

INDOOR CYCLING SMART

Alben Adrian Ramirez Gomez, Marlon Javier Cabrera Montenegro,

Henry David Bran Velasquez, Diego Fernando Cortez Lopez,

Katerine Adalinda Santos Ramirez, Julio José Orellana Ruiz

Universidad de San Carlos de Guatemala, Facultad de Ingeniería

I. INTRODUCCIÓN

Actualmente debido a la pandemia por el COVID-19 y las estrategias implementadas para controlarla, se ha creado un ambiente poco propicio para mantener las actividades de la vida diaria. En relación con la actividad física no se han entregado medidas ni recomendaciones por parte de los estamentos reguladores. Por lo que sabiendo la necesidad que existe en realizar ejercicios físicos diarios, se decidió contratar personal especializado que instale Indoor cycling que no es más que ciclismo de salon que incluye pedaleo constante para elevar la frecuencia cardiaca y trabajar diferentes grupos musculares que asemejan push ups e intervalos. Los beneficios de la actividad física y del ejercicio en el fortalecimiento del sistema inmune son documentados. El propósito de la revisión es analizar la evidencia de los datos del usuario sobre el ejercicio y actividad física en tiempos de pandemia.

I. OBJETIVOS

A. Generales

- Diseñar un dispositivo que solucione una necesidad de actividad fisica y mejorar la salud en época de pandemia.

B. Específicos

- Diseñar un dispositivo que sea una alternativa que solucione la necesidad de realizar ejercicio en casa.
- Implementar una aplicación en Processing que permite visualizar magnitudes físicas digitalizadas para una comprensión de datos humanamente legible.
- Aprender a desarrollar una solución mediante la correcta implementación del framework de iot.

I. DESARROLLO DE LA PRÁCTICA

A. Bill of Material

a. Listado de Materiales físicos:

- Arduino Mega 2560
- Sensor de pulso cardíaco
- Sensor de efecto hall KY-003
- Sensor de temperatura LM35DZ
- Protoboard
- Cable de Protoboard

- Led
- 3 Resistencias 1kohm
- Cartón

b. Listado de Materiales digitales:

- API de ingreso de parámetros
- Componentes digitales de diseño
- Processing para las gráficas y diseño para ver resultados finales.

B. Magnitudes físicas a medir

- Distancia recorrida (cm)

$$distancia = \#vueltas * 2 * 3.1416 * 20$$

Ecuación 1: Distancia recorrida por el número de vueltas dadas en la rueda.

- Velocidad promedio realizada por la bicicleta

$$Velocidad = \frac{distancia}{(tiempo/1000)}$$

Ecuación 2: Velocidad promedio calculada con el tiempo transcurrido en el sistema.

- Temperatura en grados centígrados ($^{\circ}C$)

$$Temperatura = \frac{5 * lectura * 100}{1024}$$

Ecuación 3: Temperatura que sale del sensor LM35DZ

- Calorías quemadas

$$\text{Calorías} = 0.049 * \text{Peso} * 2.2 * \text{tiempo}$$

Ecuación 4: Calorías gastadas durante vrs Tiempo de la práctica

- Pulso cardiaco y oxígeno (bpm)

C. Funciones Principales:

- **Medición de la velocidad promedio:**

El dispositivo será capaz de medir e interpretar la velocidad que recorre en la bicicleta, en cm/s.

- **Distancia recorrida por el tiempo:**

La estación deberá ser capaz de medir la distancia que se recorre durante el tiempo transcurrido cuando se inició la práctica.

- **Temperatura corporal en centígrados:**

La unidad debe poder medir la temperatura de la persona que realiza la práctica.

- **Pulso cardiaco y oxígeno:**

El dispositivo podrá guardar los pulsos cardíacos de la persona durante el tiempo en que se realizó la práctica.

- **Cantidad de calorías quemadas:**

El dispositivo podrá calcular la cantidad de calorías que se quemaron durante la práctica.

D. Procedimiento

1. Sensores

- Arduino Mega 2560

Tamaño	Lectura Sensor	Instalación	Microcontrolador	Medición voltaje
7 x 7 x 6 mm	Pines de E/S digitales: 54 (de los cuales 14 proporcionan salidas PWM) Pines de entrada analógica: 16	Maqueta Objetivo: áreas exteriores, interiores.	ATMEGA2560	7 a 12V

- link datasheet:

https://content.arduino.cc/assets/Pinout-Mega2560rev3_latest.pdf



Precio: Q229.00

- **Sensor de pulso cardíaco**

Tamaño	Lectura Sensor	Instalación	Rango de Temperatura de funcionamiento	Unidad de Medida
15.8x 3.6 mm	Óptico infrarrojo	Maqueta Objetivo: intemperie que permite percibir el ritmo cardíaco.	temperatura mínima -40°C, máxima +85°C	Pulsaciones por minuto (bpm)

- link datasheet:

<https://www.digikey.pl/htmldatasheets/production/3024658/0/0/1/pulse-sensor-datasheet.html>



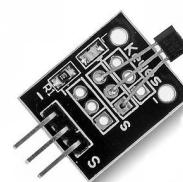
Precio: Q50.00

- **Sensor de efecto Hall KY-003**

Tamaño	Lectura Sensor	Instalación	Voltaje de Operación	Unidad de Medida
19 x 15 mm	campo magnético	Objetivo: Contar pulsos en piezas giratorias.	4.5V a 24V DC	Periodo de rotación

- link datasheet:

<https://datasheetspdf.com/pdf/1321961/Joy-IT/KY003/1>



Precio: Q15.00

- Sensor de temperatura LM35DZ

Tamaño	Medición voltaje	Lectura Sensor	Rango de Medición	Unidad de Medida
28 x 13 x 6 mm	4 a 30V DC	Análogo	temperatura: -55 a +150 °C	Temperatura de la persona (°C)

• link datasheet:
<https://pdf1.alldatasheet.com/datasheet-pdf/view/520582/TI1/LM35DZ.html>

• Precio: Q34.00



2. Conectividad

- La transmisión de datos se realiza por medio del arduino que envía los datos a través del puerto serial a la pc, esto ocurre de forma cableada.
- Los datos recibidos en la PC son almacenados en una base de datos persistente y este a su vez son procesados en la aplicación processing a través de una imagen.
- Datos que deben de enviar desde la unidad al PC:
 - Medición de la velocidad
 - Tiempo transcurrido en segundos
 - Temperatura en grados centígrados
 - Distancia recorrida en cm
 - Pulsaciones por minuto

★ Processing

Con el objetivo de poder visualizar y evaluar la medición de las magnitudes en tiempo real, se le solicita al desarrollador elaborar una aplicación en processing, el objetivo es representar de forma gráfica todos los datos que sean recolectados

En la Figura 3 se puede observar un vista previa de cómo se vería el funcionamiento del prototipo y los datos que serán visibles para el usuario, por lo que la conectividad del puerto serial se puede observar de la siguiente manera:

```
void setup(){
  //Tamaño de la ventana
  size(1000, 666);

  w = width + 16;
  for (int i = 0; i < maxwaves; i++) {
    json =
    loadJSONObject("http://localhost:4001/getDatos")
    ;
    BPMValue = json.getInt("frecuencia");
    amplitude[i] = (BPMValue*0.5)-30;;
    println(amplitude[i]);
    float period = random(100,300); // How many
pixels before the wave repeats
    println("PERIODO "+period);
    dx[i] = (TWO_PI / period) * xspacing;
  }

  yvalues = new float[w/xspacing];

  //Cargando imagenes a las variables
  temperature = loadImage("termometro.png");
  calories = loadImage("llama.png");
}

Para dibujar cada partícula y la actualización o
sincronización de la misma, se consulta los datos
de la base de datos cada 5 segundos.

void draw(){
  background(190, 209, 216); //Color de fondo
  //Grisaceo celestoso xd
  getData();
  frames();

  labelText();
  drawTemperature();
  drawCalories();
  addImages();

  calcWave();
  renderWave();
}
```

Se realiza una consulta a la base de datos para obtener los valores y realizar el cálculo de las calorías perdidas utilizando la ecuación 4.

```
void getData(){
  int currentSecond = second();
  boolean consultar =false;
```

```

if(currentSecond != lastSecond){
    tiempoConsulta = tiempoConsulta + 1;
    if(tiempoConsulta == 5){
        tiempoConsulta = 0;
        consultar = true;
    }
}

lastSecond = currentSecond;
if(consultar){
    //Consumo de Api
    json =
loadJSONObject("http://localhost:4001/getDatos")
;

tempValue = json.getFloat("temperatura");
//Obtiene los valores de temperatura corporal

amplitude[0] = amplitude[1];
amplitude[1] = amplitude[2];
amplitude[2] = amplitude[3];
BPMValue = json.getFloat("frecuencia");
//Obtiene los valores de Frecuencia cardiaca
amplitude[3] = (BPMValue*0.5)-30;

weightValue = json.getFloat("peso"); //Obtiene
valor de peso en kg de la persona
timeValue = json.getFloat("tiempo"); //Obtiene
la cantidad de segundos que lleva ejercitandose

//Formula para calcular calorias quemadas
0.049*(peso *2.2)*total de minutos de practica
calValue =
0.049*(weightValue*2.2)*(timeValue/60);
}
}

```

Se dibujan rectángulos que contendrán las imágenes y los datos, para separarlos de manera ordenada como se muestra en la figura 7.

```

void frames(){
fill(255,255,255); //color de los retangulos Blanco

//Retangulos para separar la informacion
rect(150,50,300,200,25);
rect(545,50,300,200,25);
rect(150,300,700,350,25);
fill(216, 227, 226);
rect(150,400,700,200);
}

```

Se agregan las imágenes del termómetro, representando la temperatura corporal, y de la flama, representando las calorías quemadas .

```

void addImages(){
image(temperature,300,100); //Imagen de
temperatura
image(calories,715,137); //Imagen de
temperatura
}

```

Se define el tamaño, color y la posiciones en pantalla de los label que se visualizan.

```

void labelText(){
//Temperatura
textSize(18); //Tamaño del texto
fill(255,255,255); //Color del texto Blanco
text(labelTemperature, 165, 40); //Muestra texto
en pantalla

textSize(50); //Tamaño de texto
fill(0,0,0); //Color del texto negro
text(tempValue+"°C", 170, 100); //Imprimir texto
en pantalla

//Calorias
textSize(18); //Tamaño del texto
fill(255,255,255); //Color del texto Blanco
text(labelCalories, 560, 40); //Muestra texto en
pantalla

textSize(50); //Tamaño de texto
fill(0,0,0); //Color del texto negro
text(calValue+"Cal", 560, 100); //Muestra texto
en pantalla

//Frecuencia cardiaca
textSize(18); //Tamaño del texto
fill(255,255,255); //Color del texto Blanco
text(labelBPM, 165, 290); //Muestra texto en
pantalla
}

```

Se le agrega animación al apartado de la temperatura.

```

void drawTemperature(){
    //Color Temperatura
    //Conversion para la temperatura donde maximo
    sea 36°C y minimo 15°C
        //Valor,Min,Max,Rango para devolver el
    valor
    float temp = map (tempValue, 15,36, 0, 255);
    red = temp;
    //conversion para el cambio de color.
    green = temp * -1 + 255;
    blue = temp * -1 + 255;

    //Dibujamos un rectangulo y una esfera para
    colorear la temperatura
    noStroke(); //Les quita el borde a las figuras
    fill(red,green,blue);
    rect(352.5,110.5,85,500);
    ellipseMode(CENTER);
    ellipse(355,201,26,26);
}

```

Se le agrega animación al apartado de las calorías quemadas.

```

void drawCalories(){
    //Color Temperatura
    //Conversion para la temperatura donde maximo
    sea 36°C y minimo 15°C
        //Valor,Min,Max,Rango para devolver el
    valor
    float temp = map (tempValue, 15,36, 0, 255);
    red = temp;
    //conversion para el cambio de color.
    green = temp * -1 + 255;
    blue = temp * -1 + 255;

    //Dibujamos un rectangulo y una esfera para
    colorear la temperatura
    noStroke(); //Les quita el borde a las figuras
    ellipseMode(CENTER);
    fill(247, 172, 11);
    arc(750,170,130,130, 0, PI*(calValue*0.0009));

    fill(255);
    ellipse(750,170,80,80);
}

```

Se crea las ondas que representan el ritmo cardíaco de la persona durante el ejercicio.

```

void calcWave() {
    theta += 0.1;
}

```

```

for (int i = 0; i < yvalues.length; i++) {
    yvalues[i] = 0;
}

for (int j = 0; j < maxwaves; j++) {
    float x = theta;
    for (int i = 0; i < yvalues.length; i++) {
        if (j % 2 == 0) yvalues[i] += sin(x)*amplitude[j];
        else yvalues[i] += cos(x)*amplitude[j];
        x+=dx[j];
    }
}
}

void renderWave() {
    noStroke();
    fill(0);
    ellipseMode(CENTER);
    for (int x = 0; x < yvalues.length; x++) {

        ellipse((x*xspacing)*0.69+150,(height/2+yvalues[
        x])+180,6,6);
    }
}

```

★ Arduino

Setup(): se inicializa el serial, se definen los pines, se inicializa el sensor, se configura el sensor.

```

void setup(){
    pinMode(13, OUTPUT);
    Serial.begin(9600);    detección de datos
    interruptSetup();
    Serial.println("Presiona S para iniciar y F para
finalizar");
}

```

Loop(): Primero leemos el dato que manda cuando se el imán en el sensor, Se lee el valor del pulsómetro, si se activa el sensor se comenzará a grabar los datos y si no está activo no mandara datos de momento. Al grabar los datos empieza a grabar la temperatura con la fórmula:

```

tempC = analogRead(pinLM35);
tempC = (5.0 * tempC * 100.0)/1024.0;

```

Luego de esto validamos si el sensor de vuelta está completando una vuelta. si la vuelta es dada se

calcula la distancia de la vuelta. se muestra de la manera:

```
if (estadohall == LOW){
    contadorHall++;
    distancia = contadorHall * 2 * 3.1416*20;
}
```

luego se calcula el tiempo transcurrido luego de seguir con el sistema, Al tener la distancia realizaos el cálculo de la velocidad. y enviamos los datos de tal manera:

```
if (tiempo2 > (tiempo1+1000)){
    tiempo1 = millis();
    tiemposegundos = tiempo1/1000;
    velocidad = distancia/tiemposegundos;
    String datos="";
    datos.concat(tempC);
    datos.concat(",");
    datos.concat(distancia);
    datos.concat(",");
    datos.concat(tiemposegundos);
    datos.concat(",");
    datos.concat(velocidad);
    datos.concat(",");
    datos.concat(BPM);
    Serial.println(datos);
}
```

Luego de cada ejecución se espera un tiempo necesario al sistema, en este tiempo la bandera QS el arduino busca el pulso del corazón, esto estabiliza este sensor.

```
delay(200);
if (QS == true){
    QS = false;
}
```

★ Entorno del Objeto

Este puede permanecer en un área exterior y seguir funcionando correctamente(suponiendo que este sellado el prototipo). Por ejemplo:

- Patio
- Interiores de casa

El mejor caso para probar el prototipo es en un espacio semi cerrado donde no haya mucha interferencias del ambiente entre otros factores.

★ Tamaño del objeto

La base del prototipo mide un aproximado de 15.2 x 32 cm. Y la altura cuenta con 28 cm.

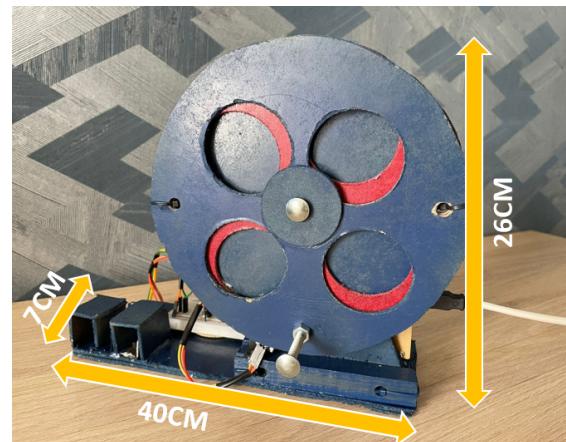


Figura 1: Dimensiones del prototipo

3. Bocetos del Prototipo

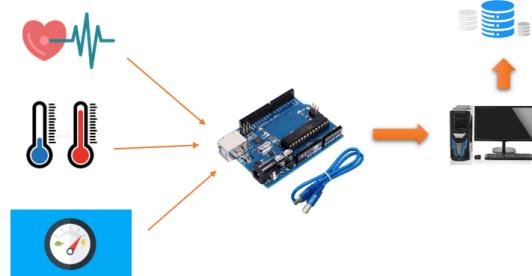


Figura 2: Comunicación entre los componentes al arduino y del arduino a el manejo de datos y software.

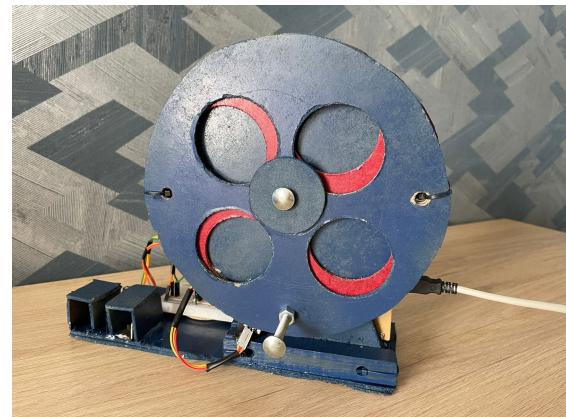


Figura 3: Prototipo

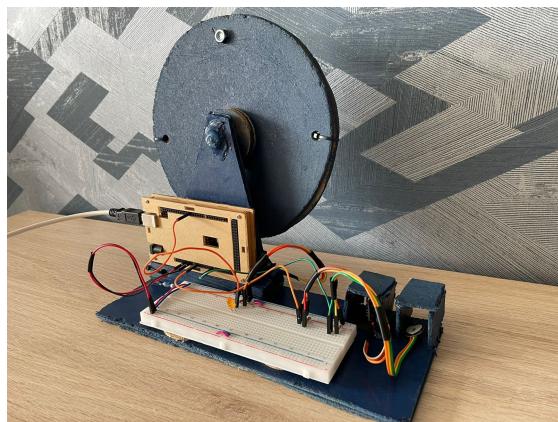


Figura 4: Prototipo visualizado de parte posterior



Figura 5: Prototipo visualizado de parte Superior

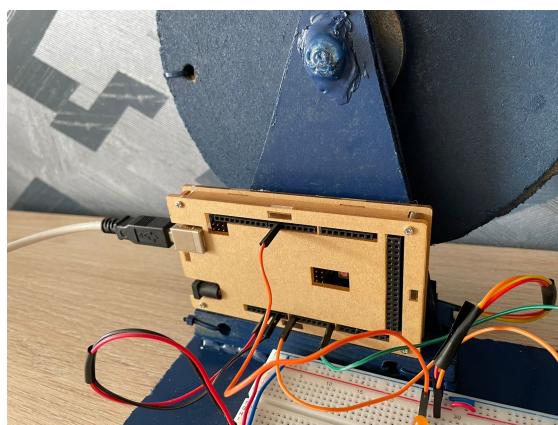


Figura 6:Protector del Arduino UNO

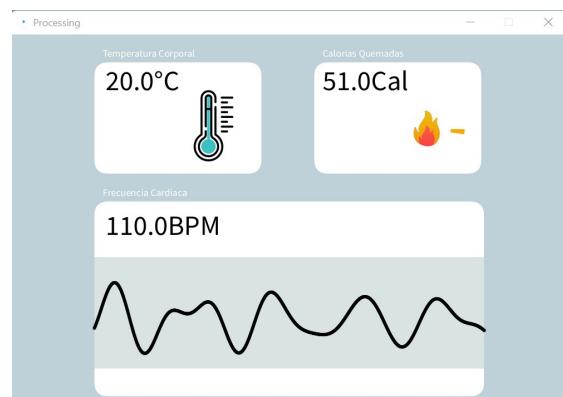


Figura 7: Gráficas de Resultados con el programa Processing

4. Base de Datos

En la base de datos se utilizo una base de datos no estructurada en esta se almacenarán objetos

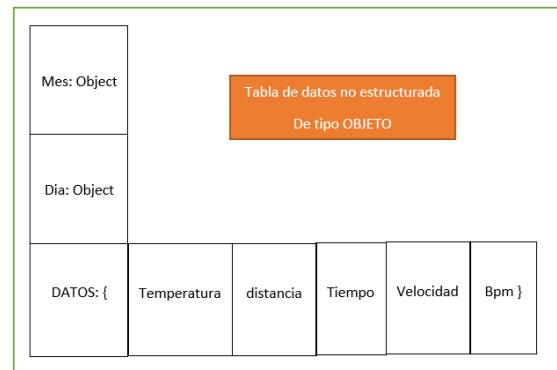


figura 7: Esquema de los datos

Los datos se representan de forma de objeto y esta función es:

`indoor { mes { dia {datos}}}`

los 3 index principales se muestra de manera:



Figura 8: datos principales

Los datos se componen de un objeto mayor, pero de igual forma siempre sigue perteneciendo a un objeto donde este objeto almacena a objetos hijo. y se representa:

DATOS: {	Temperatura	distancia	Tiempo	Velocidad	Bpm }
----------	-------------	-----------	--------	-----------	-------

figura 9: Objeto Datos

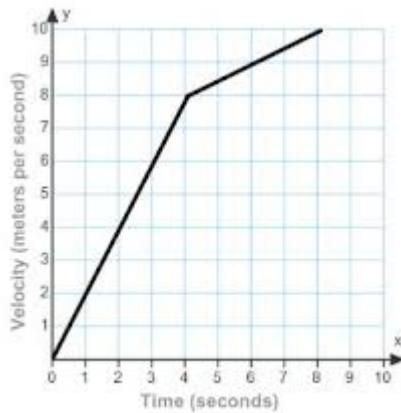


Figura 9: Figura velocidad sobre tiempo

a. Análisis Descriptivo:

- Aplicar IOT para la resolución de problemas en la vida cotidiana a través del uso de sensores y mecanismos que permitan monitorear los datos obtenidos, en este caso un mecanismo para practicar el ejercicio.

b. Análisis de Diagnóstico:

- IOT, Aplicar el IOT para la resolución de un mecanismo de ejercicio.
- Conexión de sensores al arduino uno. para extraer los datos obtenidos que se ejecuten en el sensor.
- Por medio de programación entre software's, se puede obtener, relacionar y almacenar los datos que se obtengan.
- Los datos obtenidos se manejaron desde un servidor donde aloja los resultados claramente esclarecidos y según los resultados se ejecutará el procesing para los distintos dinamismos

5. Conocimientos aplicados

a. Recuperar datos de la base de datos

```
const recuperar =
database.ref('indoor/registros/'+mes+'/'+req.query.f
echa);
```

b. Ingresar datos a la constante recuperación en API

```
recuperar.on("value",function(snapshot){
snapshot.forEach(function(childSnapshot){
var key = childSnapshot.key;
var childData = childSnapshot.val();

Datos_recuperados.indoor.registros.push(childData
);
});

res.json(Datos_recuperados);
});
```

c. Extraccion y llenado de datos para datos de logica desde la API

```
var key = childSnapshot.key;
var childData = childSnapshot.val();
```

```
Datos_recuperados.indoor.registros.push(childData
)
```

d. Conección para la base de dato FIREBASE:

```
const { initializeApp, applicationDefault } =
require('firebase-admin/app');
const { getDatabase, ref, set } =
require("firebase-admin/database");
const admin = require("firebase-admin/database")

var config = {
  credential: applicationDefault(),
  databaseURL:
'https://practical1-arq2-default-rtdb.firebaseio.com'
};
```

6. Link del repositorio de github

- https://github.com/adrianrg578/ACE2_2S22_G2.git

7. Link del video del prototipo

- https://youtube.com/playlist?list=PL6GFUI5F47n82oaGU8zTI_NskuJw2Fpl-

III. REFERENCIAS

- <https://quemarcalorias.es/deportes/ciclistas-que-queman-16-km-h>
- <https://firebase.google.com/docs/database/web/read-and-write?hl=es-419>