus funcionalidades por medio del software. Además de esto con la ayuda de internet y la API ubidots pudimos llevar nuestro sistema a que sea controlado desde cualquier lugar del mundo de una manera sencilla y eficiente.

Aprendimos que para el diseño de nuestro sistema hacer uso de los diagramas estáticos y estructurales nos da una muy buena abstracción de los componentes intervinientes al mismo. Luego pudimos identificar los requerimientos y observar el comportamiento a través de los diagramas dinámicos. Todo esto gracias a la herramienta de diagramas SysML que se adapta muy bien para este tipo de sistemas que combinan software, hardware sistemas embebidos y demás elementos todos diagramados en un mismo lugar.

La idea que estuvimos desarrollando sobre nuestro caso de estudio tiene un amplio ámbito de aplicación lo que lo lleva a un caso de estudio muy interesante por la diversidad de productos que podrían crearse, explotando en cada uno de estos diferentes funcionalidades útiles, como por ejemplo un cargador automático para cocineros el cual podría llegar a precisar de manera adecuada las cantidades requeridas de alimentos o ingredientes mediante el uso de adaptaciones de software y hardware y de esta manera cumplir con dichos requisitos. Esta es una propuesta interesante para futuras implementaciones del producto brindando mayores funcionalidades y más específicas según sea el ámbito.

Volviendo a nuestro caso de estudio una propuesta futura, sería dejar de utilizar la comunicación UART y pasar a una comunicación inalámbrica con acceso a internet como WIFI. Para lograr esto proponemos dejar de usar la interfaz programada en python, y agregar la librería de Ubidots directamente sobre el código que corre la placa arduino en su código de programación. Investigando sabemos que esto es posible de manera que todas las variables manejadas por la placa pueden ser observadas por Ubidots. De esta manera tendríamos acceso directo a las variables y, como consecuencia, el control de la placa directamente con la interfaz de internet de Ubidots. Esto traería varias ventajas, como mejoras en el rendimiento, mayor tiempo de respuesta, entre otras. Todo esto sería posible si sobre nuestra placa arduino agregamos un módulo de WIFI o otra alternativa sería migrar nuestro código sobre una placa arduino con módulo WIFI incorporado.

9. Bibliografía

1. Ubidots. (2012 - 2018). Internet de las cosas con Ubidots. Enlace externo.
2. Sommerville, Ian. Software engineering. Pearson, 2011.
3. Kordon, F., Hugues, J., Canals, A., & Dohet, A. (Eds.). (2013). Embedded systems: analysis and modeling with SysML, UML and AADL. John Wiley & Sons. Enlace externo.