Manual de Usuario de la Plataforma EMIC

1. Introducción a EMIC

1.1. ¿Qué es EMIC?

EMIC, que significa Electrónica Modular Inteligente Colaborativa, es una plataforma innovadora diseñada para el desarrollo colaborativo de soluciones tecnológicas en el contexto de la Industria 4.0. EMIC es un gestor de código que proporciona lineamientos de estandarización, permitiendo la conexión eficiente de código proveniente de diversos desarrolladores.

La plataforma EMIC integra librerías organizadas por tipo de soporte físico (microcontroladores, web, Dockers) en repositorios de GitHub. Los usuarios pueden utilizar el editor EMIC para crear aplicaciones que incluyan hardware y software basados en las librerías preexistentes.

1.2. Historia y evolución

EMIC comenzó como una herramienta para crear programas en lenguaje C basados en código reutilizable. Su evolución ha sido impulsada por la necesidad de mejorar la productividad y la calidad en el desarrollo de sistemas embebidos y soluciones para la Industria 4.0.

Hitos clave en la evolución de EMIC:

1. Normalización de la escritura de bibliotecas con buenas prácticas.
2. Desarrollo de reglas para la reutilización automática de funciones.
3. Organización del código en carpetas por niveles de abstracción.
4. Creación de una aplicación para automatizar la generación de programas.
5. Desarrollo de un entorno de desarrollo en la nube con editor y compiladores.
6. Diseño de un protocolo para comunicar microcontroladores.

1.3. Estado actual de EMIC

Actualmente, EMIC es una plataforma dinámica en constante crecimiento. Sus características principales incluyen:

- Múltiples repositorios con bibliotecas para diferentes tecnologías (PIC24, desarrollo web, redes neuronales, etc.).
- Más de 50 módulos de hardware desarrollados por la comunidad.
- Un editor visual tipo drag&drop para la creación de aplicaciones.
- Capacidad para generar código para diferentes plataformas (microcontroladores, web, servicios en la nube).

1.4. El futuro de EMIC

EMIC continúa evolucionando con el objetivo de proporcionar soluciones cada vez más completas y eficientes. Algunas áreas de desarrollo futuro incluyen:

- Incorporación de nuevas funciones y tecnologías.

- Experimentación con procesamiento de datos e inteligencia artificial.
- Desarrollo de un asistente para el diseño de hardware.
- Expansión de la comunidad de desarrolladores y usuarios.

2. Fundamentos de EMIC

2.1. Componentes principales

EMIC se compone de varios elementos clave:

1. **Módulos**: Unidades fundamentales que representan hardware, aplicaciones web, servicios en la nube, o modelos de ML.
2. **Drivers**: Grupos de funciones y características que definen los módulos.
3. **Repositorios**: Almacenes de código en GitHub donde se organizan las bibliotecas y módulos.
4. **Editor EMIC**: Herramienta visual para la creación de aplicaciones mediante drag&drop.
5. **EMIC Codify**: Lenguaje de programación específico para el manejo de documentos y código en EMIC.

2.2. Funcionamiento del sistema

El sistema EMIC opera a través de varios procesos clave:

1. **EMIC Discovery**: Transforma los documentos en información utilizable por el editor.
2. **EMIC Transcriptor**: Genera código específico a partir del script creado en el editor.
3. **EMIC Merge**: Combina los documentos originales con el script transcrito para generar el código final.
4. **EMIC Compiler**: Compila el código generado si es necesario.

Estos procesos trabajan con tres tipos de almacenes de documentos:

- **SOURCE Documents**: Contiene el código fuente y la información adicional creada por los desarrolladores.
- **SYSTEM Documents**: Almacena los documentos generados por el transcriptor.
- **TARGET Documents**: Guarda el resultado final útil.

2.3. Tipos de módulos y recursos

EMIC soporta diversos tipos de módulos y recursos:

1. **Módulos de hardware**: Representan dispositivos electrónicos físicos y sus capacidades.
2. **Módulos de software**: Incluyen aplicaciones web, servicios en la nube, y modelos de ML.
3. **Widgets**: Componentes visuales para interfaces de usuario.
4. **Drivers**: Conjuntos de funciones que manejan recursos específicos de hardware o software.
5. **Bibliotecas**: Colecciones de código reutilizable para diversas funcionalidades.

Cada tipo de módulo y recurso juega un papel específico en el ecosistema EMIC, permitiendo a los desarrolladores e integradores crear soluciones completas y eficientes para la Industria 4.0.

3. Comenzando con EMIC

3.1. Requisitos del sistema

Para utilizar la plataforma EMIC, necesitarás:

- Un navegador web moderno (Chrome, Firefox, Safari o Edge)
- Conexión a internet estable
- Cuenta de Google (para acceder al editor EMIC)
- Conocimientos básicos de programación (para integradores)
- Conocimientos avanzados de programación (para desarrolladores)

3.2. Registro y acceso a la plataforma

1. Visita editor.emic.io
2. Inicia sesión con tu cuenta de Google
3. Acepta los términos y condiciones de uso de EMIC
4. Completa tu perfil de usuario (opcional, pero recomendado)

3.3. Navegación por la interfaz de usuario

La interfaz de EMIC se divide en varias áreas principales:

1. **Barra de navegación**: Acceso a proyectos, repositorios y configuración de cuenta
2. **Panel de módulos**: Lista de módulos disponibles organizados por categorías
3. **Área de trabajo**: Espacio donde se construyen y editan los proyectos
4. **Panel de propiedades**: Muestra y permite editar las propiedades del elemento seleccionado
5. **Consola**: Muestra mensajes del sistema, errores y resultados de compilación

Familiarízate con estas áreas, ya que las utilizarás frecuentemente en tus proyectos EMIC.

4. Uso de EMIC para Integradores

4.1. Creación de proyectos integrados

1. Navegue a la carpeta "My Projects" desde la pantalla principal.
2. Haga clic en "Add Project" para iniciar un nuevo proyecto.
3. Se abrirá una nueva ventana titulada con el nombre de su proyecto.

4.2. Interfaz de creación y edición de proyectos

La interfaz de proyecto en EMIC consta de varios elementos clave:

1. Barra de título: Muestra el nombre del proyecto y los módulos incluidos (por ejemplo, "Proyecto1: HRD_USB_1 HRD_X2_RELAY_1").

2. Pestañas de navegación:

    - Configuración
    - Módulo
    - Usuario
    - Librerías
    - Reciclador

3. Panel izquierdo: Muestra los componentes y funciones disponibles en el proyecto:

    - LED

- TIMER
- USB
- EMICbus
- Funciones (con opciones como "Send")
- Eventos (con opciones como "Reception")
- SYSTEM

4. Panel central: Área de código donde se desarrolla la lógica del proyecto.

5. Barra inferior: Contiene botones para Save, Generate, Program, Run, y muestra el estado del proyecto.

4.3. Selección y configuración de módulos

1. En la fase inicial del proyecto, use el panel izquierdo para seleccionar los módulos necesarios.
2. Arrastre los módulos deseados al área de trabajo.
3. Configure cada módulo según sea necesario utilizando el panel de propiedades (no visible en la imagen actual, pero típicamente disponible al seleccionar un módulo).

4.4. Desarrollo de la lógica del proyecto

Una vez que los módulos estén en su lugar, puede comenzar a desarrollar la lógica de su proyecto en el panel central de código:

1. Defina eventos del sistema:

```
Event.SYSTEM.preReset
Led1.blink(100,200,5)
```

2. Maneje la comunicación entre módulos:

```
Event.USB.Reception(tag,msg)
Led1.blink(100,200,1)
EMICbus.Send(tag,msg)

Event.EMICbus.reception(TAG,Message)
Led1.blink(100,200,2)
USB.Send(TAG,Message)
```

3. Utilice las funciones específicas de cada módulo (como `blink` para LEDs, `Send` para USB y EMICbus).

4.5. Generación y ejecución del proyecto

1. Haga clic en "Save" para guardar los cambios en su proyecto.
2. Utilice "Generate" para que EMIC procese su script y genere el código final.
3. Si es necesario, use "Program" para cargar el código en el hardware.
4. Haga clic en "Run" para ejecutar su proyecto.
5. El estado del proyecto se mostrará en la barra inferior (por ejemplo, "inactivo").

4.6. Consejos para integradores

- Familiarícese con los diferentes componentes disponibles en el panel izquierdo.
- Utilice las pestañas de navegación para acceder a diferentes aspectos de su proyecto, como configuración y librerías.
- Pruebe su proyecto frecuentemente durante el desarrollo utilizando las opciones de Generate y Run.
- Consulte la documentación específica de cada módulo para comprender todas sus capacidades y cómo integrarlas efectivamente en su proyecto.

4. Uso de EMIC para Integradores

4.1. Creación de proyectos integrados

Para crear un nuevo proyecto:

1. Haz clic en "Nuevo Proyecto" en la barra de navegación
2. Asigna un nombre y una descripción a tu proyecto
3. Selecciona el tipo de proyecto (por ejemplo, aplicación web, sistema embebido, etc.)
4. Haz clic en "Crear"

4.2. Uso del editor visual

El editor visual de EMIC permite construir proyectos mediante drag&drop:

1. Navega por el panel de módulos para encontrar los componentes que necesitas
2. Arrastra los módulos al área de trabajo
3. Conecta los módulos según la lógica de tu aplicación
4. Utiliza el panel de propiedades para configurar cada módulo

Consejos:

- Utiliza la función de zoom para una mejor visualización de proyectos complejos
- Agrupa módulos relacionados para mantener tu proyecto organizado

4.3. Selección y configuración de módulos

Para cada módulo en tu proyecto:

1. Haz clic en el módulo para seleccionarlo
2. Utiliza el panel de propiedades para ajustar su configuración
3. Asigna valores a los parámetros requeridos
4. Conecta las entradas y salidas del módulo con otros componentes del proyecto

Recuerda:

- Algunos módulos pueden requerir configuración adicional en archivos separados
- Consulta la documentación específica de cada módulo para obtener detalles sobre su uso y configuración

4.4. Generación y compilación de proyectos

Una vez que hayas completado tu proyecto:

1. Haz clic en el botón "Generar" en la barra de herramientas
2. EMIC procesará tu proyecto y generará el código correspondiente
3. Si es necesario, el sistema compilará automáticamente el código generado
4. Revisa la consola para ver el progreso y cualquier mensaje de error

Si la generación y compilación son exitosas:

5. EMIC te proporcionará un enlace para acceder a tu aplicación (en caso de proyectos web)
6. Para proyectos de hardware, encontrarás instrucciones para cargar el firmware en tu dispositivo

En caso de errores:

7. Lee cuidadosamente los mensajes de error en la consola
8. Corrige los problemas indicados en tu proyecto
9. Vuelve a intentar la generación y compilación

Recuerda guardar tu proyecto regularmente para evitar pérdidas de trabajo.

5. Desarrollo en EMIC

5.1. Configuración del entorno de desarrollo

1. Acceso al entorno de desarrollo:
    - Ingrese a editor.emic.io con su cuenta de usuario.
    - Navegue a la carpeta "DEV" desde la pantalla principal.
2. Repositorios disponibles:
    - dashboard: Para desarrollo de interfaces de usuario.
    - PIC: Para desarrollo relacionado con microcontroladores PIC.
    - EMIC: Para desarrollo del núcleo de la plataforma EMIC.
3. Clonar un repositorio:
    - Utilice la opción "Clone Repository" para añadir nuevos repositorios a su entorno de desarrollo.

5.2. Estructura de repositorios EMIC

Los repositorios EMIC siguen una estructura específica:

```
EMIC (DEV:)
 |
 ├── _modules
 |    |
 |    └── Category1
 |         |
 |         ├── Module1
 |         |    |
 |         |    ├── System
```

```
│       │   │   │
│       │   │   └── generate.emic
│       │   │
│       │   └── m_description.json
│       │
│       └── Module2
│           │
│           ├── System
│           │   │
│           │   └── generate.emic
│           │
│           └── m_description.json
```

5.3. Creación de módulos

1. Cree una nueva carpeta dentro de la categoría apropiada en `_modules`.
2. Añada un archivo `m_description.json` en la raíz del módulo:

```
{
  "type": "hardware"  // o "web", "cloud", etc.
}
```

3. Cree una carpeta `System` dentro del módulo.
4. Añada un archivo `generate.emic` dentro de la carpeta `System`.

5.4. Desarrollo de recursos (drivers, bibliotecas, etc.)

1. Creación de drivers:
   - Desarrolle el código del driver en el lenguaje apropiado (C, JavaScript, etc.).
   - Incluya comentarios descriptivos para facilitar el uso por otros desarrolladores.
2. Desarrollo de bibliotecas:
   - Cree funciones reutilizables que puedan ser empleadas en múltiples proyectos.
   - Documente adecuadamente cada función y su uso.
3. Uso de EMIC Codify:
   - Utilice comandos EMIC Codify en sus archivos para definir cómo se integrarán en los proyectos.
   - Ejemplo básico en `generate.emic`:

```
EMIC:setOutput(TARGET:src/myModule.c)
// Código del módulo
EMIC:restoreOutput
```

5.5. Pruebas y validación

1. Cree proyectos de prueba para validar sus nuevos módulos y recursos.
2. Utilice la interfaz de proyecto que vimos en la sección 4 para probar la funcionalidad.
3. Verifique que los eventos y funciones de su módulo se integren correctamente con otros componentes.

5.6. Contribución a la comunidad EMIC

1. Documente detalladamente sus módulos y recursos.
2. Utilice el sistema de control de versiones (Git) para gestionar sus cambios.
3. Cuando esté listo, proponga sus contribuciones para ser incluidas en los repositorios principales de EMIC.

5.7. Mejores prácticas

- Siga las convenciones de nomenclatura establecidas en EMIC.
- Mantenga sus módulos y recursos lo más modulares posible para facilitar la reutilización.
- Pruebe exhaustivamente antes de contribuir a los repositorios principales.
- Manténgase actualizado con las últimas características y cambios en la plataforma EMIC.

6. EMIC Codify

6.1. Introducción a EMIC Codify

EMIC Codify es un lenguaje de programación específico creado para el manejo de documentos de texto y código dentro de la plataforma EMIC. Permite establecer relaciones entre la definición de los módulos EMIC y sus dependencias, así como adaptar las bibliotecas a las necesidades específicas de cada proyecto.

6.2. Sintaxis básica

Los comandos en EMIC Codify siguen generalmente este formato:

```
EMIC:comando(parámetros)
```

El texto que no contiene comandos se enviará directamente al archivo de salida, a menos que contenga expresiones de la forma `.{xxx.yyy}.`, que serán reemplazadas por valores definidos previamente.

6.3. Comandos principales

6.3.1. setInput

Inicializa el procesamiento de un archivo.

Sintaxis:

```
EMIC:setInput([origin:][dir/]file[[,key=value]])
```

6.3.2. setOutput

Establece el archivo de salida.

Sintaxis:

```
EMIC:setOutput([target:][dir/]file)
```

6.3.3. restoreOutput

Restablece el archivo de salida al destino anterior.

Sintaxis:

```
EMIC:restoreOutput
```

6.3.4. copy

Procesa un archivo y envía el resultado a un archivo de salida especificado.

Sintaxis:

```
EMIC:copy([origin:][dir1/]file1,[target:][dir2/]file2[[,key=value]])
```

6.3.5. define

Define una nueva macro.

Sintaxis:

```
EMIC:define([group.]key,value)
```

6.3.6. unDefine

Borra una macro.

Sintaxis:

```
EMIC:unDefine([group.]key)
```

6.4. Estructuras de control

6.4.1. Bucles

```
EMIC:foreach(group)
  .{Item}.
EMIC:endfor
```

6.4.2. Condicionales

```
EMIC:if(condition)
  // código
EMIC:elif(condition)
  // código
EMIC:else
  // código
EMIC:endif
```

6.5. Ejemplos de uso

6.5.1. Generación de código para un módulo

```
EMIC:setOutput(TARGET:src/myModule.c)
#include "myModule.h"

void initModule() {
  // Código de inicialización
}

.{functions}.

EMIC:restoreOutput
```

6.5.2. Definición y uso de macros

```
EMIC:define(MODULE.NAME, "MyAwesomeModule")
EMIC:define(MODULE.VERSION, "1.0.0")

// Uso de las macros
Module Name: .{MODULE.NAME}.
Version: .{MODULE.VERSION}.
```

6.5.3. Uso de condicionales

```
EMIC:if(PLATFORM == "PIC24")
  // Código específico para PIC24
```

```
EMIC:elif(PLATFORM == "ARM")
  // Código específico para ARM
EMIC:else
  // Código por defecto
EMIC:endif
```

6.6. Mejores prácticas

1. Utilice macros para hacer su código más flexible y reutilizable.
2. Organice su código en secciones lógicas utilizando los comandos de EMIC Codify.
3. Aproveche las estructuras de control para generar código específico según las necesidades del proyecto.
4. Documente el uso de EMIC Codify en sus módulos para facilitar su comprensión por otros desarrolladores.

6.7. Depuración

Cuando se encuentren problemas con el código generado:

1. Revise los archivos de salida generados por EMIC Codify.
2. Verifique que todas las macros utilizadas estén correctamente definidas.
3. Asegúrese de que todos los comandos de EMIC Codify estén correctamente formateados.

8. Tutoriales y Ejemplos

8.1. Creación de un proyecto simple

En este tutorial, crearemos un proyecto básico que utiliza un módulo USB para recibir datos y un LED para indicar la recepción.

Paso 1: Crear un nuevo proyecto

1. Navegue a la carpeta "My Projects" desde la pantalla principal.
2. Haga clic en "Add Project".
3. Nombre su proyecto "USB_LED_Example".

Paso 2: Añadir módulos

1. En el panel izquierdo, busque y añada los siguientes módulos:
   - HRD_USB_1
   - LED

Paso 3: Configurar el proyecto

1. En el panel central, escriba el siguiente código:

```
Event.SYSTEM.preReset
  Led1.blink(100,200,5)
```

```
Event.USB.Reception(tag,msg)
  Led1.blink(100,200,1)
  USB.Send(tag,msg)
```

Paso 4: Generar y ejecutar

1. Haga clic en "Save" para guardar su proyecto.
2. Haga clic en "Generate" para procesar su código.
3. Si está conectado al hardware, haga clic en "Program" para cargar el código.
4. Haga clic en "Run" para ejecutar el proyecto.

Resultado

Ahora tiene un proyecto que hace parpadear el LED al inicio y cada vez que se reciben datos por USB, y luego reenvía esos datos.

8.2. Desarrollo de un módulo personalizado

En este tutorial, crearemos un módulo personalizado para un sensor de temperatura.

Paso 1: Crear la estructura del módulo

1. En la carpeta DEV, navegue a `_modules`.
2. Cree una nueva carpeta llamada "CustomSensors".
3. Dentro de "CustomSensors", cree una carpeta llamada "TempSensor".
4. Dentro de "TempSensor", cree una carpeta "System" y un archivo `m_description.json`.

Paso 2: Definir el módulo

En `m_description.json`, añada:

```
{
  "type": "hardware",
  "name": "TempSensor",
  "version": "1.0.0",
  "description": "Custom temperature sensor module"
}
```

Paso 3: Crear el archivo generate.emic

En la carpeta "System", cree un archivo `generate.emic` con el siguiente contenido:

```
EMIC:setOutput(TARGET:src/tempSensor.c)
#include "tempSensor.h"

void initTempSensor() {
  // Initialization code
```

```
}

float readTemperature() {
  // Code to read temperature
  return temperature;
}

EMIC:restoreOutput

EMIC:setOutput(TARGET:inc/tempSensor.h)
#ifndef TEMP_SENSOR_H
#define TEMP_SENSOR_H

void initTempSensor();
float readTemperature();

#endif
EMIC:restoreOutput
```

Paso 4: Utilizar el nuevo módulo

Ahora puede utilizar este módulo en sus proyectos como cualquier otro módulo de hardware.

8.3. Integración de hardware y software

En este ejemplo, integraremos nuestro sensor de temperatura personalizado con una aplicación web para mostrar la temperatura en tiempo real.

Paso 1: Crear un nuevo proyecto

1. Cree un nuevo proyecto llamado "TempMonitor".
2. Añada los módulos:
    - TempSensor (nuestro módulo personalizado)
    - HRD_WiFi
    - WebApp

Paso 2: Configurar el hardware

En el panel de código para el hardware, añada:

```
Event.SYSTEM.preReset
  TempSensor.init()
  WiFi.connect("SSID", "PASSWORD")

Event.TIMER.tick(5000)  // Every 5 seconds
  float temp = TempSensor.readTemperature()
  WebApp.send("temperature", temp)
```

Paso 3: Configurar la aplicación web

En el panel de código para la aplicación web, añada:

```
// HTML
<div id="tempDisplay"></div>

// JavaScript
socket.on('temperature', function(temp) {
  document.getElementById('tempDisplay').innerHTML = 'Temperature: ' + temp +
'°C';
});
```

Paso 4: Generar y ejecutar

1. Guarde, genere y ejecute el proyecto como en ejemplos anteriores.

Resultado

Ahora tiene una aplicación que lee la temperatura cada 5 segundos y la muestra en una página web en tiempo real.