UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS

FACULTAD TECNOLÓGICA

SEMESTRE 2024-1

TECNOLOGÍA EN ELECTRICIDAD DE MEDIA Y BAJA TENSIÓN

INCLUSIÓN AL PROYECTO AIXT DE(L) (LOS) MICROCONTROLADOR(ES) XIAO-SAMD21, (SEEEDDUINO XIAO SAMD21) PARA LA TRANSCOMPILACIÓN DE SU CÓDIGO FUENTE DESDE UN LENGUAJE BASADO EN V A LENGUAJE C

ANDERSON ANTONIO MARIN RABA - 20191572002

CARLOS ALBERTO ZAPATA ZAPATA - 20191572029

DIRECTOR DE LA PROPUESTA - FERNANDO MARTÍNEZ SANTA

AGRADECIMIENTOS

Queremos expresar nuestro agradecimiento a las personas que han contribuido a la realización de este proyecto. Sus aportes, apoyo y orientación han sido fundamentales para el éxito de este trabajo.

Nuestro director de Proyecto: Fernando Martínez Santa, Agradezco al director por liderar este proyecto con sabiduría, paciencia y dedicación. Sus orientaciones han sido invaluables y han guiado cada fase.

Nuestros docentes, expresamos gratitud a aquellos profesores que nos enseñaron de su pedagogía a lo largo la carrera, por su conocimiento compartido, mentoría y por desafiarnos constantemente a superar nuestros límites académicos. Su influencia ha sido esencial para nuestro crecimiento intelectual.

A Todos los Que Contribuyeron de Manera Anónima o Inadvertida: Expresamos agradecimientos a todas las personas cuyas contribuciones pueden pasar desapercibidas pero que han sido fundamentales para el éxito de este proyecto.

Índice

I JUSTIFICACIÓN 4

II OBJETIVO GENERAL 5

III OBJETIVOS ESPECÍFICOS 5

IV PLAN DE TRABAJO 5

V DESARROLLO DEL PROYECTO 6

VI RESULTADOS Y PRUEBAS PARA TRANSCOMPILACIÓN 11

Parte I
JUSTIFICACIÓN

La programación de microcontroladores tiene algunos aspectos que pueden dificultar su trabajo, uno puede ser el punto de vista económico o de recursos, y el otro aspecto seria su lenguaje de programación. Con el proyecto Aixt se busca utilizar una sintaxis de lenguaje moderno, enfocándose en dispositivos de bajos recursos.

Este proyecto tiene 3 componentes principales: El lenguaje de programación Aixt basado en la sintaxis del lenguaje V [1], El transcompilador de Aixt a C, que traduce el código fuente de lenguaje Aixt a lenguaje C, para el compilador de C nativo específico de cada microcontrolador y la API (Application Programming Interface) de Aixt, que permite una programación estandarizada para todas las configuraciones de microcontroladores compatibles y funciones de entrada y salida de datos [2]. El lenguaje Aixt puede considerarse un subconjunto del lenguaje V, implementando partes importantes de su sintaxis, pero dejando de lado las propiedades de programación que pueden ser costosas de implementar en los microcontroladores en términos de espacio de memoria.

Establecer un punto medio entre la estructura de programación y lenguaje de los proyectos Arduino (Basado en C) y Microphyton (basado en Python), fue la idea principal del proyecto Sokae [3] que surgió el 2022 con el enfoque de establecer la estructura completa de programación incluyendo el lenguaje. Para este proceso tenemos tres componentes principales: un lenguaje basado en la sintaxis de Rust, un transcopilador que pasa el lenguaje definido como Rust a C, y una API." [4]. A diferencia del proyecto Sokae que usa como base el lenguaje de programación Rust, el proyecto Aixt usa el lenguaje V el cual tiene una sintaxis más sencilla y por está más acorde con el propósito simplificado y esencial de Aixt.

El proyecto consistirá en darle soporte a la serie de microcontroladores SAMD21 (Seeeduino XIAO) para integrarse al proyecto Aixt, esto se realizará implementando una API en C incluyendo las funciones básicas de funcionamiento del microcontrolador soportadas por el proyecto Aixt y un archivo de configuración que permita al transcompilador hacer una traducción efectiva entre los dos lenguajes. El proyecto permitirá realizar código fuente en lenguaje Aixt que sea correctamente compilado a la serie SAMD21 de manera automática, sin tener que realizar cambios manuales que puedan resultar tediosos y costosos en términos de tiempo y recursos. Aixt está optimizado para funcionar eficientemente en dispositivos con recursos limitados como los son los microcontroladores de la serie SAMD21, lo que lo hace adecuado para aplicaciones que requieren una solución eficiente en términos de recursos.

Seeeduino XIAO es una tarjeta de desarrollo basada en el microcontrolador SAMD21 diseñado y construido por los ingenieros de Seeeduino studio. Esta tarjeta se desarrolló bajo la idea de crear una tarjeta

del tamaño de un pulgar “Pequeña pero poderosa”. Este proyecto representa una oportunidad de suplir una importante demanda en el mercado tecnológico, debido a que actualmente existe una gran necesidad de soluciones que permitan la migración de aplicaciones de un lenguaje a otro de manera eficiente y efectiva. También es una ocasión para aplicar conocimientos y habilidades adquiridos durante la formación en Ingeniería Eléctrica y contribuirá con soluciones innovadoras al mercado tecnológico.

Parte II
OBJETIVO GENERAL

Dar soporte al proyecto Aixt para la integración de los microcontroladores de la serie SAMD21.

Parte III
OBJETIVOS ESPECÍFICOS

- Dar soporte a un lenguaje moderno basado en V para ser transcompilado de manera efectiva en lenguaje C para microcontroladores de la serie SAMD21.
- Desarrollar un conjunto de funciones básicas en C para microcontroladores de la serie SAMD21, que pueda ser usado desde un lenguaje basado en V luego de su transcompilado.
- Realizar una documentación que permita a los desarrolladores entender la estructura, funcionalidades y características del código, lo que les permitirá contribuir al proyecto si así lo desean, ya que sea de código abierto.
- Realizar pruebas de funcionamiento de las funciones implementadas en el microcontrolador de la serie SAMD21.

Parte IV
PLAN DE TRABAJO

Se ajusto el el plan para ejecutar mejor el proyecto siguiendo los siguientes ÍTEMS.

	Anderson Marin	Carlos Zapata	Semana
Investigación del microcontrolador y lenguaje V	X	X	1
	X	X	2

Cuadro 2: Cronograma de investigación grupal sobre el microcontrolador y el lenguaje V

API	En esta etapa se debe realizar una recopilación de funciones básicas (entrada y salida de datos, conversión análoga digital comunicación serial entre otras), soportadas en el proyecto Aixt, escritas en C y enfocadas para el microcontrolador SAMD21.
SAMD21.toml	Dado que se necesita un soporte para el transcompilador del lenguaje Aixt a C a la serie de microcontroladores SAMD21, en esta etapa se escribirá un archivo de configuración en formato TOML, que será usado como insumo por el transcompilador
quickref.md	Tanto la arquitectura del hardware de los microcontroladores SAMD21, la interfaz de programación de aplicaciones y la serialización, necesitan tener un soporte escrito que las expliquen, identifiquen y caractericen.

Cuadro 1: Tabla desarrollada para el plan de trabajo.

Parte V

DESARROLLO DEL PROYECTO

Se realizo satisfactoriamente la ejecución del proyecto comenzando el día 19 de septiembre de 2023 y terminando el día 20 de febrero de 2024. En el que se realizaron las pruebas de manera exitosa cumpliendo con los tiempos establecidos en el cronograma original, incluyendo a este ultimo 2 semanas de investigación para el lenguaje V.

PROGRAMACIÓN DE LA INTERFAZ - API

Se realizo exitosamente la API para el microcontrolador «Seeeduino Xiao SAMD21» generando los módulos para la transcompilación del comando, desde el lenguaje V para Arduino mediante archivos «***.c.v***»

Cronograma

Se llevo a cabo una distribución de trabajo 60 %-40 %

Modulo	Archivo	Anderson Marin	Carlos Zapata	Semana
ADC	adc.c.v	X	X	3
	read.c.v	X	X	
	v.mod	X		
PIN	high.c.v	X	X	4
	low.c.v	X	X	
	pin.c.v		X	
	read.c.v	X	X	
	setup.c.v	X	X	
	write.c.v	X	X	
	v.mod	X		
PWM	pwm.c.v	X	X	5
	write.c.v	X		
TIME	sleep.c.v	X	X	
	sleep_ms.c.v	X	X	
	sleep_us.c.v	X	X	
	time.c.v		X	
	v.mod	X		
UART	println.c.v	X	X	6
	println_1.c.v	X		
	read.c.v	X	X	
	read_1.c.v	X		
	ready.c.v	X	X	
	ready_1.c.v	X		
	setup.c.v	X	X	
	setup_1.c.v	X		
	uart.c.v		X	
	v.mod	X		
builtin.c.v		X		3

Cuadro 3: Cronograma para la API

Evidencia

Mediante la plataforma de «GitHub» y programando en «Microsoft Visual Studio Code» donde se retroalimento el repositorio AIXT basado en Lenguaje V, soportando el microcontrolador SAMD21 de Seeed Studio para la API, en donde se pueden destacar los archivos mencionados en el ***Cuadro 3***:

- ADC - Lectura Analógica
- PIN - Para definir los pines del microcontrolador.
- PWM - Salidas Analógicas
- Time - Definición de retardos
- UART - Comunicación Serial

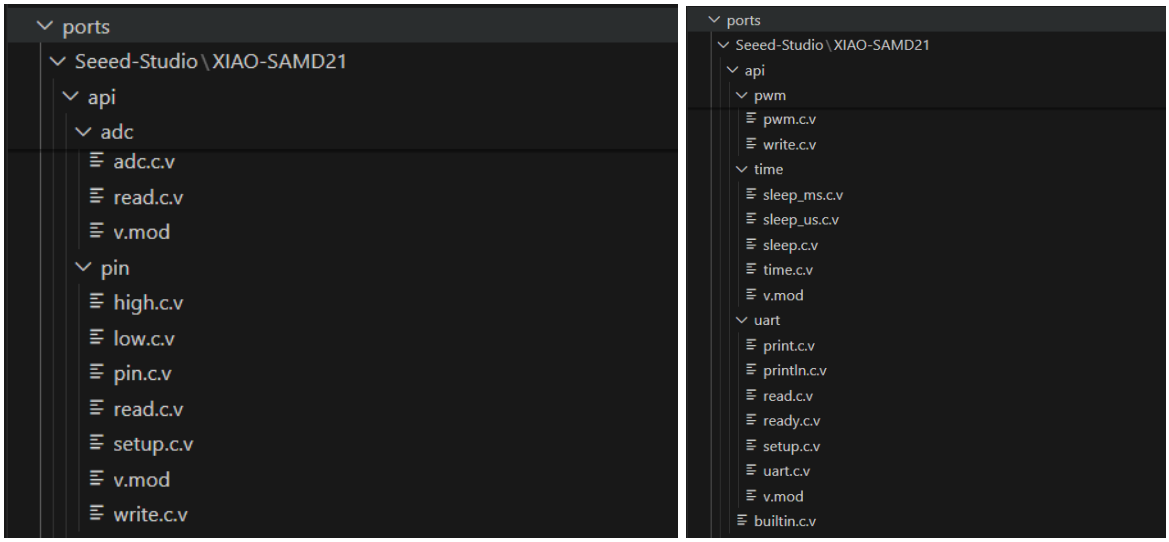


Figura 1: Se puede evidenciar su contenido en el repositorio, al cual se anexa el siguiente link: https://github.com/fermarsan/aixt/tree/main/ports/Seeed-Studio/XIAO-SAMD21/api

ARCHIVO DE CONFIGURACIÓN - SAMD21.toml

Se realizo exitosamente el archivo de transcompilación para el microcontrolador «Seeeduino Xiao SAMD21» identificando y traduciendo las funciones y variables predeterminadas de Arduino IDE para el lenguaje V.

Cronograma

El archivo «.toml» enlaza los archivos de la API para poder ejecutar la transcompilación en donde se realizo una distribución de trabajo: 40 % - 60 % respectivamente.

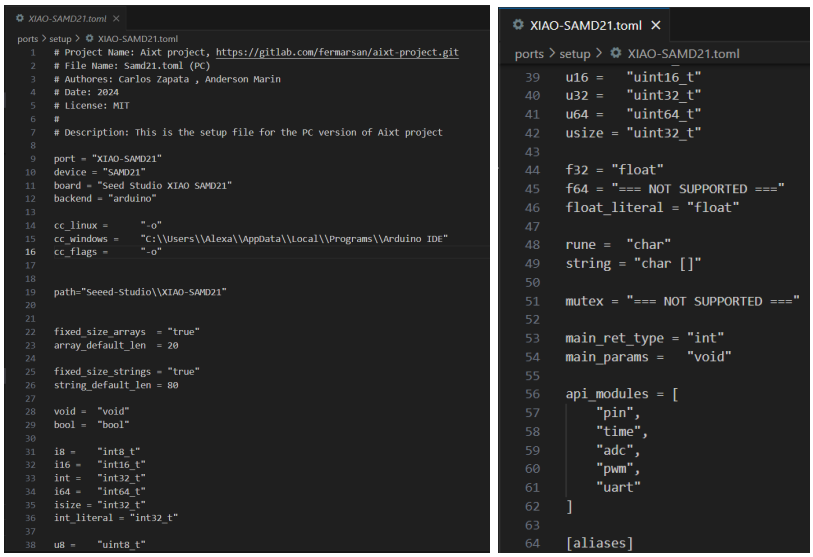
Archivo	Anderson Marin	Carlos Zapata	Semana
XIAO-SAMD21.toml		X	7
		X	8
	X	X	9
	X	X	10
	X	X	11
	X	X	12
	X	X	13
	X	X	14
	X	X	15

Cuadro 4: Cronograma Archivo de Transcompilación

Evidencia

Mediante la plataforma de «GitHub» y programando en «Microsoft Visual Studio Code» donde se retroalimento el repositorio AIXT basado en Lenguaje V, soportando el microcontrolador SAMD21 de Seeed Studio.

Al presente se anexa evidencia fotográfica de lo que se encuentra en el archivo XIAO-SAMD21.toml ubicado en la carpeta Setup, observando la figura 2



```
XIAO-SAMD21.toml
ports > setup > XIAO-SAMD21.toml
# Project Name: Aixt project, https://gitlab.com/fermarsan/aixt-project.git
# File Name: Samd21.toml (PC)
# Authores: Carlos Zapata , Anderson Marin
# Date: 2024
# License: MIT
#
# Description: This is the setup file for the PC version of Aixt project

port = "XIAO-SAMD21"
device = "SAMD21"
board = "Seed Studio XIAO SAMD21"
backend = "arduino"

cc_linux =      "-o"
cc_windows =    "C:\\Users\\Alexa\\AppData\\Local\\Programs\\Arduino IDE"
cc_flags =      "-o"


path="Seeed-Studio\\XIAO-SAMD21"


fixed_size_arrays  = "true"
array_default_len  = 20

fixed_size_strings = "true"
string_default_len = 80

void =  "void"
bool =  "bool"

i8 =    "int8_t"
i16 =   "int16_t"
int =   "int32_t"
i64 =   "int64_t"
isize = "int32_t"
int_literal = "int32_t"

u8 =    "uint8_t"
```

```
XIAO-SAMD21.toml
ports > setup > XIAO-SAMD21.toml
u16 =   "uint16_t"
u32 =   "uint32_t"
u64 =   "uint64_t"
usize = "uint32_t"

f32 = "float"
f64 = "=== NOT SUPPORTED ==="
float_literal = "float"

rune =  "char"
string = "char []"

mutex = "=== NOT SUPPORTED ==="

main_ret_type = "int"
main_params =   "void"

api_modules = [
    "pin",
    "time",
    "adc",
    "pwm",
    "uart"
]

[aliases]
```

Figura 2: Se puede verificar esto en el siguiente link: https://github.com/fermarsan/aixt/tree/main/setup

DOCUMENTACIÓN DE SOFTWARE LIBRE - quickref.md

Se realizo exitosamente la guiá rápida para el microcontrolador «Seeeduino Xiao SAMD21» explicando funciones, anexando su datasheet, identificando los pines y ejemplificando al usar los pines de manera digital o análoga y comunicación serial

Cronograma

El archivo «.md» realiza una pagina tipo Blog, en el que se dividen las especificaciones de uso para el microcontrolador y ejemplos funcionales para el micro en donde se hizo una distribución de trabajo: 50 % - 50 % respectivamente.

Archivo	Anderson Marin	Carlos Zapata	Semana
quickref.md	X	X	16
		X	17
	X		18

Cuadro 5: Cronograma Archivo para la Guía Rápida

Evidencia

Al presente se anexa evidencia fotográfica de lo de la Guía Tapida para el micro controlador «XIAO-SAMD21»; en el cual se puede evidenciar la identificación de pines, funciones y ejemplos para el uso de el aplicativo para el microcontrolador.

quickref.md

docs > Seeed-Studio > XIAO-SAMD21 > quickref.md > # **Guia Rápida Seeeduino Xiao** > ## Identificación de Pines

```md
# **Guia Rápida Seeeduino Xiao**
Esta implementación de Aixt para dae soporte a la tarjeta SAMD21.

## Resumen
* *SAMD21, tiene 14 PIN, que se pueden usar para 11 interfaces digitales, 11 interfaces simuladas, 10 interfaces PWM (d1-d10), 1 pin de salida DAC D0, 1 interfaz de pad SWD, 1 interfaz I2C, 1 interfaz SPI, 1 Interfaz UART, Indicador de comunicación serial (T/R), Luz parpadeante (L) mediante multiplexación de pines. Los colores de los LED (Power,L,RX,TX) son verde, amarillo, azul y azul. Además, Seeed Studio XIAO SAMD21 tiene una interfaz tipo C que puede suministrar energía y descargar código. Hay dos botones de reinicio, puedes conectarl2os brevemente para reiniciar la placa.La definició de pines esta descrita en la tala de* Identificación de Pines.

![Alt text](Picture/Seeeduino_XIAO_SAMD21.jpg)

## Datasheet
[Seeeduino Xiao SAMD21G18A-MU](https://files.seeedstudio.com/wiki/XIAO/Seeed-Studio-XIAO-Series-SOM-Datasheet.pdf)

## Identificación de Pines
Below are the ports used and their proper designations for programming:

Pin No. |Nombre                | Función
--      |-----                 | ---
0       |D0-A0-DAC-QT0         | Analogo; Digital; Convertidor Analogo Digital;           Circuito integrado para botón capacitivo
1       |D1-A1-QT1             | Analogo; Digital;                                        Circuito integrado para botón capacitivo
2       |D2-A2                 | Analogo; Digital;
3       |D3-A3                 | Analogo; Digital;
4       |D4-A4-SDA(I2C)        | Analogo; Digital; Protocolo control de dispositivos(Transmisión de datos)
5       |D5-A5-SCL(I2C)        | Analogo; Digital; Protocolo control de dispositivos(Envio sincronismos de reloj)
6       |D6-A6-TX-QT2          | Analogo; Digital; Comunicación Serial(Transmisor);       Circuito integrado para botón capacitivo
7       |D7-A7-RX-QT3          | Analogo; Digital; Comunicación Serial(Receptor);         Circuito integrado para botón capacitivo
8       |D8-A8-SCK(SPI)-QT4    | Analogo; Digital; Reloj serial;                          Circuito integrado para botón capacitivo
9       |D9-A9-MISO(SPI)-QT5   | Analogo; Digital; Protocolo de comunicación 4 hilos;     Circuito integrado para botón capacitivo
10      |D10-A10-MOSI(SPI)-QT6 | Analogo; Digital; Protocolo de comunicación 4 hilos;     Circuito integrado para botón capacitivo
11      |3.3V                  | Alimentación de Microcontrolador
12      |GND                   | Tierra Común
13      |5V                    | Alimentación de la Placa
```

quickref.md

docs > Seeed-Studio > XIAO-SAMD21 > quickref.md > # **Guia Rápida Seeeduino Xiao** > ## Identificación de Pines

```md
# **Guia Rápida Seeeduino Xiao**
## Programación en Lenguaje V

Las funciones que contiene la API entradas o salidas digitales y para realizar una conversión analogico a digital.

Nombre                    | Descripción
--------------------------|---------------------------
`pin.setup(pin, mode)`    | Configura `pin`como `mode` (input, out)
`pin.out`                 | Parametro `mode` configuración de salida
`pin.in`                  | Parametro `mode` configuración de entrada
`pin.high(pin)`           | Salida Digital en alto `pin`
`pin.low(pin)`            | Salida Digital en bajo `pin`
`pin.write(pin, val)`     | Escribe `val` en `pin`
`pin.read(pin)`           | Lectrura digital `pin`
`adc.read(pin)`           | Lectura analogica `pin` para el `adc`
`pwm.write(pin, val)`     | Salida PWM `pin` y un ciclo util de `val`
`uart.setup(baud_rate)`   | Iniciación Comunicacón Serial a `Baund.rate`
`uart.ready()`            | Obtiene el numero de bytes a leer
`uart.read()`             | Lectura de Comunicacón Serial
`uart.println(message)`       | Imprime  `message` a través Comunicacón Serial
`time.sleep(time)`            | Retardo en `seg`
`time.sleep_us(time)`         | Retardo en `microseg`
`time.sleep_ms(time)`         | Retardo en `miliseg`
```

quickref.md

docs > Seeed-Studio > XIAO-SAMD21 > quickref.md > # **Guia Rápida Seeeduino Xiao** > ## EJEMPLOS > ### Encender un LED con pulsador

```md
# **Guia Rápida Seeeduino Xiao**
## EJEMPLOS
### Parpadeo de un LED
A continuación se comprobara el funcionamiento de la salida digital y el retardo en milisegundos. Para esto se encendera y apagara un led 10 veces, en este ejemplo se importaran del modulo `time` la función `sleep_ms` (retardo en milisegundos), y el modulo `pin` en su totalidad.
```go
import time {sleep_ms}   //impota la fución sleep_ms del modulo time
import pin  // importa el modulo pin en su totalidad

pin.mode(5, pin.out)    // Configura el pin #5 como salida

for i in 0..10{   // 10 veces
    pin.high(5)     // Salida en Alto (Enciende el led)
    sleep_ms(500)   // Retardo de 0.5s
    pin.low(5)      // Salida en Bajo (Apaga el led)
    sleep_ms(500)   // Retardo de 0.5s
}
for{}       // For infinito necesario para la compilación
```
Encender un LED con pulsador
A continuación se comprobara el funcionamiento de entrada y salida digital. Para esto se condicionara el encendido de un led a un pulsador, en este ejemplo se importara modulo `pin` en su totalidad.

```go
import pin  // importa el modulo pin
global lectura = 0    //Crea una variable global para almacenar la lectura digital

pin.setup(3, pin.input)     // Configura el pin #3 como entrada
pin.setup(5, pin.out)       // Configura el pin #5 como salida

for{        // For infinito
    lectura=pin.read(3)     //Almacena la lectura digital del pin #3
    if(lectura==1){         //Condición si el valor de lectura es 1 (Alto)
        pin.high(5);        // Salida en Alto
    }
    pin.low(5)              // Salida en Bajo
}
```
```

quickref.md

docs > Seeed-Studio > XIAO-SAMD21 > quickref.md > # **Guia Rápida Seeeduino Xiao** > ## EJEMPLOS > ### Lectura analoga

```
# **Guia Rápida Seeeduino Xiao**
## EJEMPLOS
### Lectura analoga
A continuación se comprobara el funcionamiento de entrada analogica y salida digital. Para esto se condicionara el encendido de 3 led a un potencimetro, estos led
encenderan dependiendo de la lectura analogica, en este ejemplo se importara modulo `pin` y el modulo `adc`en su totalidad.
```go
import pin  // importa el modulo pin
import adc  // importa el modulo adc

__global val = 0   //Crea una variable global para almacenar la lectura analogica

pin.setup(2, pin.out)   // Configura el pin #2 como salida
pin.setup(3, pin.out)   // Configura el pin #3 como salida
pin.setup(4, pin.out)   // Configura el pin #4 como salida

for {       // For infinito
    val=adc.read(8)     //Almacena la lectura analogica del pin #8
    if val>=1000 {      //Condición de la lectura analogica
        pin.high(2)     // Salida en Alto
        pin.high(3)     // Salida en Alto
        pin.high(4)     // Salida en Alto
    }
    else if val>=750 {
        pin.high(2)     // Salida en Alto
        pin.high(3)     // Salida en Alto
        pin.low(4)      // Salida en Bajo
    }
    else if val>=480 {
        pin.high(2)     // Salida en Alto
        pin.low(3)      // Salida en Bajo
        pin.low(4)      // Salida en Bajo
    }
    else {
        pin.low(2)      // Salida en Bajo
        pin.low(3)      // Salida en Bajo
        pin.low(4)      // Salida en Bajo
    }
    }
```

quickref.md

docs > Seeed-Studio > XIAO-SAMD21 > quickref.md > # **Guia Rápida Seeeduino Xiao** > ## EJEMPLOS > ### Lectura analoga

```
# **Guia Rápida Seeeduino Xiao**
## EJEMPLOS

### Salida PWM
A continuación se comprobara el funcionamiento de la salida analogica (PWM). Para esto se variara la intensidad luminica de un led en un periodo de tiempo, en este
ejemplo se importara modulo `pin` y el modulo `pwm`en su totalidad y la la función `sleep_ms` del modulo`time`.
```go

import time {sleep_ms}  // importa el función sleep_ms
import pin              // importa el modulo pin
import pwm              // importa el modulo pwm

__global val = 0        //Crea una variable global para almacenar un valor el cual corresponde a la intensidad luminica

pin.setup(5, pin.out)   // Configura el pin #5 como salida



for {
    pwm.write(5, val)   // Salida pwm a un ciclo util de val
    sleep_ms(250)       // Retardo de 250ms
    val=val+10          // Suma 10 a val
    if val==250{        // Condicion si val=250
        val=0
    }
}
```

Figura 3: Archivo de programación

Parte VI
RESULTADOS Y PRUEBAS PARA TRANSCOMPILACIÓN

Se requirió de 2 semanas para la verificación final de los documentos y el funcionamiento del microcontrolador con sus ejemplos funcionando satisfactoriamente.

Archivo	Anderson Marin	Carlos Zapata	Semana
Parpadeo	X	X	19
PWM	X	X	19
ADC	X	X	20

Cuadro 6: Tiempo para realizar pruebas

Prueba de Parpadeo.v

Para este archivo se programa en lenguaje v lo que queremos ejecutar en el micro, en este caso queremos que se den una serie de parpadeos por medio de 2 LED ubicados en la Board de Conexiones, logrando que se genere el archivo .ino que sera leído por Arduino IDE.

Para lograr esto es necesario transcompilar el archivo .v generando el archivo .ino

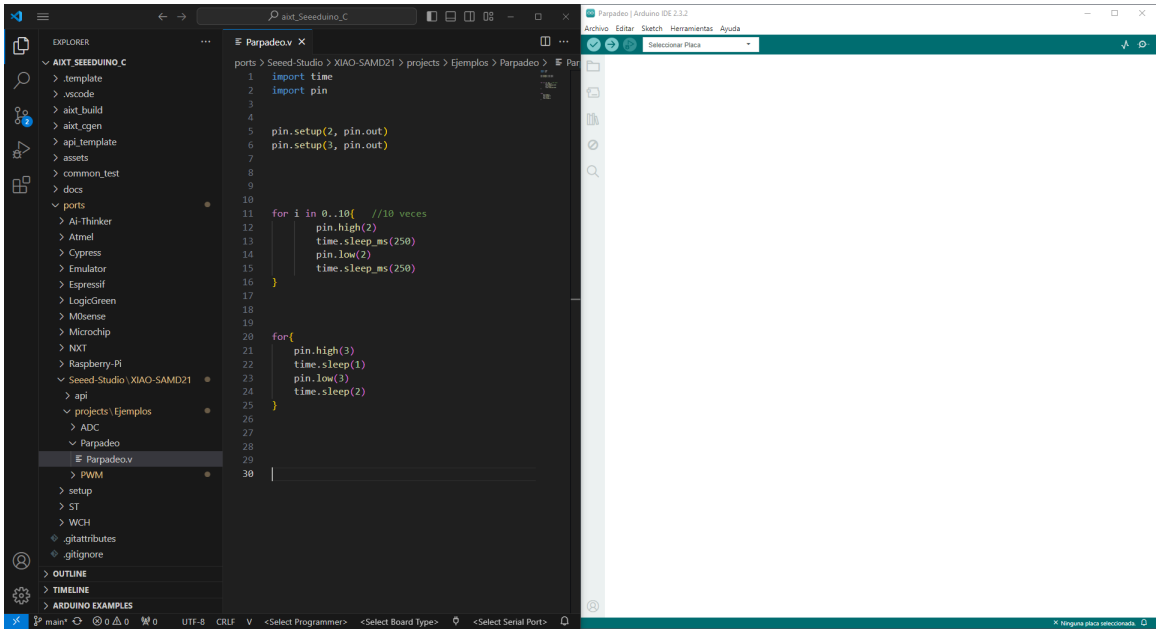


Figura 4: Ejemplo Parpadeo antes de Transcompilar

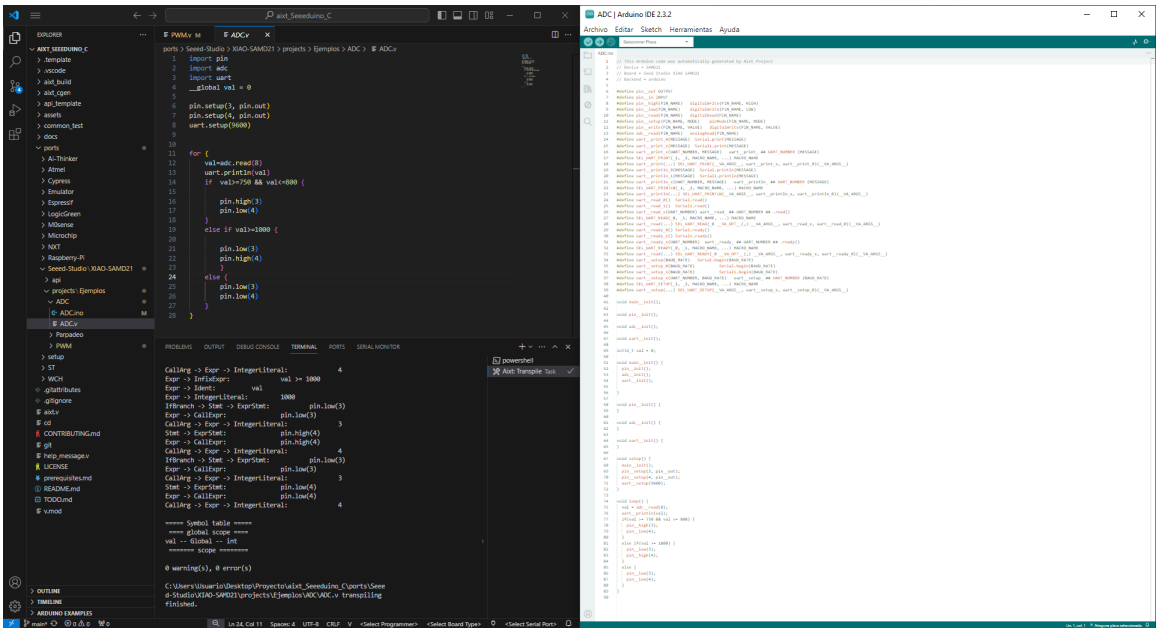


Figura 5: Ejemplo Parpadeo después de Transcompilar

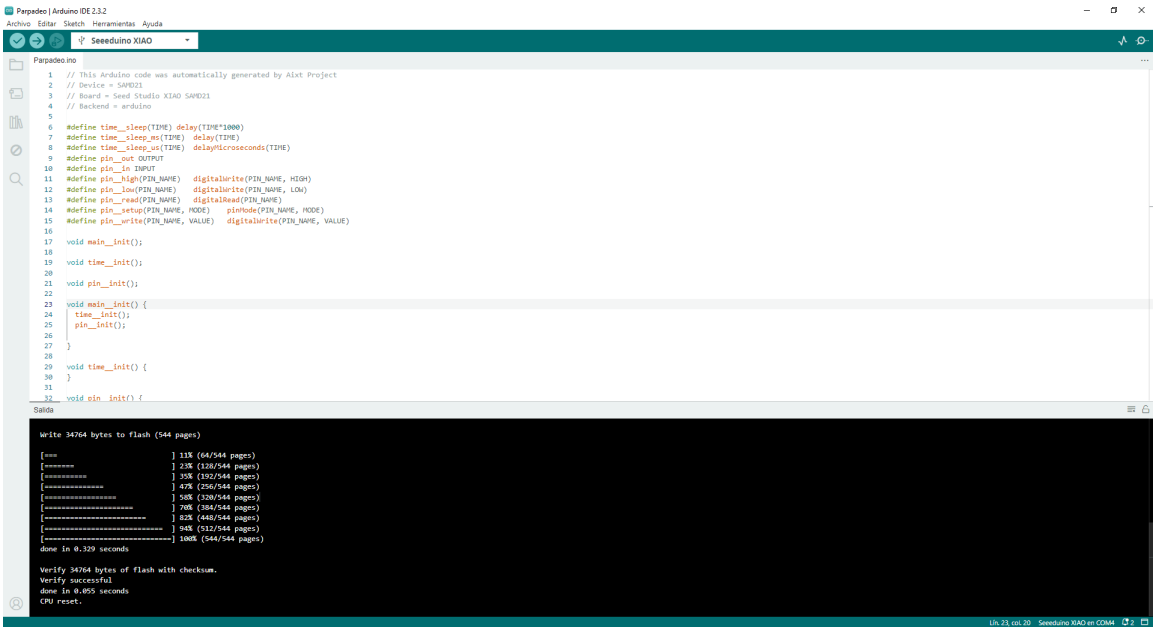


Figura 6: Verificación ejemplo parpadeo de la transcompilación en Arduino IDE (EXITOSA)

Prueba ADC

Para este archivo se programa en lenguaje v, en este caso queremos que se enciendan LED ubicados en la Board de Conexiones; en donde la señal varia de acuerdo al voltaje graduado por un potenciómetro como

señal análoga, variando en la activación sobre los LED, ejecutando el archivo .v genera el archivo .ino que sera leído por Arduino IDE.

Para lograr esto es necesario compilar el archivo .v generando el archivo .ino

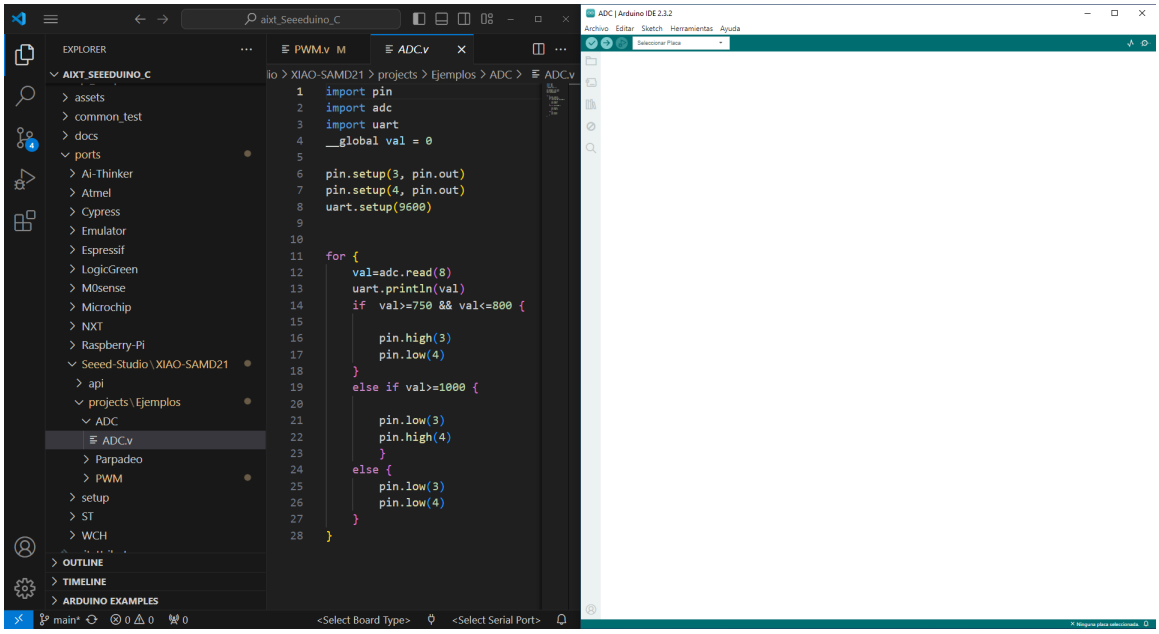


Figura 7: Ejemplo ADC antes de Transcompilar

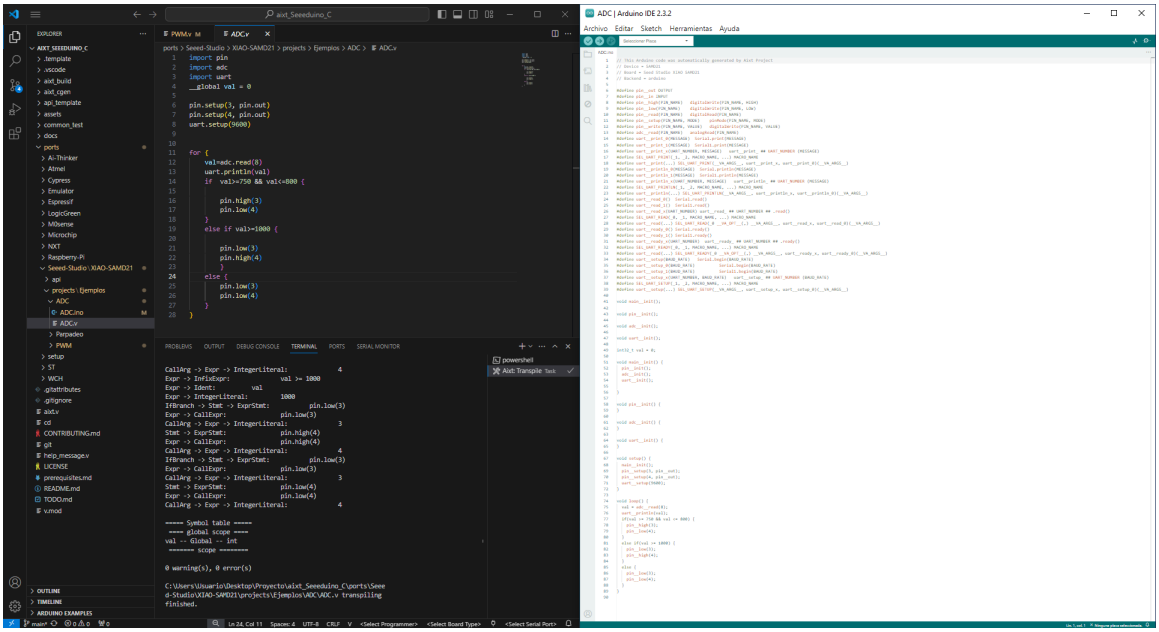


Figura 8: Ejemplo ADC después de Transcompilar

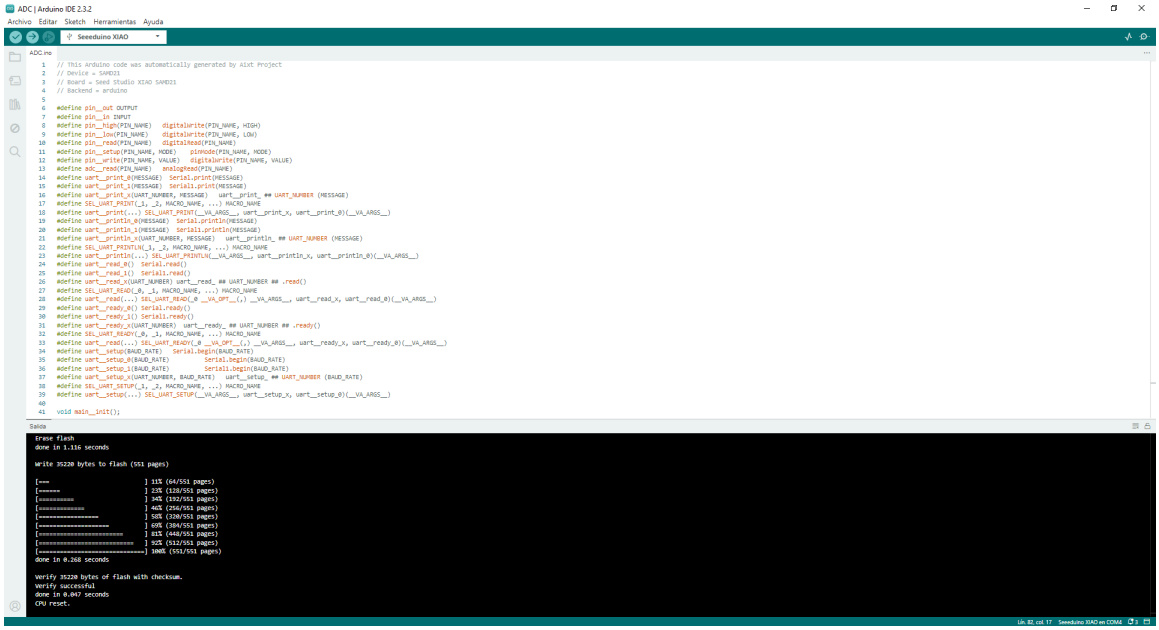


Figura 9: Verificación ejemplo ADC de la transcompilación en Arduino IDE (EXITOSA)

Prueba PWM

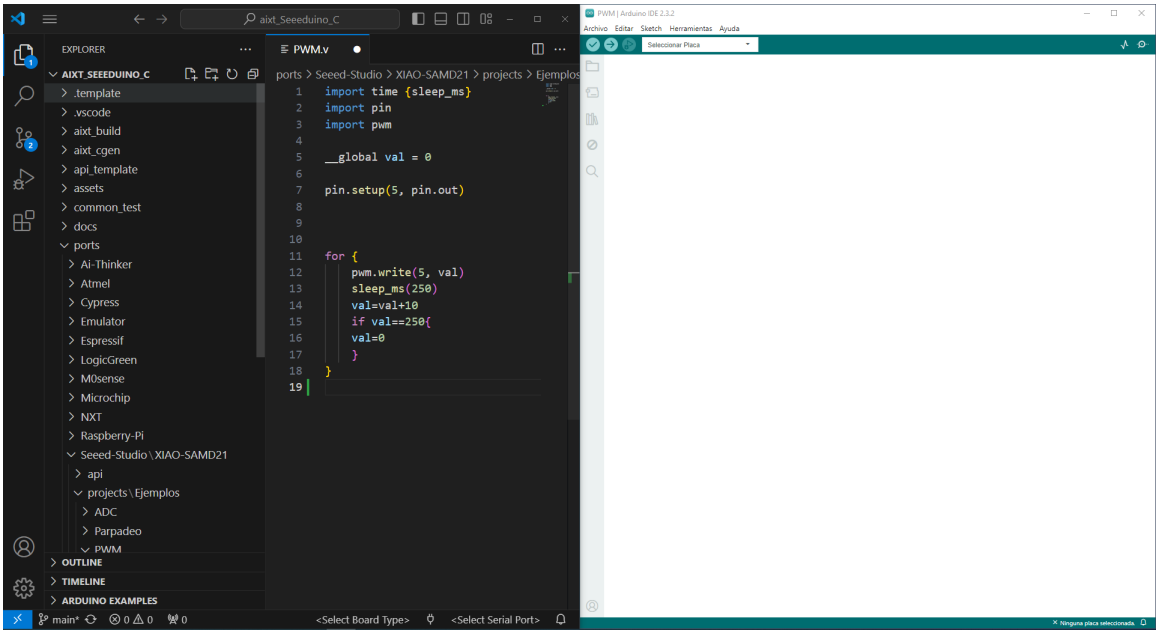


Figura 10: Ejemplo PWM antes de Transcompilar

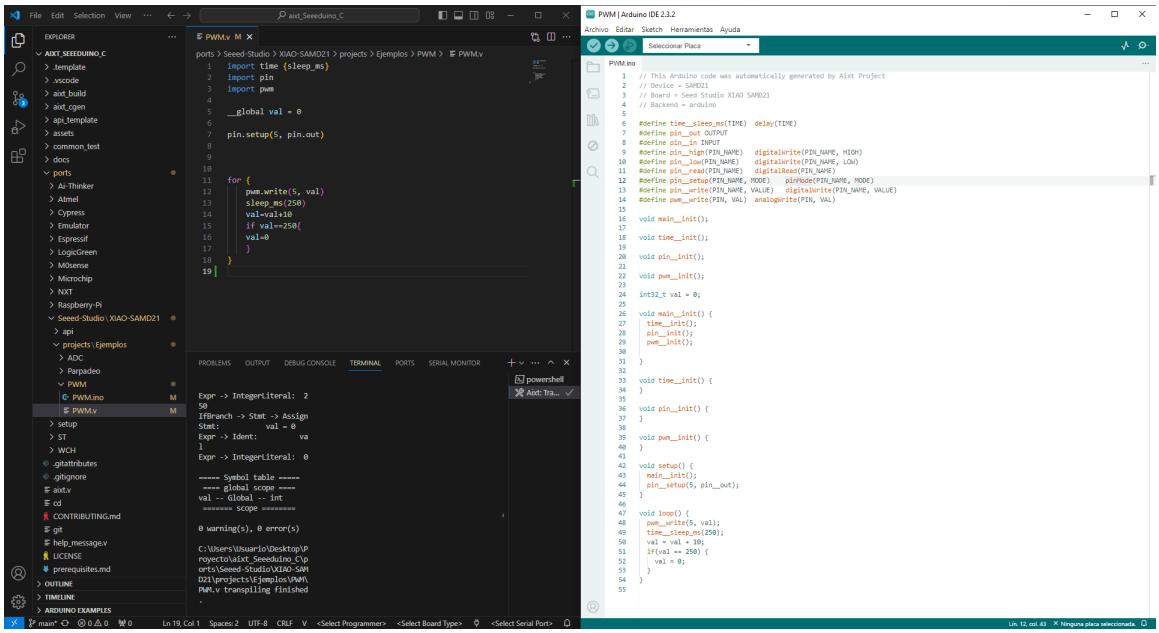


Figura 11: Ejemplo PWM después de Transcompilar

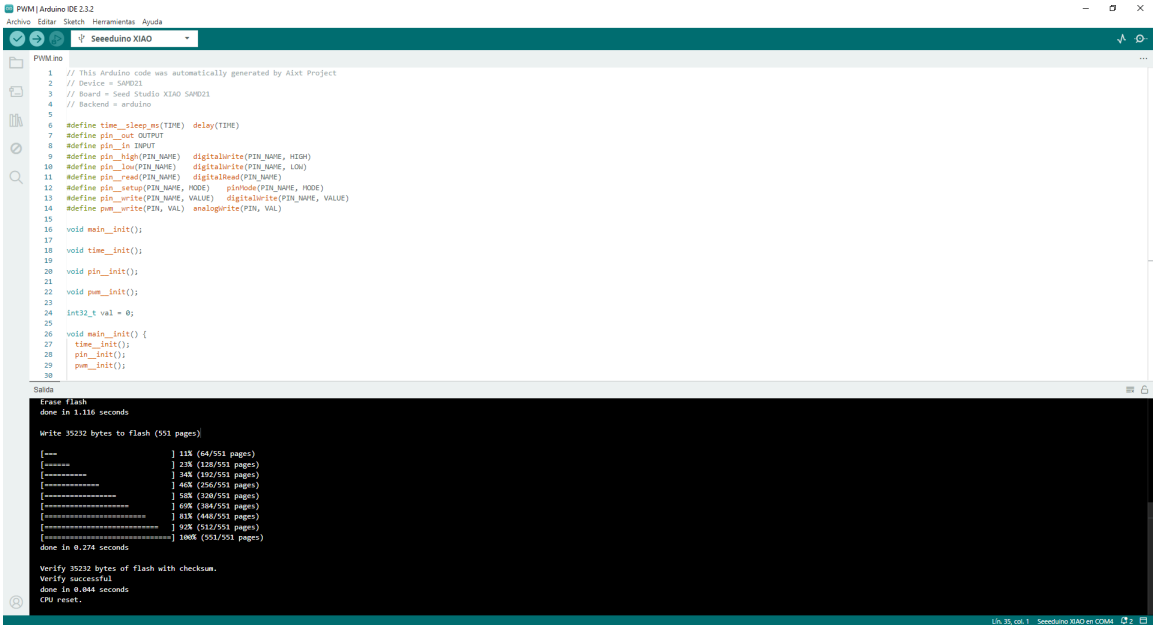


Figura 12: Verificación ejemplo PWM de la transcompilación en Arduino IDE (EXITOSA)

Referencias

1 de mayo de 2024

[1] Mart, Fernando, Santiago Orjuela Rivera, and Fredy H. Mart. "Microcontrollers Programming Framework based on a V-like Programming Language."

International Journal of Advanced Computer Science and Applications, vol. 13, nº 12, 2022.

[2] F. Martínez Santa, «Aixt,» 2022. [En línea]. Available: https://gitlab.com/fermarsan/aixt-project. [Último acceso: 27 Febrero 2024].

[3] F. Martínez Santa, «Sokae Project,» Noviembre 2022. [En línea]. Available: https://gitlab.com/fermarsan/sokae-project. [Último acceso: 27-02-2024].

[4] Santa, Fernando Martinez, Santiago Orjuela Rivera, and Fredy H. Martinez Sarmiento. "Rust-like Programming Language for Low-resource Microcontrollers." Advances in Dynamical Systems and Applications vol. 17 nº 2 pp. 599-614, 2022.

[5] "Getting Started with Seeed Studio XIAO SAMD21 | Seeed Studio Wiki". Seeed Studio Wiki. Accedido el 4 de marzo de 2024. [En línea]. Disponible: https://wiki.seeedstudio.com/Seeeduino-XIAO/

[6] Seeed Studio. Datasheet. https://files.seeedstudio.com/wiki/XIAO/Seeed-Studio-XIAO-Series-SOM-Datasheet.pdf. Consultado el 4 de marzo de 2024.