WYLIONDRIN PARA RASPBERRY: EL SISTEMA DE SEÑALIZACION INTELIGENTE.

Autor: Ing. Orlando David Orbes

Docente: Ing. Jorge Andrés Cock Ramirez

Preparar el personal técnico que ha de enfrentar los retos tecnológicos de forma competente es uno de los compromisos que el SENA adquiere cuando asume su misión de formación profesional integral como respuesta a las necesidades del sector productivo. Como parte de la institución y considerando un rol de instructor en especialidades del sector industrial, se ha de asumir la responsabilidad de orientar procesos formativos bajo la estrategia de formación basada en proyectos, teniendo como reto la búsqueda continua de herramientas y medios didácticos que faciliten el proceso de aprendizaje, una tarea sin fin considerando la dinámica del cambio tecnológico en informática, electrónica, comunicaciones y automatización.

La búsqueda de documentos técnicos que sirvan de soporte teórico y que además proporcionen ejemplos de solución a retos reales, es una tarea que invita a evaluar la amplia producción literaria que se dispone en bibliotecas y bases de datos, así como también a recurrir a lugares en la web que comparten publicaciones de documentación técnica especializada en forma de libros como allitebooks.org. Es impresionante ver como la colección de documentos crece con mucha periodicidad dando atención a temas novedosos como desarrollo Web, programación, bases de datos, sistemas operativos, computo en la red y nube, administración, hardware y Diy, marketing y SEO, seguridad y software.

En esta búsqueda hay una serie de documentos que se desarrollan en torno a IoT y que permiten observar las tendencias tecnológicas relacionadas. En una selección reciente, se encontró el libro de Aplicaciones Comerciales e Industriales de Internet de las Cosas con Raspberry Pi de Ioana Culic, Alexandru Radovici y Cristian Rusu, de la editorial Apress y año de edición 2020, que llamo la atención y ha sido objeto de seguimiento y análisis. Esta obra se caracteriza por ir develando paso a paso los conceptos, herramientas y procedimientos relacionados con el desarrollo de productos comerciales e industriales orientados a brindar sistemas inteligentes para diversas aplicaciones como despliegue de información digital, dispensación de alimentos, generación de advertencias, medición de señales usando servidores industriales, almacenamiento y procesamiento y también la representación de los datos.

En esta tarea de aprender haciendo e intentando disponer con referentes y experiencia para transferir el conocimiento a los aprendices a través de actividades prácticas, se ha intentado seguir con atención la lectura y desarrollo de actividades conducentes a validar los casos resueltos.

Dado que es una actividad que se ha desarrollado desde hace algunos días y que no ha sido concluida, se comparte las observaciones alcanzadas en la revisión y ajuste del primer reto: Sistema de Despliegue Inteligente.

Los componentes que se integran en la solución son los siguientes:

- Raspberry Pi 3, tarjeta de desarrollo.
- Wyliondrin Studio, entorno de programación y depuración.
- Display Touchscreen, hardware para la interfaz de usuario.

Desarrollo de la aplicación

Accediendo a la página de Wyliodrin STUDIO, un entorno de desarrollo para aplicaciones IOT, se procede a hacer uso de la solución basada en Cloud que no requiere de instalación a través del botón habilitado. Sin embargo, si se dispone de recursos en el computador personal puede procederse con la descarga de la aplicación e instalación en la máquina local.

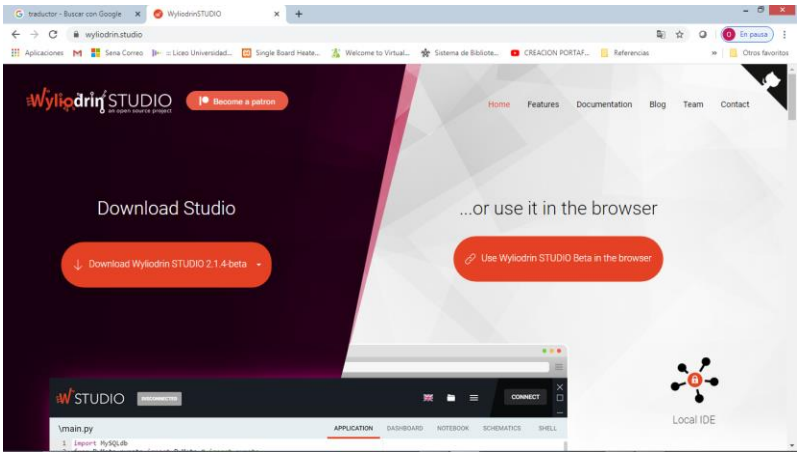

Imagen 1. Página de acceso al entorno de desarrollo

Durante el ingreso se despliega la interfaz de desarrollo y dando click al botón de CONECTAR, se procede a definir la conexión con la tarjeta Raspberry a ser usada para el proyecto. Se puede observar, que el entorno ofrece dispositivos simulados que temporalmente pueden ser usados para la programación.

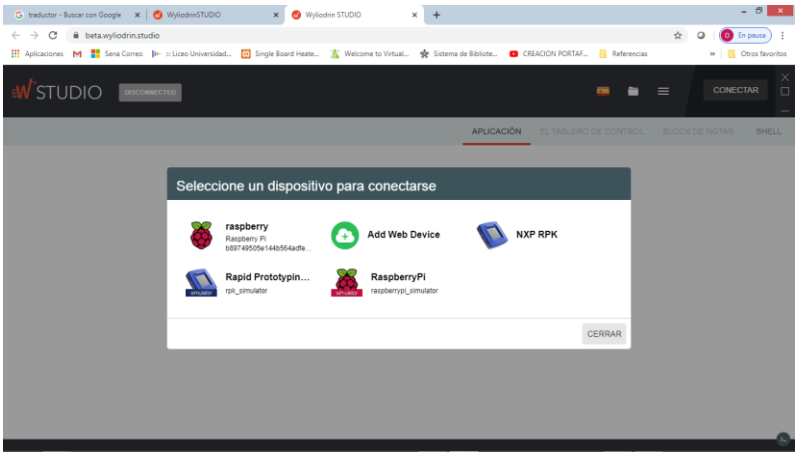

Imagen 2. Posibilidades de conexión con dispositivos.

Desde aquí se procede a elegir la opción de adicionar un nuevo dispositivo web, asignando un nombre que identificará la tarjeta para generar un archivo que permitirá la conexión remota.

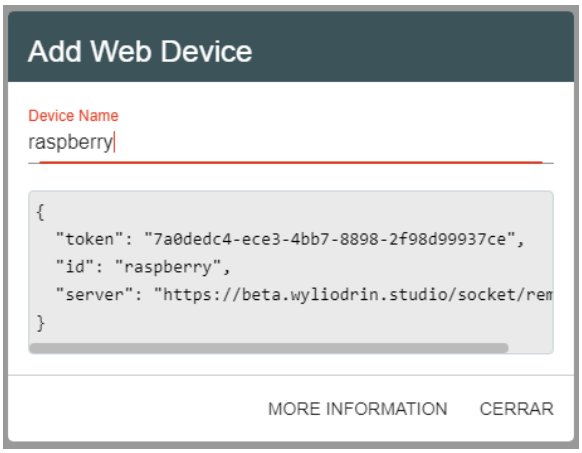

Imagen 3. Código fuente para configuración de conexión.

El código generado debe ser copiado y pegado en un nuevo archivo a denominar wyliodrin.json. Este archivo puede ser creado con un editor como Notepad++, a fin de que se garantice el tipo de archivo json, que si bien es texto debe tener esta extensión. Este archivo debe copiarse en la partición de arranque en la memoria SD que contiene la imagen descomprimida del sistema operativo ajustado para operar con el entorno de desarrollo.

Para descargar la imagen puede accederse desde https://wyliodrinstudio.readthedocs.io

La descompresión se hace en la tarjeta de memoria SD usando Etcher que garantiza la habilitación como dispositivo de arranque del sistema operativo. Para descargar Etcher puede hacerse desde este enlace: www.balena.io/etcher

Dispuesta la memoria y energizada la tarjeta de Raspberry Pi, es necesario configurar en el dispositivo el acceso a la red, para lo que se sugiere conectar un monitor y teclado. Inicie por primera vez, usando como usuario pi y contraseña raspberry; a continuación, usando derechos de administrador ejecute la herramienta de configuración con el comando sudo raspi-config.

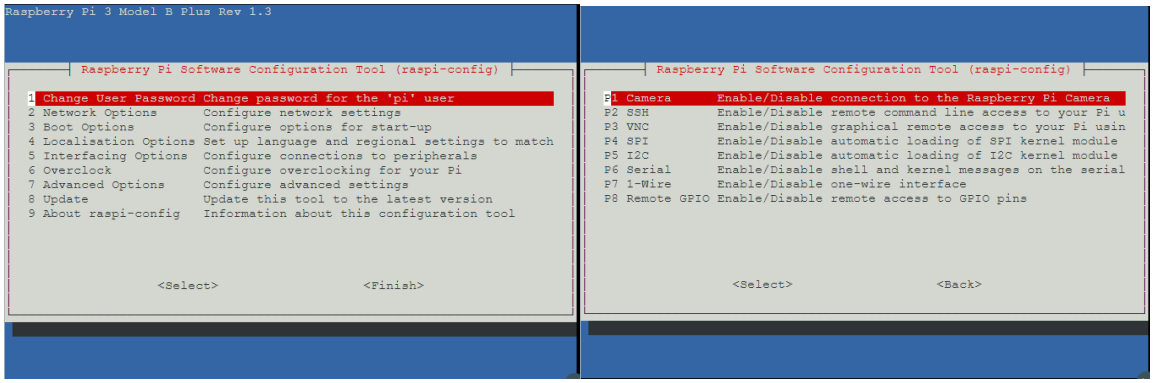

Imagen 4. Herramienta de configuración en la Raspberry.

Desde la interface desplegada, proceda a detallar las opciones de configuración requeridas relacionadas con la red para habilitación de wifi, ubicación con detalles de país e idioma, interfaz para habilitación de servicios SSH y VNC para uso de un equipo de cómputo remoto y opciones avanzadas para definir resolución de pantalla o salida de audio.

Se sugiere desde el prompt confirmar la comunicación con la red mediante el comando ifconfig y el uso de un ping a un equipo conocido de la red, como el equipo desde el que se ejecutará Wyliodin STUDIO.

Con esta verificación, estaría preparada la conexión a establecerse desde el entorno. Para lo cual desde el botón conectar, debería reportarse la tarjeta con el nombre asignado desde las opciones de selección de dispositivo.

Imagen 5. Reporte de dispositivos para el desarrollo

Elegido el dispositivo, procedemos a preparar el código para el proyecto. Desde el botón de Projects Library, procedemos a crear un nuevo proyecto, en este caso con NodeJS. Debe definirse el nombre, para este caso Despliegue.

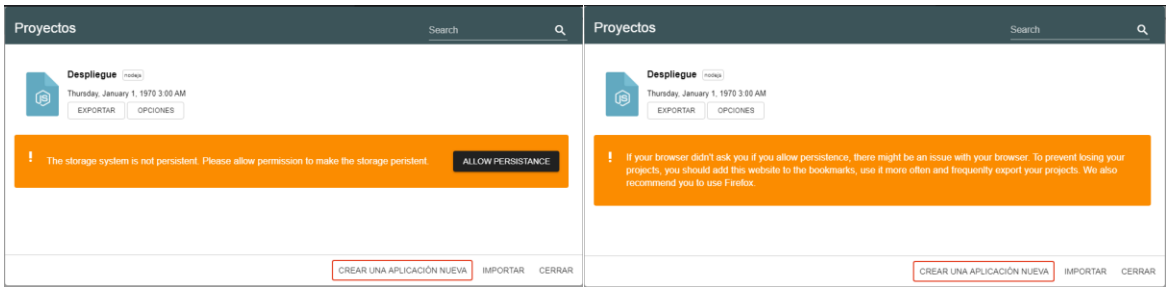

Imagen 6. Creación de proyecto

La ventana generada en el explorador recuerda que el almacenamiento es temporal o no persistente y por tanto debe autorizarse esta acción. Con un click sobre el icono del proyecto se accede al entorno de programación, presentándose un archivo por defecto que puede ser ejecutado para verificación de la conexión con la consola.

Es necesario configurar la opción de Use Advanced Mode desde el botón de menú con el propósito de habilitar el trabajo con varios archivos. A través del uso del click derecho se procede a crear la siguiente estructura de archivos, conservando el nombre y extensión correspondiente.

Imagen 7. Creación de directorio de archivos

Para empezar, se implementa a través de un archivo html, el despliegue de un mensaje que será presentado en la pantalla HDMI que acompaña la Raspberry. El código fuente puede observarse en la imagen.

Imagen 8. Código fuente de la página principal a ser desplegada.

Procedemos a ingresar el siguiente código al archivo index.html:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Bienvenido</title>
    </head>
    <body>
        <center>Despliegue Inteligente!</center>
        <center>SENA Regional Nariño</center>
    </body>
</html>
```

Este código despliega sobre la pantalla del navegador un título y un mensaje centrado.

A continuación se prepara el Archivo main.js. Este archivo lanza la aplicación y ha sido obtenido de los ejemplos de la biblioteca Electron que debe ser agregada al proyecto.

```
const electron = require('electron');
const path = require('path');
const url = require('url');
// Modulo para controlar la vida de la aplicacion
const app = electron.app;
// Modulo para crear un ventana de navegacion nativa.
const BrowserWindow = electron.BrowserWindow;
/* Mantiene una referencia global del objeto window, si no se hace,
la ventana sera cerrada automaticamente cuando el objeto JavaScript
sea basura recogida.*/
let mainWindow;
function createWindow ()
{
    // Crea la ventana del navegador.
    mainWindow = new BrowserWindow({
```

```
        height: 600,
        width: 800,
        frame: false,
        webPreferences: {
            nodeIntegration: true
        }
    });
    // y carga el index.html de la app.
    mainWindow.loadURL(url.format({
        pathname: path.join(__dirname, 'UI/index.html'),
        protocol: 'file:',
        slashes: true
    }));
    // Emitido cuando la ventana es cerrada.
    mainWindow.on('closed', function () {
    /* Quita la referencia al objeto window, usualmente deberia
    almacnar ventanas en un arreglo si la app soporta multiples
    ventanas, este es el momento cuando deberia borrar el
    correspondiente elemento.*/
        mainWindow = null;
    });
}
app.on('ready', createWindow);
// Salir cuando todas las ventanas esten cerradas.
app.on('window-all-closed', function () {
    app.quit();
});
app.on('activate', function () {
    if (mainWindow === null) {
        createWindow();
    }
});
```

Se procede con la habilitación de paquetes requeridos desde la herramienta de gestión con la instalación del paquete electron en la Raspberry.

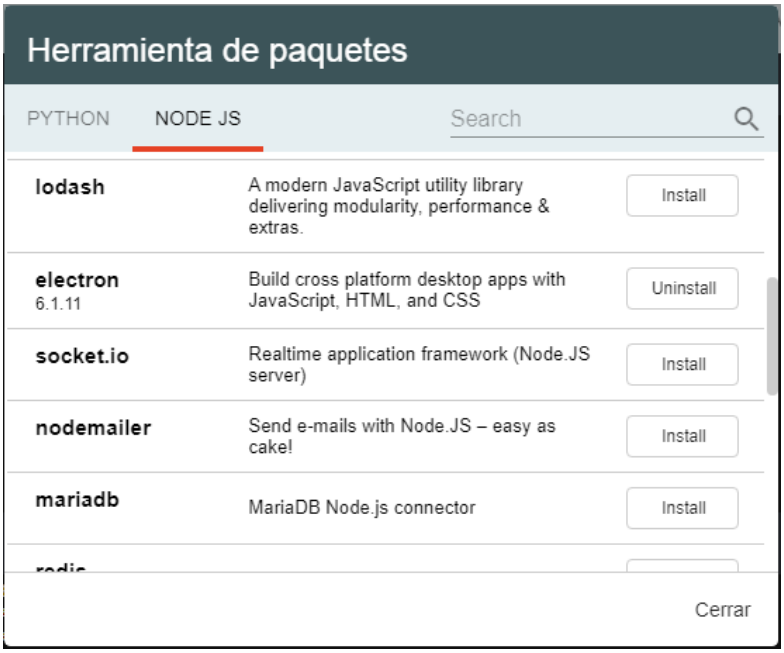

Imagen 9. Gestor para instalación de paquetes.

Por otra parte, es necesario instalar librerías adicionales desde la consola que se encuentra habilitada desde el entorno de Wyliondrin con los siguientes comandos:

sudo apt-get update
sudo apt-cache search libxss1
sudo apt-get install -y xinit xserver-xorg-core xserver-xorg-input-all xserver-xorg-video-fbturbo libgtk-3-0 libnss3 libnspr4 libgconf-2-4 libxtst6 libasound2 libxss1 --no-install-recommends

En este punto puede observarse la ejecución del Proyecto en la Raspberry con al ejecución del botón de play.

Conexión a Internet

Con objeto de agregar características de IoT a este proyecto, se modificará el código realizado a fín de obtener un panel de clima inteligente, el cual presentará información como estado del clima o temperatura actual. Para asegurarse que los datos desplegados son correctos, el panel actualiza la información cada 15 minutos.

Arquitectura de la aplicación

Se usará una API web que obtendrá datos en tiempo real del clima para reemplazar la información desplegada. Esto requiere reubicar el texto actual con el que cambio dentro de la aplicación. Ya que se tiene que escribir código JavaScript para implementar la aplicación lógica, podemos hacer uso de multitud de marcos para facilitar el proceso de desarrollo. Estos son diseñados para simplificar la manipulación de archivos HTML y hacen más fácil generar contenido dinámico. Se usará Vue.js

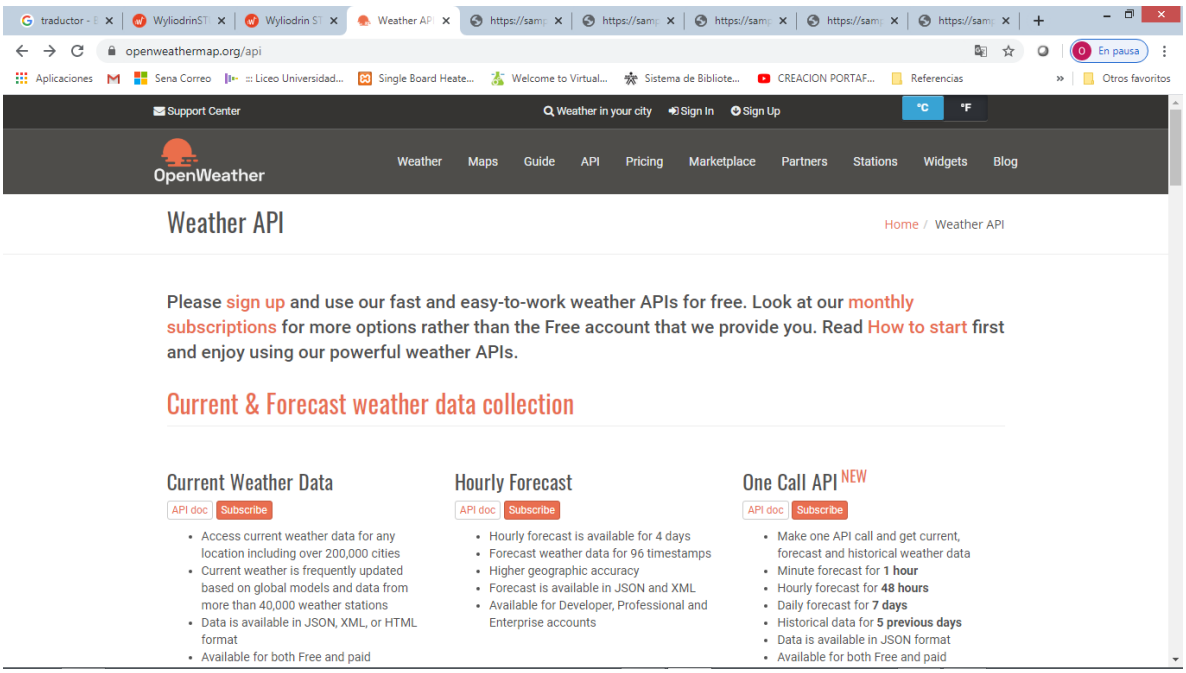

Imagen 10. Registro y acceso a OpenWeather para obtención de información del clima.

La solicitud información del clima requiere la creación de cuenta gratuita desde el enlace: https://openweathermap.org/api

Luego de la suscripción al plan gratuito, se solicitó la API key, que permite realizar hasta 60 llamadas por minuto. El código fue enviado a la cuenta de correo electrónico.

Para la ciudad de Pasto, se ajustó la llamada simplemente indicando el nombre de la ciudad y el país.

http://api.openweathermap.org/data/2.5/weather?q=Pasto,co&units=metric&APPID=12d3c392e452d2e5867dcb23dde61f08&lang=sp

El formato de la llamada puede ser ajustado con unidades de medida, lenguaje y formato de la respuesta a la solicitud en la red y puede ser accedido desde: https://openweathermap.org/current

El código para incorporar en el proyecto requiere de un nuevo archivo que se creará con el nombre app.js

Archivo de solicitud de información: app.js

```
var app = new Vue ({
    el: '#app',
    data () {
        return {
            weather: null,
            temperature: 0,
            error: ''
        }
    },
    created () {
        let getData = async ()=>{
            try{
                let response = await axios.get('http://api.openweathermap.org/data/2.5/weather?
                q=Pasto,co&units=metric&APPID=12d3c392e452d2e5867dcb23dde61f08&lang=sp');
                this.weather = response.data.weather[0].description;
                this.temperature = response.data.main.temp;
            }
            catch (err){
                this.error = err;
            }
            setTimeout (getData, 1000*60*15);
        };
        getData();
    }
});
```

Archivo index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Bienvenido</title>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/axios/0.19.0/axios.js"></script>
    </head>
    <body>
        <div id="app">
        <center> El clima afuera es {{weather}}. </center>
        <center> La temperatura es de {{temperature}} °C. </center>
        <br>
        <label> {{error}} </label>
```

```
        </div>
        <script src="app.js"></script>
    </body>
</html>
```

El archivo main.js y makefile son iguales al caso anterior.

Para mejorar el producto puede habilitarse el uso del modo gráfico del despliegue de manera que se usen imágenes que correspondan con el clima actual, jugando con el color y tamaño de fuentes de letra y fondos.

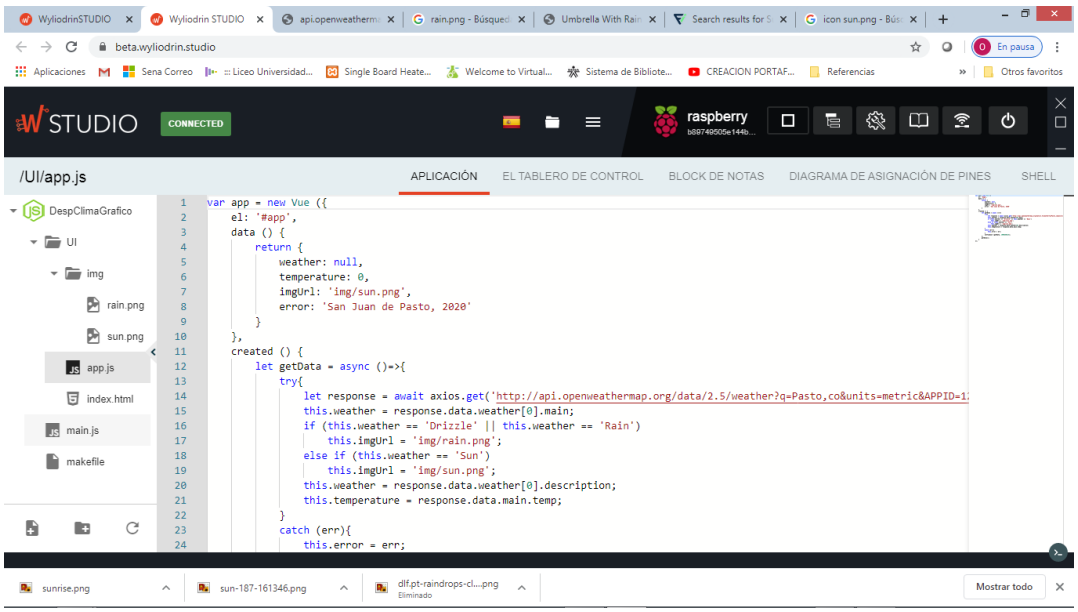

Imagen 11. Uso de Wylindrin en la edición y prueba del producto.

Los códigos que se modifican son el de index.html para la organización de contenido en web y el archivo app.js donde se accede a internet para descarga actualizada del clima y la obtención de datos para el despliegue: estado, descripción y temperatura. Acorde al estado se despliega la imagen correspondiente a un día soleado o lluvioso.

Archivo index.html:

```
<html lang="en">
<head>
    <title>Bienvenido</title>
    <style>
        body {background: blue; margin: 0; overflow: hidden; padding: 0; position:relative;
            font-size:21px; font-weight: 300; color:white;}
            #app {margin: auto; overflow: hidden; padding: 0;position:relative; width:100%;
            height:100%;}
            .app-box{top:0; bottom:0; left:0; right:0; margin:auto; display:block;
            padding:10px; position:absolute; height:fit-content; width: fit-content;}
            .w-status {padding:5px 0;}
            .error {position:absolute; bottom:0; text-align:center; width:100%; padding:5px 0;
            left:0; right:0;background:red; color:white;}
        </style>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/axios/0.19.0/axios.js"></script>
    </head>
    <body>
```

```
        <div id="app">
            <div class="app-box">
                <center>
                    <img v-bind:src=imgUrl>
                    <div class="w-status"> El clima afuera presenta {{weather}}.</div>
                    <div> La temperatura actual es {{temperature}} °C.</div>
                </center>
            </div>
            <label class="error"> {{error}} </label>
        </div>
        <script src="app.js"></script>
    </body>
</html>
```

Archivo app.js

```
var app = new Vue ({
    el: '#app',
    data () {
        return {
            weather: null,
            temperature: 0,
            imgUrl: 'img/sun.png',
            error: 'San Juan de Pasto, 2020'
        }
    },
    created () {
        let getData = async ()=>{
            try{
                let response = await axios.get('http://api.openweathermap.org/data/2.5/weather?q=
                Pasto,co&units=metric&APPID=12d3c392e452d2e5867dcb23dde61f08&lang=sp');
                this.weather = response.data.weather[0].main;
                if (this.weather == 'Drizzle' || this.weather == 'Rain')
                    this.imgUrl = 'img/rain.png';
                else if (this.weather == 'Sun')
                    this.imgUrl = 'img/sun.png';
                this.weather = response.data.weather[0].description;
                this.temperature = response.data.main.temp;
            }
            catch (err){
                this.error = err;
            }
            setTimeout (getData, 1000*60*15);
        };
        getData();
    }
});
```

Adicionalmente debe crearse un subdirectorio llamado img, para incluir en su interior dos imágenes correspondientes a sun.png y rain.png. Se sugiere descargar iconos desde internet considerando el tamaño justo para ser desplegado.

CONCLUSIONES

Se pueden realizar algunas observaciones respecto del proceso de desarrollo con la herramienta considerando las diferentes etapas:

- Configuración: Exige la preparación de la Raspberry, debiendo descargar en memoria microSD la imagen sugerida para la comunicación con Wyliondrin, incorporar datos para la conexión en la memoria SD y durante el arranque configurar los datos de red wifi en la tarjeta. Para facilitar esta tarea, podría proporcionarse un tutorial con el paso a paso sugiriendo realizar un vídeo o descripción del propósito de cada comando en el PC o en la Raspberry.
- Conexión: En adelante, basta con abrir el programa y dar la orden de conexión inalámbrica para habilitar el trabajo con la Raspberry, es decir sin requerir de cable, ni de pantalla, teclado o ratón.
- Edición y compilación: El uso de html y java script en este primer caso, requiere tener conceptos previos de programación, sin embargo, brinda también una oportunidad para el análisis, la consulta o búsqueda de elementos del lenguaje que permitan mejorar las características del programa. La motivación por el logro puede en este caso dirigir los esfuerzos para el aprendizaje significativo.
- Llama la atención el desarrollo progresivo de la solución, desde una implementación muy simple a la que se agregan dos cambios sucesivos importantes que terminan con una solución de despliegue inteligente que hace uso de la nube para traer y presentar información real.
- Los códigos sugeridos por los autores han sido verificados y ajustados, superando algunas dificultades principalmente ocasionados por falta de detalle o de interpretación en el documento.
- En general se puede decir que esta herramienta representa una gran posibilidad para el desarrollo de soluciones programables y de IoT, liberando en el ambiente formativo la necesidad de disponer de monitores y teclados dedicados a cada Raspberry al ser implementados directamente desde el computador; por otra parte, la existencia de un simulador así como herramientas de depuración favorecen la construcción de soluciones.

REFERENCIAS

1. Culic, Ioana y otros. (2020). Commercial and Industrial Internet of Things Applications with the Raspberry Pi – Prototyping IoT Solutions. Apress.

REFERENCIAS A INTERNET

1. https://wyliodrin.studio/
2. https://code.visualstudio.com/
3. https://openweathermap.org
4. www.raspberrypi.org/products/compute-module-3/

BIOGRAFÍA

Orlando David Orbes es Instructor en Electrónica en el SENA la Regional Nariño, orienta formación en los programas de tecnología en Automatización Industrial y de técnico en Implementación y Mantenimiento de Equipo Electrónico Industrial. La especialización en Redes y Servicios Telemáticos, así como la formación recibida en la entidad en temas de automatización y electrónica embebida ha permitido la orientación de proyectos con aprendices como solución a retos de control de ambientes controlados como hornos e invernaderos, así como también en temas de telemetría como el proyecto de Formula SENA Eco. Actualmente desarrolla proyecto de modernización del ambiente de formación con objeto de incorporar nuevas tecnologías asociadas con instrumentación virtual, así como la vinculación de recursos de IoT a los ambientes formativos y productivos existentes en la finca Lope del Centro Internacional de producción Limpia Lope en la ciudad de Pasto.