DESARROLLO DEL MODULO DE VISUALIZACIÓN OLED POR I2C PARA LA PLATAFORMA AIXT USANDO LOS MICROCONTROLADORES RASPBERRY PI PICO Y XIAO-SAMD21

AUTORES:

Leidy Natalia González Hernández – lngonzalezh@udistrital.edu.co – 20171572051

Daniel Esteban Leal Cortes – delealc@udistrital.edu.co – 20211572004

1. INTRODUCCIÓN

Los microcontroladores son circuitos integrados que en su interior contienen una unidad central de procesamiento (CPU), unidad de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estos componentes están interconectados dentro del microcontrolador, y en conjunto se denomina microcomputadora. El propósito fundamental de los microcontroladores es leer y ejecutar los programas proporcionados por el usuario. Su naturaleza programable simplifica el diseño de circuitos electrónicos, ofreciendo versatilidad y flexibilidad. Permitiendo así que un mismo circuito sea capaz de desempeñar diversas funciones con solo ajustar el programa del microcontrolador.

Un módulo OLED (Organic Led Emitting Diode) es un dispositivo que puede mostrar texto, gráficos e imágenes en una pantalla delgada y flexible utilizando materiales orgánicos que emiten luz cuando se aplica corriente eléctrica. Son un tipo de pantalla relativamente nuevo que ha ganado popularidad en una amplia gama de dispositivos electrónicos, como televisores, teléfonos inteligentes, tabletas inteligentes, entre otros. Este tipo de pantallas son conocidas por su alta calidad de imagen, flexibilidad y menor consumo de energía.

La pasantía tiene como objetivo principal el desarrollo de un código en *lenguaje V*, con el fin de implementar un módulo OLED en la plataforma *Aixt*. Esta implementación permitirá ampliar las funcionalidades con las que cuenta actualmente la API de los microcontroladores *RASPBERRY PI PICO* y *XIAO-SAMD21*, ofreciendo soporte para este tipo de pantallas.

El soporte para módulos OLED no solo facilitará el desarrollo de proyectos que requieran pantallas, sino que también brindará nuevas opciones para los desarrolladores, permitiéndoles crear soluciones mas interactivas y visuales con mayor eficiencia directamente desde la plataforma *Aixt*. Esto ampliara significativamente las capacidades de la plataforma, proporcionando más herramientas para proyectos futuros.

2. OBJETIVOS

Objetivo general.

Dar soporte al proyecto *Aixt* al módulo de visualización OLED por I2C para el backend de Arduino sobre los microcontroladores *RASPBERRY PI PICO* y *XIAO-SAMD21*.

Objetivos específicos.

- Ampliar archivos de configuración (que permiten la transcompilación de V/Aixt a C) de los microcontroladores *RASPBERRY PI PICO* y *XIAO-SAMD21,* para incluir el módulo de visualización OLED por I2C.
- Desarrollar un módulo de visualización OLED por I2C como protocolo de comunicación, con Arduino CLI para los microcontroladores *RASPBERRY PI PICO* y *XIAO-SAMD21*, que puedan ser usadas o invocadas desde el lenguaje *Aixt*.
- Hacer pruebas de funcionamiento de las funciones implementadas sobre los microcontroladores.
- Realizar una documentación adecuada que permita a los usuarios entender el funcionamiento del proyecto *Aixt* enfocado a los microcontroladores *RASPBERRY PI PICO* y *XIAO-SAMD21* con el módulo de visualización OLED por I2C, que a su vez permitirá contribuir al proyecto si así lo desean.

3. CUMPLIMIENTO AL CRONOGRAMA POR MES (INDIVIDUAL).

Pasante 1: Leidy Natalia González Hernández

Etapa 1.

Se inicio con la respectiva indagación en el correspondiente datasheet del microcontrolador *XIAO-SAMD21* para comprender cada una de las características que tiene este microcontrolador. Posteriormente se revisan detenidamente los módulos de las funciones que ya se tienen incorporados en el proyecto *Aixt* y se estableció la debida comunicación del microcontrolador con el computador para analizar con mayor detalle el funcionamiento de los módulos correspondientes. Tras realizar dichas pruebas, se realiza una búsqueda de un código base que nos sirva de origen para poder programar e incorporar el módulo de visualización. Las primeras pruebas correspondientes al módulo de visualización se realizaron en la plataforma *Arduino IDE* teniendo en cuenta la adecuada configuración para este microcontrolador. En esta etapa se recopiló información clave para el funcionamiento adecuado del módulo de visualización OLED con el microcontrolador *XIAO-SAMD21.*

Etapa 2.

Esta etapa se divide en dos partes. En la primera, se continúan realizando pruebas en el lenguaje nativo del microcontrolador *XIAO-SAMD21*, con el fin de comprender mejor la lógica de funcionamiento del módulo de visualización OLED, verificar la capacidad que tiene para mostrar diferentes representaciones visuales y ofrecer un soporte adecuado para la transcompilación de lenguajes en la plataforma.

En la segunda parte de esta etapa, se revisan y verifican los archivos de configuración en formato *.JSON* correspondientes al microcontrolador en uso, los cuales están incorporados en la plataforma *Aixt.* Además, se continúa con el desarrollo del módulo de soporte para el funcionamiento del módulo de visualización.

Etapa 3.

Teniendo ya el modulo de soporte de funcionamiento, se procede a programar los códigos en *lenguaje V*, con el fin de ampliar los ejemplos disponibles en la API de este microcontrolador y evaluar correctamente el desempeño de las funciones a las que se les está dando soporte, al momento de transcompilar los códigos, asegurando que el sistema responda de manera adecuada ante diferentes tipos de datos y configuraciones. Además, se comprueba que, con los distintos códigos, no se genere ningún error durante el proceso de transcompilación entre los lenguajes.

Etapa 4.

En esta última etapa, se cargan las modificaciones realizadas en la API del microcontrolador *XIAO-SAMD21* en el repositorio principal del proyecto y se añade información adicional a la guía rápida que se encuentra en la plataforma *Aixt*.

Durante todas las etapas del proyecto, fue importante registrar la información esencial para disponer de una documentación adecuada de respaldo. Al llegar a la etapa final, se consolidaron los documentos creados en cada mes con el propósito de elaborar un informe completo que detallara con precisión el proceso de integración del módulo OLED para la plataforma *Aixt* con el microcontrolador *XIAO-SAMD21*.

Pasante 2: Daniel Esteban Leal Cortes.

Etapa 1.

Se realizó el primer acercamiento con el microcontrolador *RASPBERRY PI PICO*, consultando tanto el datasheet proporcionado por el fabricante, como la documentación que ya se tiene sobre este microcontrolador. Esto nos ayudó a comprender ciertas características y funcionalidades que pueda tener este microcontrolador, facilitando la revisión y análisis de

los módulos de las funciones ya integrados en la plataforma *Aixt*. De esta forma, se pudo comprender un poco más el funcionamiento de cada módulo y como transcompila la información de un lenguaje a otro. Luego se buscó información relacionada con el módulo de visualización OLED por I2C para poder desarrollar y ejecutar un código base que nos ayudará a seguir brindando un soporte adecuado para la funcionalidad de visualización con este microprocesador.

Etapa 2.

Lo primero que se realiza en esta etapa es continuar desarrollando diversos códigos en el lenguaje nativo del microcontrolador *RASPBERRY PI PICO,* con el objetivo de identificar el funcionamiento adecuado del módulo de visualización, comprobar su capacidad para mostrar texto, imágenes y otros elementos visuales, y ofrecer un soporte óptimo para la plataforma *Aixt.*

Lo siguiente fue revisar los archivos *.JSON* existentes en la plataforma, correspondientes a este microcontrolador. Luego, se desarrolló el módulo necesario para dar soporte al módulo de visualización OLED con este microcontrolador en la plataforma *Aixt.*

Etapa 3.

Una vez desarrollado el modulo de soporte de funcionalidad para incorporar el modulo OLED, procedemos a adaptar los ejemplos existentes en el lenguaje nativo del microcontrolador, en *lenguaje V,* con el fin de ampliar la API correspondiente a este microcontrolador y evaluar adecuadamente el desempeño del sistema, identificando posibles limitaciones que se puedan presentar en su implementación en proyectos futuros.

Etapa 4.

Lo primero que se hizo en esta etapa fue incorporar las modificaciones realizadas al repositorio principal de la plataforma *Aixt.*

A lo largo de todas las fases del proyecto, fue fundamental documentar la información clave para contar con un respaldo adecuado. En la etapa final, se consolidaron los documentos elaborados durante cada mes con el propósito de generar un informe detallado que describiera el proceso de integración del módulo OLED a la plataforma *Aixt*, utilizando el microcontrolador *RASPBERRY PI PICO.*

4. DESCRIPCIÓN Y EVIDENCIA DE LOS RESULTADOS.

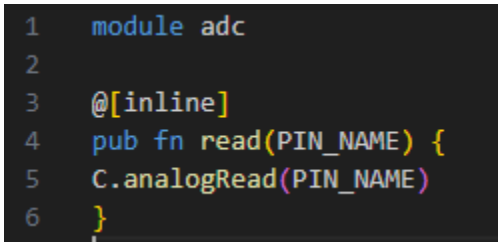
Se incluye evidencia detallada de las actividades realizadas durante el desarrollo de la pasantía, especialmente en relación con los módulos y códigos implementados. Esta evidencia no solo respalda los resultados obtenidos, sino que también facilita la comprensión

del proceso de implementación, destacando las modificaciones realizadas a lo largo del proyecto. De este modo, se ofrece una visión mas profunda de las etapas y de como se alcanzaron los objetivos.

1. Primero se presentará evidencia de los módulos de funciones y los archivos de configuración *.JSON* necesarios para el correcto funcionamiento de los microcontroladores *XIAO-SAMD21* y *RASPBERRY PI-PICO,* los cuales ya se encuentran incorporados y configurados en la plataforma *Aixt.*

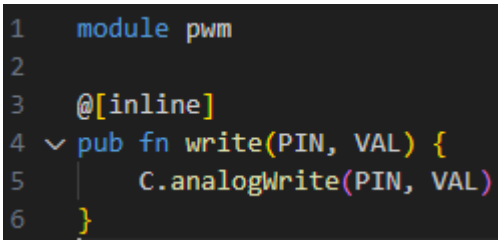
XIAO-SAMD21

1. En este segmento de código, se encuentra el módulo ADC junto con su función equivalente que es necesaria para habilitar la conversión analógica a digital en los pines del microcontrolador.



Módulo ADC

2. En este módulo, se define el modulo PWM junto con su función asociada, garantizando así una correcta transcompilación.



Módulo PWM

3. En estos fragmentos de códigos, se muestran la correcta definición de los módulos UART como una macro, nos ayudan a la comunicación serie entre dispositivos.

```
module uart


@[inline]
pub fn any_0() {
    C.Serial.available()
}

@[inline]
pub fn any_1() {
    C.Serial1.available()
}

@[inline]
pub fn any_x(UART_NUMBER) {
    C.uart.any_ ## UART_NUMBER ## .available()
}

// macro with 0 or 1 arguments
#define SEL_UART_ANY(_0, _1, MACRO_NAME, ...) MACRO_NAME
@[inline]
pub fn any(...) SEL_UART_ANY(_0 __VA_OPT__)(,) {
    C.__VA_ARGS__, uart.any_x, uart.any_0(__VA_ARGS__)
}
```

```
module uart

@[inline]
pub fn read_0() {
    C.Serial.read()
}

@[inline]
pub fn read_1() {
    C.Serial1.read()
}

@[inline]
pub fn read_x(UART_NUMBER) {
    C.uart.read_ ## UART_NUMBER ## .read()
}

// macro with 0 or 1 arguments
#define SEL_uart.read(_0, _1, MACRO_NAME, ...) MACRO_NAME
@[inline]
pub fn read(...) SEL_uart.read(_0 __VA_OPT__)(,) {
    C.__VA_ARGS__, uart.read_x, uart.read_0(__VA_ARGS__)
}
```

```
module uart

@[inline]
pub fn print_0(MESSAGE) {
    C.Serial.print(MESSAGE)
}

@[inline]
pub fn print_1(MESSAGE) {
    C.Serial1.print(MESSAGE)
}

@[inline]
pub fn print_x(UART_NUMBER, MESSAGE)        uart.print_ ## UART_NUMBER (MESSAGE) {
    C.
}

// macro with 1 or 2 arguments
#define SEL_uart.print(_1, _2, MACRO_NAME, ...) MACRO_NAME
@[inline]
pub fn print(...) {
    C.SEL_uart.print(__VA_ARGS__, uart.print_x, uart.print_0)(__VA_ARGS__)
}
```

```
module uart

@[inline]
pub fn println_0(MESSAGE) {
    C.Serial.println(MESSAGE)
}

@[inline]
pub fn println_1(MESSAGE) {
    C.Serial1.println(MESSAGE)
}

@[inline]
pub fn println_x(UART_NUMBER, MESSAGE)      uart.println_ ## UART_NUMBER (MESSAGE) {
    C.
}

// macro with 1 or 2 arguments
#define SEL_uart.println(_1, _2, MACRO_NAME, ...) MACRO_NAME
@[inline]
pub fn println(...) {
    C.SEL_uart.println(__VA_ARGS__, uart.println_x, uart.println_0)(__VA_ARGS__)
}
```

```
module uart

@[inline]
pub fn setup(BAUD_RATE) {
    C.Serial.begin(BAUD_RATE)
}


@[inline]
pub fn setup_0(BAUD_RATE) {
    C.Serial.begin(BAUD_RATE)
}
@[inline]
pub fn setup_1(BAUD_RATE) {
    C.Serial1.begin(BAUD_RATE)
}
@[inline]
pub fn setup_x(UART_NUMBER, BAUD_RATE)      uart.setup_ ## UART_NUMBER (BAUD_RATE) {
    C.
}

// macro with 1 or 2 arguments
#define SEL_uart.setup(_1, _2, MACRO_NAME, ...) MACRO_NAME
@[inline]
pub fn setup(...) {
    C.SEL_uart.setup(__VA_ARGS__, uart.setup_x, uart.setup_0)(__VA_ARGS__)
}
```

Módulo UART

4. A continuacion se observa el archivo de configuración, en el cual se especifican los tipos y tamaños de datos para el microcontorlador.



```
{
"port": "XIAO-SAMD21",
"device": "SAMD21",
"board": "Seeed Studio XIAO SAMD21",
"backend": "arduino",
"cc_linux":      "Arduino CLI",
"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",
"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"
}
```

Archivo de configuración .JSON

RASPBERRY-PI-PICO

1. A continuación se muestra el módulo ADC, correspondiente a este microcontrolador.

```
module adc

// setup function sets the ADC resolution
@[inline]
pub fn setup(res u8) {
    C.analogReadResolution(res)
}
```

Módulo ADC

2. Se muestra la definición del módulo PWM, con su respectiva función asociada en el lenguaje nativo.

```
module pwm

// setup function configures de PWM hardware
@[inline]
pub fn setup(freq u32, range u32, res u8) {
    C.analogWriteFreq(freq)
    C.analogWriteRange(range)
    C.analogWriteResolution(res)
}
```

Módulo PWM

3. Se muestran los módulos correspondientes a la comunicación serie UART.

```
module uart

// any function gets the number of bytes (characters) available for reading
@[inline]
pub fn any() int {
    return C.Serial1.available()
}
```

```
module uart

#include "pins.c"

// pins function sets the tx and tx pins for the uart
@[inline]
pub fn pins(tx_pin int, rx_pin int) {
    C.uart__pins(tx_pin, rx_pin)
}
```

```
module uart

// print function prints data to the serial port as human-readable ASCII text
@[inline]
pub fn print(message string) {
    C.Serial1.print(message)
}
```

```
module uart

@[inline]
pub fn println(message string) {
    C.Serial1.println(message)
}
```

```
module uart

// read function reads incoming serial data
@[inline]
pub fn read() u8 {
    return C.Serial1.read()
}
```

```
module uart

// setup function sets the data rate in bits
// per second (baud) for serial data transmission
@[inline]
pub fn setup(baud_rate int) {
    C.Serial1.begin(baud_rate)
}
```

```
module uart

// write function sends a single byte to the serial port
@[inline]
pub fn write(character u8) {
    C.Serial1.write(character)
}
```

Módulos complementarios UART

4. A continuación, se muestra el código de configuración correspondiente al microcontrolador *RASPBERRY-PI-PICO.*

```
{
    "port":    "Raspberry-Pi-Pico",
    "board":   "Raspberry Pi Pico",
    "backend": "arduino",

    "cc_linux":   "arduino-cli",
    "cc_windows": "arduino-cli.exe",
    "cc_make_flags":   "compile -b arduino:mbed_rp2040:pico -v --output-dir @{file_dir_name}/build",

    "api_paths": [
        "Raspberry-Pi/Raspberry-Pi-Pico",
        "Arduino/arduino-generic"
    ],
    "v_defines": [
    ],

    "inline_as_c_macro": true,
    "string_default_len": 50,

    "compiler_setup_path": "setup/arduino32.json"
}
```

Archivo de configuración. JSON

2. A continuación, se muestra el código base para comenzar a trabajar con los microcontroladores XIAO-SAMD21 y RASPBERRY PI-PICO, utilizando el sistema de conexión I2C para controlar el display OLED.

Seeeduino XIAO

```
#include <Wire.h>  //Incluye libreria para comunicación I2C
#include <Adafruit_GFX.h>  //Incluye libreria de interfaz grafica
#include <Adafruit_SSD1306.h>  //Incluye libreria para display OLED SSD1306

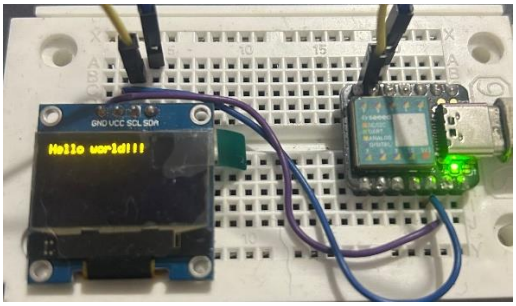
#define SCREEN_WIDTH 128  //Define ancho del display, en pixeles
#define SCREEN_HEIGHT 64  //Define altura del display, en pixeles
#define OLED_RESET    -1  //Define el pin de reset

// Declaracion del display SSD1306 connectado a I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

void setup() {
  Serial.begin(115200); //Inicializa la comunicación serial

  //Inicializa el display, indica alimentacion del display y la direccion I2C para 128x64
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    //Mensaje de error, por si falla la inicialización
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000); //Pausa de 2s
  display.clearDisplay();  //Limpia el display
  display.setTextSize(1);  //Establece el tamaño del texto
  display.setTextColor(WHITE);  //Establece el color del texto
  display.setCursor(0, 0);  //Establece la posición en la pantalla
  display.println("Hello world!!!");  //Imprime el texto
  display.display();   //Actualiza el display para que se muestren los cambios
}
```

Código base para XIAO-SAMD21



Implementación del código en el sistema XIAO-SAMD21 – OLED

Raspberry Pi Pico

```
#include <Wire.h>  //Incluye libreria para comunicación I2C
#include <Adafruit_GFX.h>  //Incluye libreria de interfaz grafica
#include <Adafruit_SSD1306.h>  //Incluye libreria para display OLED SSD1306

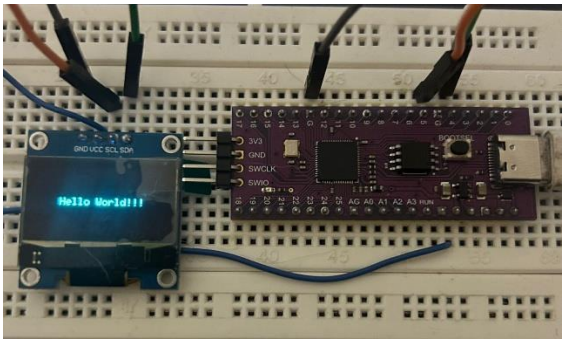
#define SCREEN_WIDTH 128 //Define ancho del display, en pixeles
#define SCREEN_HEIGHT 64 //Define altura del display, en pixeles

// Declaracion del display SSD1306 connectado a I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  Serial.begin(115200);  //Inicializa la comunicación serial

  //Inicializa el display, indica alimentacion del display y la direccion I2C para 128x64
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    //Mensaje de error, por si falla la inicialización
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);  //Pausa de 2s
  display.clearDisplay();  //Limpia el display
  display.setTextSize(1);  //Establece el tamaño del texto
  display.setTextColor(WHITE);  //Establece el color del texto
  display.setCursor(24, 28);  //Establece la posición en la pantalla
  display.println("Hola Natalia!!!");  //Imprime el texto
  display.display();  //Actualiza el display para que se muestren los cambios
}
```

Código base para RASPBERRY PI-PICO



Implementación del código en el sistema RASPBERRY PI-PICO – OLED

3. Implementamos códigos adicionales en los microcontroladores con el objetivo de comprender a fondo el alcance y las capacidades del display OLED. Esto nos permitió evaluar cómo el display maneja y muestra diferentes tipos de datos, como texto, imágenes y gráficos, y analizar su desempeño en diversas condiciones de operación. El estudio y la implementación de estos códigos nos permiten brindar un soporte más óptimo a la plataforma *Aixt.*

   Este código muestra el logo relacionado con la plataforma *Aixt.* Para lograr esto, la imagen debe ser convertida a una matriz de bytes, lo cual se realiza mediante una

plataforma web. Sin embargo, es necesario ajustar el tamaño de la imagen a 128x64, ya que ese es el tamaño del display que se está utilizando.



Imagen que se empleará en el código.

Seeeduino XIAO

```
#include <Wire.h>  //Incluye libreria para comunicación I2C
#include <Adafruit_GFX.h>  //Incluye libreria de interfaz grafica
#include <Adafruit_SSD1306.h>  //Incluye libreria para display OLED SSD1306

#define SCREEN_WIDTH 128  //Define ancho del display, en pixeles
#define SCREEN_HEIGHT 64  //Define altura del display, en pixeles
#define OLED_RESET  -1  //Define el pin de reset

// Declaracion del display SSD1306 connectado a I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

//Declaración de una variable estatica de 1024 bytes que contiene la imagen
static const uint8_t image_data_Ruiarray[1024] = {
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xe0, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0x80, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x01, 0xff, 0xc0,
  0xff, 0xff, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x01, 0xff, 0xc0,
  0xff, 0xff, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0xff, 0xc0,
  0xff, 0xff, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xff, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfe, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfe, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfe, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfc, 0x00, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfc, 0x00, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfc, 0x00, 0x00, 0x03, 0xff, 0xfe, 0x03, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x7f, 0xc0,
  0xff, 0xf8, 0x00, 0x00, 0x03, 0xff, 0xf8, 0x01, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf8, 0x00, 0x00, 0x03, 0xff, 0xf8, 0x00, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf8, 0x00, 0x00, 0x03, 0xff, 0xf0, 0x00, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf8, 0x00, 0x00, 0x01, 0xff, 0xf0, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf0, 0x00, 0x00, 0x01, 0xff, 0xf0, 0x00, 0x3f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf0, 0x00, 0x00, 0x01, 0xff, 0xf0, 0x00, 0x3f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf0, 0x00, 0x00, 0x01, 0xff, 0xf0, 0x00, 0x3f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xe0, 0x00, 0x00, 0x00, 0xff, 0xf0, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xe0, 0x00, 0x00, 0x00, 0xff, 0xf8, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x7f, 0xfc, 0x00, 0x7f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0xff, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x7f, 0xfe, 0x00, 0x7f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0xff, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x7f, 0xff, 0x00, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0xff, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x80, 0x00, 0x00, 0x00, 0x3f, 0xf8, 0x03, 0xc0, 0x1f, 0xe0, 0x0f, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x80, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x03, 0x80, 0x07, 0xc0, 0x07, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x80, 0x00, 0x00, 0x00, 0x3f, 0xe0, 0x01, 0x80, 0x03, 0x80, 0x07, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xe0, 0x01, 0xc0, 0x00, 0x00, 0x07, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xe0, 0x00, 0xc0, 0x00, 0x00, 0x01, 0x80, 0x00, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0x20, 0x00, 0x0f, 0xe0, 0x00, 0xe0, 0x00, 0x00, 0x01, 0x80, 0x01, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0x60, 0x00, 0x0f, 0xe0, 0x00, 0xe0, 0x00, 0x00, 0x03, 0x80, 0x01, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0x60, 0x00, 0x0f, 0xc0, 0x00, 0xf0, 0x00, 0x00, 0x03, 0x80, 0x01, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0x70, 0x00, 0x0f, 0xc0, 0x00, 0xf8, 0x00, 0x00, 0x07, 0x80, 0x01, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0xf0, 0x00, 0x07, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0x07, 0x80, 0x01, 0xff, 0xc0,
  0xfc, 0x00, 0x00, 0xf0, 0x00, 0x07, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0x0f, 0x80, 0x01, 0xff, 0xc0,
  0xfc, 0x00, 0x00, 0xf8, 0x00, 0x07, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0x1f, 0x80, 0x01, 0xff, 0xc0,
```

```
  0xfc, 0x00, 0x00, 0xf8, 0x00, 0x03, 0xc0, 0x00, 0xfe, 0x00, 0x00, 0x3f, 0x80, 0x01, 0xff, 0xc0,
  0xf8, 0x00, 0x00, 0xfc, 0x00, 0x03, 0xc0, 0x00, 0xff, 0x00, 0x00, 0x3f, 0x00, 0x01, 0xff, 0xc0,
  0xf8, 0x00, 0x01, 0xfc, 0x00, 0x03, 0xc0, 0x00, 0xff, 0x00, 0x00, 0x7f, 0x00, 0x01, 0xff, 0xc0,
  0xf8, 0x00, 0x01, 0xfc, 0x00, 0x01, 0xc0, 0x00, 0xfe, 0x00, 0x00, 0xff, 0x00, 0x03, 0xff, 0xc0,
  0xf8, 0x00, 0x01, 0xfe, 0x00, 0x01, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0xff, 0x00, 0x03, 0xff, 0xc0,
  0xf8, 0x00, 0x03, 0xfe, 0x00, 0x01, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0x7f, 0x00, 0x03, 0xff, 0xc0,
  0xf0, 0x00, 0x03, 0xfe, 0x00, 0x01, 0xc0, 0x00, 0xf8, 0x00, 0x00, 0x3f, 0x00, 0x03, 0xff, 0xc0,
  0xf0, 0x00, 0x07, 0xfe, 0x00, 0x00, 0xc0, 0x00, 0xf0, 0x00, 0x00, 0x3f, 0x00, 0x03, 0xff, 0xc0,
  0xe0, 0x00, 0x07, 0xfe, 0x00, 0x00, 0xc0, 0x00, 0xf0, 0x00, 0x00, 0x1f, 0x00, 0x03, 0xff, 0xc0,
  0xe0, 0x00, 0x07, 0xff, 0x00, 0x00, 0xc0, 0x00, 0xe0, 0x00, 0x00, 0x1f, 0x00, 0x03, 0xff, 0xc0,
  0xe0, 0x00, 0x07, 0xff, 0x00, 0x00, 0x80, 0x01, 0xe0, 0x00, 0x00, 0x0f, 0x00, 0x03, 0xff, 0xc0,
  0xc0, 0x00, 0x07, 0xff, 0x80, 0x00, 0x00, 0x01, 0xc0, 0x00, 0x00, 0x07, 0x00, 0x03, 0xff, 0xc0,
  0xc0, 0x00, 0x0f, 0xff, 0x80, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x07, 0x00, 0x03, 0xff, 0xc0,
  0xc0, 0x00, 0x0f, 0xff, 0xc0, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x03, 0x00, 0x03, 0xff, 0xc0,
  0xe0, 0x00, 0x1f, 0xff, 0xe0, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x03, 0x00, 0x03, 0xff, 0xc0,
  0xf8, 0x00, 0x3f, 0xff, 0xf0, 0x00, 0x38, 0x01, 0xe0, 0x00, 0xe0, 0x01, 0xc0, 0x03, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x7b, 0xf8, 0x0d, 0xe0, 0x77, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0
};

void setup() {
  Serial.begin(115200);  //Inicializa la comunicación serial

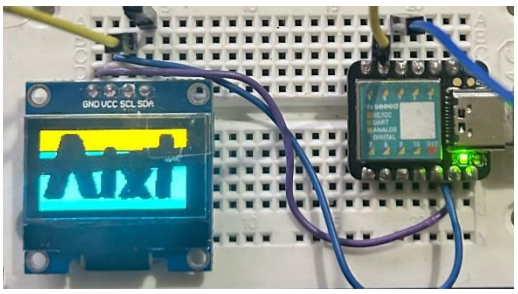
  //Inicializa el display, indica alimentacion del display y la direccion I2C para 128x64
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    //Mensaje de error, por si falla la inicialización
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000); // Pausa de 2s
  display.clearDisplay();  //Limpia el display

  //Dibuja la imagen, en las coordenadas estipuladas.
  display.drawBitmap(0, 0, image_data_Ruiarray, 128, 64, 1);
  display.display();  //Actualiza el display para que se muestren los cambio
}

void loop() {

}
```

Código para XIAO-SAMD21



Implementación del código en el sistema XIAO-SAMD21 – OLED

Raspberry Pi Pico

```
#include <Wire.h>  //Incluye libreria para comunicación I2C
#include <Adafruit_GFX.h>  //Incluye libreria de interfaz grafica
#include <Adafruit_SSD1306.h>  //Incluye libreria para display OLED SSD1306

#define SCREEN_WIDTH 128  //Define ancho del display, en pixeles
#define SCREEN_HEIGHT 64  //Define altura del display, en pixeles

// Declaracion del display SSD1306 connectado a I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

//Declaración de una variable estatica de 1024 bytes que contiene la imagen
static const uint8_t image_data_Ruiarray[1024] = {
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xe0, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0x80, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x01, 0xff, 0xc0,
  0xff, 0xff, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x01, 0xff, 0xc0,
  0xff, 0xff, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0xff, 0xc0,
  0xff, 0xff, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xff, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfe, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfe, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfe, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfc, 0x00, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfc, 0x00, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x7f, 0xc0,
  0xff, 0xfc, 0x00, 0x00, 0x03, 0xff, 0xfe, 0x03, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x7f, 0xc0,
  0xff, 0xf8, 0x00, 0x00, 0x03, 0xff, 0xf8, 0x01, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf8, 0x00, 0x00, 0x03, 0xff, 0xf8, 0x00, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf8, 0x00, 0x00, 0x03, 0xff, 0xf0, 0x00, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf8, 0x00, 0x00, 0x01, 0xff, 0xf0, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf0, 0x00, 0x00, 0x01, 0xff, 0xf0, 0x00, 0x3f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf0, 0x00, 0x00, 0x01, 0xff, 0xf0, 0x00, 0x3f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xf0, 0x00, 0x00, 0x01, 0xff, 0xf0, 0x00, 0x3f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xe0, 0x00, 0x00, 0x00, 0xff, 0xf0, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xe0, 0x00, 0x00, 0x00, 0xff, 0xf8, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x7f, 0xfc, 0x00, 0x7f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0xff, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x7f, 0xfe, 0x00, 0x7f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0xff, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x7f, 0xff, 0x00, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0xff, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0xc0, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x80, 0x00, 0x00, 0x00, 0x3f, 0xf8, 0x03, 0xc0, 0x1f, 0xe0, 0x0f, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x80, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x03, 0x80, 0x07, 0xc0, 0x07, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x80, 0x00, 0x00, 0x00, 0x3f, 0xe0, 0x01, 0x80, 0x03, 0x80, 0x07, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xe0, 0x01, 0xc0, 0x00, 0x00, 0x07, 0xc0, 0x00, 0xff, 0xc0,
  0xff, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xe0, 0x00, 0xc0, 0x00, 0x00, 0x01, 0x80, 0x00, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0x20, 0x00, 0x0f, 0xe0, 0x00, 0xe0, 0x00, 0x00, 0x01, 0x80, 0x01, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0x60, 0x00, 0x0f, 0xe0, 0x00, 0xe0, 0x00, 0x00, 0x03, 0x80, 0x01, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0x60, 0x00, 0x0f, 0xc0, 0x00, 0xf0, 0x00, 0x00, 0x03, 0x80, 0x01, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0x70, 0x00, 0x0f, 0xc0, 0x00, 0xf8, 0x00, 0x00, 0x07, 0x80, 0x01, 0xff, 0xc0,
  0xfe, 0x00, 0x00, 0xf0, 0x00, 0x07, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0x07, 0x80, 0x01, 0xff, 0xc0,
  0xfc, 0x00, 0x00, 0xf0, 0x00, 0x07, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0x0f, 0x80, 0x01, 0xff, 0xc0,
  0xfc, 0x00, 0x00, 0xf8, 0x00, 0x07, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0x1f, 0x80, 0x01, 0xff, 0xc0,
  0xfc, 0x00, 0x00, 0xf8, 0x00, 0x03, 0xc0, 0x00, 0xfe, 0x00, 0x00, 0x3f, 0x80, 0x01, 0xff, 0xc0,
```

```
  0xf8, 0x00, 0x00, 0xfc, 0x00, 0x03, 0xc0, 0x00, 0xff, 0x00, 0x00, 0x3f, 0x00, 0x01, 0xff, 0xc0,
  0xf8, 0x00, 0x01, 0xfc, 0x00, 0x03, 0xc0, 0x00, 0xff, 0x00, 0x00, 0x7f, 0x00, 0x01, 0xff, 0xc0,
  0xf8, 0x00, 0x01, 0xfc, 0x00, 0x01, 0xc0, 0x00, 0xfe, 0x00, 0x00, 0xff, 0x00, 0x03, 0xff, 0xc0,
  0xf8, 0x00, 0x01, 0xfe, 0x00, 0x01, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0xff, 0x00, 0x03, 0xff, 0xc0,
  0xf8, 0x00, 0x03, 0xfe, 0x00, 0x01, 0xc0, 0x00, 0xfc, 0x00, 0x00, 0x7f, 0x00, 0x03, 0xff, 0xc0,
  0xf0, 0x00, 0x03, 0xfe, 0x00, 0x01, 0xc0, 0x00, 0xf8, 0x00, 0x00, 0x3f, 0x00, 0x03, 0xff, 0xc0,
  0xf0, 0x00, 0x07, 0xfe, 0x00, 0x00, 0xc0, 0x00, 0xf0, 0x00, 0x00, 0x3f, 0x00, 0x03, 0xff, 0xc0,
  0xe0, 0x00, 0x07, 0xfe, 0x00, 0x00, 0xc0, 0x00, 0xf0, 0x00, 0x00, 0x1f, 0x00, 0x03, 0xff, 0xc0,
  0xe0, 0x00, 0x07, 0xff, 0x00, 0x00, 0xc0, 0x00, 0xe0, 0x00, 0x00, 0x1f, 0x00, 0x03, 0xff, 0xc0,
  0xe0, 0x00, 0x07, 0xff, 0x00, 0x00, 0x80, 0x01, 0xe0, 0x00, 0x00, 0x0f, 0x00, 0x03, 0xff, 0xc0,
  0xc0, 0x00, 0x07, 0xff, 0x80, 0x00, 0x00, 0x01, 0xc0, 0x00, 0x00, 0x07, 0x00, 0x03, 0xff, 0xc0,
  0xc0, 0x00, 0x0f, 0xff, 0x80, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x07, 0x00, 0x03, 0xff, 0xc0,
  0xc0, 0x00, 0x0f, 0xff, 0xc0, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x03, 0x00, 0x03, 0xff, 0xc0,
  0xe0, 0x00, 0x1f, 0xff, 0xe0, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x03, 0x00, 0x03, 0xff, 0xc0,
  0xf8, 0x00, 0x3f, 0xff, 0xf0, 0x00, 0x38, 0x01, 0xe0, 0x00, 0xe0, 0x01, 0xc0, 0x03, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x7b, 0xf8, 0x0d, 0xe0, 0x77, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0
};

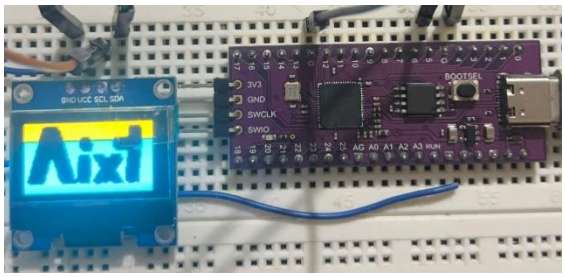
void setup() {
  Serial.begin(115200);  //Inicializa la comunicación serial

  //Inicializa el display, indica alimentacion del display y la direccion I2C para 128x64
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    //Mensaje de error, por si falla la inicialización
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000); //Pausea de 2s
  display.clearDisplay();  //Limpia el display
  //Dibuja la imagen, en las coordenadas estipuladas.
  display.drawBitmap(0, 0, image_data_Ruiarray, 128, 64, 1);
  display.display();  //Actualiza el display para que se muestren los cambio
}

void loop() {

}
```

Código para RASPBERRY PI-PICO



Implementación del código en el sistema RASPBERRY PI-PICO – OLED

A continuación, se presenta un ejemplo de código que genera una figura geométrica. Este código tiene como objetivo ayudar a comprender mejor el uso de ciertos elementos del lenguaje de programación.

Seeeduino XIAO

14a.ino

```
#include <Wire.h>  //Incluye libreria para comunicación I2C
#include <Adafruit_GFX.h>  //Incluye libreria de interfaz grafica
#include <Adafruit_SSD1306.h>  //Incluye libreria para display OLED SSD1306

#define SCREEN_WIDTH 128  //Define ancho del display, en pixeles
#define SCREEN_HEIGHT 64  //Define altura del display, en pixeles

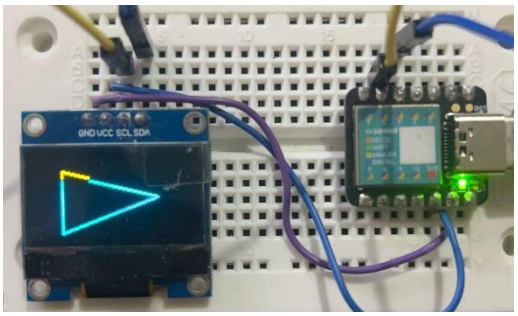
// Declaracion del display SSD1306 connectado a I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  Serial.begin(115200);  //Inicializa la comunicación serial

  //Inicializa el display, indica alimentacion del display y la direccion I2C para 128x64
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    //Mensaje de error, por si falla la inicialización
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000); // Pause for 2 seconds
  display.clearDisplay();  //Limpia el display
  //Dibuja un triangulo indicando los vertices (x1, y1, x2, y2, x3, y3)
  display.drawTriangle(20, 10, 100, 30, 20, 60, WHITE);
  display.display();  //Actualiza el display para que se muestren los cambios
}
void loop() {
  // put your main code here, to run repeatedly:

}
```

Código para XIAO-SAMD21



Implementación del código en el sistema XIAO-SAMD21 – OLED

Raspberry Pi Pico

.ino

```
#include <Wire.h>  //Incluye libreria para comunicación I2C
#include <Adafruit_GFX.h>  //Incluye libreria de interfaz grafica
#include <Adafruit_SSD1306.h>  //Incluye libreria para display OLED SSD1306

#define SCREEN_WIDTH 128  //Define ancho del display, en pixeles
#define SCREEN_HEIGHT 64  //Define altura del display, en pixeles

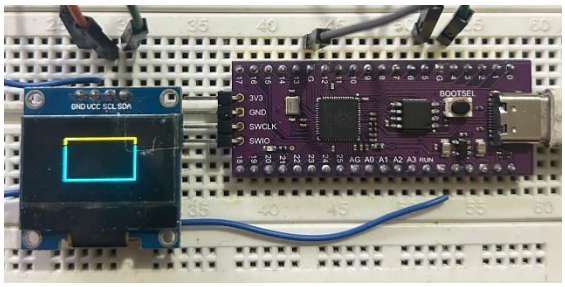
// Declaracion del display SSD1306 connectado a I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  Serial.begin(115200);  //Inicializa la comunicación serial

  //Inicializa el display, indica alimentacion del display y la direccion I2C para 128x64
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    //Mensaje de error, por si falla la inicialización
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000); //Pausa de 2s
  display.clearDisplay();  //Limpia el display
  //Dibuja un rectangulo, los dos primeros parametros son la posicion, los otros dos el tamaño
  display.drawRect(30, 10, 70, 40, WHITE);
  display.display();  //Actualiza el display para que se muestren los cambios
}

void loop() {
}
```

Código para RASPBERRY PI-PICO



Implementación del código en el sistema RASPBERRY PI-PICO – OLED

4. Al comenzar el desarrollo del soporte en lenguaje V, llegamos a la conclusión de crear un soporte genérico para la mayoría de los microcontroladores disponibles en la plataforma *Aixt.* Para ello, es necesario generar dos archivos: uno que transcriba las funciones del lenguaje nativo de los microcontroladores a lenguaje V, y otro que implemente estas funciones dentro de la plataforma, permitiendo que puedan ser llamadas.

```
#include <Wire.h>  //Incluye libreria para comunicación ADDRESS
#include <Adafruit_GFX.h>  //Incluye libreria de interfaz grafica
#include <Adafruit_SSD1306.h>  //Incluye libreria para display OLED SSD1306

#define DISPLAY_SETUP(TYPE, ADDRESS)                        display.begin(TYPE, ADDRESS)
#define DISPLAY_CLEARDISPLAY()                              display.clearDisplay()
#define DISPLAY_SETTEXTSIZE(NUMBER)                         display.stTextSize(NUMBER)
#define DISPLAY_TEXTCOLOR(CHARACTER)                        display.setTextColor(CHARACTER)
#define DISPLAY_SETCURSOR(X,Y)                              display.setCursor(X,Y)
#define DISPLAY_PRINTLN(MESSAGE)                            display.println(MESSAGE)
#define DISPLAY_DISPLAY()                                   display.display()
#define DISPLAY_DRAWPIXEL(X,Y,COLOR)                        display.drawPixel(X,Y,COLOR)
#define DISPLAY_DRAWRECT(X,Y,WIDTH,HEIGHT,COLOR)            display.drawRect(X,Y,WIDTH,HEIGHT,COLOR)
#define DISPLAY_FILLRECT(X,Y,WIDTH,HEIGHT,COLOR)            display.fillRect(X,Y,WIDTH,HEIGHT,COLOR)
#define DISPLAY_DRAWCICRLE(X,Y,RADIO,COLOR)                 display.drawCircle(X,Y,RADIO,COLOR)
#define DISPLAY_FILLCIRCLE(X,Y,RADIO,COLOR)                 display.fillCircle(X,Y,RADIO,COLOR)
#define DISPLAY_DRAWBITMAP(X,Y,ARRAY_IMAGE,WIDTH,HEIGHT,COLOR)  display.drawBitmap(X,Y,ARRAY_IMAGE,WIDTH,HEIGHT,COLOR)
```

Equivalente de funciones de lenguaje C a V

Se evidencia que las funciones que se encuentran en mayúscula son las que ya se encuentran transcritas en lenguaje V.

oled.c.v

lib > oled > arduino > oled.c.v

```
module oled

#include "oled.c"

@[as_macro] pub const black     = 0
@[as_macro] pub const white     = 1
@[as_macro] pub const inverse   = 2


fn C.DISPLAY_BEGIN (type_ any , address any) bool
fn C.DISPLAY_CLEARDISPLAY(any)
fn C.DISPLAY_SETTEXTSIZE(number any)
fn C.DISPLAY_TEXTCOLOR(character any)
fn C.DISPLAY_SETCURSOR(x any , y any)
fn C.DISPLAY_PRINTLN(message any)
fn C.DISPLAY_DISPLAY(any)
fn C.DISPLAY_DRAWPIXEL(x any , y any , color any)
fn C.DISPLAY_DRAWRECT(x any , y any , width any , height any , color any)
fn C.DISPLAY_FILLRECT(x any , y any , width any , height any , color any)
fn C.DISPLAY_DRAWCIRCLE(x any , y any , radio any , color any)
fn C.DISPLAY_FILLCIRCLE(x any , y any , radio any , color any)
fn C.DISPLAY_DRAWBITMAP(x any , y any , array_image any , width any , height any , color any)
```

Archivo de soporte de funciones.

Una vez que estos dos archivos estén debidamente configurados, procedemos a describir los módulos de las funciones para que, al ser llamadas en un proyecto, ocupen menos memoria en el microcontrolador y funcionen de manera más eficiente.

1. En el siguiente modulo, se muestra la correcta definición de la función “clear”, que al momento de ser ejecutada limpia cualquier información que se encuentre en el dispositivo OLED.

```
module oled

@[as_macro]
pub fn clear()  {
    C.DISPLAY_CLEARDISPLAY()
    C.DISPLAY_DISPLAY()
}
```

Módulo CLEAR

2. En el siguiente modulo, evidenciamos la correcta descripción de la función “draw”, la cual puede ser implementada para dibujar alguna figura en el dispositivo OLED.

```
module oled

@[as_macro]
pub fn pixel(x any , y any , color any)  {
    C.DISPLAY_DRAWPIXEL(x,y,color)
    C.DISPLAY_DISPLAY()
}


@[as_macro]
pub fn rect(x any , y any , width any , height any , color any) {
    C.DISPLAY_DRAWRECT(x,y,width,height,color)
    C.DISPLAY_DISPLAY()
}


@[as_macro]
pub fn circle(x any , y any , radio any , color any) {
    C.DISPLAY_DRAWCIRCLE(x,y,radio,color)
    C.DISPLAY_DISPLAY()
}
```

Módulo DRAW

Evidenciamos que, para el funcionamiento de esta función, se requieren una serie de datos según la figura que se desee graficar, como las coordenadas de inicio, el tamaño de la figura, el color (dependiendo del dispositivo OLED utilizado), entre otros.

3. La siguiente función está relacionada con la anterior, ya que permite rellenar las figuras graficadas previamente.

```
module oled

@[as_macro]
pub fn fillrect(x any , y any , width any , height any , color any)  {
    C.DISPLAY_FILLRECT(x,y,width,height,color)
    C.DISPLAY_DISPLAY()
}

@[as_macro]
pub fn fillcircle(x any , y any , radio any , color any)  {
    C.DISPLAY_FILLCIRCLE(x,y,radio,color)
    C.DISPLAY_DISPLAY()
}
```

Módulo FILL

De igual manera se evidencia que para rellenar una figura se requiere de una serie de datos específicos.

4. El siguiente módulo, nos ayuda a imprimir un texto en el dispositivo OLED.

```
module oled

@[as_macro]
pub fn println(message string)  {
        C.DISPLAY_PRINTLN(message)
        C.DISPLAY_DISPLAY()
}
```

Módulo PRINTLN

5. La siguiente definición está relacionada con el módulo anterior, ya que permite configurar algunas características del texto impreso previamente, como el tamaño y el color.

```
module oled

@[as_macro] {
pub fn size(number any)
    C.DISPLAY_SETTEXTSIZE(number)
    C.DISPLAY_DISPLAY()
}


@[as_macro] {
pub fn color(character any)
    C.DISPLAY_TEXTCOLOR(character)
    C.DISPLAY_DISPLAY()
}
```

Módulo TEXT

6. El siguiente módulo es esencial para trabajar con dispositivos OLED, ya que inicializa la pantalla con una comunicación y dirección específicas.

```
module oled

@[as_macro]
pub fn setup(type_ any, address any) {
    C.DISPLAY_BEGIN(type_, address)
    C.DISPLAY_DISPLAY()
}
```

Módulo SETUP

7. La siguiente definición se implementa momento de querer graficar una imagen.

```
module oled

@[as_macro]
pub fn drawbitmap(x any , y any , array_image any , width any , height any , color any) {
    C.DISPLAY_DRAWBITMAP(x,y,array_image,width,height,color)
    C.DISPLAY_DISPLAY()
}
```

Módulo BITMAP

   Se evidencia que, para el correcto funcionamiento de este módulo, requiere de ciertos parámetros, como las coordenadas, el tamaño de la imagen, el arreglo de bits donde se encuentra guardada la imagen, entre otros.

Se observa que todos los módulos inician con *@[as_macro]*, lo que indica que su contenido se expandirá en tiempo de compilación, optimizando el rendimiento del microcontrolador, al evitar llamadas innecesarias a otras funciones.

También se evidencia que todos los módulos incluyen la línea *C.DISPLAY_DISPLAY()*, la cual actualiza el display OLED para que los cambios realizados sean visibles.

5. CONCLUSIONES

Durante el proceso se hizo evidente la facilidad de uso de *Aixt*, demostrando su naturaleza intuitiva y su capacidad para simplificar el desarrollo de proyectos de software libre. Su estructura modular y su compatibilidad son diversos microcontroladores permiten a los desarrolladores integrar nuevas funcionalidades de manera eficiente, reduciendo la complejidad y el tiempo de implementación.

El éxito alcanzado en este proyecto demuestra el potencial de la plataforma *Aixt* para optimizar las capacidades y la versatilidad de los microcontroladores que son soportador por la plataforma en una amplia gama de aplicaciones. A lo largo del proceso, se logró mejorar tanto la capacidad de programación como la facilidad de uso de estos dispositivos, ofreciendo una solución más eficiente y accesible.

Con la implementación del módulo de visualización, se pudo apreciar la gran capacidad y funcionalidad de los microcontroladores, así como su facilidad de integración para controlar diversas tareas programadas. Esto los convierte en una herramienta versátil y adecuada para ser implementada en una amplia variedad de áreas.

Durante el desarrollo de este proyecto, se cumplieron satisfactoriamente los objetivos establecidos y se abrió la idea de desarrollar una librería completa para distintos microcontroladores soportados por esta plataforma.

6. REFERENCIAS

[1] Fernando Martínez Santa (2022).
Microcontrollers Programming Framework based on a V-like programming Language

[2] Fernando Martínez Santa, «Aixt,» 2022. [Online]
https://github.com/fermarsan/aixt

[3] Santa Fernando, Orjuela Santiago, Martines H Fredy (2022).
Microcontrollers Programming Framework base don a V-like Programming Language.
https://thesai.org/Publications/ViewPaper?Volume=13&Issue=12&Code=IJACSA&SerialNo=5

[6] Martinez Santa Fernando (2011).
Profe Santa [canal de YouTube].
https://www.youtube.com/@fernandomartinezsanta7153

[7] Raspberry Pi (2020-2024).
Raspberry Pi Pico Datasheet.
https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

[8] Seeed Studio XIAO Series.
System-on-Modules (SoM), User Manual.
https://files.seeedstudio.com/wiki/XIAO/Seeed-Studio-XIAO-Series-SOM-Datasheet.pdf

[9] SunFounder (2024).
Módulo de pantalla OLED.
https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/22-component_oled.html