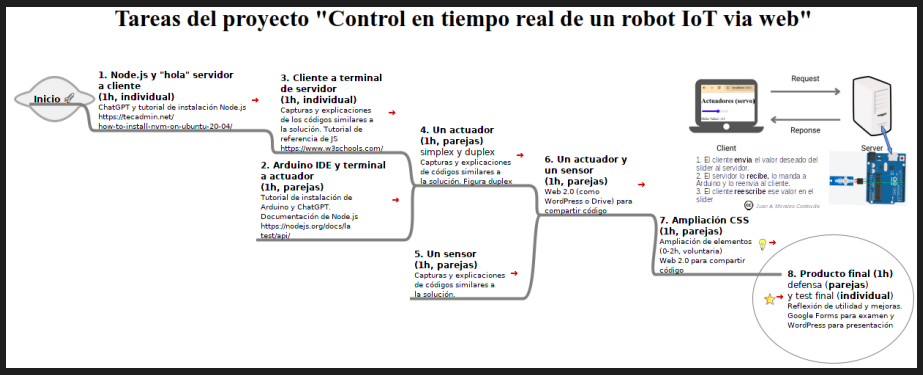
JUAN ANDRÉS MORALES CORDOVILLA

Proyecto: Control en tiempo real de un robot IoT via web (para 2ºBach. TIC y 2ºSMR AW)

diciembre 13th, 2023



Pincha aquí para navegar este mapa interactivamente.

1. 1. Instalación de Node.js y hola mundo (1h, individual)
    1. Hola mundo web
2. 2. Instalación de Arduino IDE y control por terminal de actuador (1h, en parejas)
    1. Control de actuador por puerto serie y terminal
3. 3. Envio de datos del cliente a terminal del servidor en RT(1h, individual)
4. 4. Control en RT de un actuador (1h, en parejas)
    1. Control duplex en RT de un actuador
5. 5. Control de un sensor en RT (1h, en parejas)
6. 6. Control de un sensor y un actuador en RT (1h, en parejas)
7. 7. Amplia el sistema (1 a 3h, en parejas)
    1. Amplia mejorando estilo del CSS de la web (obligatorio)
    2. Amplia añadiendo más sensores y actuadores (no para 2ºSMR)
    3. Amplia el sistema con IA (voluntario)
8. 8. Defensa y evaluación del producto final (1h)
    1. Defensa
    2. Test autocorregible
    3. Diana de evaluación

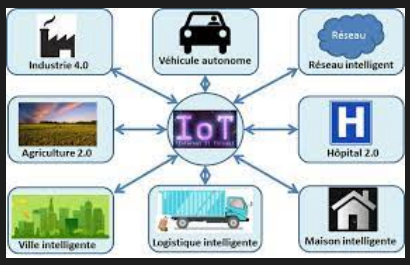
Qúe vamos a ver:

- Este proyecto es una mejora del proyecto web Invernadero IoT: https://juanandresmoralescordovilla.wordpress.com/2023/07/03/proyecto-invernadero/
- Node.js: instalación y uso (aunque se recomienda ciertos conocimientos previos sobre socket.io)
- JavaScript: uso de objetos (como slider y sockets) desde clases
- Arduino: C/C++ programación
- Desarrollo: colaboración y subproblemas en IoT (Internet of Things) en Real Time (RT)

NOTA: pon capturas de todas las tareas

1. **Instalación de Node.js y hola mundo (1h, individual)**

Internet de las Cosas (IoT) en Tiempor Real (RT): IoT es control remoto de dispositivos, pero como dice wikipedia, IoT es un termino erróneo ya que puede hacerse sobre la red de Internet pero también sobre cualquier otra red entre un cliente (operador) y un servidor (robot). RT quiere decir que los cambios efectuados por el operador sobre el robot se vean reflejados casi instantaneamente (o bajo un tiempo controlado). Ej. de aplicaciones las hay a millares como muestra la Fig. pero mencionamos robots en entornos donde el humano no puede estar como una zona radio activa



Node.js: es un framework de JavaScript para servidor que permite crear aplicaciones de RT

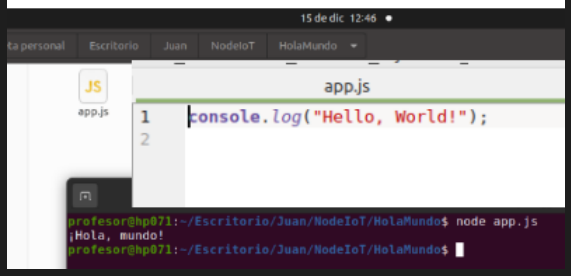
Tarea: instalarcurl y Node.js. Usa el siguiente bash script. Los # son comentarios que debes completar para mostrar que lo comprendes

```
#Instalar Node por defecto
sudo apt update
sudo apt install nodejs
#Sal de superusuario y continua como usuario
node -v #node v10
# Si te falta instala curl.
sudo apt install curl
# Instalar una NVM version como este enlace:
https://tecadmin.net/how-to-install-nvm-on-
ubuntu-20-04/
#Instala los "esenciales"
sudo apt-get install build-essential
#Checkea los esenciales
checkinstall libssl-dev
# Instala NVM
curl -o-
https://raw.githubusercontent.com/creationix/nvm
/v0.32.1/install.sh | bash
# Reinicia terminal
# Version
nvm --version
#Ultimo node 20
nvm install 20
#Ultimo npm 10 (con ultimo nvm)
sudo apt install npm
#Ultimo serialport y express
npm install serialport express
```

Ej. de captura

```
Configurando node-gyp (6.1.0-3) ...
Configurando node-libnpx (10.2.1-2) ...
Configurando npm (6.14.4+ds-1ubuntu2) ...
Procesando disparadores para man-db (2.9.1-1) ...
profesor@hp071:~$ npm install serialport

added 21 packages in 8s

14 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New patch version of npm available! 10.2.3 -> 10.2.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.5
npm notice Run npm install -g npm@10.2.5 to update!
npm notice
```

Tarea: haz un «hola mundo» en Node.js para ver que todo está OK usando ChatGPT. Sube captura similar a esta:

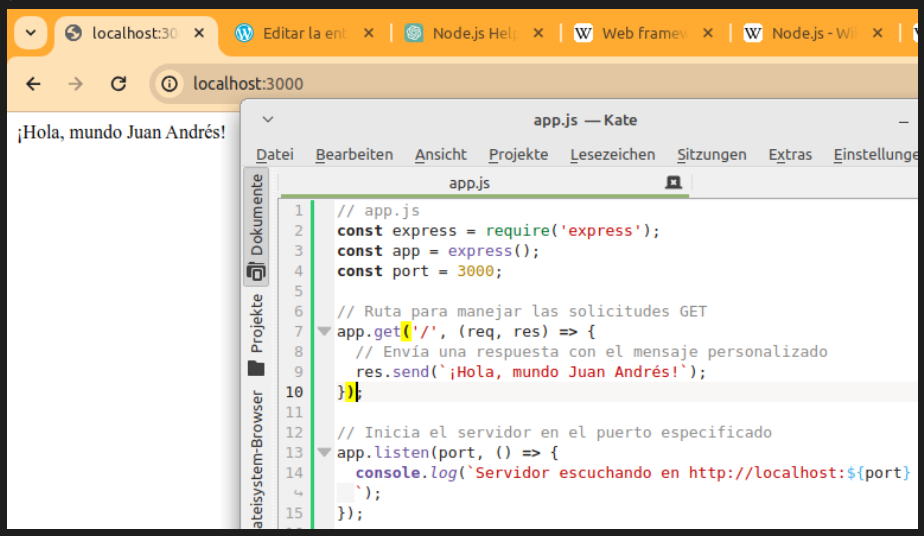


Observa que:

- console.log(«¡Hola, mundo!»);: imprime por consola texto
- node app.js: lanza app.js en modo servidor (se recomienda no lanzarlo como superusuario)

Hola mundo web

Tarea-guiada (corrijo por observación): haz un «hola mundo» node.js que salga en el navegador. Suba una captura con de código comentado, ejecución y resultado



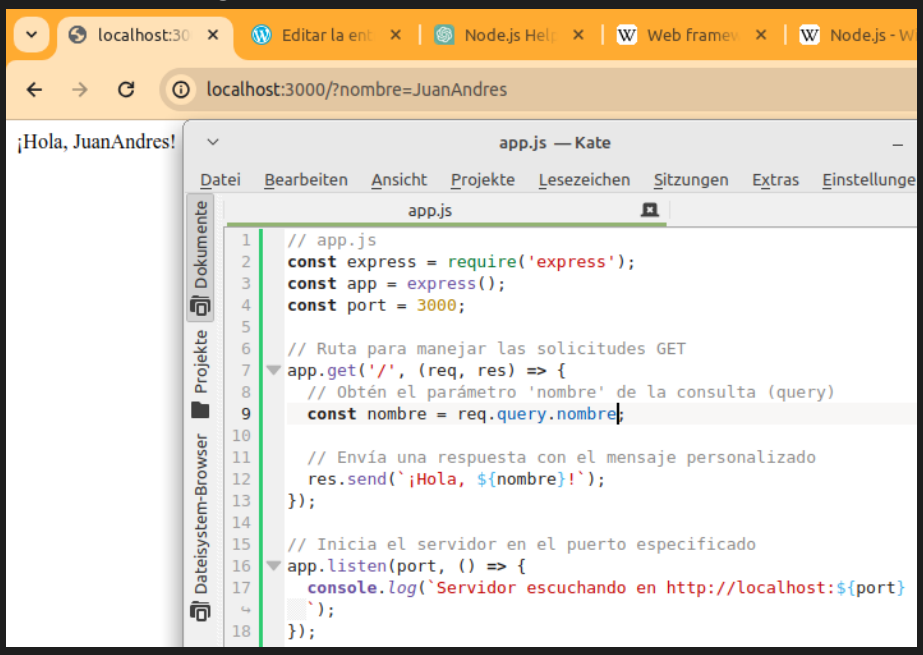
Observe que:

- Express.js: es el «estándar de facto» frameworks o libreria de los servidores Node.js. const express=require('express'); importa dicha libreria en objeto express
- const app = express(); crea el objeto que va a representar toda nuestra aplicación Node.js. App tiene muchos métodos para configurar app, definir rutas, manejar solicitudes y respuestas...
- app.get(): función asociada a petiones GET (por URL aunque aquí no se usa)
- app.listen(port): inicia tu PC como servidor escuchando en puerto 3000
- Para lanzarlo escribimos en el navegador la URL: IP+:puerto del servidor Ej. si el servidor eres tu mismo se pone localhost o 127.0.0.1 http://localhost:3000/

Tarea: conectate al servidor de tu compañero de clase y pon captura

- NOTA: es importante que el cliente y servidor estén dentro de la misma red (Ej: Andared_Coorporativo donde las IP puede ser 192.168.53.87)

Tarea-ampliación: cree un «hola mundo» que reciba por GET un nombre como mustra la Fig.:



Observe que:

- El envio del nombre (Juan) se hace por la URL del cliente: http://localhost:3000/?nombre=JuanAndres
- req.query.nombre: saca el nombre (Juan) del envio recibido en el servidor
- res.send(`¡Hola, ${nombre}!`);: junta la cadena 'Hola' con la variable nombre

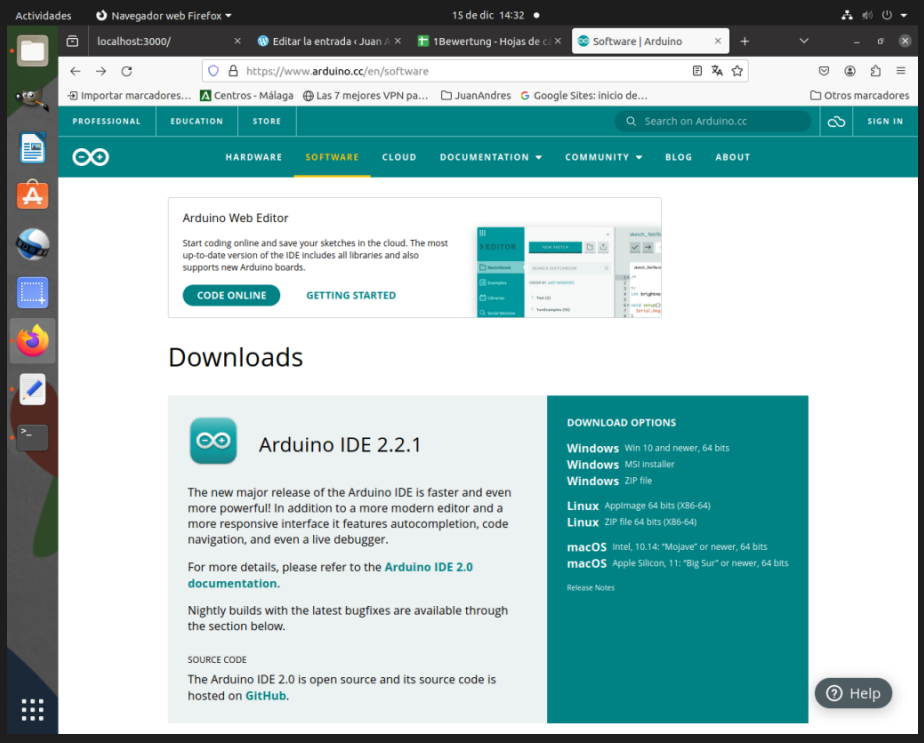
Tarea-ampliación-voluntaria: haz un «hola mundo» que reciba por POST el nombre.

2. Instalación de Arduino IDE y control por terminal de actuador (1h, en parejas)

Arduino IDE: es un Entorno de Desarrollo Integrado (IDE) para crear programas en C/C++, compilarlos y pasarlos a Arduino por puerto USB serie

Instalación:

1. Ir a la web Arduino IDE



2. Descarga Linux App Image 64 bit (a la derecha). Llevarlo a una carpeta segura (ej. /home/usuario/Escritorio/Juan), darle permiso de ejecución (chmod u+x arduino...) y lanzarlo

Tarea-voluntaria: enciende y apaga pin13, 2 veces rápido y 2 lento y flashealo sobre tu Arduino para ver que funciona.

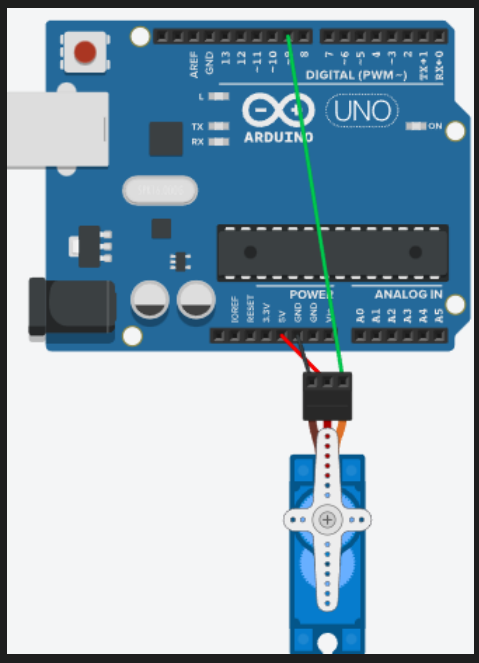
Control de actuador por puerto serie y terminal

NOTA: trabajar en parejas, uno monta código otro hardware

Robot: es un hardware o sistema físico que recibe unos input (sensores) y produce unos outputs (actuadores) según su software o algoritmo.

Tarea-guiada: mueve el actuador servomotor enviando los grados por la terminal hacia el puerto serie

1. Conecta la tierra (negro), la alimentación (rojo) y la señal (verde) del servo al pin 9 como abajo:



2. Pásale un código para que el ángulo del servo se ponga al valor recibido por puerto serie:

Código soloServo.ino:

```
#include <Servo.h>
const int servPin=9;
int inputPosition;
Servo myServo; // Create servo object

void setup()
{
  myServo.attach(servPin); // Servo signal line to Ar
  myServo.write(90); // Calibrate Servo
  Serial.begin(9600);
}

void loop()
{
  if(Serial.available()) {
    inputPosition = Serial.parseInt();
    if(inputPosition >= 10 && inputPosition < 170) {
        myServo.write(inputPosition);
    }
  }
}
```

Observaciones:

- Serial.begin(9600): establece que la comunicación por el puerto será a 9600 baudios o bps (bits per seconds)
- inputPosition=Serial.parseInt(): lee texto del puerto serie hasta un salto de linea '\n' y lo convierte en entero que se almacena en inputPosition.
- Los valores min. y max. del ángulo del servo están entre 10 y 170 grados

3. Enviale un ángulo desde la terminal y comprobar que se mueve. Quizás sea necesario tener abierto Arduino IDE, enviar en modo root o añadirle salto de linea al número (echo «40\n» > /dev/ttyACM0)

```
juan@jamc:~/Schreibtisch/CursoB2Digital/PractGuiadaExeLearning$ sudo su
root@jamc:/home/juan/Schreibtisch/CursoB2Digital/PractGuiadaExeLearning# echo "4
0" > /dev/ttyACM0
root@jamc:/home/juan/Schreibtisch/CursoB2Digital/PractGuiadaExeLearning#
```

- /dev/ttyACM0: es el archivo dispositivo (orientado a caracteres) asociado al USB. Puede variar y en Windows suele ser COM1.

Tarea: pon capturas de los códigos anteriores comentados en español

3. **Envio de datos del cliente a terminal del servidor en RT(1h, individual)**

NOTA: La ejecución es individual la pero la copia de los códigos puede hacerse en parejas, uno index.html y sketch.js, otro app.js)

Tarea-guiada: haz una web cliente con slider (deslizador) que envie su valor en Tiempo Real (RT) y que el servidor saque este valor por su consola (terminal)

1. Crea una carpeta llamada p2SliderAConsola y dentro de esta los archivos y directorios siguientes:

```
p2SliderAConsola
├── app.js
└── publicJuan
    ├── index.html
    └── sketch.js
```

2. Copia los siguientes códigos y coméntalos en español:

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Serial Data Viewer</title>
</head>

<body>
  <h1> Actuadores (servo) </h1>
  <input type="range" id="slider" min="0" max="180" value="90">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.1.3/socket.io.js"></script>
   <script src="sketch.js"> </script>
</body>
</html>
```

- <input type=»range»..> crea el slider
- <script src=»sketch.js»> enlaza el JavaScript

sketch.js:

```
//Object socket from class io
const socket = io();
//object for the slider
const slider = document.getElementById('slider');

/*Slider servomotor*/
//sent slider value to server
slider.addEventListener('input', () => {
    const value = slider.value;
    socket.emit('sliderValueJuan', value);  //sent value with Id sliderValueJuan
});
```

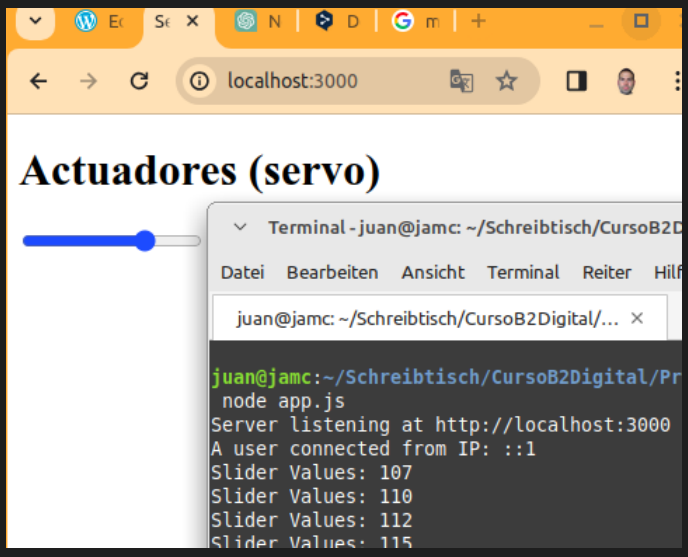
- slider: representa el objeto deslizador y slider.value su valor en cada instante
- slider.addEventListener('input', () => {..}): a cualquier elemento HTML le podemos asociar un escuchador de eventos con el formato element.addEventListener(event, function). Event suele ser 'input', 'click', 'onclick' o 'mousedown'
- () => {...}: define la función anónima a lanzar en cada evento, en este caso enviar al servidor un mensaje con Id sliderValueJuan mediante el socket creado. Esta notación se llama 'arrow function expressión' y se podria haber reemplazado por: function() {...}. Se usa junto a map, filter, .. en programación funcional.

app.js:

```
//Carga de frameworks
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
//
const app = express();
const server = http.createServer(app);
const io = socketIo(server);
const port = 3000;

//only the files from public can be served to a client
app.use(express.static('./publicJuan'));

//WebSocket connection handling
io.on('connection', (socket) => { //server waiting client connection event

  const userIP = socket.handshake.address; //IP del cliente
  console.log(`A user connected from IP: ${userIP}`); //message when client connected

  /*Slider servomotor */
  // Function to execute when a  meassage with Id 'sliderValueJuan' is recieved
  socket.on('sliderValueJuan', (value) => { //receive client value
    console.log(`Slider Values: ${value}`); //console
   });
});

// Start the server
server.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

- express, app y port: son como antes, pero ahora el servidor no lo inicia app.listen sino server.listen (que es de tipo http) por puerto 3000.
- io.on(): es una función que recibe un objeto socket (structura con datos como IP del cliente...). Permite comunicación RT bidireccional (duplex) y lanza procesos cuando un nuevo cliente se conecta al servidor. El primer proceso es mostrar la IP del cliente por consola del servidor
- socket.on('sliderValueJuan', (value) ..): es un proceso de io.on que cada vez que recibe un mensaje con Id 'sliderValueJuan' del cliente, el servidor lo saca por su consola.

3. Resultado de ejecución:



- Cuando un nuevo cliente se conecta muestra su IP

Tarea: conéctate al servidor de tu compañero y mueve el slider. Pon captura donde se vean los resultados

4. **Control en RT de un actuador (1h, en parejas)**

Tarea-guiada: modifica el código anterior para que el cliente mueva un servo del servidor

1. Conecta el servo a Arduino y métele el codigo soloServo.ino que vimos más arriba.

2. Los códigos index.html y sketch.js son como antes, solo cambia el app.js de abajo:

app.js:

```
//Carga de frameworks
const { SerialPort } = require('serialport')
const serialport = new SerialPort({path: '/dev/ttyACM0', baudRate: 9600 })
//
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
//
const app = express();
const server = http.createServer(app);
const io = socketIo(server);
const port = 3000;

//only the files from public can be served to a client
app.use(express.static('./publicJuan'));

//WebSocket connection handling
io.on('connection', (socket) => { //server waiting client connection event

  const userIP = socket.handshake.address; //IP del cliente
  console.log(`A user connected from IP: ${userIP}`); //message when client connected

  /*Slider servomotor */
  // Function to execute when a  meassage with Id 'sliderValueJuan' is recieved
  socket.on('sliderValueJuan', (value) => { //receive client value
    console.log(`Slider Values: ${value}`); //console
    serialport.write(`${value}\n`); //write value to serial port
   });
});

// Start the server
server.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

- const { SerialPort } = require('serialport'): importa la libreria del puerto serie. Se pone entre llaves para «asignar des-estructurando» el módulo (para extraer y asignar de un solo paso)
- serialport.write(` ${value}\n`): escribe al puerto value y le añade un salto de linea ('\n')

Tarea: conectate al servidor de tu compañero y mueve su servo.

Control duplex en RT de un actuador

Tarea-guiada: El código anterior es simple y lo normal es que el servidor también reenvie al cliente el dato recibido, haya comunicación duplex. Para ello, modifica los 3 códigos como abajo:

app.js:

```
//Carga de frameworks
const { SerialPort } = require('serialport')
const serialport = new SerialPort({path: '/dev/ttyACM0', baudRate: 9600 })
//
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
//
const app = express();
const server = http.createServer(app);
const io = socketIo(server);
const port = 3000;

//only the files from public can be served to a client
app.use(express.static('./publicJuan'));

//WebSocket connection handling
io.on('connection', (socket) => { //server waiting client connection event

  const userIP = socket.handshake.address; //IP del cliente
  console.log(`A user connected from IP: ${userIP}`); //message when client connected

  /*Slider servomotor */
  // Function to execute when a  meassage with Id 'sliderValueJuan' is recieved
  socket.on('sliderValueJuan', (value) => { //receive client value
    io.emit('updateSliderJuan', value); // broadcast the updated value to all connected clients
    console.log(`Slider Values: ${value}`); //console
    serialport.write(`${value}\n`); //write value to serial port
   });
});

// Start the server
server.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

- io.emit('updateSliderJuan', value); emite por broadcast (a todos los clientes conectados) el valor actual del servo. El id del mensaje es 'updateSliderJuan'

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Serial Data Viewer</title>
</head>

<body>
  <h1> Actuadores (servo) </h1>
  <input type="range" id="slider" min="0" max="180" value="90">
  <p id="sliderValue">Slider Value: 90</p>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.1.3/socket.io.js"></script>
   <script src="sketch.js"> </script>
</body>
</html>
```

- <p id=»sliderValue»>: crea un párrafo con ese id que mostrará en RT el valor nº del slider

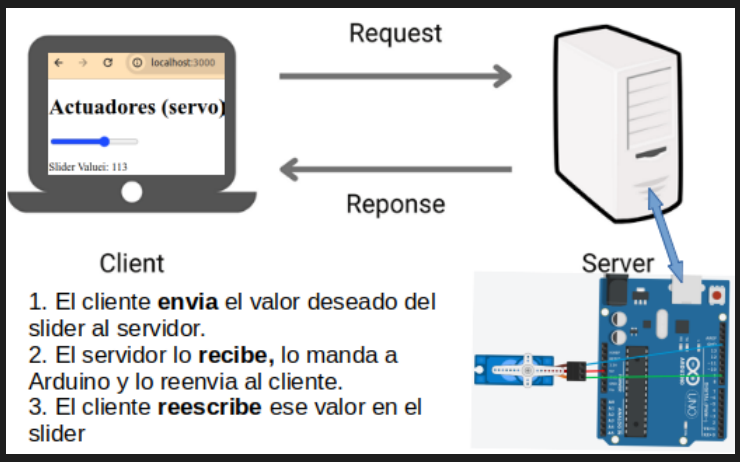
sketch.js:

```
//Object socket from class io
const socket = io();
//object for the slider
const slider = document.getElementById('slider');
const sliderValue = document.getElementById('sliderValue');

/*Slider servomotor*/
//sent slider value to server
slider.addEventListener('input', () => {
    const value = slider.value;
    socket.emit('sliderValueJuan', value);  //sent value with Id sliderValueJuan
});

//replace the slider value number with the latest
socket.on('updateSliderJuan', (value) => {  //receive server value with id updateSliderJuan
    slider.value = value;
    sliderValue.textContent = `Slider Valuei: ${value}`;
});
```

- const sliderValue: objeto que representa el slider asociado con getElementById
- socket.on('updateSliderJuan', (value)...): socket que espera a recibir el mensaje del servidor con Id 'updateSliderJuan'. Dicho mensaje incluye el

valor del servo actual

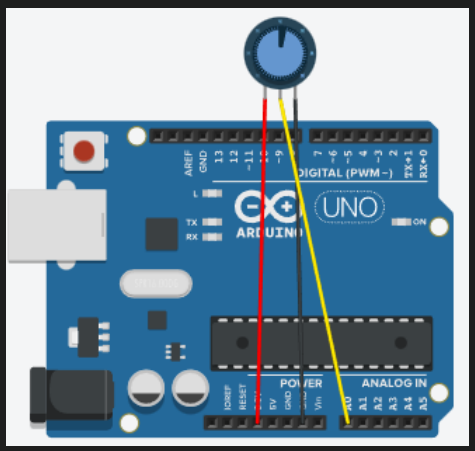
- sliderValue.textContent y slider.value: modifican el slider y el parráfo del index.html con el valor actual del servo recibido. Esto cierra el «**ciclo de comunicación duplex**» en RT como muestra la Fig.



5. **Control de un sensor en RT (1h, en parejas)**

Tarea-guiada: haz una web que muestre en RT el valor de un potenciometro

1. Conecta el potenciometro a Arduino con la tierra (negro) a GND, alimentación (rojo) a 3.5V y señal (amarillo) al A0



2. Pásale a Arduino este código:

```
//Juan Andres
const int potPin = A0;
int value;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  value=analogRead(potPin);
  value=map(value,0,690,0,255);
  Serial.println(value);
}
```

- value=analogRead(potPin): lee el valor del potenciómetro cuyo valor mínimo es 0 y máximo 690 (a 3.5V)
- map(value,0,690,0,255): mapea el valor del potenciómetro entre 0 y 255
- Serial.println(value): saca por puerto serie el valor y saca salto de linea '\n' (println) al final

3. Copia los siguientes 3 códigos con la misma estructura de ficheros anterior:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Juan A. Serial Data Viewer</title>
</head>

<body>
  <h1>Sensores (potenciometro) </h1>
  <div id="serialDataBox"></div>
  <script src="https://cdn.socket.io/4.1.3/socket.io.min.js"></script>
  <script src="sketch.js"> </script>

</body>
</html>
```

- crea el divisor donde JS escribirá en RT los valores del potenciómetro. Su id es la que aparece

sketch.js

```
//Juan A. Morales
const socket = io();
const serialDataBox = document.getElementById('serialDataBox');

/*Box potentiometer*/
socket.on('serialDataJuan', (data) => {
    // Replace the content of the box with the latest data
    serialDataBox.textContent = data;
});
```

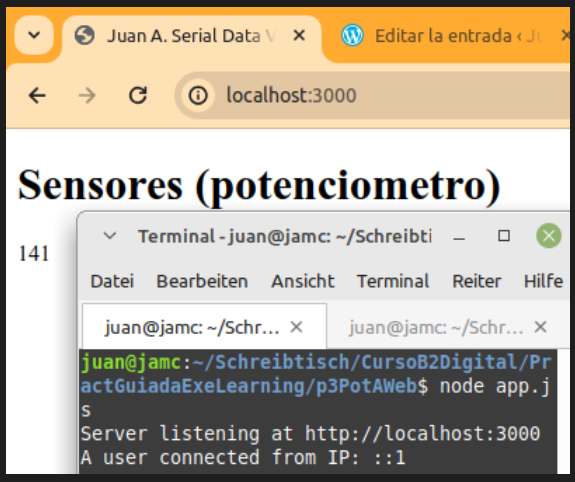
- const serialDataBox: objeto que representa el anterior
- socket.on('serialDataJuan', (data)..) cuando recibe un mensaje del sevidor con id 'serialDataJuan' y valor del potenciómetro, modifica el valor del objeto anterior.

app.js

```
//Juan A.
const { SerialPort } = require('serialport')
const serialport = new SerialPort({path: '/dev/ttyACM0', baudRate: 9600 })
const { ReadlineParser } = require('@serialport/parser-readline');
//
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
//
const app = express();
const server = http.createServer(app);
const io = socketIo(server);
const port = 3000;
//only the files from public can be served to a client
app.use(express.static('./publicJuan'));

//WebSocket connection handling
io.on('connection', (socket) => {  //server waiting for the connection client event
  //
  const userIP = socket.handshake.address; //IP del cliente
  console.log(`A user connected from IP: ${userIP}`); //message when client connected

  /*Box potentiometer*/
  // Create a parser to read lines from the serial port
  const parser = serialport.pipe(new ReadlineParser({ delimiter: '\n' }));
  // Listen for incoming serial data and emit it to the connected clients using WebSocket
  parser.on('data', (data) => { //when parser receives a data (until delimiter)
                                //execute the following
    io.emit('serialDataJuan', data.trim()); //broadcast message (with id=serialDataJuan)
                                            //to all clients connected to socket
  });
});
// Start the server
server.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

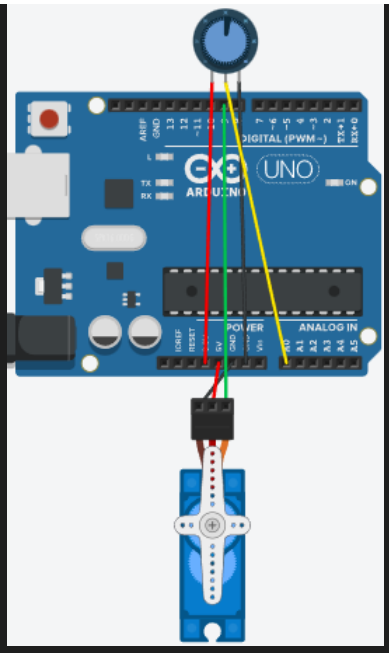
- const { ReadlineParser }: libreria de parseo (análisis) de datos del puerto serie
- const parser = serialport.pipe(new ReadlineParser({ delimiter: '\n' })): parseador que leerá dato hasta encontrar salto linea '\n'
- parser.on('data', (data)=>...): parseador escuhando y cuando detecta '\n' lo envia al servidor
- io.emit('serialDataJuan', data.trim()); emite mensaje al cliente con id 'serialDataJuan'

4. Resultado de la ejecución:



6. Control de un sensor y un actuador en RT (1h, en parejas)

Tarea: junta los dos códigos anteriores para crear una web que controle un servo (actuador) y un potenciómetro (sensor) como en la Fig.



Solución (mostrar cuando alumno suba la suya y que comparen)

Ejecución:

index.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Juan A. Serial Data Viewer</title>
</head>

<body>

  <h1> Actuadores (servo) </h1>
  <input type="range" id="slider" min="0" max="180" value="90">
  <p id="sliderValue">Slider Value: 90</p>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.1.3/socket.io.js"></script>


  <h1>Sensores (potenciometro) </h1>
  <div id="serialDataBox"></div>
  <script src="https://cdn.socket.io/4.1.3/socket.io.min.js"></script>


  <script src="sketch.js"> </script>

</body>
</html>
```

sketch.js:

```javascript
//Juan A. Morales
const socket = io();
const serialDataBox = document.getElementById('serialDataBox');
//
const slider = document.getElementById('slider');
const sliderValue = document.getElementById('sliderValue');

/*Slider servomotor*/
//sent slider value to server
slider.addEventListener('input', () => {
    const value = slider.value;
    socket.emit('sliderValueJuan', value);  //sent value with Id sliderValueJuan
});

//replace the slider value number with the latest
socket.on('updateSliderJuan', (value) => {  //receive server value with id updateSliderJuan
    slider.value = value;
    sliderValue.textContent = `Slider Valuei: ${value}`;
});

/*Box potentiometer*/
socket.on('serialDataJuan', (data) => {
    // Replace the content of the box with the latest data
    serialDataBox.textContent = data;
});
```

app.js:

```javascript
//Juan A.
const { SerialPort } = require('serialport')
const serialport = new SerialPort({path: '/dev/ttyACM0', baudRate: 9600 })
const { ReadlineParser } = require('@serialport/parser-readline');
//
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
//
const app = express();
const server = http.createServer(app);
const io = socketIo(server);
const port = 3000;
//only the files from public can be served to a client
app.use(express.static('./publicJuan'));

//WebSocket connection handling
io.on('connection', (socket) => {  //server waiting for the connection client event
  //
  const userIP = socket.handshake.address; //IP del cliente
  console.log(`A user connected from IP: ${userIP}`); //message when client connected

  /*Slider servomotor */
  // Function to execute when a  meassage with Id 'sliderValueJuan' is recieved
  socket.on('sliderValueJuan', (value) => { //receive client value
    io.emit('updateSliderJuan', value); // broadcast the updated value to all connected clients
    console.log(`Slider Values: ${value}`); //console
    serialport.write(`${value}\n`); //write value to serial port
   });

  /*Box potentiometer*/
  // Create a parser to read lines from the serial port
  const parser = serialport.pipe(new ReadlineParser({ delimiter: '\n' }));
  // Listen for incoming serial data and emit it to the connected clients using WebSocket
  parser.on('data', (data) => { //when parser receives a data (until delimiter)
                                //execute the following
    io.emit('serialDataJuan', data.trim()); //broadcast message (with id=serialDataJuan)
                                            //to all clients connected to socket
  });
});
// Start the server
server.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

7. Amplia el sistema (1 a 3h, en parejas)

Amplia mejorando estilo del CSS de la web (obligatorio)

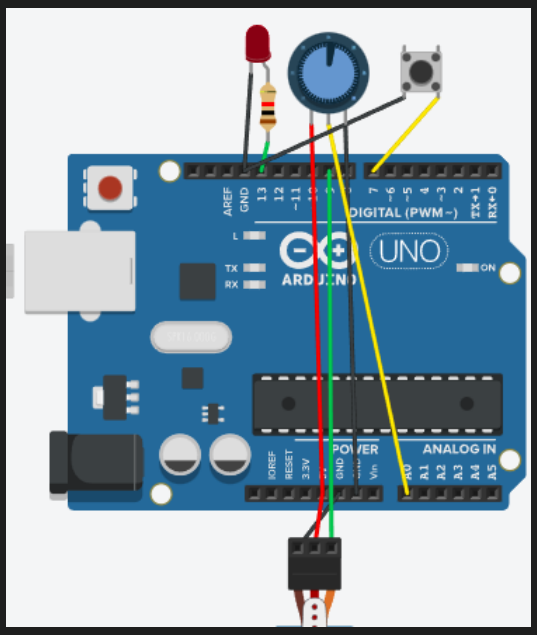
Tarea: dale estilo a tu web con CSS. Ej. que el valor del potenciómetro se muestre en un circulo que cambie de tamaño

Tarea: busca otras aplicaciones de tu sistema

Amplia añadiendo más sensores y actuadores (no para 2ºSMR)

Tarea-voluntaria (solo para 2º Bach. no para 2º SMR): añade al menos un sensor más (ej. botón al 7) y un actuador más (ej. LED al 13). Hazlo en función de la utilidad que le vas a dar a tu sistema IoT. Ej:

Circuito: ejemplo de ampliación



Código ampliaActSens.ino: donde al leer el botón le damos la vuelta a los bits, enviamos por puerto serie una cadena de tamaño fijo (añadiendo ceros con %03d) del estado de todos los sensores (ej. Po231Bu1) y recibimos por serie una cadena de tamaño fijo del estado deseado de todos los actuadores (ej. c10c21s1175)

```ino
#include <Servo.h>

const int ledRPin = 13; const int ledGPin = 12;
const int potPin = A0; const int servPin=9;
const int butPin = 7;
Servo myServo; // Create servo object
int potVal; int servPos; int butState; int ledRVal;
char s[8]; //sensor string to emit ex. Po%03dBu%d
char a[12];  //a="c10c21s1175"; string to receive

void setup()
{
  pinMode(ledRPin, OUTPUT);  pinMode(ledGPin, OUTPUT);
  pinMode(butPin, INPUT_PULLUP); myServo.attach(servPin);
  Serial.begin(9600);  myServo.write(90); // Calibrate Servo
}

void loop()
{
  /*Sensors*/
  potVal=analogRead(potPin); potVal=map(potVal,0,1023,0,255);  //for 5V
  //potVal=map(potVal,0,690,0,255); //for 3.5V
  //Sensor Buttom: we invert  digitalRead(butPin) is 1 (not pressed) or 0 (pressed)
  butState = 1-digitalRead(butPin);
  sprintf(s, "Po%03dBu%d", potVal, butState);
  Serial.println(s); //Emit sensor string

  /*Actuators*/
  if(Serial.available())    {
    Serial.readBytes(a,12);
    ledRVal = a[2]-48; //a[2]-'0'; convert to Int
    //servPos = (a[8]-'0')*100+(a[9]-'0')*10+(a[10]-'0');
    //servPos = (a[8]*100+a[9]*10+a[10]*1)-'0'*(100+10+1);
    servPos = (a[8]*100+a[9]*10+a[10])-5328;
    if(servPos >= 5 && servPos < 175) {myServo.write(servPos);}
    if(ledRVal==1) { digitalWrite(ledRPin, HIGH);}
    else {digitalWrite(ledRPin, LOW);  }
  }
}
```

index.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Juan A. Serial Data Viewer</title>
</head>

<body>
  <h1> Juan Andrés </h1>

  <h1> Actuadores (servomotor, luz) </h1>
  <p id="upVal"> Cadena valores de actuadores (re-actual): xxxxxxxxxx </p>
  <p> Servo: </p>
  <input type="range" id="slider" min="0" max="180" value="90">
  <p> Luces:   </p>
  <input type="checkbox" id="c1"> rojo </input>
  <input type="checkbox" id="c2"> verde </input>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.1.3/socket.io.js"></script>


  <h1>Sensores (potenciometro, botón) </h1>
  <p id="serialDataBox"> Cadena valores de sensores (actual): xxxxxxx </p>
  <p id="pot"> Potenciometro: xxx </p>
  <input type="checkbox" id="c3"> botón </input>
  <script src="https://cdn.socket.io/4.1.3/socket.io.min.js"></script>
  <script src="sketch.js"> </script>

</body>
</html>
```

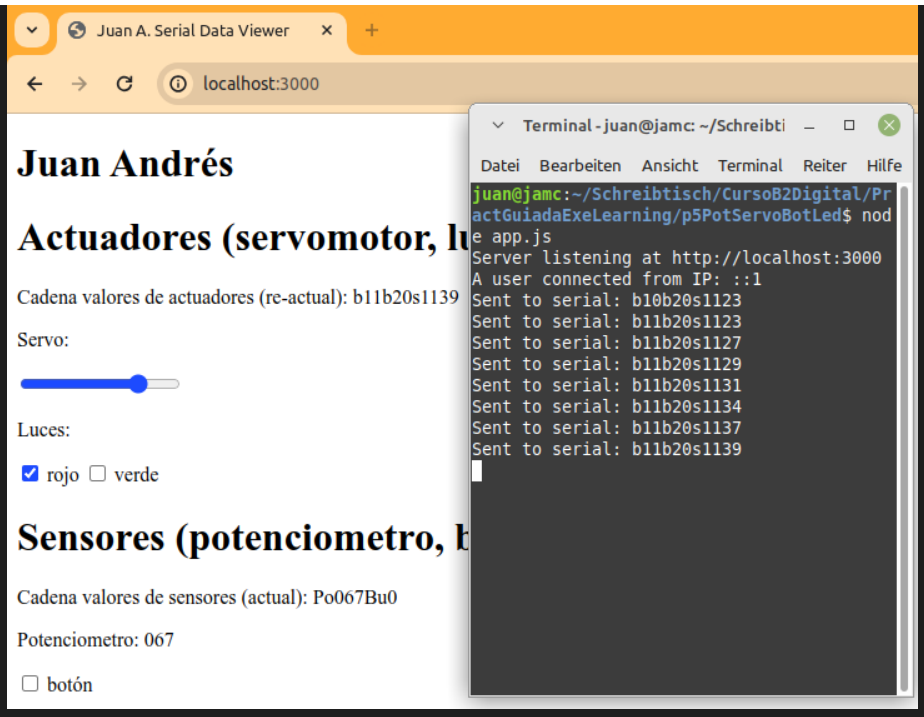
sketch.js:

```
//Juan A. Morales
const socket = io();
const serialDataBox = document.getElementById('serialDataBox');
const s1 = document.getElementById('slider');
const upVal = document.getElementById('upVal');
const c1 = document.getElementById('c1');
const c2 = document.getElementById('c2');
const c3 = document.getElementById('c3');
const p1 = document.getElementById('pot');
var eVal, s1ValPad;
/*Actuators: Slider and checker*/
function eFun(){
    //emited Values (+ c1.checked) convert bolean to 0/1
    s1ValPad=s1.value.toString().padStart(3, "0");
    eVal = 'b1' + (+ c1.checked) + 'b2' + (+ c2.checked) + 's1' + s1ValPad;
    socket.emit('valuesJuan', eVal);  //sent value with Id sliderValueJuan
};
//sent slider value to server
c1.addEventListener('change', eFun);
c2.addEventListener('change', eFun);
s1.addEventListener('input', eFun);
//replace the actuators values with the one reemited by the server»
socket.on('updatedValuesJuan', (val) => {  //receive server value with id updateSliderJuan
    s1.value=val.substring(8, 11);
    upVal.textContent = `Cadena valores de actuadores (re-actual): ${val}`;
});

/*Sensors: Box potentiometer and button*/
socket.on('serialDataJuan', (data) => {
    // Replace the content of the box with the latest data
    serialDataBox.textContent = `Cadena valores de sensores (actual): ${data}`; //Po255Bu1
    p1.textContent = `Potenciometro: ${data.substring(2,5)}`;
    c3.checked=!!+data.substring(7,8); //convert string '0' or '1' to false or true
});
```

app.js:

```
//Juan A.
const { SerialPort } = require('serialport')
const serialport = new SerialPort({path: '/dev/ttyACM0', baudRate: 9600 })
const { ReadlineParser } = require('@serialport/parser-readline');
const express = require('express'); const http = require('http');
const socketIo = require('socket.io');
const app = express(); const server = http.createServer(app);
const io = socketIo(server); const port = 3000;
//only the files from public can be served to a client
app.use(express.static('./publicJuan'));

//WebSocket connection handling
io.on('connection', (socket) => {  //server waiting for the connection client event
  const userIP = socket.handshake.address; //IP del cliente
  console.log(`A user connected from IP: ${userIP}`); //message when client connected
  /*Actuators: slider servomotor */
  // Function to execute when a  meassage with Id 'ValuesJuan' is recieved
  socket.on('valuesJuan', (rVal) => { //receive client value
    io.emit('updatedValuesJuan', rVal); // re-broadcast the updated value to all connected clients
    console.log(`Sent to serial: ${rVal}`); //console
    serialport.write(`${rVal}\n`); //write value to serial port
   });
  /*Sensors: box potentiometer*/
  // Create a parser to read lines from the serial port
  const parser = serialport.pipe(new ReadlineParser({ delimiter: '\n' }));
  // Listen for incoming serial data and emit it to the connected clients using WebSocket
  parser.on('data', (data) => { //when parser receives a data (until delimiter) execute that:
    io.emit('serialDataJuan', data.trim()); //broadcast message (with id=serialDataJuan)
    });                                     //to all clients connected to socket
});

// Start the server
server.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

Ejecución: observa la cadena del estado deseado de los actuadores re-actual (enviada por el cliente al servidor y devuelta al cliente ej. b11b20s1139) y la cadena de los sensores actual (enviada por el servidor ej. Po067Bu0)



Tarea: reflexiona, considerando la escalabilidad, las ventajas y desventajas de enviar desde el cliente los estados deseados de todos los actuadores en una sola cadena de texto. Haz lo mismo con la cadena de sensores. Discute otras alternativas.

Amplia el sistema con IA (voluntario)

Tarea-voluntaria: intenta hacer que el sistema sea controlado por una IA (ej. otra web)

8. **Defensa y evaluación del producto final (1h)**

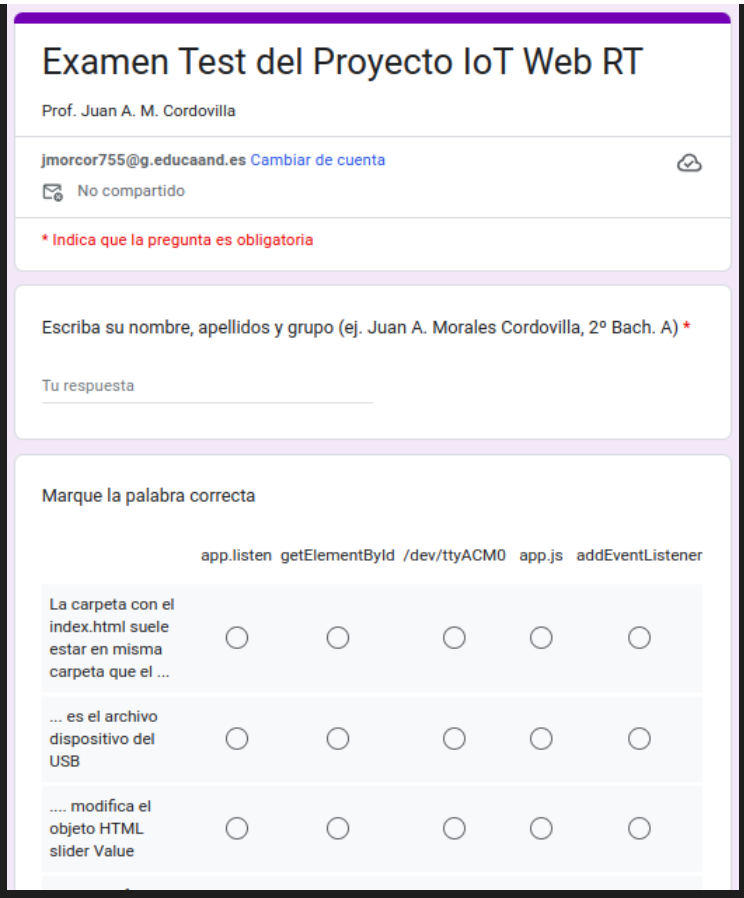
Defensa

Tarea: prepara una entrada de WordPress y presentala al profesor. Debe incluir al menos un video corto o GIF. La mejor propuesta será presentada al resto del grupo (el profesor usará la diana de evaluación de abajo).

Tarea: reflexiona sobre el impacto de tu sistema IoT en la sociedad.

Test autocorregible

Tarea: evaluate y evalua al profesor con el siguiente Test autocorregible: https://classroom.google.com/



Examen Test del Proyecto IoT Web RT

Prof. Juan A. M. Cordovilla

jmorcor755@g.educaand.es Cambiar de cuenta

No compartido

* Indica que la pregunta es obligatoria

Escriba su nombre, apellidos y grupo (ej. Juan A. Morales Cordovilla, 2º Bach. A) *

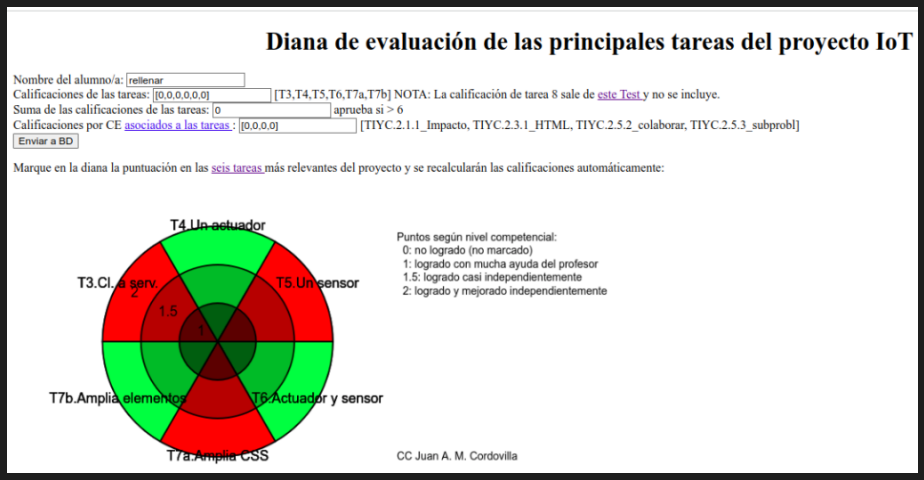
Tu respuesta

Marque la palabra correcta

	app.listen	getElementById	/dev/ttyACM0	app.js	addEventListener
La carpeta con el index.html suele estar en misma carpeta que el ...	○	○	○	○	○
... es el archivo dispositivo del USB	○	○	○	○	○
.... modifica el objeto HTML slider Value	○	○	○	○	○

Diana de evaluación

Diana de evaluación: de las seis tareas más significativas. Es interactiva y la usará el profesor



Diana de evaluación de las principales tareas del proyecto IoT

Nombre del alumno/a: rellenar
Calificaciones de las tareas: [0,0,0,0,0,0] [T3,T4,T5,T6,T7a,T7b] NOTA: La calificación de tarea 8 sale de este Test y no se incluye.
Suma de las calificaciones de las tareas: 0 aprueba si > 6
Calificaciones por CE asociados a las tareas : [0,0,0,0] [TIYC.2.1.1_Impacto, TIYC.2.3.1_HTML, TIYC.2.5.2_colaborar, TIYC.2.5.3_subprobl]
Enviar a BD

Marque en la diana la puntuación en las seis tareas más relevantes del proyecto y se recalcularán las calificaciones automáticamente:

Puntos según nivel competencial:
0: no logrado (no marcado)
1: logrado con mucha ayuda del profesor
1.5: logrado casi independientemente
2: logrado y mejorado independientemente

CC Juan A. M. Cordovilla

En este enlace puede probar la diana JavaScript interactivamente

diciembre 13, 2023 Juan Andrés Morales Cordovilla Sin categoría

Deja un comentario