Construcción un ASIC para el Dispensador Automático de Alimento para Mascotas

Javier Vega, *Cód.:261975,* Manuel Neira, *Cód.:261990,* Nathalia Cante, *Cód.:262103,*
Byron Martinez, *Cód.:262063,* Saitel Agudelo, *Cód.:262018,* Luis Limas, *Cód.:261721,*
Rubén Hernández, *Cód.:262103*
(jafvegago, mfneirae, nrcanteh, sdagudelos, lmlimasd, rdhernandezve) @unal.edu.co
Universidad Nacional de Colombia

En el diseño de Circuitos Integrados para Aplicaciones Específicas (Asic) requiere de diferentes compuertas, la conexión entre estas compuertas establece las funciones que van a realizar, si se desea que el ASIC realiza cierto proceso se debe poner atención a esas conexiones. En este informe se presenta el proceso de integración del dispositivo ASIC para el control del dispensador de alimentos desde la construcción de los módulos hasta la implementación de la celda estándar y los módulos de. Una revisión a los objetivos y planteamiento es hecha como una discusión del resultado obtenido

I. Introducción

La fabricación de un producto de este tipo en un dispositivo ASIC requiere que el diseñador integre todos los módulos de su proyecto en uno solo. Cada módulo está compuesta por una serie de compuertas lógicas que dependiendo de su cantidad y conexión y configuración permiten a todo un módulo realizar una tarea específica. Esta tarea debe ser realizada con precaución ya que un error o un defecto en el diseño de alguna de estas compuertas se verán replicado en todo el diseño. A continuación se explica el proceso para integrar las diferentes compuertas a los modulos del proyecto y asi, obtener los planos del ASIC propuesto.

II. Justificación

La modernidad nos ha conducido a una cultura de la urgencia y muchas personas por cuestiones laborales o estudiantiles no tienen el suficiente tiempo para realizar algunas actividades que aunque son sencillas demandan tiempo o simplemente prefieren hacer este tipo de actividades desde la oficina o universidad. Para este tipo de personas se ofrece un dispositivo que permite dosificar y suministrar alimento a sus mascotas desde cualquier parte con sólo tener acceso a internet y a un smartphone,lo cual brinda una mayor comodidad y tranquilidad.

Por otro lado, también puede ser utilizado por aquellas personas que por motivos de viaje tienen que dejar a sus mascotas solas por perodos cortos de tiempo. Esta tecnologıa tambien puede ser aplicada en masa en veterinarias o lugares que entrenan mascotas o que cuidan de ellas (guardiıas), generando confianza en el dueno acerca del cuidado de sus mascotas, pues puede conocer en tiempo real cuando y cuánta comida se les ofrece

III. Objetivos

A. General

- Diseñar y construir un dispositivo dispensador de alimentos para mascotas que implemente el internet de las cosas (IoT).

B. Específicos

- Construcción de dispositivo dispensador.
- Diseño y construcción de aplicación IoT.
- Implementación de un sistema de comunicación Wifi con el dispensador y aplicación IoT.
- Diseño de compuertas lógicas básicas
- Diseño de los módulos estándar para el dispositivo ASIC.

IV. Descripción general

El dispositivo consta de un mecanismo que utiliza un contenedor y un dispensador, suministrando controladamente porciones de alimento. Además controla el horario de suministro y el tamaño de la ración, éste fue implementado haciendo uso de la integración de un sistema mecánico y un control con señales eléctricas; en la siguiente sección se describirá cada módulo del proyecto.

V. Módulos

A. Módulo de comunicaciones

Es un módulo Wi-Fi diseñado por Espressif, sirve como puente de comunicación wifi, se caracteriza por tener un gran soporte en la red y una comunidad cada vez más grande que lo respalda. Este dispositivo fue creado pensando en el tema de moda en el ámbito de las comunicaciones y la electrónica “El internet de las cosas (IOT)”. Que implica la comunicación y control de objetos de uso cotidiano. La característica más importante, a diferencia de otros similares, es que es un dispositivo de bajo costo aproximadamente 5 Dólares. Este dispositivo se caracteriza por su facilidad de uso. Su programación es fácilmente realizada a través de una conexión UART serial. Fue creado pensando en diseñadores de plataformas móviles, el cual provee de una capacidad insuperable para integrar capacidades Wi-Fi con otros sistemas. Las capacidades de almacenamiento y procesamiento internas del módulo ESP8266 le permiten ser integrado con sensores y actuadores a través de puerto GPIOs incorporados

sin que su uso afecte en gran medida la velocidad de operación del módulo. La alta integración de componentes dentro de un pequeño módulo hace que tenga tamaño muy reducido. El dispositivo contiene una pequeña CPU de bajo consumo energético, así como memorias ROM y SRAM, lo cual permite cargar y procesar configuraciones información relacionada con control, conectividad y web server. A continuación se muestran las características técnicas proporcionadas por el fabricante:

1) Protocolos soportados: 802.11 b/g/n.
2) Wi-Fi Direct (P2p), Soft Access Point.
3) Stack TCP/IP integrado.
4) PLL, reguladores y unidades de manejo de energía integrados.
5) Potencia de salida: +19.5dBm en modo 802.11b.
6) Sensor de temperatura integrado.
7) Consumo en modo de baja energía: menor a 10 μA.
8) Procesador integrado de 32 bits, puede ser utilizado como procesador de aplicaciones.
9) Wi-Fi 2.4 GHz, soporta WPA/WPA2.
10) Tamaño ultra reducido (11.5mm x 11.5mm).
11) Conversor analógico a digital de 10-bit.
12) Soporta variedad de antenas.
13) Integrated low power 32-bit MCU.
14) SDIO 2.0, SPI, UART, I2C.
15) Encendido y transmisión de datos en menos de 2 ms.
16) Rango de operación $-40°C$-$125°C$.

Este módulo se muestra en la figura 1.

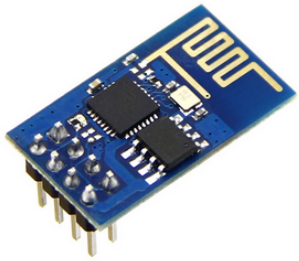

Figure 1. Módulo Wifi ESP8266

B. Programación módulo ESP8266

A continuación, en la figura 2, se muestra los pines para programación y funcionamiento del módulo Wifi.

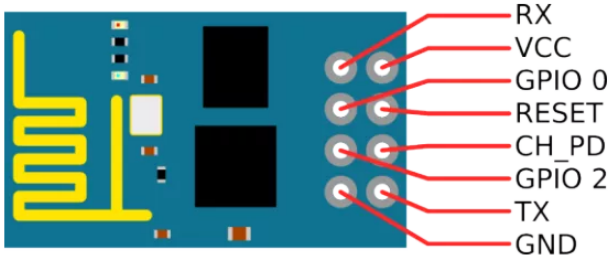

Figure 2. Esquemático módulo ESP8266

Para la programación del módulo se debe realizar a través de Windows o Linux y para realizar la conexión al módulo se debe incluir un conversor USB TTL Serial – FTDI. Las conexiones entre estos dos componentes se deben realizar como se muestra en la figura 3.

- RX $->$ TX
- TX $->$ RX
- CH·PD $->$ 3.3V
- GPIO 0 $->$ GND
- VCC $->$ 3.3V
- GND $->$ GND

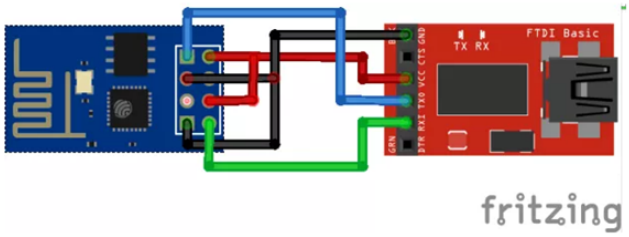

Figure 3. Conexión USB TTL Serial – FTDI

Es importante que la alimentación sea de 3.3V ya que un voltaje inferior o mayor puede llegar a dañar con facilidad el módulo.

El módulo originalmente funciona mediante comandos AT pero mediante la inclusión de un firmware se puede hacer programación de más alto nivel en un lenguaje llamado LUA. Lo anterior facilita mucho la programación del módulo, por ende se trabajó así. Para la inclusión del firmware se debe descargar la herramienta llamada NodeMCU flasher según sea el caso del sistema operativo que se esté usando.

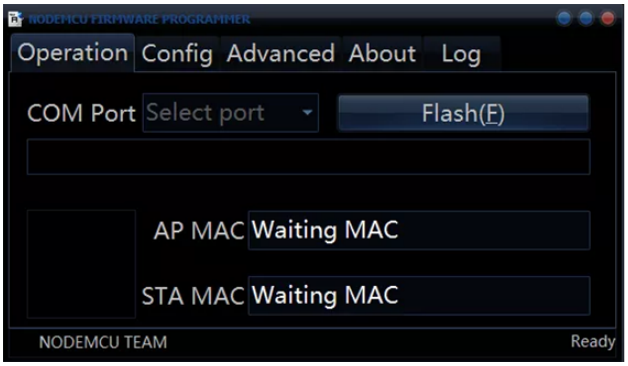

Figure 4. Herramienta NodeMCU flasher

En el siguiente link se puede encontrar toda la documentación de las funciones e información requerida para la actualización del firmware: https://github.com/nodemcu/nodemcu-flasher. Para crear y guardar los archivos LUA en el módulo ESP8266 se utilizó una herramienta llamada ESPlorer creada por 4refr0nt. Con esta herramienta es posible correr y guardar los archivos LUA desarrollados. Al guardar dichos archivos en memoria del ESP8266, estos se ejecutan automáticamente una vez se enciende el módulo. Es decir sólo se necesita cargar el

módulo una vez para cada determinada aplicación. Para ellos se debe realizar los siguientes pasos (Ver figura 5):

1) Conectar la FTDI al computador
2) Seleccionar el puerto FTDI
3) Abrir el puerto
4) Seleccionar NodeMCU+MicroPtyhon tab
5) Crear un nuevo archivo llamado init.lua
6) Presionar Save to ESP

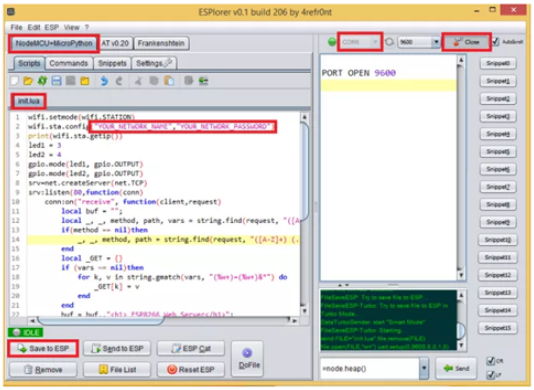

Figure 5. Herramienta ESPlorer

El código creado que se grabó en memoria, configura el modem en modo estación, se conecta a una red wifi determinada. Red que va servir de puente para controlar otros dispositivos. Una vez conectado a la red wifi local, se programa al módulo como un pequeño servidor, al cual se le asigna una ip estática para la comunicación. De esta manera la aplicación móvil redirecciona a la ip correspondiente. También se programó para que recibiera peticiones GET desde la app, con eso se hace control y se activa las dos salidas, que corresponde a dos bits. Así se puede proporcionar 4 estados.

C. Módulo de desarrollo de la aplicación

1) Entorno de desarrollo: Para el desarrollo de la interfaz y programación de la aplicación, se usó Google App Inventor que es una plataforma de Google Labs para crear aplicaciones de software para el sistema operativo Android. El sistema es gratuito y se está compuesto por dos partes.

La primera parte es una página web donde de forma visual y a partir de un conjunto de herramientas básicas, el usuario puede ir enlazando una serie de bloques para crear la aplicación. Esta plataforma es bastante intuitiva y se divide en el diseño de bloques y de diseñador. El primero de estos abarca la programación de los objetos colocados en la interfaz del diseñador. Se usan bloques matemáticos, lógicos, de control, procedimientos,listas etc.

El entorno de diseño se muestra en la Figura 6.

En la Figura 7 se muestra un bloque de programación realizado para cambiar de una pantalla a otra. Allí puede apreciarse que la programación es prácticamente gráfica y la forma de los componentes ayuda al usuario a entender los elementos que pueden actuar conjuntamente y los que no funcionan, ya que simplemente no encajan.En la Figura también se aprecia la inserción de elementos multimedia.

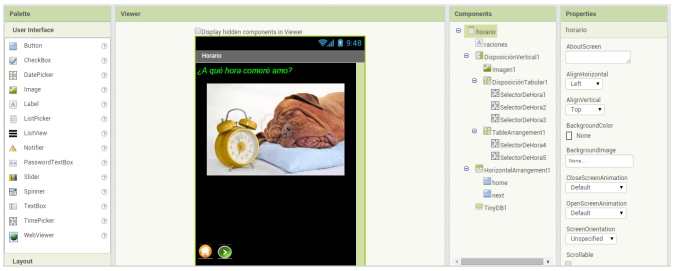

Figure 6. Entorno de diseño de la interfaz de MIT App Inventor

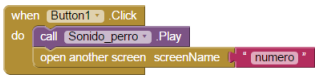

Figure 7. Bloques de programación para cambiar de la pantalla Inicio a numero

La otra parte es una aplicación para android que permite enlazar un teléfono inteligente o tableta a la aplicación para verificar el correcto funcionamiento de la misma. Esta última se llama: MIT AI2 Companion, y se puede descargar fácilmente de la tienda de aplicaciones de google.

En la Figura 8 se muestra la página principal de la aplicación mencionada, la cual permite escribir el código o escanearlo de la pantalla del computador, para hacer efectiva la conexión.

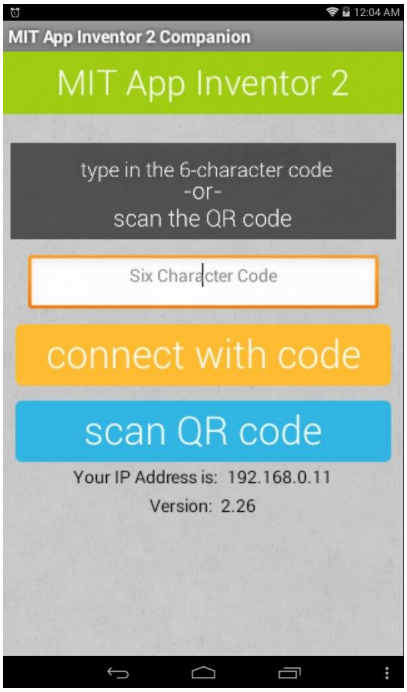

Figure 8. MIT AI2 Companion

Las aplicaciones fruto de App Inventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil. [2]

VI. Celdas estándar y simulaciones

A. GND

Es la tierra del circuito y consiste de un camino metálico en Electric de una entrada. (Figura 9)

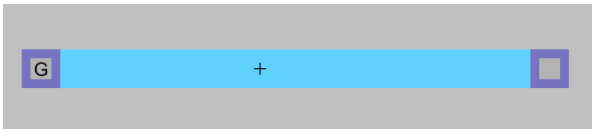

Figure 9. Layout GND

B. Inversor o Compuerta NOT

Esta compuerta implementa la negación lógica. Siempre que su entrada está en 0 (cero) lógico o en bajo, su salida está en 1 lógico o en alto; mientras que cuando su entrada está en 1 lógico o en alto, su salida va a estar en 0 lógico o en bajo [3]. Para realizar el diseño de los transistores se deben seguir los siguientes pasos:

- Crear un nuevo Proyecto: *File – New Library*.
- Guardar el Proyecto: *File – Save as*.
- Escoger la tecnología *mocmos*: *File – Preferences - Technology – Technology*.
- Se debe escoger la escala de *lambda* de 300 nm: *File – Preferences – Technolgy – Scale*.
- Se verifican las reglas de diseño *mocmos* en Electric: *File – Preferences – Technology - Design Rules*.
- Se crea una nueva celda: *Cell – New Cell*.
- Se agregan los componentes PMOS y NMOS.
- Para el NMOS se agregan las propiedades de *W*=1,5um y *L*=0,6 um y para el PMOS *W*=3um y *L*=0,6 um.

En la Figura 10 se puede observar el transistor constuido con las anteriores especificaciones.

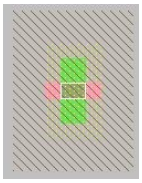

Figure 10. Transistor NMOS

Con los transistores creados se arma el circuito inversor de la Figura 11.

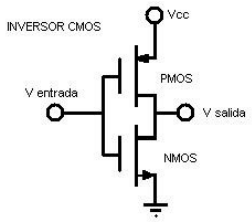

Figure 11. Inversor CMOS

Se procede a ubicar los contactos para NMOS con *pWell* y PMOS con *nWell*; luego se unen los contactos y los pozos. Por último, se le adicionan los barrajes y se les da sus respectivos nombres (OUT, VDD, IN, GND). Para poder realizar la simulación es necesario brindarles atributos de modelo SPICE a los transistores, lo cual se hace con *Tools - Simulation (Spice) - Set Spice Model*. Finalmente se obtiene el inversor de la Figura 12.

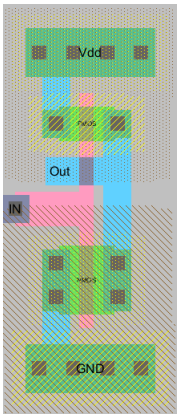

Figure 12. Inversor con Pmos y Nmos.

Para simular se debe adicionar un código en *Components - Misc - Spice code* que describe la señal de entrada y de alimentación. También se agrega el archivo *C5˙models.txt*. Lo anterior, es seguido por la generación del archivo *inversor.spi*, el cual se muestra en la Figura 13.

```
*** SPICE deck for cell INV{lay} from library Compuerta-INV
*** Created on dom may 24, 2015 17:16:06
*** Last revised on sáb jun 13, 2015 15:03:05
*** Written on sáb jun 13, 2015 15:13:42 by Electric VLSI Design System,
*** version 9.05
*** Layout tech: mocmos, foundry MOSIS
*** UC SPICE *** , MIN_RESIST 4.0, MIN_CAPAC 0.1FF

*** TOP LEVEL CELL: INV{lay}
Mnmos@0 gnd IN Out gnd NMOS L=0.6U W=3U AS=3.487P AD=16.695P PS=7.8U PD=24.9U
Mpmos@0 vdd IN Out vdd PMOS L=0.6U W=1.5U AS=3.487P AD=14.22P PS=7.8U PD=21.9U

* Spice Code nodes in cell cell 'INV{lay}'
vdd Vdd 0 DC 5
vin IN 0 dc 0 pulse 0 5 5m 10n 10n 5m 10m
.tran 0 100m
.include C:\Users\ACER-V3\Desktop\PROYECTOTECNICAS\C5_models.txt
.END
```

Figure 13. Código inversor.

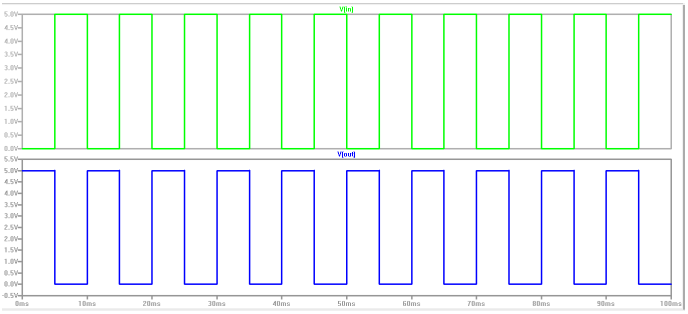

Figure 14. Simulación del inversor.

A continuación se hace la simulación con ayuda de la herramienta *LTSpice IV*. Como se ve en la Figura 14, el inversor

está funcionando adecuadamente. Se varío la frecuencia de simulación hasta 1MHz pero el inversor no presento mayor variación.

C. Compuerta XOR

La compuerta lógica XOR produce una salida verdadera o uno lógico cuando los valores de entrada son desiguales. La tabla de valores se muestra en el Cuadro I. El circuito equivalente de la compuerta implementado en tecnología CMOS se muestra en la Figura 15. El procedimiento de diseño es el mismo que para la compuerta inversora.

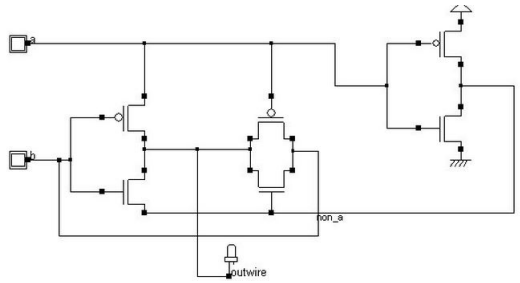

Figure 15. Compuerta lógica XOR mejorada implementada con CMOS

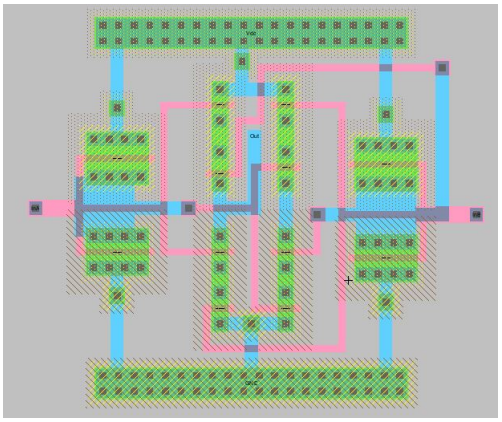

Figure 16. Compuerta lógica XOR Mejorada Implementada en Electric VLSI

El código generado por Electric se muestra en la Figura 17. Los resultados obtenidos se muestran en la Figura 18 para la cual se toman VinA y VinB como entradas del sistema. Tal y como se muestra, su comportamiento es mucho mejor que el obtenido anteriormente, por lo que se considera que el diseño fue el adecuado.

Table I
TABLA DE SALIDA CORRESPONDIENTE A LA COMPUERTA LÓGICA XOR

A	B	Output
1	1	0
1	0	1
0	1	1
0	0	0

D. Compuerta OR

Para crear la compuerta OR se debe usar una compuerta NOR y a la salida conectarla a una compuerta inversora como

```
*** UC SPICE *** , MIN_RESIST 4.0, MIN_CAPAC 0.1FF
*** SUBCIRCUIT XORM FROM CELL Compuerta-XOR-Mejorada:XORM{lay}
.SUBCKT XORM InA InB Out gnd vdd
Mnmos@6 gnd InA net@129 gnd NMOS L=0.5U W=5U AS=8.75P AD=21.875P PS=13.5U
+PD=27U
Mnmos@7 gnd InB net@137 gnd NMOS L=0.5U W=5U AS=8.75P AD=21.875P PS=13.5U
+PD=27U
Mnmos@8 net@166 InA Out gnd NMOS L=0.5U W=1.25U AS=1.797P AD=2.5P PS=5.375U
+PD=5.25U
Mnmos@9 gnd InB net@166 gnd NMOS L=0.5U W=1.25U AS=2.5P AD=21.875P PS=5.25U
+PD=27U
Mnmos@10 net@171 net@137 net@152 gnd NMOS L=0.5U W=1.25U AS=2.031P AD=2.5P
+PS=5.75U PD=5.25U
Mnmos@11 gnd net@129 net@171 gnd NMOS L=0.5U W=1.25U AS=2.5P AD=21.875P
+PS=5.25U PD=27U
Mpmos@6 net@129 InA vdd vdd PMOS L=0.5U W=5U AS=22.5P AD=8.75P PS=27.25U
+PD=13.5U
Mpmos@7 net@137 InB vdd vdd PMOS L=0.5U W=5U AS=22.5P AD=8.75P PS=27.25U
+PD=13.5U
Mpmos@8 net@145 InA vdd vdd PMOS L=0.5U W=1.25U AS=22.5P AD=3.094P PS=27.25U
+PD=6.75U
Mpmos@9 Out net@137 net@145 vdd PMOS L=0.5U W=1.25U AS=3.094P AD=1.797P
+PS=6.75U PD=5.375U
Mpmos@10 net@150 InB vdd vdd PMOS L=0.5U W=1.25U AS=22.5P AD=2.812P PS=27.25U
+PD=5.75U
Mpmos@11 net@152 net@129 net@150 vdd PMOS L=0.5U W=1.25U AS=2.812P AD=2.031P
+PS=5.75U PD=5.75U
.ENDS XORM

*** TOP LEVEL CELL: XORMEJ{lay}
XXORM@1 InA InB Out gnd vdd XORM

* Spice Code nodes in cell cell 'XORMEJ{lay}'
Vdd Vdd gnd dc 5
VinA InA gnd 0 pulse 0 5 5m 10n 10n 5m 10m
VinB InB gnd 0 pulse 0 5 7m 14n 14n 7m 14m
.tran 0 40m
.include C:\Users\Manuel\Desktop\Electric\C5_models.txt
.END
```

Figure 17. Código generado como modelo SPICE de la compuerta XOR Mejorada

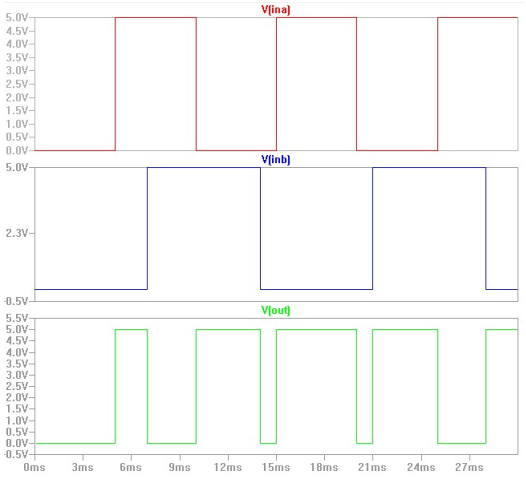

Figure 18. Resultados de la simulación con LT spice IV

la Figura 19, debido a que la salida de la compuerta NOR solo es 1 lógico cuando las entradas A y B son 0. El inversor invierte relación, como ese muestra en el Cuadro II. Por consiguiente, se usó las compuertas NOR e inversor fabricadas para crear la compuerta OR, como se ilustr en la Figura 20. En las figuras 21 y 22 se observa el código generado por VLSI y la grafica de simulacion en Electric, respectivamente. Para la simulación se usaron 2 señales de diferente frecuencia con el fin de evaluar todos los casos, solo cuando ambas señales están en cero o nivel bajo la salida se hace cero. El resto del tiempo la salida permanece en 1.

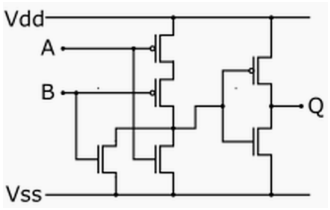

Figure 19. Circuito lógico compuerta OR Cmos.

Table II
TABLA DE SALIDA CORRESPONDIENTE A LA COMPUERTA LÓGICA OR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

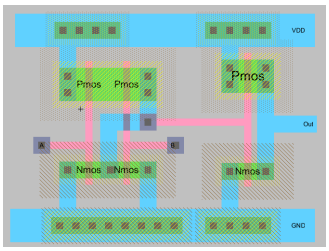

Figure 20. Compuerta OR con Pmos y Nmos.

```
*** SPICE deck for cell or_2_IN{lay} from library OR
*** Created on dom jun 14, 2015 20:12:11
*** Last revised on mar jun 16, 2015 00:18:52
*** Written on mar jun 16, 2015 00:19:05 by Electric VLSI Design System,
*version 9.03
*** Layout tech: mocmos, foundry MOSIS
*** UC SPICE *** , MIN_RESIST 4.0, MIN_CAPAC 0.1FF
*** TOP LEVEL CELL: or_2_IN{lay}
Mnmos@2 net@10 A gnd gnd Nmos L=0.6U W=1.5U AS=12.45P AD=3.825P PS=21.6U PD=6.5U
Mnmos@3 gnd nmos@3_poly-left net@10 gnd Nmos L=0.6U W=1.5U AS=3.825P AD=12.45P PS=6.5U PD=21.6U
Mnmos@4 gnd net@10 Out gnd Nmos L=0.6U W=1.5U AS=4.725P AD=12.45P PS=8.7U PD=21.6U
Mpmos@0 net@2 A vdd vdd Pmos L=0.6U W=3U AS=15.75P AD=6.3P PS=25.5U PD=7.2U
Mpmos@3 net@10 B net@2 vdd Pmos L=0.6U W=3U AS=6.3P AD=3.825P PS=7.2U PD=6.5U
Mpmos@4 vdd net@10 Out vdd Pmos L=0.6U W=3U AS=4.725P AD=15.75P PS=8.7U PD=25.5U

* Spice Code nodes in cell cell 'or_2_IN{lay}'
vdd Vdd 0 DC 5
vin  A 0 DC 3 pulse 0 5 0f 10f 10f 5n 10n
vin1 B 0 dc 3 pulse 0 5 0f 10f 10f 10n 20n
.tran 0 100n
.include C:\Users\DeZX\Desktop\inversor2\C5_models.txt
.END
```

Figure 21. Archivo .spi de la compuerta OR.

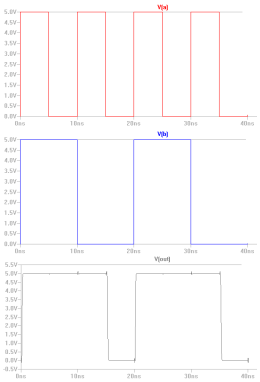

Figure 22. Resultados de la simulación compuerta OR con LT spice IV

E. Buffer o Seguidor

Para crear el buffer basta con conectar la salida de un inversor a otro. La negación de la salida del inversor dará el nivel de la señal de entrada, como se muestra en la Figura 23. Este comportamiento se resumen en la Tabla III. En la fabricación se usó la compuerta inversora que ya se habia diseñado para la compuerta OR, como se ilustra en la Figura 20. En la simulación se usó una señal que tenia un nivel bajo. Al observar la señal de salida, esta tiene la misma forma pero con un nivel de tensión cercano al de la fuente; luego el buffer esta cumpliendo su trabajo que es restablecer el nivel de tensión de una señal. En las Figuras 25 y 26 se observa el código y la grafica de simulacion en Electric, respectivamente.

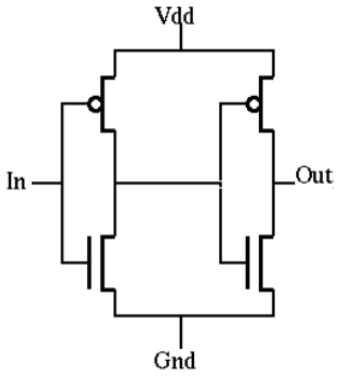

Figure 23. Circuito lógico compuerta Buffer Cmos.

Table III
TABLA DE SALIDA CORRESPONDIENTE A LA COMPUERTA LÓGICA BUFFER

IN	Output
0	0
1	1

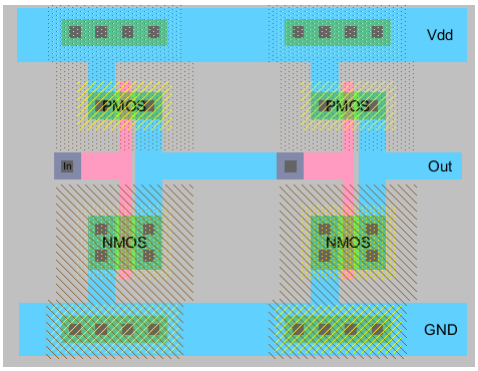

Figure 24. Buffer con Pmos y Nmos.

```
*** SPICE deck for cell or_2_IN{lay} from library OR
*** Created on dom jun 14, 2015 20:12:11
*** Last revised on mar jun 16, 2015 00:18:52
*** Written on mar jun 16, 2015 00:19:05 by Electric VLSI Design System,
*version 9.03
*** Layout tech: mocmos, foundry MOSIS
*** UC SPICE *** , MIN_RESIST 4.0, MIN_CAPAC 0.1FF
*** TOP LEVEL CELL: or_2_IN{lay}
Mnmos@2 net@10 A gnd gnd Nmos L=0.6U W=1.5U AS=12.45P AD=3.825P PS=21.6U PD=6.5U
Mnmos@3 gnd nmos@3_poly-left net@10 gnd Nmos L=0.6U W=1.5U AS=3.825P AD=12.45P PS=6.5U PD=21.6U
Mnmos@4 gnd net@10 Out gnd Nmos L=0.6U W=1.5U AS=4.725P AD=12.45P PS=8.7U PD=21.6U
Mpmos@0 net@2 A vdd vdd Pmos L=0.6U W=3U AS=15.75P AD=6.3P PS=25.5U PD=7.2U
Mpmos@3 net@10 B net@2 vdd Pmos L=0.6U W=3U AS=6.3P AD=3.825P PS=7.2U PD=6.5U
Mpmos@4 vdd net@10 Out vdd Pmos L=0.6U W=3U AS=4.725P AD=15.75P PS=8.7U PD=25.5U

* Spice Code nodes in cell cell 'or_2_IN{lay}'
vdd Vdd 0 DC 5
vin  A 0 DC 3 pulse 0 5 0f 10f 10f 5n 10n
vin1 B 0 dc 3 pulse 0 5 0f 10f 10f 10n 20n
.tran 0 100n
.include C:\Users\DeZX\Desktop\inversor2\C5_models.txt
.END
```

Figure 25. Archivo .spi de la compuerta Buffer.

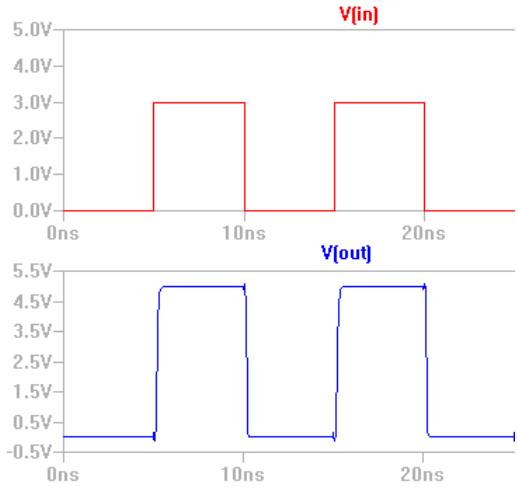

Figure 26. Resultados de la simulación Buffer con LT spice IV

F. AND

A continuación se muestra el modelo de una compuerta AND de dos entradas, a partir de la cual se construirá una celda básica en Electric. (Figura 27). En la Figura 28 se muestra la tabla de estados para la compuerta AND, obtenida a partir del circuito lógico.

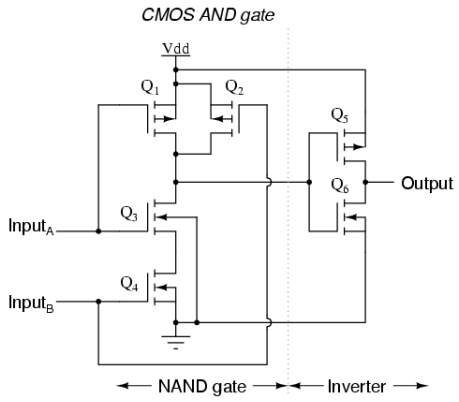

Figure 27. Circuito lógico de la compuerta AND Cmos

INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Figure 28. Tabla de estados de la compuerta AND Cmos

En la Figura 29, se muestra el layaout obtenido en Electric para la compuerta AND.

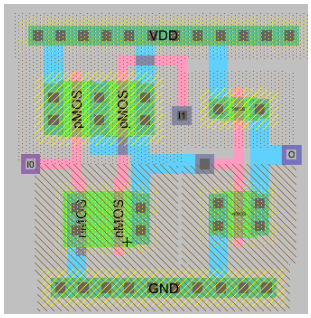

Figure 29. Layout AND de dos entradas Cmos.

A continuación, se presentan las compuertas AND necesarias para el proyecto y la simulación de la AND de 8 entradas en la Figura 34.

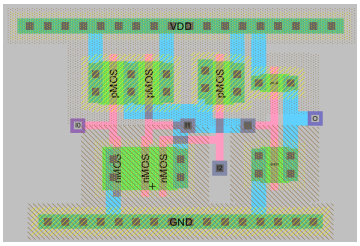

Figure 30. Layout AND de tres entradas Cmos.

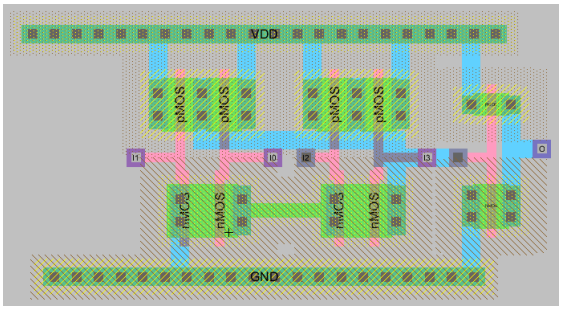

Figure 31. Layout AND de cuatro entradas Cmos.

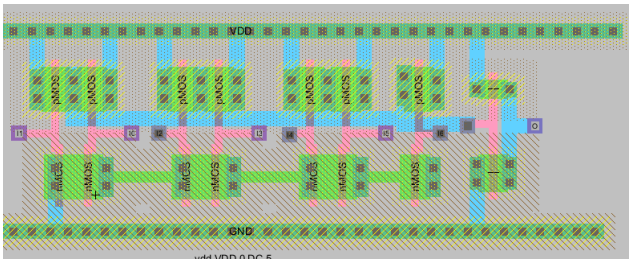

Figure 32. Layout AND de siete entradas Cmos.

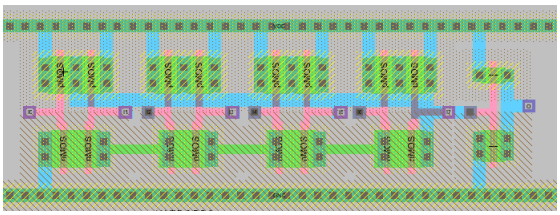

Figure 33. Layout AND de ocho entradas Cmos.

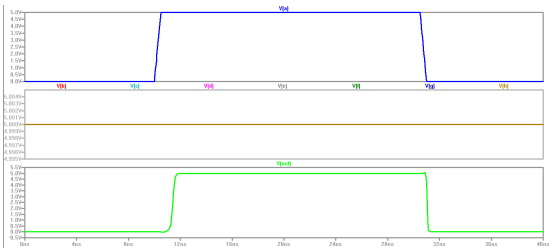

Figure 34. Simulación de la compuerta AND de ocho entradas Cmos.

G. FD

El flip-flop tipo D captura el valor de la entrada D durante un intervalo del ciclo de reloj (como el flanco de subida). El valor capturado se convierte en la salida Q. Este tipo de flip-flop puede ser visto como una celda de memoria, un almacenador de orden cero o una línea de retraso. El bloque de circuito de este componente se ilustra en la Figura 35 y su respectivo layout en la Figura 36.

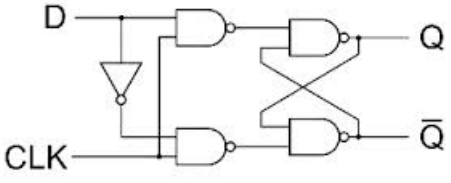

Figure 35. Flip-Flop tipo D.

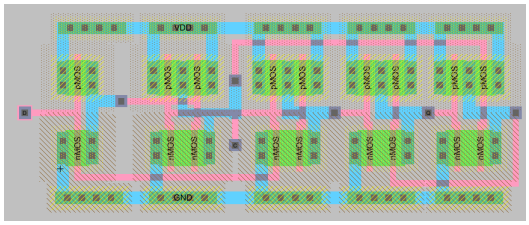

Figure 36. Layout Flip-Flop tipo D.

H. FDCE

El FDCE es un simple flip-flop tipo D con habilitación de reloj y reseteo asíncrono. Cuando la habilitación del reloj (CE) está en alto y el reseteo asíncrono (CLR) está bajo, el dato en la entrada D del FDCE es transferido a la correspondiente salida Q durante el flanco de subida del reloj. Cuando CLR es alto, esto se sobrepone a todas las otras entradas y resetea la salida Q a bajo. Cuando CE es bajo, la transición de reloj es ignorada. El bloque de circuito de este componente se ilustra en la Figura 37 y su respectivo layout en la Figura 38.

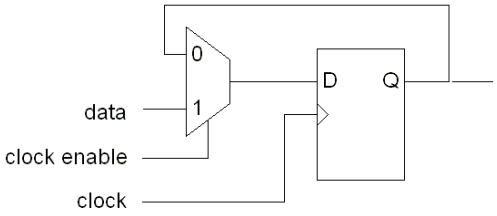

Figure 37. Flip-Flop tipo D con habilitación de reloj y reseteo.

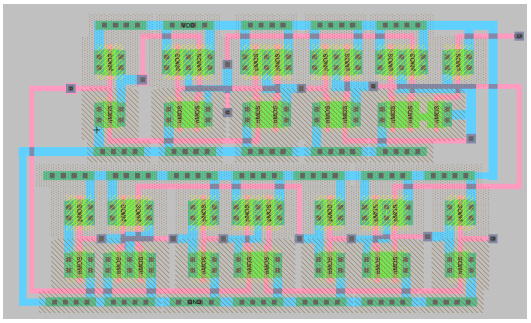

Figure 38. Layout Flip-Flop tipo D con reset y activación de reloj.

VII. Mapeo de celdas y enrutamiento

A. Generación archivo verilog

Lo primero que se hizo fue entrar a la ruta del proyecto TopM y abrir una terminal para luego implementar la siguiente instrucción *netgen -ofmt verilog TopM.ngc*. A continuación, se genero el archivo verilog al cual se le hicieron algunas modificaciones para eliminar los errores al compilar en Electric.

A continuación, con el código verilog se confirmaron las celdas básicas que usaba el proyecto y se creó una nueva librería con el layout de todas las celdas especificadas en la sección anterior para luego ser copiadas a la líbreria SCLIB mediante la opción *cell– cross library copy*. Las celdas del proyecto se ven en la Figura 39

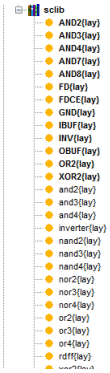

Figure 39. Celdas usadas en la librería SCLIB.

Luego se crea una nueva celda tipo verilog con el nombre del módulo del proyecto y se copia el código verilog generado anteriormente para luego compilarlo en Electric mediante la opción *Tools– Silicom compiler– Compile Verilog to NetList view*.

Una vez no se presentan errores en el netlist se genera el Layout mediante *Tools– Silicom compiler– Convert current cell to layout*.

Finalmente, se genera el Layout, el cual se puede observar en las Figura 44 y Figura 40.

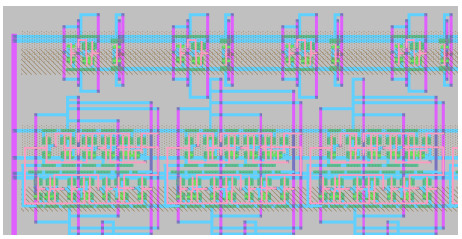

Figure 40. Sección del Layout del proyecto.

Para este proyecto se implementarón 341 compuertas, 3464 transistores y 6644 nodos.

VIII. Implementación

Como paso previo, se tenía diseñado el esquemático de caja negra del módulo el cual se muestra en la Figura 41, el cual al ser expandido se descompone en los sub-módulos mostrados en la Figura 42, para la cual se determinó que el sub-módulo más importante era el motor que daría movimiento a la distribución de alimento.

Figure 41. Esquemático de Caja Négra del módulo

Se procedió a implementar la estructura básica del dispensador de comida de tal forma que se lograse realizar una prueba de funcionalidad directamente con el control de un motor que se encargará de distribuir y proporcionar el alimento de la mascota. Como primer paso se escogió el tipo de mecanismo a usar. Para esto se tomó la decisión de que el más conveniente era un motor DC de 12 voltios debido a que el control sobre este se podría realizar de forma fácil y eficiente con *Pulse Width Modulation*.

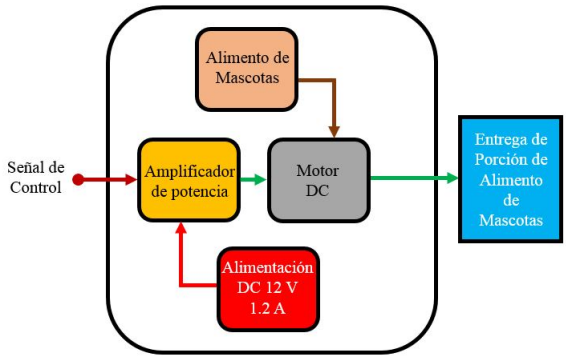

Figure 42. Caja négra expandida

Como segundo paso se procedió a diseñar el distribuidor de comida o mecanismo de racionalización del alimento. Este fue creado tomando en cuenta el movimiento y la posición que ocuparían el motor en la estructura. El diseño final se muestra en la Figura 43. Además se escogió una alimentación adecuada para suplir de energía tanto al motor como a los módulos del proyecto, para esto se usó un cargador universal variable de 12 voltios a 1.2 amperios.

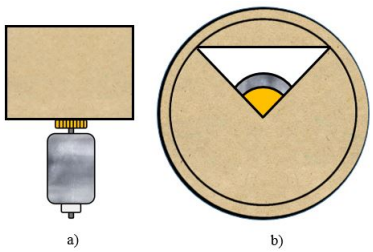

Figure 43. Diseño del distribuidor de comida a) vista lateral, b) vista superior

Finalizada la etapa de diseño se procedió a implementar físicamente cada una de las partes de la estructura. Los resultados obtenidos se muestran en la Figura 45 en la que se muestra la estructura final diseñada.

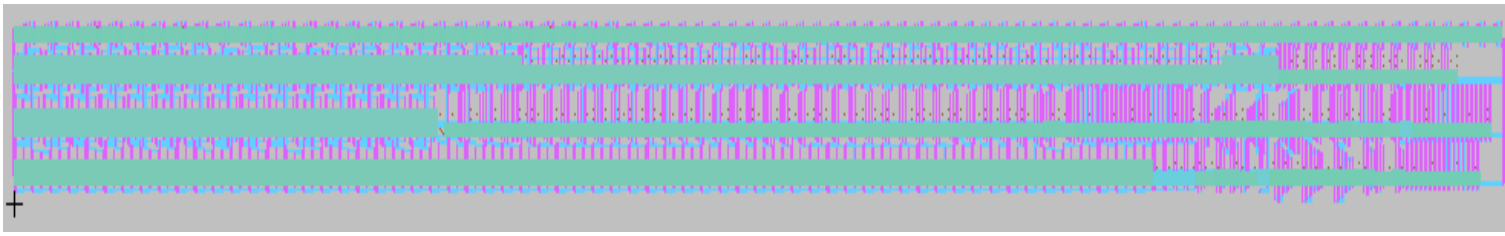

Figure 44. Layout completo del proyecto.

Figure 45. Estructura externa del prototipo

Fase	Actividad / Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Inicio.	Definir proyecto: Descripción y especificaciones.																
Conceptualización.	Consulta: Internet de las cosas																
	Revisión de concepto: Programación en HDL																
Programación de la aplicación	Consulta: Programación de aplicaciones móviles																
	Revisión: Hoja de datos de los dispositivos a utilizar																
	Diseño y programación: Aplicación móvil.																
Comunicaciones	Consulta: Formas de establecer conexión a internet																
	Revisión: Hoja de datos de los dispositivos a utilizar																
	Diseño y programación: Sistema de comunicación																
	Verificación: Prueba de comunicación entre Rx y Tx																
Diseño y construcción del prototipo: dispensador	Consulta: Dispensadores y formas de control																
	Revisión: Hoja de datos de los dispositivos a utilizar																
	Montaje: Etapa de potencia y tratamiento de señales.																
	Verificación: Prueba de funcionamiento emulando señales de entrada.																
Sistema Completo	Integración: Módulos del sistema																
	Verificación: Prueba de funcionamiento del sistema en acción.																
Celdas estándar	Diseño de celdas estándar																
Etapa final	Modificación: revisión y modificaciones finales																
Entregas.	Presentación de propuesta del proyecto.																
	Presentación de resultados: Prueba de módulos en FPGA. Informe 2.																
	Presentación de resultados: Celdas estándar. Informe 3																
	Presentación de resultados: Informe final.																

Figure 46. Cronograma del proyecto.

IX. Responsables

El compromiso de los integrantes del grupo fue un factor determinante para la consecución de los objetivos propuestos. Para que el equipo fuera eficiente, tuvo que cumplir con algunas características, como una buena comunicación, una organización que incluye la distribución de roles, cooperación entre los integrantes y tolerancia. Se reconoció la importancia de permitir el desarrollo de habilidades en los miembros del grupo, de modo que pudiean desempeñarse en tareas que les gustan y manejaran bien, a la vez que intercambian conocimientos con los demas miembros. Por lo anterior, la asignación de tareas se realizó teniendo en cuenta los intereses de cada miembro del equipo. Algunos roles especiales fueron asignados con la intencion de facilitar el trabajo y propender por el alcance de los objetivos del proyecto.

- **Líder:** Encargado de darle cohesion al grupo, mediante el establecimiento de lazos de comunicacion adecuados, la asignacion de tareas y la verificación de la coherencia entre diferentes partes y actividades del proyecto. Adicionalmente, asistira a las reuniones con el docente. El líder de este proyecto es Javier Vega.
- **Revisor:** Encargado de verificar el avance de las actividades en relacion con el cronograma establecido. De ser necesario, sugiere cambios en el orden o fecha de las actividades. La revisora de este proyecto es Nathalia Cante.
- **Software:** El diseno y construcción de la aplicación software para el dispositivo movil estuvieron a cargo de Nathalia Cante y Daniela Agudelo.
- **Hardware:** Esta dividida en 2 partes: El mecanismo de dispensor de alimentos, a cargo de Manuel Neira, Ruben Hernández y Rafael Rincón. El sistema de comunicacion entre dispensador y la red de internet, a cargo de Miguel Limas, Byron Martínez y Javier Vega.

X. Cronograma

El cronograma de trabajo establecido para realizar este proyecto de manera organizada se muestra en la Figura 46. Este contempló tanto el periodo de diseño, como de desarrollo del proyecto. Las actividades fueron seleccionadas a través de debates generados entre todos los integrantes del proyecto, asumiendo los roles de cargo mostrados en la sección IX de Responsables.

Las actividades fueron cumplidas de principio a fin, por tal razón se podría inferir que el desarrollo del proyecto fue el esperado y se logró llegar al prototipo final planteado a inicio de semestre.

XI. Costos

La Tabla IV presenta un estimado de los costos debido a mano de obra en cada una de las tareas del proyecto y el tiempo dedicado por los integrantes, Estos valores fueron tomados de los estándares de la Universidad Nacional de Colombia para pagos a estudiantes, asistentes docentes y docentes, estos datos fueron utilizados para ensamblar de manera adecuada la tabla V en la cual se muestra el valor total del proyecto considerando los costos de los materiales que fueron utilizados durante la implementación en físico del proyecto, y los costos de mano de obra de trabajo de todos los integrantes del equipo.

Debido a que se está generando un prototipo, es permitido llegar a valores tan elevados como el obtenido al final de la tabla V, pero esto no significa que el precio comercial del proyecto sea de ese valor, se pretende que a futuro el proyecto sea implementado en producción en serie, lo cual permita reducir el costo de venta de cada unidad y de esta forma hacer el proyecto más asequible al público.

En la siguiente tabla, C indica cantidad y M indica meses laborados.

Table IV
Estimación de costos de personal

Perfil	Costo	C	M	Total
Ingeniero Electricista o Electrónico con estudios de postgrado y doctorado Técnicas de Integración	$ 9'000.000	1	4	$ 36'000.000
Estudiante de posgrado con conocimientos en Técnicas de integración	$1'848.000	1	4	$7'392.000
Estudiante de pregrado en Ingeniería Electrónica Revisor	$ 812.000	1	4	$ 3'240.000
Estudiante de pregrado en Ingeniería Electrónica Lider	$ 932.000	1	4	$ 3'728.000
Estudiante de pregrado en Ingeniería Electrónica	$600.000	6	4	$14'400.000
	Total			$ 65'960.000

Table V
Costos de Implementación

Material	Unidades	Precio	Total
Palos de Balso	12	$ 600,00	$ 7.200,00
Barra de silicona	15	$ 800,00	$ 12.000,00
Carton Paja Pliego	4	$ 1.200,00	$ 4.800,00
Motor DC 12 V	1	$ 12.000,00	$ 12.000,00
FPGA Nexis 3	1	$ 360.000,00	$ 360.000,00
Jumpers X15	2	$ 15.000,00	$ 30.000,00
Estaño	1	$ 1.000,00	$ 1.000,00
Acrilico Transparente	2	$ 2.000,00	$ 4.000,00
Gastos de personal	1	$ 65.960.000,00	$ 65.960.000,00
		Total	$ 66'391.000,00

XII. Conclusiones

- La utilización del módulo Wi-Fi ESP8266 trajó con si numerosas ventajas, entre las cuales se destaca la reducción de la programación en la tarjeta de desarrollo, lo cual finalmente se vió reflejado en una menor complejidad del ASIC.
- La creación de software que permite agilizar el proceso de fabricación de CIs y el flujo de diseño como Electric VSLI, ha sido de gran importancia para mejorar los procesos y disminuir los errores que invalidan un circuito.
- Las diferentes normas de diseño junto a los programas que permiten la realización de Layouts le brindan la posibilidad al diseñador de interactuar y elegir varios parámetros del circuito.

- La disposición adecuada de los transistores es clave para lograr un diseño óptimo que cumpla con las reglas de diseño y aproveche el espacio, permitiendo ahorrar elementos de fabricación y por ende costos.
- La herramienta Electric aunque es muy poderosa podría mejorarse, lo que es claro indicio que aún queda mucho desarrollo de software para que el diseñador pueda hacer sus tareas más rápidamente; un ejemplo de esto es que al pasar el código del proyecto a formato vhdl para ser reconocido por Electric se presentan algunos errores con los vectores por lo que aún es necesario especificar cada uno de los espacios del vector.
- El lanzamiento de este dispositivo al mercado suplirá la necesidad de muchas personas que no disponen del tiempo suficiente para cuidar de su mascota, o que sencillamente quieren automatizar esta tarea para su comodidad.
- El crecimiento de las necesidades de los usuarios junto con el agotamiento de recursos como el espectro de radio frecuencia, abrirá paso a aplicaciones mucho más robustas que integren diferentes tipos de tecnologías. Lo anterior, con el fin de ofrecer mayor cantidad de funciones al usuario, todo esto relativo al internet de las cosas.
- La realización de un proyecto como este se hace con mayor eficacia al tener trabajo de varias disciplinas. De haber sido así, el primer prototipo habría llegado a un punto de avance mayor.

REFERENCES

[1] Espressif Systems. *Espressif Smart Connectivity Platform: Esp8266*. [online]. Disponible en: https://nurdspace.nl/images/e/e0/ESP8266_Specifications_English.pdf

[2] Wikipedia. *App Inventor*. Consultado el 6 Marzo de 2015. Modificado el 9 Abril de 2015 [Online]. Disponible en: http://es.wikipedia.org/w/index.php?title=Especial:Citar&page=App_Inventor&id=80423305.

[3] Wikipedia. *Puerta NOT*. Consultado el 14 de Junio de 2015. Modificado por última vez el 24 de Abril de 2015 a las 17:54. [Online]. Disponible en: https://es.wikipedia.org/wiki/Puerta_NOT