PÓS DE SISTEMAS EMBARCADOS

RENÊ DO NASCIMENTO CORRÊA

LINGUAGENS DE PROGRAMAÇÃO: Biblioteca para AVR em C++

São Paulo/SP

2024

Renê do Nascimento Corrêa

LINGUAGENS DE PROGRAMAÇÃO: Biblioteca para AVR em C++

Trabalho de entrega parcial para classe Linguagens de Programação pertencente ao curso de Sistemas Embarcados do Senai-SP.
Prof. Luis Carlos Canno

São Paulo/SP

2024

RESUMO

Este trabalho visa a criação de uma biblioteca na linguagem C/C++ e a implementação da mesma biblioteca em um sistema embarcado. Esta biblioteca simplifica a criação de códigos como o exemplo dado, ao desenvolver projetos que utilizem módulos joystick.

Palavras-chave: Joystick. AVR328P. Biblioteca. Linguagem C/C++.

ABSTRACT

This work seeks to create a library with the language C/C++ and the implementation of the same library in an embedded system. This library simplify the creation of future codes like the example given in this work, while developing projects that utilize joystick modules.

Keywords: Joystick. AVR328P. Library. Language C/C++.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Este trabalho aborda o ciclo de desenvolvimento de uma biblioteca na linguagem de programação C/C++, aplicado a área de sistemas embarcados, nesse caso, resumindo o seu uso aplicado à um microcontrolador. Este trabalho abordará desde a necessidade da criação da biblioteca, seu propósito, como foi desenvolvida e se seu propósito foi atingido ou não.

1.1 SISTEMA ESCOLHIDO

Para esta biblioteca, limitamos sua utilização para o microcontrolador AVR328P Atmel (2015). A biblioteca deste trabalho terá como finalidade auxiliar no desenvolvimento do código para um joystick KY-023 Joy-It (2017), usando o AVR para ler os valores vertical, horizontal e pressionável do joystick. Acompanhado da biblioteca, foi desenvolvido um código de exemplo utilizando leds para acender de acordo com o estado dos sinais do KY-023.

1.2 OBJETIVO A SER ALCANÇADO

A biblioteca a ser desenvolvida tem como objetivo simplificar a adição de um joystick como o do modelo KY-023 em qualquer projeto, nesse caso exclusivo a projetos que utilizem do AVR328P. Desenvolvida em C++, a biblioteca possibilita a criação de objetos do tipo joystick que possui os métodos de leitura vertical, leitura horizontal e leitura selecionável para simplificar o desenvolvimento.

2 REFERENCIAL TEÓRICO

Na linguagem de programação C ou C++, pode-se criar arquivos secundários a programação principal, dos quais realizam algumas funções separadas do código "main", para complementar ele.

Esses arquivos secundários surgiram com alguns objetivos, o principal deles é o de simplificar o código principal e modularizar o mesmo, ou seja, se mais de um código necessita usar um joystick, mas para objetivos distintos, o "módulo"do joystick pode apenas ser adicionado a esses projetos, dessa forma, encurtando o retrabalho de refazer as mesmas funções.

Em C/C++ uma biblioteca pode ser implementada utilizando um arquivo do tipo *header* ou ".h"do qual normalmente contém as dependências da biblioteca e seus métodos, e um arquivo de corpo do tipo ".c"ou ".cpp"para linguagem C ou linguagem C++ respectivamente, onde temos o descritivo do que a biblioteca realmente faz. Ambos os arquivos *header* e corpo carregam o nome dado a biblioteca, mesmo nome utilizado para incluir a mesma biblioteca em qualquer código em C/C++.

3 METODOLOGIA

Neste capítulo, será abordado será abordado as ferramentas utilizadas e o desenvolvimento da biblioteca em si.

3.1 FERRAMENTAS

Para este trabalho foi utilizado o editor de texto visual studio Code e algumas extensões que facilitam no desenvolvimento da biblioteca. Além disso foi utilizado o simulador online Wokwi para a montagem do circuito simulado e teste da biblioteca.

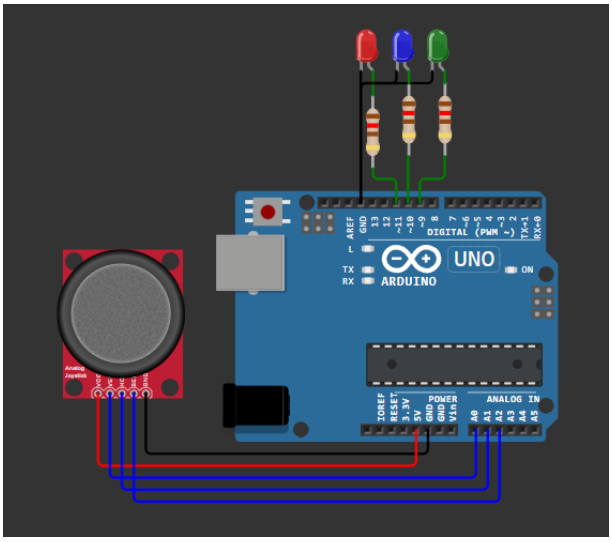
3.1.1 Visual Studio Code

Este editor de texto foi o escolhido por ser versátil com suas extensões e pela simplicidade na hora de desenvolver. Uma das extensões utilizadas foi a do PlatformIO que é voltada para o desenvolvimento com microcontroladores.

3.1.2 Wokwi

No Wokwi foi montado um circuito de exemplo utilizando o Arduino Uno, que possui o AVR328P, um módulo joystick qualquer e alguns LEDs para a demonstração da funcionalidade da biblioteca. Além disso os arquivos de código de exemplo e header e corpo da biblioteca foram recriados no simulador. A Figura 1 representa a montagem do circuito no simulador.

Figura 1 – Circuito montado no Wokwi



Fonte: Elaboração do Autor

3.2 DESENVOLVIMENTO DA BIBLIOTECA

A biblioteca foi desenvolvida para que independente da quantidade de joysticks o projeto use, você pode sempre utilizar a biblioteca para cada joystick. Os arquivos de Header e Corpo da biblioteca podem ser encontrados ao final do trabalho em Anexos.

3.2.1 Header

Essas biblioteca tem as dependências da *stdio.h* (Standard Input Output) para possibilitar utilizar variáveis sem sinal, inteiras de 8-bits; Além da dependência da *avr/io.h* que é o pacote de Inputs e Outputs do AVR, o microcontrolador utilizado. Para o Header, foi criado uma classe do tipo joystick, onde parâmetros como os pinos do joystick são privados e passados no método criador da classe, assim evitando que o usuário os edite por engano. A classe joystick possui três métodos públicos desconsiderando o construtor e o deconstrutor. São esses o readHorizontal, readVertical e readSelectable que servem para retornar os valores lidos pelo joystick para seu encoder horizontal, vertical e seu botão pressionável respectivamente.

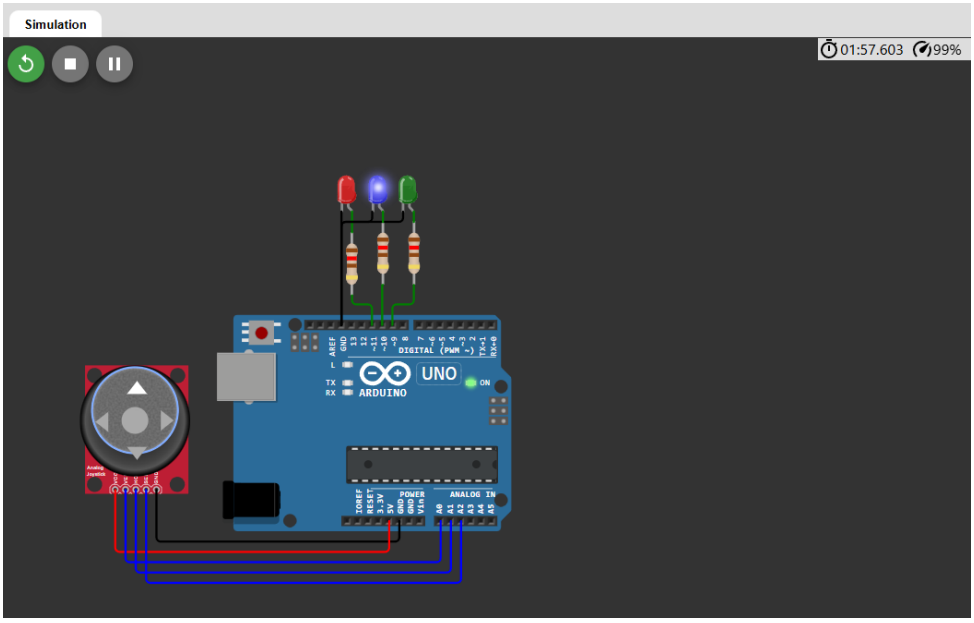
3.2.2 Corpo

O corpo da biblioteca em C++ implementa o construtor da nossa classe, que reatribui as portas e pinos utilizados do AVR pelo nosso joystick, além de inicializar os registradores do ADC do AVR e definir o bit do pressionável como entrada. Os métodos readHorizontal e readVertical convertem os valores ADC dos canais vertical e horizontal do AVR e retornam os valores após o final da conversão analógica-digital. Já o método readSelectable retorna o valor lido na entrada do pino utilizado pelo pressionável do joystick.

4 RESULTADOS

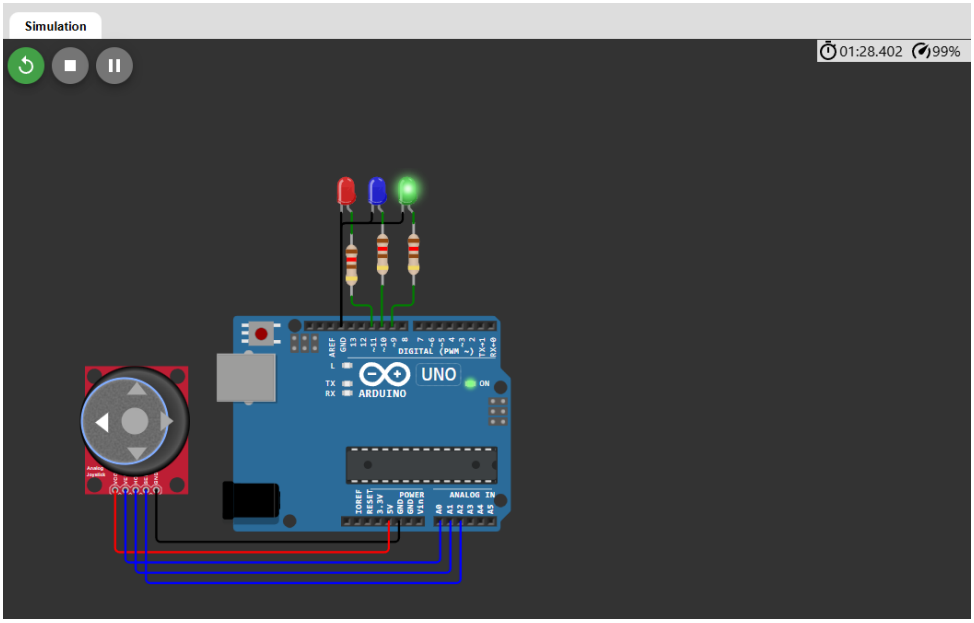
Para analisar os resultados da biblioteca criada, foi utilizado o simulador Wokwi previamente mencionado. Nele foi montado o circuito de exemplo, onde é utilizado um joystick e três leds para representar visualmente o estado de cada uma das três leituras do joystick como pode-ser visto a seguir. O código utilizado nesse exemplo também está presente no anexo do trabalho.

Figura 2 – Leitura Vertical



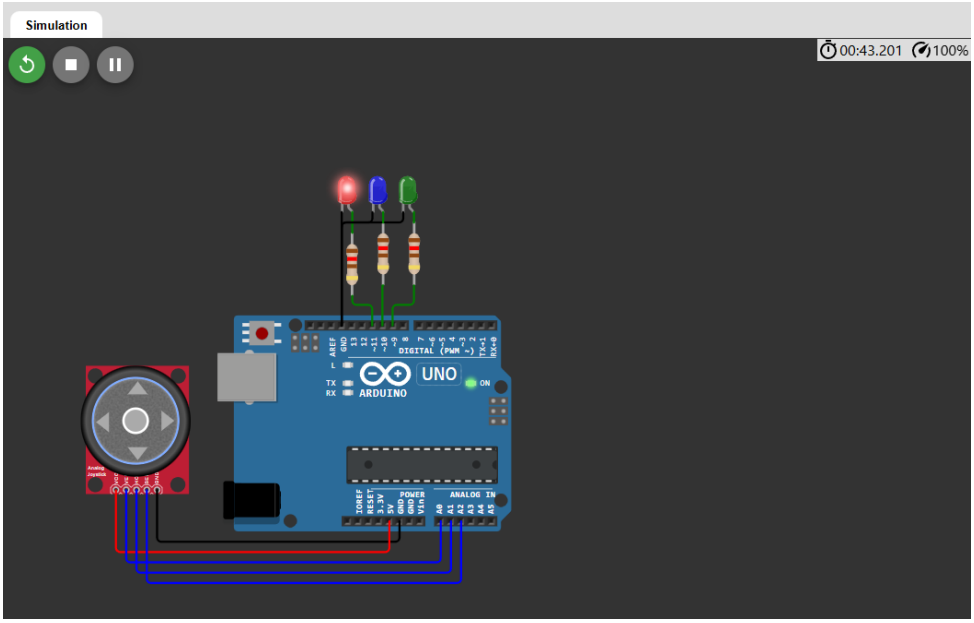
Fonte: Elaboração do Autor

Figura 3 – Leitura Horizontal



Fonte: Elaboração do Autor

Figura 4 – Leitura Pressionável



Fonte: Elaboração do Autor

5 CONSIDERAÇÕES FINAIS

A biblioteca como visto funcionou como esperado, simplificando o código principal que utiliza o joystick, modularizando ao usar uma classe para joysticks, assim cada joystick criado como objeto terá as mesmas funcionalidades da classe principal e mantendo a consistência e a simplicidade, ao longo do código.

REFERÊNCIAS

ATMEL, C. *ATmega328P*. [S.l.], 2015. Disponível em: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.

JOY-IT. *KY-023 Joystick module (XY-Axis)*. [S.l.], 2017. Disponível em: https://naylampmechatronics.com/img/cms/Datasheets/000036%20-%20datasheet%20KY-023-Joy-IT.pdf.

ANEXO A – JOYSTICK.H

```
/**
 *  @ Author: Rene Correa
 *  @ Create Time: 20-10-24 16:02:25
 *  @ Description:
 */

#ifndef JOYSTICK_H
#define JOYSTICK_H

#include <stdio.h>
#include <avr/io.h>

class Joystick{
    private:
        volatile uint8_t* horizontalPort;
        uint8_t horizontalPin;

        volatile uint8_t* verticalPort;
        uint8_t verticalPin;

        volatile uint8_t* selectablePort;
        uint8_t selectablePin;


    public:
        Joystick(volatile uint8_t* verticalPort, uint8_t verticalPin,
                volatile uint8_t* horizontalPort, uint8_t horizontalPin,
                volatile uint8_t* selectablePort, uint8_t selectablePin)
    ;
        ~Joystick();
        uint16_t readHorizontal();
        uint16_t readVertical();
        uint16_t readSelectable();
};

#endif
```

Listing A.1 – Original

ANEXO B – JOYSTICK.CPP

```cpp
/**
 *  @ Author: Rene Correa
 *  @ Create Time: 20-10-24 16:02:20
 *  @ Description:
 */

#include "joystick.h"

Joystick::Joystick(volatile uint8_t* verticalPort, uint8_t verticalPin,
                volatile uint8_t* horizontalPort, uint8_t horizontalPin,
                volatile uint8_t* selectablePort, uint8_t selectablePin)
    {
     this->horizontalPort = horizontalPort;
     this->horizontalPin = horizontalPin;
     this->verticalPort = verticalPort;
     this->verticalPin = verticalPin;
     this->selectablePort = selectablePort;
     this->selectablePin = selectablePin;

     ADMUX |=  (1<<REFS0);   // Tensao de referencia
  ADMUX &=~ (1<<REFS1);   // Tensao de refer ncia
     //AVcc with etxernal capacitor at AREF pin
  ADMUX &=~ (1<<ADLAR);   // Ajuste de organiza ao do resultado

  ADCSRA |= (1<<ADPS0);   // Divisor fator = 128
  ADCSRA |= (1<<ADPS1);   // Divisor fator = 128
  ADCSRA |= (1<<ADPS2);   // Divisor fator = 128
     //Leitura a cada 16MHZ / 128 = 126Khz     8us
     //Leitura a cada 16MHZ / 2 = 8Mhz     125ns
  ADCSRA |= (1<<ADEN);

     *(selectablePort-1) &=~ (1<<selectablePin); //Define o pino
    selectable da porta selectable como INPUT
}

Joystick::~Joystick(){

}

uint16_t Joystick::readHorizontal(){
     uint8_t canal = horizontalPin;
     canal &= 0b00001111;        // Limite de canal em 15
  ADMUX = (ADMUX & 0xF0)|canal;   // Limpa os  ltimos  4 bits de ADMUX 0
    xF0     0b11110000
  ADCSRA |= (1<<ADSC);        // Inicia a conversao
```

```
43   while((ADCSRA)&(1<<ADSC));     // Enquanto se complete a coversao
44   return(ADC);           // Devolve o valor de ADC
45 }
46
47 uint16_t Joystick::readVertical(){
48     uint8_t canal = verticalPin;
49     canal &= 0b00001111;         // Limite de canal em 15
50   ADMUX = (ADMUX & 0xF0)|canal;   // Limpa os  ltimos  4 bits de ADMUX 0
      xF0     0b11110000
51   ADCSRA |= (1<<ADSC);         // Inicia a conversao
52   while((ADCSRA)&(1<<ADSC));     // Enquanto se complete a coversao
53   return(ADC);
54 }
55
56 uint16_t Joystick::readSelectable(){
57   return *(selectablePort-2) & (1<<selectablePin); //retorna valor lido
      no selectablePin
58 }
```

Listing B.1 – Original

ANEXO C – CÓDIGO DE EXEMPLO

```
/**
 *  @ Author: Rene Correa
 *  @ Create Time: 20-10-24 15:53:33
 *  @ Description:
 */

#ifdef F_CPU
#undef F_CPU
#define F_CPU 16000000UL
#endif

#include <avr/io.h>
#include "joystick.h"

#define Led1 (1<<PORTB1)
#define Led2 (1<<PORTB2)
#define Led3 (1<<PORTB3)

Joystick joy1 = Joystick(&PORTC, PORTC0, &PORTC, PORTC1, &PORTC, PORTC2)
    ;

int main(){
    DDRB |= Led1 | Led2 | Led3;
    PORTB &=~ Led1 &~ Led2 &~ Led3;

    while(1){
        if(joy1.readHorizontal() >512) PORTB |= Led1;
        else PORTB &=~ Led1;

        if(joy1.readVertical() >512) PORTB |= Led2;
        else PORTB &=~ Led2;

        if(joy1.readSelectable()) PORTB |= Led3;
        else PORTB &=~ Led3;
    }
    return 0;
}
```

Listing C.1 – Original