1 Agradecimientos

Quiero agradecer al estado en democracia y a los ciudadania que lo compone por haberme brindado la educacion publica y gratuita la cual me brindo por medio de los docentes una educacion de calidad y competitiva.

Recuerdo mis primeros dias en FaMAF, hice mi ingreso en Agosto por adelantado para tener mis vacaciones extendidas en verano y no cursar en Febrero el ingreso.

Los 3 primeros años fueron durisimos. Luego comence a trabajar para hacer algo distinto. Volvi con muchas ganas despues de ponerme de novio buscando mas conocimiento y experiencia. Termine de cursar y volvi a trabajar, en cosas mucho mejores.

El espiritu de los docentes me mantuvo vivo, esto es fruto de eso.

Muchas cosas decantan con el tiempo, sobretodo las ideas mas profundas.

Ciencias de la computacion es la carrea que volveria a elegir si tuviese que volver a empezar, por el temple de mis docentes y lo que significa la computacion en estos tiempos.

Este camino no termina aca, "te sobran años Bovina" una frase de un profesor que me enseño que el tiempo aun lo tengo, que vale y lo tengo que usar muy bien.

Gracias FaMAFC, familia, amigos y director de tesis!

Paciencia Paciencia Más Paciencia!

era verdad ...

2 Marco de trabajo

En el proceso de investigacion del fenomeno fisico de Resonancia Magnetica Nuclear el investigador manipula modulos electronicos digitales de medicion precisos, estables y en algunos casos si intervienen mas de uno a la vez entre ellos sincronizados por medio de interfaces digitales. Estos modulos electronicos digitales colaboran con la creacion del contexto necesario para investigar lo planeado segun la necesidad del investigador. En general junto con los modulos electronicos intervienen otros modulos electronicos de naturaleza analogica tales como amplificadores operacionales, mezcladores de señales, filtros pasa bajos entre otros.

La correcta conexion entre los diferentes modulos electronicos digitales, analogicos y su configuracion durante el proceso de experimentacion son responsabilidad del investigador y una tarea de suma importancia para el exito de la experiencia a realizar.

En este marco de responsabilidades del investigador y el modulo digital a ser utilizado es deseable y necesaria mecanismos sencillos de manipulacion del mismo y su monitoreo.

2.1 Rol del Software

El rol primario del software en el contexto descripto previamente es el de manipular el modulo digital a traves de la PC utilizada por el investigador de forma local o remota.

El rol secundario la administracion de usuarios del modulo digital, monitoreo de los experimentos en curso y resultados.

3 Partes de un experimento

Estos son algunos conceptos clave para comprender el contexto y definicion de un experimento, los cuales se tuvieron en cuenta al momento de la implementacion del sistema.

3.0.1 Pulso de radio frecuencia

Es una señal senoidal que tiene como objetivo estimular la muestra presente en el resonador y que puede ser manipulada previamente por otros modulos externos. Tiene los siguientes atributos:

- Frecuencia: 0 a 200 megahercios.
- Fase: 0 a 360 grados.
- Duracion: 0 a 16 segundos.

3.0.2 Pulso TTL

Es una señal digital de 5 voltios con la finalidad de trasmitirla como entrada a otros aparatos digitales o analogicos para sicronizar momentos de relajacion y estimulacion de las muestras durante la experiencia.

3.0.3 Muestras

Las muestras son un grupo de datos obtenidos a cierta frecuenta de muestreo y agrupados de manera contigua en un bloque de longitud 2^n con $n \in \mathbb{N}$.

3.1 Tipo de Experiencia

El tipo de experiencia a realizar esta determinada por el investigador, existen 2 bien difrenciadas:

3.1.1 Experiencia Promedio

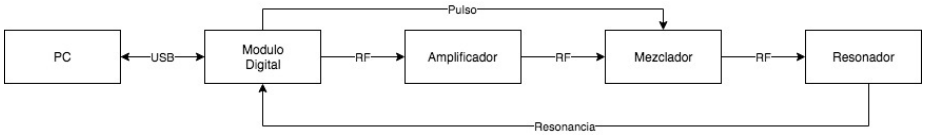
Una experiencia promedio es una que se repite y donde los bloques de muestras ordenadas se suman uno a uno obteniendo una mejor representacion los datos para su analisis.

3.1.2 Experiencia Promedio con Pulso variable

Es una experiencia promedio donde la configuracion de los pulsos cambia en las sucesivas iteraciones de la experiencia. Los pulsos cambian su configuracion para suprimir interferencias de estimulaciones previas y obtener mediciones mas precisas. Los atributos del pulso que pueden cambiar en las sucesivas iteraciones son la fase y duracion.

4 Diagrama de conexiones

Este diagrama representa la interconexion entre los diferentes modulos digitales y analogicos que forman parte de una experiencia planificada.



.

Figure 1: Diagrama de conexiones

4.1 Mixer

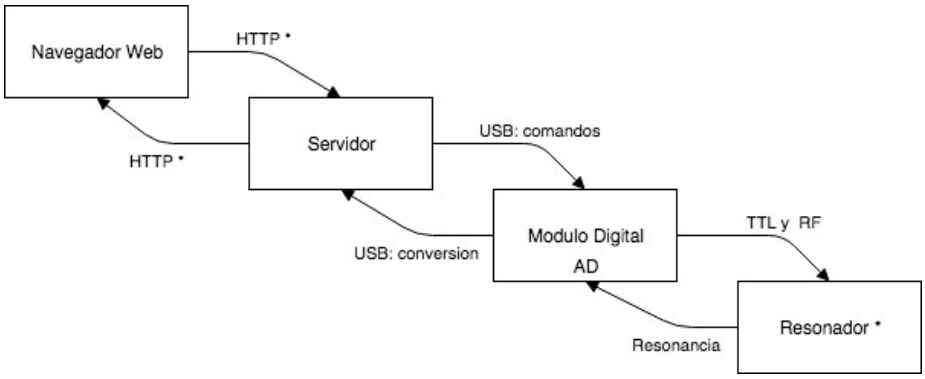
Los pulsos de salida y la señal de radiofrecuencia son las entradas del mezclador donde se convierten en una sola cuando el pulso TTL esta activo en 5 voltios. Esto permite crear pulsos de estimulacion y relajacion con presicion.

4.2 Amplificador

4.3 Resonador

5 Vision general del sistema

Este diagrama respresenta la interacion entre los diferentes elementos de hardware y software en una experiencia planificada.



.

Figure 2: Vision General del sistema

5.1 Navegador web

El navegador web es la plataforma donde se provee al usuario final la interfaz con el sistema.

5.2 Servidor

El servidor provee servicios REST solicitados por la interfaz grafica durante la vida de la sesion del usuario. Estos servicios hacen llamadas al controlador del Modulo Digital via usb.

5.3 Modulo Digital

El modulo digital procesa los mensajes del controlador via usb y ejecuta el microcodigo del mismo para la configuracion y ejecucion de las secuencias de pulsos.

5.4 Resonador

El resonador recibe los pulsos provenientes del Modulo Digital generando una señal de resonancia enviada al Modulo Digital para su conversion digital.

6 Descripcion del modulo digital

El siguiente diagrama de bloques muestra la configuracion interna del aparato junto a sus caracteristicas tecnicas.

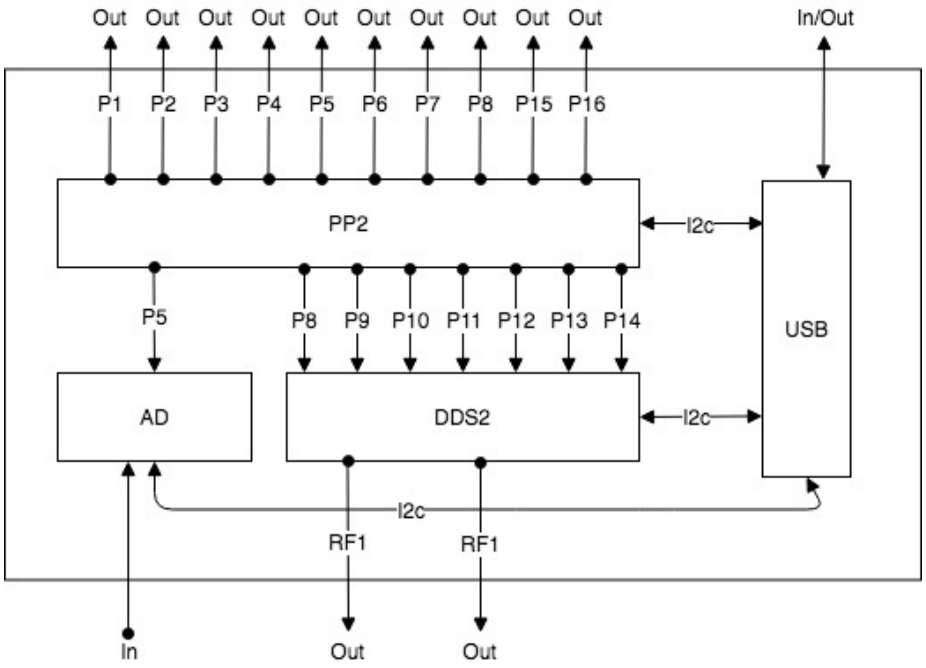


Figure 3: Diagrama de bloques del modulo digital

7 Submodulo DDS2

El DDS2 es un generador de señales con un rango de frecuencia de 0 a 200 Mhz.

El usuario puede almacenar hasta 2 frecuencias y un total de 16 fases para combinarlos en la ejecucion de un programa.

Tambien hay una relacion entre el PP2 y el DDS2, puesto que este tiene como entrada los Pulsos 8,9,10,11,12,13,14 para su manipulacion externa por aquel durante la ejecucion de un programa.

7.1 Configuracion inicial, activacion y desactivacion.

Direccion	B7	B6	B5	B4	B3	B2	B1	B0	Hex
1D	0	0	0	1	1	1	1	1	0x17
1E	0	1	0	0	0	1	0	0	0x44
1F	0	0	0	0	0	0	1	0	0x02
20	0	0	0	0	0	0	0	0	0x00

Table 1: Reset

Direccion	B7	B6	B5	B4	B3	B2	B1	B0	Hex
1D	0	0	0	1	0	0	0	0	0x10
1E	0	1	0	0	0	1	0	0	0x44
1F	0	0	0	0	0	0	1	0	0x02
20	0	0	0	0	0	0	0	0	0x00

Table 2: Activacion

Direccion	B7	B6	B5	B4	B3	B2	B1	B0	Hex
1D	0	0	0	1	1	1	1	1	0x17
1E	0	1	0	0	0	1	0	0	0x44
1F	0	0	0	0	0	0	1	0	0x02
20	0	0	0	0	0	0	0	0	0x00

Table 3: Desactivacion

7.2 Registros del DDS2

7.3 Fases

El DDS2 tiene disponible 32 posiciones de 8 bits para el almacenamiento de fases y cuenta con un bus de 1 byte.

El MSB de una fase debe almacenarse siempre en una direccion par de la RAM, luego el LSB de la misma en el valor impar siguiente contiguo.

Direccion	Descripcion	Modo
0x70	direccionamiento	Escritura
0x71	modo	Escritura
0x72	reset	Escritura
0x73	test	Lectura
0x74	señal de escritura	Escritura
0x75	direccionamiento	Escritura
0x76	señal de transferencia	Escritura
0x77	test	Lectura
0x78	señal de escritura	Escritura

Table 4: Resgistros internos del DDS2.

La fase es un valor de 14 bits, por lo que los 2 bits restantes del MSB son descartados.

Antes de almacenar el valor entero de la fase F debemos convertirlo a su equivalente en unidades de 45 de la siguiente manera:

$$F_h \in \mathbb{N} \tag{1}$$

$$F_l \in \mathbb{N} \tag{2}$$

$$F \in \mathbb{N} \tag{3}$$

$$F_h = (45 * F)/256 \tag{4}$$

$$F_l = (45 * F) - (F_h * 256) \tag{5}$$

direcciones ram disponibles	Modo
0x00 0x02 0x04 0x06 0x08 0x0A 0x0C 0x0E	0x02
0x10 0x12 0x14 0x016 0x18 0x1A 0x1C 0x1E	0x02

Table 5: Direcciones de Fase.

7.3.1 Algoritmo base de almacenamiento de fases

Algorithm 1 almacenamiento de una fase en direccion 0x00

1: **procedure** SAVEPHASE(F)
2: // MSB y LSB de valor de fase
3: $F_h = (45 * F)/256$
4: $F_l = (45 * F) - (F_h * 256)$
5: // direcciones contiguas
6: $dir_h \leftarrow 0x00$
7: $dir_l \leftarrow dir_h + 1$
8: // Modo carga de fases
9: $write(0x71, 0x02)$
10: // MSB
11: $write(0x70, dir_h)$
12: $write(0x74, f_h)$
13: // LSB
14: $write(0x70, dir_l)$
15: $write(0x74, f_l)$
16: // Modo PC
17: $write(0x71, 0x00)$

7.3.2 direccionamiento de fases

Los 16 valores de fase son direccionados con 4 pulsos correspondientes a los pulsos 11,12,13,14. Con el pulso 9 se activa la carga de fases y con el pulso 10 se transfiere la fase direccionada al registro de trabajo. El tiempo entre el pulso 9 y 10 debe ser menor a 100 nanosegundos.

7.4 Frecuencias

El DDS2 tiene disponible 12 registros internos de frecuencia, cada frecuencia ocupa 6 registros, por lo tanto se pueden almacenar 2 frecuencias.

Las frecuencias disponibles van de 0 a 200 Mhz y el valor que se almacena en los registros en representacion se obtiene con la siguiente calculo:

$$clock = 2 \times 10^6 \tag{6}$$

$$frec \in \mathbb{N} \wedge 0 < frec < clock \tag{7}$$

$$valor = frec \times (2^{48} - 1)/clock \tag{8}$$

Nro. Frecuencia	Registros	Modo
1	0x04 0x05 0x06 0x07 0x08 0x09	0x00
2	0x0A 0x0B 0x0C 0x0D 0x0E 0x0F	0x00

Table 6: Registros de Frecuencia.

7.4.1 Algoritmo base de almacenamiento de frecuencias

Algorithm 2 almacenamiento de una frecuecia de trabajo 1.

procedure SAVEFREQUENCY(F)
    // Conversion de la frecuencia deseada al valor
    $clock \leftarrow 2 \times 10^6$
    $value = frec \times (2^{48} - 1)/clock$
    // Modo PC
    $write(0x71, 0x00)$
    // primero MSB hasta LSB
    // Byte 5
    $write(0x75, 0x04)$
    $write(0x78, value_5)$
    // Byte 4
    $write(0x75, 0x05)$
    $write(0x78, value_4)$
    // Byte 3
    $write(0x75, 0x06)$
    $write(0x78, value_3)$
    // Byte 2
    $write(0x75, 0x07)$
    $write(0x78, value_2)$
    // Byte 1
    $write(0x75, 0x08)$
    $write(0x78, value_1)$
    // Byte 0
    $write(0x75, 0x09)$
    $write(0x78, value_0)$
    // Actualizacion registro de trabajo
    $write(0x76, 0x00)$

7.4.2 direccionamiento de frecuencias

Se admiten hasta 2 frecuencias de trabajo, se seleccionan con el pulso 8.

8 Submodulo AD

El submodulo conversor analogico digital (AD) tiene 2 canales de adquisicion, con una resolucion de 12 bits por canal, una frecuencia de muestreo maxima de 10 Mhz y capacidad de almacenamiento en bloques de 1KB,2KB,4KB,8KB,16KB,32KB,64KB,128KB.

Direccion	Descripcion	modo
0x0B	comando y control	lectura/escritura
0x0C	muestreo	escritura
0x08	canal B	lectura
0x09	canal AB	lectura
0x0A	canal A	lectura

Table 7: Registros del AD

8.1 Registro de comando

Bit	Descripcion
0	modo
1	modo
2	-
3	-
4	bloque
5	bloque
6	bloque
7	reset

Table 8: Registro de comando.

8.2 Registro de datos

Direccion	Descripcion
0x08	canal B
0x09	canal AB
0x0A	canal A

Table 9: Registros de datos.

8.3 Configuracion de un ciclo de adquisicion

Un ciclo de adquisicion requiere la configuracion del intervalo de muestreo de acuerdo a la siguiente formula:

$$period = 100ns \tag{9}$$
$$interval \in \mathbb{N} \wedge 0 < interval < 25500 \tag{10}$$
$$n = interval/period \tag{11}$$
$$delta = 255 - n \tag{12}$$
$$\tag{13}$$

8.3.1 Configuracion bloque de 1K muestreo 1 micro seg

Algorithm 3 Configuracion 1K 1micro

1: **procedure** SETUP
2:
3: $config \leftarrow 0x00$
4: $config \leftarrow config \vee 0x02$
5: $config \leftarrow config \vee 0x80$
6: $config \leftarrow config \wedge 0xFE$
7: $write(0x0B, config)$
8:
9: $config \leftarrow 0x00$
10: $config \leftarrow config \vee 0x02$
11: $config \leftarrow config \wedge 0x7F$
12: $config \leftarrow config \vee 0x01$
13: $write(0x0B, config)$
14:
15: $samples = 1000/100$
16: $delta = 255 - samples$
17: $write(0x0C, delta)$

8.4 Extraccion de los datos de la memoria interna

El bus de datos de la memoria del AD es de 8 bits y las muestras de 12 bits, siendo necesarias 3 lecturas de 8 bits y una operacion de particion para obtener la muestra final de los canales A y B.

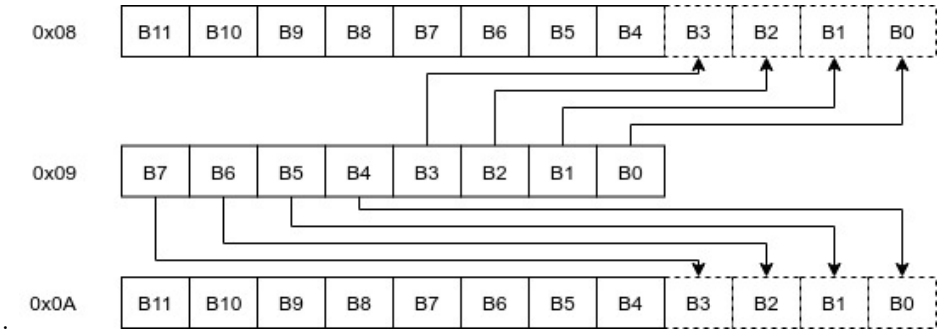


Figure 4: Buffers del submodulo AD

Para la extraccion de los datos es necesaria esta serie de pasos:

- modo PC, deshabilitada la adquisicion y reset contador de direcciones
- modo PC, deshabilitada la adquisicion y contador de direcciones en modo normal
- lectura del registro 0x08 del canal B
- modo PC, deshabilitada la adquisicion y reset contador de direcciones
- modo PC, deshabilitada la adquisicion y contador de direcciones en modo normal
- lectura del registro 0x0A del canal A
- modo PC, deshabilitada la adquisicion y reset contador de direcciones
- modo PC, deshabilitada la adquisicion y contador de direcciones en modo normal
- lectura del registro 0x09 con nibles del canal A y B.
- armado de los datos de los respectivos canales.

Algorithm 4 Extraccion de datos

```
1:  procedure SETUP
2:      config ← 0x00
3:      config ← config ∨ 0x02
4:      config ← config ∨ 0x80
5:      config ← config ∧ 0xFE
6:      write(0x0B, config)
7:      config ← 0x00
8:      config ← config ∨ 0x02
9:      config ← config ∧ 0x7F
10:     config ← config ∧ 0xFE
11:     write(0x0B, config)
12:     A ← read(0x08)
13:
14:     config ← 0x00
15:     config ← config ∨ 0x02
16:     config ← config ∨ 0x80
17:     config ← config ∧ 0xFE
18:     write(0x0B, config)
19:     config ← 0x00
20:     config ← config ∨ 0x02
21:     config ← config ∧ 0x7F
22:     config ← config ∧ 0xFE
23:     write(0x0B, config)
24:     B ← read(0x0A)
25:
26:     config ← 0x00
27:     config ← config ∨ 0x02
28:     config ← config ∨ 0x80
29:     config ← config ∧ 0xFE
30:     write(0x0B, config)
31:     config ← 0x00
32:     config ← config ∨ 0x02
33:     config ← config ∧ 0x7F
34:     config ← config ∧ 0xFE
35:     write(0x0B, config)
36:     AB ← read(0x09)
37:
38:     sample_A ← join_buffers(A, AB)
39:     sample_B ← join_buffers(B, AB)
```

9 Submodulo PP2

El programador de pulsos funciona como un microprocesador de instrucciones que generan un patron de salida por un tiempo determinado.

Un patron de salida es una combinacion de 16 pulsos TTL en paralelo por un periodo de tiempo determinado.

Un programa ejecutable por el PP2 sera una secuencia de no mas de 512 instrucciones y la ejecucion derivara en una secuencia de pulsos de duracion determinada.

9.1 Instrucciones

El PP2 tiene 4 instrucciones basicas: Continue, RetL, Loop, End, cada una es una secuencia de 64 bits con la siguiente estrucutura:

Patron de salida	Dato	Nivel de lazo	Codigo de instruccion	Demora
16 bits	11 bits	2 bits	3 bits	32 bits

Table 10: Estructura de las instrucciones.

9.1.1 Duracion de una instruccion

La base de tiempo es 40ns (nanosegundos. 1ns = 1* 10 EXP –9 seg.) Cada instrucción requiere de 4 pulsos de reloj (4 * 40ns = 160ns) y la demora mínima es 2 (2 * 40ns = 80ns). Por lo tanto el pulso de valle mínimo es de 160ns + 80ns = 240ns.

9.1.2 Continue

Mantiene una combinacion de 16 pulsos de salida por un tiempo determinado.

- patron de salida: es la combinacion de 16 pulsos de salida.
- dato: no utilizado.
- nivel de lazo: no utilizado.
- codigo de instruccion: 0x01.
- demora: duracion de la instruccion.

9.1.3 Loop

Marca el inicio de un bloque de repeticion.

- patron de salida: es la combinacion de 16 pulsos de salida.
- dato: contador de repeticiones.
- nivel de lazo: nivel de anidamiento entre lazos
- codigo de instruccion: 0x02.
- demora: duracion de la instruccion.

9.1.4 Retl

Marca de fin de un bloque de repeticion.

- patron de salida: es la combinacion de 16 pulsos de salida.
- dato: direccion de la instruccion Loop de inicio.
- nivel de lazo: no utilizado.
- codigo de instruccion: 0x03.
- demora: duracion de la instruccion.

9.1.5 End

Finaliza la secuencia de pulsos.

- patron de salida: es la combinacion de 16 pulsos de salida.
- dato: no utilizado.
- nivel de lazo: no utilizado.
- codigo de instruccion: 0x07.
- demora: duracion de la instruccion.

9.2 Registros del PP2

El PP2 cuenta con los siguientes registros de 8 bits:

Direccion	Descripcion	modo
0x50	comando	escritura
0x51	carga	escritura
0x52	señal	escritura

Table 11: Estructura de las instrucciones.

9.2.1 Carga de una instruccion en el PP2.

Algorithm 5 Carga de una instruccion en el PP2

1: **procedure** UploadProgram
2: // Reset
3: $write(0x50, 0x02)$
4: // Modo carga
5: $write(0x50, 0x03)$
6: // Continue 0x55AA 4
7: $write(0x51, 0x04)$
8: $write(0x51, 0x00)$
9: $write(0x51, 0x00)$
10: $write(0x51, 0x00)$
11: $write(0x51, 0x01)$
12: $write(0x51, 0x00)$
13: $write(0x51, 0xAA)$
14: $write(0x51, 0x55)$
15: $write(0x52, 0x00)$
16: // End
17: $write(0x51, 0x00)$
18: $write(0x51, 0x00)$
19: $write(0x51, 0x00)$
20: $write(0x51, 0x00)$
21: $write(0x51, 0x00)$
22: $write(0x51, 0x00)$
23: $write(0x51, 0x00)$
24: $write(0x51, 0x00)$
25: $write(0x52, 0x00)$

9.2.2 Ejecutar un programa.

Algorithm 6 Ejecutar un programa.

1: **procedure** Setup
2: // Modo microprocesador
3: $write(0x50, 0x00)$
4: // Señal de ejecucion
5: $write(0x08)$

10 Submodulo USB

11 Definicion de un experimento

En la actualidad una numerosa cantidad de proyectos utilizan notacion JSON para representar objetos por muchas razones entre ellas su facilidad para ser comprendido por parte de desarrolladores y el soporte buil-in para ser manipulado por navegadores web y nuestros lenguajes elegidos para desarrollar, python y javascript. Idealmente estos objetos son tambien sencillos de almacenar en bases de datos no relacionales.

Haciendo uso de aritmetica podemos deducir que la representacion de un experimento es liviana puesto que el numero de instrucciones de un programa P sera de $N < 512$ instrucciones lo que nos daria un peso aproximado en 50KB por experimento representado en esta notacion.

La validacion del esquema JSON de un experimento, sera llevada a cabo por un modulo programado dentro del sistema.

Cabe destacar que los tipos de datos presentes en la especificacion de un experiemento son soportados sin perdidad de presicion por parte de la notacion JSON.[1]

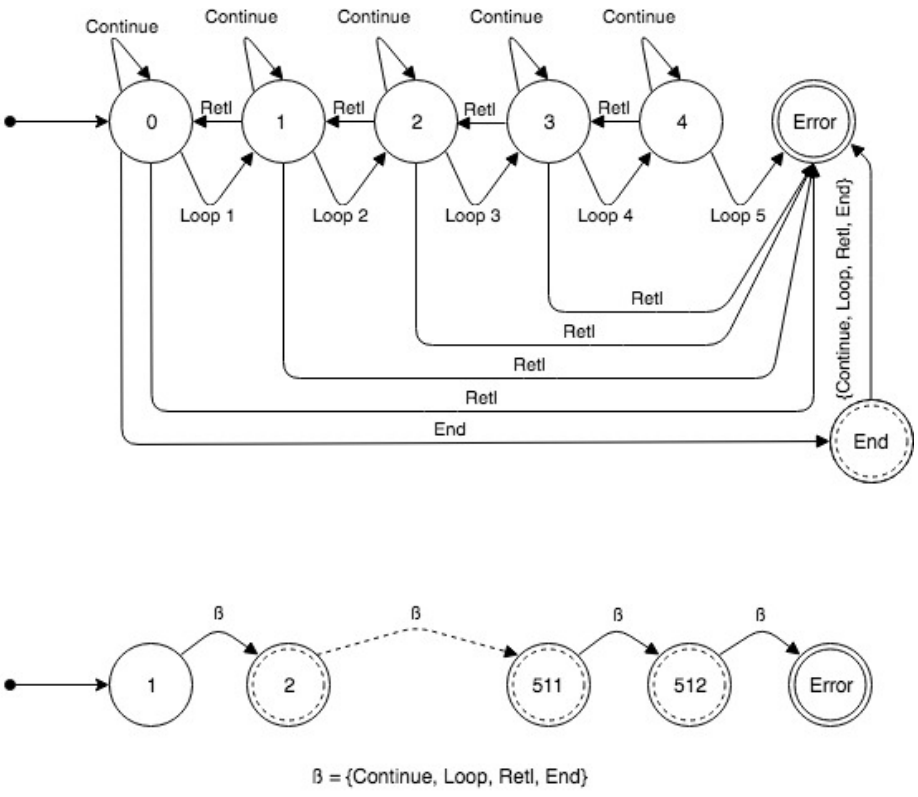
12 Algoritmo de traduccion de experimentos

Como mencionamos en la seccion 9 un programa almacenable y ejecutable por el PP2 es una secuencia de instrucciones $I_1...I_N$ con $1 < N < 512$.
Las instrucciones disponibles son:

- Continue
- Loop
- Retl
- End

No todos los programas escritos con estas instrucciones son admitidos como validos para el PP2, por esto tenemos las siguientes reglas:

- Regla 1: la instruccion End siempre es la ultima instruccion.
- Regla 2: la intruccion End aparece solo una vez.
- Regla 3: la intruccion End siempre esta presente.
- Regla 4: un bucle siempre inicia con instruccion Loop y finaliza con Retl.
- Regla 5: puede haber hasta 3 loops dentro de un loop.
- Regla 6: la longitud del programa no puede ser mayor a 512 instrucciones.



.

Figure 5: Los programas aceptados por el PP2.

Algorithm 7 Algoritmo de traduccion base

```
1: procedure TRANSLATE(P, L=0, S=[])
2:     ins ← pop(P)
3:     if ins = Continue then
4:         S.append(ins)
5:     if ins = Retl then
6:         if L > 0 then
7:             S.append(ins)
8:             L ← L − 1
9:             Translate(P, L, S)
10:        else
11:            return error
12:    if ins = Loop then
13:        if L < 4 then
14:            S.append(ins)
15:            L ← L + 1
16:            Translate(P, L, S)
17:        else
18:            return error
19:    if ins = End then
20:        if len(P) = 0 then
21:            S.append(ins)
22:        else
23:            return error
24:
25:    assert(0 < len(S) < 512)
26:    return S
```

13 Ejecucion de un experimento

Un modo de capturar fallos inesperados es simular la ejecucion de un experimento, evitando la interaccion con el modulo digital via usb.

En modo simulado el experimento se ejecuta en un entorno limitado con el objetivo de detectar tres situaciones no deseadas:

- una exepcion de software no capturada.
- parametros fuera de rango para alguno de los submodulos.
- un error en la traduccion del experimento a un programa ejecutable en el PP2.

Este experimento simulado no se ejecuta en un hilo separado puesto que la demora es baja.

De manera alternativa el modo simulado puede verse tambien como un test de integracion entre las unidades de software desarrolladas.

Si el experimento simulado tiene exito entonces se procede con la ejecucion efectiva del experimento el cual debemos tener algunos detalles en cuenta:

- puede durar horas.
- puede ser cancelado en cualquier momento.
- debe estar sincronizado con la base de datos.

El siguiente diagrama clases muestra el diseño de un experimento simulado y un experimento.

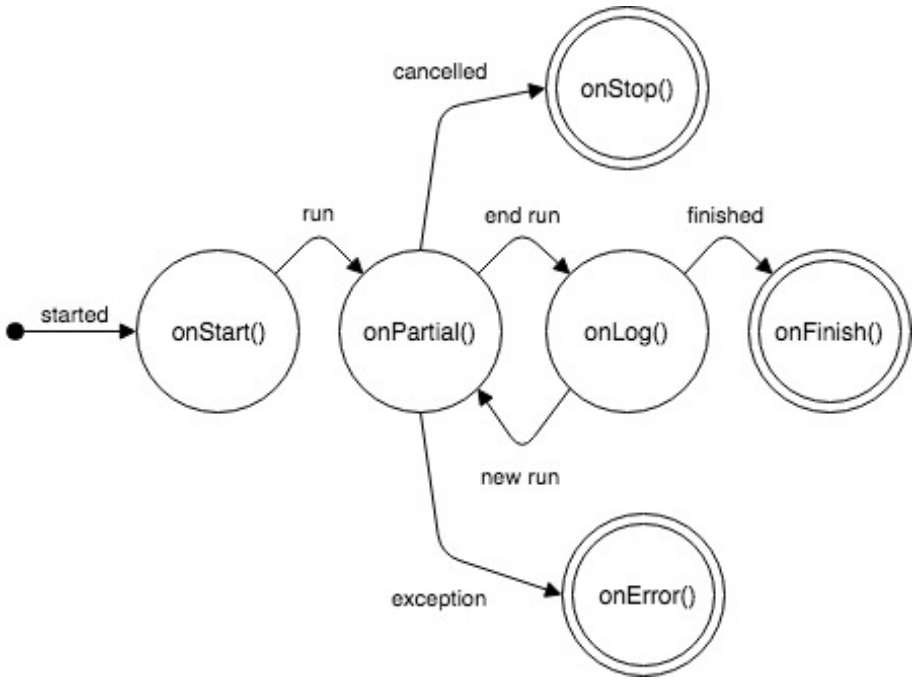
14 Hilo de ejecucion de experimento

La duracion de un experimento puede ser de horas, esta caracteristica involucra tres problematicas a resolver:

- evitar time-out del lado del cliente
- desacoplar la atencion de una peticion http de usuario de la ejecucion de un experimento
- lograr una experiencia de usuario agradable

Modelando el proceso de ejecucion con un hilo separado del principal es una aproximacion valida para resolver el problema y por lo tanto tener en cuenta los siguientes:

- seguimiento del estado del hilo
- control sobre el numero de hilos creados
- gestion de los datos durante la ejecucion del hilo



.

Figure 6: Estados de un hilo de experiemento

15 Run y DryRun

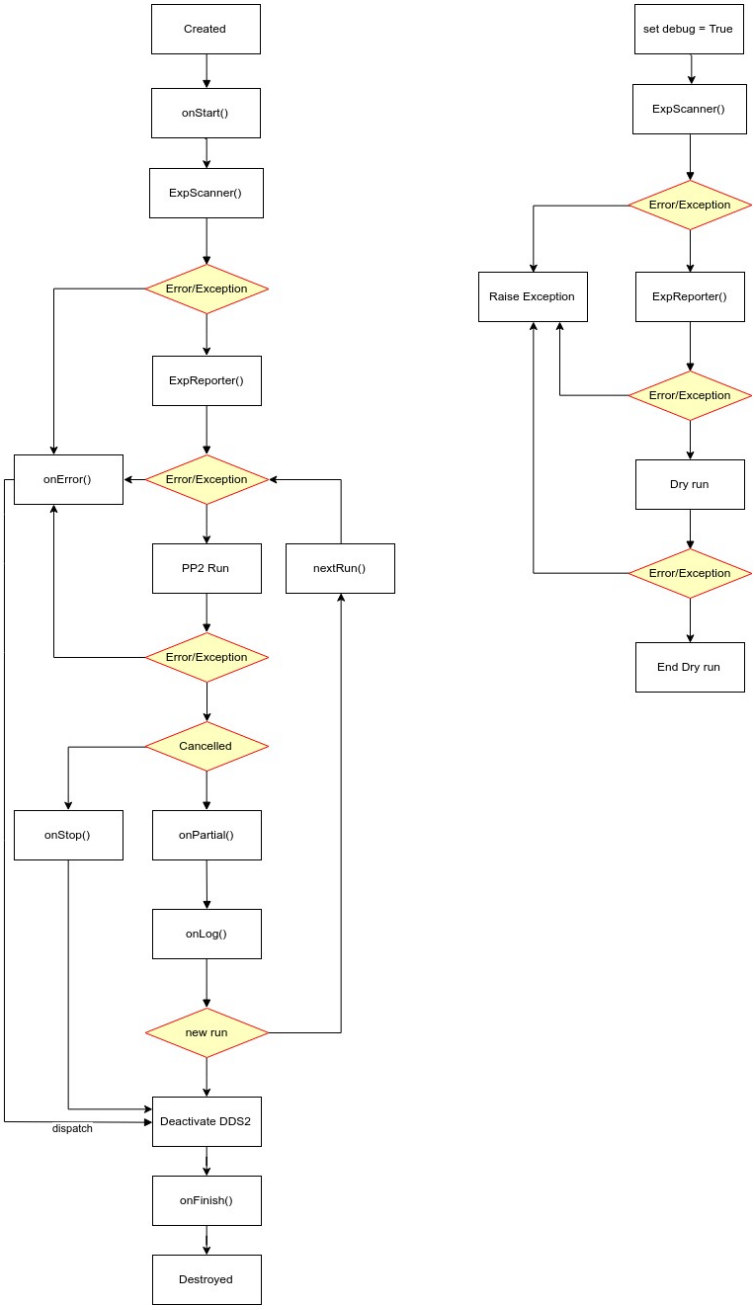


Figure 7: Run y DryRun

16 Requisitos del sistema

Podemos ver los requerimientos en generales

- permitir a mas de un usuario utilizar el sistema
- permitir a usuarios la gestion de experimentos
- modelar un experimento y su ejecucion
- proveer acceso remoto al sistema

De estos requerimientos se desprenden los siguientes especificos:

- el usuario tiene una sesion asignada al ingreso del sistema y revocada a la salida del mismo.
- varios usuarios pueden acceder al sistema al mismo tiempo
- el usuario puede ver si hay un experimento en ejecucion y el detalle del mismo
- el usuario tiene un espacio de trabajo donde puede crear/ver/actualizar/eliminar sus experimentos
- el usuario puede cancelar el experimento en ejecucion
- el usuario puede solicitar un reporte parcial de un experimento en ejecucion
- el usuario puede solicitar el reporte final de un experimento finalizada la ejecucion
- el usuario puede verificar el estado de un experimento:
- el usuario es notificado antes de la ejecucion de un experimento cuando:
  - salida voluntaria por error:
    - no es una secuencia ejecutable por el PP2
    - ya existe algun experimento ejecutandose
    - algun parametro de configuracion es invalido en algun submodulo PP2, DDS2, AD, USB
    - fallo en conexion via USB
  - salida involuntaria por error:
    - hubo alguna excepcion de sistema no controlada
    - no se pudo establecer conexion con el servidor
    - un experimento finalizado no actualizo su estado impidiendo la ejecucion de solcitido

- un experimento tiene una secuencia definidida con sus parametros
- un experimento tiene una marca de tiempo de creacion/actualizacion
- un experimento eliminado no es recuperable
- un experimento tiene un autor que es el usuario que lo creo
- un experimento tiene estado created cuando esta almacenado en base de datos
- un experimento es solo visible para el usuario que lo creo
- un experimento tiene un titulo
- un experimento tiene una descripcion
- un experimento no tiene un historial de edicion asociado
- un resultado es el producto de la ejecucion de un experimento
- un resultado es parcial cuando la ejecucion del experimento asociado no finalizo
- un resultado es final cuando la ejecucion del experimento asociado finalizo
- un resultado tiene un unico experimento asociado
- un reporte parcial es el grupo de datos de un resultado parcial
- un reporte final es el grupo de un resultado final
- un reporte contiene:
  - log de la ejecucion
  - datos del AD en formato CSV
  - experimento ejecutado
- el sitema tiene servicios REST para:
  - crear/ver/editar/eliminar experimentos
  - iniciar/cancelar ejecucion de un experimento
  - cancelar la ejecucion de todos los experimentos
  - descargar reportes parciales y finales
  - proveer autenticacion a todos los servicios
- el sistema tiene una interfaz de usuario web

17 Planificacion del desarrollo

De acuerdo con los requerimientos descriptos en el capitulo XX se propone implementar el sistema con:

- Django framework
- Pythton 2.7
- React js
- Javascript ECM6
- Git

Las tecnologias elegidas son ampliamente soportadas por la comunidad de desarrolladores, de codiglo libre y soporte multiplataforma en PC.

Con el objetivo de asegurar que lo primero que se desarrolle cumpla son los requeriminetos asociados con el control del Modulo Digital, luego proveer una manipulacion remota y finalmente una interfaz de usuario para el manejo a traves del navegador web, se organizo el desarrollo de la siguiente manera:

- Modulo Digital
- Servidor
- Interfaz Grafica Web

El mapeo entre los requerimentos y el orden propuesto genera el siguiente esquema de tareas a realizar.

- Modulo Digital
  - Configuracion del proyecto y entornos
  - Implementacion control AD
  - Implementacion control DDS2
  - Implementacion control PP2
  - Implementacion abstraccion Experimento
  - Implementacion de interfaz USB
  - Implementacion de un programa principal
  - Gestion de logs
  - Integracion con Servidor
- Servidor
  - Configuracion del proyecto y entornos
  - Servicios REST para la edicion de experimentos
  - Servicios REST para control de ejecucion de experimentos

  - Servicios REST para la generacion de reportes
  - Modelado de Base de datos
  - Configuracion de sesiones
  - Configuracion de acceso a los servicios REST
  - Gestion de logs
  - Integracion con Interfaz WEB
  - Integracion con Modulo Digital
- Interfaz WEB
  - Configuracion de proyecto y entornos
  - Implementacion de transiciones
  - Componente de edicion de experimentos
  - Componente de control de ejecucion de experimentos
  - Componente de login
  - Componente de visualizacion de errores
  - Componente de definicion de experimento
  - Implementacion de control de estados
  - Integracion con Servidor

18 Planificacion de Testing

La planificacion de testing del sistema tiene varios enfoques en respuesta a diferentes objetivos:

- brindar una experiencia predecible y agradable al usuario final
- lograr una integracion coherente entre modulos
- aumentar cobertura de requerimientos de usuario y sistema
- facilitar la comprension del comportamiento esperado del sistema

Por unidad existen las siguientes validaciones:

- Modulo Digital
  - entradas de configuracion para el PP2, DDS2 y AD
  - entradas de programa ejecutable por el PP2
  - existencia de conexion via interfaz USB durante la ejecucion de experimentos
  - salidas de señales RF
  - salidas de pulsos TTL
  - input de conversion de datos del AD
  - salida de reportes de los canales A y B.
- Servidor
  - autenticacion obligatoria en los servicios
  - ejecucion secuencial de los experimentos
  - aislamiento de espacio de trabajo de usuarios
  - coherencia en los estados de los experimentos
  - generacion de reportes
  - gestion de procesos de ejecucion de experimentos
- Interfaz de usuario web
  - visualizacion coherente de elementos y estilos
  - tiempos cortos de carga de las vistas
  - visualizacion de alertas de usuario
  - transiciones entre vistas

Dados los diferentes enfoques son necesarias se eligieron las siguientes para probar:

- Postman para simular comunicacion http con servicios REST

- Arduino para visualizar los pulsos TTL via Serial Plotter
- Osciloscopio para medir señales RF generadas en los experimentos.
- Generador de señales para simular adquisicion de datos del AD.
- Google Chrome y herramientas para validar estados de la interfaz grafica.
- Mock del servidor escrito en Flask-Python para simular servicios REST.

19 Arquitectura del Modulo Digital

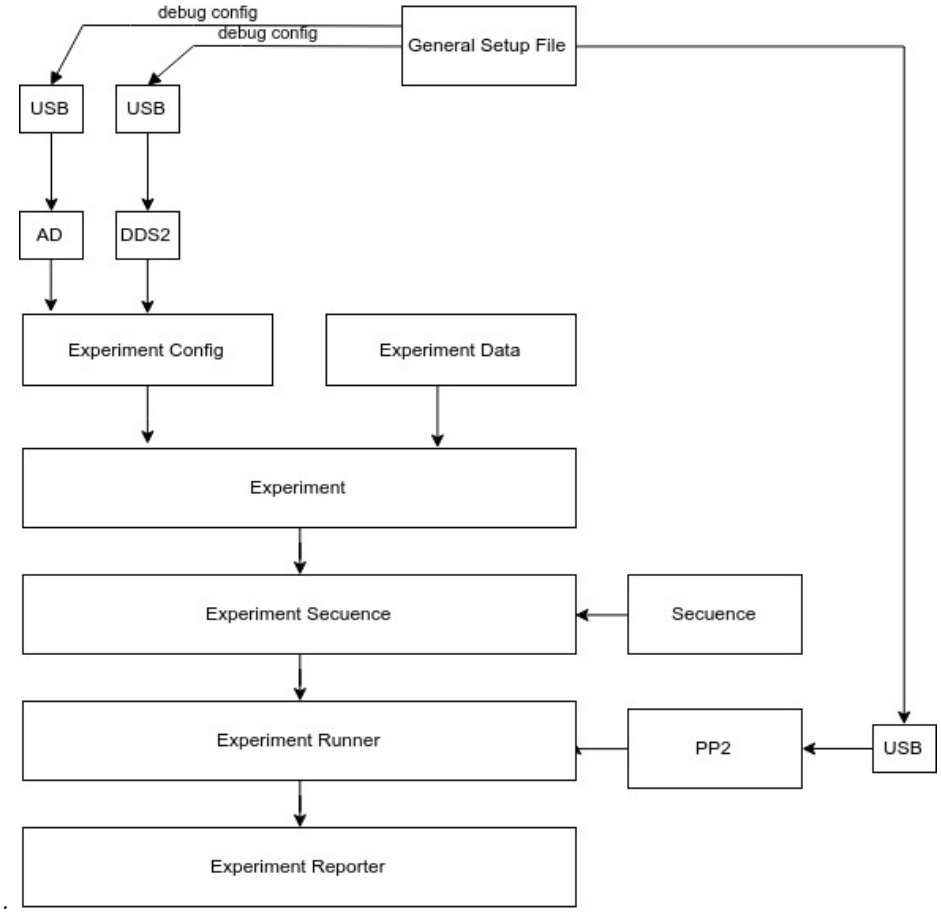


Figure 8: Arquitectura Modulo Digital

20 Arquitectura del Servidor

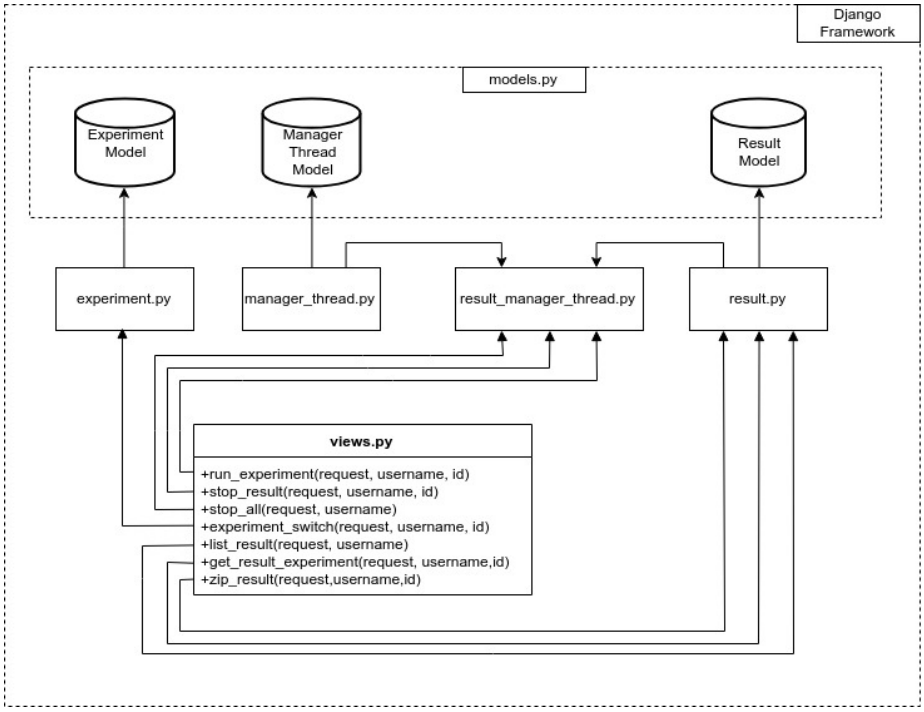


Figure 9: Arquitectura del Servidor

21 Arquitectura de Interfaz Grafica

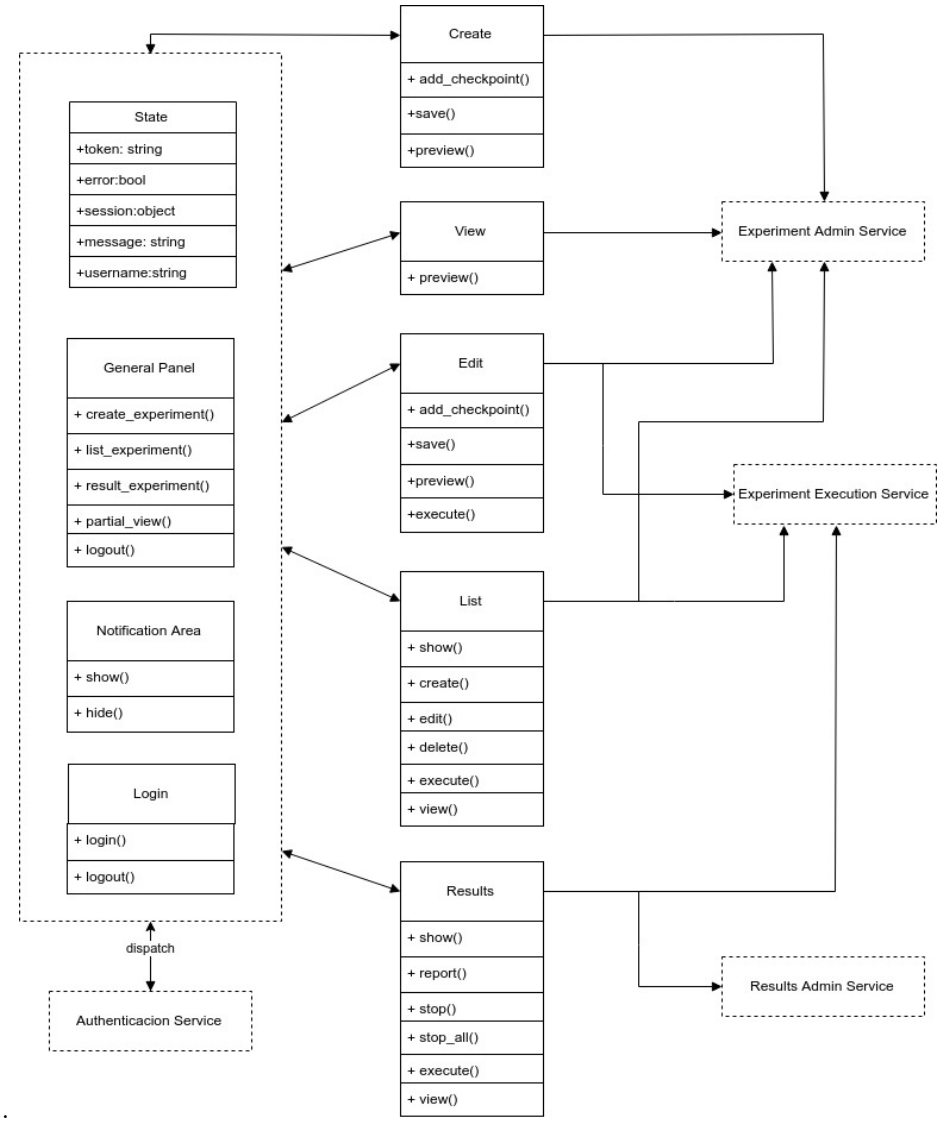


Figure 10: Arquitectura del UI

22 Integracion de los submodulos

La integracion de los modulos del sistema se implementa a diferentes niveles

22.1 Repositorio

Un submodulo git es un repositorio dentro de otro que lo incluye, con su propio historial de cambios. Asi mismo cuando hay un cambio en un submodulo el contenedor lo contemplara siempre y cuando lo agregue al historial de este. El repositorio afectado por esta metodologia es el del Servidor incluyento al Modulo Digital y a la Interfaz Grafica.

El arbol de directorios del repositorio queda asi:

```
Server
├── build
│   └── static
├── error
├── experiments
│   └── Modulo Digital
├── log
├── login
├── out
├── Interfaz Grafica
└── settings
```

22.2 Codigo

El servidor provee el ManagerThread para poder interactuar con el Modulo Digital y con el modelo de la base de datos segun el estado de la ejecucion del experimento en curso.

Para integrar con la Interfaz grafica lo hace de manera estandar via configuracion del servidor que provee seguridad y autenticacion de los servicios via CSRF-Token presente en el archivo principal index.html de la Interfax Grafica.

23 Analisis de Performance

Se propone como analisis de performace un experimento con las siguientes caracteristicas:

- 500 promedios
- 2 loops con un pulso cada uno
- bloque de 8K
- 4 fases

Los consumos de CPU - RAM - DISCO

24 Casos de Prueba

De acuerdo con los enfoques generales y por unidad se diseñaron los siguientes casos de prueba

24.1 Modulo Digital

- experimento con configuracion invalida del AD
- experimento con configuracion invalida del DDS2
- experimento con secuencia invalida para el PP2
- experimento un pulso largo
- experimento un pulso corto
- experimento con pulsos largos y cortos intercalados
- experimento promedio con un pulso
- experimento promedio con un pulso variable en duracion
- experimento promedio con un pulso variable en fase
- experimento un pulso dentro de un loop
- experimento un pulso dentro de un loop de nivel 4
- experimento con un RETL sin LOOP asociado
- experimento con un LOOP sin RETL asociado
- experimento con un LOOP de nivel 5

24.2 Servidor

- solicitud de servicio sin token de autenticacion
- solicitud de autenticacion con usuario no registrado
- solicitud de autenticacion con usuario registrado pero contraseña invalida
- solicitud de fin de sesion
- solicitud de creacion de un experimento
- solicitud de edicion de un experimento
- solicitud de edicion de un experimento inexistente
- solicitud de edicion de un experimento en ejecucion
- solicitud de eliminacion de un experimento

- solicitud de eliminacion de un experimento inexistente
- solicitud de eliminacion de un experimento en ejecucion
- solicitud de ejecucion de un experimento habiendo uno en ejecucion
- solicitud de ejecucion de un experimento inexistente
- solicitud de ejecucion de un experimento
- solicitud de cancelacion de una ejecucion en curso
- solicitud de reporte durante la ejecucion
- solicitud de reporte finalizada la ejecucion
- solicitud de reporte de un experimento que nunca se ejecuto
- solicitud de reporte de una ejecucion cancelada
- solicitud de reporte de una ejecucion fallida

24.3 Interfaz Grafica

- ingreso al sitio con usuario y contraseña validos
- ingreso al sitio con usuario ó contraseña invalidos
- salida del sitio con boton login
- crear un experimento
- visualizacion de lista de experimentos creados
- filtrado de lista de experimentos creados
- ver un experimento seleccionado de la lista
- edicion de un experimento seleccionado de la lista
- eliminacion de un experimento seleccionado de la lista
- ejecucion de un experimento seleccionado de la lista
- visualizacion de lista de resultados
- ver detalles de un resultado
- visualizacion de menu de gestion de experimentos
- visualizacion de menu de gestion de resultados
- visualizacion de menu de navegacion
- visualizacion de alertas

25 Errores conocidos

- si el sistema es interrumpido por un apagado imprevisto durante una ejecucion es posible que un estado sea inconcistente
- en vista edicion hacer click en boton "guardar y ejecutar" demora 5 segundos la transicion a vista de resultados
- las alertas deben ser cerradas para obtener nuevas si las hubiese
- soporte parcial a diferentes tamaños de pantalla

26 Limitaciones de la platforma

Existen limitaciones de diversos tipos:

- Plataforma
  - driver y dll para windows posiblemente reemplazable usando libusb
  - administracion de archivos limitada
  - soporte librerias de matematica y graficos limitada
- Modulo Digital
  - requiere la implementacion de un manager para su utilizacion
  - requiere permisos para creacion de archivos de logs y reportes
- Server
  - gestion del proceso del servidor es manual
  - la gestion de usuarios es manual via interfaz Django
  - seguimientos de recursos de sistema es manual
- Interfaz Grafica
  - programacion orientada a componentes
  - JSX una variacion de JavaScript
  - JQuery no es soportado

27 Mejoras a futuro

- Modulo digital
  - interfaces de servicio
  - independencia como programa de consola
  - test automaticos
  - integracion continua
- Server
  - cola de experimentos
  - soporte a otros Modulos Digitales u Aparatos
  - post-procesado de datos en tiempo real
  - test automaticos
  - integracion continua
- Interfaz Grafica
  - soporte a telefonos moviles
  - visualizacion de graficos
  - tests automaticos
  - vista de manipulacion de resultados
  - vista de gestion de perfil
  - tests automaticos
  - integracion continua

28 Analisis de esfuerzo

- Niveles:
  - Muy Alto
  - Alto
  - Medio
  - Bajo
  - Trivial
- Modulo Digital
  - Usb: Alto
  - PP2: Alto
  - AD: Alto
  - DDS2: Muy Alto
  - Reporte: Bajo
  - Hilo: Alto
- Server:
  - Servicios: Alto
  - Login: Alto
  - Integracion: Muy Alto
  - Modelado: Medio
  - Manager Hilos: Alto
- Interfaz Grafica:
  - Vistas: Alto
  - Integracion: Alto
  - Alertas: Alto

References

[1] JSON notation