

Satempox Prototype

Medidor de Variables Biomédicas

Daniel Felipe Torres Robles Miguel Ángel Rodríguez Fuentes Sergio Felipe Rodríguez Mayorga.
 Universidad Nacional de Colombia
 Facultad de Ingeniería - Ingeniería Eléctrica y Electrónica
 Electrónica Digital I

Abstract—This report shows the results obtained in the development of the biomedical variables measurer (Satempox Prototype), evidencing its modular stages, its operation and source codes, additionally, the difficulties presented throughout the process are pointed out.

keywords—UART communication protocol, SPI, I2C, ESP32, bluetooth module, FPGA, LCD, transmission, reception.

I. OBJETIVOS

Diseñar un dispositivo capaz de medir diferentes variables biomédicas de forma simple, por medio de una tarjeta de desarrollo (FPGA) para la evaluación del estado de salud actual de pacientes de cualquier tipo de edad.

- Desarrollar la descripción de hardware para la obtención y envío de datos biomédicos.
- Hacer uso de dispositivos externos para prototipo modular y amigable con el usuario.
- Implementar protocolos I2C y UART para la conexión con la tarjeta de desarrollo.

II. MARCO TEÓRICO

II-A. Protocolo UART

UART (universal asynchronous receiver / transmitter), define un protocolo para el intercambio de datos en serie entre dos dispositivos. Utiliza solo dos hilos entre el transmisor y el receptor para transmitir y recibir en ambas direcciones. La comunicación en UART puede ser simplex (los datos se envían en una sola dirección), semidúplex (cada extremo se comunica, pero solo uno al mismo tiempo), o dúplex completo (ambos extremos pueden transmitir simultáneamente). En UART, los datos se transmiten en forma de tramas.

Una de las mayores ventajas de UART es que es asíncrono: el transmisor y el receptor no comparten la misma señal de reloj. Puesto que no comparten un reloj, ambos extremos deben transmitir a la misma velocidad, previamente concertada, con el fin de mantener la misma temporización de los bits. Además de tener la misma velocidad en baudios, ambos

extremos de una conexión UART deben utilizar también la misma estructura y parámetros de trama.

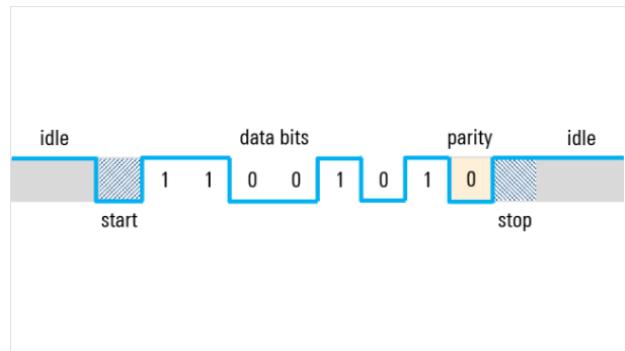


Figura 1: Trama UART

Teniendo en cuenta que el protocolo UART no define tensiones específicas o rangos de tensión para estos niveles, a veces se denomina al nivel alto «marca» y al bajo «espacio». En el estado de reposo (cuando no se transmiten datos) la línea se mantiene en el estado alto. Esto permite detectar con facilidad una línea o un transmisor averiado.

II-B. ESP32

Es la denominación de una familia de chips SoC de bajo costo y consumo de energía, con tecnología Wi-Fi y Bluetooth de modo dual integrada. El ESP32 emplea un microprocesador Tensilica Xtensa LX6 en sus variantes de simple y doble núcleo e incluye interruptores de antena, balun de radiofrecuencia, amplificador de potencia, amplificador receptor de bajo ruido, filtros, y módulos de administración de energía. El ESP32 fue creado y desarrollado por Espressif Systems y es fabricado por TSMC utilizando su proceso de 40 nm. Es un sucesor de otro SoC, el ESP8266.

Algunas de las características del ESP32 incluyen:

- Procesador:
 - CPU: microprocesador de 32-bit Xtensa LX6 de doble núcleo (o de un solo núcleo), operando a 160 o 240 MHz y rindiendo hasta 600 DMIPS

- Co-procesador de ultra baja energía (ULP)
- Memoria: 520 KiB SRAM
- Conectividad inalámbrica:
 - Wi-Fi: 802.11 b/g/n
 - Bluetooth: v4.2 BR/EDR y BLE
- Interfaces periféricas:
 - 12-bit SAR ADC de hasta 18 canales
 - 2 × interfaces I²S
 - 2 × interfaces I²C
 - 3 × UART
 - Controlador esclavo SDIO/SPI
 - Controlador remoto infrarrojo (TX/RX, hasta 8 canales)
 - LED PWM (hasta 16 canales)

A continuación se encuentran las dos versiones diferentes de ESP32 utilizadas en el proyecto junto con sus pines correspondientes:

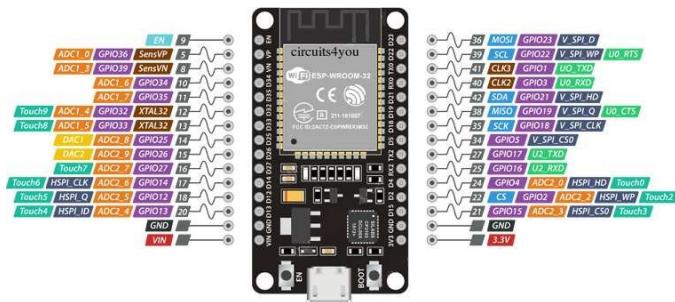


Figura 2: Pinout ESP32

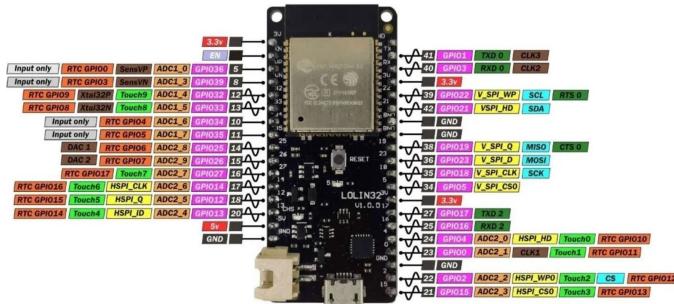


Figura 3: Pinout ESP32 LOLIN32

II-C. Módulo Bluetooth HC-05

Es un módulo Maestro-Esclavo, por lo que además de recibir conexiones desde una PC o tablet, también es capaz de generar conexiones hacia otros dispositivos bluetooth. El HC-05 tiene un modo de comandos AT que deben activarse mediante un estado alto en el PIN34 mientras se enciende (o se resetea) el módulo. En las versiones para protoboard este pin viene marcado como "Key". Una vez que se está en el modo

de comandos AT, se puede configurar el módulo bluetooth y cambiar parámetros como el nombre del dispositivo, password, modo maestro/esclavo, etc.

Características Hardware

- Compatible con Arduino
- Sensibilidad Típica: -80dBm.
- Hasta +4 dBm de potencia de transmisión RF.
- Fully Qualified Bluetooth V2.0 +modulación EDR 3Mbps.
- Funcionamiento de bajo consumo.
- PIO control.
- Interfaz UART con velocidad de modulación en baudios programable.
- Antena PCB Integrada.

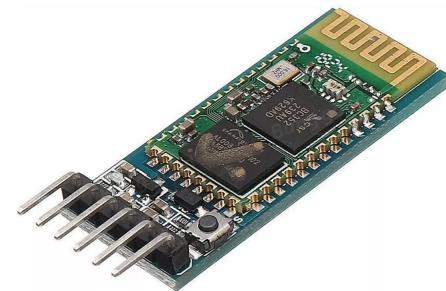


Figura 4: Módulo HC-05

II-D. Display 16x2

El tipo de pantallas más utilizadas son conocidas como displays de siete segmentos, dispositivos que muestran datos alfanuméricos y algún otro tipo de símbolo o imagen. Este tipo de pantallas han evolucionado dando varios pasos hacia el frente convirtiéndose en los dispositivos LCD 16x2.

LCD son las siglas en inglés de Liquid Crystal Display, un tipo de dispositivo utilizado para la visualización de diferentes tipos de contenidos o información de manera gráfica. La segunda parte de este término, 16x2, se refiere a que la pantalla cuenta con dos filas, cada una con la capacidad para mostrar hasta dieciséis caracteres, símbolos o figuras, según su programación.

Las altas capacidades de estos dispositivos permiten mostrar todo tipo de información sin importar qué tipo de símbolos o caracteres sean, estando cada artefacto controlado por un microcontrolador que está programado para dirigir el funcionamiento y la imagen mostrada en la pantalla. Una de las ventajas al utilizar este tipo de artefactos es que tiene el mínimo consumo de energía o corriente eléctrica, además la programación es sumamente sencilla y es por lo general cargada por el fabricante, dependiendo del uso y tipo de equipo que se trate.

Características:

- 16 caracteres x 2 líneas
- Caracteres de 5x8 puntos
- Tamaño de carácter: 5.23 x 3 mm
- Puede mostrar letras, números, caracteres especiales, y hasta 8 caracteres creados por el usuario
- Backlight de LED color azul
- Caracteres color blanco
- Interface paralela. Puede operar en modo de 8 bits, o de 4 bits para ahorrar pines del microcontrolador
- Posee controlador KS0066U o equivalente on-board (compatible Hitachi HD44780)
- Voltaje de alimentación: 5 V

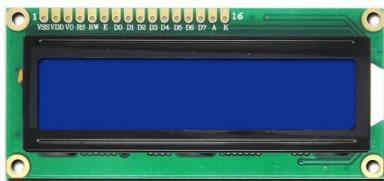


Figura 5: Display 16x2

II-E. Protocolo de comunicación serial I2C:

El protocolo de comunicación serial I2C es síncrono por lo cual necesita una señal de reloj generada por un dispositivo denominado maestro. Los otros dispositivos en el bus I2C son los denominados esclavos (generalmente sensores, memorias, etc.) los cuales no pueden generar una señal de reloj durante la comunicación. Generalmente hay un solo maestro en el bus, pero el protocolo permite que hayan más en tanto no traten de comunicarse al tiempo.

La comunicación se da a través de dos pines llamados SDA (serial Data) y Scl (serial clock), todos los dispositivos en el bus I2C tienen dos pines con estos nombres, y al momento de realizar la conexión se deben colocar todos los SDA y los SCL juntos. Durante la comunicación se encuentra

- bus inactivo: el maestro coloca ambos pines en alto.
- condición de inicio: el maestro baja el pin de SDA mientras SCL esta en alto, luego baja el SCL y comienza a mandar por él la señal de clock.
- envío de dirección de esclavo: El maestro manda un byte con la dirección de esclavo, seguido de un bit de escritura o lectura (0 escribir, 1 leer).
- confirmación de mensaje: El dispositivo que recibe la información coloca la línea de SDA en 0 durante un ciclo de reloj, para confirmar que recibió el mensaje.
- byte de dirección de registro: el emisor envía la dirección de un registro interno en el receptor, en este registro es donde leerá o escribirá.

- byte de información: El emisor envía un Byte con la información solicitada.
- bit de no confirmación: es un bit en alto luego de recibir un byte, con él el maestro prepara la condición de parada para finalizar la comunicación.
- bit de inicio repetido: es una replica de la condición de inicio, la cual se utiliza si se desea continuar con la comunicación pero cambiando el bit de lectura o escritura.
- condición de parada: la realiza el maestro subiendo el pin de SDA mientras SCL esta en alto, todos los esclavos entienden que se terminó la comunicación y el bus queda libre.

La trama de datos se aprecia en la siguiente imagen:

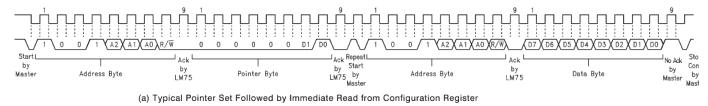


Figura 6: Trama I2C

II-F. Sensor de temperatura LM75

El sensor de temperatura de gama media alta fabricado por Texas Instruments, es usado con frecuencia para controlar la temperatura en varios dispositivos y sistemas embebidos debido a su buena incertidumbre(mas o menos 2°C).

Este sensor utiliza el protocolo de comunicación serial I2C para enviar los datos de temperatura a un maestro, para ello utiliza 9 bits en complemento a 2 siendo el bit menos significativo un 0.5°C(de estar en alto), es decir tiene una resolución de 0.5°C.A continuación se muestra un mapa de los registros internos del LM75 , un diagrama de pines y una imagen del sensor:

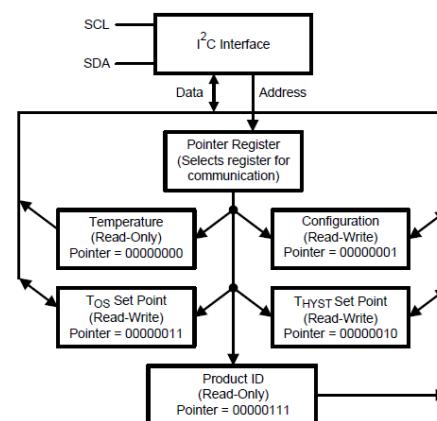


Figure 8. Register Structure

Figura 7: Mapa registros LM75

5 Pin Configuration and Functions

8-Pins SOIC (D) and VSSOP (DGK) Packages Top View

PIN NO.	NAME	DESCRIPTION	TYPICAL CONNECTION
1	SDA	I ² C Serial Bi-Directional Data Line, Open Drain	From Controller, tied to a pullup resistor or current source
2	SCL	I ² C Clock Input	From Controller, tied to a pullup resistor or current source
3	O.S.	Overtemperature Shutdown, Open Drain Output	Pull-up Resistor, Controller Interrupt Line
4	GND	Power Supply Ground	Ground
5	A2		
6	A1	User-Set I ² C Address Inputs	Ground (Low, "0") or +V _S (High, "1")
7	A0		
8	+V _S	Positive Supply Voltage Input	DC Voltage from 2.7 V to 5.5 V 100-nF bypass capacitor with 10-μF bulk capacitance in the near vicinity

Figura 8: Diagrama pines LM75

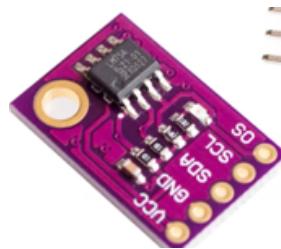


Figura 9: Diagrama pines LM75

III. SOFTWARE - MATERIALES

- FPGA.
- Quartus - design software.
- Sensor de temperatura LM75
- Jumpers
- Resistencias de 1k
- ESP32
- Sensor MAX30102
- Lector RFID

IV. IMPLEMENTACIÓN

A continuación se describirá a detalle la elaboración de cada etapa implementada en el proyecto.

IV-A. Lector RFID y transponder:

Siendo ésta la primera etapa del dispositivo, se realiza la identificación del usuario por medio de un tag o transponder (Tarjeta de identificación), la cual al ser leída por el Lector RFID, se habilitará la recepción y transmisión de datos del dispositivo bluetooth maestro al bluetooth esclavo, asociando todo tipo de variable biomédica obtenida al usuario registrado en el tag. Si llega a situarse en el Lector un transponder con información no válida, se notificará al usuario a través de la comunicación con el computador. A continuación se presenta la conexión del Lector RFID con el ESP32:

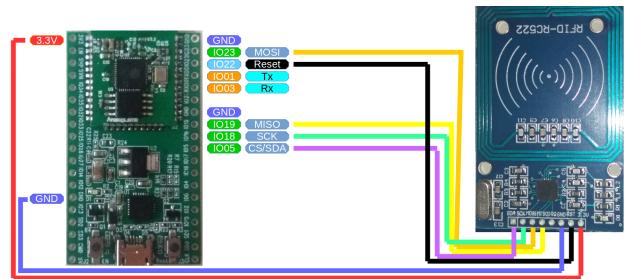


Figura 10: Conexión lector RFID y el ESP32

IV-B. Sensor MAX30102:

Este sensor es el encargado de obtener los datos de las variables biomédicas, es decir, la temperatura, la oxigenación y las pulsaciones por minuto de los pacientes, las cuales serán enviadas posteriormente al modulo Bluetooth maestro por medio de un protocolo de comunicación UART. A continuación se presenta la conexión del sensorMAX 30102 con el ESP32:

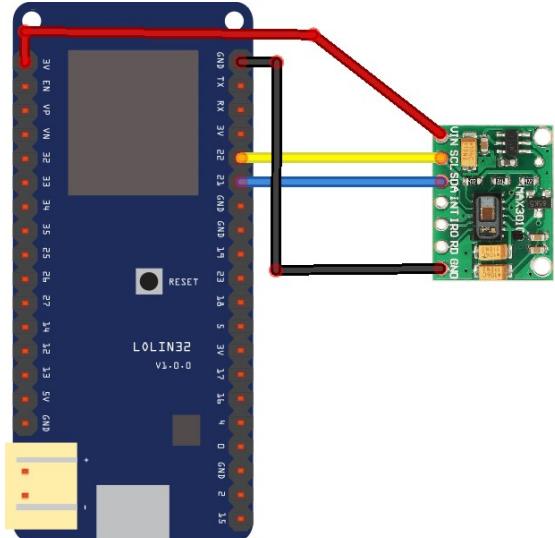


Figura 11: Conexión sesor MAX30102 y el ESP32

IV-C. EPS32 - Bluetooth (Maestro y esclavo):

Para el módulo bluetooth tanto maestro como esclavo hacen uso del sistema embebido ESP32, en donde se recibirá la información del módulo MAX30102 por el maestro, y por medio de protocolo UART se envía la información correspondiente al módulo bluetooth esclavo, el cual se encargará de habilitar el flujo de datos desde el maestro en el caso en el que sea utilizada una tarjeta correcta en el lector RFID. Por otro lado, se encargará de redireccionar los datos a la FPGA una vez sean obtenidos del maestro.

Para esta implementación se tomó como base el siguiente diagrama de estados:

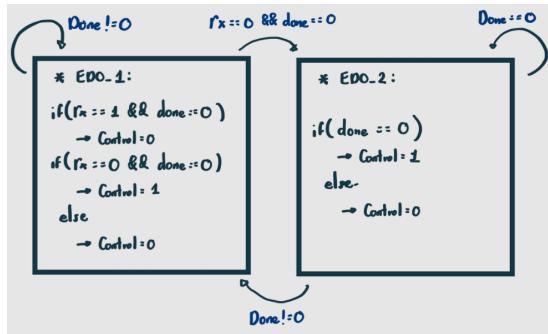


Figura 12: Conexión sesor MAX30102 y el ESP32

La función principal de esta máquina de estados es la de modificar una variable de control que es la que comanda el proceso de recepción de datos. Se poseen dos estados principales:

- EDO_1: En el caso de que detecte alteraciones en el estado del pin definido como Rx modifica la variable de control para habilitar la recepción de los datos, en caso contrario se mantiene en dicho estado.
- EDO_2: Posterior a un conteo de bits dictado por el protocolo UART, modifica de nuevo la variable de control y da por finalizada la transmisión del paquete de datos.

IV-D. LCD - Display:

Se eligió como dispositivo de visualización la pantalla LCD de 16 x 2 bits y se decidió disponer de esta interfaz por medio de dos formas:

Inicialmente se encuentran unos caracteres fijos que corresponden a la indicación de cada una de las medidas, es decir, estos espacios no variarán en ningún momento durante el tratamiento de la información del paciente; por otro lado, se encuentran los espacios dispuestos de manera dinámica, por lo que cambiarán por medio del refresco del display una vez se realice la toma de datos.

X	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	B	:	P	M	:	N1	N2	N3	▼	S	A	T	:	N4	N5	%
2	T	:	E	M	P	:	N6	N7	,	N8	°	I	D	:	N9	N10

Caracteres fijos.
Caracteres dinámicos.

Figura 13: Diseño y disposición de caracteres - Display LCD 16 x 2

Para los caracteres fijos se encontró su equivalente en código ASCII, ya que de esta forma es que se codifican los datos en la memoria interna del display. Dado que la transmisión de datos se realizó mediante 4 pines se dividieron los registros en 2 respetando la significancia de cada bit al momento de enviar la información.

Por otra parte, para los caracteres dinámicos se repitió el proceso de codificación, con la diferencia de que se crearon registros en el código en los cuales se iban recibiendo los datos que se encontraban como salida en el módulo bluetooth, estos registros se actualizaban conforme llegaban nuevos paquetes de información correspondientes a la variable medida.

Por último, se implementó un botón incluido en la tarjeta de desarrollo para el refresco del display, esto se dispuso como un periférico de entrada teniendo en cuenta que se desea visualizar la información una vez se capturen los datos y mantenerlos para una posible interpretación.

Para llevar acabo esta disposición se realizó una máquina de estados:

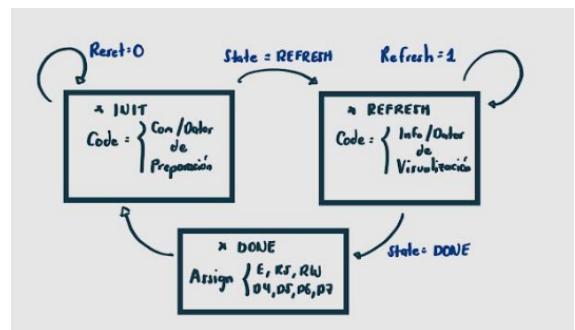


Figura 14: Diseño y disposición de caracteres - Display LCD 16 x 2

Se tienen tres estados principales de operación cuyas funciones son las siguientes:

- INIT: En este estado se habilita y se dispone la pantalla para poder modificar los registros en su memoria interna.
- REFRESH: En este estado se reciben de manera continua los registros asociados a los caracteres dinámicos, de esta forma se mantiene en perpetuidad hasta que se habilita la asignación de los valores mediante el refresco.
- DONE: En este estado se asignan los datos recibidos a los registros de la memoria de la pantalla para así visualizarlos.

IV-E. Interfaz I2C:

La elaboración de esta interfaz se realizó mediante el diseño de una maquina de estados finita (FSM) basada en la trama de datos del protocolo que se mostró en el marco teórico. El diagrama de transición de estados de dicha maquina es el siguiente:

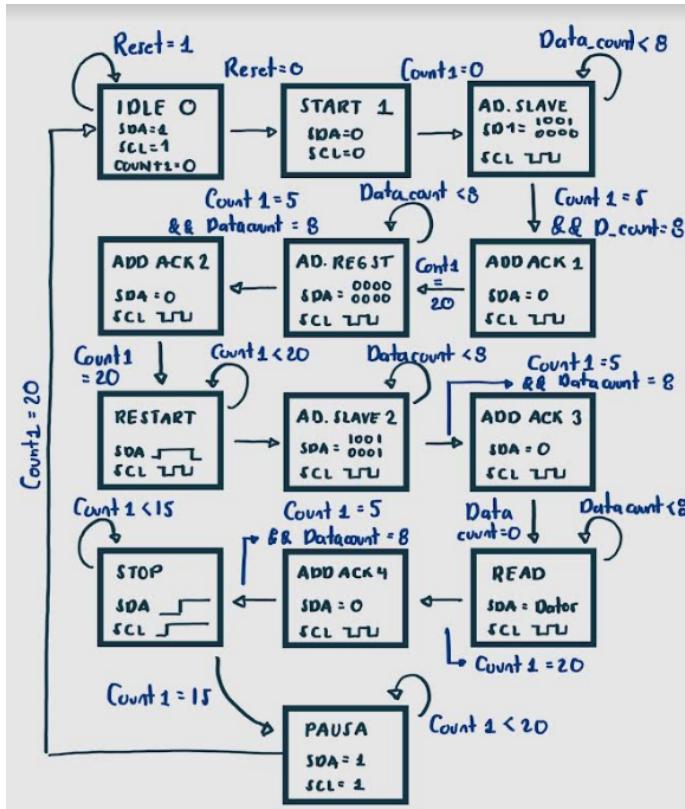


Figura 15: Diseño y disposición de caracteres - Display LCD 16 x 2

La máquina consta de 12 estados cuyo funcionamiento se describe de manera general:

- IDLE: SDA y SCL están en alto.
 - START: el maestro baja el pin de SDA mientras SCL está en alto, luego baja el SCL y comienza a mandar por él la señal de clock.
 - AddressSlave: El maestro manda un byte con la dirección de esclavo, seguido de un bit de escritura.
 - Addack1: El Esclavo coloca la línea de SDA en 0 durante un ciclo de reloj para confirmar que recibió el mensaje.
 - Adress Register: el Maestro envía la dirección de un registro interno en el esclavo (8'd0 en este caso), en este registro es donde leerá posteriormente la temperatura.
 - Addack2: El Esclavo coloca la línea de SDA en 0 durante un ciclo de reloj, para confirmar que recibió el mensaje.
 - Restart: El maestro sube SDA mientras SCL está en bajo, luego SCL también sube y quedan así un tiempo hasta que baje SDA para dar una nueva condición de inicio.

- AddressSlave2: El maestro manda un byte con la dirección de esclavo, seguido de un bit de lectura.
 - Addack3: El Esclavo coloca la línea de SDA en 0 durante un ciclo de reloj, para confirmar que recibió el mensaje.
 - Read: El esclavo manipula SDA y el Maestro lee mientras SCL esta en alto8 en toda la mitad, para dar tiempo de estabilización).
 - Addack4: El Maestro coloca la línea de SDA en 0 durante un ciclo de reloj, para confirmar que recibió el mensaje.
 - Stop: El Maestro coloca SCL en alto y luego sube SDA para generar la condición de parada.
 - Pausa: El maestro mantiene ambas líneas en alto durante cierto tiempo para dar oportunidad a los sensores de volver a realizar otra medición y actualizar su valor.

V. ANÁLISIS Y RESULTADOS

Como se mencionó anteriormente el proyecto consta de diferentes módulos, para los cuales se presenta a continuación sus resultados :

V-A. Lector RFID y transponder:

En general, se dispuso del lector de RFID bajo una conexión UART con el modulo ESP32, asociado al bluetooth esclavo del mismo. Los siguientes corresponden a las imágenes del lector RFID y las tag o tarjetas de identificación utilizadas en el proyecto.



Figura 16: Lector RFID



Figura 17: Lectores

Dentro de las partes más importantes del código para el RFID se encuentra la definición de los pacientes, para la cual se disponen los dos tags mostrados en el Figura anterior. En este caso específico se generaron dos usuarios, con sus respectivos mensajes de bienvenida, y por otro lado, un mensaje de advertencia en el caso en el que no se reconozcan ninguno de los usuarios creados anteriormente, para lo anterior se tiene como evidencia las figuras encontradas a continuación.

```
if(comparaUID(LecturaUID, Usuario1)){
    Serial.println("Bienvenido Usuario 01: Sr. Rodrigo Robles");
    enable = 1;
    ID = "01";
}
else if(comparaUID(LecturaUID, Usuario2)){
    Serial.println("Bienvenido Usuario 02: Sr. Samuel Mendez");
    enable = 1;
    ID = "02";
}
else{
    Serial.println("No te conozco");
    enable = 0;
}
```

Figura 18: Definición de pacientes.

```
Listo
The device started, now you can pair it with bluetooth!
```

Figura 19: Mensaje de espera para tarjeta de identificación

```
Listo
The device started, now you can pair it with bluetooth!
UID: EA 05 60 D3      Bienvenido Usuario 01: Sr. Rodrigo Robles
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
```

Figura 20: Mensaje de bienvenida Usuario 1

```
Listo
The device started, now you can pair it with bluetooth!
UID: 72 B1 0B 24      Bienvenido Usuario 02: Sr. Samuel Mendez
| 0 0 0 0 0 0 0 0 0 2
| 0 0 0 0 0 0 0 0 0 2
| 0 0
```

Figura 21: Mensaje de bienvenida Usuario 2

```
Listo
The device started, now you can pair it with bluetooth!
UID: 08 21 82 D7      No te conozco
```

Figura 22: Mensaje por usuario erróneo

Como vemos en las Figuras 17, y 18, en el momento en el que el usuario ha situado tu tarjeta en el lector, este empieza a enviar una paquete de 10 dígitos, lleno de ceros, excepto el último dígito que corresponde al ID del usuario, lo anterior mientras el bluetooth maestro empieza a enviar información, para lo cual empieza a llenarse el mismo arreglo pero ahora con la información correspondiente, como vemos en la siguiente figura.

```
Listo
The device started, now you can pair it with bluetooth!
UID: EA 05 60 D3      Bienvenido Usuario 01: Sr. Rodrigo Robles
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 0 0 0 0 0 0 0 0 1
| 0 6 4 9 9 2 5 1 0 1
| 0 6 1 9 7 2 5 9 0 1
| 0 6 4 9 7 2 5 8 0 1
|
```

Figura 23: Impresión datos obtenidos por Bluetooth esclavo

El paquete de información mencionado anteriormente se divide en subpaquetes, en donde los primeros 3 dígitos corresponden a las pulsaciones por minuto del paciente, los siguientes 2 a la saturación de oxígeno (0 %-99 %), los siguientes 3 a la temperatura, con un dígito decimal, y los últimos dos a la identificación del usuario mencionada con anterioridad.

V-B. Sensor MAX 30102:

Este sensor será el encargado de obtener las variables biomédicas de pulsaciones por minuto, temperatura y saturación de oxígeno del paciente, variables que posteriormente serán enviadas al modulo ESP32, que actúa de bluetooth

maestro del sistema. A continuación se presenta el sensor MAX30102, junto con un fragmento de código, en el cual se muestra el envío de datos al puerto serial.

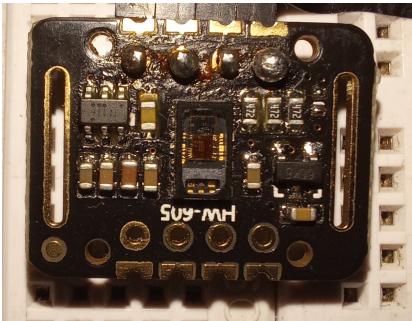


Figura 24: Visualización LCD previo a la recepción de datos.

```

Serial.print(F("HR="));
BPM = String(heartRate);
Serial.print(BPM);

Serial.print(F(", SPO2="));
SAT = String(spo2);
Serial.print(SAT);

TEMP = String(random(250, 260));
Serial.print(", Temp=");
Serial.print(TEMP);

ID = "ID";
Serial.print(", ID=");
Serial.println(ID);

```

Figura 25: Código Envío de datos obtenidos al puerto serial

```

always@(negedge capture) begin
    if (control)
        if(i >= 9) begin
            i = 0;
            done = 1;
            datos <= tmp[8:1];
            if(datos == 8'b0111_1100) enable <= 1;
            if(enable == 1'b1 & datos != 8'b0111_1100) begin
                case(cont)
                    4'b0000: BPM [23:16]      <= datos;
                    4'b0001: BPM [15:8]       <= datos;
                    4'b0010: BPM [7:0]        <= datos;
                    4'b0111: SAT [15:8]       <= datos;
                    4'b0100: SAT [7:0]        <= datos;
                    4'b0101: TEMP [23:16]     <= datos;
                    4'b0110: TEMP [15:8]      <= datos;
                    4'b0111: TEMP [7:0]       <= datos;
                    4'b1000: ID [15:8]        <= datos;
                    4'b1001: begin ID [7:0] <= datos; enable <= 0; end
                endcase
                cont <= cont + 1;
                if(cont >= 4'b1001) begin cont <= 4'b0000; end
            end
        else begin
            i = i+1;
            done = 0;
        end
        else done = 0;
    end
endmodule

```

Figura 26: Visualización LCD previo a la recepción de datos.

Su funcionamiento a grandes rasgos radica en un contador que actúa según el protocolo UART cuantificando los bits que llegan por la linea de recepción, por otra parte una vez se envía el byte completo se modifican los registros y se envian por medio de paquetes según la variable de medición al módulo LCD.

V-C. FPGA - Bluetooth (Modulo de procesamiento de datos):

Para este módulo, se implementó la maquina de estados tratada anteriormente, no obstante se realizó además de ello un proceso de recepción y empaquetamiento de datos conveniente para la visualización en el display LCD como se muestra a continuación.

V-D. ESP 32 - Bluetooth (Maestro):

Este modulo es el encargado de la recepción de los datos biomédicos y el envío al bluetooth esclavo. A continuación se presenta la conexión final junto con el código para el envío de datos al bluetooth esclavo, para lo anterior se creó un serial alternativo llamado *SerialBT* el cual será el encargado del flujo de información entre los dispositivos bluetooth.

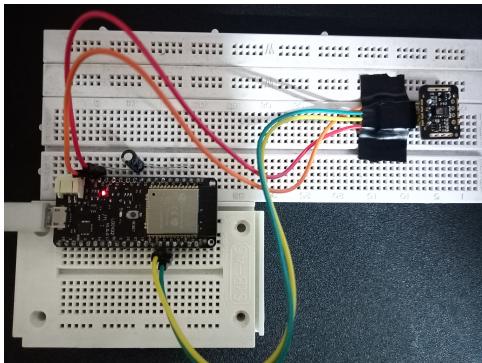


Figura 27: Sensor



Figura 29: Conexión ESP32 y sensor

```

if(validHeartRate && heartRate > 40 && heartRate < 120 && BPM.length() == 2 && validSPO2 && spo2 > 50){
    SerialBT.write('1'); //Bandera
    SerialBT.write(BPM[0]);
    SerialBT.write(BPM[1]);
    SerialBT.write(SAT[0]);
    SerialBT.write(SAT[1]);
    SerialBT.write(TEMP[0]);
    SerialBT.write(TEMP[1]);
    SerialBT.write(TEMP[2]);
    SerialBT.write(ID[0]);
    SerialBT.write(ID[1]);
}

else if(validHeartRate && heartRate > 40 && heartRate < 120 && BPM.length() == 3 && validSPO2 && spo2 > 50){
    SerialBT.write('1'); //Bandera
    SerialBT.write(BPM[0]);
    SerialBT.write(BPM[1]);
    SerialBT.write(BPM[2]);
    SerialBT.write(SAT[0]);
    SerialBT.write(SAT[1]);
    SerialBT.write(TEMP[0]);
    SerialBT.write(TEMP[1]);
    SerialBT.write(TEMP[2]);
    SerialBT.write(ID[0]);
    SerialBT.write(ID[1]);
}

```

Figura 28: Código para envío por SerialBT al bluetooth esclavo

En el código anterior, se usaron condicionales de validación para que en el momento que el conjunto de variables leídas tengan un valor más estable, estas sean enviadas por el serialBT al otro módulo bluetooth. Al igual que se condiciona el hecho de tener variable con 2 o 3 cifras en su representación.

```

if (SerialBT.available()) {
    dato = SerialBT.read();
    if(dato == '|'){
        Serial.println();
        Serial.print(" ");
        Serial.print(dato);
        Serial2.write(dato);
    }
    else if(dato == 'I'){
        Serial.print(" ");
        Serial.print(ID[0]);
        Serial2.write(ID[0]);
    }
    else if(dato == 'D'){
        Serial.print(" ");
        Serial.print(ID[1]);
        Serial2.write(ID[1]);
    }
    else{
        Serial.print(" ");
        Serial.print(dato);
        Serial2.write(dato);
    }
    delay(1000);
}

```

Figura 30: Asignación de valores para mostrar en pantalla y envío a FPGA.

V-E. ESP 32 - Bluetooth (Esclavo):

V-F. FPGA:

Una vez el módulo bluetooth esclavo reconoce la información del maestro, este procederá a enviarla de nuevo por un puerto creado para el flujo de datos entre la FPGA y el esclavo, llamado *Serial2*. A continuación, se presenta la imagen del módulo ESP32 o bluetooth esclavo, junto con un fragmento de código que muestra el flujo de datos a la FPGA.

Dado que se trabajó de manera modular, el código principal del proyecto tiene como función únicamente interconectar módulos por medio de wires y definir entradas, salidas y registros según sea necesario como se muestra a continuación.

```

module satempox_prototype(CLOCK_50, reset_bluetooth, rx, delay, capture, reset_display,
    input CLOCK_50; //50 MHZ
    // VARIABLES MÓDULO BLUETOOTH
    input reset_bluetooth;
    input rx;
    output delay;
    output capture;
    // VARIABLES MÓDULO LCD
    output reg[7:0] led;
    output RS;
    output E;
    output db_4;
    output db_5;
    output db_6;
    output db_7;
    output CLOCK_Freq;
    output aux_start;
    // VARIABLES DIVISOR DE FRECUENCIA
    reg [26:0] frecuencia = `d 960;
    // VARIABLES PAQUETES
    wire [7:0] datos;
    wire [23:0] BPM, TEMP;
    wire [15:0] SAT, ID;
    always @ (posedge CLOCK_50) begin
        led <= datos;
    end
    divisor_de_frecuencia(CLOCK_50, frecuencia, CLOCK_Freq);
    bluetooth_prueba(CLOCK_50, reset_bluetooth, rx, datos, delay, capture, BPM, SAT, TEMP, ID);
    display(CLOCK_50, reset_display, BPM, TEMP, SAT, ID, RS, RW, E, db_4, db_5, db_6, db_7);
endmodule

```

Figura 31: Visualización LCD previo a la recepción de datos.

V-G. LCD - Display:

En las siguientes figuras se muestran ejemplos de las tres posibles visualizaciones del display, cada una actuando de manera coherente a la etapa de procesamiento de la información que corresponda.



Figura 32: Visualización LCD previo a la recepción de datos.



Figura 33: Visualización LCD posterior a la recepción de datos (Sin lectura desde el sensor).



Figura 34: Visualización LCD posterior a la recepción de datos (Con lectura desde el sensor).

En cuanto a la forma en la que se envían los datos a la LCD, se realizó una disposición de bits como la que se muestra a continuación.

```

46:   code <= {2'b10, 4'b0101};           //S
47:   code <= {2'b10, 4'b0011};           //S
48:   code <= {2'b10, 4'b0100};           //A
49:   code <= {2'b10, 4'b0001};           //A
50:   code <= {2'b10, 4'b0101};           //T
51:   code <= {2'b10, 4'b0100};           //T
52:   code <= {2'b10, 4'b0011};           //T
53:   code <= {2'b10, 4'b1010};           //T
54:   code <= {2'b10, SAT[15:12]};         //SAT - DATO 1
55:   code <= {2'b10, SAT[11:8]};         //SAT - DATO 1
56:   code <= {2'b10, SAT[7:4]};          //SAT - DATO 2
57:   code <= {2'b10, SAT[3:0]};          //SAT - DATO 2

```

Figura 35: Visualización LCD posterior a la recepción de datos (Con lectura desde el sensor).

En este caso se realizó una concatenación para hacer mas evidente la labor de cada serie de bits; los dos primeros encargados de definir la función que se va a realizar con la siguiente cadena de bits enviados, y la segunda enviar la información pertinente por los cuatro pines de comunicación.

En el ejemplo se puede observar que los bit 10 corresponden a la escritura en los registros de la LCD y los 4 bits que proceden corresponden al código ASCII del carácter que se desea visualizar.

VI. DIFICULTADES PRESENTADAS

VI-A. Conexión entre los dispositivos:

A lo largo del desarrollo del proyecto, se tuvieron diferentes dificultades para la conexión de los dispositivos, ya que las velocidades de recepción y transmisión de datos variaban de acuerdo al dispositivo, por ejemplo, en el caso de la recepción de los datos de la FPGA fue necesario reducir la velocidad de transmisión y pasar 9600bits/seg a una de 8bits/seg para que la LCD conectada a la FPGA tuviera un refresco de datos adecuado.

VI-B. Información reducida:

En el momento de buscar información respecto a la posible conexión de los módulos, velocidad de transmisión, códigos abiertos, esta fue muy limitada, debido la complejidad del proyecto.

VI-C. Interfaz I2C:

En cuanto a la implementación de este protocolo la principal dificultad que encontramos fue tener que desarrollar la máquina de estados finita desde cero, ya que la información que se encontraba en internet e inclusive en libros no fue suficiente. Una vez se comenzó a desarrollar la FSM encontramos problemas por la complejidad del protocolo, ya que este es síncrono y está diseñado para conectar una gran cantidad de dispositivos a un mismo bus de datos usando solo dos pines del maestro y por esta razón las tramas de datos son largas y complicadas de implementar.

Pese a todo esto, el problema que más tiempo costó solucionar fue la asignación de pin in-out de SDA, ya que este es bidireccional, y eso hacia que muchas veces se sobre escribieran datos en la línea, problema que finalmente se solucionó investigando y utilizando la configuración de alta impedancia para colocar el pin como una entrada, y asignándole datos para colocarlo como una salida.

VII. CONCLUSIONES

En general, la FPGA como dispositivo principal para la recepción y transmisión a través de módulos, a una escala mayor, como es el caso en el presente proyecto, hace que la complejidad así mismo aumente de manera considerable, ya que el manejo de la información bit a bit, su transmisión y recepción, incluyendo la diversa gama de módulos posibles a conectar y la interconexión entre todos es una tarea ardua y que requiere muchos conocimientos de la electrónica digital.

Por otro lado, se evidenció la importancia de las máquinas de estado para la descripción de Hardware en la medida que permiten abstraer un amplia panorama de situaciones de una forma sencilla e intuitiva, como se evidencia en los módulos I2c, Bluetooth y la LCD. Lo anterior, teniendo especial cuidado con las asignaciones paralelas de las variables al trabajar con máquinas de estado.

Finalmente, la interconexión modular de diferentes dispositivos electrónicos hace posible el diseño de prototipos completos para la obtención de variables reales, como en este caso, la obtención de datos como la temperatura, pulsaciones por minuto y saturación, dando una panorama más acertado del estado de salud de los pacientes, mejorando la calidad de vida de las personas que posiblemente interactuarán con este prototipo.

REFERENCIAS

- [1] Qué es UART. (s.f.). Branchenführende Technologiekompetenz | Rohde & Schwarz. https://www.rohde-schwarz.com/es/productos/test-y-medida/osciloscopios/educational-content/que-es-uart_254524.html
- [2] (s. f.). Wi-Fi & Bluetooth MCUs and AIoT Solutions I Espressif Systems. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- [3] .ESP32”[En línea]. Tomado de: https://www.rohde-schwarz.com/es/productos/test-y-medida/osciloscopios/educational-content/que-es-uart_254524.html
- [4] “Máquina de Estados en Ruby on Rails con AASM,” Rubentejera.com, 06-Sep-2018. [En línea]. Tomado de: <https://rubentejera.com/maquina-de-estados-en-ruby-on-rails-con-aasm/>
- [5] “16x2 LCD Interface with Spartan6 FPGA Project Kit” Pantech Solutions. [En línea]. Tomado de: <https://www.pantechsolutions.net/16x2-lcd-interface-with-spartan6-fpga-project-kit>
- [6] “ESP32 With RFID: Access Control” Fernando Koyanagi - instructables. [En línea]. Tomado de: <https://www.instructables.com/ESP32-With-RFID-Access-Control/>

[1] Qué es UART. (s.f.). Branchenführende Technologiekompetenz | Rohde & Schwarz. https://www.rohde-schwarz.com/es/productos/test-y-medida/osciloscopios/educational-content/que-es-uart_254524.html