

Universidad de Guadalajara

Centro Universitario de los Valles



Maestría en Ingeniería Mecatrónica

Nombre de la tesis:

**"Sistema inalámbrico para domótica en aulas de CU
Valles"**

Presentado por:

**Ing. José Alberto Beristain Cornelio
Ing. Roque Isaac Gómez Zambrano**

Director

Dr. Héctor Huerta Ávila

Ameca, Jalisco, Noviembre de 2019

Resumen

Debido al incremento en la tarifa de la energía eléctrica, se han desarrollado esquemas para ahorrar energía, de manera que la eficiencia se incremente y se utilice solo lo necesario.

El ahorro de energía es notorio en los lugares con alto consumo, donde pequeños cambios pueden derivar en disminuciones importantes en el pago de electricidad. De esta manera, con el objetivo de gestionar el uso de energía y control de diversos dispositivos se han desarrollado sistemas domóticos para casas y edificios.

La palabra domótica proviene del latín domus (hogar, casa, domicilio) y del griego autónomo (que se gobierna a si mismo). De acuerdo a la RAE la palabra se define de la siguiente manera: Conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda [1].

Se llama domótica a los sistemas y dispositivos que proporcionan algún nivel de automatización dentro de una casa, pudiendo ser un temporizador para encender/apagar una luz o aparato a una hora determinada, hasta sistemas más complejos capaces de interactuar con cualquier elemento eléctrico del hogar. La vivienda domótica es aquella que integra un conjunto de automatismos en materia de electricidad, electrónica, robótica, informática y telecomunicaciones con el objetivo de asegurar al usuario un aumento de confort, de seguridad, ahorro energético, facilidad de comunicación y posibilidad de entretenimiento [2]. También se ha utilizado el término Inmótica para referirse a la automatización de edificios terciarios o de servicios, que procede del vocablo immobilis (inmueble) y automática.

Hoy en día estos sistemas son tan complejos que se han considerado en tres niveles: El primer nivel corresponde a sistemas aislados, el segundo a interconexión de equipos y el tercero a aplicaciones y servicios, incluyendo seguridad [2]. La mayoría de las empresas que ofrecen estos servicios son extranjeras y las soluciones ofrecidas presentan costos elevados, sobre todo por los gastos de importación, además de que, en general, no se ajustan a las necesidades particulares. Este tipo de esquemas pueden ser difíciles de instalar, además de costosos, en instalaciones con edificios distribuidos, como en el caso del CUValles.

Índice

1.	Introducción.....	1
1.1	Planteamiento del problema	1
1.2	Estado del arte.....	1
1.3	Justificación	3
2.	Metodología.....	4
2.1	Operación del sistema.....	4
2.2	Etapas del proyecto.....	7
3.	Cronograma	8
4.	Componentes del sistema	9
4.1	Red inalámbrica	9
4.2	Configuración del Xbee	11
4.2.1	Modo de operación.....	12
4.2.2	UART	13
4.2.3	Modo transparente.....	13
4.2.4	Modo Inactivo (Iddle)	14
4.2.5	Modo transmisión.....	14
4.2.6	Modo recepción.....	14
4.2.7	Modo comando.....	15
4.2.8	XCTU	15
4.2.9	Configuración de nodo	15
4.3	PCB.....	20
4.3.1	Alimentación	21
4.3.2	Reguladores de tensión.....	21
4.3.3	Xbee.....	21
4.3.4	Leds indicadores de RX, TX e inactividad.....	22
4.3.5	CI 74240	22
4.3.6	CI CD4072	23
4.3.7	PIC18F4550.....	23
4.3.8	Pines para entradas y salidas	23
4.3.9	Push Bottons.....	24
4.3.10	Mounting Holes	25
4.3.11	Capas - Top	26

4.3.12	Capas - Boton	26
4.3.13	Capas – Serigrafia	27
4.3.14	Diseño completo.....	27
4.4	Control en aula.....	29
4.5	Programación.....	31
4.5.1	Programación de Microcontrolador.....	33
4.5.2	Librería EEMPROM_Libreria.h	36
4.5.3	Librería Serial.h.....	38
4.5.4	Librería config_reg.h.....	40
4.5.5	Programación de Software, estación central.	41
4.5.6	Ciclo de Sub-programas	43
4.5.7	Ciclo Monitor	44
4.5.8	Ciclo Ejecutor de Tareas por Edificio	51
4.5.9	Inicialización de variables	53
4.5.10	Sub Programa “Sub_vi_ejecutar_Edificio”.....	53
4.5.11	Ciclo Ejecutor de Tareas por Aula	60
4.5.12	Inicialización de variables	62
4.5.13	Sub Programa “Sub_vi_ejecutar_Aula”	63
4.5.14	<i>Programador de Tareas</i>	66
4.5.15	<i>Programación de Tareas por Aula</i>	66
4.5.16	<i>Programación de Tareas por Edificio</i>	68
4.5.17	Diagrama a bloques de programador de tarea	69
4.5.18	Programa “Servicios Manual”	75
4	Funcionamiento del Sistema	81
4.6	Vista general	81
4.7	Integración de sistemas.....	82
4.8	Pantallas del sistema	83
4.8.1	Pantalla principal de monitoreo.....	83
4.8.2	Pantalla de programación de tareas por Aula	84
4.8.3	Pantalla de programación de tareas por Edificio.....	85
5	Referencias.....	86

Índice de Figuras

Figura 1 Diagrama a bloques de estación central.....	5
Figura 2 Diagrama a bloques de estación remota	5
Figura 3 Etapas del proyecto	7
Figura 4 Diagrama de Conexión de nodos DigiMesh.....	9
Figura 5 Digi XBee-PRO 900HP	10
Figura 6 Distribución de los pines del xbee	12
Figura 7 Modo de operación del Xbee	12
Figura 8 Flujo de datos del sistema en un entorno con interfaz UART.....	13
Figura 9 Modo transparente	14
Figura 10 Placa de desarrollo Grove.....	16
Figura 11 Puerto COM asignado a la placa.....	16
Figura 12 Agregando el xbee a la herramienta de configuración XCTU	16
Figura 13 Dispositivo agregado a la herramienta.....	17
Figura 14 Parámetros del dispositivo	17
Figura 15 Botones de acciones en XCTU	19
Figura 16 Actualización de firmware de xbee	19
Figura 17 Esquemático	20
Figura 18 Vista 3D de la PCB.....	20
Figura 19 Alimentación.....	21
Figura 20 Reguladores de tensión y leds indicadores.....	21
Figura 21 Xbee	21
Figura 22 Leds TX, RX, RSSI.....	22
Figura 23 Led de inactividad.....	22
Figura 24 CI 74240	22
Figura 25 CI CD4072	23
Figura 26 CI PIC18F4550	23
Figura 27 Pines de Entradas y Salidas	24
Figura 28 Push Botton para reset	24
Figura 29 Push Bottons conectados al puerto D.....	24
Figura 30 Entradas adicionales en paralelo a los push bottons del puerto D	25
Figura 31 Mounting Holes.....	25
Figura 32 Pistas - Top	26
Figura 33 Pistas - Botton.....	26
Figura 34 Serigrafía – Top	27
Figura 35 Diseño completo	27
Figura 36 PCB Real.....	28
Figura 37 PCB Final.....	28
Figura 38 PIC18F4550.....	29
Figura 39 Sensor PIR HC-SR501	29
Figura 40 Kit de control de acceso	30

Figura 41 Pantalla de monitor	41
Figura 42 Pantalla de Ejecutor de Tareas	42
Figura 43 Ciclo Sub Programas	44
Figura 44 Ciclo Monitor.....	47
Figura 45 Case False "Leer datos de puerto y desgloza Edif-Aula"	48
Figura 46 panel frontal subVi_extr_serv.....	48
Figura 47 Diagrama de subVi_extr_serv	48
Figura 48 Inicialización de variables asociadas al ciclo monitor.....	49
Figura 49 Diagrama de Bloque de ciclo Ejecutor de Tareas por Edificio	51
Figura 50 Variables de inicializacion de ciclo Ejecutor de Tareas por Edificio	53
Figura 51 ícono de sub_vi_ejecutar_edificio.....	54
Figura 52 Panel frontal de Sub_vi_ejecutar_Edificio	54
Figura 53 Diagrama a bloques de "Sub_vi_ejecutar_Edificio"	55
Figura 54 Case FALSE de bit de rango.....	56
Figura 55 Case FALSE de bit de rango.....	57
Figura 56 Case FALSE de bit de rango.....	57
Figura 57 Case TRUE detección vacío.....	58
Figura 58 Case TRUE detección vacío.....	59
Figura 59 Ciclo ejecutor de tareas por Aula.....	60
Figura 60 Case false de índice de tabla.....	61
Figura 61 Variable de inicialización, led indicador de ejecución de tarea	62
Figura 62 Panel Frontal de "Sub_VI_ejecutar_Aula"	63
Figura 63 ícono de "Sub_VI_ejecutar_Aula"	63
Figura 64 Panel Frontal de "Sub_VI_ejecutar_Aula"	64
Figura 65 Case FALSE de bit de rango.....	65
Figura 66 Case FALSE de bit de rango.....	66
Figura 67 Programador de tareas por aula	66
Figura 68 Programador de tareas por Edificio	68
Figura 69 Diagrama a bloques de programador de tareas.....	69
Figura 70 Columnas Base de Datos.....	70
Figura 71 Case FALSE rango de Aula	71
Figura 72 Panel Frontal subVi_extr_dias.....	72
Figura 73 Diagrama a bloques subVi_extr_dias	72
Figura 74 Case FALSE Concatenación	72
Figura 75 Case FALSE agregar / quitar tareas.....	73
Figura 76 Case TRUE botón + / -.....	73
Figura 77 Diagrama de tab para Aula.....	74
Figura 78 Panel frontal Aula de programa "Servicios Manual"	75
Figura 79 Panel frontal Edificio de programa "Servicios Manual"	75
Figura 80 Diagrama a bloques de programa "Servicios Manual", tab Aula.....	77
Figura 81 Case TRUE sección 2.2	77
Figura 82 Case FALSE sección 2.....	78
Figura 83 Diagrama a bloques de programa "Servicios Manual", tab Edificio	79
Figura 84 Case FALSE tab Edificio.....	80

Figura 85 Detección de arreglo vacío	81
Figura 86 Case FALSE Contador de error de vacío	81
Figura 87 Sistemas del proyecto.....	81
Figura 88 Integración de sistemas.....	82
Figura 89 Pantalla principal de monitoreo	83
Figura 90 Pantalla de programación de tareas por aula.....	84
Figura 91 Pantalla de programación de tareas por edificio	85

Índice de Tablas

Tabla 1 Cronograma de actividades	8
Tabla 2 Especificaciones de XBee-PRO 900HP	10
Tabla 3 Descripción de pines de Xbee	11
Tabla 4 Especificaciones de PIC18F4550.....	29
Tabla 5 Bytes Estación Central --> Microcontrolador en Aula.....	31
Tabla 6 Bytes Microcontrolador --> Estación central.....	33
Tabla 7 Pines Microcontrolador	33
Tabla 8 Servicios	49

1. Introducción

1.1 Planteamiento del problema

El proyecto surge como una necesidad del centro universitario de gestionar energía y controlar accesos al plantel y aulas, así como controlar servicios existentes en las aulas.

Este proyecto forma parte de la etapa 1 de 6 de un proyecto integral en el centro. El objetivo del proyecto es gestionar la energía eléctrica que activan/desactivan los diversos servicios en un aula, en éste caso aire acondicionado, proyector, despachador de agua y cerradura eléctrica para control de acceso, de manera remota.

El problema se debe a que parte o todos los servicios quedan funcionando en aulas aun cuando éstas no están siendo utilizadas o no son necesarios, los servicios no son necesarios antes de las 8:00 am y después de las 18:00 horas; es común encontrar el aire acondicionado y luces encendidos después de terminar clases o en días festivos, debido a que alumnos / profesores olvidan apagar éstos servicios.

El presente proyecto plantea trabajar con un par de aulas, sin embargo, el proyecto será escalable y flexible; es decir se podrán incorporar más sensores y módulos en aulas necesarios para monitorear el centro por completo.

1.2 Estado del arte

En la actualidad los escenarios donde los seres humanos interactúan son del tipo inteligente, es decir que están dotados de un gran número de funciones para su bienestar y/o confort. Actualmente estas edificaciones están dotadas de un sistema nervioso donde la palabra inteligente se ha empezado a posar sobre todo lo que la constituye, desde los electrodomésticos más conocidos como los televisores, refrigeradores, hornos microondas, hasta la arquitectura misma, todo esto con el único fin de integrar tecnología a las edificaciones tradicionales para convertirlo en algo inteligente donde el hombre puede tele-gestionar sus actividades ya sea personales o de trabajo [3].

La nueva área de la ingeniería conocida como domótica o como ciencia de la innovación, está creciendo a niveles exponenciales, y las principales empresas diseñadoras de electrodomésticos están dotándolos de inteligencia para aumentar las funciones que

tiene cada uno de ellos y cada vez más suplir las necesidades que se presentan en el hogar o trabajo [4].

Por otra parte, cabe señalar que, a pesar de un desarrollo tecnológico acelerado, la comunicación por redes y automatización de procesos a nivel latinoamericano es incipiente, sin embargo, las redes están interconectadas y han ido creciendo con la aparición de internet, y por tanto este desarrollo ha permitido el crecimiento de los procesos de domotización [2].

La domótica se ha ido desarrollando en el mundo, existiendo países como Estados Unidos, Japón y Alemania, con mayores adelantos e implementaciones, tanto en el mercado de consumo como industrial. En referencia específica a sistemas domóticos vigentes, podemos encontrar el caso destacable de KNX, un sistema de instalación domótico e inmótico. Este tiene la particularidad de ser un estándar abierto, y maneja diferentes tipos de medios de transmisión como TP1 (Par trenzado), PL1110 (Powerline), RF y Ethernet (TCP/IP). Entre las ventajas que podemos encontrar en este sistema es la arquitectura distribuida que maneja, es decir, no se requiere un controlador central para manejar la instalación, cada elemento dispone de su propia inteligencia y se comunican entre ellos, lo que permite, además, una rápida modificación de la instalación, pero un gran punto en contra es que para utilizar este sistema es necesario configurar gateways adecuados para cada tipo de dispositivo [5].

Por otro lado, actualmente podemos encontrar diversos sistemas de automatización basados en arduino, así como diversos elementos eléctricos, electromecánicos para el control de diversos servicios en el hogar, incluso hay diversas personas que ofrecen un kit con diferentes elementos para automatizar en cierto grado el hogar, éstos los podemos encontrar en páginas populares de ventas, como lo es ebay.com, mercadolibre.com.mx, etc [8].

También podemos encontrar varias compañías que proveen elementos de control para luces y contactos, como lo es SONOFF [6] y BROADLINK [7], provee elementos de control, en cierto grado, inalámbrico por medio de wifi, con temporizadores, entre otras cosas, la desventaja es que los dispositivos tienen un número limitado de contactos y se tiene que adquirir todo un equipo de la misma marca para realizar el control, además de que el grado de personalización es limitado.

De igual manera se pudo identificar proyectos relacionados con objetivos similares al presente, como por ejemplo "Desing and development of an automatic small-sccale house for teaching domotics" en el cual en el año 2001 se desarrolló una casa a escala en el que

se controlan diversos servicios de la misma, tales como luces, conectores, calefacción y alarmas todo esto mediante un PLC Simatic S7-200 así como los software Simatic 2.1 y Visir 1.6. Lo destacable de este proyecto es que incluye elementos que pudieran ser automatizados en un escenario real, sin embargo, al estar orientado a la enseñanza y en su mayoría son simulaciones, carece de condiciones que pudieran afectar el funcionamiento del escenario óptimo planteado hasta ahora [9].

Otro proyecto relacionado es el llamado IoT based Power Efficient System Design using Automation for Classrooms, en el cual se utiliza una tarjeta de desarrollo Intel Galileo 2da generación con el propósito de ahorrar energía eléctrica en aulas, los puntos destacados de este proyecto es el hecho de que se controla todo automáticamente mediante un web server remoto aprovechando el internet de las cosas [10].

Por otra parte también se identificó el trabajo "Design and Implementation of Lighting Control System for Smart Rooms" que utiliza el microprocesador ARM combinado con una variedad de module así como una interfaz CAN bus para realizar el control inteligente de las luces. Cabe resaltar que el trabajo está enfocado en controlar luces de cuartos de hoteles sin embargo el objetivo de este trabajo es administrar los niveles de intensidad de las luces para que cumplan con ciertas reglas establecidas que a su vez incrementaran el confort de los ocupantes de los cuartos y disminuirán el uso de energía [11].

1.3 Justificación

Uno de los principales problemas que existe en el Centro Universitario de los Valles es el uso desmedido de energía eléctrica, al estar activos los servicios del aula, que son la iluminación, aire acondicionado, proyector y el despachador de agua, cuando estos no son necesarios. Es común que el aire acondicionado y las luces de un aula permanezcan encendidos después de acabar una sesión de clases, debido a que alumnos y profesores se retiran olvidan de apagar los mismos y no se tienen un mecanismo para apagarlos de manera automática. De igual forma, se tienen servicios activos que no son necesarios antes y/o después de las jornadas de clases, los fines de semana o días festivos; incluso se pretende que las aulas en las que no esté programado alguna actividad o clase no se pueda entrar.

Todos los elementos mencionados estarán interconectados mediante un sistema de monitoreo y control central, utilizando un servidor. Desde una aplicación en el servidor se podrá activar o desactivar los servicios arriba mencionados ya sea de manera manual o programada.

Objetivo general

Diseñar, desarrollar e implementar un sistema de gestión de energía y control de acceso para las aulas de CU Valles que controle el encendido y apagado de los servicios, cuando éstos no se requieran o estén fuera del horario de servicio programado, mediante una aplicación en un servidor.

Objetivo particular

1. Crear una red inalámbrica de comunicación entre servidor (estación central) y aulas (estaciones remotas).
2. Desarrollar una aplicación para controlar, monitorear y programar horarios de servicios en aulas, así como controlar el acceso a las mismas.
3. Desarrollar un sistema embebido para aulas (estación remota) que controle diversos servicios y sensores tanto de corriente alterna como continua.
4. Instalar sensores de movimiento en aulas, de tal manera que cuando no se detecte movimiento alguno, los servicios de cada aula se apaguen de manera automática.
5. Desarrollar un software para la estación central (PC o servidor), que se comunique de manera serial con un transceptor y éste a su vez de forma inalámbrica con las estaciones remotas.

2. Metodología

2.1 Operación del sistema

El proyecto consiste en 2 sistemas de comunicación inalámbrica, al que llamamos estación central consta de un software y un transceptor; el otro que estará colocado en cada una de las aulas, que llamamos estación remota, consta de un transceptor y un microcontrolador que controlará los diversos servicios.

En la figura 1 y 2, se muestra un diagrama a bloques de la estación central y la estación remota respectivamente:



Figura 1 Diagrama a bloques de estación central

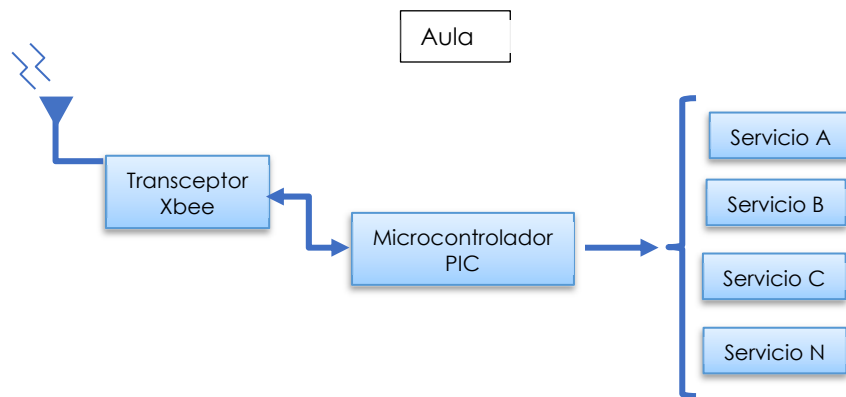


Figura 2 Diagrama a bloques de estación remota

Estación central

La figura 1 describe la estación central, como se comentó anteriormente, además de contar con un transceptor, dispondrá de una interfaz gráfica diseñada en labview, desde la cual se podrá controlar los diversos servicios de manera individual o grupal. Desde la interfaz gráfica además se podrá monitorear el estado de los diversos servicios en las aulas, así como activar/desactivar de manera manual, y programar la activación/desactivación de manera automática de los diversos servicios.

La interfaz se comunicará de manera serial a un transceptor, y de manera inalámbrica a todos los dispositivos de cada una de las aulas; así mismo se encargará de recibir del transceptor los datos provenientes de las aulas y los enviará a la interfaz gráfica para su monitoreo.

La comunicación entre todos los dispositivos se realizará por medio de una cadena de caracteres en los que se describe la dirección del dispositivo receptor (Edificio y Aula) y la

activación/desactivación de cada servicio, por ejemplo: se envía un conjunto de caracteres, de 8 bits cada uno, uno que describe el edificio, otro el número de aula, otro el estado de cada servicio y un último carácter para indicar el fin de la cadena.

En la estación remota, la comunicación entre el transceptor y el PIC será de manera serial por medio del protocolo de comunicación EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter); entre la PC y el transceptor por medio de USB.

Sistema en aulas

Cada estación remota, descrita en la figura 2, contará con una etiqueta/variable identificadora (Edificio y aula) dentro del programa del microcontrolador, para que desde la estación base se controle cada servicio de cada aula; cuando la estación base envía un mensaje, todos los dispositivos de cada aula lo reciben, por lo que únicamente ejecutará las acciones aquella aula que coincida con los caracteres de "aula" y "edificio", la variable aula será asociada de manera única a cada estación remota

El transceptor se encarga de recibir datos de la estación base, se comunica con el microcontrolador y éste interpreta el mensaje y ejecuta las órdenes. Es importante señalar que los servicios de manera local contarán con su respectivo switch para encender/apagar de manera manual cada servicio. Por otro lado, el sensor de movimiento nos ayudará a cortar la energía de los servicios cuando no se detecte movimiento en el aula después de un periodo de tiempo, por ejemplo, después de 15 minutos; cuando se detecta movimiento los servicios que estaban activos se reestablecerán.

2.2 Etapas del proyecto

Las etapas del proyecto son 5 las cuales se describen a continuación:

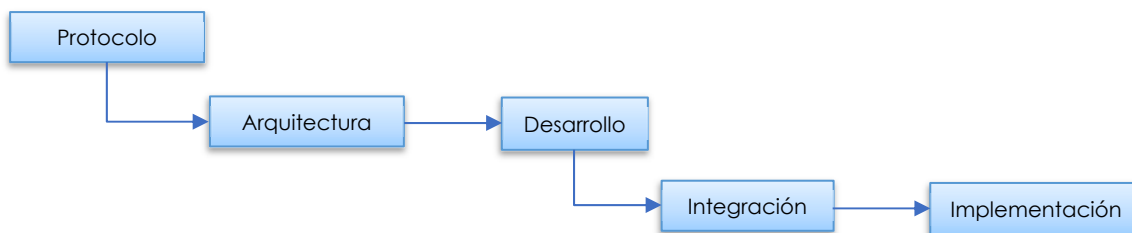


Figura 3 Etapas del proyecto

1. *Protocolo*. En ésta primera etapa se definen y se limitan los alcances del proyecto de manera particular, se pasa por un proceso de revisión con el director.
2. *Arquitectura del sistema*. Se realizarán los primeros bocetos, de manera conceptual el diseño del hardware y software, así como la radiocomunicación, buscando solución a los posibles problemas que pudieran presentarse, tanto en comunicación como en programación.
3. *Desarrollo del sistema*. Esta etapa se dividirá en sub etapas, en las que se desarrollará por separado cada una de las tareas del sistema embebido:
 - 3.1. *Estación central*. Se desarrollará un preliminar de la aplicación en labview y se establecerá una comunicación serial bidireccional con el transceptor, y se realizarán pruebas de su funcionamiento; se harán pruebas de activación/desactivación de manera manual y programada, así como pruebas de monitoreo.
 - 3.2. *Estación remota*. Se desarrollará y se probará la electrónica necesaria para activar/desactivar los servicios, los cuales utilizan voltaje de 120 / 220 volts. Se establecerá una comunicación serial con el transceptor y se realizarán pruebas.
 - 3.3. *Comunicación inalámbrica*. Consiste en realizar pruebas con el transceptor, desde el software enviar señales para la activación / desactivación al dispositivo embebido en aulas; también se harán pruebas de comunicación bidireccional, es decir enviar señales de solicitud de estado de servicios actuales en el aula y enviar datos hasta el software para su monitoreo; aquí mismo se realizarán pruebas de distancia e interferencia.

4. *Integración de sistemas.* Se terminará con el diseño del software y programación de hardware, el cual contendrá el control para 16 aulas con opción a integrarse más en un futuro; la estación central será capaz de comunicarse con cada una de las aulas por separado, se podrá visualizar el estado de los servicios de cada una de las aulas en el software. En ésta etapa se procederá a fabricar las PCB's necesarias y se realizarán pruebas.
5. *Implementación en sitio.* Se instalará el sistema embebido en un par de aulas y desde una PC controlar los diversos servicios, se realizarán pruebas de control inalámbrica y se realizarán los ajustes necesarios tanto al software como a la programación del embebido hasta dejar completamente funcional el sistema.

3. Cronograma

Se considera el siguiente diagrama de Gantt, tabla 1, donde se presenta la duración de cada etapa de proyecto.

Cronograma de actividades																									
Sistema inalámbrico para domótica en aulas de CU Valles	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10	Semana 11	Semana 12	Semana 13	Semana 14	Semana 15	Semana 16	Semana 17	Semana 18	Semana 19	Semana 20	Semana 21	Semana 22	Semana 23	Semana 24	Semana 25
Anteproyecto - Protocolo																									
Creación y revisión de protocolo																									
Aprobación de protocolo																									
Planteamiento de arquitecturas del sistema																									
Arquitectura de hardware																									
Arquitectura de software																									
Arquitectura de radiocomunicación																									
Desarrollo del sistema																									
Desarrollo de sistema embebido de control																									
Desarrollo de sistema remoto de control (aulas)																									
Desarrollo de sistema estación central																									
Desarrollo de plataforma de comunicación inalámbrica																									
Integración de sistemas																									
Integración de sistema embebido remoto y central																									
Integración de sistemas embebidos y aplicación central																									
Pruebas FAT (factory acceptance test)																									
Implementación en sitio																									
Pruebas SAT (site acceptance test)																									

Tabla 1 Cronograma de actividades

4. Componentes del sistema

4.1 Red inalámbrica

Para poder tener comunicación entre aula y estación central, se determinó usar transceivers que funcionan en la banda 900, llamados Xbee de la compañía Digi.

Estos dispositivos utilizan un protocolo propietario llamado DigiMesh, la compañía los define de la siguiente manera: DigiMesh es una topología de red tipo malla inalámbrica patentada desarrollada por los expertos en ingeniería de RF de Digi que permite nodos "durmientes" sincronizados en el tiempo y de bajo consumo. Un aspecto único de DigiMesh, en comparación con otros protocolos como ZigBee® o Z-Wave, es que todos los dispositivos en una red DigiMesh son del mismo tipo de dispositivo. No se requiere una arquitectura compleja para definir diferentes nodos en una red como nodos finales, enrutadores, coordinadores, enrutadores fronterizos, etc. Todos los dispositivos son iguales y capaces de enrutar, "dormir" para la optimización de energía y comunicarse a través de una red tipo malla [12].

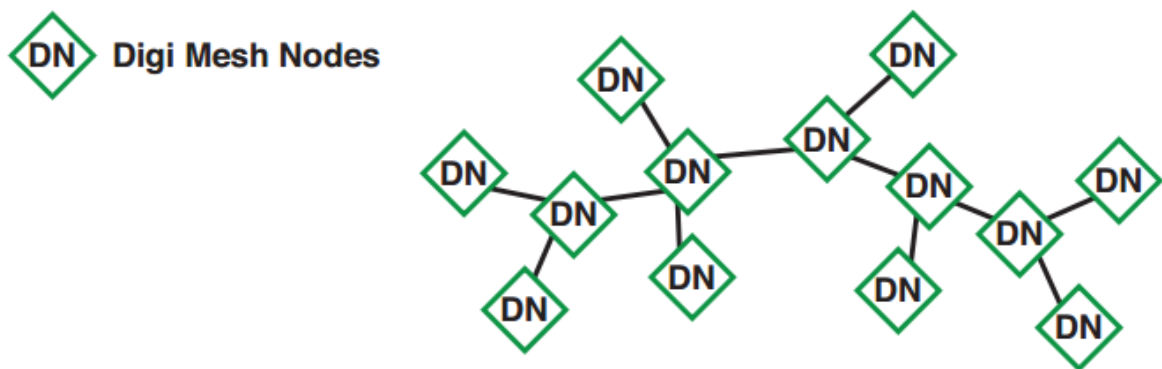


Figura 4 Diagrama de Conexión de nodos DigiMesh

A su vez se eligió el modelo Digi XBee-PRO 900HP el cual se describe de la siguiente forma: Los módulos integrados Digi XBee-PRO 900HP proporcionan la mejor conectividad inalámbrica de su clase. Aprovechan el protocolo de red DigiMesh®, que ofrece una operación de red potente y soporte para enrutadores inactivos, y también están disponibles en una configuración patentada de punto a multipunto. Admiten rangos de línea de visión de RF de hasta 28 millas (con antenas de alta ganancia) y velocidades de datos de hasta 200 Kbps, estos módulos son ideales para aplicaciones de rango extendido que requieren un mayor rendimiento de datos [13].



Figura 5 Digi XBee-PRO 900HP

Las especificaciones completas se encuentran a continuación:

Hardware	Digi XBee-PRO 900HP
Hardware	
Processor	ADF7023 transceiver, Cortex-M3 EFM32G230 @ 28 MHz
Frequency Band	902 to 928 MHz, software selectable channel mask for interference immunity
Antenna Options	Wire, U.FL and RPSMA
Performance	
RF Data Rate	10 Kbps or 200 Kbps
Indoor/Urban Range*	10 Kbps: up to 2000 ft (610 m); 200 Kbps: up to 1000 ft (305 m)
Outdoor/ Line-Of-Sight Range*	10 Kbps: up to 9 miles (14 km); 200 Kbps: up to 4 miles (6.5 km) (w/ 2.1 dB dipole antennas)
Transmit Power	Up to 24 dBm (250 mW) software selectable
Receiver Sensitivity	-101 dBm @ 200 Kbps, -110 dBm @ 10 Kbps
Features	
Data Interface	UART (3V), SPI
GPIO	Up to 15 Digital I/O, 4 10-bit ADC inputs, 2 PWM outputs
Networking Topologies	DigiMesh, Repeater, Point-to-Point, Point-to-Multipoint, Peer-to-Peer
Spread Spectrum	FHSS (Software Selectable Channels)
Power	
Supply Voltage	2.1 to 3.6 VDC
Transmit Current	215 mA
Receive Current	29 mA
Sleep Current	2.5 uA
Regulatory Approvals	
FCC (USA)	MCQ-XB900HP
IC (Canada)	1846A-XB900HP
C-Tick (Australia)	Yes
Anatel (Brazil)	Yes
IDA (Singapore)	Yes
IFETEL (Mexico)	Yes

Tabla 2 Especificaciones de XBee-PRO 900HP

Adicionalmente se acoplara una antena dipolo de alta ganancia para poder transmitir a distancia más grandes.



Figura 6 Antena dipolo 2.1 dBi

Las características de la antena son las siguientes:

- Jack RPSMA-macho
- 2,1 dBi
- Dipolo de media onda, omnidireccional
- 902-928 MHz
- Temperatura de funcionamiento -30 a 60 C
- 50 ohms

4.2 Configuración del Xbee

Primeramente es necesario indicar que el Xbee cuenta con 20 pines los cuales para fines del proyecto estamos utilizando únicamente 6, los cuales se describen a continuación:

Pin#	Name	Direction	Default State	Description
1	VCC			Power supply
2	DOUT/DIO13	Both	Output	GPIO/UART data in
3	DIN/CONFIG /DIO14	Both	Input	GPIO/UART data in
5	RESET	Input		Device reset. Drive low to reset the device. This is also an output with an open drain configuration with an internal 20 kΩ pull-up (never drive to logic high, as the device may be driving it low). The minimum pulse width is 1 ms.
10	GND			Ground
15	Associate/DIO5	Both	Output	GPIO/associate indicator

Tabla 3 Descripción de pines de Xbee

En la siguiente imagen se pueden observar la distribución de los pines del Xbee de manera física:

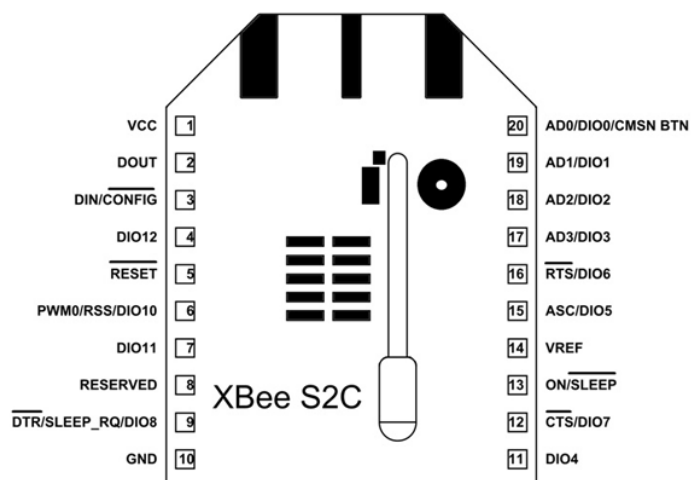


Figura 6 Distribución de los pines del xbee

4.2.1 Modo de operación

El módulo de comunicación inalámbrica RF XBee-PRO 900HP utiliza una base de firmware de varias capas para ordenar el flujo de datos, dependiendo de la configuración de hardware y software que se elija. El siguiente gráfico muestra un diagrama de bloques de la configuración, con la interfaz serial del host como el punto de partida físico y la antena como el punto final para los datos transferidos. Mientras un bloque pueda tocar otro bloque, las dos interfaces pueden interactuar. Por ejemplo, si el dispositivo está usando el modo SPI, el modo transparente no está disponible.

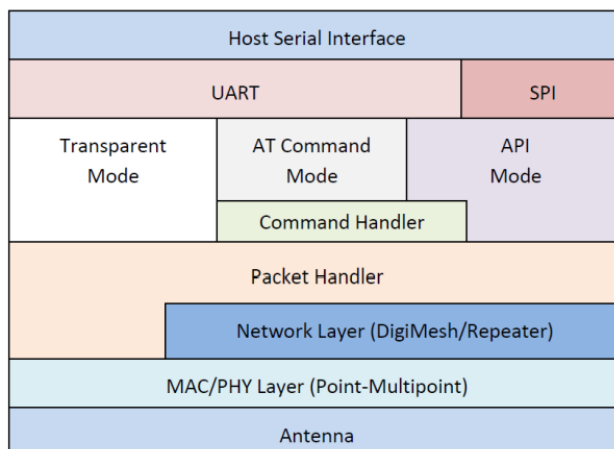


Figura 7 Modo de operación del Xbee

4.2.2 UART

Los dispositivos que tienen una interfaz UART se conectan directamente a los pines del módulo RF XBee-PRO 900HP como se muestra en la siguiente figura.

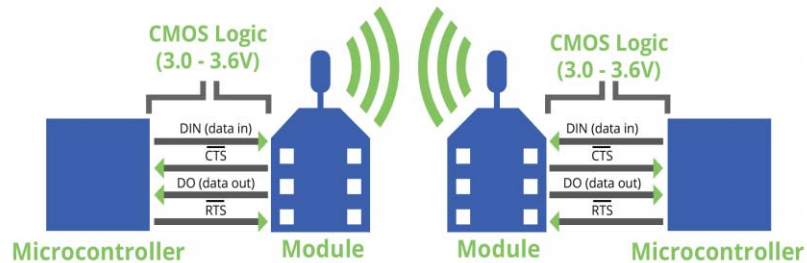


Figura 8 Flujo de datos del sistema en un entorno con interfaz UART

4.2.3 Modo transparente

Los dispositivos funcionan en este modo de manera predeterminada. El dispositivo actúa como un reemplazo de línea en serie cuando está en modo de operación transparente. El dispositivo pone en cola todos los datos UART que recibe a través del pin DIN para la transmisión de RF. Cuando un dispositivo recibe datos de RF, los envía a través del pin DOUT.

El dispositivo almacena los datos en el búfer de recepción en serie hasta que uno de los siguientes provoca que los datos se empaqueten y transmitan:

- El dispositivo no recibe caracteres en serie durante el tiempo determinado por el parámetro RO (Packetization Timeout). Si RO = 0, el empaquetado comienza cuando se recibe un carácter.
- El dispositivo recibe la secuencia de modo de comando (GT + CC + GT). Cualquier carácter almacenado en el búfer de recepción en serie antes de que se transmita la secuencia.
- El dispositivo recibe el número máximo de caracteres que cabe en un paquete RF (100 bytes).

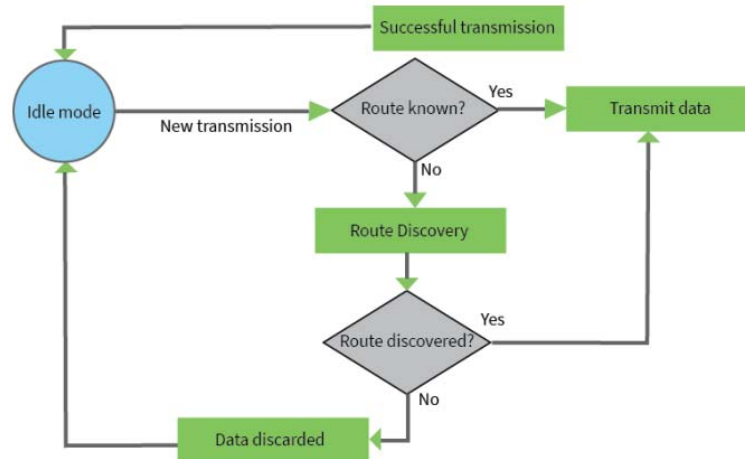


Figura 9 Modo transparente

4.2.4 Modo Inactivo (Idle)

Cuando no recibe ni transmite datos, el dispositivo está en modo inactivo. Durante el modo inactivo, el dispositivo escucha datos válidos en los puertos serie y RF. El dispositivo cambia a los otros modos de operación en las siguientes condiciones:

- Modo transmisión (los datos en serie en el búfer de recepción en serie están listos para ser empaquetados).
- Modo recepción (datos de RF válidos recibidos a través de la antena).
- Modo comando (secuencia de modo de comando emitida, no disponible con el software Smart Energy o cuando se usa el puerto SPI).

4.2.5 Modo transmisión

Los datos en DigiMesh se transmiten de un nodo a otro, el nodo de destino transmite un "ACK" de nivel de red a través de la ruta establecida al nodo de origen. Este paquete de recepción (ACK) indica al nodo de origen que el nodo de destino recibió el paquete de datos. Si el nodo de origen no recibe un ACK de recibo de la red, retransmite los datos.

4.2.6 Modo recepción

Este es el modo predeterminado para el módulo de RF XBee-PRO 900HP. El dispositivo está en modo Recibir cuando no está transmitiendo datos. Si un nodo de destino recibe

un paquete RF válido, el nodo de destino transfiere los datos a su búfer de transmisión en serie.

4.2.7 Modo comando

El modo de comando es un estado en el que el firmware interpreta los caracteres entrantes como comandos. Le permite modificar la configuración del dispositivo utilizando parámetros que puede establecer utilizando comandos AT.

4.2.8 XCTU

XCTU es una aplicación multiplataforma gratuita diseñada para permitir a los desarrolladores interactuar con los módulos Digi RF a través de una interfaz gráfica fácil de usar. Incluye nuevas herramientas que facilitan la configuración y prueba de los módulos XBee RF.

Con esta herramienta se realiza la configuración de los xbee, para que se pueda crear una red de los mismos. Recordemos que las redes Digimesh se asemejan a una topología punto-multipunto, donde todos los dispositivos están conectados entre ellos para poder garantizar la entrega de los mensajes.

Es importante señalar que estamos usando dos tipos de configuraciones para el proyecto:

Nodo master – Nodo que estará conectado al servidor y tendrá comunicación con los nodos de las aulas.

Nodo aula – Nodo que se encuentra en las aulas, y manda datos al microcontrolador, este a pesar de tener comunicación con otros nodos aula, desecha esos mensajes antes de enviarlos al microcontrolador.

4.2.9 Configuración de nodo

La configuración se realiza conectando el xbee al equipo de cómputo, una manera sencilla es utilizar una placa de desarrollo grove, provistas por Digi, la placa es la siguiente:

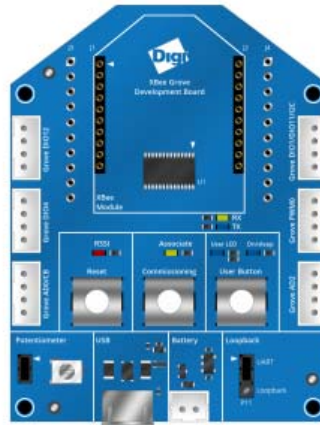


Figura 10 Placa de desarrollo Grove

Esta placa cuenta con un puerto micro usb, el cual se puede conectar al equipo de cómputo mediante un cable micro USB - USB b, existen placas genéricas llamadas Xbee explorer que cumplen con la misma función que la placa anteriormente mencionada, ambas darán los mismos resultados.

Una vez conectado el xbee al equipo de cómputo, mediante el programa XCTU, se debe agregar el dispositivo, utilizando el puerto COM asignada a la placa así como las especificaciones de la comunicación serial.

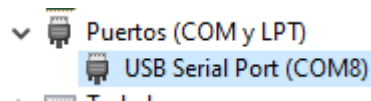


Figura 11 Puerto COM asignado a la placa

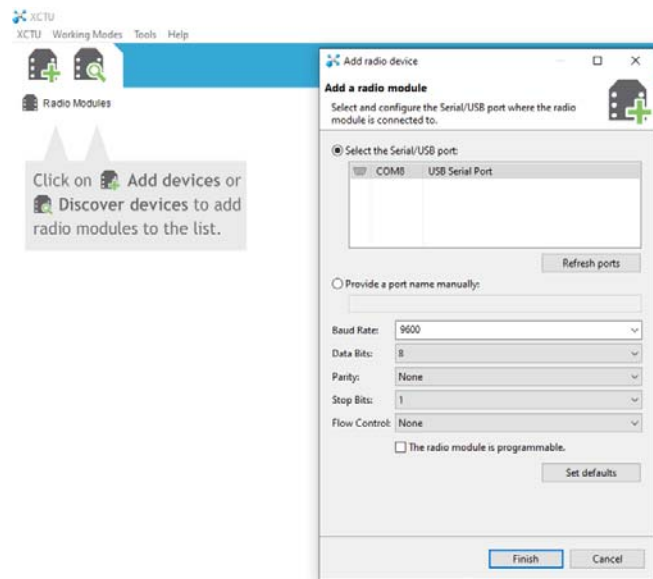


Figura 12 Agregando el xbee a la herramienta de configuración XCTU

Una vez agregado el dispositivo, tendremos acceso a sus configuraciones que se explicaran a continuación.

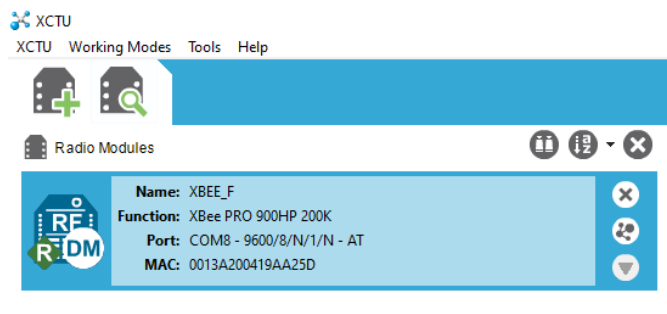


Figura 13 Dispositivo agregado a la herramienta

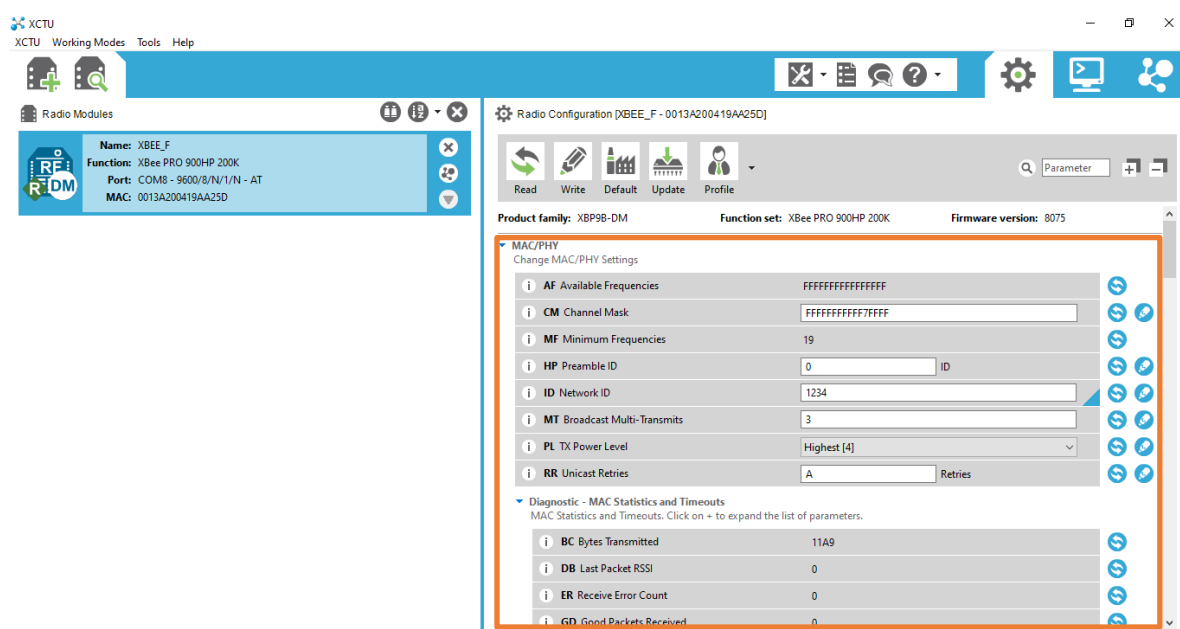


Figura 14 Parámetros del dispositivo

Es importante explicar ciertos parámetros a modificar:

ID – Network ID: Este parámetro es el indicador único de la red, es decir si lo comparamos con una red wifi, sería el equivalente al nombre de la red. Este parámetro es idéntico en los nodos master y aula. Valor configurado: 1234

PL – TX Power Level: Controla la ganancia en la salida de señal RF, es decir controla los dBm en la antena. Este parámetro es idéntico en los nodos master y aula. Valor configurado: Highest[4].

CE – Routing/Messaging Mode: Determina el tipo de nodo a utilizar, es posible configurar para que ciertos nodos no retransmitan un mensaje de tipo broadcast a otros nodos. Este parámetro es idéntico en los nodos master y aula. Valor configurado: Standard Router[0]

MR – Mesh Unicast Retries: Controla las veces que un mensaje fallido intentara entregarse de nuevo. Este parámetro es idéntico en los nodos master y aula. Valor configurado: 2

DH – Destination Address High: Determina parte de una dirección física del xbee único donde tomara mensajes como válidos, si recibe mensajes de otros dispositivos, así estén en la misma red, los desechara. Este parámetro es diferente en nodo master y en el nodo aula para evitar mensajes innecesarios en múltiples aulas (ruido).

Valor configurado en master: 0, Valor configurado en aula: 13A200 (DH de xbee master)

DL – Destination Address Low: Determina parte de una dirección física del xbee único donde tomara mensajes como válidos, si recibe mensajes de otros dispositivos, así estén en la misma red, los desechara. Este parámetro es diferente en nodo master y en el nodo aula para evitar mensajes innecesarios en múltiples aulas (ruido).

Valor configurado en master: 0000FFFF, Valor configurado en aula: 41807E6F (DL de xbee master)

NI – Node Identifier: Nombre del nodo. Este parámetro es diferente en todos los nodos.

EE – Encryption Enable: Determina si se utilizara encriptación en la red (AES 128-bit). Este parámetro es idéntico en los nodos master y aula. Valor configurado: Enabled[1]

KY – AES Encryption Key: Llave de la encriptación AES 128-bit. Este parámetro es idéntico en los nodos master y aula.

BD – Baud Rate: Baud Rate de la comunicación serial, recordar que tiene que ser el mismo que el dispositivo con el cual se conecta, es decir, Servidor (Labview) o Microcontrolador. Valor configurado: 9600[3].

NB – Parity: Paridad de la comunicación serial, recordar que tiene que ser el mismo que el dispositivo con el cual se conecta, es decir, Servidor (Labview) o Microcontrolador. Valor configurado: No Parity[0].

SB – Stop Bits: Bits de Parada de la comunicación serial, recordar que tiene que ser el mismo que el dispositivo con el cual se conecta, es decir, Servidor (Labview) o Microcontrolador. Valor configurado: One Stop Bit[0].

FT – Flow Control Threshold: Limite de control de flujo, controla la cantidad máxima de bytes que se almacenan en el buffer antes de mandar los mensajes por RX o TX, recordar que los mensajes enviados y recibidos no superan 5 bytes. Valor configurado: 11.

AP – API Enable: Determina el modo de trabajo del xbee, transparente o API. Este parámetro es idéntico en los nodos master y aula. Valor configurado: Transparent Mode[0].

SM – Sleep Mode: Determina si el xbee puede entrar en estado de inactividad (Sleep), para ahorro de energía, como no se alimentara con baterías, se configuro para que no entrada en inactividad. Este parámetro es idéntico en los nodos master y aula. Valor configurado: Normal[0].

El resto de parámetros se pueden dejar con configuración default. Una vez que se realizaron los cambios pertinentes en el xbee deseado, se tendrán que grabar dichos cambios en el mismo, para eso se utiliza el botón Write ubicado en la parte superior.

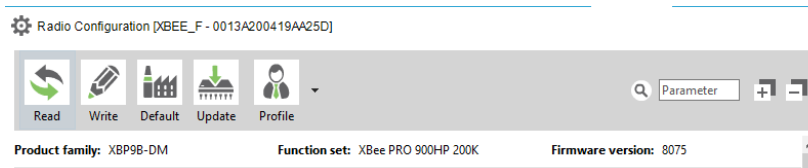


Figura 15 Botones de acciones en XCTU

Adicionalmente es recomendable que todos los xbee tengan cargados el firmware más actual, en este caso la versión 8075, en caso de requerir actualizar algún Xbee, se puede usar el botón update, donde se puede elegir la versión de firmware a cargar en el dispositivo.

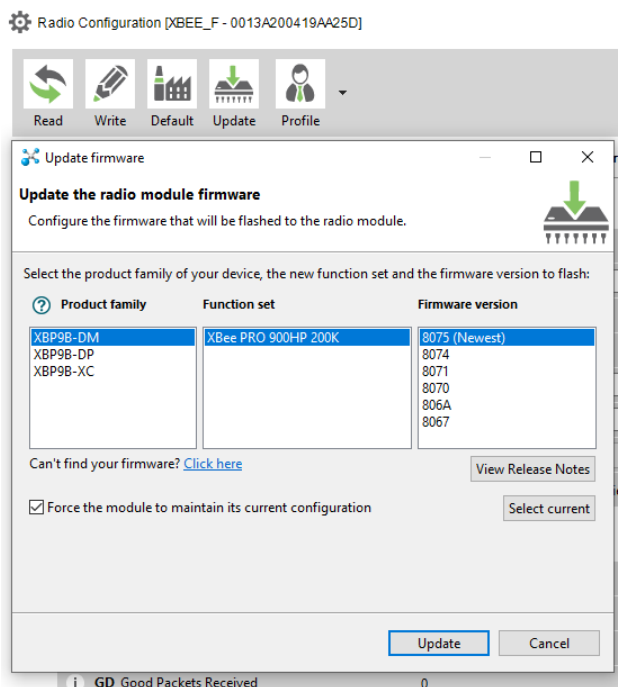


Figura 16 Actualización de firmware de xbee

Header Female 2-54 1x2

P2 WJ127-50-2P

DC1 DC005-0.0MM

12V_IN

12V_IN

U1 LM7805EE

12V

5V

C1 330nF

C2 100nF

GND

U5 L0117V03

12V

5V

C3 330nF

C4 100nF

GND

CD4072BE

A=B+C+D

K=E+F+G+H

VDD

VSS

P1R1

P1R2

P1R3

P1R4

P1R5

P1R6

P1R7

P1R8

P1R9

P1R10

P1R11

P1R12

P1R13

P1R14

P1R15

P1R16

P1R17

P1R18

P1R19

P1R20

P1R21

P1R22

P1R23

P1R24

P1R25

P1R26

P1R27

P1R28

P1R29

P1R30

P1R31

P1R32

P1R33

P1R34

P1R35

P1R36

P1R37

P1R38

P1R39

P1R40

P1R41

P1R42

P1R43

P1R44

P1R45

P1R46

P1R47

P1R48

P1R49

P1R50

P1R51

P1R52

P1R53

P1R54

P1R55

P1R56

P1R57

P1R58

P1R59

P1R60

P1R61

P1R62

P1R63

P1R64

P1R65

P1R66

P1R67

P1R68

P1R69

P1R70

P1R71

P1R72

P1R73

P1R74

P1R75

P1R76

P1R77

P1R78

P1R79

P1R80

P1R81

P1R82

P1R83

P1R84

P1R85

P1R86

P1R87

P1R88

P1R89

P1R90

P1R91

P1R92

P1R93

P1R94

P1R95

P1R96

P1R97

P1R98

P1R99

P1R100

P1R101

P1R102

P1R103

P1R104

P1R105

P1R106

P1R107

P1R108

P1R109

P1R110

P1R111

P1R112

P1R113

P1R114

P1R115

P1R116

P1R117

P1R118

P1R119

P1R120

P1R121

P1R122

P1R123

P1R124

P1R125

P1R126

P1R127

P1R128

P1R129

P1R130

P1R131

P1R132

P1R133

P1R134

P1R135

P1R136

P1R137

P1R138

P1R139

P1R140

P1R141

P1R142

P1R143

P1R144

P1R145

P1R146

P1R147

P1R148

P1R149

P1R150

P1R151

P1R152

P1R153

P1R154

P1R155

P1R156

P1R157

P1R158

P1R159

P1R160

P1R161

P1R162

P1R163

P1R164

P1R165

P1R166

P1R167

P1R168

P1R169

P1R170

P1R171

P1R172

P1R173

P1R174

P1R175

P1R176

P1R177

P1R178

P1R179

P1R180

P1R181

P1R182

P1R183

P1R184

P1R185

P1R186

P1R187

P1R188

P1R189

P1R190

P1R191

P1R192

P1R193

P1R194

P1R195

P1R196

P1R197

P1R198

P1R199

P1R200

P1R201

P1R202

P1R203

P1R204

P1R205

P1R206

P1R207

P1R208

P1R209

P1R210

P1R211

P1R212

P1R213

P1R214

P1R215

P1R216

P1R217

P1R218

P1R219

P1R220

P1R221

P1R222

P1R223

P1R224

P1R225

P1R226

P1R227

P1R228

P1R229

P1R230

P1R231

P1R232

P1R233

P1R234

P1R235

P1R236

P1R237

P1R238

P1R239

P1R240

P1R241

P1R242

P1R243

P1R244

P1R245

P1R246

Figura 18 Vista 3D de la PCB

Todos los componentes de la PCB se describirán a continuación:

4.3.1 Alimentación

En cuanto a la alimentación se puede hacer de dos maneras, ya se utilizando el conector JACK 5x2.1mm (DC1) o en su caso tiene la opción de una bornera (P1), así mismo en P2 se puede colocar un switch externo para encender/apagar la placa por completo.

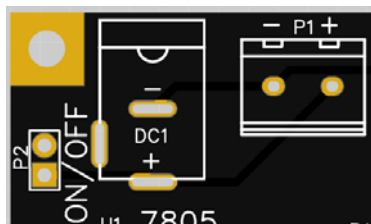


Figura 19 Alimentación

4.3.2 Reguladores de tensión

Los circuitos utilizados se alimentan con dos tipos de tensiones, 5V y 3v3 para lo cual se colocaron reguladores de tensión a 5v (U1) y 3v3 (U5) con sus respectivos capacitores. Se colocó un led indicador a la salida de cada uno de los reguladores de tensión.

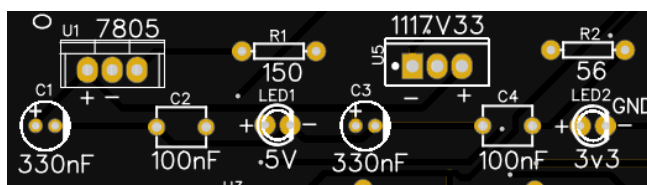


Figura 20 Reguladores de tensión y leds indicadores

4.3.3 Xbee

Para el Xbee se tuvo que realizar una modificación en el socket del mismo, ya que utiliza pines de 2mm en lugar del estándar de 2.54mm, únicamente fuimos capaces de conseguir un conector de 13x2 en esa medida, así que se realizó la adecuación en el componente.

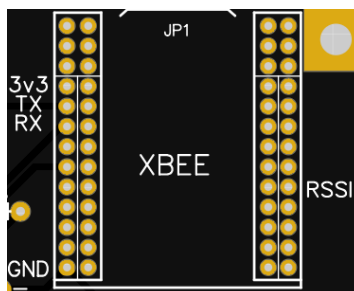


Figura 21 Xbee

4.3.4 Leds indicadores de RX, TX e inactividad

Para poder visualizar el momento en el Xbee mande datos al microcontrolador, se colocaron leds en RX y TX, adicionalmente existe un pin del Xbee que es una señal intermitente que indica que el dispositivo funciona correctamente (RSSI), aparte de esto se tiene un led de inactividad que indica cuando el PIR no detecta actividad y se apagan los servicios actuales.

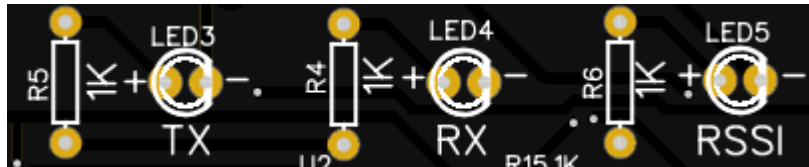


Figura 22 Leds TX, RX, RSSI

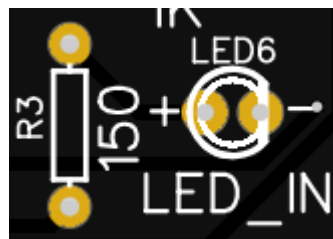


Figura 23 Led de inactividad

4.3.5 CI 74240

Este circuito integrado es un CI de inversores óctuple (U2), que se utilizan en las salidas del puerto B, los cuales controlan un módulo externo de relevadores, los cuales se activan con GND, y esto último es el propósito del circuito mencionado. Es importante mencionar que cuentan con resistencias pull down.

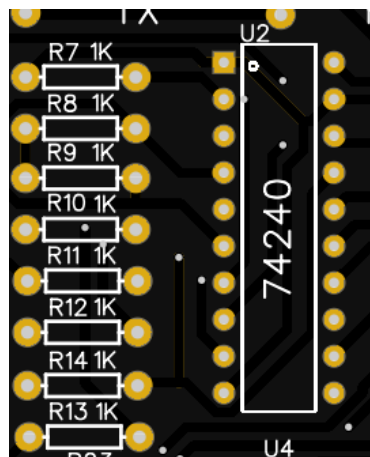


Figura 24 CI 74240

4.3.6 CI CD4072

Este circuito integrado es una compuerta lógica OR de 4 entradas (U8), esto con el fin de colocar todas las entradas de los PIR y tener una única salida hacia el PIC, con esto se controla la inactividad en el aula.

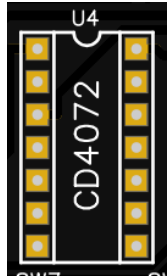


Figura 25 CI CD4072

4.3.7 PIC18F4550

Este circuito integrado es el microcontrolador el cual es el principal de la placa, donde todo se procesara.

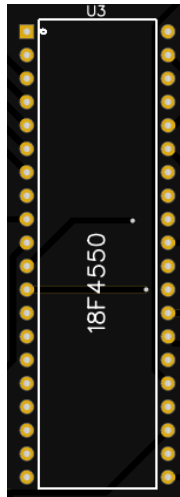


Figura 26 CI PIC18F4550

4.3.8 Pines para entradas y salidas

Principalmente se tiene el puerto D como entradas, es decir aquí se conectarán señales para encender servicios de manera local, así mismo las salidas correspondientes a los servicios están en el puerto B. Adicionalmente los puertos A, C y E tienen pines disponibles para poder usarse en un futuro como entradas/salidas si se requiriera. Adicionalmente también se tienen las entradas de los PIR, así como algunas salidas de voltaje/GND.

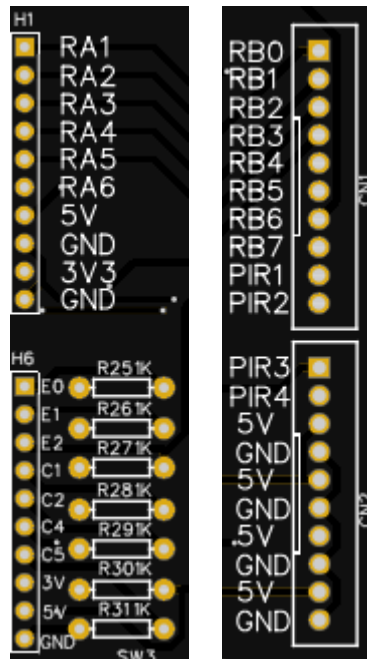


Figura 27 Pines de Entradas y Salidas

4.3.9 Push Buttons

Se colocaron 10 push button en la PCB, 2 de ellos se utilizar para reiniciar el PIC y el Xbee respectivamente, el resto están conectados al Puerto D para poder activar los servicios localmente, de manera adicional a estos, se colocaron salidas disponibles en caso de requerir un botón que no esté conectado directamente en la PCB, como puede ser en un enclosure.

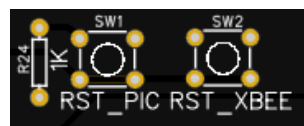


Figura 28 Push Botton para reset

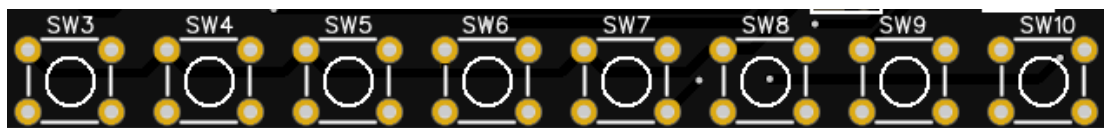


Figura 29 Push Buttons conectados al puerto D

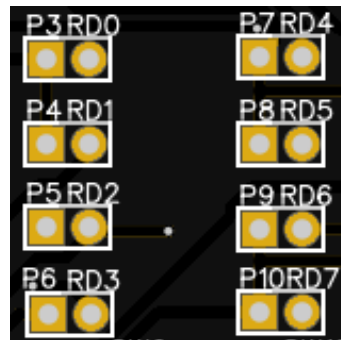


Figura 30 Entradas adicionales en paralelo a los push buttons del puerto D

4.3.10 Mounting Holes

En cada esquina se colocaron mounting holes de medida M3 para poder sujetar la PCB a un enclosure.

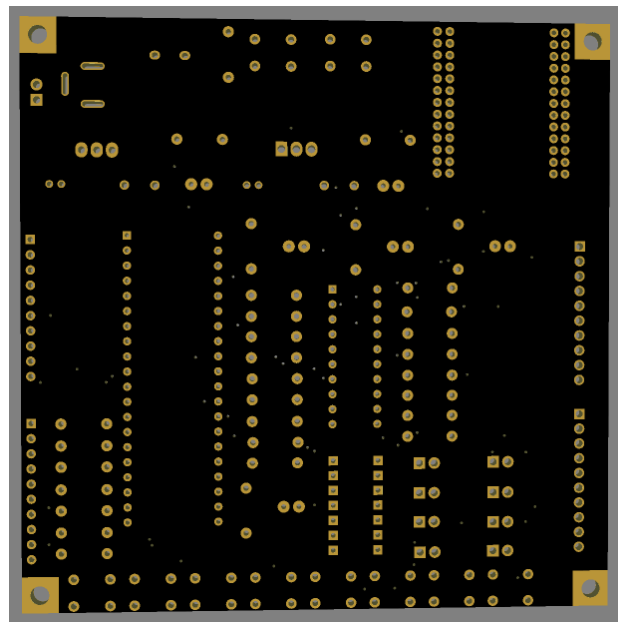


Figura 31 Mounting Holes

4.3.11 Capas - Top

En la siguiente imagen podemos observar las pistas en el top del PCB.

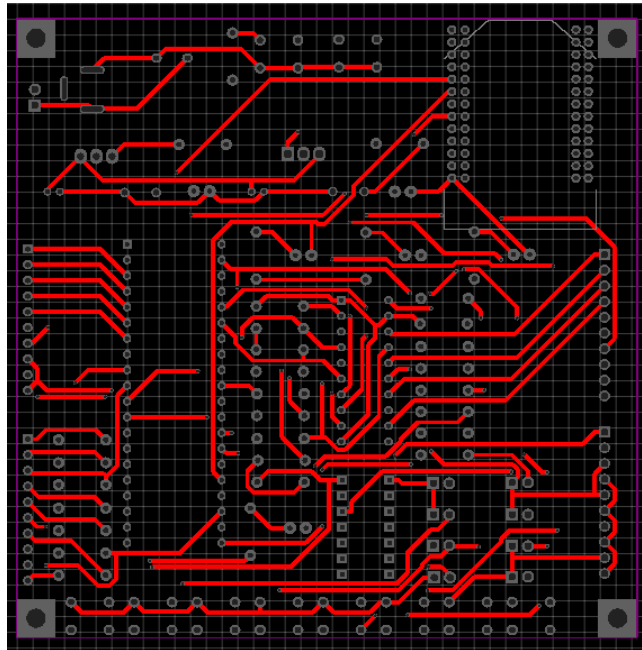


Figura 32 Pistas - Top

4.3.12 Capas - Boton

En la siguiente imagen podemos observar las pistas en el top del PCB.

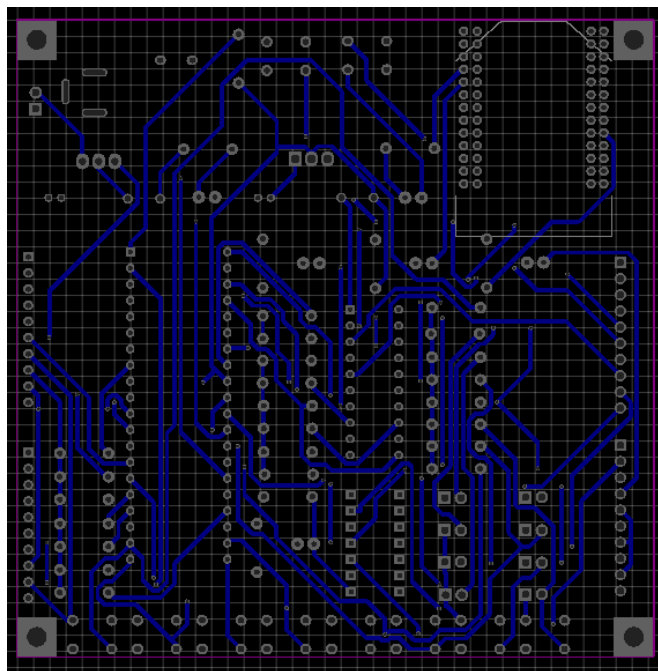


Figura 33 Pistas - Boton

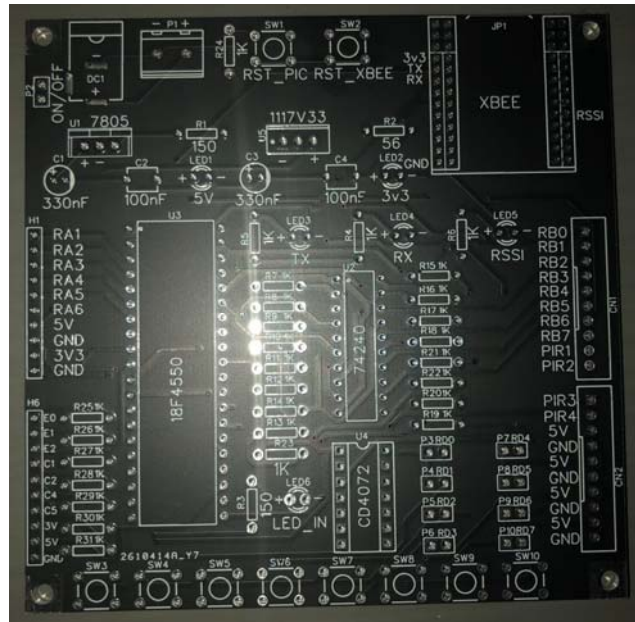


Figura 36 PCB Real

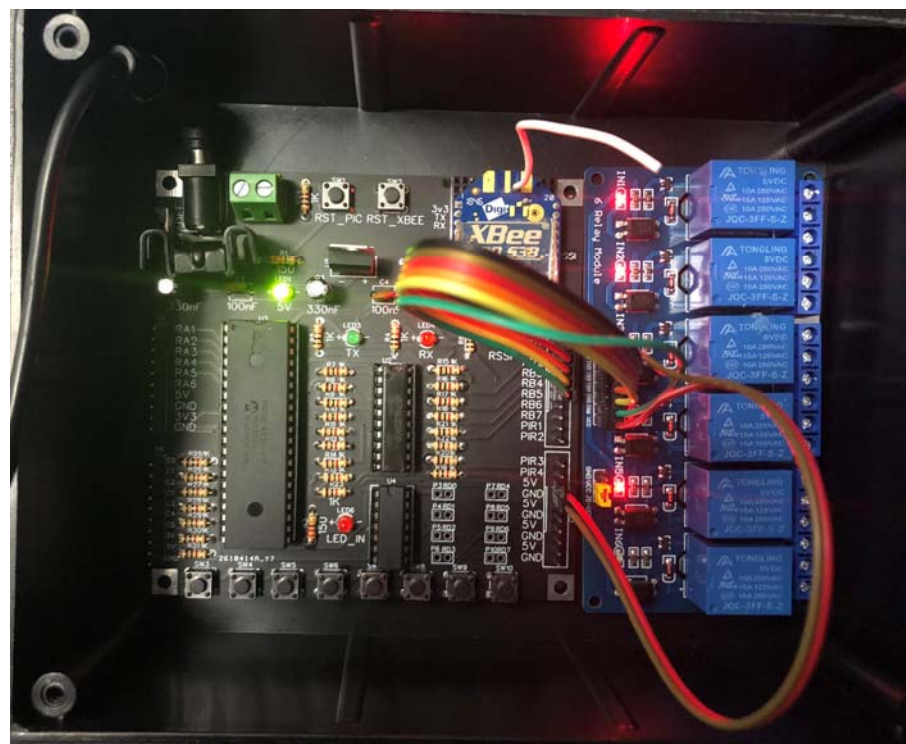


Figura 37 PCB Final

4.4 Control en aula

El control principal del aula se basa en un microprocesador PIC18F4550 de Microchip.



Figura 38 PIC18F4550

Las características principales del microprocesador son las siguientes:

Parametrics	
Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	32
CPU Speed (MIPS/DMIPS)	12
SRAM Bytes	2048
Data EEPROM/HEF (bytes)	256
Digital Communication Peripherals	1-UART, 1-SPI, 1-I2C1-MSSP(SPI/I2C)
Capture/Compare/PWM Peripherals	1 CCP, 1 ECCP,
Timers	1 x 8-bit, 3 x 16-bit
ADC Input	13 ch, 10-bit
Number of Comparators	2
Number of USB Modules	1, FS Device
Temperature Range (°C)	-40 to 85
Operating Voltage Range (V)	2 to 5.5
Pin Count	40

Tabla 4 Especificaciones de PIC18F4550

Otro dispositivo esencial en la parte de control del aula, es un sensor PIR de presencial el cual ayudara a determinar cuando no exista actividad en la misma. Se eligió el modelo HC-SR501 ya que cuenta con 2 potenciómetros que permiten regular la sensibilidad y el tiempo de duración del pulso. También posee unos jumpers para configurar el modo de la señal de salida.



Figura 39 Sensor PIR HC-SR501

Las características del sensor son las siguientes:

- Voltaje de operación: 4.5VDC - 20VDC
- Consumo de corriente en reposo: <50uA
- Voltaje de salida: 3.3V (alto) / 0V (bajo)
- Rango de detección: 3 a 7 metros, ajustable mediante Trimmer (Sx)
- Angulo de detección: <100° (cono)
- Tiempo de retardo: 5-200 S (puede ser ajustado (Tx), por defecto 5S +-3%)
- Tiempo de bloqueo: 2.5 S (por defecto)
- Temperatura de trabajo: -20°C hasta 80°C
- Dimensión: 3.2 cm x 2.4 cm x 1.8 cm (aprox.)
- Re-disparo configurable mediante jumper de soldadura
-

Respecto al control de acceso, se utilizara un kit de la marca Obo Hands, el cual cuenta con Lector RFID, cerradura electrónica, botón de salida, así como una fuente regulada.

El lector RFID cuenta con las siguientes características:

- Voltaje de suministro: 12VDC/2A
- Temperatura de trabajo: -20°C-70 °C
- Humedad Relativa: 0% ~ 90%
- Capacidad de memoria: 8000 usuarios de 8000 piezas de tarjetas
- Método de desbloqueo: tarjeta RFID, contraseña, tarjeta RFID + contraseña
- Contraseña: código PIN de 4 dígitos
- Tipo de identificación de la tarjeta: EM o compatible
- Rango de lector de tarjetas: 0-10 cm
- Interfaz Weigand: entrada y salida WG26/34.
- Dimensión: 126*51*20mm



Figura 40 Kit de control de acceso

4.5 Programación

La programación se divide en 2 partes, Aula y Estación Central.

Como Aula entendemos cada uno de los salones que componen un Edificio, se consideran aulas desde el 1 hasta el 16 y los Edificios los identificamos con letras mayúsculas del abecedario (A, B, C, D, etc). Cada Aula tiene 1 XBEE (con su antena) que recibe datos de forma inalámbrica y 1 microcontrolador que interpreta los datos recibidos y controla los servicios del Aula, el Micro es único para cada Aula, puesto que en su programación se identifica con su Edificio y Aula correspondiente.

Como Estación Central entendemos el servidor o computadora que aloja el programa del usuario final junto con el XBEE y su antena correspondiente, ésta debe estar en un rango adecuado de distancia (dependiendo de las antenas y obstáculos se pueden lograr distancias de 5 Km), de tal forma que se puede tener comunicación con cada aula de cada edificio.

Sabemos que 1 byte consta de 8 bits y que se tiene la capacidad de tener 255 valores; se debe entender la cadena de comunicación Serial que se emplea, la cadena consta de 5 bytes que se envía del Microcontrolador a la Estación central, y 4 bits que se envían de la estación central hacia aulas, las cuales están de la siguiente manera:

Estación Central --> Microcontrolador en Aula

Byte	Uso	Ejemplo ASCII	Decimal
0	Edificio	A	65
1	Aula	(41
2	Status	s,t,x	115,116,34
3	Fin de cadena	_	95

Tabla 5 Bytes Estación Central --> Microcontrolador en Aula

Byte 0.- Corresponde al Edificio al que se enviaran datos y son letras en mayúsculas como A, B, C, D, etc.

Byte 1.- Es el valor correspondiente al Aula del edificio (Byte0) con el que se establecerá la comunicación; del 1 al 16, haciendo un corrimiento al sumarle 33; esto es para evitar los caracteres del 0-31 ya que son caracteres de control; se presentaron problemas durante pruebas debido a que el programa en la estación central interpretaba dichos caracteres como de sistema (de Windows), por esta razón se decidió hacer un corrimiento.

Byte 2.- Byte de Status, aquí se le ordena al aula encender o apagar cuales quiera de los 5 servicios (denotado por "x" pero puede ser cualquier carácter que se forma con máximo 6 bits). Cuando se envía "s" o "t" se solicita el estado de los servicios; si la solicitud es del Monitor (programa principal) éste envía "s"; si la solicitud por medio de un sub programa, éste envía "t". La diferencia estriba en que al monitor se le envía el status actual de los servicios, mientras que a los subprogramas se les envía el valor de una variable (backup) en la que se guarda el estado de los servicios cuando el aula entra en estado de inactividad (debido al sensor de presencia), esta variable se va actualizando constantemente cuando hay un cambio en los servicios; cuando hay inactividad y el monitor solicita status, éste recibe 0, mientras que si un subprograma solicita status recibe el valor de la variable backup, esto con la intención de conocer que servicios se tienen permitido utilizar.

Es importante mencionar que **x**, es un número DEC o carácter ASCII que se pueda construir con máximo 6 bits (en decimal 63), puesto que el Byte 2 se utiliza para solicitar status y establecer servicios; pudiera existir un caso de traslape, cuando sean 7 bits de servicios, ya que en decimal se tendría un valor máximo de 127 de servicios, pero para status utilizamos los caracteres "s" y "t", que son los caracteres 115 y 116 respectivamente, el sistema pudiera confundirse, ya que en vez de establecer un estado de los servicios, estará regresando una solicitud de status. Se tienen contemplado como máximo 6 servicios (6 bits, 1 para cada servicio), en este proyecto se implementaron 5 servicios, pero se puede expandir de diversas maneras; ya sea recorriendo los caracteres de solicitud de estado a partir del 161 en decimal, con lo que se podrán tener 7 servicios, o en dado caso de requerir más, se puede implementar otro bit de status en el que se manipulen más servicios, limitado por pines de I/O del microcontrolador.

Byte 3.- Carácter de fin de cadena, el carácter "_" lo utilizamos para denotar el fin de la cadena que envía la estación central.

Microcontrolador --> Estación central

Byte	Uso	Ejemplo	Decimal
0	Edificio	A	65
1	Aula	(41
2	Status	"	34
3	Inactividad	!	33
4	Fin de cadena	?	95

Tabla 6 Bytes Microcontrolador --> Estación central

Byte 0.- Identifica el Edificio del aula, igual que en la tabla anterior

Byte 1.- Numero de aula en la que se encuentra el dispositivo, con su corrimiento, explicado en la sección anterior.

Byte 2.- Status, dependiendo quien lo solicite; si es Monitor se envía el status actual de los servicios (valor del puerto B del micro; si son los subprogramas, se envía el valor de la variable backup, con el corrimiento correspondiente.

Byte 3.- Este bit nos dice si el aula está en estado de inactividad, lo determinan los sensores de presencia PIR y es 1 o 0, valor del puerto RC0, con su respectivo corrimiento.

Byte 4.-Carácter de fin de cadena.

4.5.1 Programación de Microcontrolador

Se optó por utilizar el oscilador interno a 8 Mhz del micro; los pines utilizados del microcontrolador se distribuyen de la siguiente manera:

Pin	Uso
RA0	Entrada de sensor PIR
RA1 – RA5	No utilizado, disponible en PCB
RA6	No Conectado
RB0 – RB4	Salida (invertida) para relevadores de Servicios
RB5 – RB7	No utilizado, disponible en PCB, salida invertida para relevador
RC0	Salida de led de inactividad
RC1 – RC5	No utilizado, disponible en PCB
RC6	Serial TX
RC7	Serial RX
RD0 – RD7	Entrada de pushbuttons
RE0 – RE2	No utilizado, disponible en PCB

Tabla 7 Pines Microcontrolador

De manera general, el microcontrolador en el main(), realice un escaneo del puerto D y del pin RA0. En el puerto D se tienen conectados pushbottoms, con los que se modifica el

estado del mismo bit pero en el Puerto B, es decir, encender/apagar los servicios; en el bit RA0, se tiene conectado la salida de una compuerta OR, en la cual están conectados los sensores de presencia PIR.

Al mismo tiempo, el micro se configuró para trabajar con una interrupción cuando se recibe un dato en el pin RC7, es decir la recepción del serial RX; se reciben 4 bytes: Edificio, Aula, Status ("s", "t" o cualquier otro valor hace referencia a prender/apagar servicios), y el ultimo byte corresponde al final de cadena.

En la siguiente tabla, se muestran las líneas de código del programa principal, en la primera columna se muestra en número de línea, en la segunda, el código del programa y en la tercera una explicación del código por bloques.

1	#define _XTAL_FREQ 8000000	Se define frecuencia de
2	#include <xc.h>	Oscilador (_XTAL_FREQ) para
3	#include "FusesCom.h"	la función __delayms().
4	#include "serial.h"	Se cargan librerías
5	#include "config_reg.h"	
6	#include "EEPROM_Libreria.h"	
7	#define AULA 8	definir Aula y Edificio, así
8	#define EDIF 'A'	como la variable LED_PIR
9	#define LED_PIR LATCbits.LC0 //LED de PIR en RC0	como salida LATC0
10	//en un primer inicio, todos los servicios activos	Cuando se programa guarda
11	__EEPROM_DATA(0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00);	0xFF en el primer byte de
		la EEPROM
12	//global var	Global var: backup guarda
13	char backup;	status, inpt -> guarda
14	char inpt[10];	datos de serial RX, x->
15	char x[6]={0x00,0x00,0x00,0x00,0x00,0x00};	array enviado por TX,
16	//var para contar segundos mult de 5	contador -> múltiplos de 5
17	unsigned int contador=0;	seg de timer, para
		establecer tiempo de
		inactividad
18	void interrupt RXx(void);	Dar de alta función
19	void Delay1Second();	interrupción y Delay
20	void main() {	Función main del programa
21	serial_close();	Deshabilita puerto serial
22	internal_clock();	Carga conf de Osc como
		interno
23	config_reg();	Conf de registros E/S,
		timer
24	//PORTB valor del byte de EEPROM, ultimo valor	PORTB = val en EEPROM, byte
25	LATB = EEPROM_Lectura(0);	0
26	LATCbits.LATC0=0;	Led inactividad apagado
27	backup = EEPROM_Lectura(0);	Guarda EEPROM en backup
28	//Habilita interrupcion RX	Habilita interrupción de RX
29	RX_interrupt_enabled();	
30	clear_buffer();	Limpia buffer de serial
31	serial_open(9600);	Conf Serial asíncrono, 8
		bits a 9600 bauds
32	TMR0 = 26472; //TMR0 para timer de 5 seg	Valor de TMR0 para timer de
33	T0CONbits.TMR0ON = 1; //habilita el timer	5 seg, iniciar de timer
34	while(1) {	Inicia ciclo infinito
35	//Espera interrupcion por RX	

<pre> 36 // SERVICIOS MANUAL /////////////////////////////////// 37 if ((PORTDbits.RD0==1) (PORTDbits.RD1==1) (PORTDbits.RD2==1) 38 (PORTDbits.RD3==1) (PORTDbits.RD4==1)){ 39 if (PORTDbits.RD0==1) LATBbits.LATB0=!PORTBbits.RB0; 40 if (PORTDbits.RD1==1) LATBbits.LATB1=!PORTBbits.RB1; 41 if (PORTDbits.RD2==1) LATBbits.LATB2=!PORTBbits.RB2; 42 if (PORTDbits.RD3==1) LATBbits.LATB3=!PORTBbits.RB3; 43 if (PORTDbits.RD4==1) LATBbits.LATB4=!PORTBbits.RB4; 44 //if (PORTDbits.RD5==1) LATBbits.LATB5=!PORTBbits.RB5; 45 //if (PORTDbits.RD6==1) LATBbits.LATB6=!PORTBbits.RB6; 46 //if (PORTDbits.RD7==1) LATBbits.LATB7=!PORTBbits.RB7; 47 LED_PIR=0; 48 contador = 0; 49 backup=PORTB; 50 TMR0 = 26472; //TMR0 para timer de 5 seg 51 INTCONbits.TMR0IF = 0; 52 Delay1Second(); 53 } </pre>		<p>Revisa el puerto D, si se presiona 1 pushbutton en RD0-RD4, hace cambio de estado en el mismo pin, pero del puerto B. Después se apaga led de inactividad, se reestablece variable contador a 0, que cuenta rondas de 5 segundos; se guarda el valor/status del puerto B en backup. Se reestablece valor del timer y se apaga su bandera de desbordamiento; por último, un delay de .5 segundos.</p>
<pre> 53 // PARA TIMER + PIR /////////////////////////////////// 54 if (PORTAbits.RA0==0){ 55 if (INTCONbits.TMR0IF == 1){ 56 contador ++; 57 if (contador == 2){ //timer 5 seg, 10 seg 58 LED_PIR=1; 59 backup=PORTB; 60 LATB = 0x00; 61 } 62 if (contador > 2) contador = 3; //para desbor 63 TMR0 = 26472; //timer 5 seg 64 INTCONbits.TMR0IF = 0; 65 } 66 } 67 if (PORTAbits.RA0==1){ //hay movimiento 68 LED_PIR=0; 69 contador = 0; 70 TMR0 = 26472; //TMR0 para timer de 5 seg 71 LATB=backup; 72 } 73 /// 74 } 75 } 76 } </pre>		<p>En esta sección se verifica el sensor PIR, si se recibe 0, significa que no hay mov, se cuentan múltiplos de 5 seg, al llegar a 10 seg (contador=2) se entra en inactividad, enciende led de inactividad, guarda el status del puerto B en backup y apaga servicios, PORTB=0. Cuando hay mov, apaga el led, reestablece variable contador y de timer y por ultimo despliega en el puerto B el valor almacenado en backup. También agregamos una condición para evitar el desbordamiento de la variable contador y reinicia variables.</p>
<pre> 77 void interrupt RXx(void){ 78 serial_read(inpt); 79 if (inpt[0]==EDIF && inpt[1]==(AULA+33) && inpt[3]=='_'){ 80 if (inpt[2]=='s'){ //Solicita Status de monitor 81 x[0]=EDIF; //Edificio 82 x[1]=(unsigned char)(AULA+33); // +33 ascii 10 83 x[2]=PORTB; 84 x[2]=(unsigned char)(x[2]+33); //+ 33 85 x[3]=(unsigned char)(PORTCbits.RC0+33); // inact 86 x[4]='?'; 87 serial_write(x); 88 } 89 else if (inpt[2]=='t'){ //Sol de sub_vi 90 x[0]=EDIF; //Edificio 91 x[1]=(unsigned char)(AULA+33); // +33 ascii 10 92 x[2]=backup; 93 x[2]=(unsigned char)(x[2]+33); 94 x[3]=(unsigned char)(PORTCbits.RC0+33); 95 x[4]='?'; 96 serial_write(x); 97 } </pre>		<p>Funcion Interrupción cuando hay dato en serial, lee el serial y lo guarda en inpt; Se verifica si el mensaje recibido tiene la forma y si es dirigido a esta aula; si es así, se revisa el 3er carácter, si es "s", se solicita estado del Monitor, con lo que construimos el array x: cargamos Edif, Aula, valor de PORTB, Inactividad y fin de cadena "?", por último, se envía x. Si el dato es "t", solicita status un sub programa, se construye x: se carga Edificio, aula, backup, inactividad y fin de cadena.</p>

98		Si no es "s", "t", entonces
99		es nuevo status de servicio,
100	else{ //escribir dato en PORTB (nuevo dato de serv)	se extrae el valor (quitando
101	inpt[2]=(char)(inpt[2]-33);	corrimiento), se despliega
102	LATB=inpt[2];	el valor en PORTB, se
103	backup = inpt[2];	actualiza backup, y se
104	EEPROM_Guardar(0,inpt[2]);	guarda el nuevo status en
105	LED_PIR=0; //apaga led de inactividad	EEPROM, apaga led de
106	contador = 0; //reestablece conteo	inactividad, se reestablece
107	TMR0 = 26472; //TMR0 para timer de 5 seg	variables de TIMER y se
108	INTCONbits.TMR0IF = 0; //Clear timer flag	limpia bandera.
109	}	
110	PIR1bits.RCIF = 0; //RX interrupt flag;	Limpia bandera de
111	}	interrupción y termina
112		main()
113	void Delay1Second(){	
114	int i = 0;	Función delay, múltiplos de
115	for(i=0;i<50;i++){	500 ms
116	__delay_ms(10);	
	}	
	}	

A continuación, se muestran el código de las librerías:

4.5.2 Librería EEPROM_Libreria.h

Se utilizó para almacenar el estatus de los servicios, si en algún momento el micro se queda sin energía, el micro conservará encendidos los servicios que tenía disponibles antes del corte de energía.

Es importante señalar que cuando se escribe en EEPROM, se tiene que deshabilitar todas las interrupciones, debido a que el micro se cuelga cuando se está escribiendo/leyendo en EEPROM y llegan interrupciones; esto es debido a que cuando se entra en la interrupción, la dirección de retorno se guarda en el STACK y el Program Counter (PC, que es el apuntador a memoria de programa, es decir lo que se va ejecutado) se mueve al vector de interrupción (0000008h – 000018h).

La rutina de EEPROM es como una especie de interrupción ya que obligamos al PC a moverse a otra dirección de memoria diferente a la del programa,

Por tanto, si entramos en rutina de EEPROM ya sea guardar o escribir datos, se guarda en el STACK la dirección de retorno al programa principal (donde se quedó antes de entrar a rutina EEPROM) y si en el proceso de EEPROM entra una interrupción, se reemplaza en STACK con una dirección de retorno de la rutina de EEPROM y por lo tanto se pierde la dirección de retorno del programa principal.

En nuestro caso, utilizamos el anterior concepto de modo inverso, el cual ha funcionado sin problema alguno; es decir, contamos con una interrupción por recepción de serial y

dentro de la rutina de interrupción se hace el manejo de la EEPROM, cuando el programa está en la rutina de la EEPROM no se interrumpe el programa, puesto que ya estamos dentro de una interrupción; una vez terminada la rutina de EEPROM, regresa a la función de interrupción y una vez terminada la interrupción, regresa al programa principal.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	<pre> void EEPROM_Guardar(int dir, char data){ EEADR = (unsigned char)dir; EEDATA = data; EECON1bits.EEPGD = 0; //Acceder a la memoria EEPROM EECON1bits.CFGS = 0; // = EECON1bits.WREN = 1; //write enabled //INTCONbits.GIE = 0; EECON2 = 0x55; EECON2 = 0x0AA; EECON1bits.WR = 1; //Inicializa la escritura //INTCONbits.GIE = 1; while(!PIR2bits.EEIF); PIR2bits.EEIF = 0; EECON1bits.WREN = 0; //Apagando la escritura } </pre>	<p>Función que guarda datos, se establece en que byte (dir) de 0-255 y el valor del byte a guardar (data).</p>
17 18 19 20 21 22 23	<pre> unsigned char EEPROM_Lectura(int dir){ EEADR = (unsigned char)dir; EECON1bits.EEPGD = 0; //Acceder a la memoria EEPROM EECON1bits.CFGS = 0; // = EECON1bits.RD = 1; //Inicializa la lectura return EEDATA; } </pre>	<p>Función de leer el dato en un byte especificado (dir) de la EEPROM</p>

4.5.3 Librería Serial.h

Librería desarrollada para la comunicación serial entre el microcontrolador y el XBEE.

1	<code>void serial_open(unsigned int baud){</code>	Función que habilita serial a los bauds deseados (baud)
2	<code> //RCSTA register as default is 0x00</code>	Valores por default en registros
3	<code> //TXSTA register as default is 0x02 --> TSR empty</code>	RCSTA (Recieve Status & Control),
4	<code> //BAUDCON as default is 0b0100 0x00</code>	TXSTA (Transmit Status & Control)
5	<code> //-----</code>	y BAUDCON (Baud Rate Control).
6	<code> //Libreria calculada para 8 Mhz Osc y 9600 bauds</code>	En estos comentarios se describe
7	<code> /* Calculando valor de SPBRG (variable x) para 8Mhz</code>	como calcular el valor del
8	<code> x = ((Foc / BR*64)-1);</code>	registro SPBRG en base a la
9	<code> x = (8000000/(9600*64))-1;</code>	frecuencia de trabajo del Micro y
10	<code> x = 12.02 --> x = SPBRG = 12;</code>	los Bauds con que se trabaja el
11	<code> BRcalculated = Fosc / (64*(n+1));</code>	serial, así como el error.
12	<code> BRcalculated = 9615.38 ;</code>	En base al valor del registro
13	<code> Error = (BRcal - 9600) / 9600;</code>	SPBRG el micro controla el
14	<code> error = 0.16 %;</code>	periodo de un timer (baud rate
15	<code>*/</code>	generator) que utiliza para leer
16	<code> //Conf para 9600 Bauds</code>	datos a los bauds seleccionados
17	<code> int bds;</code>	Se configura serial como modo
18	<code> TXSTAbits.SYNC = 0; //Modo Asincrono</code>	Asíncrono; solo se usan 8 bits
19	<code> BAUDCONbits.BRG16 = 0; //8 bits para BaudRateGenerator</code>	para el generador Baud Rate.
20	<code> TXSTAbits.BRGH = 1; //usando high speed</code>	Se configura la Recepción para
21	<code> SPBRGH = 0; // 8 bits para BRG, parte alta=0, reg de 2</code>	alta velocidad y por último se
22	<code> bytes</code>	calcula el SPBRG en base al
23	<code> SPBRG = (unsigned char)((((_XTAL_FREQ/ baud)/16)-1);</code>	cristal y la velocidad requerida
24	<code> TXSTAbits.TX9 = 0; //8 bits TX</code>	(baud, la parte alta SPBRGH = 0
25		debido a que solo se usan 8 bits
26	<code> //-----</code>	para el generador.
27	<code> RCSTAbits.RX9 = 0; //8 bits RX</code>	Se establece una recepción de 8
28	<code> RCSTAbits.CREN = 1; //Continuos RX</code>	bits, así como una recepción
29	<code> //-----</code>	continua.
30	<code> BAUDCONbits.RXDTP = 0; //Rx data polarity no inverted</code>	Polaridad de reloj y datos en RX
31	<code> BAUDCONbits.TXCKP = 0; //Clock polarity no inverted</code>	no invertidos.
32	<code> //-----</code>	Define pin RX y TX como entrada,
33	<code> TRISCbts.RC7 = 1;</code>	según data sheet (p 239) debe ser
34	<code> TRISCbts.RC6 = 1;</code>	así, pues el pic los cambia a
35	<code> RCSTAbits.SPEN = 1; //Habilita Serial port</code>	conveniencia.
36	<code> TXSTAbits.TXEN = 1; // TX enable</code>	Habilita puerto Serial y
37		Recepción (TX)
38	<code> //-----</code>	
39	<code>}</code>	Fin de función serial_open
40		
41	<code>void serial_close(void){</code>	Función que deshabilita el serial
42	<code> RCSTAbits.SPEN = 0; //Habilita Serial port</code>	
43	<code>}</code>	
44		
45	<code>char TX_busy(void){</code>	Función que revisa si el byte
46	<code> return (char) !TXSTAbits.TRMT; //0 TSR full; 1 TSR empty</code>	cargado al registro TXREG es
47	<code>}</code>	enviado. TRMT es el TSR (Transmit
		Shift Register Status), registro
		que se envía.

48 49 50 51 52 53 54 55 56	<pre> void serial_write(char word[]){ int i; for (i=0; word[i]!=(0x00); i++){ //search for end string character TXREG = word[i]; //load on TXREG to transmit while(TX_busy()); //wait until TSR sent the byte } } </pre>	<p>Función que envía una cadena de caracteres por serial, escribe byte por byte en el registro TXREG hasta encontrar el carácter de vacío (NULL/0x00) del arreglo word[]; luego espera hasta que el registro TSR (transmit shift register) se vacíe, pues el contenido en TXREG se copia a TSR (TRMT) que es el byte que se envía por serial. 0=TSR lleno; 1=TSR vacío.</p>
57 58 59 60 61 62 63 64	<pre> void clear_buffer(){ int i; char y; for (i=0;i<5;i++){ y=RCREG; } } </pre>	<p>Función que se encarga de limpiar el Buffer del serial, esto se hace leyendo el registro RCREG.</p>
65 66 67 68	<pre> //Revisa si hay datos en el serial. char RX_ready(void){ return (char)PIR1bits.RCIF; } </pre>	<p>Función que revisa si han llegado datos en el serial, revisa la bandera de recepción RCIF.</p>
69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86	<pre> void serial_read(char info[]) { int i = 0; while (1) { while (!RX_ready()); info[i] = RCREG; //PIR1bits.RCIF=0; if (info[i] != '_') { i++; } else { break; } if (i>5){ info = 0x00; break; } } } </pre>	<p>Función que lee una cadena de caracteres hasta encontrar el carácter "_", que es el fin de cadena. Cada carácter recibido se guarda en el array info si no encuentra "_"; si lo encuentra, rompe el ciclo while. También se colocó una restricción, si el índice es mayor a 5, la cadena no es válida, por lo que se escribe 0 en el array info y rompe el ciclo infinito.</p>

4.5.4 Librería config_reg.h

En esta librería se configuran los registros para los pines de entrada / salida, el timer utilizado para la inactividad, el oscilador interno e interrupción por recepción, etc

1	<code>void internal_clock()</code>	
2	<code>{</code>	
3	<code> OSCCONbits.IRCF0 = 1; //clock 8 Mhz</code>	
4	<code> OSCCONbits.IRCF1 = 1; //clock 8 Mhz</code>	
5	<code> OSCCONbits.IRCF2 = 1; //clock 8 Mhz</code>	
6	<code> OSCCONbits.SCS1 = 1; //internal Osc</code>	Función que configura el uso del reloj interno a 8 Mhz
7	<code>}</code>	
8	<code>void config_reg(){</code>	
9	<code> //Puertos Analogicos como Digitales</code>	
10	<code> //PORT A,B,E,</code>	
11	<code> ADCON1=0x0F;</code>	
12	<code> //puerto A0 entrada de sensores: PIR</code>	
13	<code> TRISA = 0x01;</code>	
14	<code> //PORTB como salida</code>	
15	<code> TRISB = 0x00;</code>	
16	<code> //led sensor PIR</code>	
17	<code> TRISCbits.RC0 = 0;</code>	
18	<code> //PORTD entrada, boton para cambio manual</code>	
19	<code> TRISD = 0xFF;</code>	
20	<code> //Conf Timer</code>	
21	<code> //TMRO = 65535 - (T * Fos/(4*Preescaler))</code>	
22	<code> TOCONbits.T08BIT = 0; //Timer de 16 bits</code>	
23	<code> TOCONbits.PSA = 0; //preescaler activado</code>	
24	<code> TOCONbits.T0CS = 0; //reloj int para timer</code>	
25	<code> TOCONbits.T0PS = 7; //preescaler para 256</code>	
26	<code> // 5 Seg para 8 Mhz</code>	
27	<code> //TMRO = 65535 - ((5seg * 8Mhz)/(4*256))</code>	
28	<code> TMR0 = 26472; //TMR0 para timer de 5 seg</code>	
29	<code> TOCONbits.TMR0ON = 0; //deshabilita timer</code>	
30	<code> INTCONbits.TMR0IE = 0; //deshabiita inter timer</code>	
31	<code> INTCONbits.TMR0IF = 0; //bandera de inter timer</code>	
32	<code>}</code>	Fin de función config_reg()
33	<code>void RX_interrupt_enabled(){</code>	
34	<code> //Enable global Interrupts</code>	
35	<code> INTCONbits.GIE = 1; //enable global interr</code>	
36	<code> INTCONbits.PEIE = 1; //enable peripheral interrupt</code>	
37	<code> RCONbits.IPEN = 0; //prioridad off</code>	
38	<code> PIR1bits.RCIF = 0; //RX interrupt flag</code>	
39	<code> PIR1bits.RCIE = 1; //Enable RX interrupt</code>	
40	<code> RCSTAbits.CREN = 1; //Continuos RX</code>	
41	<code>}</code>	

4.5.5 Programación de Software, estación central.

El software desarrollado se programó en Labview, versión 2017. El programa se compone de 1 programa principal (Monitor) y varios subprogramas, como es el ejecutor de tareas, programador de tareas y el que establece servicios de manera manual. En la siguiente imagen se observa la pantalla del monitor:

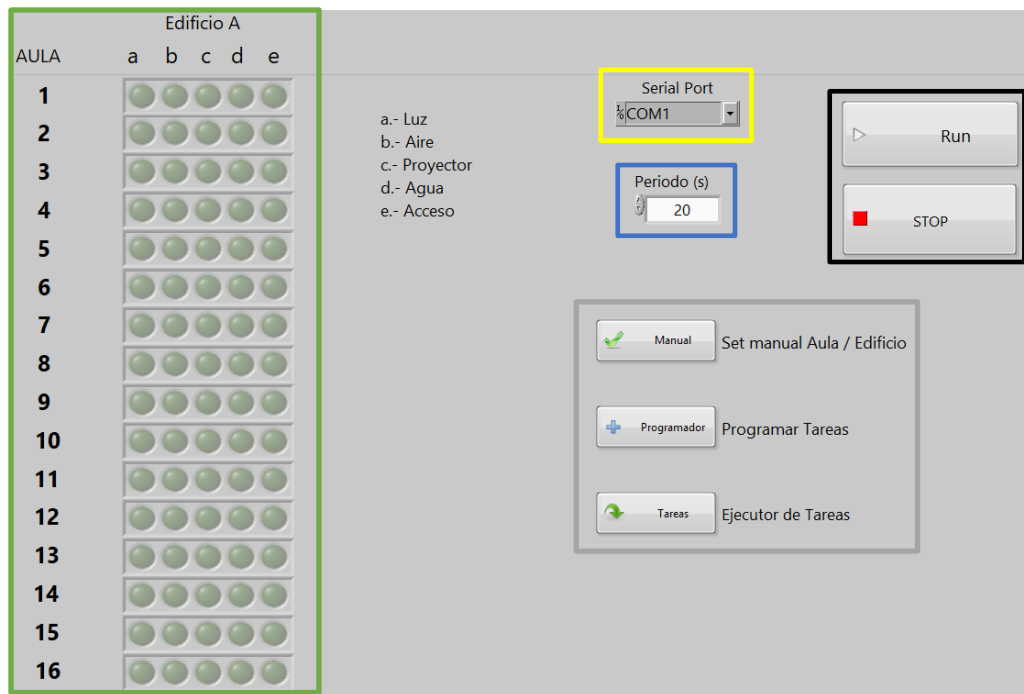


Figura 41 Pantalla de monitor

En el rectángulo negro, se encuentran los botones que iniciar o parar el programa. En la sección del rectángulo en naranja es donde se observan el status de los servicios (a.- Luz, b.-Aire, c.-Proyector, d.-Agua, e.-Acceso) de cada una de las aulas (1-16).

En el rectángulo amarillo, seleccionamos el puerto COM en el que se encuentra conectado el XBEE.

En el rectángulo azul, seleccionamos el periodo de muestreo de las aulas, es decir cada cuanto tiempo se revisa el status de cada aula.

El rectángulo verde se encuentran los botones para abrir los diversos subprogramas, en el caso del botón "Ejecutor de tareas", al presionar el botón lo que hace el programa es cambiar de pantalla, puesto que el ejecutor está en ejecución en background del monitor y se tiene por separado para Aulas y Edificios, a continuación en la siguiente figura se muestra su pantalla y se describen sus componentes:

Ejecutor de Tareas 09:41:12 p. m.

Tareas por Edificio

ID	Fecha Registro	Inicio	Hora_Inicio	Fin	Hora_Fin	Edificio	Aula	Serv	Dias	Notas
217	27/10/2019 02:44:00 p. m.	25/10/2019	01:43:00 p. m.	31/12/2019	02:43:00 p. m.	A	5 - 10	- e	Ma Mi Ju Vi Sa Do	
216	27/10/2019 02:44:00 p. m.	25/10/2019	01:43:00 p. m.	31/12/2019	02:43:00 p. m.	A	5 - 10	+ a e	Vi Sa Do	
215	27/10/2019 02:43:00 p. m.	25/10/2019	01:43:00 p. m.	31/12/2019	02:43:00 p. m.	A	5 - 10	a b c d e	Vi Sa Do	
214	27/10/2019 02:43:00 p. m.	25/10/2019	03:43:00 p. m.	31/12/2019	02:43:00 p. m.	A	5 - 10	c d e	Vi Sa Do	
193	29/09/2019 04:53:00 p. m.	29/09/2019	08:25:40 p. m.	31/12/2019	04:52:00 p. m.	A	10 - 14	c d	Lu Ma Sa Do	
194	29/09/2019 04:53:00 p. m.	29/09/2019	08:25:45 p. m.	31/12/2019	04:52:00 p. m.	A	10 - 14	+ e	Lu Ma Sa Do	
195	29/09/2019 04:53:00 p. m.	29/09/2019	08:25:50 p. m.	31/12/2019	04:52:00 p. m.	A	10 - 14	- d	Lu Ma Sa Do	
196	29/09/2019 04:53:00 p. m.	29/09/2019	09:25:55 p. m.	31/12/2019	04:52:00 p. m.	A	10 - 14		Lu Ma Sa Do	
209	06/10/2019 04:47:00 p. m.	04/10/2019	09:28:00 p. m.	31/12/2019	04:46:00 p. m.	A	8 - 14	a b c d e	Lu Ma Mi Ju Vi Sa Do	
210	06/10/2019 04:47:00 p. m.	04/10/2019	09:28:10 p. m.	31/12/2019	04:46:00 p. m.	A	8 - 14	- a b	Lu Ma Mi Ju Vi Sa Do	
211	06/10/2019 04:47:00 p. m.	04/10/2019	09:28:20 p. m.	31/12/2019	04:46:00 p. m.	A	8 - 14	- c d	Lu Ma Mi Ju Vi Sa Do	
212	06/10/2019 04:47:00 p. m.	04/10/2019	09:28:30 p. m.	31/12/2019	04:46:00 p. m.	A	8 - 14	+ a	Lu Ma Mi Ju Vi Sa Do	
213	06/10/2019 04:48:00 p. m.	04/10/2019	09:28:40 p. m.	31/12/2019	04:46:00 p. m.	A	8 - 14	+ c	Lu Ma Mi Ju Vi Sa Do	
218	27/10/2019 09:40:00 p. m.	01/10/2019	10:39:00 p. m.	31/12/2019	09:39:00 p. m.	A		c d	Lu Ma Sa Do	

Tarea por Aula

ID	Fecha Registro	Inicio	Hora_Inicio	Fin	Hora_Fin	Edificio	Aula	Serv	Dias	Notas
112	31/08/2019 01:57:00 p. m.	29/08/2019	07:00:00 a. m.	30/12/2019	09:00:00 p. m.	E	15	a d e	Lu Ma Mi Ju Vi Sa Do	
160	29/09/2019 09:35:00 p. m.	26/09/2019	10:00:00 p. m.	29/11/2019	09:34:00 p. m.	A	9	- e	Lu Ma Sa Do	
159	29/09/2019 09:35:00 p. m.	26/09/2019	10:00:00 p. m.	29/11/2019	09:34:00 p. m.	A	9	+ e	Lu Ma Sa Do	
158	29/09/2019 09:34:00 p. m.	26/09/2019	10:00:00 p. m.	29/11/2019	09:34:00 p. m.	A	9	c d e	Lu Ma Sa Do	

Home

Programar Tarea

ID a ejecutar

218

●

ID a ejecutar

160

●

Servicios:

- a Luz
- b Aire
- c Proyector
- d Agua
- e Acceso

Figura 42 Pantalla de Ejecutor de Tareas

En el rectángulo azul se muestra la hora; tanto en el rectángulo amarillo y verde, se observan las tareas que se ejecutarán en el día, labview lee una base de datos en donde se guardan las tareas, y se encarga de filtrar las que se ejecutarán el día en curso así como ordenarlas de acuerdo a la hora de ejecución (columna hora_inicio), el rectángulo amarillo pertenece a las tareas por Edificio mientras que el verde a las tareas por aulas.

El ambas tablas se puede observar la información completa de cada tarea, el número de tarea (ID), fecha de registro, fecha de inicio así como la hora de inicio (ejecución), la fecha en la que la tareas dejará de ejecutarse junto con su hora (Fin, hora_fin), el Edificio y el o las aulas en caso de haber un rango, los servicios y tipo de tarea, ya sea de reemplazo, añadir (+) o eliminar (-) servicios, los días en los que se ejecutará dicha tareas y notas.

En los rectángulos de color naranja, se observa la tarea próxima a ejecutarse, tanto para aulas como edificio; el led se enciende cuando la tarea está en ejecución.

En el rectángulo negro, están los botones de inicio (Home) y programador de tareas, el primero nos lleva de regreso a la pantalla de monitor; mientras que el segundo abre el subprograma "programador de tareas".

Por último, en el lado derecho se observa las etiquetas de cada servicio, referentes en la columna Serv, donde "a" representa Luz, "b" Aire, "c" Proyector, "d" Agua y "e" Acceso.

El Block Diagram del programa principal, se compone de 4 ciclos while que trabajan "en paralelo"; 1 se encarga de revisar si se apretó algún botón de los subprogramas (Programador de tareas y Establecer Manual) o para revisar el ejecutor de tareas; otro ciclo while se encarga del monitoreo de las aulas, es decir, según el periodo seleccionado, éste envía solicitud de status a cada una de las aulas, los status recibidos se despliegan en la pantalla de monitor.

Los otros dos ciclos, se encargan de la ejecución de las tareas programadas, uno es para Aulas y el otro para Edificios.

A continuación, se muestra cada uno de los ciclos por separado:

4.5.6 Ciclo de Sub-programas

Este ciclo se encarga de revisar si se apretó algún botón para abrir un subprograma, así como también de iniciar el ejecutor de tareas de manera automática, ver figura 43.

La sección naranja sirve para iniciar el ejecutor de tareas a la hora especificada, cuando el programa termina con la ejecución de todas las tareas del día, el ejecutor se detiene, tanto para aulas y edificio, e inicia ejecución automática a la hora especificada.

En la sección amarilla, ejecuta la acción de moverse de pantalla, cuando se presiona el botón "Ejecutor de Tareas" en el monitor o cuando se presiona "Home" cuando se está en la pantalla de ejecutor de tareas.

En la sección azul y verde se inspecciona los botones "Programador de Tarea" y "Set Manual Aula/Edificio"; esto nos permite abrir el subprograma correspondiente. En la sección azul, se observa que se puede ejecutar el programa desde 2 botones distintos, esto porque tenemos 1 botón en la pantalla de monitor y otro en la pantalla de ejecutor de tareas. Se observa también que deshabilitamos una variable global (variable "Boolean") en el botón "Set Manual Aula/Edificio", ésta variable nos permite deshabilitar el ciclo de monitor para liberar el puerto serial y dejarlo libre para el subprograma. El programador de tareas no utiliza el puerto serial, únicamente una conexión a una base de datos.

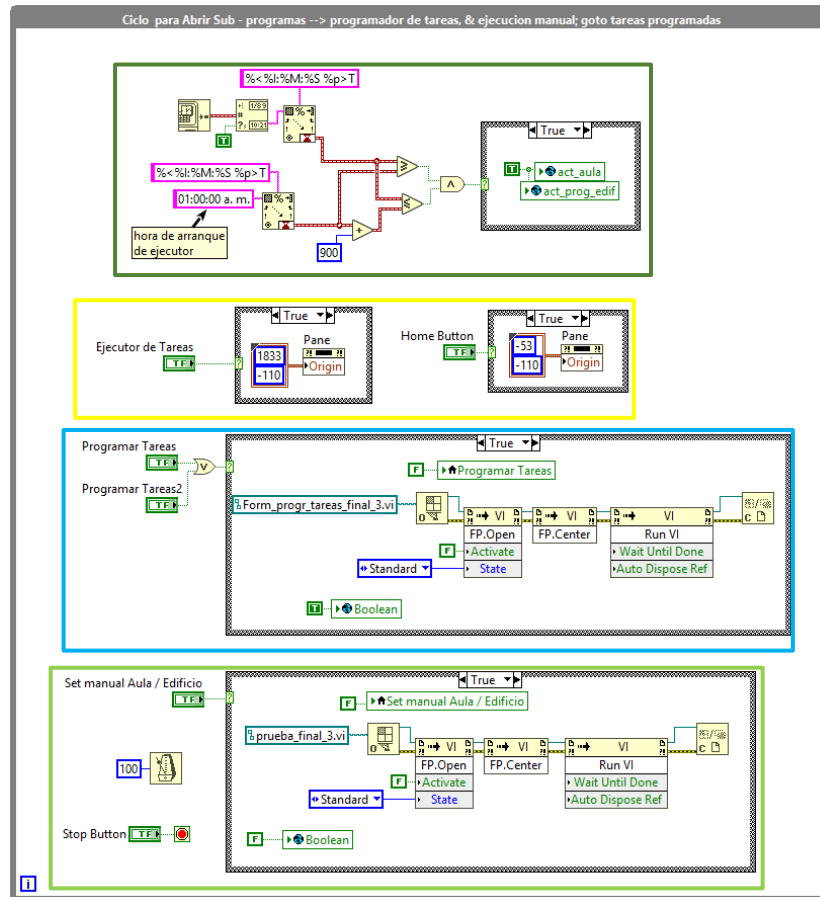


Figura 43 Ciclo Sub Programas

4.5.7 Ciclo Monitor

Este ciclo se encarga de desplegar en los leds de la pantalla de monitor los servicios activos en cada aula, por lo que envía, recibe y decodifica cadena de caracteres. A continuación, se explica por secciones de acuerdo a la figura 44.

1. *Puerto COM.* Se selecciona el puerto al que estará conectado el XBEE, su contraparte en el front panel es el rectángulo amarillo en figura 1. Por default tiene una velocidad de 9600 bps, 8 bits de datos y sin paridad; el puerto COM seleccionado, se carga a una variable global llamada "VISA resource name", ésta nos sirve para llevar el mismo puerto COM cuando se utiliza un sub programa.
2. *Periodo de Muestreo.* Corresponde al rectángulo azul de la pantalla de la figura 1. Cuando el ciclo inicia, $i = 0$, por lo que se ejecuta todo el ciclo la primera vez; cuando $i \neq 0$ la ejecución se realiza cada periodo de muestreo, esto se realiza con la función "Elapsed Time" y el periodo seleccionado. Esta función, está conectada

a una compuerta AND con una variable global tipo boole. Esta variable es la misma que se observa en la figura 3 (rectángulo azul y verde) y nos permite controlar a su vez la activación o desactivación del ciclo de monitoreo por medio de una estructura "Case", aparece en diversas secciones de la figura 4, para detener cualquier ciclo en el que se encuentre. Cuando un sub programa entra en ejecución el ciclo de monitoreo se desactiva, con la intención de liberar el puerto COM y que sea utilizado por el correspondiente subprograma.

3. *For por Edificio.*- Este ciclo se agregó para ejecutar el monitor en diversos edificios, actualmente está configurado para un Edificio A; con ayuda de la variable i de éste ciclo, se puede hacer un barrido de los edificio necesarios.
4. *Solicitud de Estatus.*- Las secciones siguientes se encuentran dentro del while "While para aulas 1-16". El ciclo for etiquetado como "For para escribir 2 veces solicitud de status y lectura de status", como su nombre lo dice, envía y recibe 2 veces la misma palabra de status, esto se hace debido al problema de concatenación descrito anteriormente. En ésta sección se crea la palabra para solicitar status, con base al Edificio, Aula, carácter de status "s" y carácter de fin de cadena "_". El número de aula se va actualizando después de cada ejecución del while, gracias a un shift register. La condición de paro del while está condicionada por el número de aulas, por el botón de paro y por la variable boole, que cambia a True, cuando se abre un subprograma.
5. *Lectura de Serial.*- Primero se verifica si existe algún byte en el puerto serial, en caso de ser >0, se activa la estructura Case "Lee datos de puerto y desgloza Edif - Aula" y lee los datos del puerto, al mismo tiempo se separa la cadena, cuando se encuentra el carácter "?" que nos indica fin de cadena que el microcontrolador envía, verificamos que la longitud de la cadena de caracteres antes de "?" corresponda a una cadena valida, puesto que sabemos que es de 4 bytes, si esto es verdad, habilita la estructura Case "long de cadena valida".
6. *Extracción de caracteres.*- En ésta sección se extrae cada byte de la cadena de caracteres, el primero nos indica el Edificio cuyo valor se inyecta a una estructura Case "Detecta Edificio", como únicamente tenemos edificio A, solo contiene este caso, pero puede expandirse a edificios necesarios. Posteriormente se extrae el

segundo, tercer y cuarto carácter; el segundo que es el número de aula, se inyecta un Case en cuyo cada caso es un numero de aula. El tercer carácter es de servicios, este valor se propaga al case de numero de aula, se conecta a cada uno de los leds de monitoreo del panel frontal. Por último el 4º carácter nos indica inactividad, el valor extraído también se programa en el Case de numero de Aula, y se conecta a un led en el que parpadea y se hace visible cuando el valor es 1, y se hace invisible cuando es 0, este led aparece en el lado izquierdo de los leds de monitoreo parpadeando y es asociado al sensor de presencia PIR.}

7. *Escritura en base de datos.* En ésta parte se arman los datos a guardar en la base datos que registra cada periodo de muestreo, primero agregamos un número ID, que viene de una variable global "log_count" guardada en un archivo, después fecha y hora, enseguida Edificio, Aula y por último el Estatus de los servicios; aquí observamos que usamos un sub programa (subVi_extr_serv), éste nos ayuda para codificar el servicio como una letra (a-d), con la intención de ser comprensible cuando se guarda en la base de datos. Una vez teniendo los datos a guardar, se escriben los datos en la base de datos "logs".
8. *Guardar ID en archivo.-* Aquí se guarda en archivo la variable "log_count", ésta nos sirve para identificar con un ID cada línea del logger; cuando se inicia el programa carga el ultimo valor que se guardó en la base de datos, de esta forma no se pierde la secuencia.
9. *Case de Aulas.-* Cada caso corresponde a cada una de las aulas (1-16) y cada caso contiene leds de inactividad y servicios correspondientes. Para activar o desactivar dichos leds, se observa que están alimentados por la extracción de datos en 6.

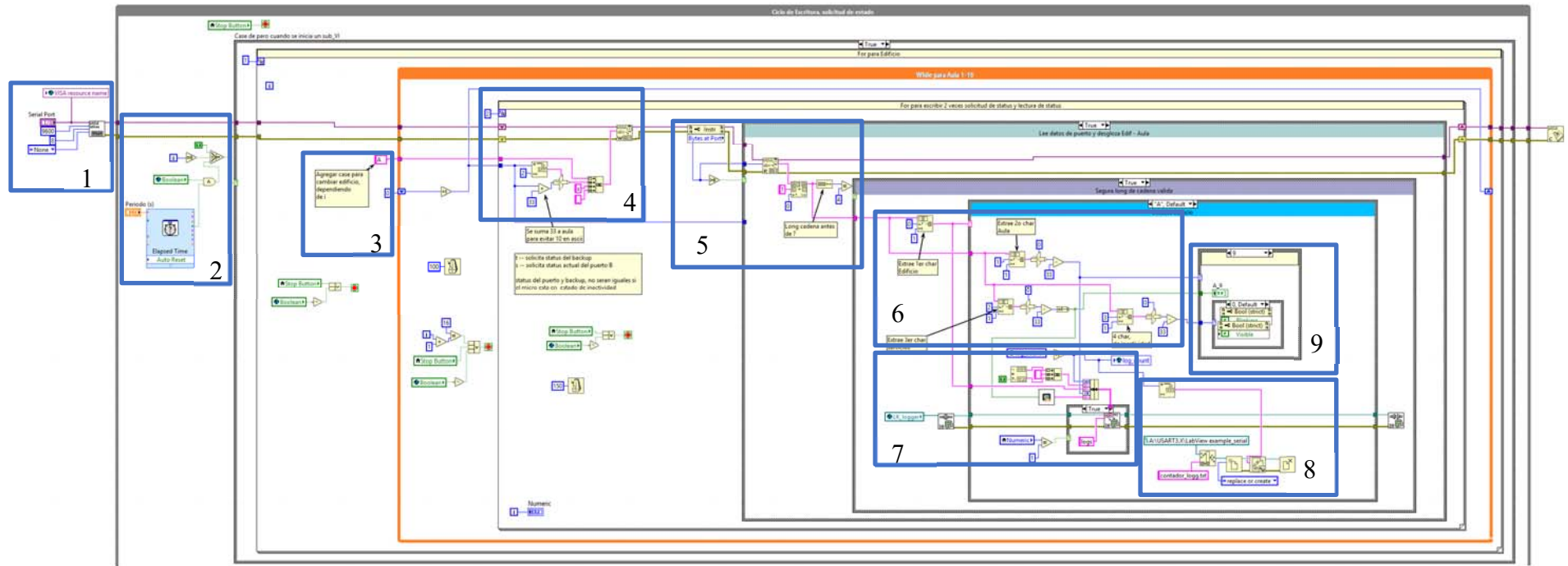


Figura 44 Ciclo Monitor

Es importante mencionar que las estructuras Case etiquetada como “Leer datos de puerto y desgloza Edif-Aula”, tiene el siguiente diagrama:

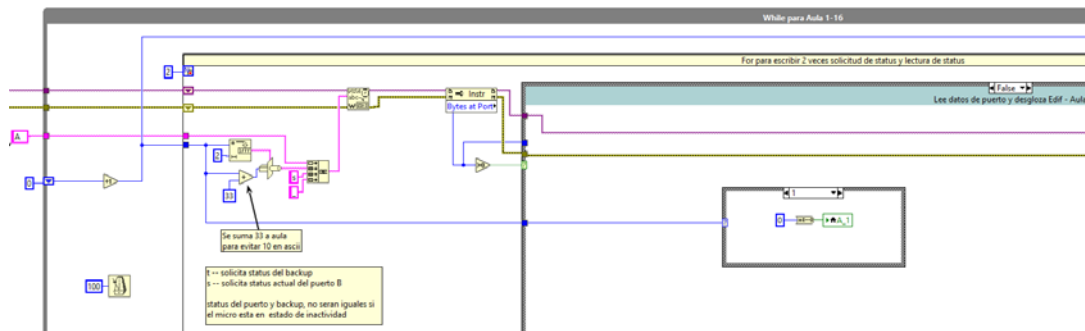


Figura 45 Case False “Leer datos de puerto y desgloza Edif-Aula”

La función del case FALSE, es desplegar 0 en el led de monitor; esto pasa cuando no se recibe respuesta de alguna de las aulas, con lo que puede diagnosticarse como una falla en la tarjeta del aula o en la comunicación.

La estructura case “segura long de cadena”, en su caso FALSE, se encuentra vacia, pues al no cumplir con la longitud de cadena valida, no realiza acción alguna.

En 7, hicimos uso del sub programa subVi_extr_serv, a continuación, se muestra el panel frontal y el diagrama.

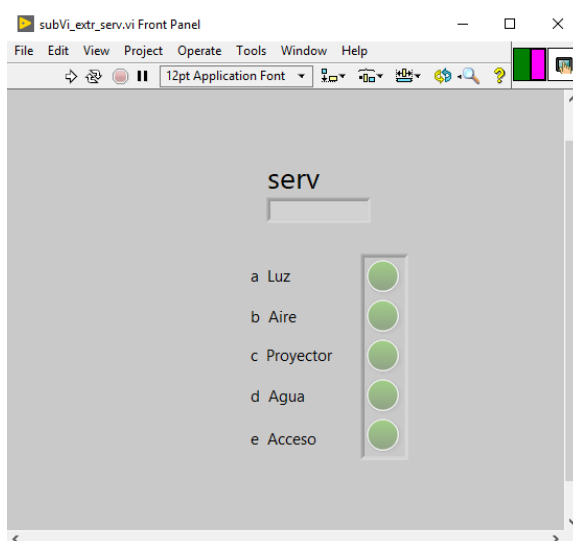


Figura 46 panel frontal subVi_extr_serv

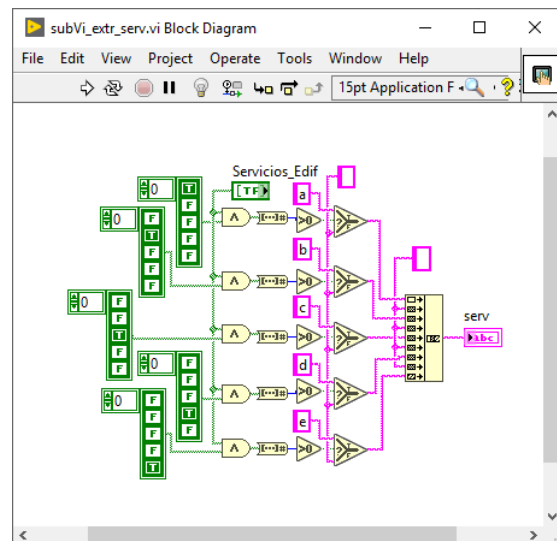


Figura 47 Diagrama de subVi_extr_serv

Como dato de entrada para éste subVi, es un array de booleanos, que en nuestro caso son los servicios y como salida se tiene un string, que es producido por la concatenación de letras a-e.

En el diagrama de bloques se observa el funcionamiento, el array que se recibe pasa por una compuerta AND, y dependiendo de la posición da como resultado 1 o 0, si es 1 concatena la letra correspondiente a la posición y si es 0 concatena un espacio en blanco.

Como resultado tenemos una concatenación de letras y espacios, éstos se transfieren a la base de datos, cada letra describe el siguiente servicio:

letra	Servicio
a	Luz
b	Aire
c	Proyector
d	Agua
e	Acceso

Tabla 8 Servicios

En la siguiente imagen se muestra la inicialización de variables, asociadas al ciclo de monitor, éstas variables se inicializan fuera del ciclo while, por lo que el programa la ejecuta primero y una sola vez.

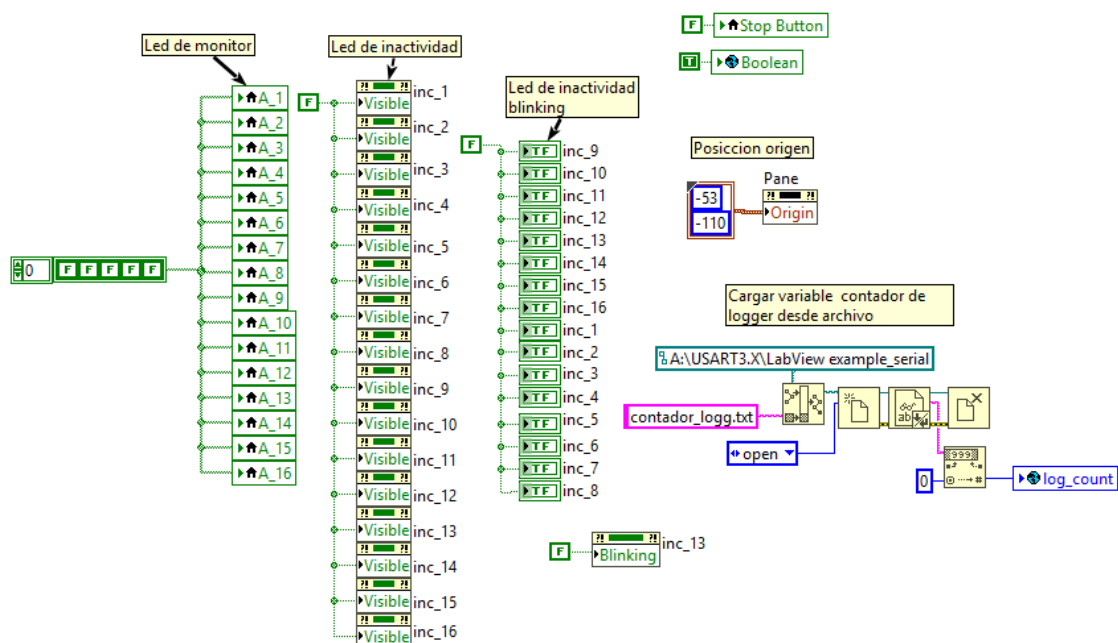


Figura 48 Inicialización de variables asociadas al ciclo monitor

Como se observa en el diagrama, los leds que vemos en la pantalla de monitor, siempre inician con una valor FALSE; los leds que señalan inactividad inician ocultos y con un valor FALSE también.

También se incluye la variable global "Boolean" como TRUE, recordemos que ésta nos ayuda a desactivar/activar el ciclo de monitor, por lo que al inicio permanece activado.

El botón de STOP lo cargamos con FALSE, puesto que cuando detenemos el programa se queda activado, la función está configurada de ésta manera, debido a que éste botón detiene varios ciclos while/for.

La posición inicial de la pantalla de monitor está dada por las coordenadas -53, -110 .

Y por último cargamos la variable "log_count" desde un archivo; ésta variable nos sirve para identificar con un valor único cada línea del logger.

4.5.8 Ciclo Ejecutor de Tareas por Edificio

Como se observa en el diagrama de la figura 49, éste ciclo contiene un Estructura Case, por medio de la variable global “act_prog_edif”, controlamos la ejecución del diagrama que contiene; puesto cuando una lista de tareas llega a su fin, ponemos en FALSE la variable y así evitamos una ejecución redundante; por lo tanto cuando el ejecutor se inicia la variable se coloca en TRUE.

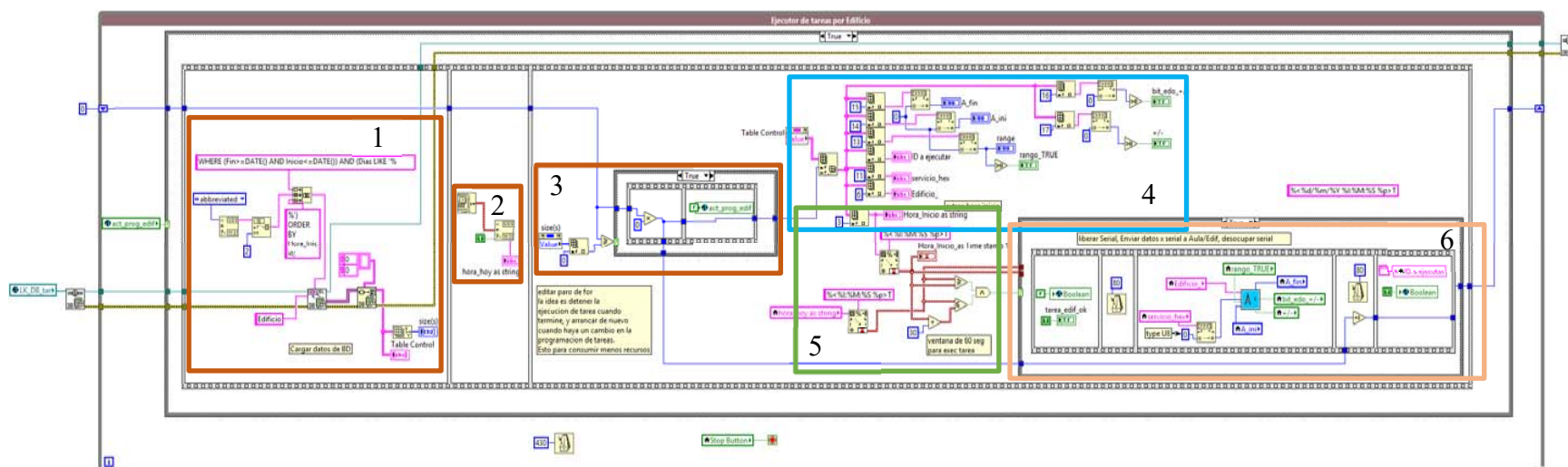


Figura 49 Diagrama de Bloque de ciclo Ejecutor de Tareas por Edificio

Dentro del Case se tiene una estructura del tipo “Flat Sequence”, con la intención de ejecutar de manera secuencial el diagrama. Al igual que el ciclo anterior, se describe por secciones:

1. *Lectura de Base de Datos.*- En ésta sección se cargan las tareas que se encuentran en la base de datos, es importante mencionar que se utiliza un query, con el objetivo de cargar únicamente aquellas tareas que se tienen que ejecutar el día en curso y cuya fecha de finalización no haya pasado; al mismo tiempo las tareas son ordenadas por la hora de inicio y se despliegan en la tabla de "Tareas por Edificio" que aparece en la pantalla de Ejecutor de Tareas (figura 42), también guardamos en la variable "size" el tamaño de la tabla (filas, columnas).
2. *Fecha y hora.*- Se despliega la fecha y hora de hoy en la pantalla de Ejecutor de Tareas (figura 42, rectángulo azul).
3. *Índice de tabla.*- Como se observa en el diagrama, esta sección es alimentada por un "shift register", el cual nos permite movernos de fila en fila en la tabla "Tareas por Edificio" (figura 42); cada vez que hay una ejecución del ciclo, el shift register aumenta en 1, lo que significa moverse a la siguiente fila de la tabla. Se hace una comparación para verificar que shift register no sea mayor al número de filas de la tabla; en el caso que sea igual o mayor, el shift register se multiplica por 0 (reinicia el shift register) y la variable "act_prog_edif" se carga con FALSE (desactivando la ejecución de tareas), esto significa que ya no hay tareas por ejecutar puesto que se recorrió toda la tabla. En caso de ser menor, el numero cargado en el shift register se pasa a las siguientes secciones.
4. *Extracción de datos.*- Se extrae de la tabla los datos necesarios para la ejecución de las tareas, se cargan a variables, como son Edificio, hora de inicio, bit de rango, si hay rango inicio y fin de rango, bit de sumar o restar estados. En el programador de tareas se explican a detalle los campos de la base de datos.
5. *Hora de ejecución.*- Leyendo la hora de inicio de la tarea a ejecutar, ésta entra a una serie de comparadores de hora; cuando la hora actual del sistema es mayor o igual a la hora de inicio, inicia la ejecución de la tarea. Por otro lado establecemos un margen/ventana de ejecución, en este caso 30 segundos, esto nos sirve cuando hay varias tareas a ejecutarse a la misma hora, y si el tiempo de ejecución de una tarea se extiende de tal forma que el tiempo de inicio de la siguiente tarea se haya pasado, esto permite aun así ejecutar la siguiente tarea. Recordemos que la condición de paro es el índice de la tabla.
6. *Ejecución de tarea.*- Una vez que la hora de ejecución es cumplida, se dispara un Case en el que se encuentra una estructura "flat sequence"; en la primera sección se carga FALSE a la variable global "Boolean", con esto desactivamos el proceso

de Monitor, al mismo tiempo prendemos el Led que se encuentra en la pantalla Ejecutor de Tareas (figura 42, rectángulo naranja), a un costado de la tabla correspondiente. En la segunda y cuarta sección, son retardos. En la tercera se cargan las variables leídas en la sección 4 al sub_programa “Sub_vi_ejecutar_Edificio_”, las variables cargadas son Edificio, Servicio, Aula inicio (A_ini), Aula final (A_fin), bit de rango, bits de suma o resta de estados. En la quinta y última sección del “flat sequence”, se pone en TRUE la variable “Boolean”, por lo que activamos nuevamente el proceso de monitoreo.

4.5.9 Inicialización de variables

Como inicialización, fuera del ciclo While, se aplica TRUE a la variable “act_prog_edif”, con lo que activa el CASE dentro del ciclo while, permitiendo la ejecución del diagrama desde que el programa arranca. También carga un FALSE la variable “tarea_edif_ok”, que representa el led de ejecución del panel frontal (figura 2, cuadro naranja).



Figura 50 Variables de inicialización de ciclo Ejecutor de Tareas por Edificio

4.5.10 Sub Programa “Sub_vi_ejecutar_Edificio”

Se identifica a dicho sub programa por el ícono de la figura 49; en la figura 51 se muestra el panel frontal.

Aunque el panel frontal no se utiliza como tal en el programa principal (monitor), las variables (datos) en el panel frontal (figura 51), están asociadas como entrada de datos extraídos de la base de datos, ver figura 49 sección 6.

En la figura 52 se muestra el ícono asociado a éste sub programa.

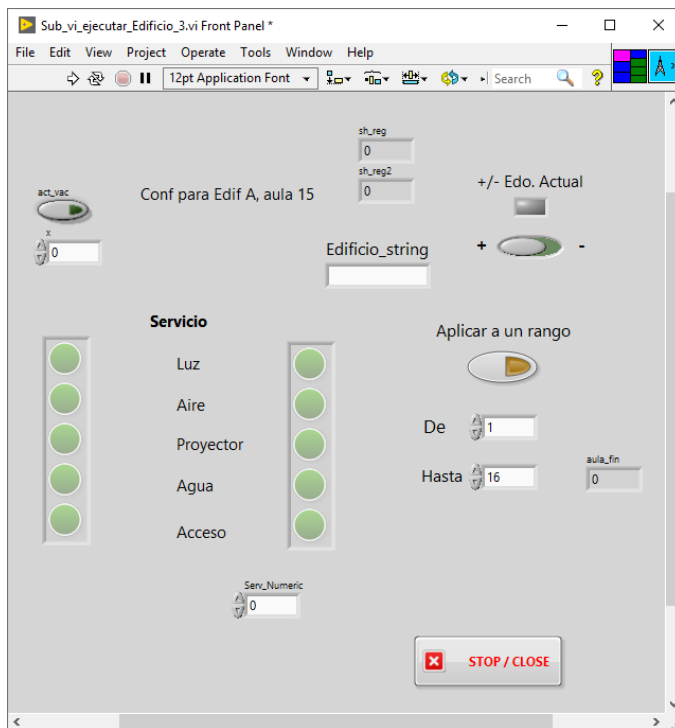


Figura 52 Panel frontal de Sub_vi_ejecutar_Edificio



Figura 51 ícono de sub_vi_ejecutar_edificio

En la siguiente figura se muestra el diagrama de funcionamiento, al igual que los diagramas anteriores, se explica en secciones.

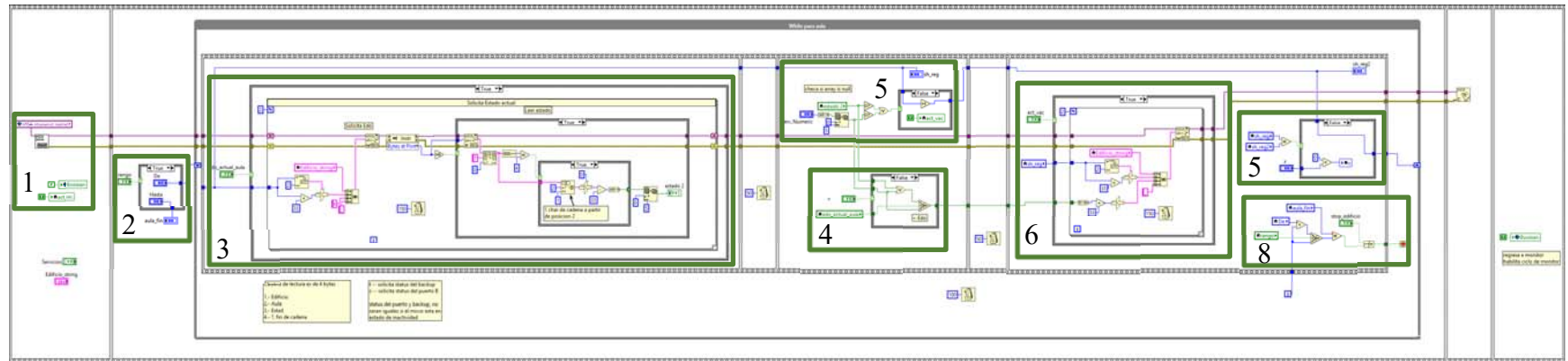


Figura 53 Diagrama a bloques de "Sub_vi_ejecutar_Edificio"

El diagrama está compuesto de una estructura "flat sequence", un ciclo while y una segunda estructura "flat sequence", el while nos sirve para iterar el número de cada aula. A grandes rasgos, el programa revisa si hay nuevo estatus de servicio o si se trata de sumar o restar servicios al estado de los servicios actuales; por lo que se envía un requerimiento para leer estatus hacia el microcontrolador (en caso de sumar o restar status), hace las comparaciones necesarias y por último envía el status resultante, que a su vez se verá reflejado en el PORTB del microcontrolador; a continuación, se describe con mayor detalle:

1. *Puerto COM.*- Se carga la variable global "VISA resource name", ésta nos permite acceder al puerto COM establecido desde el programa de monitor (panel front de figura 41, rectángulo amarillo). La variable global "Boolean" se pone en FALSE para desactivar el ciclo de monitor y liberar el puerto COM para la ejecución de la tarea en curso. Así mismo se escribe TRUE en la variable "act_vac", que nos sirve para activar el envío del nuevo status y para prevenir algún error en la lectura del status, ver sección 5.
2. *Bit de rango.*- Si es TRUE, el rango se define como la variable A_ini y A_fin de la base de datos, que indica aula de inicio y final respectivamente; en caso de ser FALSE, se toma como inicio 1 y fin 16. La variable de inicio se carga al shift register. En la siguiente figura se muestra el case FALSE.

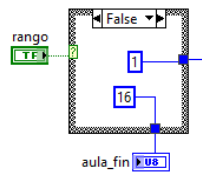


Figura 54 Case FALSE de bit de rango

3. *Bit suma/resta de estado.*- Se verifica la variable "edo_actual_aula" que está asociada a un bit que nos indica si el nuevo estado (leído de la base de datos) es para sumar/restar el estado almacenado al estado actual. En caso de ser TRUE, se solicita una lectura de estatus del PORTB del aula en curso, se solicita estatus con "t" indicándole al microcontrolador que la solicitud viene de un subprograma; el resultado de la lectura se guarda en el array "estado 2". En caso de ser FALSE, no se realiza ninguna lectura, esto significa que el mismo estado leído de la base de datos, se va a enviar como salida hacia el puerto B del microcontrolador. En la figura 14, se observa el case cuando es FALSE.

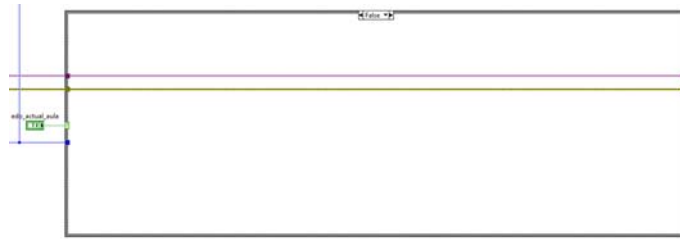


Figura 55 Case FALSE de bit de rango

4. *Estado final*.- En esta estructura CASE se define el estado final que se enviará por el puerto serial hacia el / las aulas. Como variable de comparación está el bit de sumar/restar, y dentro de cada CASE ya sea FALSE/TRUE hay un SELECT que depende de la variable "edo_actual_aula", esta variable nos indica si vamos a sumar/restar estado o enviar el estado leído como un nuevo estado hacia las aulas. En la figura 53 se observa el caso FALSE, indicando que se van a sumar los estados; ésta operación es el resultado de una compuerta OR entre el estado leído de la base de datos (variable "Serv_Numeric") y el estado actual del aula (variable "estado 2"). En la figura 56 se observa el estado TRUE. En el caso TRUE significa que vamos a restar los servicios leído de la base de datos al estado actual en el aula. El estado final de la resta es el resultado de una compuerta AND entre el estado actual del aula (variable "estado 2") y el NOT (para invertir estados) del estado leído de la base de datos (variable "Serv_Numeric"). Como anteriormente se mencionó, si el caso es TRUE/FALSE, el resultado también depende del bit "edo_actual_aula", si esta variable es FALSE independientemente del caso TRUE/FALSE, el estado final es el estado leído desde la base de datos, lo que significa que es un nuevo status para el aula, es decir no hay sumas ni restas de estados.

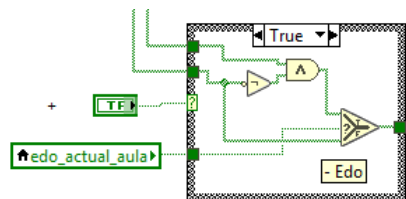


Figura 56 Case FALSE de bit de rango

5. *Detección de vacío.*- En algunas ocasiones durante las pruebas del programa la lectura de status del microcontrolador llegaba un array en vacío, problema que atribuimos al hecho del uso del puerto COM que de alguna manera es compartido, por lo tanto se hizo este diagrama para obtener una nueva lectura en caso de que se lea un arreglo vacío, en la figura 53 se observa cuando la condición es false, es decir hay un dato válido, por lo tanto la variable "act_vac" se pone en TRUE permitiendo el envío del status final (sección 6) calculado en la sección 4 y al mismo tiempo incrementa el valor del shift register, con el que en la siguiente iteración se evalúe para el aula siguiente. En caso de ser TRUE, significa que hubo un error en la solicitud de status o lectura de la base de datos, la variable "act_vac" se pone el FALSE y por tanto no se permite el envío del status calculado, el valor del shift register se pasa igual, con lo que la siguiente iteración del ciclo while, se ejecuta nuevamente el ciclo para la misma aula. En caso de detectar el mismo error 3 veces, se omite el aula, se incrementa el shift register y por lo tanto el programa se mueve hacia el aula siguiente (sección 7)

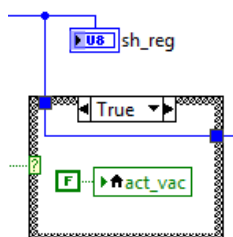


Figura 57 Case TRUE detección vacío

6. *Envío de Status.*- El envío de status procedente de la sección 4, está condicionado por la variable que detecta error de vacío "act_vac", en caso de no existir error, la variable es cargada con TRUE, por lo que se permite el envío del status, si es FALSE, simplemente no hace nada. El número de aula se envía de acuerdo al valor cargado en el shift register, en este caso es la variable "sh_reg". Por lo tanto se concatena en una misma palabra el edificio, aula, status calculado y el carácter "_".
7. *Persistencia de error.*- Cuando se detecta un vacío, se considera un error suma a una variable contador "x", por lo que éste case es una condición para incrementar el shift register o dejar igual, con la intención de repetir la iteración para la misma aula. En la figura 53 se observa el caso en que no existe error por lo que el valor del shift register se pasa y en la siguiente iteración corre el ciclo para la siguiente aula.

En la figura 58, se observa el caso en que hay un error; recordemos que cuando hay error, no hay envío de status, pues la variable "act_vac" se establece en FALSE y el valor del shift register no cambia; por lo tanto cuando hay un error, la variable x, que es un contador, incrementa su valor, si el número de errores es menor a 3, se ejecuta todo el ciclo nuevamente para la misma aula, cuando el contador llegue a 3, automáticamente se descarta el aula actual y se incrementa en 1 el valor del shift register.

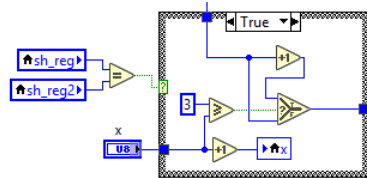


Figura 58 Case TRUE detección vacío

8. *Condición de paro.* - La condición para detener el ciclo está condicionado por el botón stop o cuando el valor de la iteración iguale a la variable "aula_fin", que es el número de aula final, 16 cuando no hay un rango seleccionado o el aula final cuando existe un rango.

4.5.11 Ciclo Ejecutor de Tareas por Aula

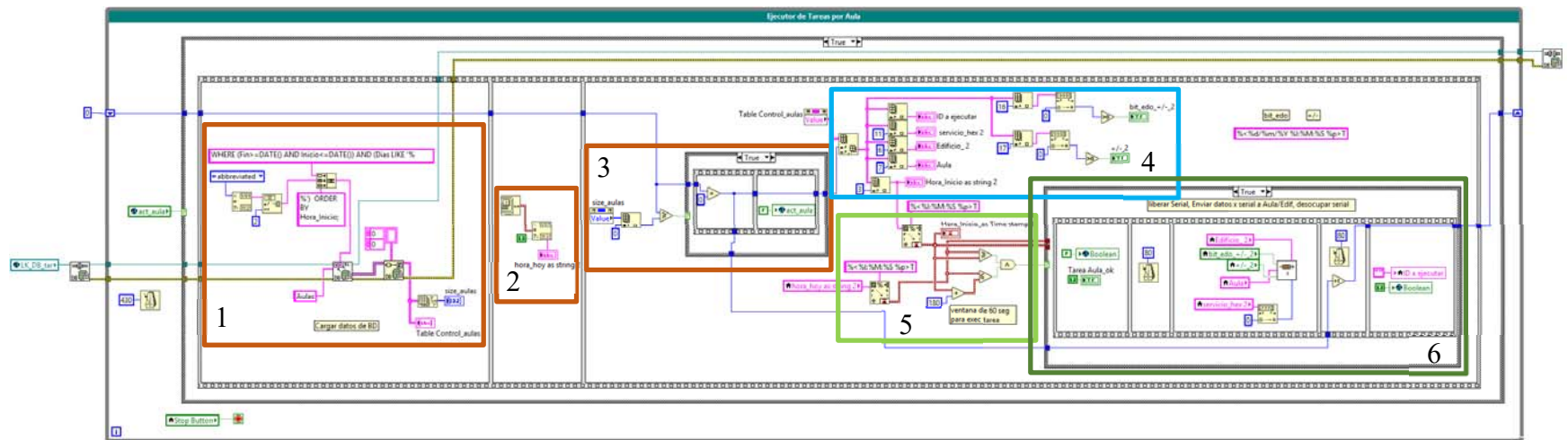


Figura 59 Ciclo ejecutor de tareas por Aula

La ejecución en este ciclo es similar al ciclo de Edificio, la diferencia es que aquí sólo se realiza para una aula, pero el proceso es muy similar, lectura de base de datos, extracción de datos de la tabla de tareas, comparador de hora y ejecutor de tarea. A continuación, se describe por secciones.

1. *Lectura de Base de Datos.*- Se cargan tareas que se encuentran en la base de datos, se utiliza un query para cargar únicamente aquellas tareas que se tienen que ejecutar el día en curso y cuya fecha de finalización no haya pasado; al mismo tiempo las tareas son ordenadas por hora de inicio y se despliegan en la tabla "Tarea por Aula" que aparece en la pantalla de Ejecutor de Tareas (figura 42), se guardamos en la variable "size" el tamaño de la tabla (filas, columnas)

2. *Fecha y hora.*- Se despliega la fecha y hora de hoy pero no es desplegado en pantalla visible para el usuario, solo se dejó visible la del ciclo por Edificio.
3. *Índice de tabla.*- Como se observa en el diagrama, ésta sección es alimentada por un "shift register", el cual nos permite movernos de fila en fila en la tabla "Tarea por Aula" (cuadro verde en figura 42); cada vez que hay una ejecución del ciclo, el shift register aumenta en 1, lo que significa moverse a la siguiente fila de la tabla. Se hace una comparación para verificar que shift register no sea mayor al número de filas de la tabla; en el caso que sea igual o mayor, el shift register se multiplica por 0 (reinicia el shift register) y la variable "act_aula" se carga con FALSE (desactivando la ejecución de tareas), esto significa que ya no hay tareas por ejecutar puesto que se recorrió toda la tabla. En caso de ser menor, el número cargado en el shift register se pasa a las siguientes secciones en la siguiente figura se muestra el case FALSE.

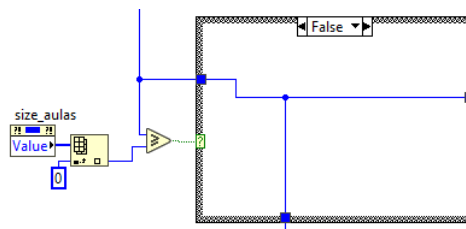


Figura 60 Case false de índice de tabla

4. *Extracción de datos.*- Se extrae de la tabla los datos necesarios para la ejecución de las tareas, se cargan a variables como son Edificio, hora de inicio, bit de rango, si hay rango inicio y fin de rango, bit de sumar o restar estados.
5. *Hora de ejecución.*- Leyendo la hora de inicio de la tarea a ejecutar, ésta entra a una serie de comparadores de hora; cuando la hora actual del sistema es mayor o igual a la hora de inicio, inicia la ejecución de la tarea. Por otro lado establecemos un margen/ventana de ejecución, en este caso 180 segundos, esto nos sirve cuando hay varias tareas a ejecutarse a la misma hora.
6. *Ejecución de tarea.*- Una vez que la hora de ejecución es cumplida, se dispara un Case en el que se encuentra una estructura "flat sequence". En la primera sección del "flat sequence", se carga FALSE a la variable global "Boolean", con esto desactivamos el proceso de Monitor, se enciende el Led que se encuentra en la

pantalla Ejecutor de Tareas (figura 42, rectángulo naranja), a un costado de la tabla correspondiente. En la segunda y cuarta sección, son retardos. En la tercera sección se cargan las variables leídas en la sección 4 al subprograma “Sub_vi_ejecutar_Aula_”, las variables cargadas son Edificio, Servicio, aula, bit de rango, bits de suma o resta de estados. En la última sección del “flat sequence”, se pone en TRUE la variable “Boolean”, por lo que activamos nuevamente el proceso de monitoreo.

4.5.12 Inicialización de variables

La inicialización de variables se realiza fuera del ciclo while, para el ejecutor de tareas por aula, inicializamos como FALSE el led que indica cuando hay una ejecución de tarea, el led se encuentra en la pantalla de Ejecutor de tarea en el rectángulo naranja de la figura 42 a lado derecho de la tabla “tarea por Aula”.

También se inicializa la variable “act_aula”, por lo que se activa el case que se encarga activar/desactivar todo el proceso de la ejecución de la tarea.

La inicialización se muestra en la siguiente figura:



Figura 61 Variable de inicialización, led indicador de ejecución de tarea

4.5.13 Sub Programa “Sub_vi_ejecutar_Aula”

En las siguientes figuras 62 y 63, se muestra el panel frontal y el ícono representativo, recordemos que éste sub programa tiene como variables de entrada los datos extraídos de la tabla Aula de base de datos “tareas_prog”, y se manda llamar en la sección 6 del ciclo “Ejecutor de tareas por Aula”.

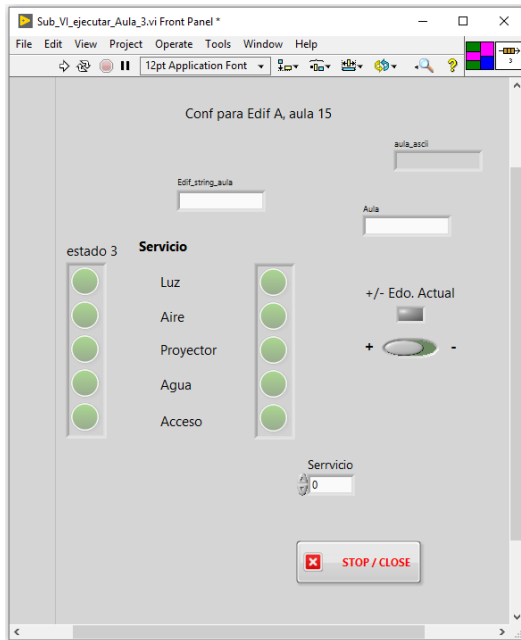


Figura 62 Panel Frontal de “Sub_VI_ejecutar_Aula”

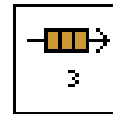


Figura 63 ícono de “Sub_VI_ejecutar_Aula”

Al igual que el subprograma ejecutor por Edificio, el panel frontal no se muestra al usuario; la mayoría de los elementos ahí presentes son variables que se suministran al subprograma desde la tabla Aulas base de datos “tareas_prog”, en donde se almacenan las tareas por aula.

En la figura 64 se muestra el diagrama a bloques:

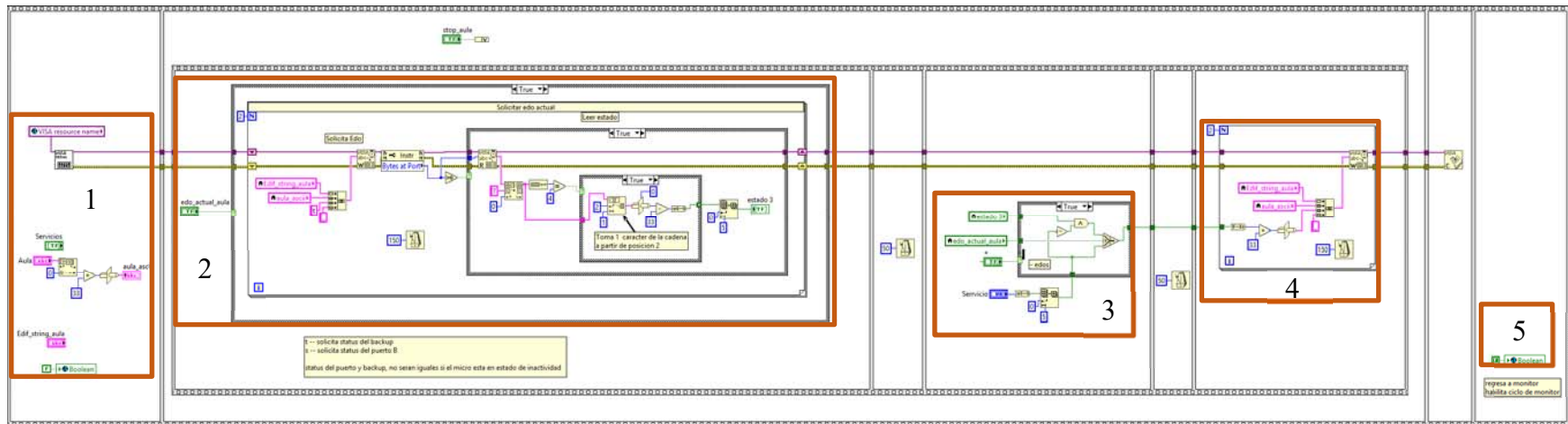


Figura 64 Panel Frontal de "Sub_VI_ejecutar_Aula"

Podemos decir que este sub_vi es un caso particular del "Sub_vi_ejecutar_Edificio"; puesto que el anterior realiza una iteración en un ciclo while para recorrer cada una de las aulas; aquí no se tiene dicho while puesto que es sólo 1 aula. El diagrama se compone de una "flat sequence" y un ciclo for en la sección 2. Al igual que el "sub_vi_ejecutar_Edificio", el programa revisa si hay nuevo estatus de servicio o si es sumar o restar servicios al estado de los servicios actuales, basado en la variable "edo_actual_aula"; por lo que se solicita el estatus hacia el microcontrolador (en caso de sumar o restar status), hace las comparaciones necesarias para obtener el status resultante y por último envía el status resultante por el puerto serial, que a su vez se verá reflejado en el PORTB del microcontrolador del aula especificada. A continuación, se describe con mayor detalle

- 1 *Puerto COM.-* Se carga la variable global "VISA resource name" para acceder al puerto COM establecido en el programa de monitor (panel front de figura 41, rectángulo amarillo). La variable global "Boolean" se pone en FALSE para desactivar el ciclo de monitor y liberar el puerto COM. Así mismo se hace una conversión de un string en decimal a su equivalente en código ASCII con un corrimiento.
- 2 *Bit suma/resta de estado.-* Se verifica la variable "edo_actual_aula", que nos indica si el nuevo estado es para sumar/restar el estado almacenado al estado actual. En caso de ser TRUE, se solicita estatus del PORTB del aula, se solicita estatus con "+" indicándole al microcontrolador que lo solicita un subprograma; se verifica que la longitud de la cadena recibida sea correcta y se extrae el byte 2, que es el status y guarda en "estado 3"; en caso de ser FALSE, no se realiza ninguna lectura, esto significa que el mismo estado leído de la base de datos se enviará como status final hacia el microcontrolador. En la figura 65, se observa el case cuando es FALSE.

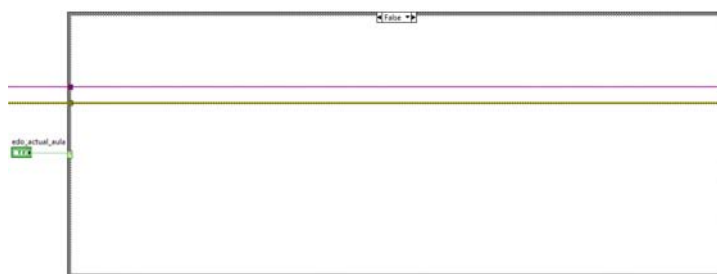


Figura 65 Case FALSE de bit de rango

- 3 *Estado final.-* En este CASE se define el estado final que se enviará al microcontrolador. Como variable de comparación está el bit de sumar/restar, y dentro de cada CASE ya sea FALSE/TRUE hay un SELECT que depende de la variable "edo_actual_aula", esta variable nos indica si vamos a sumar/restar estado o enviar el estado leído como un nuevo estado hacia el aula. En la figura 64 se observa el caso TRUE, indicando que el estado leído (en la base de datos) se va a restar del estado actual (en el aula); es el resultado de una compuerta AND entre el estado actual del aula (variable "estado 3") y el NOT (para invertir estados) del estado leído en la base de datos (variable "Servicio"). En la figura 66 se observa el estado FALSE. El caso FALSE indica una suma de servicios, el estado resultado viene de una compuerta OR entre la variable "Servicio" y el estado actual del aula (variable "estado 3"). El estado final también depende del bit "edo_actual_aula", si esta variable es FALSE independientemente del case TRUE/FALSE, el estado final es el estado leído desde la base de datos, es decir no hay sumas ni restas de estados si no estado nuevo

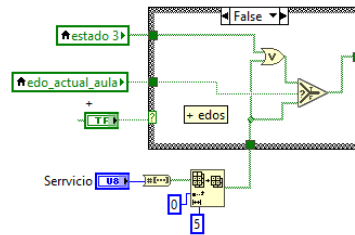


Figura 66 Case FALSE de bit de rango

4. *Envío de Status.*- Una vez que se recibe el estatus final de la sección 3, se construye la palabra para enviar a través del puerto serial, se concatena Edificio, Aula, estatus resultante y el carácter “_”.

5. *Variable Boolean.*- La variable global se pone en TRUE activando nuevamente el ciclo de monitor y con esto el sub programa termina su ejecución.

4.5.14 Programador de Tareas

Este programa se ejecuta cuando se presión el botón “Programador/Programar Tareas”, se encuentra dentro del rectángulo verde en la Figura 42. En las figuras 67 y 68 se muestran los tabs para aula y edificio respectivamente.

4.5.15 Programación de Tareas por Aula

ID	Fecha_Registro	Inicio	hora_inicio	Fin	hora_Fin	Edificio	Aula	Servicio	Dias	Notas
112	31/08/2019 01:57:00 p. m.	29/08/2019	07:00:00 a. m.	30/12/2019	09:00:00 p. m.	E	15	a d e	Lu Ma Mi Ju Vi Sá Do	
114	01/09/2019 03:35:00 a. m.	03/09/2019	01:17:00 p. m.	27/09/2019	08:00:00 p. m.	A	3	a d e	Vi Sá Do	
134	08/09/2019 04:40:00 p. m.	08/09/2019	04:55:00 p. m.	30/09/2019	04:40:00 p. m.	A	4	b c d	Lu Vi Sá Do	

Figura 67 Programador de tareas por aula

La programación consiste en un form en la que se seleccionan los datos. En la figura 67 para programar tareas para aulas, primeramente, se selecciona el Edificio y Aula de los menús correspondientes. En seguida se seleccionan los servicios a utilizar dando un clic al led correspondiente a cada servicio, después seleccionamos los días en los que la tarea se va a ejecutar.

Enseguida seleccionamos primeramente la fecha final, es decir la fecha y hora en que la tarea va a ser válida, esto no significa que se ejecutará una rutina de finalización, simplemente es una hora y fecha de validación; después seleccionamos la fecha y hora de inicio, aquí la hora de inicio es la hora en que la tarea se ejecutará.

Con estos datos se enviará un status nuevo al aula, sin importar los servicios actualmente activos, es decir sobre escribirá el status actual en el aula.

Sin embargo, el botón "+/- Edo. Actual", nos permite agregar o quitar servicios al estado actual del aula, cuando se activa este botón automáticamente aparece otro en el que por default se encuentra en el signo "+", con esto indicamos que la tarea consiste en agregar servicios, con "-" se quitan estados. Por lo que, al ejecutarse la tarea primeramente se revisa qué servicios se encuentran activos y posteriormente agrega o quita estados según corresponda.

En el recuadro de Notas, se puede agregar notas o información auxiliar de la tarea. Una vez que se oprime el botón guardar, la tarea es almacenada en una base de datos con un ID específico (columna ID) y al mismo tiempo la información completa de la tarea se puede ver en la tabla.

En la tabla se muestran todas las tareas y son ordenadas en base al ID, en la tabla se observa la información completa de la tarea: ID, fecha en que se registró la tarea, fecha en que inicia ejecución así como su hora de ejecución (hora_inicio); "Fin" y "hora_fin" indica la fecha y hora en que la tarea será válida, Edificio, Aula, servicios que activarán y los días en que se ejecutarán, así como las notas.

También se agregó un botón para eliminar tareas, las tareas se eliminan en base a su ID; se coloca el número de ID en el recuadro "ID Tarea" y se presiona el botón "Eliminar Tareas". Automáticamente se observa que la tarea se elimina de la tabla y por tanto de la base de datos.

4.5.16 Programación de Tareas por Edificio

The screenshot shows a software window titled 'Form_prog_tareas_final_3.vi'. It has a menu bar (File, Edit, View, Project, Operate, Tools, Window, Help) and a toolbar. The main area is divided into several sections:

- Programación por Edificio:**
 - Edificio:** A dropdown menu showing 'A'.
 - Servicio:** A list of services with green circular indicators: a Luz, b Aire, c Proyector, d Agua, e Acceso.
 - Fecha y hora de inicio - fin:** Two date/time pickers. 'Fin' is set to 31/12/2020 12:00. 'Inicio' is set to 10/11/2019 12:00.
 - Notas:** A large text area for notes.
 - Aplicar a un rango:** A section with a 'De' field (5) and a 'Hasta' field (12), and a '+/- Edo. Actual' button.
 - Guardar:** A button to save the task.
- Tabla Edificio:** A table with the following data:

ID	Fecha_Registro	Inicio	hora_inicio	Fin	hora_Fin	Edificio	Aula	Servicio	Dias	Notas
220	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:00 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	a b c d e	Lu Ma Sá Do	
221	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:05 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	- a b	Lu Ma Sá Do	
222	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:10 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	- c d	Lu Ma Sá Do	
223	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:15 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	+ c	Lu Ma Sá Do	
224	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:20 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	+ a	Lu Ma Sá Do	
225	02/11/2019 02:05:00 p. m.	02/11/2019	02:28:25 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10		Lu Ma Sá Do	
226	02/11/2019 02:05:00 p. m.	02/11/2019	02:28:35 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	a b c d e	Lu Ma Sá Do	
- Eliminar Tarea:** A button with a red 'X' icon, located below the 'ID Tarea' field.
- SALIR:** A red button to exit the program.

Figura 68 Programador de tareas por Edificio

El funcionamiento es idéntico a la pantalla de Tareas por Aula. La diferencia está en que se agregó un nuevo botón, "Aplicar a un rango". Como su nombre lo indica, nos sirve para aplicar modificaciones al rango de aulas que es inclusivo. Cuando se oprime el botón aparece automáticamente 2 casillas nuevas en las que se selecciona el rango de aulas.

En la tabla de la figura 68 se observa que la columna aula tiene rango específico, en este caso 7-10; cuando no se selecciona un rango la columna Aula está en blanco.

Por otro lado, en la columna servicio se despliegan los servicios programados, sin embargo, también observamos en el ID 222 y 223 (figura 27) que aparecen signos "+" y "-", con esto se indica que la tarea agregará o quitará servicios. Lo mismo ocurre la programación por aula, pero en la figura 26 no hay una tarea registrada con esta característica.

El botón "SALIR" funciona de la misma forma tanto en programador por Edificio como por Aulas, primeramente, detiene la ejecución del programa y después cierra la ventana por completo.

4.5.17 Diagrama a bloques de programador de tarea

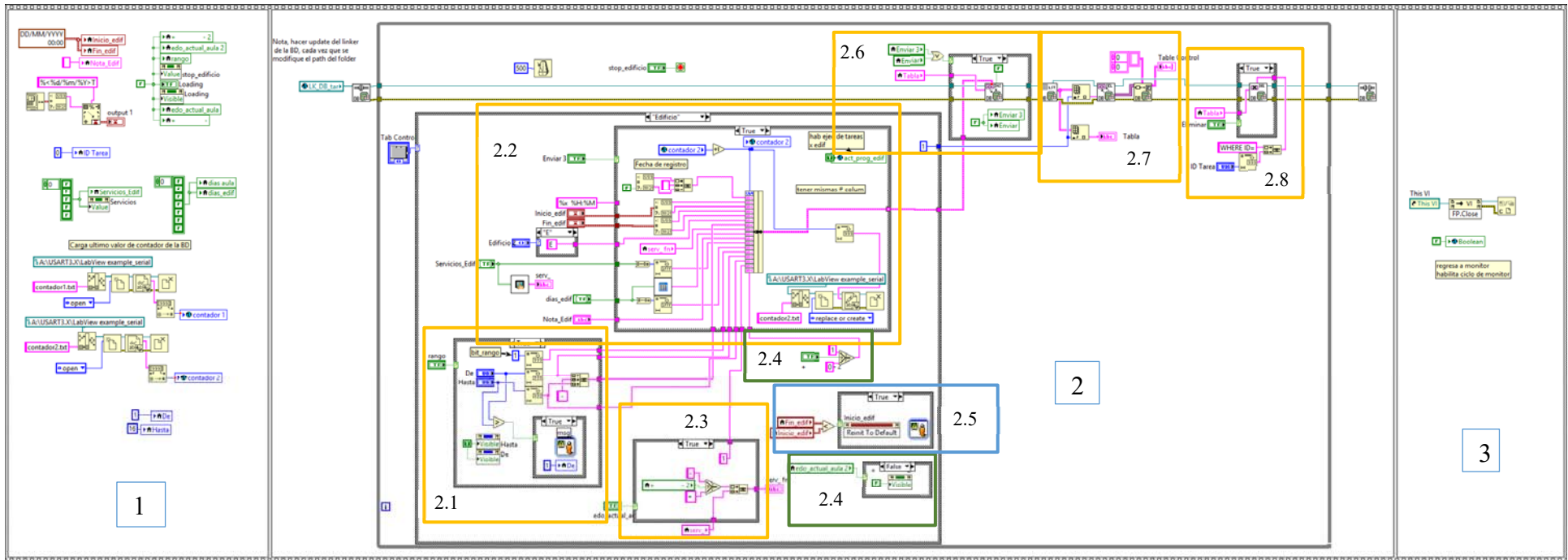
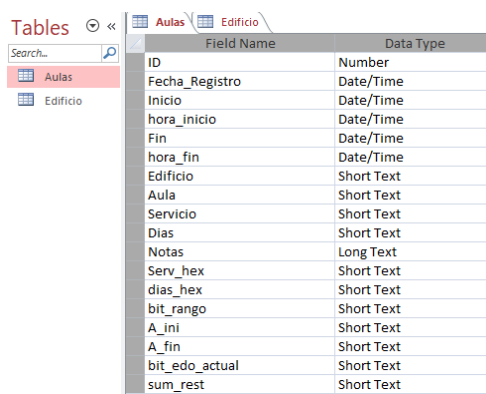


Figura 69 Diagrama a bloques de programador de tareas.

Como se observa en la figura 69, el programa se consiste en una estructura "flat sequence" de 3 secciones, en la primera observamos inicialización de variables y se cargan variables globales desde un archivo de texto, etc; en la segunda un ciclo while en el que se le lee y se escribe una base de datos tanto para Aulas y Edificio; la base de datos contiene un total de 18 columnas, como se muestra en la figura 70; en la última sección, cuando termina el programa hacemos que la ventana se cierra de forma automática. A continuación, se explica a detalle por sección:



Field Name	Data Type
ID	Number
Fecha_Registro	Date/Time
Inicio	Date/Time
hora_inicio	Date/Time
Fin	Date/Time
hora_fin	Date/Time
Edificio	Short Text
Aula	Short Text
Servicio	Short Text
Dias	Short Text
Notas	Long Text
Serv_hex	Short Text
dias_hex	Short Text
bit_rango	Short Text
A_ini	Short Text
A_fin	Short Text
bit_edo_actual	Short Text
sum_rest	Short Text

Figura 70 Columnas Base de Datos

1. *Inicialización de variables.*- Se cargan variables globales “contador 1” y “contador 2” desde archivos, éstas variables se utilizan para el ID de la tarea; “contador 1” para Aulas y “contador 2” para Edificio. Se carga con FALSE el array de los servicios, es decir los leds que aparecen en el panel frontal, tanto para el tab de aula y edificio (“Servicios_Edif” y “Servicios”), también se carga FALSE en el array de los check box, donde se seleccionan los días y demás variables involucradas en la lectura de la base de datos.
2. *Ejecución de programa.*- En la figura 28, se observa la ejecución consiste en ciclo while para el tab de Edificio, el programa lee y escribe en la base de datos, en la escritura, los datos se cargan en formato de string, se concatenan los datos en el mismo orden que las columnas que se encuentran en la base de datos.
 - 2.1. *Rango de Aula*, cuando se requiere un rango (caso TRUE), se oprime el botón “Aplicar a un rango”, dos box aparecen inmediatamente “De” y “Hasta”, como se observa en la figura 27; y los rangos se introducen en la correspondiente box; los números se transforman en tipo string, se concatenan con un separador “-” y se envían a la sección 2.2, donde se coleccionan todos los datos para cargar en la base de datos, también se escribe en la columna “bit_rango” “1” en la base de datos. Se agregó una validación para que la variable “De” no sea mayor a “Hasta”, en caso de serlo, manda un mensaje al usuario y se coloca en “De” el valor de 1. Cuando no se selecciona rango, los box “De” y “Hasta” no son visibles, se concatena solo espacio en blanco y en la columna “bit_rango” se coloca “0”, en la siguiente figura se observa el case FALSE.

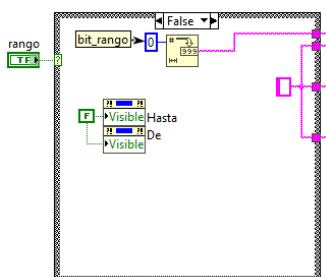


Figura 71 Case FALSE rango de Aula

2.2. *Concatenación.* – La estructura Case se hace TRUE cuando el botón “Guardar” (figura 68) se presiona. Se concatena toda la información que se escribirá en la base de datos y se envían a la sección 2.3, los datos que se recopilan se convierten en string excepto el ID, es numérico. Se acomodan los datos en el mismo orden de las columnas de la base de datos. Al usuario final se muestra en las tablas tanto de programador de tareas como en el ejecutor de tareas desde la columna “ID” hasta la columnas “Notas”; las columnas que no se muestran, son las que utilizamos para el procesamiento/ejecución de la tarea, se almacenan datos de tipo hexadecimal, número y booleano (puesto que se facilita la manipulación), pero que son convertidos en tipo string cuando se almacena en la base de datos; cuando se leen datos de la base de datos (es decir, cuando se ejecuta una tarea) se recupera el tipo de dato y se procesa. La columna “Serv_hex” y “días_hex” almacena los servicios y días como valores de un array en hexadecimal, “bit_rango”, “bit_edo_actual” y “sum_rest” son de tipo booleano; y las columnas “A_ini” y “A_fin” almacenan números referentes al rango de aulas, en caso de existir un rango.

Para concatenar Servicios y Días, utilizamos un par de subVI. Para servicios (subVI_extr_serv) ya se explicó en la sección ciclo Monitor de figura 44, sección 7; para días (subVI_extr_dias) el funcionamiento es el mismo, por lo que se omite explicación y se muestra el diagrama en la siguiente figura.



Figura 72 Panel Frontal subVi_extr_dias

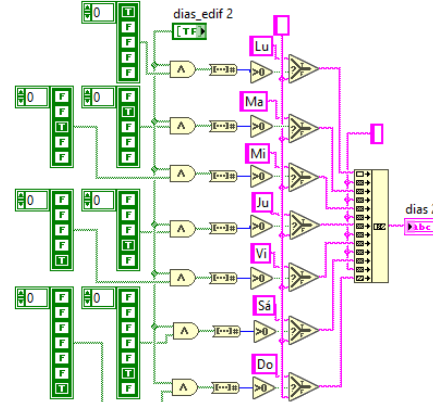


Figura 73 Diagrama a bloques subVi_extr_dias

También aquí se escribe en archivo el valor de la variable contador incrementado en 1, recordemos que nos sirve para almacenar en la base de datos un ID único. Cuando no se presiona el botón “Guardar” no se realiza ninguna actividad.

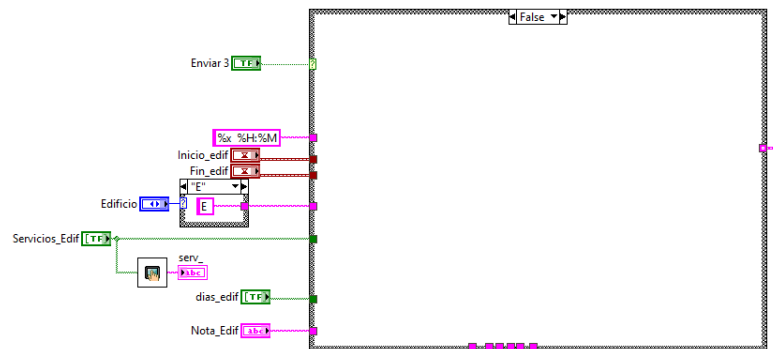


Figura 74 Case FALSE Concatenacion

2.3. *Agregar / quitar tareas.* Se concatena el símbolo “+” o “-” a los servicios y se agrega la cadena a la columna “Servicio” de la Base de datos en caso de que la variable “edo_actual” sea TRUE, con lo que se indica que habrá una suma o resta de servicios, al mismo tiempo se escribe “1”, en la columna “bit_edo_actual” de la base de datos para el mismo fin. En caso de ser FALSE, no habrá suma ni resta de servicios, por el valor de variable servicio se agrega tal cual a la columna “Servicio” de la base de datos, en la siguiente figura se observa el case FALSE.

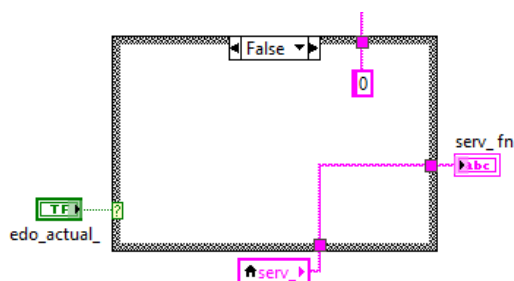


Figura 75 Case FALSE agregar / quitar tareas

2.4. *Boton Visible +/-*. Cuando se presiona el Led "+/- Edo. Actual" (Case TRUE), significa que se considerará el estado de los servicios actuales y por tanto el case hace visible el botón "+ / -". En la figura 69 se observa el Case FALSE, en la siguiente figura se tiene el case TRUE, en el que se establece un TRUE a la propiedad de visible. Con la ayuda de un SELECT escribimos en la base de datos "1" o "0" en la columna "sum_rest", dependiendo de la posición del botón, si el botón está en "+" (cuando es FALSE), escribe "0".

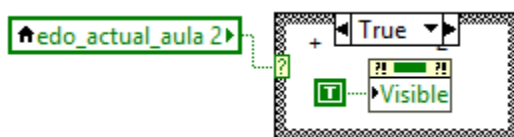


Figura 76 Case TRUE botón + / -

2.5. *Validación de fecha*. Se valida que la hora y fecha final no sea menor que la inicial, en caso de ser mayor, como se ve en la figura 69, envía un mensaje al usuario acerca del error y reinicia valores de los Time stamp (controles de fecha y hora). En case de ser FALSE, no se hace ninguna acción puesto que la validación pasa y la fecha, hora de inicio y final se agregan como tipo "string" a la base de datos.

El diagrama para el TAB de AULA, es muy similar, la diferencia es que no se selecciona un rango de aula (ver figura 67) y por lo tanto en las columnas "bit_rango", "A_ini" y "A_fin" se concatena espacio en blanco, en la columna aula, en vez de escribir un rango de aulas, es un valor único. en la siguiente figura se muestra el diagrama.

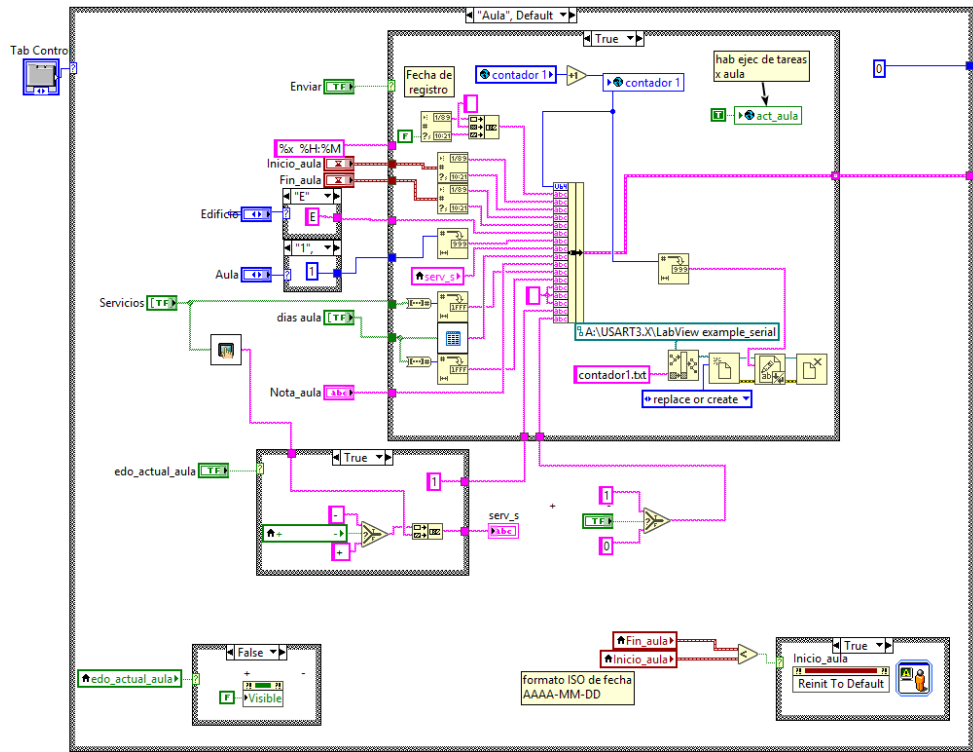


Figura 77 Diagrama de tab para Aula

2.6. *Guardar datos.* Cuando se oprime el botón "Guardar" se almacenan los datos en la base de datos procedentes de la sección 2.2.

2.7. *Desplegar datos.* Aquí se lee la base de datos y se despliegan las tareas almacenada en la tabla correspondiente, las tablas se seleccionan de acuerdo a un índice, 0 es para la tabla AULA y 1 para Edificio. En la figura 28 se observa el valor 1, que viene del tab de Edificio.

2.8. *Eliminar tarea.* De acuerdo al ID de tarea ingresado, en esta sección se elimina de la base de datos.

3. *Cerrar VI.* Cuando se presiona el botón salir de la figura 27, el ciclo while termina y pasa a la sección 3, con esto el subVi se cierra automáticamente.

4.5.18 Programa “Servicios Manual”

En este programa se establecen en los servicios de forma inmediata para Aula o Edificio de manera manual. El programa está organizado por tabs, uno para edificio y otro para aula. El panel frontal y diagrama a bloques es muy similar a los de ejecutor de tareas y programador de tarea, por lo que la funcionalidad es muy similar; la diferencia es que los datos son leídos del panel frontal en vez de la base de datos. En las siguientes figuras se muestra el panel frontal tanto para aulas y Edificio

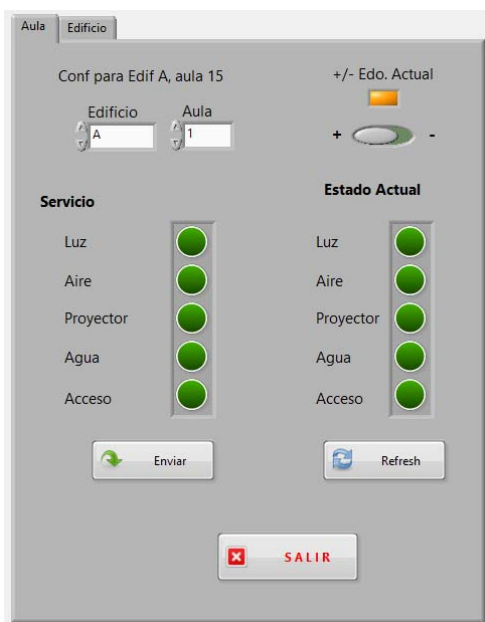


Figura 78 Panel frontal Aula de programa “Servicios Manual”



Figura 79 Panel frontal Edificio de programa “Servicios Manual”

La funcionalidad es la misma que el programador de tareas. Para Aulas, seleccionamos primero Edificio, después número de Aula; el botón “+/- Edo. Actual” nos permite decirle al programa que se va a tomar en cuenta los servicios que actualmente están habilitados en el aula, por lo que se van agregar o quitar servicios; cuando se selecciona dicho botón, éste se ilumina en naranja, como se muestra en las figuras, e inmediatamente se hace visible el botón “+ -”, de éste botón depende si se suman o restan servicios. Cuando no se selecciona el botón “+/- Edo. Actual” significa que los servicios seleccionados son los que estarán presentes en el aula.

En la parte de servicios se da click a los que servicios requeridos y se presiona el botón “Enviar”; inmediatamente los cambios son efectuados en el aula de edificio seleccionado. El botón “Refresh” permite visualizar los servicios que están habilitados en el aula, éstos se ven reflejados en la columna “Estado Actual”.

Para el tab de Edificio, misma funcionalidad que el programador de tareas. Seleccionamos Edificio, los servicios requeridos, en caso de aplicar a un rango de aulas apretamos el botón “Aplicar a un rango” los

boxes de "De" y "Hasta" son visibles y escribimos el rango requerido. Al igual que en el aula, en caso de requerir agregar o quitar estados pulsamos el botón "+/- Edo. Actual" y se hace visible el botón "+ -" para seleccionar la opción deseada. Al presionar "Enviar", los cambios son efectuados en cada una de las aulas seleccionadas; en caso de no seleccionar un rango, se tiene considerado 16 aulas por edificio, por lo que los cambios se realizan del aula 1 a la 16 por default.

En la siguiente figura veremos el diagrama a bloques, que a su vez es muy semejante al ejecutor de tareas tanto para aulas como para edificio.

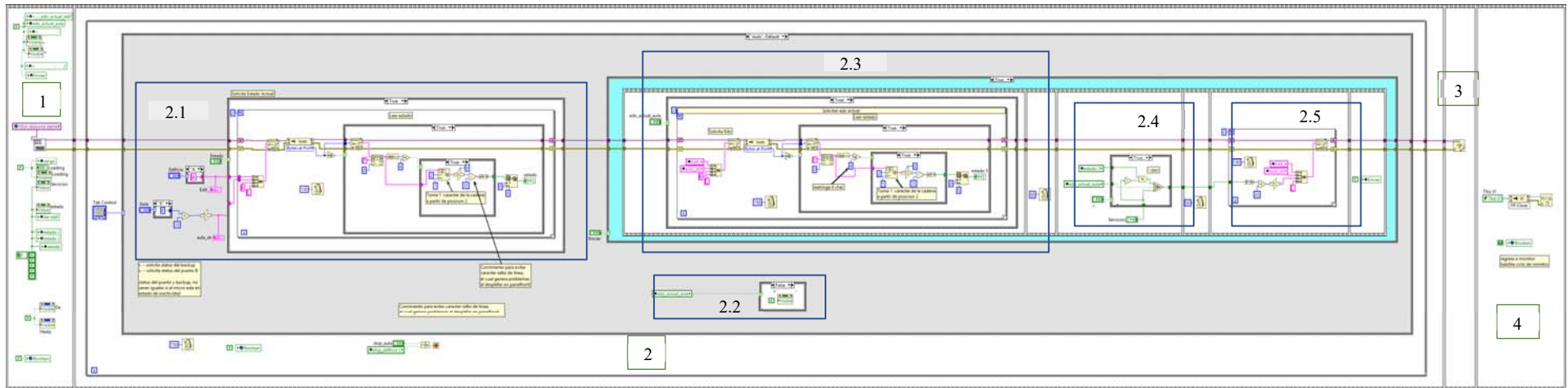


Figura 80 Diagrama a bloques de programa "Servicios Manual", tab Aula

El programa está en una estructura "flat sequence" dividido en 4 regiones:

1. *Inicialización de variables.*- Se inicializan en FALSE y se abre el puerto COM, es importante mencionar que la variable "Boolean" es la variable global que desactiva el ciclo de monitor, con lo que se libera el puerto y permite ser usado en este programa.
2. *Ejecución de Programa.*- Observamos aquí el tab, que en el caso de la figura anterior es para Aula.
 - 2.1. *Boton Refresh.* Al presionar el botón, en ésta sección se envía una petición de status de los servicios y se despliega los servicios activos en la columna "Estado Actual".
 - 2.2. *Estructura Case encargada se hacer visible el botón "+ -",* cuando se presiona el botón "+/- Edo. Actual", en la figura se observa el caso FALSE, en la siguiente se observa el caso TRUE:

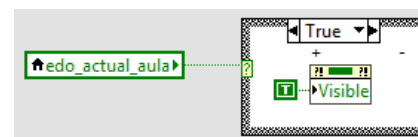


Figura 81 Case TRUE sección 2.2

2.3. *Solicitud de Estatus*.-Las secciones 2.3, 2.4 y 2.5, se ejecutan cuando se presiona el botón "Enviar". En 2.3 cuando se tiene activado el botón "+/- Edo. Actual", se solicita el status actual del aula y se guarda en la variable "estado 3", cuando no se requiere agregar o quitar servicios, el estado FALSE no hace nada, como se muestra en la siguiente figura:

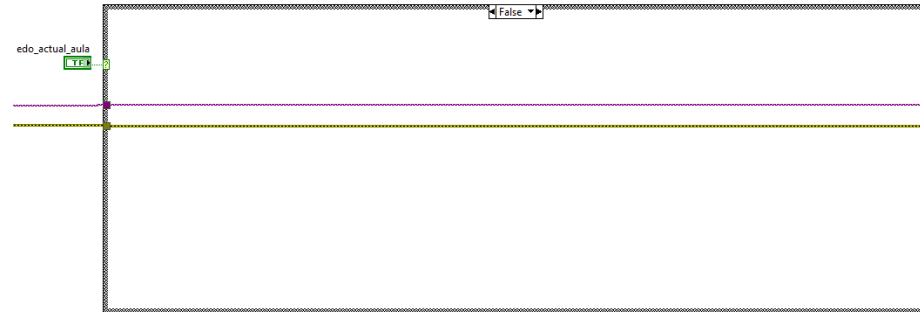


Figura 82 Case FALSE sección 2.

2.4. *Estado Final*.- Aquí se determina el estado final de los servicios que se enviará al Aula, tiene el mismo funcionamiento que el subvi "Sub_VI ejecutar_aula" del ejecutor de tareas por aula; el estado final de los servicios dependen si se requiere o no sumar o quitar servicios a los que actualmente se encuentran activos en el aula.

2.5. *Envío de datos*. Se envía el por el puerto serial el estado final hacia el aula seleccionada.

3. *Cerrar puerto*.- Cuando se presiona el botón "SALIR", interumpimos el ciclo WHILE de la sección 2 y se pasa a ésta sección, que consiste en cerrar la conexión del puerto
4. *Cerrar programa*.- Después de cerrar el puerto, se pone la variable "Boolean" en TRUE, para permitir al ciclo Monitor continuar con su ejecución y a su vez, hacemos que el programa cierre por completo.

En la siguiente figura se observa el TAB para edificio, el diagrama es muy similar, solo que se agrega la sección de rango de aulas y no se incluye el botón de "Refresh", muy similar también al subVi "Sub_vi_ejecutar_Edificio" que se utiliza en el ciclo "ejecutor de tareas por edificio"; de igual manera se explica por secciones.

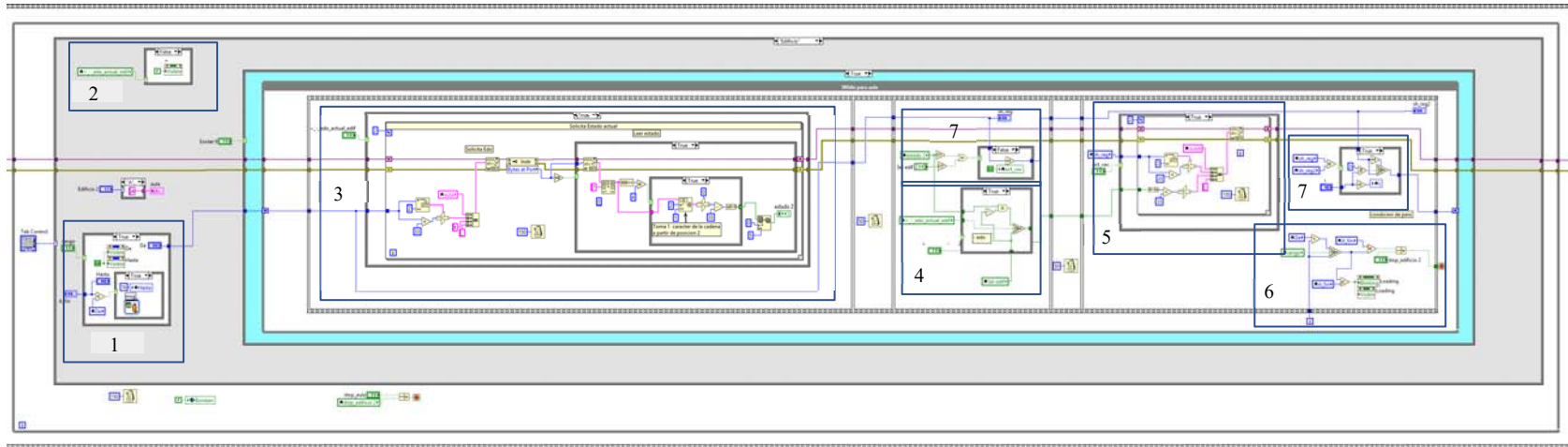


Figura 83 Diagrama a bloques de programa "Servicios Manual", tab Edificio

1. *Rango de Aula*, en la figura se muestra el caso TRUE, cuando se presiona el botón de rango, se hacen visibles los boxes "De" y "Hasta" y a su vez pasan por una validación para evitar que inicio sea mayor que fin, En el caso que no se seleccione rango de aula (caso FALSE), el valor por default son 1 y 16. El valor de "Inicio" se carga a un "shift register", que junto con un ciclo while se ejecutan las acciones a todas las aulas. En la siguiente figura se muestra el case FALSE.

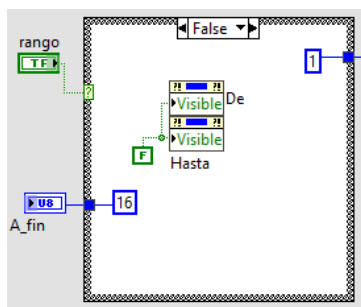


Figura 84 Case FALSE tab Edificio

2. *Botón "+ -"*, cuando se habilita el botón "+/- Edo Actual" para agregar o quitar servicios, esto hace visible un nuevo botón "+ -", el cual nos sirve para indicar si se agregarán o quitarán servicios. Cuando no está habilitado, el botón permanece oculto.
3. *Solicitud de Estado*.- Cuando se presiona "Enviar" y el Botón "+/- Edo Actual" se habilita, entonces el programa primeramente solicita el estado de los servicios del aula en curso y guarda el resultado en el array "estado 2". Recordemos que el botón nos sirve para especificar al programa que vamos a agregar o quitar servicios al estado actual; cuando el botón está deshabilitado, no ejecuta acción alguna.
4. *Estado Final*. - Aquí se determina el estado final de los servicios que se enviará al Aula, el estado final de los servicios depende si se requiere o no sumar o quitar servicios a los actuales en el aula.
5. *Envío de datos*. Se envía el por el puerto serial el estado final hacia el aula en curso.
6. *Condición de Paro*.- El ciclo WHILE detiene su ejecución cuando la iteración iguala valor del box "Hasta", que es el aula final, que por default es 16, o el valor especificado por el rango.
7. *Persistencia de error*. - Cuando se detecta un arreglo vacío se considera un error, y se incrementa una variable contadora "x" para contabilizar los errores, ésta estructura CASE es una condición para incrementar el shift register o dejarlo igual, con la intención de repetir la iteración para la misma aula hasta en 3 ocasiones. En la siguiente figura se observa el caso en que existe error (se detecta un arreglo vacío), por lo que el valor del shift register se pasa a la seccion de contador, y al llegar a 3 el contador de error, el ciclo corre para la siguiente aula.

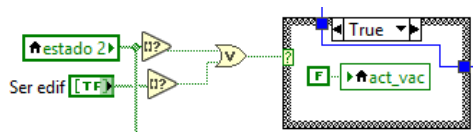


Figura 85 Detección de arreglo vacío

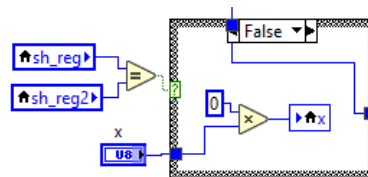


Figura 86 Case FALSE Contador de error de vacío

Para pasar a la sección 5, envió de datos, tiene que pasar la validación de la sección 7, cuando hay error, no hay envío de status, pues la variable “act_vac” se pone en FALSE y el valor del shift register no cambia; cuando hay un error, la variable x, incrementa su valor, si el número de errores es menor a 3, se ejecuta todo el ciclo nuevamente para la misma aula, cuando el contador llegue a 3, automáticamente se descarta el aula actual y se incrementa en 1 el valor del shift register, con esto se pasa al aula siguiente.

4 Funcionamiento del Sistema

4.6 Vista general

A continuación, se muestran los elementos principales del sistema, así como las funciones que desempeñan en el sistema.

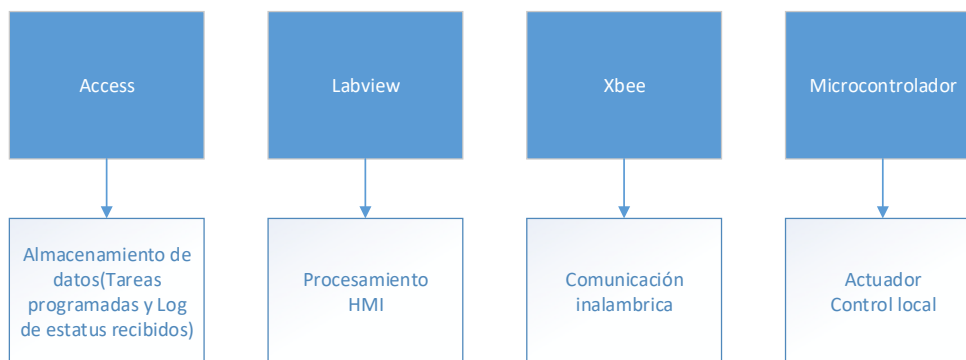


Figura 87 Sistemas del proyecto

4.7 Integración de sistemas

A continuación, se muestran los elementos principales del sistema, así como las funciones que desempeñan en el sistema

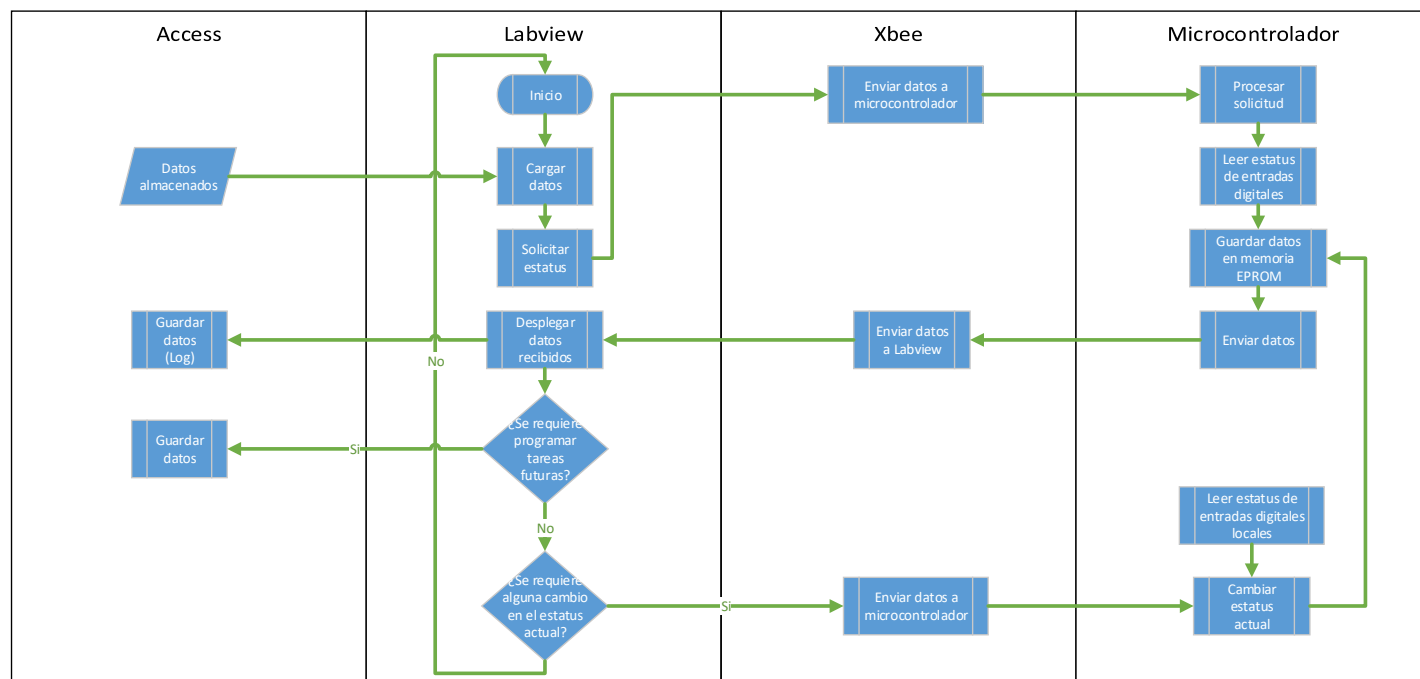


Figura 88 Integración de sistemas

4.8 Pantallas del sistema

Las siguientes son las pantallas con las que el usuario interactúa en el programa Labview.

4.8.1 Pantalla principal de monitoreo

En esta pantalla se muestra el estatus actual de las aulas monitoreadas, es decir, Edificio, Aulas y Servicios activos/inactivos.

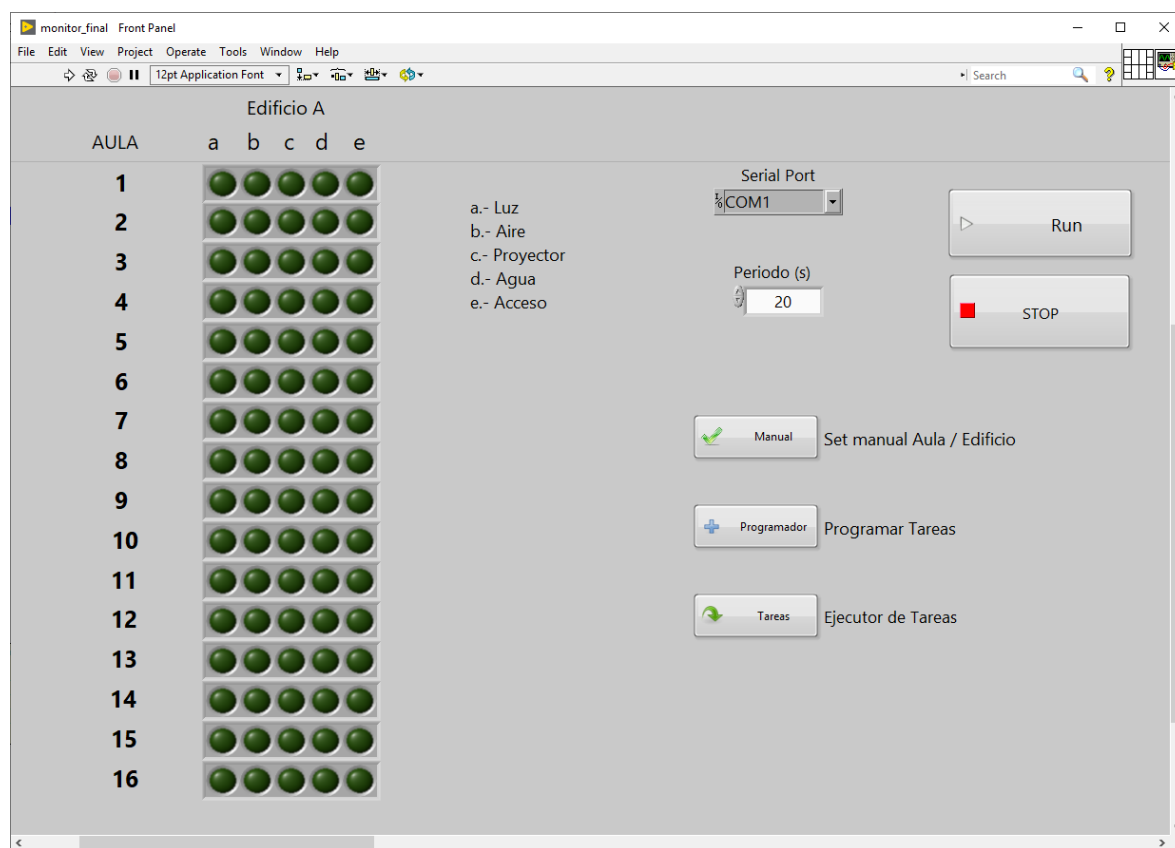


Figura 89 Pantalla principal de monitoreo

4.8.3 Pantalla de programación de tareas por Edificio

En esta pantalla se pueden programar tareas por edificios completos y al igual que en la programación por aula, se determinan la fecha inicio, fecha final y los días de la semana entre el periodo.

Form_prog_tareas

File Edit View Project Operate Tools Window Help

Programación por Edificio

Edificio: A

Servicio:

- a Luz ☒
- b Aire ☒
- c Proyector ☒
- d Agua ☒
- e Acceso ☒

Fecha y hora de inicio - fin:

Fin: DD/MM/YYYY 00:00

Inicio: DD/MM/YYYY 00:00

Aplicar a un rango: De 1 Hasta 16

Notas:

ID Tarea: 0

Eliminar Tarea

SALIR

ID	Fecha_Registro	Inicio	hora_Inicio	Fin	hora_Fin	Edificio	Aula	Servicio	Dias	Notas
220	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:00 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	a b c d e	Lu Ma Sá Do	
221	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:05 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	- a b	Lu Ma Sá Do	
222	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:10 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	- c d	Lu Ma Sá Do	
223	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:15 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	+ c	Lu Ma Sá Do	
224	02/11/2019 02:04:00 p. m.	02/11/2019	02:28:20 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	+ a	Lu Ma Sá Do	
225	02/11/2019 02:05:00 p. m.	02/11/2019	02:28:25 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10		Lu Ma Sá Do	
226	02/11/2019 02:05:00 p. m.	02/11/2019	02:28:35 p. m.	31/12/2019	02:03:00 p. m.	A	7 - 10	a b c d e	Lu Ma Sá Do	

Figura 91 Pantalla de programación de tareas por edificio

5 Referencias

- [1] Dle.rae.es. (2018). RAE. [online] Available at: <http://dle.rae.es/srv/fetch?id=E7W0v9b> [Accessed 8 Apr. 2018].
- [2] H. M. Domínguez, F. Sáez, Domótica: un enfoque sociotécnico. Centro de domótica integral, Primera edición, Madrid, España, 2006.
- [3] Ramirez, H. (2018). Domótica. [en línea] Losavancesdelaingenieria.blogspot.mx. Disponible en: <http://losavancesdelaingenieria.blogspot.mx/2011/05/la-domotica.html> [Fecha de consulta: 8 de Abril 2018].
- [4] Herrera Quintero, Luis Felipe, Viviendas inteligentes (Domótica). Ingeniería e Investigación [en línea] 2005, 25 (agosto) : [Fecha de consulta: 8 de abril de 2018] Disponible en:<<http://www.redalyc.org/articulo.oa?id=64325207>> ISSN 0120-5609
- [5] S.L.U., U. (2018). Domótica KNX, un sistema eficiente para tu casa, empresa u oficina. [en línea] Unitel - Soluciones e infraestructuras Tecnológicas. Disponible en: <https://unitel-tc.com/domotica-knx/> [Fecha de consulta: 8 de Abril de 2018].
- [6] SONOFF: <http://sonoff.itead.cc/en/>
- [7] BROADLINK <https://www.broadlink-mexico.com>
- [8] Publicación en página de Mercado libre: Kit Arduino Uno R3 Desarrollo Casa Inteligente – Domótica, recuperado de https://articulo.mercadolibre.com.ar/MLA-712839074-kit-arduino-uno-r3-desarrollo-casa-inteligente-domotica-_JM
- [9] Mateos Felipe, González Víctor M, Poo Reyes, García María, Olaiz Rosana: Desing and development of an automatic small-sccale house for teaching domotics, Guijón, España, 2001.
- [10] Gupta Anisha, Gupta Punit, Chhabra Jasmeet: IoT based Power Efficient System Design using Automation for Classrooms, Himachal Pradesh, India, 2015.
- [11] Feng Jiajia: Design and Implementation of Lighting Control System for Smart, Nantong,China 2017.
- [12] Products Tagged DigiMesh | Digi International. (2019). Retrieved 18 August 2019, from <https://www.digi.com/products/tag/digimesh>
- [13] Long Range 900 MHz OEM RF Module | Digi International. (2019). Retrieved 18 August 2019, from <https://www.digi.com/products/embedded-systems/rf-modules/sub-1-ghz-modules/xbee-pro-900hp#specifications>