Inclusión al proyecto Aixt de los microcontroladores de la familia Microchip PIC16F para la transcompilación de su código fuente desde un lenguaje basado en V a lenguaje C.

AUTORES:

Luis Alfredo Pinto Medina - lapintom@udistrital.edu.co - 20192572029

Cesar Alejandro Roa Acosta - caroaa@udistrital.edu.co – 20192572044

1. Introducción:

Los microcontroladores desempeñan un papel crucial al actuar como cerebros inteligentes detrás de una amplia gama de dispositivos electrónicos, exploraremos qué distingue a un microcontrolador de la familia PIC16F, sumergiéndonos en sus características técnicas y su capacidad, demostrando por qué estos microcontroladores son la columna vertebral de muchas innovaciones tecnológicas.

Por otro lado, el proyecto Aixt presenta una estructura de programación diseñada para microcontroladores de recursos limitados, utilizando un lenguaje moderno inspirado en V. Su estructura se compone de tres elementos principales: el lenguaje Aixt basado en V, el transcompilador que traduce Aixt a C, y la API de Aixt que normaliza la configuración y operaciones en microcontroladores.

El lenguaje Aixt es un subconjunto de V, incorporando su sintaxis y excluyendo funciones de programación costosas en memoria. Nacido en 2022 bajo las aspiraciones del proyecto Sokae, Aixt busca llenar el vacío entre las estructuras de Arduino (basadas en C) y Microphyton (basadas en Python).

A diferencia de Sokae, que se basa en Rust, Aixt utiliza el lenguaje V por su enfoque minimalista. El proyecto aborda la necesidad de migración eficiente entre lenguajes, especialmente en microcontroladores. Enfocándose en microcontroladores de la familia Microchip PIC16F, busca integrarla en Aixt mediante una API en C y archivo de configuración, permitiendo la traducción automatizada entre los dos lenguajes. Aixt optimiza la eficiencia en recursos para microcontroladores de la familia Microchip PIC16F, adecuándose a aplicaciones con limitaciones de recursos. Este proyecto ofrece una oportunidad para aplicar habilidades en Tecnología en Electricidad de Media y Baja Tensión, contribuyendo con soluciones innovadoras en el mercado tecnológico.

2. Objetivos

Objetivo general

Dar soporte al proyecto Aixt para la integración de los microcontroladores de la marca Microchip de la serie PIC16F.

Objetivos específicos

- Escribir un archivo de configuración que contenga las características específicas de los microcontroladores de la marca Microchip de la serie PIC16F y del compilador MPLAB XC8 para que el transcompilador pueda convertir efectivamente su código fuente en Aixt (lenguaje basado en V) a lenguaje C.

- Desarrollar un conjunto de funciones básicas de funcionamiento en lenguaje C para los microcontroladores de la marca Microchip de la serie PIC16F que puedan ser usadas o invocadas desde el lenguaje Aixt.

- Hacer pruebas de funcionamiento de las funciones implementadas sobre los microcontroladores de la marca Microchip de la serie PIC16F.

- Realizar una documentación adecuada que permita a los usuarios entender el funcionamiento del proyecto aixt enfocado a los microcontroladores de la marca Microchip de la serie PIC16F, que a su vez permitirá contribuir al proyecto si así lo desean.

3. Plan de trabajo (Individual)

Cesar Alejandro Roa Acosta

Semana 1 a la 6

Para comenzar a trabajar con los microcontroladores PIC16F676 y PIC16F873A, se requirió familiarizarse con el compilador XC8. Esto implicó la lectura e interpretación del manual para comprender sus características principales. Además, se procedió a la descarga del software correspondiente para llevar a cabo pruebas de funcionamiento en los microcontroladores. Dichas pruebas tenían como objetivo determinar las aplicaciones que cada microcontrolador podría soportar. Para el PIC16F84A, que solo dispone de entradas y salidas digitales, se realizaron pruebas para verificar el funcionamiento de todas las entradas y salidas. En cuanto al PIC16F886, que cuenta con entradas y salidas digitales, salida PWM, comunicación serial y conversor analógico a digital ADC, se programaron archivos en C para cada una de estas funciones con el fin de comprobar su correcto funcionamiento.

Semana 7 a la 12

Considerando el conocimiento previo sobre las aplicaciones compatibles con el PIC16F676 y PIC16F873A, se procede a la creación de la API. En esta API se incluirán todas las funciones admitidas por los microcontroladores, ya sean de entrada y salida digital, PWM, conversión analógica a digital (ADC) o comunicación serial. La API estará compuesta por archivos en lenguaje C que detallarán cada una de estas funciones.

Semana 13 a la 16

Se implementó un respaldo para el transcompilador del lenguaje Aixt al C dirigido a la serie de microcontroladores PIC16F676 y PIC16F873A. En esta fase, se creó un archivo de configuración en formato TOML, el cual se empleará como insumo esencial para el funcionamiento del transcompilador.

Semana 17 a la 20

Una vez que cada función de la API ha sido transcompilada para los microcontroladores PIC16F676 y PIC16F873A, se llevan a cabo pruebas de funcionamiento del archivo resultante de la transcompilación. Esto se hace para asegurar su correcto desempeño. Es importante resaltar que, independientemente de que se escriba en V y se transcompile en Aixt para su conversión al lenguaje C, el código debe funcionar de manera idéntica a como lo haría si fuera escrito directamente en C.

Semana 21 a la 24

La arquitectura del hardware de los microcontroladores PIC16F676 y PIC16F873A, así como la interfaz de programación de aplicaciones y la serialización, requieren contar con un respaldo documental que brinde explicaciones detalladas, identificación precisa y caracterización exhaustiva de cada aspecto. En consecuencia, se llevó a cabo la creación de este documento para satisfacer dichas necesidades.

Luis Alfredo Pinto Medina

Semana 1 a la 6

Para comenzar a trabajar con los microcontroladores PIC16F84A y PIC16F886, se requirió familiarizarse con el compilador XC8. Esto implicó la lectura e interpretación del manual para comprender sus características principales. Además, se procedió a la descarga del software correspondiente para llevar a cabo pruebas de funcionamiento en los microcontroladores. Dichas pruebas tenían como objetivo determinar las aplicaciones que cada microcontrolador podría soportar. Para el PIC16F84A, que solo dispone de entradas y salidas digitales, se realizaron pruebas para verificar el funcionamiento de todas las entradas y salidas. En cuanto al PIC16F886, que cuenta con entradas y salidas digitales, salida PWM, comunicación serial y conversor analógico a digital ADC, se programaron archivos en C para cada una de estas funciones con el fin de comprobar su correcto funcionamiento.

Semana 7 a la 12

Considerando el conocimiento previo sobre las aplicaciones compatibles con el PIC16F84A y el PIC16F886, se procede a la creación de la API. En esta API se incluirán todas las funciones admitidas por los microcontroladores, ya sean de entrada y salida digital, PWM, conversión analógica a digital (ADC) o comunicación serial. La API estará compuesta por archivos en lenguaje C que detallarán cada una de estas funciones.

Semana 13 a la 16

Se implementó un respaldo para el transcompilador del lenguaje Aixt al C dirigido a la serie de microcontroladores PIC16F84A y el PIC16F886. En esta fase, se creó un archivo de configuración en formato TOML, el cual se empleará como insumo esencial para el funcionamiento del transcompilador.

Semana 17 a la 20

Una vez que cada función de la API ha sido transpilada para los microcontroladores PIC16F84A y PIC16F886, se llevan a cabo pruebas de funcionamiento del archivo resultante de la transpilación. Esto se hace para asegurar su correcto desempeño. Es importante resaltar que, independientemente de que se escriba en V y se transcompile en Aixt para su conversión al lenguaje C, el código debe funcionar de manera idéntica a como lo haría si fuera escrito directamente en C.

Semana 21 a la 24

La arquitectura del hardware de los microcontroladores PIC16F84A y PIC16F886, así como la interfaz de programación de aplicaciones y la serialización, requieren contar con un respaldo documental que brinde explicaciones detalladas, identificación precisa y caracterización exhaustiva de cada aspecto. En consecuencia, se llevó a cabo la creación de este documento para satisfacer dichas necesidades.

4. Descripción de cada resultado alcanzado en el desarrollo de la pasantía en función de los objetivos propuestos, debidamente ordenados y expuestos en forma coherente, de acuerdo con el plan de trabajo.

Hemos establecido coincidencias entre las funciones incorporadas en el compilador y aquellas implementadas en el proyecto AIXT. Este procedimiento tiene como objetivo normalizar la API para asegurar su compatibilidad con cualquier microcontrolador. La premisa fundamental consiste en redefinir estas funciones y proporcionar un respaldo uniforme. Este método ofrece la ventaja de que, al familiarizarnos con la API y los nombres de las funciones, podemos programar cualquier otro microcontrolador si ya tenemos dominio del lenguaje de programación.

- PIC16F676

  Se integraron:

  - ✓ Entradas y salidas digitales
  - ✓ Configuración para controlar un puerto como salida digital
  - ✓ Conversión analógica a digital (ADC)

  A cada PIN y cada PUERTO se le asignaron el siguiente nombre:

PIN O PUERTO	NOMBRE	REGISTRO	TIPO
PUERTO A	#define TRISa	TRISA	Configura el puerto A
PUERTO C	#define TRISc	TRISC	Configura el puerto C
PUERTO A	#define PORTa	PORTA	Configura el puerto A
PUERTO C	#define PORTc	PORTC	Configura el puerto C
A0	#define a0_s	TRISAbits.TRISA0	Configura el pin A0
A1	#define a1_s	TRISAbits.TRISA1	Configura el pin A1
A2	#define a2_s	TRISAbits.TRISA2	Configura el pin A2
A3	#define a3_s	TRISAbits.TRISA3	Configura el pin A3
A4	#define a4_s	TRISAbits.TRISA4	Configura el pin A4
A5	#define a5_s	TRISAbits.TRISA5	Configura el pin A5
C0	#define c0_s	TRISAbits.TRISC0	Configura el pin C0
C1	#define c1_s	TRISAbits.TRISC1	Configura el pin C1
C2	#define c2_s	TRISAbits.TRISC2	Configura el pin C2
C3	#define c3_s	TRISAbits.TRISC3	Configura el pin C3

C4	#define c4_s	TRISAbits.TRISC4	Configura el pin C4
C5	#define c5_s	TRISAbits.TRISC5	Configura el pin C5
A0	#define a0	PORTAbits.RA0	Entrada o salida
A1	#define a1	PORTAbits.RA1	Entrada o salida
A2	#define a2	PORTAbits.RA2	Entrada o salida
A3	#define a3	PORTAbits.RA3	Entrada o salida
A4	#define a4	PORTAbits.RA4	Entrada o salida
A5	#define a5	PORTAbits.RA5	Entrada o salida
C0	#define c0_s	PORTCbits.RC0	Entrada o salida
C1	#define c1_s	PORTCbits.RC1	Entrada o salida
C2	#define c2_s	PORTCbits.RC2	Entrada o salida
C3	#define c3_s	PORTCbits.RC3	Entrada o salida
C4	#define c4_s	PORTCbits.RC4	Entrada o salida
C5	#define c5_s	PORTCbits.RC5	Entrada o salida

- PIC16F84A

  Se integraron:

  - ✓ Entradas y salidas digitales
  - ✓ Configuración para controlar un puerto como salida digital

PIN O PUERTO	NOMBRE	REGISTRO	TIPO
PUERTO A	#define TRISa	TRISA	Configura el puerto A
PUERTO B	#define TRISb	TRISB	Configura el puerto B
PUERTO A	#define PORTa	PORTA	Configura el puerto A
PUERTO B	#define PORTb	PORTB	Configura el puerto B
A0	#define a0_s	TRISAbits.TRISA0	Configura el pin A0
A1	#define a1_s	TRISAbits.TRISA1	Configura el pin A1
A2	#define a2_s	TRISAbits.TRISA2	Configura el pin A2
A3	#define a3_s	TRISAbits.TRISA3	Configura el pin A3
B0	#define b0_s	TRISAbits.TRISB0	Configura el pin B0
B1	#define b1_s	TRISAbits.TRISB1	Configura el pin B1
B2	#define b2_s	TRISAbits.TRISB2	Configura el pin B2
B3	#define b3_s	TRISAbits.TRISB3	Configura el pin B3
B4	#define b4_s	TRISAbits.TRISB4	Configura el pin B4
B5	#define b5_s	TRISAbits.TRISB5	Configura el pin B5

B6	#define b6_s	TRISAbits.TRISB6	Configura el pin B6
B7	#define b7_s	TRISAbits.TRISB7	Configura el pin B7
A0	#define a0	PORTAbits.RA0	Entrada o salida
A1	#define a1	PORTAbits.RA1	Entrada o salida
A2	#define a2	PORTAbits.RA2	Entrada o salida
A3	#define a3	PORTAbits.RA3	Entrada o salida
B0	#define b0_s	PORTBbits.RB0	Entrada o salida
B1	#define b1_s	PORTBbits.RB1	Entrada o salida
B2	#define b2_s	PORTBbits.RB2	Entrada o salida
B3	#define b3_s	PORTBbits.RB3	Entrada o salida
B4	#define b4_s	PORTBbits.RB4	Entrada o salida
B5	#define b5_s	PORTBbits.RB5	Entrada o salida
B6	#define b6_s	PORTBbits.RB6	Entrada o salida
B7	#define b7_s	PORTBbits.RB7	Entrada o salida

- PIC16F873A

  Se integraron:

  - ✓ Entradas y salidas digitales
  - ✓ Conversión analógica a digital (ADC)
  - ✓ Configuración para controlar un puerto como salida digital
  - ✓ Salida PWM
  - ✓ Uart recepción de datos
  - ✓ Uart transmisión de datos

PIN O PUERTO	NOMBRE	REGISTRO	TIPO
PUERTO A	#define TRISa	TRISA	Configura el puerto A
PUERTO B	#define TRISb	TRISB	Configura el puerto B
PUERTO C	#define TRISc	TRISC	Configura el puerto C
PUERTO A	#define PORTa	PORTA	Configura el puerto A
PUERTO B	#define PORTb	PORTB	Configura el puerto B
PUERTO C	#define PORTc	PORTC	Configura el puerto C
A0	#define a0_s	TRISAbits.TRISA0	Configura el pin A0
A1	#define a1_s	TRISAbits.TRISA1	Configura el pin A1
A2	#define a2_s	TRISAbits.TRISA2	Configura el pin A2
A3	#define a3_s	TRISAbits.TRISA3	Configura el pin A3
A4	#define a4_s	TRISAbits.TRISA4	Configura el pin A4
A5	#define a5_s	TRISAbits.TRISA5	Configura el pin A5
B0	#define b0_s	TRISAbits.TRISB0	Configura el pin B0

B1	#define b1_s	TRISAbits.TRISB1	Configura el pin B1
B2	#define b2_s	TRISAbits.TRISB2	Configura el pin B2
B3	#define b3_s	TRISAbits.TRISB3	Configura el pin B3
B4	#define b4_s	TRISAbits.TRISB4	Configura el pin B4
B5	#define b5_s	TRISAbits.TRISB5	Configura el pin B5
B6	#define b6_s	TRISAbits.TRISB6	Configura el pin B6
B7	#define b7_s	TRISAbits.TRISB7	Configura el pin B7
C0	#define c0_s	TRISAbits.TRISC0	Configura el pin C0
C1	#define c1_s	TRISAbits.TRISC1	Configura el pin C1
C2	#define c2_s	TRISAbits.TRISC2	Configura el pin C2
C3	#define c3_s	TRISAbits.TRISC3	Configura el pin C3
C4	#define c4_s	TRISAbits.TRISC4	Configura el pin C4
C5	#define c5_s	TRISAbits.TRISC5	Configura el pin C5
C6	#define c6_s	TRISAbits.TRISC6	Configura el pin C6
C7	#define c7_s	TRISAbits.TRISC7	Configura el pin C7
A0	#define a0	PORTAbits.RA0	Entrada o salida
A1	#define a1	PORTAbits.RA1	Entrada o salida
A2	#define a2	PORTAbits.RA2	Entrada o salida
A3	#define a3	PORTAbits.RA3	Entrada o salida

A4	#define a4	PORTAbits.RA4	Entrada o salida
A5	#define a5	PORTAbits.RA5	Entrada o salida
B0	#define b0_s	PORTBbits.RB0	Entrada o salida
B1	#define b1_s	PORTBbits.RB1	Entrada o salida
B2	#define b2_s	PORTBbits.RB2	Entrada o salida
B3	#define b3_s	PORTBbits.RB3	Entrada o salida
B4	#define b4_s	PORTBbits.RB4	Entrada o salida
B5	#define b5_s	PORTBbits.RB5	Entrada o salida
B6	#define b6_s	PORTBbits.RB6	Entrada o salida
B7	#define b7_s	PORTBbits.RB7	Entrada o salida
C0	#define c0_s	PORTCbits.RC0	Entrada o salida
C1	#define c1_s	PORTCbits.RC1	Entrada o salida
C2	#define c2_s	PORTCbits.RC2	Entrada o salida
C3	#define c3_s	PORTCbits.RC3	Entrada o salida
C4	#define c4_s	PORTCbits.RC4	Entrada o salida
C5	#define c5_s	PORTCbits.RC5	Entrada o salida
C6	#define c6_s	PORTCbits.RC6	Entrada o salida
C7	#define c7_s	PORTCbits.RC7	Entrada o salida

- PIC16F886

  Se integraron:

  - ✓ Entradas y salidas digitales
  - ✓ Conversión analógica a digital (ADC)
  - ✓ Configuración para controlar un puerto como salida digital
  - ✓ Salida PWM
  - ✓ Uart recepción de datos
  - ✓ Uart transmisión de datos

PIN O PUERTO	NOMBRE	REGISTRO	TIPO
PUERTO A	#define TRISa	TRISA	Configura el puerto A
PUERTO B	#define TRISb	TRISB	Configura el puerto B
PUERTO C	#define TRISc	TRISC	Configura el puerto C
PUERTO A	#define PORTa	PORTA	Configura el puerto A
PUERTO B	#define PORTb	PORTB	Configura el puerto B
PUERTO C	#define PORTc	PORTC	Configura el puerto C
A0	#define a0_s	TRISAbits.TRISA0	Configura el pin A0
A1	#define a1_s	TRISAbits.TRISA1	Configura el pin A1
A2	#define a2_s	TRISAbits.TRISA2	Configura el pin A2
A3	#define a3_s	TRISAbits.TRISA3	Configura el pin A3
A4	#define a4_s	TRISAbits.TRISA4	Configura el pin A4
A5	#define a5_s	TRISAbits.TRISA5	Configura el pin A5
A6	#define a6_s	TRISAbits.TRISA6	Configura el pin A6

A7	#define a7_s	TRISAbits.TRISA7	Configura el pin A7
B0	#define b0_s	TRISAbits.TRISB0	Configura el pin B0
B1	#define b1_s	TRISAbits.TRISB1	Configura el pin B1
B2	#define b2_s	TRISAbits.TRISB2	Configura el pin B2
B3	#define b3_s	TRISAbits.TRISB3	Configura el pin B3
B4	#define b4_s	TRISAbits.TRISB4	Configura el pin B4
B5	#define b5_s	TRISAbits.TRISB5	Configura el pin B5
B6	#define b6_s	TRISAbits.TRISB6	Configura el pin B6
B7	#define b7_s	TRISAbits.TRISB7	Configura el pin B7
C0	#define c0_s	TRISAbits.TRISC0	Configura el pin C0
C1	#define c1_s	TRISAbits.TRISC1	Configura el pin C1
C2	#define c2_s	TRISAbits.TRISC2	Configura el pin C2
C3	#define c3_s	TRISAbits.TRISC3	Configura el pin C3
C4	#define c4_s	TRISAbits.TRISC4	Configura el pin C4
C5	#define c5_s	TRISAbits.TRISC5	Configura el pin C5
C6	#define c6_s	TRISAbits.TRISC6	Configura el pin C6
C7	#define c7_s	TRISAbits.TRISC7	Configura el pin C7
A0	#define a0	PORTAbits.RA0	Entrada o salida
A1	#define a1	PORTAbits.RA1	Entrada o salida

A2	#define a2	PORTAbits.RA2	Entrada o salida
A3	#define a3	PORTAbits.RA3	Entrada o salida
A4	#define a4	PORTAbits.RA4	Entrada o salida
A5	#define a5	PORTAbits.RA5	Entrada o salida
A6	#define a6	PORTAbits.RA6	Entrada o salida
A7	#define a7	PORTAbits.RA7	Entrada o salida
B0	#define b0_s	PORTBbits.RB0	Entrada o salida
B1	#define b1_s	PORTBbits.RB1	Entrada o salida
B2	#define b2_s	PORTBbits.RB2	Entrada o salida
B3	#define b3_s	PORTBbits.RB3	Entrada o salida
B4	#define b4_s	PORTBbits.RB4	Entrada o salida
B5	#define b5_s	PORTBbits.RB5	Entrada o salida
B6	#define b6_s	PORTBbits.RB6	Entrada o salida
B7	#define b7_s	PORTBbits.RB7	Entrada o salida
C0	#define c0_s	PORTCbits.RC0	Entrada o salida
C1	#define c1_s	PORTCbits.RC1	Entrada o salida
C2	#define c2_s	PORTCbits.RC2	Entrada o salida
C3	#define c3_s	PORTCbits.RC3	Entrada o salida
C4	#define c4_s	PORTCbits.RC4	Entrada o salida

C5	#define c5_s	PORTCbits.RC5	Entrada o salida
C6	#define c6_s	PORTCbits.RC6	Entrada o salida
C7	#define c7_s	PORTCbits.RC7	Entrada o salida

Se llevó a cabo la integración de un código fuente en Aixt con el objetivo de garantizar una traducción adecuada y oportuna hacia la estructura general definida para los microcontroladores PIC16F676, PIC16F84A, PIC16F873A y PIC16F886. Para este propósito, se estableció una carpeta dedicada a la API, la cual alberga archivos .c.v correspondientes a las funciones disponibles para cada uno de los microcontroladores mencionados.

- PIC16F676

FUNCIÓN PIN		
NOMBRE	DESCRIPCIÓN	TIPO
#define pin__digital ()	ANSEL = 0	Entradas o Salidas Digitales
#define pin__high (PIN_NAME)	PIN_NAME = 1	Activa el pin
#define pin__low (PIN_NAME)	PIN_NAME = 0	Desactiva el pin
#define pin__output	0	Salida
#define pin__input	1	Entrada
#define pin__read (PIN_NAME)	PIN_NAME	Lee el pin
#define pin__write (PIN_NAME, VAL)	PIN_NAME = VAL	Escribe el valor del pin
#define pin__setup (PIN_NAME, PIN_MODE)	PIN_NAME ##_s = PIN_MODE	Configura el pin

FUNCIÓN PORT		
#define port__read(PORT_NAME)	PORT ## PORT_NAME	Lee el puerto
#define port__setup(PORT_NAME, VALUE)	TRIS ## PORT_NAME = VALUE	Configura el puerto
#define port__write (PORT_NAME, VALUE)	PORT ## PORT_NAME = VALUE	Escribe el valor del puerto

FUNCIÓN ADC		
#define adc__read(CHANNEL)	CHANNEL	Configura el canal
#define adc__setup()	ADCON0, ADCON1	Configura el ADC

FUNCIÓN TIME		
#define time__sleep(TIME)	__delay_s(TIME)	Tiempo en segundos
#define time__sleep_ms(TIME)	__delay_ms(TIME)	Tiempo en milisegundos
#define time__sleep_us(TIME)	__delay_us(TIME)	Tiempo en microsegundos

- PIC16F84A

FUNCIÓN PIN		
NOMBRE	DESCRIPCIÓN	TIPO
#define pin__digital ()	ANSEL = 0	Entradas o Salidas Digitales
#define pin__high (PIN_NAME)	PIN_NAME = 1	Activa el pin

#define pin__low (PIN_NAME)	PIN_NAME = 0	Desactiva el pin
#define pin__output	0	Salida
#define pin__input	1	Entrada
#define pin__read (PIN_NAME)	PIN_NAME	Lee el pin
#define pin__write (PIN_NAME, VAL)	PIN_NAME = VAL	Escribe el valor del pin
#define pin__setup (PIN_NAME, PIN_MODE)	PIN_NAME ##_s = PIN_MODE	Configura el pin

FUNCIÓN PORT		
#define port__read(PORT_NAME)	PORT ## PORT_NAME	Lee el puerto
#define port__setup(PORT_NAME, VALUE)	TRIS ## PORT_NAME = VALUE	Configura el puerto
#define port__write (PORT_NAME, VALUE)	PORT ## PORT_NAME = VALUE	Escribe el valor del puerto

FUNCIÓN TIME		
#define time__sleep(TIME)	__delay_s(TIME)	Tiempo en segundos
#define time__sleep_ms(TIME)	__delay_ms(TIME)	Tiempo en milisegundos
#define time__sleep_us(TIME)	__delay_us(TIME)	Tiempo en microsegundos

- PIC16F873A

FUNCIÓN PIN		
NOMBRE	DESCRIPCIÓN	TIPO
#define pin__digital ()	ANSEL = 0	Entradas o Salidas Digitales
#define pin__high (PIN_NAME)	PIN_NAME = 1	Activa el pin
#define pin__low (PIN_NAME)	PIN_NAME = 0	Desactiva el pin
#define pin__output	0	Salida
#define pin__input	1	Entrada
#define pin__read (PIN_NAME)	PIN_NAME	Lee el pin
#define pin__write (PIN_NAME, VAL)	PIN_NAME = VAL	Escribe el valor del pin
#define pin__setup (PIN_NAME, PIN_MODE)	PIN_NAME ##_s = PIN_MODE	Configura el pin

FUNCIÓN PORT		
#define port__read(PORT_NAME)	PORT ## PORT_NAME	Lee el puerto
#define port__setup(PORT_NAME, VALUE)	TRIS ## PORT_NAME = VALUE	Configura el puerto
#define port__write (PORT_NAME, VALUE)	PORT ## PORT_NAME = VALUE	Escribe el valor del puerto

FUNCIÓN PWM		
#define pwm__setup()	PR2, TMR2, CCPR1L	Configura el pwm
#define pwm__write (duty)	Duty	Calcula el ciclo de trabajo

FUNCIÓN TIME		
#define time__sleep(TIME)	__delay_s(TIME)	Tiempo en segundos
#define time__sleep_ms(TIME)	__delay_ms(TIME)	Tiempo en milisegundos
#define time__sleep_us(TIME)	__delay_us(TIME)	Tiempo en microsegundos

FUNCIÓN UART		
#define uart__setup()	SPBRG	Configura el uart
#define uart__write(data)	TXREG	Escribe datos uart
#define uart__read()	RCREG	Recibe datos uart

- PIC16F886

FUNCIÓN PIN		
NOMBRE	DESCRIPCIÓN	TIPO
#define pin__digital ()	ANSEL = 0	Entradas o Salidas Digitales
#define pin__high (PIN_NAME)	PIN_NAME = 1	Activa el pin
#define pin__low (PIN_NAME)	PIN_NAME = 0	Desactiva el pin
#define pin__output	0	Salida

#define pin__input	1	Entrada
#define pin__read (PIN_NAME)	PIN_NAME	Lee el pin
#define pin__write (PIN_NAME, VAL)	PIN_NAME = VAL	Escribe el valor del pin
#define pin__setup (PIN_NAME, PIN_MODE)	PIN_NAME ##_s = PIN_MODE	Configura el pin

FUNCIÓN PORT		
#define port__read(PORT_NAME)	PORT ## PORT_NAME	Lee el puerto
#define port__setup(PORT_NAME, VALUE)	TRIS ## PORT_NAME = VALUE	Configura el puerto
#define port__write (PORT_NAME, VALUE)	PORT ## PORT_NAME = VALUE	Escribe el valor del puerto

FUNCIÓN PWM		
#define pwm__setup()	PR2, TMR2, CCPR1L	Configura el pwm
#define pwm__write (duty)	Duty	Calcula el ciclo de trabajo

FUNCIÓN TIME		
#define time__sleep(TIME)	__delay_s(TIME)	Tiempo en segundos
#define time__sleep_ms(TIME)	__delay_ms(TIME)	Tiempo en milisegundos
#define time__sleep_us(TIME)	__delay_us(TIME)	Tiempo en microsegundos

FUNCIÓN UART		
#define uart__setup()	SPBRG	Configura el uart
#define uart__write(data)	TXREG	Escribe datos uart
#define uart__read()	RCREG	Recibe datos uart

Posteriormente, procedimos con la elaboración de un archivo de configuración ".TOML" destinado a cada microcontrolador. Este archivo sirve como respaldo para el transcompilador del lenguaje V al C, especialmente diseñado para los microcontroladores PIC16F676, PIC16F84A, PIC16F873A y PIC16F886.

Aquí se encuentran las categorías ADC, PIN, PORT, PWM, UART y TIME, y se procede a configurar cada una de sus funciones correspondientes.

El siguiente paso consistió en realizar pruebas de funcionamiento para desarrollar la estructura de programación en lenguaje C destinada a cada microcontrolador PIC16F676, PIC16F84A, PIC16F873A y PIC16F886.

Posteriormente, se llevó a cabo la programación de cada función estandarizada para los microcontroladores mencionados anteriormente utilizando el lenguaje V. Esto permite al usuario programarlas de manera sencilla.

Se creó un archivo quickref.md, el cual proporciona al usuario una explicación detallada sobre el alcance de los microcontroladores PIC16F676, PIC16F84A, PIC16F873A y PIC16F886, así como instrucciones sobre cómo utilizar cada función en su código fuente Aixt (basado en el lenguaje V) y las funciones compatibles con cada microcontrolador.

A continuación, se proporcionan los respectivos Links del archivo de la guía rápida quickref.md para los microcontroladores PIC16F676, PIC16F84A, PIC16F873A y PIC16F886.

https://github.com/roa53/aixt-PIC16-Cesar-Alejandro-Roa-Acosta/blob/main/docs/Microchip/PIC16/PIC16F676/quickref.md

https://github.com/roa53/aixt-PIC16-Cesar-Alejandro-Roa-Acosta/blob/main/docs/Microchip/PIC16/PIC16F873A/quickref.md

https://github.com/Lpinto96/aixt-PIC16-Luis-Pinto/blob/main/docs/Microchip/PIC16/PIC16F84A/quickref.md

https://github.com/Lpinto96/aixt-PIC16-Luis-Pinto/blob/main/docs/Microchip/PIC16/PIC16F886/quiickref.md

5. CONCLUSIONES

El desarrollo de este proyecto nos ha proporcionado una valiosa comprensión de cómo algo aparentemente simple puede evolucionar hacia un nivel de complejidad considerable. Inicialmente, llevamos a cabo pruebas de funcionamiento en los microcontroladores PIC16F676, PIC16F84A, PIC16F873A y PIC16F886, generando códigos de programación muy básicos que cualquier persona familiarizada con microcontroladores podría comprender. Estas pruebas marcaron el punto de partida para la creación de una API, que culminó con un producto más elaborado y completo.

El proyecto Aixt muestra un enfoque innovador al abordar la integración de microcontroladores de la marca Microchip de la serie PIC16F con un lenguaje de programación moderno inspirado en V. Esta iniciativa demuestra la capacidad de adaptación y evolución dentro del campo de la programación de microcontroladores, permitiendo una migración eficiente entre diferentes lenguajes y plataformas.

La premisa esencial del proyecto Aixt radica en su accesibilidad. El objetivo es que cualquier individuo con conocimientos mínimos de programación pueda comprender los códigos implementados en la API. Esto garantiza que la programación de los microcontroladores sea más accesible y menos intimidante para una amplia gama de usuarios.

Además, se ha llevado a cabo una exhaustiva labor de documentación para facilitar la interacción con la API. Esta documentación se presenta como una herramienta invaluable que permite a cualquier persona, independientemente de su nivel de experiencia, comprender y utilizar la API de manera efectiva. En esencia, la documentación actúa como un puente entre la complejidad inherente de la programación de microcontroladores y la comprensión práctica y aplicada que se busca promover. En conjunto, estas iniciativas se traducen en una experiencia de programación más accesible, intuitiva y eficiente para todos los usuarios involucrados en el desarrollo y la implementación de proyectos con microcontroladores.

REFERENCIAS

S. O. R. F. H. S. Fernando Martínez Santa, «Microcontrollers Programming Framework based on a V-like programming Language,» (IJACSA) International Journal of Advanced Computer Science and Aplications, vol. 13, nº 12, p. 8, 2022.

F. Martínez Santa, «Aixt,» 2022. [En línea]. Available: https://github.com/fermarsan/aixt [Último acceso: 27 Febrero 2023].

F. Martínez Santa, «Sokae Project,» Noviembre 2022. [En línea]. Available: https://gitlab.com/fermarsan/sokae-project. [Último acceso: 4 Marzo 2023].

F. Martinez Santa y S. Orjuela Rivera, «Rust-like Programming Language for Low-resource,» Research India Publications, vol. 17, nº ISSN 0973-5321, pp. 599-614, 2022.

MicroPython, «MicroPython,» [En línea]. Available: https://micropython.org/. [Último acceso: 12 Agosto 2023].

Arduino, «Arduino,» 2023. [En línea]. Available: https://www.arduino.cc/. [Último acceso: 10 Agosto 2023].