E.T.S. de Ingeniería de Sistemas Informáticos

Universidad Politécnica de Madrid

Memoria de Práctica Externa

<**CITSEM**>

<Jiliang Qiu>
<Sistmeas de información>
<4º>
<Fecha>

Introducción.

La práctica desarrollada se ha basado en la programación del reloj Bangle.js. Bangle.js es un reloj inteligente (smartwatch) de código abierto que se programa en javascript. Lo que distingue a Bangle.js de otros smartwatches es su enfoque en la programación y la personalización. La característica más importante de este producto es que es código abierto y es "hackable" por lo que se puede experimentar y hacer sus propios programas. La desventaja que hay es que está muy limitado en términos de potencia de hardware. Aun estando limitado por las capacidades su flexibilidad ya que es código abierto y se deja manipular libremente lo hace muy interesante. Además, se puede crear múltiples programas que tengan la misma funcionalidad que los relojes de alta gama.

El objetivo de mi práctica ha sido intentar implementar las funciones de mi reloj que es detectar la frecuencia cardiaca para así mandarlo a un videojuego terapéutico ya que con mi trabajo el juego podría saber si el ejercicio que está haciendo el usuario le está costando o no para así poder bajar la dificultad o subirla según su situación a tiempo real.

ÍNDICE DE CONTENIDOS

1. ACTIVIDADES Y COMPETENCIAS DESARROLLADAS.

El objetivo principal de la práctica es medir la frecuencia cardíaca del usuario a través del reloj bangle.js . Se inicia explorando el dispositivo descargando aplicaciones de otros creadores disponibles en la

página web oficial del reloj. Esto se hace para poder permitir familiarizarse con el dispositivo y su interacción así tener una primera toma de contacto con sus funcionalidades. Posteriormente, se desarrollan programas simples, como la obtención y la posibilidad de ver el tiempo y la temperatura. Durante esta fase, aparecen inconvenientes relacionados con la falta de información detallada sobre los métodos incorporados en el reloj, ya que la documentación oficial resulta ser limitada en este aspecto.

Una vez realizado el primer programa del reloj. Se plantea el siguiente objetivo que es de extraer información del dispositivo bangle.js. Aquí se enfrenta un problema que, aunque se supera en pocas horas, representa un obstáculo significativo. Se descubre que la solución está disponible en de la WEB IDE de Espruino, la cual permite programar directamente utilizando los métodos ya implementados en el dispositivo, algo que se había utilizado desde el inicio. Sin embargo, la limitación de la WEB IDE se sitúa en la imposibilidad de enviar datos en tiempo real, ya que solo permite la descarga manual de la información de la base de datos del reloj. Para superar esta limitación, se opta por configurar un servidor local, dado que el dispositivo Bangle.js no permite la bidireccionalidad. Aunque el ordenador esté conectado al reloj a través de Bluetooth para enviar programas, no puede transferir datos automáticamente al ordenador.

Esta experiencia proporciona un entendimiento capacidades del dispositivo y la de exploración de soluciones alternativas para superar limitaciones del dispositivo.

2. TÉCNICAS, TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS.

En total se ha programado en dos entornos diferentes la primera de ellas es WEB IDE y la segunda en visual studio code los dos principalmente en lenguaje javascript y en el último de ellos, en visual studio también se programó cierta parte en html.

Bangle.js es un dispositivo wearable que se destaca por su versatilidad y su enfoque en la programación y la creación de aplicaciones para dispositivos portátiles. Se trata de un reloj inteligente de código abierto, diseñado específicamente para los entusiastas de la programación, los desarrolladores y cualquiera interesado en la creación de software para wearables.

Lo que distingue a Bangle.js del resto es su capacidad de ejecutar JavaScript directamente en el dispositivo. Esto significa que los desarrolladores pueden escribir y cargar código directamente en el reloj así proporcionando más versatilidad, lo que lo convierte en una plataforma ideal para experimentar con el desarrollo de aplicaciones, juegos y otros proyectos interactivos.

Además, Bangle.js viene con una amplia gama de sensores ya instalados, como acelerómetro (nos ayuda a medir el movimiento en el espacio), monitor de frecuencia cardíaca, GPS y muchos más. Estos sensores permiten a los desarrolladores aprovechar diversas funcionalidades para crear aplicaciones que pueden rastrear la actividad física, medir la frecuencia cardíaca, registrar datos de movimiento etc.

El dispositivo está respaldado por una comunidad activa de desarrolladores como Gordon (una persona que pertenece a la empresa y está siempre muy activo ayudando a las personas que lo necesiten en foros) que contribuyen con mejoras y tutoriales. Esto significa que aquellas personas que entran en el mundo de Bangle.js tienen acceso a una gran cantidad de recursos en ayuda a su aprendizaje. Aunque alguna información sobre los metodos implementados no fuero faciles de encontrar.

Resumiendo Bangle.js no es un smartWatch cualquiera; es una plataforma de desarrollo versátil que ofrece la posibilidad de explorar la creación de aplicaciones y proyectos innovadores directamente en un dispositivo wearable. Su capacidad de ejecutar JavaScript en el propio dispositivo, combinada con una amplia gama de sensores integrados, lo convierte en una herramienta atractiva y poderosa para la experimentación y la creación.

3.PROBLEMAS PLANTEADOS Y PROCEDIMIENTOS SEGUIDOS PARA SU RESOLUCIÓN.

El primer problema como se ha comentado en el punto dos fue encontrar la información de los métodos ya implementados en la documentación de su página oficial no había esa información o estaba plasmada muy vagamente. La solución que se llego fue ir a programas ya hechos y ver su código ya que los programas para esta plataforma eran de código abierto se escogió las 5 más populares de frecuencia cardiaca y se estudió a partir su código incluyendo sus similitudes. Después de un estudio exhaustivo se pudo ver llamadas a métodos sin declaración previa. Así que se probó también y resultó ser cierto. A continuación, voy a dejar esos códigos a continuación:

Date() esta compuesto por horas minutos y segundos que se puede acceder como este ejemplo

```
let d = new Date();

var timeStr = d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds();
```

En el ejemplo lo que se hace es que convierto todos los datos para que se pueda leer en formato

HH:MM:SS

Para poder ejecutar el código se tiene que poner" Bangle.loadWidgets(); "

Para la detección de pulsos hay otros métodos.

```
{ "bpm": number,             // Beats per minute
  "confidence": number,      // 0-100 percentage confidence in the heart rate
  "raw": Uint8Array,         // raw samples from heart rate monitor
} To get this event you must turn the heart rate monitor on with Bangle.setHRMPower(1).
```

Figura 1 lo que contiene el objeto heart rate data

para llamarlo y poder manipularlo

h.bpm

h.confidence

h.raw

La detección del corazón hasta ahora solo se ha encontrado que funciona con "h"

Una vez finalizado el código, el siguiente paso fue extraer datos del dispositivo. Inicialmente, se estudiaron los códigos creados por otros usuarios para determinar si podían almacenar datos. Se descubrió que algunos si que lo hacían, aunque en cantidad limitada. Este descubrimiento impulsó un estudio más detallado. Sin embargo, al implementar estas soluciones, los datos no se obtenían. La solución se encontró al descubrir una página web que decía información sobre cómo acceder directamente al 'storage' o base de datos del dispositivo a través de la WEB IDE de Espruino. Utilizando el icono con tres círculos apilados, se logró navegar y visualizar los datos en el 'storage' del reloj, pero para ello se requería un código específico, que se detalla a continuación:"

```
var file = require("Storage").open("Datos.js","a");
```

"Declarar datos y almacenar datos en ella"

"función"

```
file.write(datos.join(",")+"\n");
```

Pero como se ha dicho en el punto dos del informe, este método fallaba algo. En que no transmitía información a tiempo real al ordenador para que el videojuego de terapia lo pudiese utilizar para modificar la dificultad del juego el usuario entonces había que encontrar la siguiente manera.

La siguiente solución que se propuso fue mover todo a Visual Studio Code ya que WEB IDE solo funcionaba en la web y no en el ordenador así que no podía escribir en ningún sitio. Desgraciadamente no se pudo implementar Espruino en Visual Studio Code y siempre daba errores. Entonces se tuvo que buscar en numerosos sitios como foros, videos de personas con intención similar hacerlo junto con la ayuda de profesionales sin tener éxito. Finalmente se encontró una página que lograba conectar con código html con el reloj Bangle.js esto lo cual solucionaba dos problemas: la primera de ellas es que soluciona que el reloj no tenga bidireccionalidad es decir mientras el ordenador le envía el código al reloj no puede mandar información de vuelta y también que con html si lo podemos codificar en visual studio.

Finalmente se encontró la solución y se procede a cambiar el código encontrado para implementar el nuestro básicamente se sustituye todas las funciones menos el “conect” cuando se implementó el programa en html surgió otro problema que no había forma de acceder a los datos ya que el protocolo “HTTP” no dejaba acceder a las variables y más tarde se encontroó una forma de extraer los datos incorrecto, incorrecto porque otra vez había que descargarlo manualmente hasta que se ocurrió la idea de hacer un back-end y un front-end (cliente-Servidor). Siendo el front el html que sea ha creado previamente y el back nuestro server.js (escrita en javascript) así con el servidor local encendido se podía extraer los datos y guardarlos.

Además de la aplicación de la frecuencia de pulsos también se implementó el movimiento de la muñeca, aunque ese programa solo se consiguió hacer en la WEB IDE Espruino.

4.APORTACIONES EN MATERIA DE APRENDIZAJE.

4.1 RESPECTO A LOS CONOCIMIENTOS ADQUIRIDOS.

En el transcurso de mi práctica con Java, HTML y Bangle.js he adquirido conocimientos valiosos más allá de la programación pura. He mejorado mi capacidad de resolución de problemas, aprendiendo a abordar desafíos complejos y a encontrar soluciones eficientes y a pensar out of the box. La experiencia en el desarrollo web me ha enseñado sobre prácticas de diseño y accesibilidad. Además, el trabajo con Bangle.js me ha permitido comprender mejor el mundo de la programación orientada a dispositivos físicos, lo que ha aumentado mi capacidad para trabajar en proyectos innovadores y aplicaciones con un enfoque en la Internet de las cosas (IoT).

A lo largo de mi trayecto, he mejorado significativamente en mi habilidad para buscar información de manera efectiva y eficiente.

4.2 RESPECTO A LAS COMPETENCIAS ADQUIRIDAS.

Durante mi experiencia trabajando con Java, HTML y Bangle.js, he adquirido una sólida comprensión de la programación orientada a objetos a través de Java. Mi capacidad para desarrollar aplicaciones y soluciones utilizando este lenguaje me ha permitido solucionar las dificultades que ha habido.

5. CONCLUSIONES.

Trabajar con Bangle.js, HTML y JavaScript ha sido una experiencia enriquecedora que ha ampliado mis habilidades en el desarrollo de aplicaciones interactivas y dispositivos portátiles. La combinación de estas tecnologías me ha permitido explorar nuevos horizontes en la creación de soluciones innovadoras. Esta experiencia ha fortalecido mi base de conocimientos y mi capacidad para adaptarme a entornos cambiantes, brindándome una base sólida para seguir creciendo en un campo apasionante y en constante evolución como es el desarrollo de aplicaciones y dispositivos tecnológicos.

6. REFERENCIAS.

https://banglejs.com/

Pagina oficial de bangle.js

https://www.espruino.com/Bangle.js2

Su “libro de instrucciones”

https://www.espruino.com/Reference#_global

Manual de instrucciones mas detallado

https://www.espruino.com/Bangle.js?print

Funciones integradas en bangle.js

https://forum.espruino.com/conversations/356084/

Bangle.js en visual studio fallido

https://www.espruino.com/Bangle.js+Data+Streaming

Data streaming

7. ANEXO.

Codigo final HTML

```
<html>
 <head>
  <title>Bangle.js Accelerometer streaming</title>
 </head>
 <body>
<script src="https://www.puck-js.com/puck.js"></script>
<button id="btnConnect">Connect</button>

<script>

// codigo a subir al reloj
var BANGLE_CODE = `
// Inicializar el registro de datos
var Pulsos = [];
```

```
var Tiempo = [];

function onHRM(h) {
  let d = new Date();
  var timeStr = d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds();
  Pulsos.push(h.bpm);
  Tiempo.push(timeStr);
  g.clear();
  g.setFont("6x8", 2);
  g.drawString("Pulsos: " + h.bpm, 10, 40);
  g.drawString("Tiempo: ",10, 80);
  g.drawString(timeStr, 10, 120);
  Bluetooth.println(h.bpm + ' ; ' +timeStr);
  g.flip();
}

function enviarDatos(bpm, timeStr) {
  var dataStr = JSON.stringify({ tiempo: timeStr, pulsos: bpm });
  Bluetooth.println(dataStr);
}

Bangle.on('HRM', onHRM);
Bangle.loadWidgets();
Bangle.setUI("clock");
Bangle.setHRMPower(1);
```

```
// lo que hace cuando se haga click en el conect
var connection
document.getElementById("btnConnect").addEventListener("click", function() {
  // disconnect if connected already
 if (connection) {
   connection.close();
   connection = undefined;
 }
 // cuando se conecta
 Puck.connect(function(c) {
   if (!c) {
     alert("Couldn't connect!");
     return;
   }
   connection = c;
   // manejar los datos que recibimos se llama onLine
   // Cada vez que cogemos line (datos)
   var buf = "";
   connection.on("data", function(d) {
     buf += d;
     var l = buf.split("\n");
     buf = l.pop();
     l.forEach(onLine);
```

```
    // Reiniciar bangle js primero
    connection.write("reset();\n", function() {
      // Wait for it to reset itself
      setTimeout(function() {
        // Now upload our code to it
        connection.write("\x03\x10if(1){"+BANGLE_CODE+"}\n",
          function() { console.log("Ready..."); });
      }, 1500);
    });
  });

// revisar si hemos recibido los datos indicados
// si son del corazon lo updatemaos
function onLine(line) {
    console.log("RECEIVED: " + line);
    // mandar line a servidor
    fetch('/guardarDatos', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ line })
    })
```

```
    .then(response => {
        if (response.ok) {
            console.log('Datos enviados al servidor');
        } else {
            console.error('Error al enviar datos al servidor');
        }
    })
    .catch(error => {
        console.error('Error de red:', error);
    });


</script>
<style>


</style>
 </body>
</html>
```

Server.js

```
//guardar tiene que estar tambien
const express = require('express');
const app = express();
const port = 3000;
```

```
const fs = require('fs');

// Servir archivos estáticos desde una carpeta (por ejemplo, 'public')
app.use(express.static('public'));

// Configura una ruta POST para recibir los datos
app.use(express.json()); // Para parsear application/json

app.post('/guardarDatos', (req, res) => {
    const data = req.body.line; // obtener la variable 'line' enviada desde la página web
    // Guarda los datos en un archivo
    fs.appendFile('datos1.json', data + '\n', (err) => {
        if (err) {
            console.error('Error al guardar los datos en el archivo');
            res.status(500).send('Error al guardar los datos');
        } else {
            console.log('Datos guardados correctamente en el archivo');
            res.send('Datos guardados correctamente');
        }
    });
});
app.listen(port, () => {
    console.log(`Servidor corriendo en http://localhost:${port}`);
});
```

Guardar.js

```
fetch('/guardar', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(datos)
})
.then(response => {
  if (response.ok) {
    console.log('Datos guardados en el servidor');
  } else {
    console.error('Error al guardar datos en el servidor');
  }
})
.catch(error => {
  console.error('Error de red:', error);
});
```