Yo no sabía que decir, mi boca
no sabía
nombrar,
mis ojos eran ciegos,
y algo golpeaba en mi alma,
fiebre o alas perdidas,
y me fuí haciendo solo,
descifrando
aquella quemadura,
y escribí la primera línea vaga,
vaga, sin cuerpo, pura
tontería,
pura sabiduría
del que no sabe nada,
y vi de pronto
el cielo
desgranado
y abierto,
planetas,
plantaciones palpitantes,
la sombra perforada,
acribillada
por flechas, fuego y flores,
la noche arrolladora, el universo.

Fragmento: *La poesía*
Pablo Neruda

4

El nuevo sistema de adquisición de datos del Observatorio de Rayos Cósmicos

Hasta el momento hemos analizado las particularidades que un sistema de adquisción de datos para RC presenta. El siguiente paso es planear y proyectar el diseño del nuevo adquisidor de datos. Comenzaremos por mencionar los requerimientos técnicos del proyecto y las necesidades que este debe de cubrir.

En el caso del observatorio de rayos cósmicos de Ciudad Universitaria, el siguiente diagrama ilustra la distribución de los elementos que componen al SDAQ (Fig. 4.1).

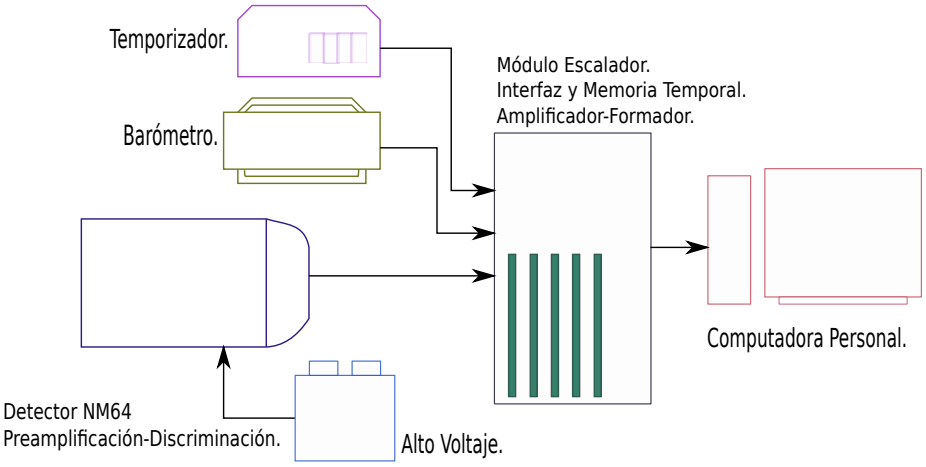

Figura 4.1: *Sistema de adquisción del observatorio de RC en CU.*

Se puede observar que el sistema, cumple con las características mencionadas

en la sección 3.2. Es importante notar que la función de almacenamiento de datos se lleva a cabo por medio de una PC y el modo de operación del sistema es en tiempo fijo, es decir, existe un módulo temporizador que lleva el control del tiempo de adquisición. Por medio de este tiempo se establecen los periodos de conteo y almacenamiento del SDAQ.

El nuevo sistema de adquisición debe cumplir con el mismo principio de funcionamiento. En el diseño no se considera, por el momento, la modernización de la etapa de preamplificación y discriminación ya que su funcionamiento es correcto.

El requerimiento más importante del diseño es fundamentar el proyecto sobre tecnología moderna. La principal razón tras esta idea es garantizar un mejor desempeño y prolongar su vida útil.

Igualmente, para prologar la vida útil del equipo y con vista hacia los nuevos retos que se presenten en el observatorio, se explorará la posibilidad de crear un instrumento que facilite su reproducción y rediseño.

De los elementos que componen el sistema, el temporizador es uno de los más importantes y requiere un cuidado especial en su diseño. En el observatorio de rayos cósmicos en CU, la tarea de temporización se realiza mediante un oscilador eléctronico y un contador digital. Tal circuito presenta el incoveniente de tener desviaciones en su frecuencia de operación y por lo mismo, variaciones en el tiempo de adquisición. Estas variaciones provocan que los datos de fecha y hora sean incorrectos. En consecuencia, se tiene la necesidad de ajustar periódicamente la frecuencia del temporizador y los datos de hora y fecha. La meta en el nuevo adquisidor, es lograr que el ajuste se realice de forma automática. Con esto se logrará una mejor autonomía, lo cual es deseable pues el sistema debe trabajar las $24 hrs$ del día, los 365 días del año.

Hasta ahora, no se ha explicado la utilidad de contar con un sensor de presión en el sistema —Fig 4.1. La importancia de este módulo reside en la influencia directa que ejerce la presión atmosférica sobre la radiación cósmica —Sección 1.1.2. Sí al mismo tiempo que obtenemos los datos de radiación, sensamos la presión, podemos realizar el ajuste correspondiente en las cuentas, para alcanzar un estimado de la intensidad de la radición cósmica sin el efecto atmosférico. El observatorio de rayos cósmicos en CU cuenta con un barómetro digital para realizar este trabajo. Dicho instrumento es confiable por lo que se buscará seguir obteniendo datos de él.

Contar con un mejor estimado pone al observatorio a la vanguardia y nos brinda información confiable. Una de las desventajas del NM64 es el *tiempo muerto* que se genera tras el arribo de una partícula. Durante este tiempo, el detector es incapaz de producir un nuevo pulso si se presenta otra partícula. Como consecuencia, sólo se genera un pulso de mayor duración. A este fenómeno se le conoce como *multiplicidad del detector* y se relaciona directamente con el

tiempo de relajación que sufre el gas tras ser ionizado. El nuevo adquisidor de datos contempla superar este obstáculo distinguiendo entre los pulsos de duración normal y aquellos que sobrepasan un umbral temporal.

Finalmente, en el nuevo diseño, la PC no sólo sirve como medio de almacenamiento, sino que realiza otras funciones. Entre estas funciones la más importante es el control de la adquisición de datos. El control de la adquisición consiste en una serie de instrucciones que permiten: iniciar la adquisción, detener la adquisición, transferir la información, cambiar el tiempo de adquisición y reestablecer las condiciones iniciales del sistema. En el caso de realizar pruebas o diagnósticos, una herramienta de dignóstico muy util es la generación de gráficas y despliegue de información. El uso de la PC facilita al usuario estas operaciones.

La exposición previa tiene la intención de plantear el problema a resolver y encaminarnos hacia la toma de decisiones. Se han expresado los requerimientos y necesidades del observatorio, así como los recursos con los que se cuenta. Es momento de plantear la pregunta; ¿qué camino tomar?.

4.1. ¿Qué camino tomar?

Sí comparamos las características que nos ofrecen los instrumentos modulares y los sistemas de aplicación específica, son claras las ventajas que nos proporciona el diseño a la medida.

La ventaja mas evidente del diseño a la medida, se encuentra en los productos que se obtienen durante el desarrollo del proyecto. Además de contar con un equipo que satisface nuestras necesidades, en el proceso de diseño se generan diagramas eléctricos, documentación y software. Dicha información es valiosa, debido a que facilita la comprensión del funcionamiento del sistema y proporciona material necesario para su mantenimiento, producción y rediseño.

En el caso de que el diseño se hiciera con algún sistema comercial no se contaría con estos beneficios. Por otro lado, sería necesario ajustar el equipo a nuestras necesidades debido a que ningún sistema cumple al cien por ciento los requerimientos expuestos.

De los sistemas de aplicación específica sólo se encuentra a nuestro alcance el desarrollo de sistemas embebidos basados en microcontroladores —MCU— o dispositivos lógicos programables —PLD— ya que los sistemas basados en ASIC's son demasiado costosos.

Un sistema embebido, desarrollado bajo cualquiera de estas tecnologías, puede cumplir con las necesidades del observatorio. Sin embargo, se prefiere el desarrollo de sistemas embebidos basados en PLD´s por las siguientes razones:

Procesos simultáneos e independientes. Esta característica puede ser explotada de manera muy eficiente en el diseño del sistema. Como se explicó con anterioridad, el monitor NM64 está compuesto de contadores proporcionales y cada uno de ellos entrega un tren de pulsos. El procesamiento de la información de cada uno de estos contadores se realiza de manera simultánea e independiente.

Facilidad en el rediseño del sistema. La elección del tipo de dispositivo depende en gran parte de su capacidad de sintetizar funciones lógicas —número de celdas lógicas. Sí se requiere agregar un mayor número de módulos al sistema, es decir, otros procesos que trabajen de forma independiente, únicamente se demandaría un dispositivo de mayor capacidad.

Independencia entre el tipo de dispositivo y HDL´s. Como ventaja extra hacia el futuro del sistema, se tiene la flexibilidad que nos brindan los HDL´s con respecto a la elección de la plataforma de desarrollo y tipo de dispositivo —Veáse Sección 3.3.2.

Por último, debemos enfatizar la capacidad de los PLD's para realizar diseños cien por ciento a la medida.

4.2. Concepción del sistema

Para comenzar el diseño, se requiere tener un panorama general del funcionamiento de nuestro sistema. Podemos visualizar nuestro sistema como una *caja negra*, cuyo contenido es desconocido. Se distinguen dos tipos de señales: aquellas que se desean procesar —señales de entrada— y aquellas señales que son resultado del procesamiento —señales de salida.

El primer grupo comprende la señal proveniente del monitor NM64 —el funcionamiento de este detector fue detallado en el capitulo dos— y la señal proveniente del sensor de presión Meteolabor AG GB1.

Consideraremos dos señales más, la señal de tiempo y la señal de usuario. La primera servirá para llevar el control del tiempo de adquisición, el periodo de tiempo durante el cual se acumulan los datos antes de almacenarlos y reiniciar el proceso. La señal usuario es el conjunto de instrucciones que da dirección al sistema sobre las tareas a realizar.

Como resultado del procesamiento se espera obtener: número de partículas que atraviesan el detector por unidad de tiempo —cuentas sencillas—, estimado del número de partículas que no se logra contar debido a la multiplicidad del detector o *cuentas múltiples* —este punto se retomará más adelante—, datos de

presión atmosférica, fecha, hora y gráficas con el despliegue de información que permitan diagnosticar el funcionamiento del detector de neutrones.

El siguiente paso del diseño fue determinar los módulos que constituyen nuestra *caja negra*. El funcionamiento de cada uno de estos subsistemas será descrito por medio de sus entradas y salidas (Fig. 4.2).

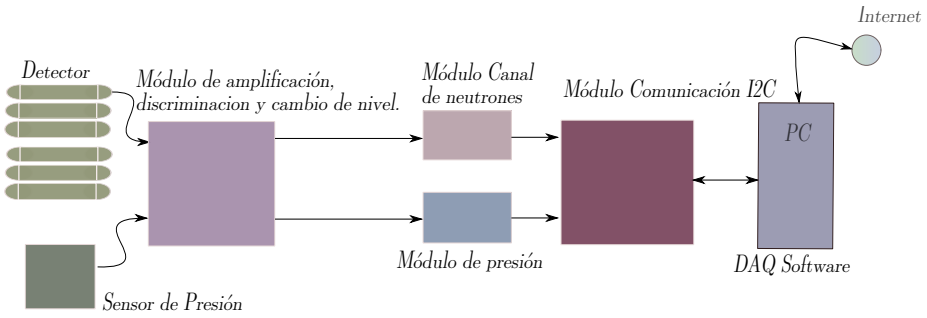

Figura 4.2: *Módulos que componen al nuevo sistema de adquisición de datos.*

Un módulo será el encargado de procesar la señal del detector y producir la información de cuentas múltiples y cuentas sencillas, dicho módulo tiene por nombre, módulo de *canal de neutrones*.

Otra parte del sistema lleva por nombre *módulo de presión*. Este módulo recibe la señal del sensor Meteolabor AG GB1 y como resultado del procesamiento se obtienen los datos de presión atmosférica.

El módulo de *canal de neutrones* no puede trabajar directamente con la señal proveniente del monitor NM64 ya que esta señal, además de ser de bajo nivel, contiene ruido eléctrico y otras interferencias. Por esta razón, previo al conteo de pulsos, es necesario agregar una etapa de acondicionamiento.

De la misma forma se requiere acondicionar la señal del sensor de presión para poder trabajar con ella.

Al módulo que nos proporciona la conexión entre el sensor de presión y el módulo de presión, el NM64 y el módulo *canal de neutrones*, se le denomina *módulo de amplificación, discriminación y cambio de nivel*.

Es importante recordar que los datos deben ser almacenados de forma periódica. Como se mencionó anteriormente, se utilizará una PC como medio de almacenamiento, control del sistema y despliegue de información. De esta manera, es necesario contar con un módulo que facilite la comunicación entre la PC y el resto del sistema —*módulo de comunicación* I^2C. La entrada al módulo es la información de cuentas múltiples, cuentas sencillas y datos de presión atmosférica. A la salida se tiene una trama de datos que pueden ser procesada por la computadora.

Para que la PC ejecute las funciones mencionadas, se necesita de un software especializado, al que nos referiremos como *módulo de software de adquisión*. Como entrada de este módulo se tiene: la señal de tiempo, la cual se obtiene de un servidor; la señal de usuario y la trama de datos. Como salida se tiene: señales de control, gráficas con despliegue de información y datos de fecha y hora.

Los módulos que se encargan de procesar las señales provenientes del monitor NM64, el sensor Meteolabor AG GB1 y el módulo de comunicación I^2C, se sintetizarán a través del dispositivo lógico programable.

Por esta razón, el diseño del módulo *canal de neutrones* se hará considerando una sola entrada, es decir, solo se tomará en cuenta la señal de un contador proporcional. Al final, aprovechando las propiedades de los PLD's, se replicará este módulo para cubrir la cantidad de contadores necesaria.

Una vez precisados los requerimientos del diseño, las necesidades a cubrir y haber delineado la concepción del proyecto, continuaremos con la exposición del diseño de cada uno de los módulos y su interconexión. El funcionamiento del sistema, pruebas y resultados será motivo del capítulo final de esta tesis.

4.3. Módulo de comunicación I^2C

Retomando lo expuestó en la sección anterior, el objetivo principal del módulo de comunicación I^2C es facultar al sistema de adquisición de datos con una interfaz de comunicación. Tal interfaz comunica dispositivos periféricos con una computadora personal.

Previo a la descripción de este módulo, profundizaremos en el protocolo de comunicación I^2C, protocolo sobre el cual basamos nuestro diseño.

4.3.1. ¿Qué es el bus I^2C?

El bus *Inter-Integrated Circuit* —I^2C— es un bus de comunicaciones serie que permite la comunicación entre diversos dispositivos conectados al bus y que normalmente se encuentran en la misma placa de circuito. Actualmente es una de las interfaces de comunicación serie más utilizadas y ha sido integrada en más de mil circuitos integrados diferentes. Entre sus aplicaciones se encuentran: lectura de sensores de dignóstico dentro de una PC, lectura de relojes en tiempo real, encendido y apagado de fuentes de voltaje de un sistema, entre otras.

El estándar fue desarrollado por Phillips a finales de los años 80's y cuenta con las siguientes características:

Interfaz de comunicación serie. El I^2C utiliza dos líneas de comunicación: *SCL* y *SDA*. La línea *SCL* proporciona un reloj de referencia para la

transferencia de datos. La línea *SDA* asegura que los datos serie esten sincronizados con los cambios de la señal de reloj.

Protocolo de comunicación sincrona. La transferencia de datos siempre es iniciada por un dispositivo que denominaremos *maestro*. Por medio de una señal de reloj, que genera el maestro, se sincroniza la transferencia de información.

Cumple con el modelo maestro-esclavo. El dispositivo maestro controla la señal de reloj e inicia y finaliza la comunicación. Esta señal dicta los tiempos de transmisión en el bus. Los esclavos no tienen control alguno sobre la comunicación.

Transferencia de datos bidireccional. La información en el bus puede fluir en cualquier dirección: de maestro a esclavo o de esclavo a maestro.

Por otra parte, el bus soporta velocidades de transmisión de $100^{kbits}/_{s}$ en el modo estándar y hasta $400^{kbits}/_{s}$ en modo rápido. El bus I^2C puede accesar hasta 128 dispositivos conectados a él.

Existen varias razones que hacen al bus I^2C idóneo para nuestro proyecto. La principal razón yace en su estructura sencilla, que requiere de pocos elementos. Esto implica que su síntesis en un PLD es factible.

Otra ventaja se encuentra en la gran cantidad de dispositivos que incluyen una interfaz I^2C y su facilidad para ser interconectados al bus. Dentro de estos dispositivos se encuentran: sensores de presión, temperatura, de campo magnético y relojes en tiempo real. Estos módulos bien podrían ser útiles en un futuro.

Dentro de los incovenientes que presenta el bus se encuentran su baja velocidad de transmisión y corta distancia de interconexión. Para nuestro diseño ninguno de estos presenta una limitante.

4.3.2. Protocolo de comunicación I^2C

Las líneas *SDA* y *SCL* se conectan a un voltaje positivo por medio de una resistencia de *pull-up* (Fig. 4.3). Esta resistencia es necesaria ya que los dispositivos que se conectan al bus son de *Open Drain*. Esto quiere decir que a su salida tienen un transistor de efecto de campo sin polarización. Este transistor se comporta como un interruptor; cuando el dispositivo quiere enviar un *uno lógico*, el transistor se conecta en alta impedancia; cuando el dispositivo quiere enviar un *cero lógico*, el transistor se conecta a $0V$.

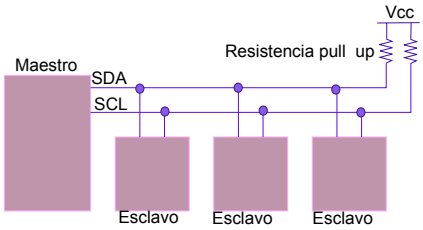

Figura 4.3: *Conexión de dispositivos al bus.*

Esta consideración es importante si se desea diseñar un dispositivo que se conecte al bus.

El protocolo de comunicación I^2C (Fig. 4.4) se expone en los siguientes pasos:

1. Antes de la transmisión, la línea *SCL* y *SDA* se encuentran en nivel alto. Sí el maestro desea inciar la transferencia, genera una condición de *START*. Esta condición se define como la transición de nivel alto a nivel bajo de la línea *SDA*, mientras la línea *SCL* permanece en alto.

2. EL maestro genera nueve pulsos de reloj en la línea *SCL*. En los primeros siete pulsos, en la línea *SDA* se encuentra la dirección del esclavo con el que se desea establecer la comunicación. Cada esclavo tiene una dirección única en el bus. El octavo bit indica al esclavo qué operación desea realizar el maestro. Sí el valor de *SDA* es bajo, el maestro desea escribir en el esclavo; sí el valor es alto el maestro desea leer datos del esclavo.

3. Si la dirección enviada por el maestro es reconocida por algún esclavo conectado al bus, éste pone en nivel bajo la línea *SDA*. Esto sucede durante el noveno pulso generado por el maestro y se conoce como ACKNOWLEDGE —*ACK*. En caso de que el maestro no reciba el *ACK* por parte de un esclavo, la comunicación finaliza.

4. Después de que el maestro recibe el ACK, la comunicación puede seguir dos caminos distintos:

   - Sí el maestro va a escribir en el esclavo, genera los pulsos necesarios para transmitir los datos a grabar, en paquetes de $8bits$. Al finalizar cada paquete el maestro genera un pulso de reloj extra para que el esclavo conteste con un *ACK*.

- Sí el maestro va a leer datos del esclavo, genera los pulsos necesarios para que el esclavo envie la información que el maestro esta esperando. Tras cada *byte* recibido, el maestro genera un pulso de reloj extra y le informa al esclavo que recibió el dato.

5. Para finalizar la transmisión, el maestro genera la condición de *STOP*. Esta condición se define como la transición de nivel bajo a nivel alto en *SDA* mientras *SCL* esta en nivel alto. La condición de *STOP* también se genera cuando el maestro no recibe *ACK* de ningún esclavo.

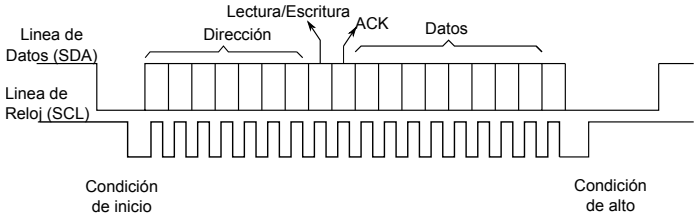

Figura 4.4: *Protocolo de comunicación I^2C*

Sólo el maestro puede generar las condiciones de *START* y *STOP* en el bus. Solamnete en estas condiciones se permite un cambio de *SDA* mientras *SCL* esta en alto. En cualquier otro caso *SDA* debe mantener un valor constante cuando *SCL* este en nivel alto.

4.3.3. Descripción del módulo I^2C

En todo esquema de comunicación I^2C, lo primero es definir el dispositivo maestro y los dispositivos esclavos. En el caso de nuestro proyecto la PC ocupará el lugar del maestro y el resto del sistema —Módulo *canal de neutrones* y Módulo de *presión*— figura como esclavo.

La elección del maestro, conviene al proceso de adquisición de datos debido a que la PC lleva el control del tiempo de adquisición y recibe las instrucciones por parte del usuario.

En conclusión, la descripción del módulo de comunicación I^2C en el PLD corresponde a la configuración del esclavo. La especificación del maestro en la PC se reanudará en una sección posterior.

El módulo de comunicación I^2C se compone de tres subsistemas: *startstop*, *asmi2c* y *control* (Fig. 4.5). Estos elementos trabajando en conjunto tienen como tarea responder con *ACK* cuando el maestro invoque al esclavo a través de su dirección, recibir las instrucciones por parte del maestro y decodificarlas, y por último enviar la información requerida por el maestro.

La unidad *startstop* tiene la función de generar una señal *busy* que indique

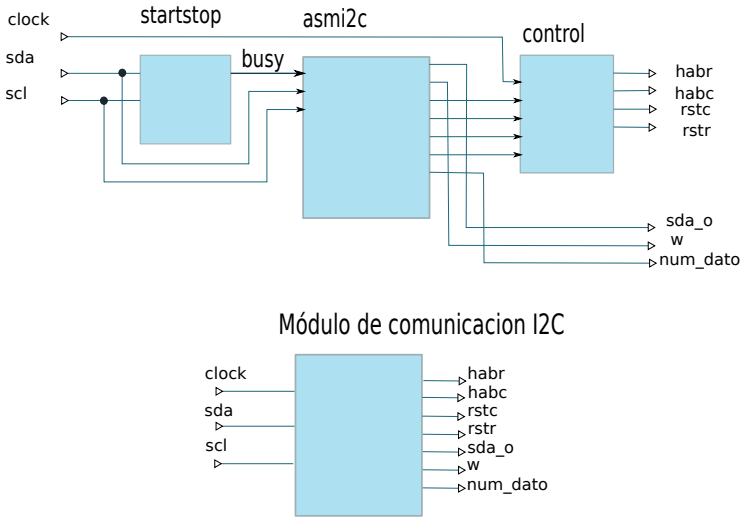

Figura 4.5: *Diagrama del módulo de comunicación I^2C.*

al resto del módulo cuando el maestro está ocupando el bus para transmitir. El valor de esta señal es alto cuando el bus está libre y bajo cuando el bus está ocupado.

El modo de operación de este módulo se describe de mejor manera por la siguiente figura:

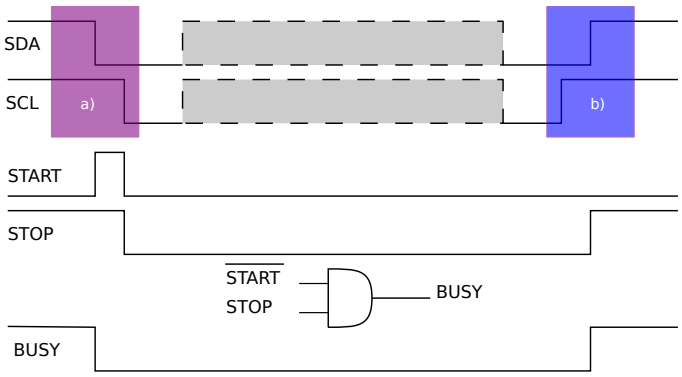

Figura 4.6: *Principio de funcionamiento del circuito startstop. a) Se detecta la condición de START con el flanco de bajada de SDA, mientras SCL se encuentra en alto. b) Se detecta la condición de STOP con el flanco de subida de SDA, mientras SCL se encuentra en alto.*

El siguiente elemento es el *asmi2c*, el cual está compuesto de una máquina de estados y un contador. Con ayuda de las señales *busy*, *SDA* y *SCL*, la máquina de estados se encarga de reconocer la dirección que envia el maestro e indentificar

las operaciones que el maestro desea hacer con el esclavo.

Sí la operación es de escritura, la máquina de estados recibe la instrucción y la traduce al resto del sistema. Son tres las instrucciones que se pueden recibir: *reestablecer*, *iniciar adquisción* y *copiar datos*. Para cada operación se genera una señal *a*, *b* o *c*, respectivamente. Estas señales entran al módulo de control.

En caso de tener una operación de lectura, la máquina de estados controla el envio de datos por parte del esclavo. Para enviar los datos es necesario convertirlos de paralelo a serie. Esta tarea se realiza con ayuda del contador y un multiplexor externo al módulo. El contador trabaja con la señal de *SCL* y sólo se habilita cuando la máquina de estados lo permite. El multiplexor tiene conectado a su línea de selección la salida del contador, lo que le permite poner los datos en el bus de manera ordenada.

El algoritmo que sigue la máquina de estados es el siguiente (Fig. 4.7):

1. Recibir dirección y enviar *ACK* cuando la dirección corresponde a la del esclavo. En caso contrario se espera el fin de la transmisión.
2. Sí la dirección fue correcta, identifica la tarea a realizar. Se tienen dos opciones; leer datos o escritura de datos.
   a) En caso de que la operación sea leer datos, se habilita el contador y el módulo que se encarga de colocar los datos en el bus. La habilitación permanece hasta que el envio de datos se completa.
   b) En caso que la operación sea escrituria de datos, espera hasta recibir la instrucción e identifica:

      Reestablecer. Sí la operación es *reestablecer* genera la señal *a*.

      Iniciar. Sí la operación es *iniciar adquisición* genera la señal *b*.

      Copiar. Sí la operación es *copiar datos* genera la señal *c*.
3. Tras concluir alguno de los pasos anteriores se espera a que se indique el fin de la tranmisión.

Finalmente, el módulo *control* traduce las señales *a*, *b*, *c* a los módulos *canal de neutrones* y presión. Las señales que genera este módulo son; *habr*, *habc*, *rstr* y *rstc*; estas señales representan el control del sistema y en ellas se define el proceso de funcionamiento.

Cuando el SDAQ se enciende, los módulos que lo componen pueden no encontrarse en condiciones inciales. Para garantizar que el sistema parta de un estado definido, el maestro envia la instricción *reestablecer*. Como resultado de reestablecer el sistema; las señales *habr* y *habc* se encuentran en nivel bajo; y las señales *rstr* y *rstc* estan en nivel alto. En otras palabras, en este punto el sistema se encuentra detenido, sin ningún valor almacenado y esperando la instrucción *incio de adquisción*.

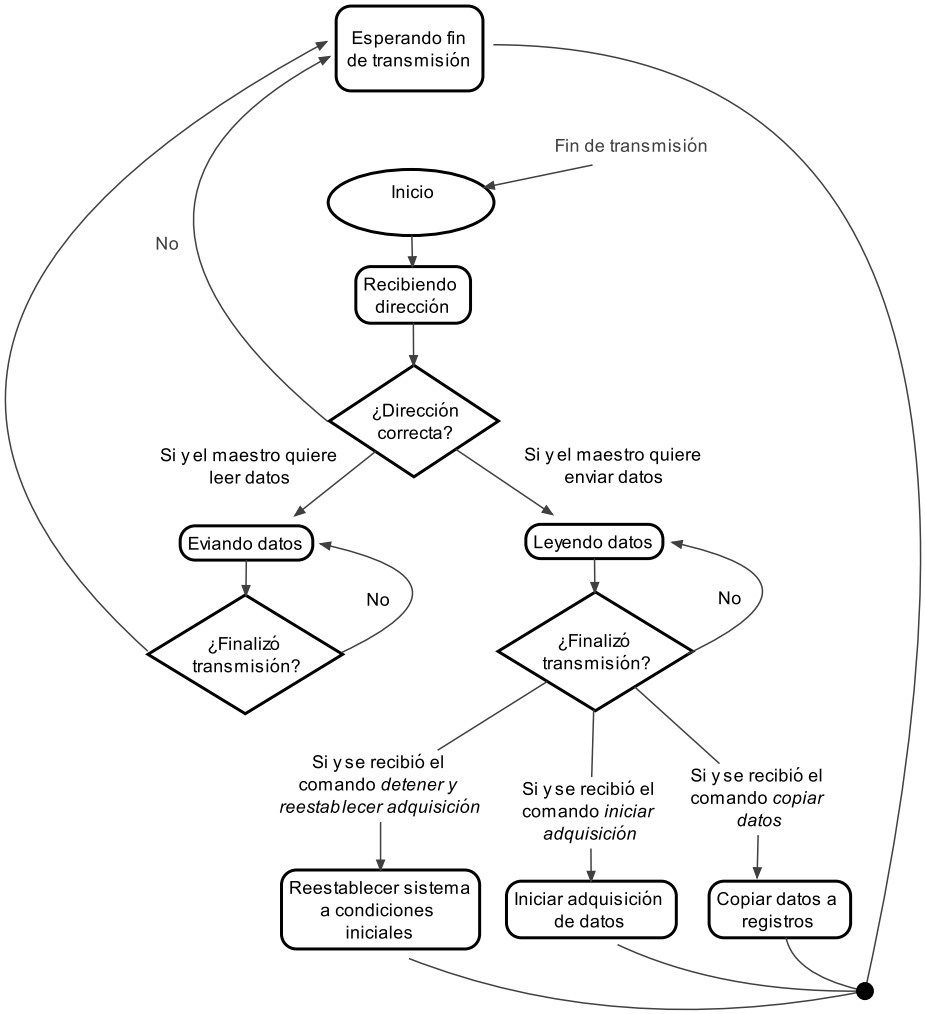

Figura 4.7: *Diagrama de flujo de la máquina de estados del módulo asmi2c.*

En el instante que el maestro manda la instrucción *incio de adquisición*, la señal *habc* pasa a nivel alto y las señales *rstr* y *rstc* pasan a nivel bajo. En este punto el sistema se encuentra adquiriendo datos.

Al finalizar el tiempo de adquisición, el maestro envia el comando *copiar datos*. El sistema detiene la adquisición por un lapso corto de aproximadamente $10\mu s$. En el transcurso de este periodo *habc* permanece en nivel bajo, mientras se copian los datos a una memoria temporal —*habr* pasa a nivel alto. El proceso anterior tiene como finalidad; retener los datos mientras estos se transmiten y reanudar el proceso de adquisición lo más pronto posible —$10\mu s$— sin que el envio interfiera en esto. Antes de reanudar el proceso de adquisición, es necesario

reestablecer el módulo *canal de neutrones*.

El algoritmo que describe el comportanmiento del módulo de control se expresa a continuación (Fig. 4.8):

1. Se establecen condiciones inciales. Sólo se puede regresar a este punto mediante la instrucción *reestablcer*.
2. Esperar comando *inicio de adquisición*.
3. Esperar finalice tiempo de adquisición y se envie la instrucción *copia de datos*.
4. Se detiene la adquisición de datos y se copian los datos.
5. Se reestablece el módulo *canal de neutrones* y se regresa al punto 3.

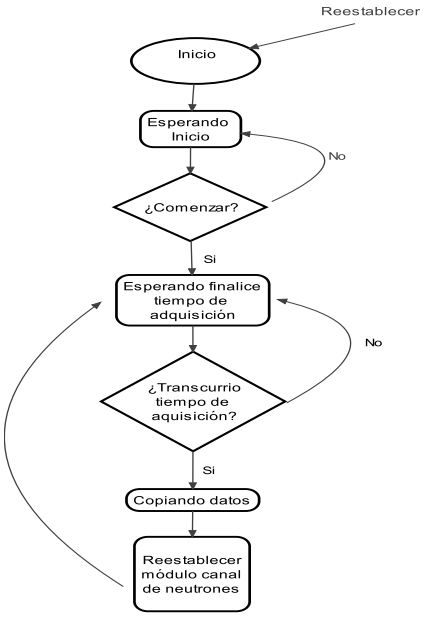

Figura 4.8: *Diagrama de flujo de la máquina de estados de control.*

La máquina de estados require de una señal de reloj externa para su funcionamiento, al igual que muchos otros circuitos de nuestro diseño. Sin embargo, por el momento, no profundizaremos en este punto. Será en el apartado final de este capítulo donde hablaremos de las señales que sincronizan al sistema y el módulo que es necesario para generarlas.

4.4. Módulo de canal de neutrones

El módulo *canal de neutrones* tiene la finalidad de contar los pulsos provenientes del monitor NM64 y posteriormente almacernarlos en un registro (Fig. 4.9). El sistema se compone de dos contadores: contador A y contador B.

El contador A incrementa su cuenta cada vez que detecta un flanco de subida en el pulso proveniente del detector. Al dato que se obtiene como resultado lo llamamos *cuentas sencillas*.

El contador B registra todos aquellos pulsos que tienen una duración mayor de $20\mu s$. Se toma como referencia esta duración ya que este parámetro define la duración que tienen la mayoría de los pulsos producidos por el detector. Aquellos

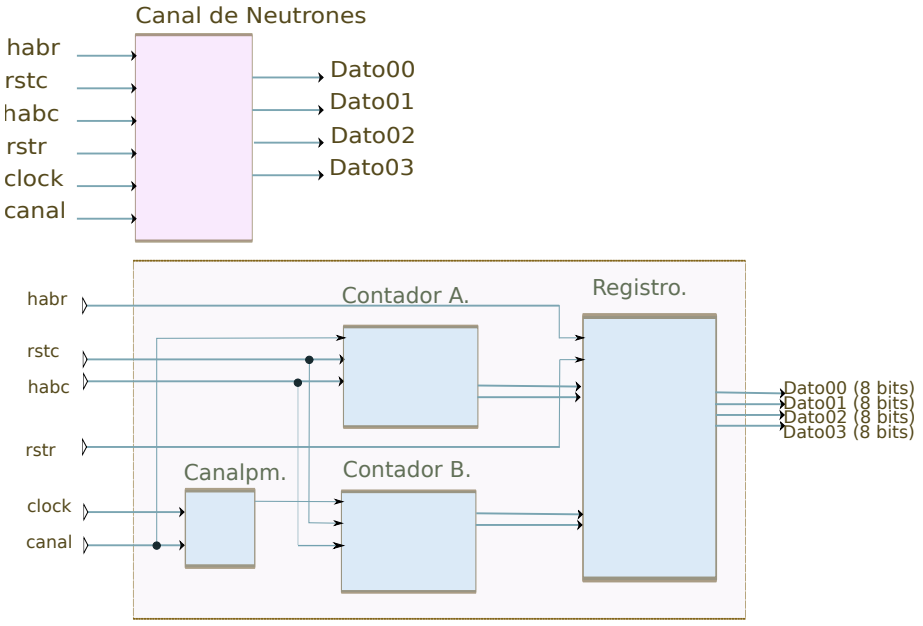

Figura 4.9: *Diagrama del módulo canal de neutrones.*

pulsos que superan el umbral son debidos a la multiplicidad del detector. Como resultado de este procedimiento se obtienen *cuentas múltiples*.

La señal del contador proporcional no llega directamente al contador B. Previamente se requiere de un módulo que discrimine entre los anchos de pulsos. Este módulo lleva por nombre *canalpm*.

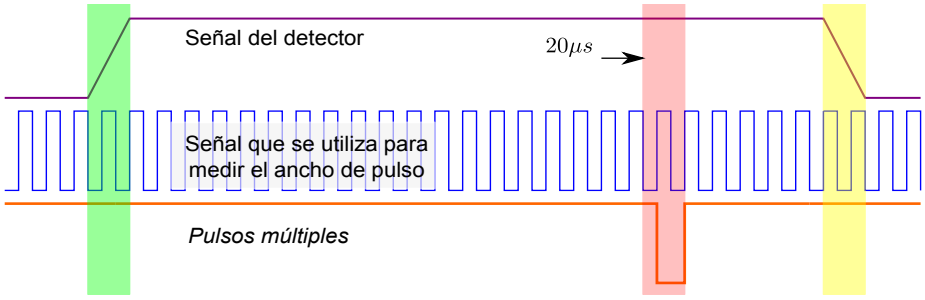

Figura 4.10: *Principio de funcionamiento del módulo canalpm.*

El *canalpm* se compone de un contador y un comparador. El contador se habilita cada vez que se sensa el flanco de subida de un pulso y finaliza cuando detecta su flanco de bajada (Fig. 4.10 secciones verde y amarilla). El contador trabaja a una frecuencia de $1MHz$ con objeto de medir la duración del pulso.

A la salida del contador se tiene un comparador; cuando el ancho de pulso de la señal supera la referencia se genera un pulso (Fig. 4.10 sección roja). Este pulso es el que entra al contador B para ser contabilizado.

En general, el módulo recurre a cuatro señales de control: *habr*, *habc*, *rstr* y*rstc*. La señal *habc* habilita o deshabilita a los contadores; mientras que *habr* activa el registro para copiar los datos acumulados en los contadores. La copia se realiza mientras los contadores están detenidos.

La señal *rstr* y *rstc* sirven para reestablecer el registro y los contadores, respectivamente. En el caso de *rstr*, sólo se activa al inicio de la adquisición. La señal *rstc* se activa al inicio de la adquisición y cada vez que finaliza un *tiempo de adquisición*.

4.5. Módulo de presión

El barómetro digital Meteolabor AG GB1 es un instrumento de precisión cuyo principio de funcionamiento se basa en la deformación de una cápsula aneroide —celda métalica de parades delgadas, sellada al alto vacío. Fabricada de un material especial, dicha cápsula forma parte de un oscilador LC. Los cambios en la presión atmosférica, producen deformación en la cápsula, lo que a su vez mofica su inductancia y la frecuencia de operación del circuito. Los cambios de frecuencia son registrados por un MCU, el cual asocia estos cambios con el valor de presión correspondiente.

El equipo cuenta con dos líneas de salida para transmitir los datos a otras unidades: *DCLK* y*DATA*. La línea *DATA* se encarga de transmitir los datos de presión atmosférica en forma serie, codificados en BCD —binary code decimal— a 4 digitos. La señal de reloj *DCLK* tiene una frecuencia aproximada de $3kHz$. Es enviada por el equipo para sincronizar el envio y recepción de datos (Fig. 4.11).

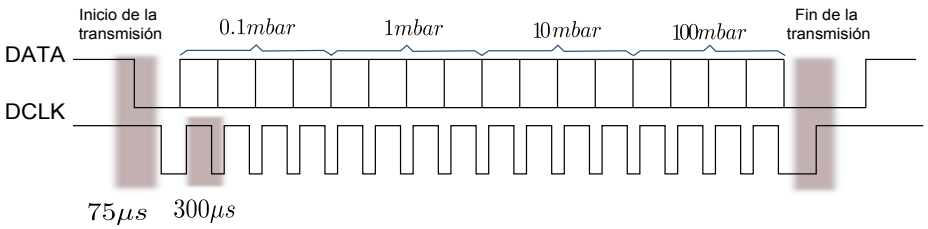

Figura 4.11: *Señales correspondientes al sensor de presión.*

Cuando el sensor no se encuentra transmitiendo datos, las líneas *DCLK* y*DATA* se encuentran en un nivel alto. Cada dos o tres segundos, el barómetro envia un dato de presión nuevo. En el observatorio de rayos cósmicos de CU sólo se recoge un dato de presión por cada periodo de adquisición.

Para el procesamiento de los datos de presión se cuenta con el *módulo de Presión*. Se compone de tres circuitos básicos; registro, contador y máquina de estados (Fig 4.12).

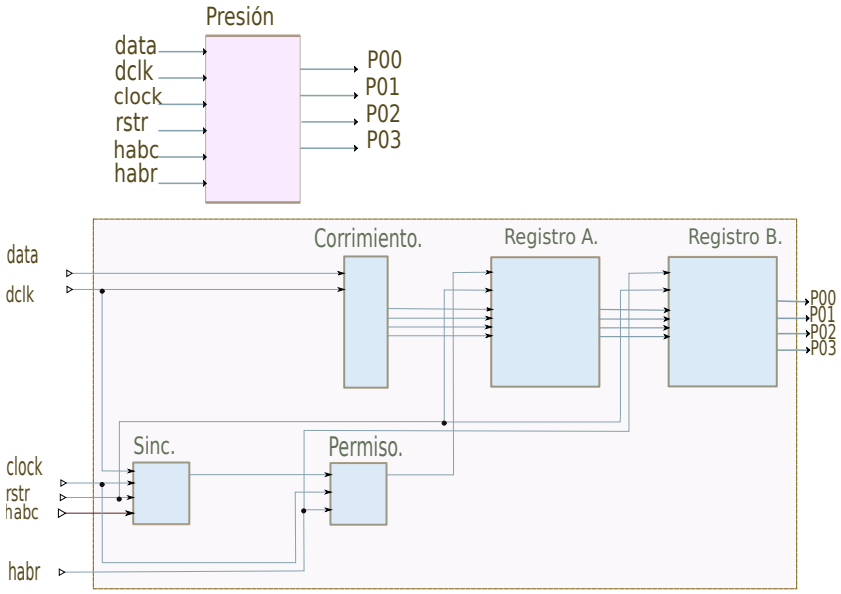

Figura 4.12: *Esquema del Módulo de presión.*

El primer paso es convertir los datos del sensor —*DATA*— transmitidos de forma serie a paralelo mediante un registro de corrimiento. La operación se hace con el fin de poder mantener los datos fijos durante un lapso de tiempo, mientras se copian a la memoria temporal. La memoria se compone de dos registros; registro X y registro Y.

El registro X sólo almacena datos del sensor de presión, cuando éste finaliza la transmisión. El registro Y copia datos del registro X, sólo cuando el sistema de control se lo indica. En conjunto, ambos registros se aseguran que se transmita el último dato de presión completo que llegó al sistema de adquisición.

Para que los registros cumplan su función se necesitan dos máquinas de estado que los controlen. Las máquinas de estado llevan por nombre; *sinc* y *permiso*, respectivamente.

La primera máquina se encarga de generar una señal —*sync*— que nos indica cuando el sensor se encuentra transmitiendo. El valor de esta señal es un nivel alto sí el sensor no transmite datos y un nivel bajo en caso contrario.

Considerando que el conjunto de datos que envia el sensor tiene una duración fija de aproximadamente $5ms$, podemos generar la señal *sync*, midiendo este

intervalo de tiempo y detectando el inicio de la transmisión por parte del sensor. Si a través de la maquina de estados sensamos la señal de reloj *DCLK*, se puede detectar cuando ésta sufre el primer cambio de nivel alto a nivel bajo. En ese instante, se debe habilitar un contador y permitirle alcanzar la cuenta necesaria para medir el intervalo de tiempo —24000 cuentas trabajando a una frecuencia de $5MHz$. Una vez que el contador llega al valor establecido, la máquina de estados detiene el contador, lo reestablece y vuelve a sensar la señal *DCLK* para repetir el proceso.

El siguiente algoritmo (Fig. 4.13) describe el funcionamiento de la máquina de estados *sinc*:

1. Se establecen condiciones iniciales —contador detenido y *sync* en nivel bajo. Sólo se puede regresar a este punto por medio de la instrucción *Reestablecer*.
2. Esperar a que se indique el inicio de la adquisición de datos.
3. Se evalua sí el sensor de presión comenzó la transmisión de datos. En caso contrario se espera hasta que inicie una nueva transmisión.
4. Esperar a que se termine la transmisión de datos —se activa contador.
5. Se evalua si el sensor comenzó una nueva transmisión de datos —contador se reestablece y *sync* en nivel alto. Si esto sucede se regresa al punto anterior.

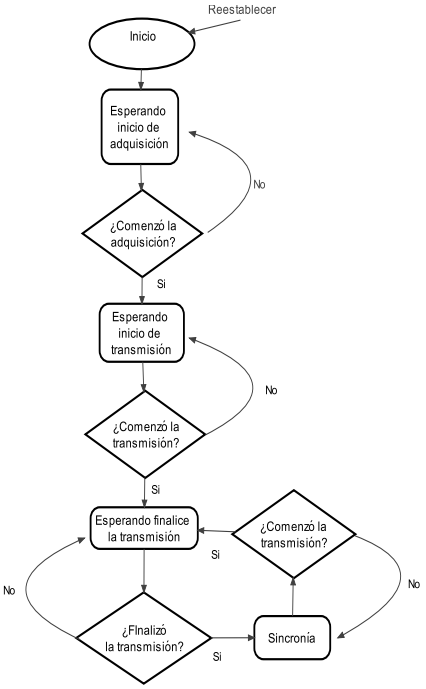

Figura 4.13: *Diagrama de flujo de la máquina de estados sinc.*

Una vez generada la señal *sync*, ésta entra a la siguiente máquina de estados —*permiso*. El algoritmo que describe el funcionamiento de esta máquina de estados es:

1. Evaluar si *sync* está en nivel alto.
2. Asegurar que no se esten pidiendo datos para transmitir.
3. Habilitar el registro A y regresar al primer punto.

Permiso se encarga de habilitar el registro A únicamente cuando se obtenga un dato completo. Además, esta máquina de estados se asegura que los registros

A y B no se habiliten al mismo tiempo, pues esto ocasionaría que se transmita un dato incorrecto. Al fenómeno detrás de este problema se le denomina *metaestabilidad*, se profundizará en las repercusiones que tuvo en nuestro diseño en el siguiente capítulo.

4.6. Módulo de amplificación, discriminación y cambio de nivel

Con el fin de poder conectar el detector con el PLD, se necesita una etapa de acoplamiento. Debido a que la señal proveniente del detector tiene una amplitud de $130mV$, se requiere una etapa de amplificación previa para poder alcanzar los niveles que utiliza el PLD. Los PLD's al ser de tecnología CMOS, pueden trabajar con señales de niveles de voltaje de $3{,}3V$ y $5V$, entre otros.

Para el desarrollo de la etapa de amplificación, consideraremos amplificar la señal del detector a una amplitud de $5V$. Utilizaremos un amplificador operacional *MC33178* en configuración inversora. Se usaron dos etapas de esta configuración para una ganacia total de 39 y reducir los efectos por distorsión.

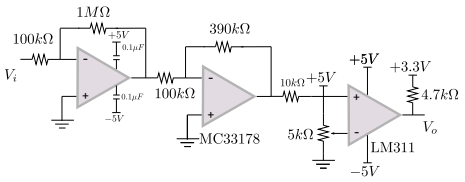

Figura 4.14: *Diagrama eléctrico de la etapa de amplificación y discriminación para el detector.*

A la salida de la etapa de amplificación se colocó un comparador *LM311*. La señal de salida de la etapa amplificadora entra al comparador por la terminal positiva. En la terminal negativa se conecta un voltaje de referencia que puede ser variado con un potenciometro. El objetivo del comparador es funcionar como discriminador y eliminar todas las interferencias que se encuentran en la señal amplificada.

Por último, el comparador cuenta con una salida de colector abierto que nos permite cambiar de nivel de tensión a $3{,}3V$ (Fig. 4.14). La explicación del cambio de nivel está relacionada con el dispositivo PLD que se seleccionó. En el siguiente capítulo aclararemos este punto.

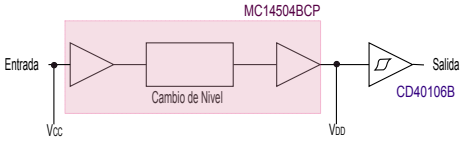

Figura 4.15: *Diagrama de la etapa de acoplamiento para el canal de presión.*

Para completar el diseño de la etapa de acoplamiento, es necesario acondicionar las señales del sensor de presión. Estas señales manejan niveles lógicos TTL y son transmitidas por un largo cable que las distorsiona. Por estas razones utilizamos un circuito cambiador de nivel y un buffer inversor *schmitt trigger* (Fig. 4.15).

El circuito cambiador de nivel *MC14504BCP* se utiliza para cambiar de niveles lógicos TTL a niveles CMOS —3,3V para nuestro caso.

El *schmitt trigger* nos sirve para reconstruir la señal y eliminar el ruido de ésta.

4.7. Módulo software de adquisición

Finalmente, hablaremos del módulo *software de adquisición de datos*. Las unidades que componen a dicho módulo son:

- Interfaz de comunicación *USB-*I^2C.
- Descripción del maestro en la PC.
- Herramientas para el depliegue de la información.
- Módulo de corrección de tiempo.

Como principal objetivo de nuestro módulo se encuentra lograr la comunicación entre la PC y el PLD por medio del protocolo I^2C. No obstante, una PC no cuenta con puerto de comunicación I^2C. Por esta razón se necesita una interfaz de comunicación. De preferencia, esta interfaz de comunicación debe contar con la posibilidad de conexión via *USB* —Universal serie Bus. Actualmente *USB* es el puerto de comunicaciones estándar y se prefiere su uso al de otro puerto de computadora para garantizar compatibilidad en nuevos equipos de cómputo que solo tienen puertos USB.

Resultado de la investigación y prediseño de la interfaz USB- I2C, en la que se pretendía utilizar un dispositivo del fabricante FTDI-chip, se encontró una interfaz comercial con las mismas características necesarias establecidas en el diseño de la interfaz, por lo que se prefirió adquirir la interfaz comercial por su relativo bajo costo y tamaño reducido. El módulo *USB-*I^2C Devantech nos proporciona una interfaz I^2C completa para la PC a un bajo costo.

Una de las ventajas de trabajar con dicho módulo, radica en el uso del chip *FTDI-FT232R*, componente principal de la interfaz. Su uso, en conjunto con el controlador provisto por el fabricante, permite manejar el puerto USB mediante instrucciones propias de un puerto *RS232*. Esto facilita enormemente el manejo de la comunicación por parte del software, ya que se puede trabajar con un puerto serie virtual, disfrutando de la conexión práctica del puerto USB y al mismo tiempo, contar con la interfaz I^2C necesaria.

El módulo *USB-*I^2C está diseñado para trabajar sólo en modo maestro; será el encargado de generar las condiones de *START* y *STOP* en la transmisión de datos.

De esta manera, nuestro esquema de comunicación queda de la siguiente forma: en primer lugar, la PC debe comunicarse con el módulo *USB-I^2C*, por medio de un puerto serie virtual; al recibir las instrucciones de la computadora, el módulo *USB-I^2C* administrará las tareas en el bus I^2C.

El programa en la computadora tiene la tarea de establecer la comunicación serie asincrona entre la PC y la interfaz. Además debe proporcionar los comandos necesarios para que la interfaz gestione correctamente el bus.

Los primeros pasos para establecer la comunicación serie son los siguientes:

1. Tener el controlador de la interfaz.
2. Contar con un lenguaje de programación que nos ofrezca bibliotecas que faciliten el manejo de comunicaciones serie asincronas.

La plataforma sobre la que se desarrolla el programa es *Software libre*. El sistema operativo con el que se trabaja es *Ubuntu 10.04*.

Trabajar con este sistema operativo facilita el manejo de la interfaz, ya que dentro de los módulos del Kernel se encuentran los controladores para diversos dispositivos I^2C. Entre ellos nuestra interfaz.

Del segundo punto tendremos que decir que una de las ventajas de trabajar con un sistema operativo libre reside en la libertad de manejar los puertos de comunicación sin ninguna restricción, por lo que la elección del lenguaje de programación se basó principalmente en la facilidad de la estructura del lenguaje.

El lenguaje de programación elegido es *Python*.

Python es un lenguaje de programación de alto nivel que tiene una sintaxis sencilla y legible, permite el desarrollo de programás compactos y se puede ajustar a diferentes paradigmas de programación. Entre estos se encuentra: programación orientada a objetos, programación imperativa y programación funcional.

El interprete de Python se encuentra bajo una licencia que lo hace de uso libre y distribuible incluso en aplicaciones comerciales.

Las razones por las que se prefirió programar en Python son:

Sintaxis legible. El software que se desarrolla en Python tiene una sintaxis clara, de tal manera que cualquier programador puede comprender su funcionamiento y reutilizar el código.

Código compacto. Python ayuda a reducir el tiempo de desarrollo, ya que puede ejecutar varias instrucciones con poco código.

Portabilidad. Los programás en Python pueden ejecutarse en diversas plataformás; tanto en Linux como en Windows.

Baterías incluidas. Se refiere a la colección de bibliotecas que vienen incluidas en el lenguaje. Tales bibliotecas nos permiten realizar variedad de funciones incluso en aplicaciones científicas.

Dentro del software que viene incluido en Ubuntu 10.04 se encuentra el interprete de Python y las bibliotecas necesarias para el acceso al puerto serie. La biblioteca *pyserial* encapsula el acceso al puerto serie y proporciona métodos a Python para manejar el puerto serie en Windows y Linux.

Una vez que se cuenta con las herramientas para establecer la comunicación serie, el primer paso en nuestro programa es abrir el puerto serie y configurarlo de acuerdo a las especificaciones que nos brinda el fabricante de la interfaz; velocidad de transmisión, número de bits de *stop*, bit de paridad y tamaño de palabra.

La información que requiere la interfaz *USB-*I^2C para administrar el bus I^2C se compone de una cadena de 3 *bytes*. El programa envia a través del puerto serie, 3 bytes con la información siguiente:

Comando de la interfaz *USB-*I^2C —$0H53$ o $0H54$. Por medio de este comando se le informa a la interfaz si se va a leer-escribir un solo byte o un conjunto de ellos.

Dirección del esclavo +R/W. Los primeros siete bits de la instrucción contienen la dirección del esclavo —$0H16$. El bit menos significativo informa a la interfaz si la operación que se va a realizar es de escritura o lectura.

Bytes de dato Si la operación es de escritura, este byte contiene la información que va a enviar el maestro al esclavo. Sí la operación es de lectura, el byte indica cuantos datos va a esperar el maestro del esclavo.

El software de adquisición puede enviar tres instrucciones; para la instrucción *reestablecer* el byte $0H48$, para *iniciar adquisición* el byte $0H49$ y para *copiar dato* el byte $0H4A$.

El dato recibido se alamacena junto con la referencia de la fecha y hora en la que fue recogido. Estos datos se guardan en un archivo que lleva por nombre la fecha del día. Al finalizar el día el software de adquisición debe cerrar el archivo y abrir uno nuevo con la fecha correspondiente.

EL algoritmo que describe el módulo software de adquisición se enlista a continuación:

1. Abrir y configurar el puerto serie.
2. Preguntar la fecha y abrir un archivo donde se almacenen los datos.

3. Enviar instrucción *reestablecer*.
4. Enviar instrucción *iniciar adquisición*.
5. Esperar a que se cumpla tiempo de adquisición.
6. Enviar instrucción *copiar datos*.
7. Solicitar datos al esclavo.
8. Registrar la hora.
9. Almacenar datos con fecha y hora.
10. Sí finalizó el día abrir un nuevo archivo. Regresar al paso 5.

Sí el esclavo no responde a las intrucciones enviadas por el maestro, se vuelve a intentar reestablecer la comunicación con él. Si no se logra, el programa se cierra y le envia un mensaje de error al usuario que contiene la causa que provocó la interrupción de la adquisición. Para lograr este diagnóstico, el programa debe leer una byte que envia la intefaz cada vez que tiene respuesta por parte de un esclavo.

En caso de que se deseen hacer pruebas con el sistema, se cuenta con otro programa que facilita la fijación del tiempo de adquisición y espera que el usuario dé las instrucciones para comunicarse con el esclavo.

La información se despliega impresa en pantalla en todo instante. Además se puede solicitar al programa graficar los datos almacenados contra tiempo. Para generar las gráficas, el programa utiliza la biblioteca *matplotlib* de Python.

El programa ejecuta las siguientes instrucciones:

1. Preguntar al usuario que archivo desea graficar.
2. Abrir el archivo.
3. Preguntar al usuarios los datos que quiere se muestren.
4. Procesar la información del archivo para obtener la información de fecha, hora y datos.
5. Mostrar la gráfica.

Por último, para llevar el tiempo de adquisición el programa hace uso del reloj interno de la computadora. El ajuste de este reloj se realiza mediante *ntp*. Éste es un protocolo que permite la sincronización de relojes por medio de internet. Una vez a la semana *ntp* se encarga de ejecutar el reloj de la computadora conectándose al servidor: ntp.astrosmo.unam.mx.

4.8. Interconexión del sistema

En esta sección se mostrarán los detalles finales que se consideraron para la interconexión del sistema.

El primer punto concierne al envio de los datos. Para poder conectar el esclavo al bus, es necesario que éste tenga una salida de *open drain*. Utilizamos un buffer con salida *open drain* para hacer esto.

Para enviar los datos a través de la línea SDA es imprescindible convertirlos de forma paralela a serie. Como se mencionó en secciones anteriores esto se logra mediante un sistema multiplexor-contador. Es importante precisar: la salida del multiplexor debe conectarse al buffer, el buffer debe conectarse a la línea SDA y el multiplexor debe controlarse de tal forma que sólo opere durante el envio. Este control lo ejerce el *módulo de comunicación* I^2C, el cual detecta la solicitud del maestro de *leer datos* del esclavo y genera una señal w que habilita al multiplexor. Al mismo tiempo se activa un contador —*contE*— que opera con la señal SCL.

La salida de *contE* se conecta a las líneas de selección del multiplexor —*numdato*. De este modo se logran enviar al maestro los datos que se encuentran a la entrada del multiplexor. En la siguiente figura se muestra la disposición de los módulos para el envio de datos.

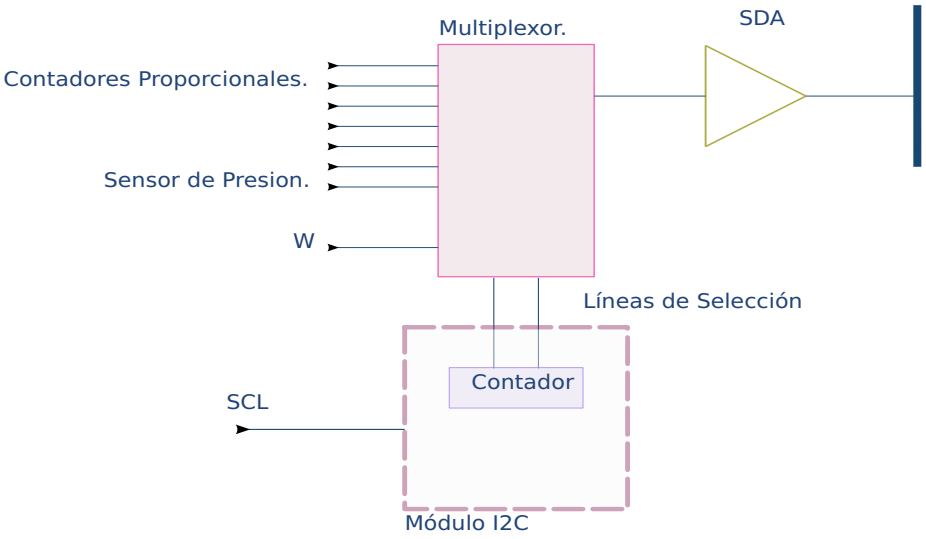

Figura 4.16: *Conexión del sistema de adquisición para el envio de datos.*

Cuando el *módulo de comunicación* I^2C registra, mediante la salida del contador, que se alcanzó el barrido total de los datos, deshabilita el multiplexor y reestablece el contador.

La parte final de la interconexión del sistema corresponde la sincronización de sus elementos. La sincronización es el proceso por el cual se establece el orden en que se ejecutan las tareas del sistema. Para establecer este orden las máquinas de estado utilizan una señal de reloj que les permite trabajar de forma secuenciada, sin que un proceso interfiera con otro.

El módulo que genera las señales de reloj opera con una señal de $40MHz$, que se divide para obtener señales de $5MHz$, $1MHz$ y $0{,}5MHz$ —Fig. 4.17. A continuación se enlista la función de cada una de estas señales:

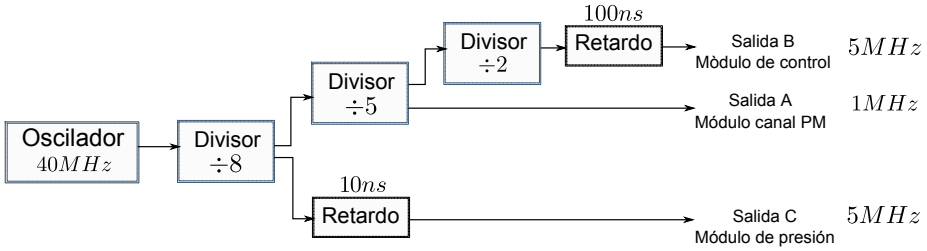

Figura 4.17: *Circuito divisor de frecuencia.*

- La señal de $1MHz$ sirve para medir el ancho de pulso de las señales provenientes del detector.
- La señal de $5MHz$ se ocupa en el módulo *sinc* para sincronizar la recepción de los datos del sensor de presión. Con esta señal trabajan las máquinas de estado y se mide el intervalo de $5ms$ —Sección 4.5.
- El módulo de *control* requiere una señal de reloj de $0{,}5MHz$ para que los demás módulos puedan detectar las señales que genera, es decir, tengan la duración necesaria.

Finalmente, las señales de reloj se adaptan para el resto del sistema de adquisición por medio de circuitos de retardo. La utilidad de dichos circuitos se explicará en el siguiente capítulo.

En resumen, el diseño del nuevo adquisidor de datos comprende; la descripción en VHDL de todos los módulos expuestos, su interconexión y posterior síntesis en el PDL; el programa de comunicación con la PC e interfaz del usuario, escrito en lenguaje Python, junto con las herramientas de diagnóstico y despliegue de información.

El nuevo sistema —Fig. 4.18— cubre las necesidades del Observatorio de Rayos Cósmicos de Ciudad Universitaria y aporta nuevas funciones que facilitan su operación y brindan nueva información sobre el monitor NM64. Debido a las herramientas con las que se elaboró el diseño, el nuevo adquisidor de datos puede reproducirse e incluso modificarse, si así se desea. Además se cuenta con

la documentación y diagramas eléctricos necesarios para dar mantenimiento y así garantizar su funcionamiento por varios años.

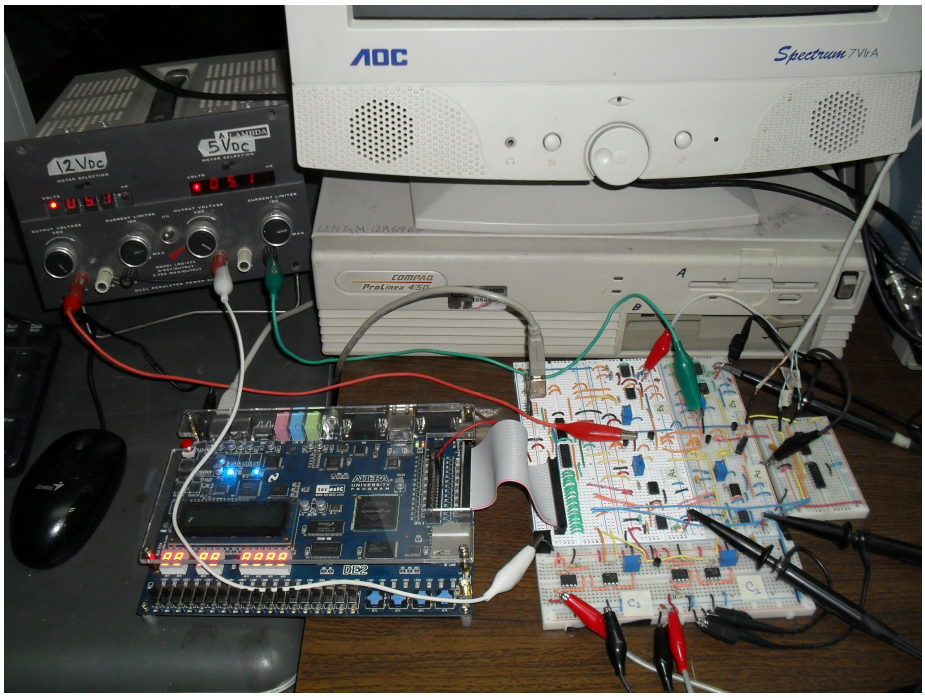

Figura 4.18: *Fotografía del nuevo sistema de adquisición de datos.*