IFSC
Engenharia de Telecomunicações
Sistemas Embarcados
Professor: Roberto de Matos
Aluna: Schaiana Sonaglio

ATIVIDADE 2

1. Análises prática e conceitual

Versão 1.0:

Nesta implementação, o construtor da classe GPIO é extenso, com partes de códigos redundantes, por exemplo, todas as condições ‘case’ tem uma porção de código que configuram a direção do pino, isso pode ser otimizado, como é feito na versão 1.2. Em todos os métodos é feito o cálculo de posição do bit do pino, isso também poderia ser otimizado sendo feito o cálculo somente uma vez no construtor.

O método ‘clear’ é mais custoso que o método ‘set’, pois o ‘clear’ chama o método ‘set’ passando como parâmetro o ‘val’ igual a zero, para resolver isso seria necessário repetir a lógica do ‘set’ no método ‘clear’, que não teria mais a chamada de função adicional.

Versão 1.2:

Nesta versão, a posição do bit do pino foi calculada no construtor, foram criados ponteiros que apontam para os registradores referentes ao pino, direção e estado, deste modo, todos os outros métodos que precisam dessas informações não tiveram que ficar calculando o bit e a porta em toda a chamada de método.

Versão 1.3:

Nesta versão, foi implementada a classe GPIO_Port, que possui as características da porta referente ao pino utilizado; essa classe também tem os métodos para a manipulação dos registradores. Uma desvantagem é que todas as funções da classe GPIO terão uma chamada de função a mais, que é a chamada de função da classe GPIO_Port.

Versão 2.0:

Nesta versão, foram utilizadas arrays constantes para fazer a indexação dos bits dos pinos e das portas, e os métodos ‘get’, ‘set’, ‘clear’ e ‘toggle’ são genéricos, ou seja, eles são iguais independente de qual é a porta sendo utilizada, com isso, a extensão da classe GPIO_Port para funcionar com todos os pinos do Arduino é realizada de forma simplificada, somente sendo necessário configurar os arrays de pinos e portas. Caso fosse necessário

expandir as versões anteriores para abranger todos os pinos do Arduino, os construtores teriam um tamanho muito grande.

Discussão da modificação da implementação:

Nesta versão, foi adicionada a macro PROGMEM, que armazena uma constante na memória Flash ao invés de armazenar na memória RAM, com isso se economiza espaço da RAM que normalmente é reduzida em microcontroladores.

Na Tabela 1, é mostrado um comparativo entre o tamanho das classes GPIO de cada versão disponibilizada pelo professor. O tamanho do ‘main’ foi desconsiderado e no tamanho do ‘clear’ está incluso o tamanho do ‘set’.

Tabela 1 - Comparativo do tamanho das classes GPIO

Versão	V1.0	V1.2	V1.3	V2.0
Total (bytes)	1362	820	770	494
Construtor (bytes)	790	604	424	130
Clear (bytes)	526	146	152	170

Fonte: elaboração própria