Prototipo de Sistema de Monitoreo de Pacientes en Cama Mediante Tecnología Wearable e IoT (Julio 2017)

Fátima Vásquez
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco, Soyapango, El Salvador
ivonvasquez29@gmail.com

Tania Martínez
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco, Soyapango, El Salvador
tania.martinez@udb.edu.sv

Pedro Álvarez
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco, Soyapango, El Salvador
pedro.alvarez@udb.edu.sv

***Abstract*—This article describes a prototype of system to monitor the temperature and movements of patients in bed through a bracelet that sends information over the Internet, this information can be accessed through a web page that shows the current temperature of the patient, a historical data table and chart, and alarms in case of detecting temperature outside of normal range, or if it is detected that the patient is moving too much that indicates discomfort and sleep problems, in addition these alarms are also sent to the caregiver by e-mail.**

***Index Terms*—Biomedical telemetry, Accelerometers, Internet of Things, Temperature sensor, Wearable, WIFI.**

I. INTRODUCCIÓN

La monitorización remota de parámetros fisiológicos tal como se indica en [1] existe hace varios años pero había estado limitada por el tamaño, capacidad computacional, consumo energético y costos de los dispositivos necesarios para llevarla a cabo, sin embargo los avances tecnológicos de esta última década en el área de sensores, sistemas embebidos y tecnologías de la información hacen posible cada vez más que se pueda contar con dispositivos que pueden medir una gran variedad de parámetros fisiológicos a través de dispositivos de menor tamaño y bajo consumo energético que puedan agregarse en objetos cotidianos para comodidad del paciente (tecnología wearable), y que además envíen la información automáticamente por medios inalámbricos sin que el paciente tenga que hacer nada y que esta información esté disponible para las personas interesadas en cualquier parte del mundo.

Como se indica en [2] es un reto para la comunidad biomédica brindar soluciones para los pacientes que no solo sean funcionales sino que también puedan ser accesibles en cuanto a costos para la mayoría de las personas.

En este artículo se presenta un prototipo de bajo costo para el cuidado remoto de pacientes en cama mediante la monitorización de la temperatura y de sus movimientos.

La temperatura es un parámetro fisiológico muy importante de medir pues como se indica en [3] el ser humano al igual que los demás animales de sangre caliente tiene una temperatura corporal determinada para proteger sus órganos vitales. Esta temperatura está regulada por un mecanismo similar a un termostato que se encuentra en una parte del cerebro, el hipotálamo, y que se dispara cuando detecta algún factor que puede dañar órganos vitales.

Con el dispositivo creado no solo se muestra la temperatura actual del paciente sino que se dispone de una tabla y gráfica donde puede observarse el comportamiento de la temperatura a lo largo del tiempo, además se cuenta con una alarma si se detecta una subida o bajada de temperatura que ponga en riesgo la salud del paciente.

Además de la temperatura también se mide el movimiento del paciente, si se determina que es mucho para alguien que debe estar en reposo en su cama, se activa una alarma para alertar al cuidador.

El prototipo de bajo costo creado consta de un sensor de temperatura, un sensor de movimiento, un chip WIFI y una batería que alimenta los dispositivos, todo esto integrados en un brazalete que el paciente debe portar y que se comunica a través de internet, la información que brinda el brazalete puede ser accedida mediante una página web en cualquier parte del mundo, en esta se encuentra el valor actual de la temperatura del paciente, tabla y gráfica de datos históricos y las alarmas si se activaron, además el cuidador del paciente recibe vía email el aviso si se activa alguna alarma.

II. DISEÑO DEL SISTEMA Y ARQUITECTURA

Para el diseño conceptual del prototipo se utilizó una metodología de diseño top-down (de arriba abajo), inicialmente se concibió la idea de adquirir mediante Internet tanto la temperatura como el movimiento de un paciente en cama, además de la generación de alertas si se detectan valores anormales; para comodidad del paciente los elementos de medición y procesado de la información debían estar integrados en un artículo de uso cotidiano, se seleccionó un brazalete ya que permite más espacio para colocar los elementos. Luego de la idea general se empezó a trabajar en cada una de las partes y luego se integraron, en la Fig. 1 se muestra el diagrama de bloques del sistema y después se hace una descripción de cada una de las partes.

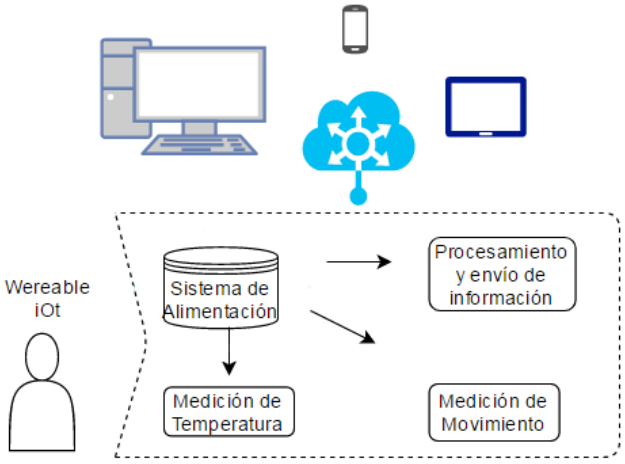

Fig.1. Diagrama de bloques del prototipo de sistema de monitorización para pacientes en cama.

A. *Medición de temperatura*

En esta etapa se utilizó el sensor DS18B20 que es un dispositivo que se comunica de forma digital con el chip WIFI. Esta comunicación la realiza a partir del dato que capta el sensor, este dato lo lee y se procesa a partir de un protocolo que utiliza el sensor que ya está predeterminada para la familia de *1-wire temperature sensors*, los cuales son de bajo costo y fáciles de usar, proporcionando lecturas de temperatura digital.

Este sensor utiliza la comunicación *OneWire*, que es un protocolo que permite enviar y recibir datos utilizando un solo cable, a diferencia de la mayoría de los protocolos que requieren dos cables [4].

.

El sensor tiene las siguientes características:

- Resolución de 9 y 12 bits
- Opera en un rango de -50 a 125 °C
- Precisión +- 0.5 grados
- Alimentación: 3.0 V a 5.5V
- Consumo de corriente: 4 mA Máx.

B. *Medición de movimiento*

La medición del movimiento se logra mediante el acelerómetro ADXL345 que es el encargado de detectar la posición del movimiento del brazo del paciente y después mandar este dato al chip WIFI.

La comunicación que utilizan el acelerómetro y el chip WIFI es I2C [5], esta comunicación permite ser mucho más rápida y efectiva ya que se puede conectar el microcontrolador con varios dispositivos y tener aun así una velocidad de envío de datos aceptable.

Las características del acelerómetro son las siguientes [7]:

- Comunicación I2C y SPI.
- Rango de medición de ajuste hasta +- 2, 4, 8 y 16g.
- Resolución 13 bits.
- Alimentación: 3.3 V a 5 V
- Consumo de corriente 40 uA a 100uA.

C. *Procesamiento y envío de información*

El elemento encargado de leer y procesar la información proveniente de los sensores es el módulo ESP-12 que se muestra en la Fig. 2, el cual consta de un microcontrolador ESP8266 el cual es un microcontrolador que contiene una solución para conexión WI-FI permitiendo ser un puente entre cualquier otro microcontrolador para que se conecte a Internet, pero también es capaz de correr aplicaciones por sí mismo.

A continuación se enlistan algunas de sus características:

- Consumo de voltaje: 3.3V
- Consumo de corriente: 10uA-170mA
- Memoria Flash: 16MB
- Procesador: Tensilica L106 32 bit
- Velocidad del procesador: 80-160 Mhz
- GPIOS: 17 (multiplexadas con otras funciones)
- RAM: 32k
- Maximum concurrent TCP connections: 5

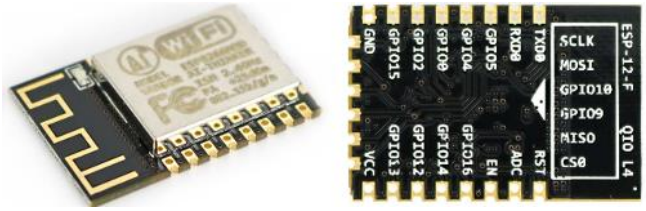

Fig. 2. Módulo ESP-12 de AI-Thinker.

El chip ESP8266 viene de fábrica para ser programado mediante comandos AT, sin embargo desarrolladores crearon un plugin [6] que posee un compilador y librerías para poder programar el ESP8266 con el IDE de Arduino prácticamente con los mismos comandos Arduino sin tener que utilizar más los comandos AT para ninguna de sus operaciones [8]. Simplemente se descarga el plugin en el IDE de Arduino como se indica en [10], se conecta los pines RX y TX a un conversor USB a FTDI UART o incluso usando una de las placas Arduino y ya puede programarse el ESP8266.

El ESP-12 lee el sensor de temperatura y movimiento y verifica que esté dentro del rango normal, si es así simplemente envía el valor de temperatura a la página web, si se detecta una temperatura fuera del rango o demasiado movimiento entonces además de enviar el dato a la página web envía la o las alarmas correspondientes tanto a la página web como al correo electrónico del encargado del paciente.

D. *Sistema de alimentación*

Tanto el ESP-12 como el acelerómetro y el sensor de temperatura necesitan una alimentación de 3.3V y cuando el chip WIFI envía la información a la nube se consume hasta un máximo de 250 mA, cuando el sistema está en reposo el consumo es mucho menor entre 70 y 80 mA. Para proveer el voltaje y corriente adecuados se utiliza una batería LIPO (Lithium-Ion Polymer battery) de 50x30x10 cms recargable con un valor de 3.7VDC 2000mAh, que se hace pasar por un regulador de voltaje LD33V[9] el cual da los 3.3V y la corriente adecuada para todo el sistema.

D. *Interfaz gráfica*

Se creó una interfaz gráfica para poder ver la información que envía el chip WIFI, para ello se utilizó código HTML y PHP, también se necesitó generar una base de datos MySQL para almacenar los datos y poder ver el comportamiento de la temperatura a lo largo del tiempo y de la librería php JpGraph [11] para ver esta tendencia en un gráfico, en la Fig. 3 Se muestra la interfaz creada la cual muestra el valor actual de temperatura y tiene dos botones para poder ver una tabla con los datos medidos a lo largo del tiempo (Fig. 4) y otro para ver estos mismos datos en un gráfico (Fig.5), además si la temperatura está fuera de rango o el movimiento detectado es inusual para un paciente que está en reposo en su cama se muestra una alerta en pantalla. Esta alerta también es recibida en el correo electrónico de la persona que cuida del paciente.

Fig. 3. Ventana de la interfaz gráfica.

Datos de temperatura almacenados

# de registro	Fecha y hora	Temperatura
1	17-05-18 (03:20:49pm)	36.2
2	17-05-18 (03:21:30pm)	36.3
3	17-05-18 (03:22:04pm)	36.2
4	17-05-18 (03:23:37pm)	36.1
5	17-05-18 (03:24:13pm)	36.3
6	17-05-18 (03:24:47pm)	37
7	17-05-18 (03:25:21pm)	38.94
8	17-05-18 (03:25:54pm)	37
9	17-05-18 (03:26:28pm)	38
10	17-05-18 (03:27:02pm)	36.06
11	17-05-18 (03:27:36pm)	37
12	17-05-18 (03:28:12pm)	36
13	17-05-18 (03:28:49pm)	36
14	17-05-18 (03:29:23pm)	36.2
21	17-05-18 (03:57:44pm)	35.38
22	17-05-18 (03:58:18pm)	35.63

Fig. 4. Tabla con datos de la temperatura a lo largo del tiempo.

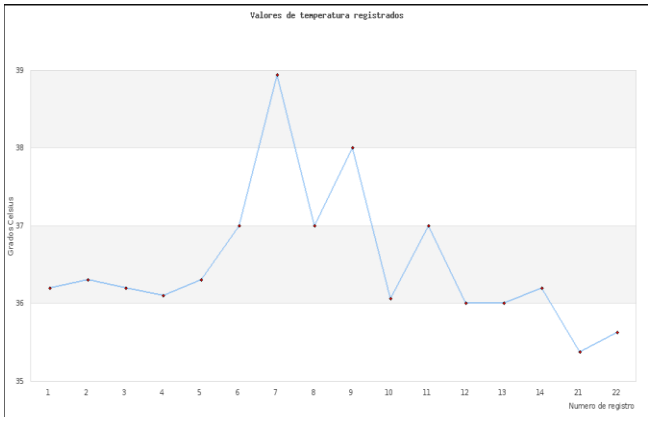

Fig. 5. Gráfico de los datos de temperatura a lo largo del tiempo.

III. IMPLEMENTACIÓN

Como se mencionó anteriormente el módulo ESP-12 es el encargado de leer los sensores, procesar los datos y enviarlos por Internet, para que su funcionamiento sea estable es necesario conectar resistencias de pull-up de 10k en los pines de reset, selección, GPIO0 y de pull-down de 10k en GPIO15, además de un capacitor de 100nF para desacople de la fuente de alimentación tal como se muestra en la Fig. 6.

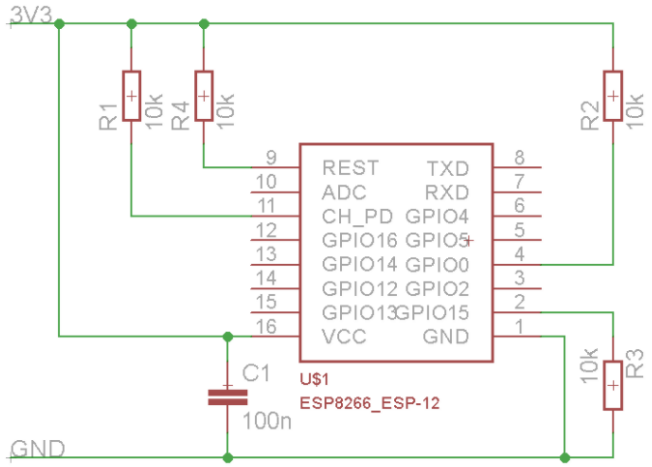

Fig. 6. Conexión del ESP-12 para un funcionamiento estable.

El sensor de temperatura DS18B20 se conecta al pin GPIO14 el cual es un pin de entrada/salida de propósito general del ESP8266, para leer el valor de este es necesario que se instalen en el IDE de Arduino las últimas versiones de las librerías *OneWire.h* y *DallasTemperature.h* ya que versiones antiguas puede que no tengan soporte aún para el ESP8266, luego en el código es necesario iniciar el sensor con la sentencia *sensors.begin()* y *sensors.requestTemperatures()* y para leer el valor colocar en un variable la sentencia *sensors.getTempCByIndex(0).*

Con respecto a la alarma, en [12] se indica que la temperatura normal en el ser humano se encuentra entre 36 y 37 grados, por lo que cuando el sensor detecta valores fuera de este rango por más de 5 minutos envía una alarma a la página web y un correo electrónico al cuidador del paciente.

El acelerómetro ADXL345 se conecta a los pines GPIO5 y GPIO4 que además de ser pines de entrada/salida de propósito general tiene la función de SCL (línea de los pulsos de reloj que sincronizan el sistema) y SDA (línea por la que se mueven los datos entre los dispositivos) para la comunicación I2C y por medio de la librería *Wire.h* se procesan los datos de la posición en que está el acelerómetro.

Con respecto a la alarma en el void loop se ejecuta un algoritmo que cada vez que el paciente se mueve demasiado de un rango predefinido se reiniciará el conteo del movimiento en el paciente y se mandará una alerta al cuidador o encargado del paciente que vaya a observar el porqué es que el paciente se mueve demasiado ver Fig. 7.

```
t=millis();// Medir actividad de paciente en 1 seg
 if (t-ta > 1000) {

    if (cx>=20 || cy>=20 || cz>=20){
    Serial.println (" Se esta moviendo mucho ");
    ta=t;
    cx=0;
    cy=0;
    cz=0;
```

```
x: 31 y: 64 z: 304
 cx: 13
 cy: 19
 cz: 16
 Se esta moviendo mucho
x: 32 y: 65 z: 303
 cx: 0
 cy: 0
 cz: 0
x: 38 y: 57 z: 309
```

Fig. 7. Fragmento de código que controla el movimiento del paciente y envío de mensaje de alarma.

El tiempo en que se efectúa cada conteo es de 1ms en donde permite ser preciso y continuo de contar el movimiento del paciente y luego almacenar este dato, para cuento cumpla el rango en que el paciente se ha movido 20 veces entonces mandar la alarma y volver a comenzar el conteo.

Para que el ESP8266 pueda enviar la información por WIFI es necesario incluir las librerías *ESP8266WiF.h*, *WiFiClient.h*, *WiFiClientSecure.h*, *WiFiServer.h* y *WiFiUdp.h*, activar el cliente WIFI, configurar las credenciales de la red WIFI a la que se va a conectar el chip, la dirección de la página web que recibirá los datos y finalmente enviar los datos y cerrar la conexión, esperar el tiempo indicado entre cada envío y el proceso se repite. En la Fig. 8 se muestra el fragmento de código donde se envía a la página web la información que consiste en el dato de temperatura y los valores de las alarmas de temperatura y de movimiento (valen 1 cuando están activadas y 0 cuando están desactivadas), en el código toda esta información va en la variable *data*.

```
//hacer el POST al subdominio
client.println("POST /enviodelesp.php HTTP/1.1");
//Indicar el Host y datos necesarios para la comunicacion
client.print("Host: electronicaymanufactura.000webhostapp.com\n");
client.println("User-Agent: ESP8266/1.0");
client.println("Connection: close");
client.println("Content-Type: application/x-www-form-urlencoded");
client.print("Content-Length: ");
client.print(data.length()); //data contiene el valor de temperatura y alarmas
client.print("\n\n");       //se envía el tamaño seguido de los valores
client.print(data);
```

Fig. 8. Fragmento del código donde se hace el envío de la información a la página web.

También se hace el envío de un correo electrónico a la dirección del cuidador del paciente si se detecta una anomalía de temperatura o movimiento, para ello se utilizó una librería llamada "Gsender.h" que puede encontrarse en [14], esta permite enviar el correo electrónico desde una cuenta de Gmail a otra, para ello es necesario colocar en la cuenta de Gmail remitente en las opciones de "Inicio de sesión y seguridad" que se permita el acceso de aplicaciones menos seguras, así es posible entrar simplemente ingresando usuario y contraseña que es como el ESP va a ingresar. En el código de la librería Gsender debe colocarse la dirección de correo Gmail remitente y la contraseña para entrar pero estas deben tener una codificación BASE64 para ello tenemos páginas como: https://www.base64encode.org/ que nos permiten obtener la codificación BASE64 de cualquier texto.

El archivo .h y cpp de la librería deben estar almacenados en la misma carpeta del código .ino que se descargará en el ESP. En este además de hacer un llamado a la librería con la sentencia *#include "Gsender.h"* es necesario colocar el puntero a la instancia de clase con la sentencia: *Gsender *gsender = Gsender::Instance()* y para enviar el correo es necesario colocar lo siguiente:

if(gsender->Subject("Colocar aquí asunto")->Send("nombre_del_destinatario@gmail.com", "texto que irá en el cuerpo del correo"))

La cuenta de destinatario no requiere que se le cambien las opciones de inicios de sesión.

El ESP-12 aumenta su consumo de corriente cuando arranca y cuando envía los datos por WIFI y se mantiene a un valor intermedio el resto del tiempo y si se coloca en modo SLEEP el consumo se reduce al mínimo por lo que para optimizar el tiempo de duración de la batería se hace entrar al chip en este modo entre cada envío de datos, para ello se coloca la siguiente sentencia en el programa *ESP.deepSleep(tiempo)*, donde la variable tiempo debe contener en microsegundos el valor de tiempo que se necesite que el chip entre en modo de sueño. Para despertarlo es necesario que reciba una señal en el pin de RESET por lo que este pin además de conectarse con una resistencia de 10k a VCC como se muestra en la Fig 6, debe conectarse también a uno de los pines de propósito general del chip como por ejemplo el GPIO16; en el código es necesario definir a este pin con la sentencia const *int pingpio16 = 16* y luego configurarlo como pin para despertar al chip con la sentencia *pinMode(pingpio16, WAKEUP_PULLUP)*, con esto al finalizar el tiempo de espera

el chip recibirá una señal de reset y se despertará automáticamente.

Para la alimentación del sistema se utilizó una batería que tiene un tamaño mediano, se buscaron alternativas con baterías de menor tamaño que podían comprarse localmente pero no cumplían con el requisito de corriente que el sistema necesita y no fue posible experimentar con baterías que se pudieran conseguir por Internet pues El Salvador no aparece en la lista de países autorizados para la compra de baterías de Litio y no se permitía la compra, por esta razón el brazalete resultó más voluminoso de lo que se esperaba pero si es posible que alguien lo tenga en su muñeca sin mayores problemas.

Para el alojamiento de la interfaz se utilizó el servicio de hosting gratuito de https://www.000webhost.com, el cual también permite la creación y administración de bases de datos. Se creó una base de datos con una tabla compuesta por cuatro campos: N° de registro, Fecha y hora, Temperatura, Alarma de temperatura y Alarma de movimiento para ir almacenando los datos que envía el ESP8266.

Se alojaron cuatro archivos .php y la librería JpGraph para la ejecución de la página web, el principal contiene código HTML incrustado para definir aspectos como color de fondo, imágenes, texto, botones y la configuración de la actualización automática cada cierto tiempo, luego está el código php para extraer de la base de datos el valor de temperatura y alarmas para ser mostrados en pantalla. Otro archivo .php es el encargado de recibir el dato desde el ESP8266 el cual hace un envío de la siguiente forma:

https://electronicaymanufactura.000webhostapp.com/enviodelesp.php?sensor=xx.xx&alarma1=x&alarma2=x

Este archivo lee los datos recibidos en la URL y los guarda junto con la fecha y hora actual en la base de datos para que luego sean extraídos por el programa principal para ser mostrados en pantalla.

Un tercer archivo .php es incluido para generar una tabla con todos los valores de temperatura registrados en el tiempo, este archivo se ejecuta cuando se da clic en el respectivo botón de la ventana principal. El cuarto archivo .php genera una gráfico de línea con los valores de temperatura registrados a lo largo del tiempo, pero para poder hacerlo es necesario que también se agreguen los archivos correspondiente a la librería JpGraph, al igual que con la tabla este archivo se ejecuta si el usuario da clic en el botón correspondiente de la ventana principal.

En la Fig. 9 se muestra el diseño completo del brazalete, la parte superior que contendrá el sensor de temperatura, acelerómetro y chip WIFI tiene un tamaño de 5.1x3.1x1.2cm y la parte de abajo que contendrá la batería tiene un tamaño de 5.5x3.9x1.6cm, el cable que va de la batería en la parte inferior al sistema de la parte superior pasa a través de la tela con velcro que une ambas partes..

El prototipo inicial implementado se muestra en la Figura 10, la pieza superior es de plástico y fue elaborada mediante impresión 3D, por cuestiones de tiempo no se alcanzó a fabricar la pieza que contiene la batería, pero en la imagen puede observarse el tamaño de la batería con respecto a la muñeca para tener una idea de cómo se verá el brazalete completo.

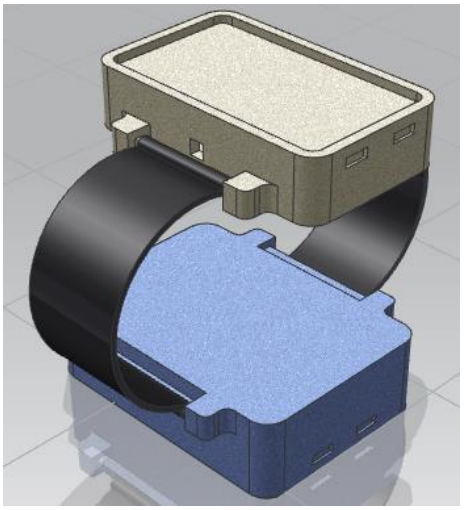

Fig. 9. Diseño del brazalete completo.

Fig. 10. Prototipo de inicial del brazalete.

IV. RESULTADOS

Al implementar el sistema completo fue posible obtener correctamente la medida de temperatura corporal y cuando se simuló un aumento o disminución de temperatura y este valor se mantuvo durante 5 minutos se disparó una alarma que se mostró en la página web y a la vez también se envió un correo electrónico con el aviso, de igual forma se obtuvieron resultados exitosos con la medición del movimiento, si se detecta que la persona cambia su posición muchas veces también se genera una alarma en la página web y se envía un correo con el aviso. En la interfaz se pudo observar el registro de los datos de temperatura a lo largo del tiempo a través de una tabla y de un gráfico.

V. CONCLUSIONES

El prototipo de dispositivo wearable creado en este proyecto fue capaz de medir y enviar el valor actual de la temperatura corporal de su portador a través de Internet para

ser visualizado en una página web en cualquier parte del mundo, además de generar alarmas para una atención más inmediata del cuidador del paciente y tener acceso a un historial de los datos.

El dispositivo puede mejorarse en el futuro, se puede reducir su tamaño si se consiguen sensores y sobre todo una batería más pequeña, resultando así en una alternativa más cómoda para el usuario que la actual, además puede mejorarse su funcionalidad ya que por el momento solo se conecta vía WIFI y si la red falla se pierde la comunicación, se espera en un futuro migrar a una comunicación a través de una red GSM (sistema global de comunicaciones móviles).

VI. REFERENCIAS

[1] M. Pattichis, A. Jossif, L. Paraskeva, A. Konstantinides y D. Vogiatzis, "An m-Health Monitoring System for Children with Suspected Arrhythmias", en Proc. 29th Annual International Conference of the IEEE EMBS, Lyon, Francia,2007, pp.1794-1797.

[2] A. Chaudhuri, A. Dasgupta, S. Chakrborty y A.Routray, "A Low-Cost, Wearable, Portable EOG Recording System", en Proc. 2016 International Conference on Systems in Medicine and Biology, India, 2016, pp. 103-105.

[3] Edit on GitHub, "Using the DB18B20 Temperature Sensor", [En línea]. Disponible en: https://goo.gl/w09ntH

[4] Rolando R. "DS18B20, Sensor de temperatura", [En línea]. Disponible en: https://goo.gl/uRav5x

[5] Eduardo J. Carletti, "Comunicación - Bus I2C", [En línea]. Disponible en: http://robots-argentina.com.ar/Comunicacion_busI2C.htm

[6] "Esp8266/Arduino", Github, s.f. [En línea]. Disponible en: https://github.com/esp8266/Arduino

[7] Análoga Devices, "datasheet ADXL345", [En línea]. Disponible en: https://goo.gl/X5bFVL

[8] S. Borsay, "Send ESP8266 Data to Your Webpage - no AT Commands!", Hackster, 2017. [En línea]. Disponible en: https://goo.gl/lZBNd7

[9] "LD1117 Series low fixed and adjustable positive voltage regulators", Sparkfun , s.f. [En linea] . Disponible en: https://goo.gl/3K8McW

[10] "ESP8266 Pluggin para Arduino IDE", Prometec, s.f. [En línea]. Disponible en: http://www.prometec.net/esp8266-pluggin-arduino-ide

[11] A. Lugo, "Gráficas con JpGraph: gráfica de línea", 2013. [En línea]. Disponible en: https://goo.gl/riR2pd

[12] Eroski Consumer, "Salud/Fiebre", Eroski Consumer, N°. 1588, pp. 18-21, 2011 [En línea]. Disponible en: https://goo.gl/Ew3jDm

[13] N. Kolban, Kolban's Book on ESP8266. 2016. [En línea]. Disponible en: http://neilkolban.com/tech/esp8266/

[14] B. Shobat. "ESP8266 GMail Sender", Instructables, s.f. [En línea]. Disponible en: http://www.instructables.com/id/ESP8266-GMail-Sender/

[15] K. Darrah, "Arduino WiFi (IoT) Tutorial - ESP8266 "Full Communication From Anywhere in the World" p3", 2015. [Vídeo]. Disponible en: https://www.youtube.com/watch?v=uWbLpMJ8jiA.