UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

Proposta de Curso Prático em Algoritmos e Programação Computadores Utilizando Placa Intel ® Galileo

Luiz Fernando de Andrade Gadêlha

Relatório submetido ao Departamento de Engenharia
Elétrica como requisito parcial para obtenção
do grau de Engenheiro Mecatrônico

Banca Examinadora

Prof. Alexandre Zaghetto, CIC/UnB
Orientador

Dedicatória

Dedico este trabalho em primeiro lugar a Deus, por todas bençãos que me fizeram continuar e todas dificuldades que me fizeram crescer. Dedico este trabalho também à minha família, que esteve comigo em todos momentos da minha formação.

Luiz Fernando de Andrade Gadêlha

Agradecimentos

Agradeço a meus colegas de curso, projetos e ao meu professor orientador por este trabalho

Luiz Fernando de Andrade Gadêlha

RESUMO

Este trabalho tem como objetivo propor um curso prático em Algoritmos e Programação de Computadores Utilizando a placa Intel® Galileo voltada para alunos de graduação dos curso de engenharia mecatrônica, elétrica e de computação. Tal proposta se fundamenta na noção de que a inclusão de práticas laboratoriais . A disciplina tem como base de desenvolvimento o microcontrolador Galileo e conceitos de eletrônica de todos níveis.

ABSTRACT

This work aims to propose a discipline aimed at undergraduate students for learning development of embedded circuits. This proposal is based on the growing need and popularization of embedded circuits geared to various purposes, such as home automation, building and industrial. The course has the development of basic microcontroller Galileo and electronics concepts of all levels.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

A	Área	[m^2]
Cp	Calor especifico a pressão constante	[kJ/kg.K]
h	Entalpia especifica	[kJ/kg]
$\dot{m}$	Vazão mássica	[kg/s]
T	Temperatura	[°C]
U	Coeficiente global de transferência de calor	[W/m^2.K]

Símbolos Gregos

α	Difusividade térmica	[m^2/s]
Δ	Variação entre duas grandezas similares	
ρ	Densidade	[m^3/kg]

Grupos Adimensionais

Nu	Número de Nusselt
Re	Número de Reynolds

Subscritos

amb	ambiente
ext	externo
in	entrada
ex	saída

Sobrescritos

$\cdot$	Variação temporal
$-$	Valor médio

Siglas

ABNT	Associação Brasileira de Normas Técnicas

Capítulo 1

Introdução

A economia mundial está passando por uma grande revolução neste século. As bases econômicas de muitos países, outrora baseadas em *extração de matérias-primas* e *indústrias de transformação* de tais matérias primas, são agora baseadas em *conhecimento e transminssão de informação* [2]. O desenvolvimento tecnológico da Ciência da Computação é a principal responsável por tal revolução e todas Engenharias e Ciências Exatas são, direta ou indiretamente, influenciados por ela. Neste contexto, são fundamentais os conhecimentos e habilidades relacionadas a Ciência da Computação para o desenvolvimento de todas as Engenharias. Em especial, é importante a revisão da sua forma de ensino e aprendizagem para a realidade na qual vivemos atualmente.[3].

A educação como a conhecemos atualmente foi idealizada na Prússia, no final do século XVII. Tal modelo educacional é chamada de *aprendizagem centrada no professor* ou *aprendizagem passiva*. No modelo de *aprendizagem passiva*, os estudantes são meros receptores do conhecimento oriundo do professor. Tal modelo se adequou bem as necessidades econômicas da época. Nessa época, era exigido do trabalhador habilidades ligadas a pura repetição e obediência[4].

Hoje em dia, principalmente por causa da revolução engredada pela computação, boa parte das universidades no mundo já começaram a modificar seus paradigmas educacionais realizando uma transição da *aprendizagem passiva*) para o modelo de *aprendizagem ativa*) ou *aprendizagem centrada no aluno*). Nesse modelo, o estudante é o principal responsável por sua aprendizagem e o professor é o orientador das experiências de ensino. Com esse modelo, tem-se conseguido obter altos índices de paradigmas ligados a criatividade, liderança, trabalho em equipe, gerenciamento e auto-gerenciamento além de um aumento substancial na motivação dos estudantes, visto que eles podem se apropriar verdadeiramente de seu processo de aprendizagem além terem se mostrado próprios para a aprendizagem e ensino de conceitos ligados a computação[5].

A realidade da educação brasileira, no entanto, não tem acompanhado as tendências supracitadas. Técnicas eficientes de ensino de conhecimentos relacionados a ensino de Ciências Exatas e Engenharia são parte estratégica para o desenvolvimento de qualque país. Entretanto, as mudanças nos cursos de engenharia, no Brasil, em geral têm sido relacionadas a simples adição ou supressão de conteúdos,mas não uma revisão profunda das bases de ensino, levando em consideração as transformações atuais[6].

Segundo o jornal *A Gazeta do Povo*, a taxa de evasão no curso de engenharia no Brasil é de aproximadamente 57% [7] e em geral a evasão ocorre nas partes iniciais dos cursos, onde os alunos têm seu primeiro contato com computação . Pode-se afimar, tendo em vista essa estatística, que o paradigma educacional atual é o maior responsável pelo grande déficit de engenheiros qualificados no Brasil.

Com relação ao ensino de habilidades e conceitos de computação básica, no mundo se observa - não apenas no Brasil - que os estudantes usualmente têm grandes dificuldades de aprendizagem, que o conhecimento e aprendizagem dos alunos tende a se estagnar nos níveis mais rasos de entendimento, de forma que os conhecimentos não são interconectados, mas são apenas específicos ao contexto estudado[8] além de, muitas vezes, se sentirem desmotivados devido a fragmentação do conhecimento nas disciplinas[6, 5]. Tais problemas de aprendizagem também se mostram presentes nos profissionais que saem das faculdades. Boa parte dos profissionais, no Brasil, possuem formação deficiente. Não tem capacidade plenamente desenvolvida para serem *aprendizes-estudantes autônomos*. Tal habilidade é essencial para terem sucesso na econômia mundial atual, que é centrada em conhecimento e informação[3].

Para realizar a transição entre o modelo *aprendizagem-passiva* para o modelo *aprendizagem-ativa* , muitas universidades já se utilizam de placas eletrônicas com microcontroladores como Arduino e similares[9]. Nas referências citadas, o ensino de computação básica aliada a projetos práticos tem alcançados grande aumento nos índices acadêmicos dos alunos e diminuição nas taxas de evasão.

Este trabalho tem como objetivo primário a proposta de um curso prático para a disciplina *Algoritmos e Programação de Computadores* utilizando a placa de desenvolvimento *Intel® Galileo* com dinâmicas pedagógicas próprias do paradigma de *aprendizagem-ativa* de formar a atacar os problemas elencados anteriormentee de forma a propor uma disciplina factível a realidade da Universidade de Brasília (UnB).

1.1 Objetivo

O objetivo deste trabalho é propor para a disciplina de *Algoritmos e Programação de Computadores* um modelo de curso prático de programação.

Busca-se, por meio desta proposta de curso, oferecer um modelo pedagógico mais eficiente para o ensino de programação na linguagem C e o básico de circuitos embarcados. Tal proposta tem como motivação buscar tratar do problema da usual baixa profundidade na aprendizagem de programação e o problema da desmotivação dos alunos, causada por diversos fatores, dentre eles a separação artificial entre as disciplinas.

Neste trabalho são expostos XXXXXXXXXXX planos de aula prática que contêm uma explicação aprofundada de todos conceitos tratados no laboratório - tanto os conceitos diretamente ligados a programação na linguagem C, quanto os conceitos ligados a circuitos eletrônicos. Em todas descrições do plano de laboratório são também sugeridas formas de organização pedagógicas

para ampliar e aprofundar a aprendizagem dos alunos.

1.2 Apresentação do manuscrito

No capítulo 2 a placa Intel® Galileo é apresentada e todos conceitos relacionados a seus componentes são explicados. São apresentados também as teorias educacionais que dão suporte ao método de *educação ativa*. Por fim, ainda no capítulo 2, é feita a proposta de reformulação da disciplina *Algoritmos e Programação de Computadores* apresentando um possível plano de ensino para tal disciplina.

Em seguida, o capítulo 3 apresenta as práticas laboratórias planejadas. Nessas práticas, é exposta a lista de conceitos de programação e eletrônica básica tratados, os materiais necessários, os esquemáticos dos circuitos e os códigos a serem utilizados com a placa Intel® Galileo.

Capítulo 2

Revisão Bibliográfica

O objetivo deste capítulo é explicitar todos conceitos relevantes deste trabalho relativos a ensino e aprendizagem de Ciências Exatas e Engenharia e a estrutura detalhada da placa Intel Galileu. Esses estudos servirão de base para a proposição um modelo de aula de programação básica para estudantes de graduação levando em conta os paradigmas de educação mais eficientes dentre os elencados.

2.1 Teorias Pedagógicas

Este trabalho tem como proposta a atualização dos curso de *Algoritmos e Programação de Computadores* seguindo as transformações atuais que apontam para métodos de ensino focados em *educação ativa*, Nesta seção, são apresentados:

- Estudo de currículos relacionadas a *Algoritmos e Programação de Computadores*
- Estudo da teoria pedagógica de aprendizagem ativa focando na teoria de *aprendizagem por masterização*
- Estudo da metodologia SCRUM para desenvolvimento agíl de projetos

2.1.1 Estudo de currículos relacionados a *Algoritmos e Programação de Computadores*

Fez-se, para os estudos de currículo, uma breve pesquisa das 5 melhores universidade mundiais e as cinco melhores universidades no Brasil. Os resultados de tal estudo são mostrados na tabela 2.1. A universidade de Brasília (UnB) está, atualmente, entre as 800 melhores universidades e é a quinta melhor universidade brasileira.

Os índices utilizados pela instituição de pesquisa *timeshighereducation* são os seguintes:

1. Score Ensino

2. Score Panorama Internacional

3. Score Impacto na Indústria

4. Score Pesquisa

5. Score Citação

Como dito em [10], pode-se concluir, sem perda de exatidão, que tais índices são totalmente dependentes e proporcionais à qualidade de ensino nas universidade em questão.

UNIVERSIDADE	PAÍS	POSIÇÃO	SCORE MÉDIO	SCORE ENSINO	SCORE PANORAMA INTERNACIONAL	SCORE IMPACTO NA INDÚSTRIA	SCORE PESQUISA	SCORE CITAÇÃO
Universidade de Brasilia	Brasil	800	20	21.9	24.1	20	7.4	10.4
Universidade Pontificia Católica do Rio de Janeiro	Brasil	600	20	24.5	31.3	100	24.1	23.1
Universidade Federal do Rio de Janeiro	Brasil	600	20	32.4	25.1	42.9	19.7	18.3
Universidade de Campinas	Brasil	400	20	44.6	21.1	49.4	42.3	22.6
Universidade de São Paulo	Brasil	300	20	59.1	25.3	30.5	57.1	20.4
University of Chicago	EUA	10	87.9	85.7	65	36.6	88.9	99.2
ETH Zurich – Swiss Federal Institute of Technology Zurich	Suiça	9	88.3	77	97.9	80	95	91.1
Imperial College London	Reino Unido	8	89.1	83.3	96	53.7	88.5	96.7
Princeton University	EUA	7	90.1	85.1	78.5	52.1	91.9	99.3
Harvard University	EUA	6	91.6	83.6	77.2	45.2	99	99.8
Massachusetts Institute of Technology	EUA	5	92	89.4	84	95.4	88.6	99.7
University of Cambridge	Reino Unido	4	92.8	88.2	91.5	55	96.7	97
Stanford University	EUA	3	93.9	92.5	76.3	63.3	96.2	99.9
University of Oxford	Reino Unido	2	94.2	86.5	94.4	73.1	98.9	98.8
California Institute of Technology	EUA	1	95.2	95.6	64	97.8	97.6	99.8

Figura 2.1: Tabela com scores de avaliação das 5 melhores universidades do mundo e das 5 melhores universidades brasileiras

Pesquisou-se o currículo da UnB e o currículo da universidade *California Institute of Tecnology* para disciplinas com ementa similar ao de *Algoritmos e Programação de Computadores*. O resultado de tal pesquisa é mostrado na Figura 2.2.

Fez, também, uma pesquisa mais aprofundada do conteúdo programático de tais disciplinas. O resultado de tal pesquisa é mostrado na Figura 2.3.

Universidade - Curso	Disciplina
CALTECH - Computer Science	Introduction to Computer Programming
CALTECH - Computing and Mathematical Sciences	Introduction to Matlab and Mathematica
UnB - Engenharia Mecatrônica	Algoritmos e Programação de Computadores - UnB
UnB - Engenharia de Computação	Algoritmos e Programação de Computadores - UnB
UnB - Engenharia de Produção	INTRODUCAO A CIÊNCIA DA COMPUTACÃO - UnB
UnB - Ciências da Computação	INTRODUCAO A CIÊNCIA DA COMPUTAÇÃO - UnB
UnB - Engenharia de Redes	Computação para Engenharia - UnB
UnB - Engenharia Eletrônica	Algoritmos e Programação de Computadores - UnB
UnB - Engenharia de Software	Algoritmos e Programação de Computadores - UnB
UnB - Engenharia AeroEspacial	Algoritmos e Programação de Computadores - UnB
UnB - Engenharia Automotiva	Algoritmos e Programação de Computadores - UnB
UnB - Engenharia de Energia	Algoritmos e Programação de Computadores - UnB

Figura 2.2: Tabela listando as disciplinas na UnB e na *California Insitute of Tecnology* que possuem currículo similar a disciplina *Algoritmos e Programação de Computadores*

CONTEÚDO	JÁ ENSINADO COM APRENDIZAGEM ATIVA?	PODE SER ENSINADO COM GALILEU?	DESCRIÇÃO
Python	No	Yes	Python
n, solving nonlinear equations, fast Fourier transform, and ODE solvers	No	No	n, solving nonlinear equations, fast Fourier transform, and ODE solvers
vectorization; scripts, and functions; file i/o; arrays, structures, and strings	No	No	ectorization; scripts, and functions; file i/o; arrays, structures, and strings
debugging; help interface; basic linear algebra;	No	No	debugging; help interface; basic linear algebra;
. Matlab: basic syntax and development	No	No	. Matlab: basic syntax and development
8. Registros Composição de structs. Arrays e structs. Structs em arquivos.	No	Yes	8. Registros Composição de structs. Arrays e structs. Structs em arquivos.
7. Vetores Tipo Array. Pesquisa em Vetores. Ordenação de Vetores. Strings. Arrays multidimensional.	No	Yes	7. Vetores Tipo Array. Pesquisa em Vetores. Ordenação de Vetores. Strings. Arrays multidimensional.
6. Arquivos Arquivos text. Manipulação de arquivos.	No	Yes	6. Arquivos Arquivos text. Manipulação de arquivos.
5. Ponteiros e Funções Ponteiros. Funções.	No	Yes	5. Ponteiros e Funções Ponteiros. Funções.
4. Estruturas de Repetição Comando Do-While. Comando While. Comando For.	No	Yes	4. Estruturas de Repetição Comando Do-While. Comando While. Comando For.
3. Estruturas Condicionais Comando IF. Comando Switch.	No	Yes	3. Estruturas Condicionais Comando IF. Comando Switch.
Linguagem C Estrutura de um programa em C. Variáveis e tipos básicos. Constantes Operadores, Funçoes e Expressões. Atribuição. Entrada	No	Yes	Linguagem C Estrutura de um programa em C. Variáveis e tipos básicos. Constantes Operadores, Funçoes e Expressões. Atribuição. Entrada e Saída de dados. Operador

Figura 2.3: Tabela listando parte dos conteúdos tratados pelas disciplinas na *California Insitute of Tecnology* e na UnB e levantanto a questão da metodologia de ensino ser uma *metodologia ativa* e se tal conteúdo pode ser ensinado usando a placa Galileo

Para cada conteúdo identificado, procurou-se saber também se ele era tratado de seguindo algum paradigma moderno de *Educação Ativa* e, se pela natureza do conteúdo, seria possível utilizar a placa Galileo no processo de *ensino-aprendizagem*.

A proporção de conteúdos que poderiam ser ensinados com a placa Galileo é mostrada na Figura 2.4.

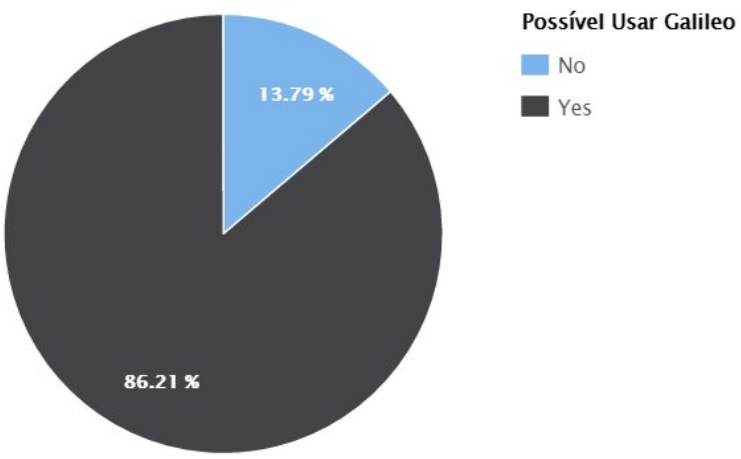

Figura 2.4: Percentagem dos conteúdos tratados na UnB e *California Institute of Tecnology* que podem ser ensinados utilizando a placa Galileo

Pode-se concluir, pelas informações colhidas nesta seção, que a mudança currícular da disciplina *Algoritmos e Programação de Computadores* e disciplinas similares em outras faculdades é possível. Tal mudança curricular em direção à aprendizagem ativa, além de trazer grandes benefícios à qualidade do ensino [11][12], também pode ajudar a formação de novos profissionais alinhados com as competências mais exigidas no século XXI, como proatividade e independência [13].

2.1.2 Estudo da teoria pedagógica de aprendizagem ativa focando na teoria de *aprendizagem por masterização*

Esta seção trata do paradigma de aprendizagem por masterização e da Taxonomia de definição de objetivos educacionais propostas por Bejamim S. Bloom. Além disso, o é descrito um aplicativo, desenvolvido do ambiente de programação QT, cujo objetivo é auxiliar a definição de objetivos educacionais de forma mais clara e precisa.

2.1.2.1 Taxonomia dos objetivos educacionais - Taxonomia de Bloom

A taxonomia dos objetivos educacionais (Taxonomia de Bloom) é um modelo de estruturação de objetivos educacionais segundo níveis hierárquicos e crescentes de níveis de aprendizagem. A taxonomia de Bloom foi planejada para três domínios distintos de aplicação [14]. São eles:

- Domínio cognitivo, cuja abrangência é a aprendizagem intelectual.
- Domínio afetivo, cuja abrangência está ligado aos aspectos de sensibilização, socialização e gradação de valores
- Domínio psicomotor, cuja abrangência está ligada a habilidades de execução de tarefas que envolvem o aparelho motor.

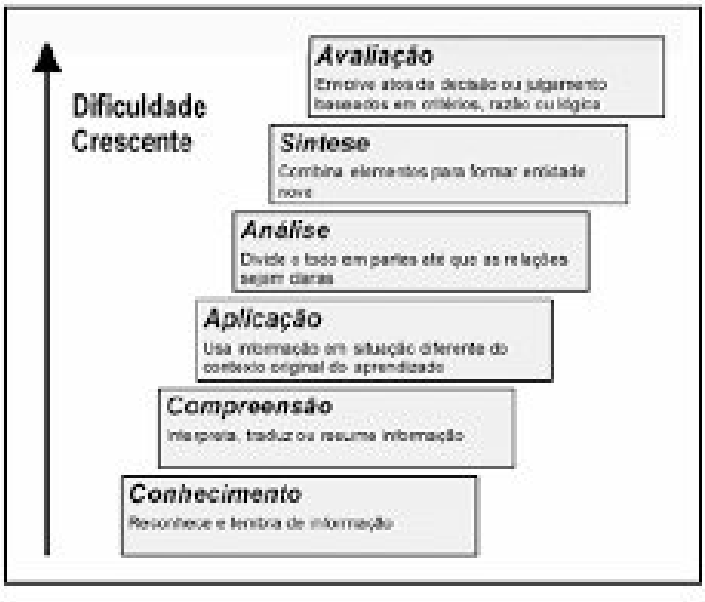

Figura 2.5: Níveis de aprendizagem no domínio cognitivo

Cada um destes domínios tem diversos níveis de profundidade de aprendizado. Por isso a classificação de Bloom é denominada hierarquia: cada nível é mais complexo e mais específico que o anterior como mostrado na Figura 2.5.

Os níveis cognitivos de Bloom são usados para definir *objetivos educacionais* de forma precisa. A Figura 2.6 mostra os verbos a serem usados para se definir objetivos educacionais para cada dos níveis citados de acordo com a Taxonomia de Bloom.

CONHECIMENTO	COMPREENSÃO	APLICAÇÃO	ANÁLISE	SÍNTESE	AVALIAÇÃO
Apontar	Descrever	Aplicar	Analisar	Armar	Ajuizar
Arrolar	Discutir	Demonstrar	Calcular	Articular	Apreciar
Definir	Esclarecer	Dramatizar	Classificar	Compor	Avaliar
Enunciar	Examinar	Empregar	Comparar	Constituir	Eliminar
Inscrever	Explicar	Ilustrar	Contrastar	Coordenar	Escolher
Marcar	Expressar	Interpretar	Criticar	Criar	Estimar
Recordar	Identificar	Inventariar	Debater	Dirigir	Julgar
Registrar	Localizar	Manipular	Diferenciar	Reunir	Ordenar
Relatar	Narrar	Praticar	Distinguir	Formular	Preferir
Repetir	Reafirmar	Traçar	Examinar	Organizar	Selecionar
Sublinhar	Traduzir	Usar	Provar	Planejar	Taxar
Nomear	Transcrever		Investigar	Prestar	Validar
			Experimentar	Propor	Valorizar
				Esquematizar	

Figura 2.6: Verbos de descrição de objetivos educacionais a serem usados em cada nível cognitivo

Um objetivo educacional definido segundo a taxonomia de Bloom segue o seguinte padrão:

"O(s) aluno(s) será(ão) capaz(es) de (**Verbo próprio do nível cognitivo considerado**) (**Conteúdo estudado**).

Um exemplo de aplicação para o caso de primeiro nível cognitivo, no estudado de equações diferenciais:

"O aluno será capaz de **definir** o que são **equações diferenciais**."

Para este trabalho, foi criado um aplicativo na plataforma de programação QT® para facilitar a definição de objetivos educacionais de acordo com a Taxonomia de Bloom.

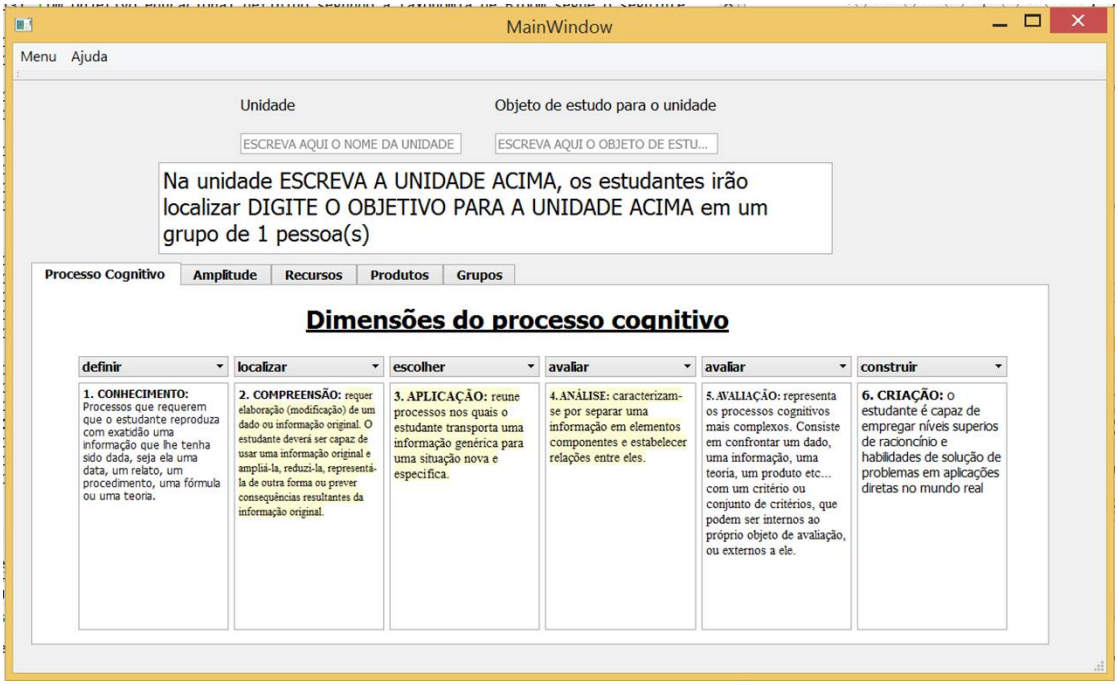

Figura 2.7: Tela inicial do aplicativo desenvolvido em QT para definir os objetivos educaionais

A Figura 2.7 mostra a tela inicial do aplicativo desenvolvido. Na região de inserção de texto Descrita por **Unidade**, deve-se escrever de qual unidade de ensino se está tratando. Na região textual central, marcada na Figura 2.8, é mostrado resultado do objetivo educacional definido.

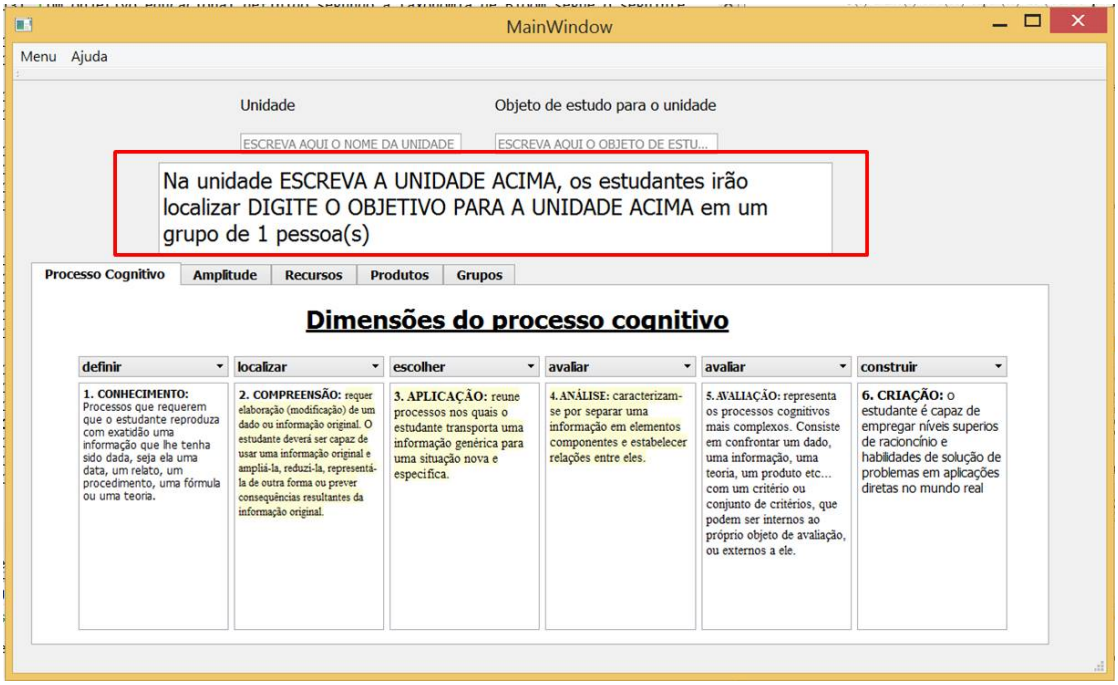

Figura 2.8: Tela inicial do aplicativo desenvolvido em QT para definir os objetivos educacionais, retângulo vermelho marcando a região onde o objetivo educacional definido é mostrado

A Figura 2.9 mostra o resultado, na região textual central , da definição do **Unidade**.

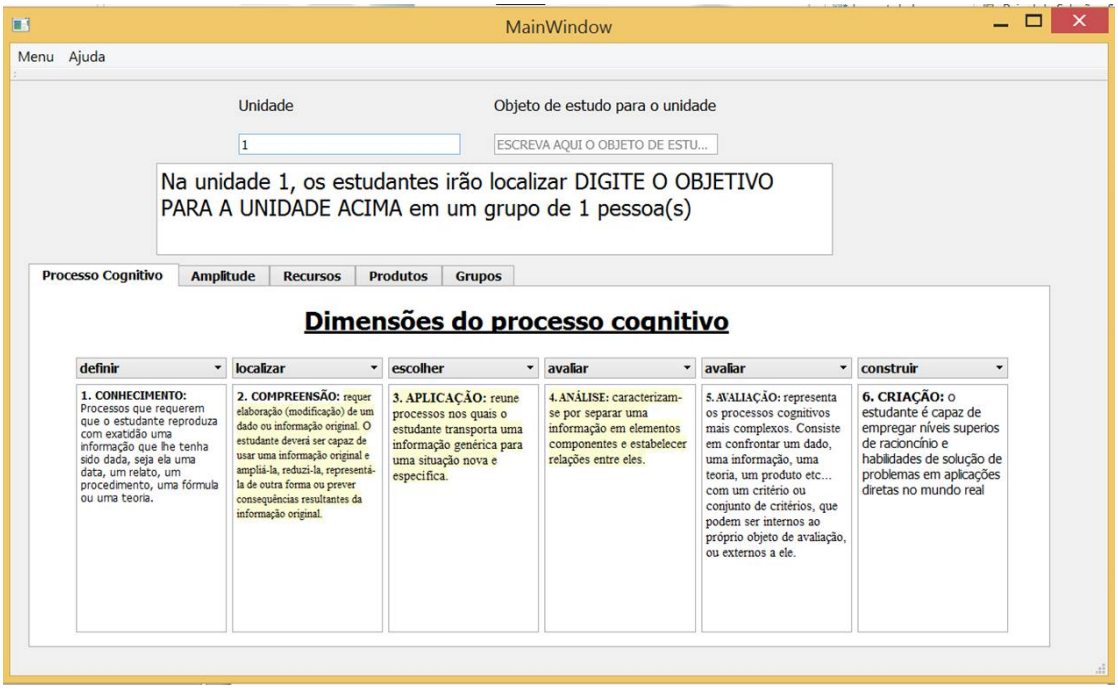

Figura 2.9: Definindo uma unidade de ensino

A Figura 2.10 mostra o resultado, na região textual central , da definição do **Objeto de estudo** específico para a **Unidade** definida.

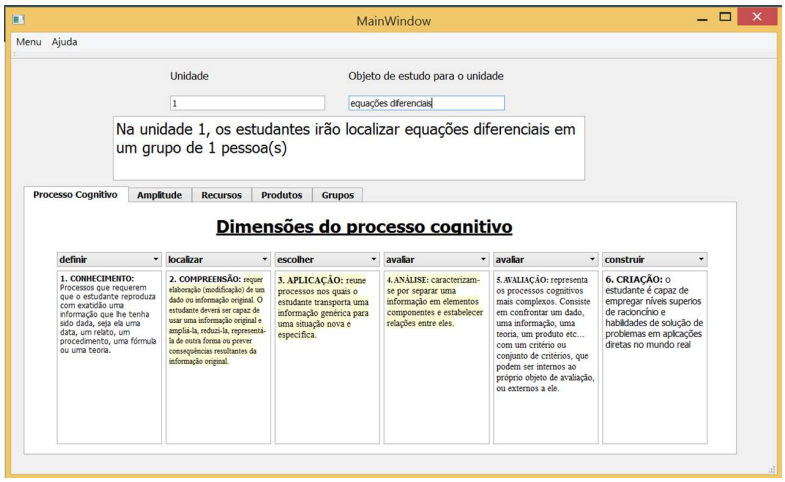

Figura 2.10: Definindo um objetivo de estudo para a unidade de estudo

A Figura 2.11 mostra os níveis do processos cognitivos da Taxonomia de Bloom.

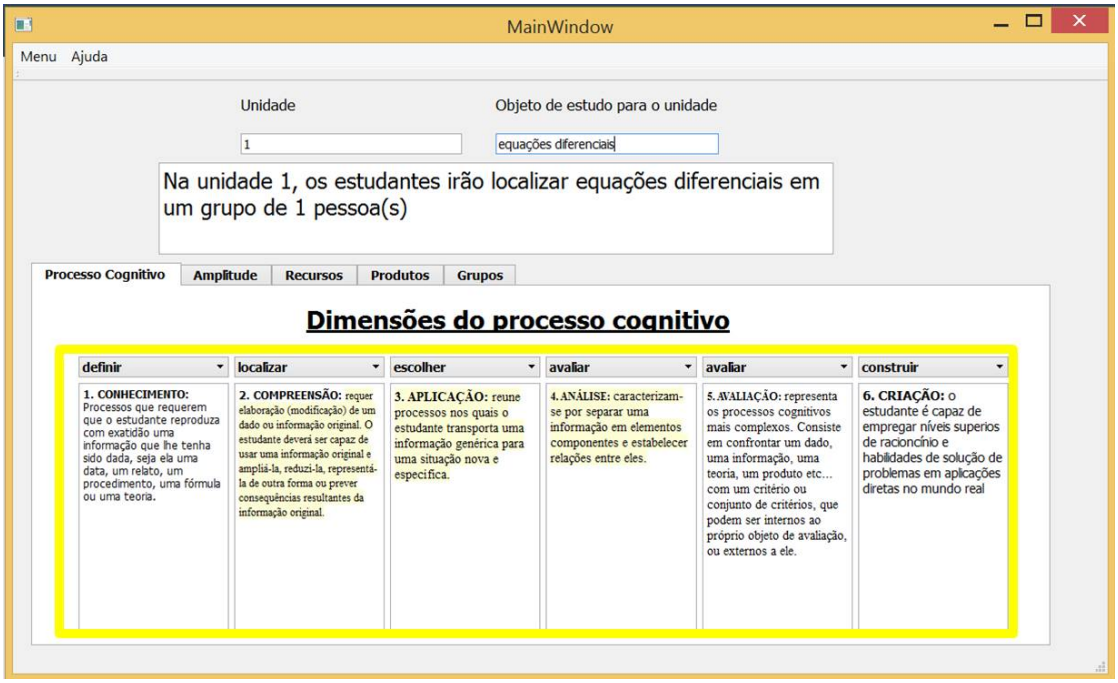

Figura 2.11: Retângulo amarelo destacando os níveis cognitivos

Cada *Combo-box* mostrado possui os verbos associados a cada nível cognitivo. Quando um verbo é selecioando, o objetivo educacional exposto na região retangular é alterado, como exposto na Figura 2.12.

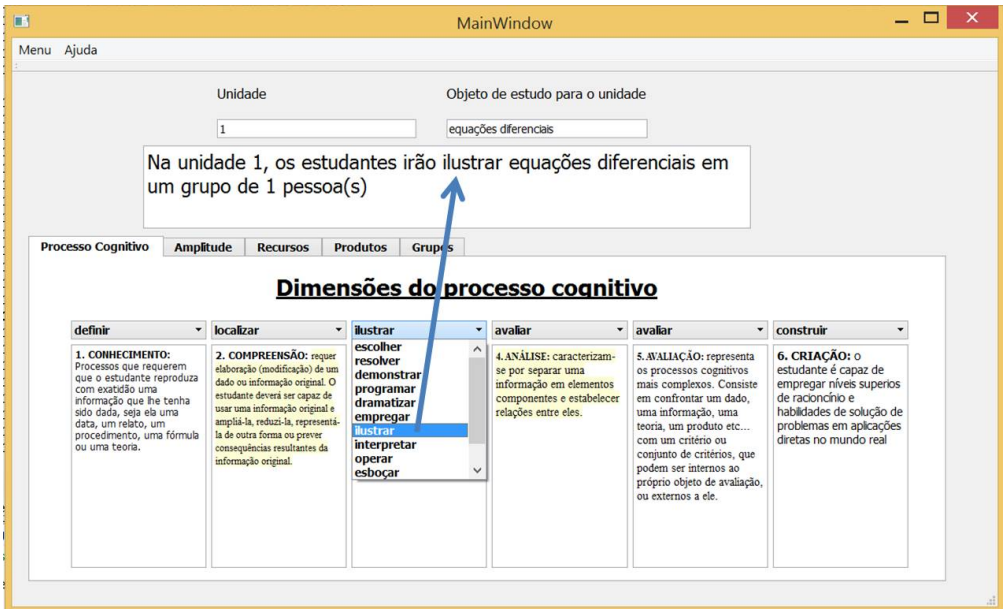

Figura 2.12: Selecionando um verbo de um nível cognitivo

Na aba *Amplitude*, são selecionados modificadores para especificação do objetivo educacional. A Figura 2.13 mostra o uso do modificador para alterar o objetivo educacional adicionando o modificador "o objetivo geral".

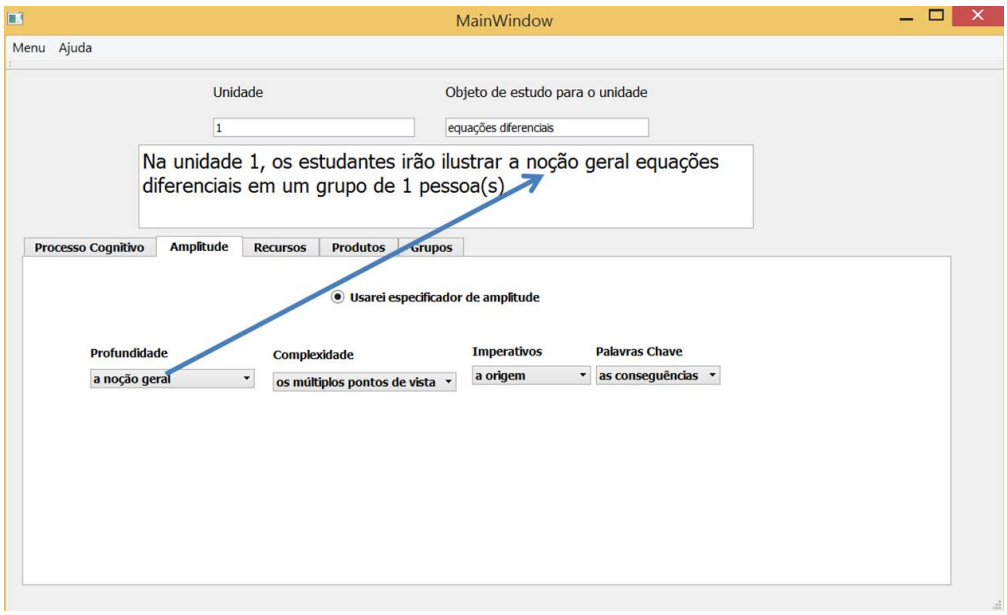

Figura 2.13: Selecionando a amplitude do estudo a ser realizado

Na aba *Recursos*, são especificados recursos com os quais os estudantes realizarão atividades em direção à masterização do conteúdo. A Figura **??** mostra o uso do especificador para alterar o

objetivo educacional adicionando o recurso "livro".

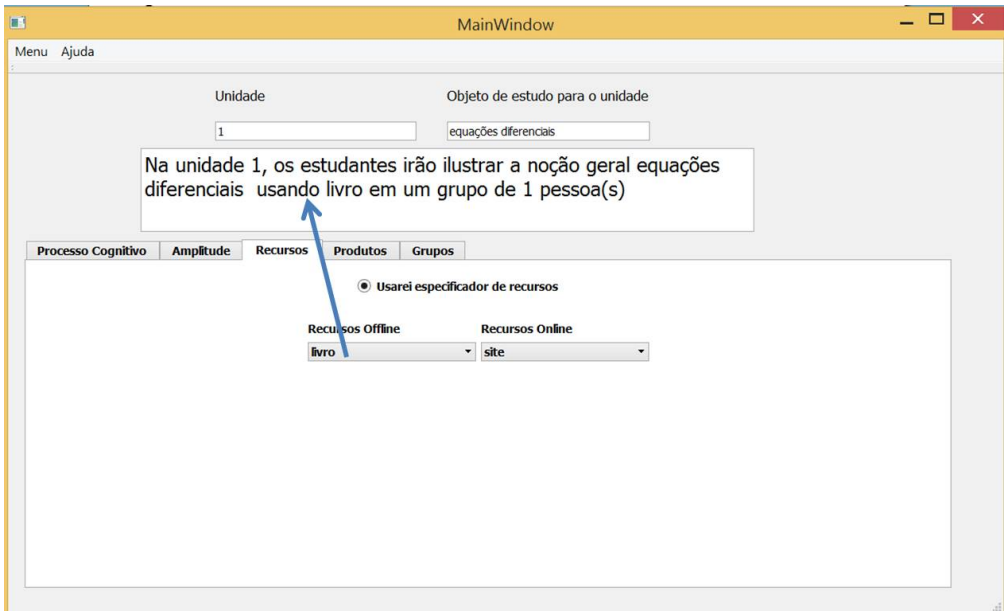

Figura 2.14: Selecionando o recursos a ser utilizado para alcançar o obejtivo educacional especificado

Na aba *Produtos*, é especificado o que os alunos desenvolverão para atingir o objetivo educacional especificado. A Figura 2.15 mostra o uso do especificador de produto a ser desenvolvido para alterar o objetivo educacional adicionando o objetivo de produzir um "artigo".

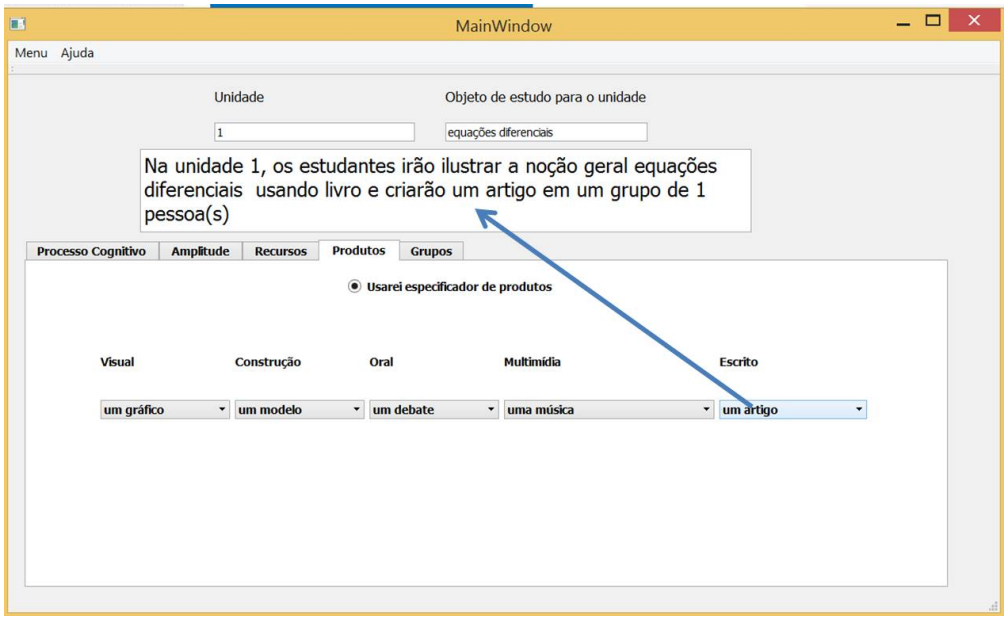

Figura 2.15: Selecionando um produto a ser desenvolvido como atividade de masterização

Finalmente, na aba *Grupos*, é especificado a quantidade de alunos por grupo que realizarão, conjuntamente as atividades de masterização. A Figura 2.16 mostra a alteração da quantidade de pessoas por grupo para, por exemplo, 4 pessoas.

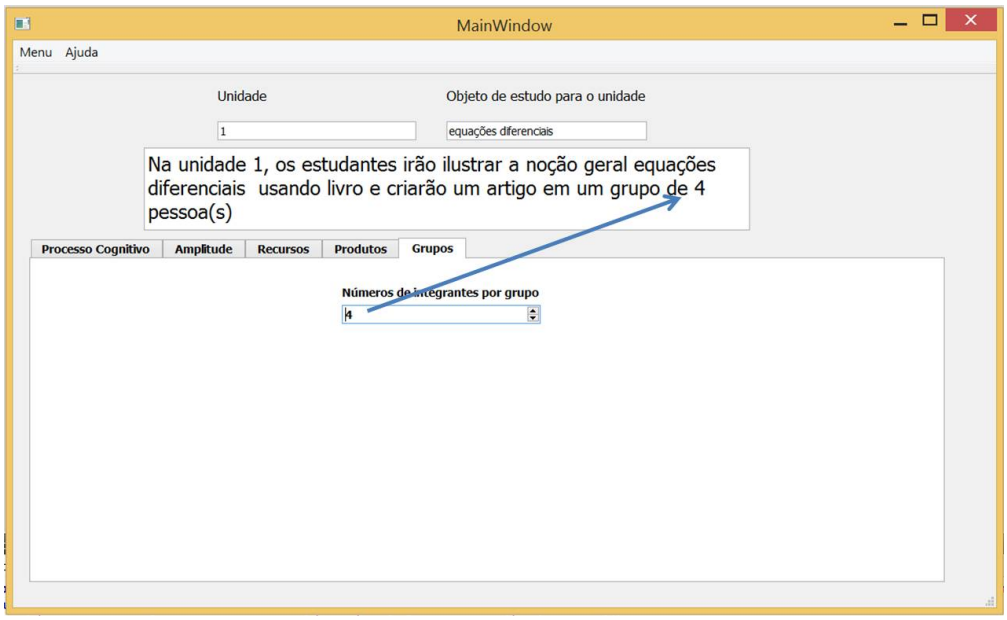

Figura 2.16: Selecionando a quantidade de pessoas por grupo

2.1.2.2 Aprendizagem por masterização

Aprendizagem por Masterização é uma metodologia de ensino proposta por Benjamim S. Bloom em 1968 juntamente com uma Taxonomia própria. Essa metodologia se baseia nas pesquisa de Bloom relacionadas aos *níveis cognitivos de aprendizagem.*

Como mostrado na seção 2.1.2.1, Os níveis cognitivos de aprendizagem são crescente do mais simples ao mais complexo. Isso significa que, para adquirir uma nova habilidade pertencente ao próximo nível, o aluno deve ter dominado e adquirido a habilidade do nível anterior. Ao ensinar um turma sob o paradigma da masterização, a maioria dos alunos (80 a 90%) deve ter masterizado a hablidade proposta para que outro conteúdo possa ser tratado.

Só após conhecer um determinado assunto alguém poderá compreendê-lo e aplicá-lo. Nesse sentido, a taxonomia proposta não é apenas um esquema para classificação, mas uma possibilidade de organização hierárquica dos processos cognitivos de acordo com níveis de complexidade e objetivos do desenvolvimento cognitivo desejado e planejado.

Os processos categorizados pela Taxonomia dos Objetivos Cognitivos de Bloom, além de representarem resultados de aprendizagem esperados, são cumulativos, o que caracteriza uma relação de dependência entre os níveis e são organizados em termos de complexidades dos processos mentais.

Encerrando um modo de utilização bastante prático, uma vez que permite, a partir da utili-

zação de uma tabela Domínio Cognitivo perceber qual o verbo a utilizar / aplicar, em função do comportamento esperado, organizando os objectivos de aprendizagem em seis níveis, os quais são, por ordem crescente de complexidade os seguintes.

A metodologia de aprendizagem sob masterização se baseia sob 2 crenças fundamentais:

1. Virtualmente todos estudantes podem aprender todos conteúdos acadêmicos importantes ao nível de excelência
2. A função primária das escolas é definir objetivos de aprendizagem e ajudar todos estudante a alcançá-los.

Para que um curso baseado em masterização tenha sucesso, são seguintes características são mandatórias:

- Indicar claramente objetivos e metas instrucionais.
- Garantir clara ligação entre objetivos, ensino e testes.
- Comunicar grandes expectativas de sucesso aos alunos.
- Unidades de ensino pequenas e sequenciadas.
- Manter o ciclo básico de ensino-teste-correção-teste.
- Predefinir os padrões de maestria de conhecimento.
- Manter registros claros e atualizados de progresso dos alunos de forma compreensível para os estudantes para fornecer feedback imediato.
- Comprometimento ao desenvolvimento da equipe nas práticas de *masterização*.

A chave para a eficácia do método de aprendizagem por masterização reside na disponibilização sistemática e uso de testes e instruções corretivas. corretiva.

Instrução corretiva, por sua própria natureza, deve ser direcionada a alunos pessoalmente e em portanto a problemas e dificuldades pessoais. Este processo será flexível em termos de:

- a) Tempo: O tempo necessário, nas unidades iniciais, para utilização de instruções corretivas irá reduzir tal necessidade em unidades futuras e mais complexas, permitindo que a turma, possa, em sua maioria, atingir o nível de masterização.
- b) Estratégias de Grupo:

  As estratégias de agrupamento devem incluir a instrução grupo inteiro, aulas em grupo pequeno,tutoria entre pares, e tutoria individual. As decisões serão tomadas à luz dos testes de formação e recursos disponíveis. O agrupamento de alunos para corretivos será depende unicamente no feedback de teste de formação com permissição de movimento de maestria para subgrupos de não-mestria, caso seja identificada a necessidade. A associação a grupos não deve permanecer constante.

- c) Variedade: Os professores e instrutores devem possui uma gama diversificada de teste e instruções corretivas. Estudantes que masterizaram o conteúdo no primeiro teste formativo, devem ser envolver em atividades de ampliação de conhecimento no mesmo conteúdo masterizado, preferencialmente, e em atividades de tutoria em par para estudantes que ainda não masterizarão o conteúdo.

O fluxo da metodologia de masterização é o seguinte:

1. **Definição de objetivos educacionais:** Neste ponto, os objetivos educacionais são determinados pelo professor usando um currículo especificado ou não. Tanto professor quanto alunos devem compreender e focar nos objetivos especificados.

2. **Ensino:** Inicialmente, uma turma seguindo a metodologia de masterização se assemelhará com a metodologia tradicional. O material poderá ser apresentado via aulas, apresentações, demonstrações, discussões ou qualquer forma que o professor considerar mais conveniente. As duas características mais importantes de diferenciação em sala da aula da teoria de masterização em relação a teoria tradicional rapidamente devem ser evidenciados. A primeira característica é a seguinte: **O objetivo de cada aula deve ser claramente exposto**. A segunda característica é a seguinte: O deve explicitar aos estudantes que **ele acredita que todos estudantes podem aprender bem o material e espera que eles assim o façam**.

3. **Primeiro teste formativo:**

   Depois que todo material foi passado (o que pode levar um dia ou várias semanas), o professor realiza o teste formativo para verificar a aprendizagem dos alunos. Esse teste não conta como nota para gradação final. Ele serve para que tanto professor quanto alunos saibam onde mais trabalho é necessário. Esse passo é necessário para a existência de feedback contínuo e identificaçãod e erros na dinâmica de grupo.

4. **Alternativas de aprendizagem:**

   Após o primeiro teste formativo, é fornecido aos estudantes altenativas de aprendizagem. Aqueles que tiveram problemas no teste formativo serão reensinados de formas diferentes para corrigir os erros. Aqueles que já masterizaram o material participarão de atividades de enriquecimento e/ou ajudarão os estudantes que não masterizaram o conteúdo.

5. **Segundo teste formativo (ou reteste):**

   Depois que as alternativas de aprendizagem foram concluídas, o professor deve realizar um segundo teste relacioando ao mesmo material. Assumindo que após isso, a maioria dos estudantes pode masterizar o material, então, a turma está pronta para o processo conteúdo e o fluxo volta para o primeiro ponto dets enumeração.

6. **Teste geral:**

   Esse teste deve ser realizado após a masterização de um número predeterminado de unidades. Esse teste informará ao professor e estudantes o que foi aprendido até então

2.2 Estudo da metodologia *Scrum* para desenvolvimento agíl de projetos

Scrum é um metodologia ágil de densenvolvimento de projeto que surgiu no contexto de engenharia de software. O termo ágil se aplica nesse contexto para ser um contraste a metodologia em *Cascata* (metodologia clássica) de desenvolvimento de projetos de software. Enquanto na metodologia *Cascata*, foca-se principalmente em *processos* e na obediência a eles, na metodologia Scrum, foca-se na *melhora contínua* e *entrega constante de valor*.

Apesar de suas origens, tanto a metodologia em *Cascata* com o *Scrum* são amplamente utilizados no mais diversos campos.

Nesta seção, descreve-se as rapidamente a metodologia de desenvolvimento de projetos *Cascata*, para depois apresentar uma descrição detalhada da metodologia de desenvolvimento de projetos *Scrum*, a qual é a metodologia a ser aplicada no curso *Algoritmos e Programação de Computadores* proposto.

2.2.1 Metodologia de desenvolvimento de projetos *Cascata*

A Figura 2.17 mostra a um breve resumo das fases que devem ser obedecidas na metodologia de desenvolvimento de projetos *Cascata*. São elas:

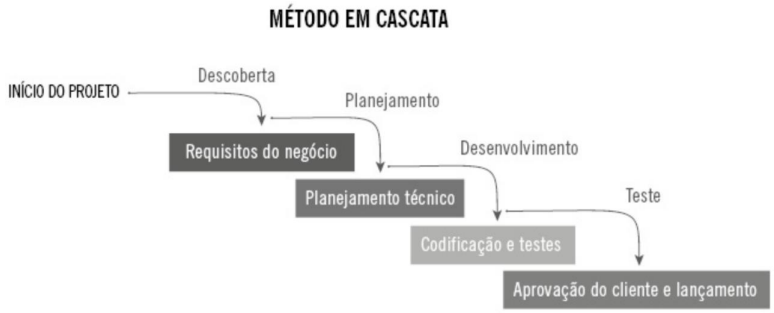

Figura 2.17: Fluxo da metodologia de desenvolvimento de projeto *Cascata*

1. Analíse e descoberta de requisitos de negócio
2. Planejamento de produto (Design)
3. Implementação (codificação)
4. Testes
5. Entrega do produto ao cliente (instalação)
6. Manutenção

Tabela 2.1: Pontos positivos e negativos da metodologia de desenvolvimento de projetos *Cascata*

Pontos positivos	**Pontos negativos**
Documentação detalhada	Começo de projeto lento
Requerimentos de projeto totalmente colhidos e acordados no início do projeto	Requerimentos fixos e dificilmente modificáveis
O projeto pode ser desenvolvido por pessoas ainda inexperientes.	Sem acompanhamento do projeto pelo cliente e outros stakeholders até o projeto estar concluído
Número de defeitos reduzido por meio de planejamento e desenvolvimento mais rigoroso	Pouca flexibilidade para mudar a direção e modus-operandi do projeto.
Início e fim de cada estágio plenamente definidos, permitindo acompanhamento mais eficiente do progresso.	Clientes e Stakeholders possivelmente podem alterar os requisitos de projeto após a fase de início de projeto

Todas etapas mostrada nessa figura são seguenciais planejadas extensivamente pela equipe de projeto, produzindo, nesse intervalo entre etapas, um conjunto amplo de documentos.

Muitos argumentam que a metodologia em *Cascata* torna o desenvolvimento do projeto muito engessado [15]. Os pontos positivos e negativos da metodologia de desenvolvimento de projeto *Cascata* são mostrados na tabela 2.1.

O principal problema, pontuado por pesquisadores [16], quanto à metodologia *Cascata*, é, como mostrado na tabela 2.1 a falta de dinamicidade, especialmente quanto a definição de pre-requisitos de projeto. Uma vez que a primeira fase do projeto, colhimento de requisitos, é concluída, não mais os requisitos são revistos. Isso causa, muitos problemas de retrabalho , em especial, nas fases finais e atrasos grandes.

2.2.2 Metodologia de desenvolvimento de projetos *Scrum*

A metodologia *Scrum* de gerenciamento de projeto surgiu como contraste à metodologia *Cascata* tratada na seção 2.2.1 sendo uma metodologia ágil de desenvolvimento de projetos.

A metodologogia Scrum é uma metodologia a ser desenvolvida em pequenas equipes, chamadas *Time ou Equipe Scrum* e umas de suas principais características são a revisão frequente dos ***requisitos de projeto***, a melhora contínua e o ***acréscimo de valor direto*** acima da obediciência a cronogramas e processos (Próprios de metodologias similares a metodologia Cascata).

Os papéis, na metodologia Scrum, são os seguintes:

- ***Scrum Master***: O *Scrum Master* é o responsável por manter a *Equipe Scrum* nas dire-

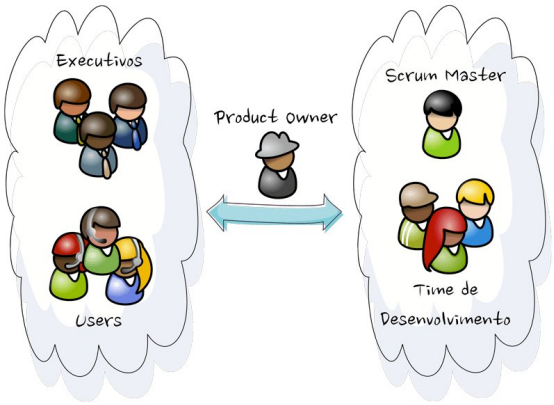

Figura 2.18: Papel *Product Owner*, metodologia *Scrum*

trizes do processo Scrum. Ele deve treinar e orientar os demais participantes. Além disso, ele é responsável por proteger a equipe Scrum de pertubações externas alheias ao projeto incentivando tal equipe na tomada de decisões para torna-lá progressivamente mais auto-gerenciável. Com relação a todos envolvidos no projeto, além da equipe Scrum, o *Scrum Master* deve zelar pela visibilidade do progresso do projeto para que a verificação e gerência sejam processos compartilhados.

- ***Product Owner***: O *Product Owner* é representa a "voz do cliente". Como a Figura 2.18 mostra, o *Product Owner* atua como "ponte"ente a equipe desenvolvimento e os *Stakeholders (Pessoa com interesse direto no produto)*. Ele deve entender as necessidades e prioridades dos *Stakeholders* e passar claramente os ***critérios de aceitação*** para a equipe de desenvolvimento

- ***Equipe Scrum***: As equipes Scrum são auto-organizadas e multidisciplinares. Como tal, ela deve escolher a melhor forma de desenvolver seus trabalhos ao invés de serem comandadas por outros de fora da equipe. Equipes multidisciplinares possuem todas as competências necessárias para desenvolverem seus trabalhos sem dependerem de outros que não fazem parte da Equipe Scrum. O modelo da Equipe Scrum é desenvolvido para otimizar a flexibilidade, criatividade e produtividade.

  Equipes Scrum entregam produtos iterativamente e incrementalmente maximizando a oportunidade de feedback. Entregas incremetais de produto "pronto"garantem que uma versão do produto potencialmente utilizável está sempre disponível para uso.

A Figura 2.19 mostra a intersecção entre os papéis da metodologia Scrum de acordo com suas respectivas tarefas.

A Figura 2.20 mostra em resumo o fluxo a ser estabelecido na metodologia Scrum. Tal fluxo segue a seguinte enumeração:

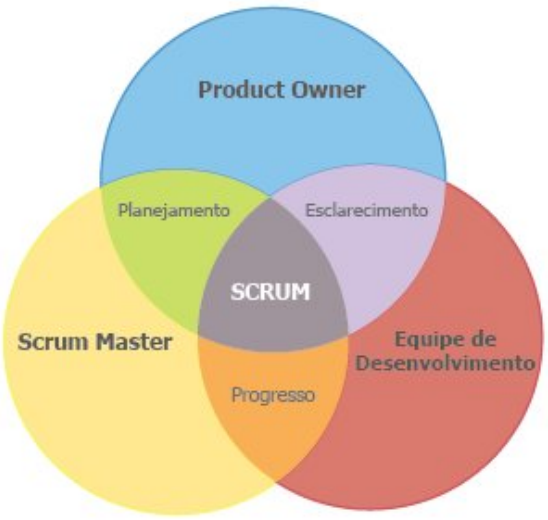

Figura 2.19: Metodologia Scrum, intersecção entre os papéis

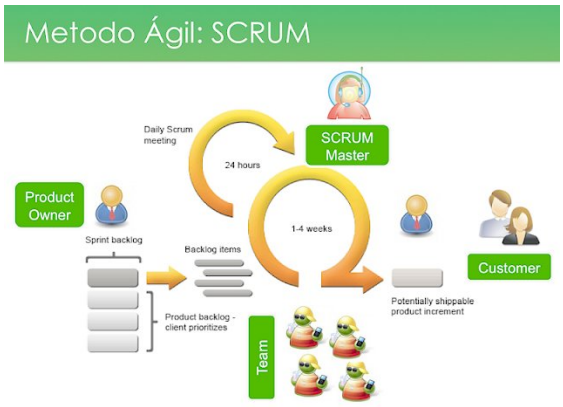

Figura 2.20: Fluxo da metodologia Scrum

- **Fase 1 - Início**: Para a fase inicial de um projeto Scrum, devem se desenvolvidas as seguintes passos:

  1.1)Criar Visão de Projeto: Descrição da necessidade ou desejo dos clientes e as características do produto que resolvem as necessidades. Sendo elas necessidades relacionadas a:

  Características funcionais: Características que descrevem uma funcionalidade direta do produto.

  Características não funcionais Características ligadas a eficiência e eficácia do produto ao realizar uma de suas característica funcionais.

  1.2) Para descrever as necessidades ou desejo dos clientes, devem ser criadas *User Stories* no formato: "Como um (***Tipo de Usuário***), eu quero, poder fazer (***Característica***

Funcional)".

- **Fase 2 - Plano de Projeto e Estimações**:

  2.1) Critérios de aceitação são criados (a ser feito pelo Product Owner juntamente com o Scrum Team).

  2.2) Scrum Master e time estimam o esforço necessário para desenvolver as User Stories.

  2.3) Scrum master e time se comprometem a desenvolver o produto de acordo com as Epics(Conjunto de User Stories) e critérios de aceitação.

  2.4) User Stories são divididas em subtarefas para criar uma Task List.

  2.5) Time Scrum se reune para decidir quais tarefas serão realizadas na Sprint (Sprint Backlog).

- **Fase 3 - Implementação**:

  Na fase de implementação, o trabalho é dividido em ciclos curtos de trabalho, de 1 a 4 semanas de duração, como mostrado na Figura 2.21. Esse ciclos são chamados de *Sprints*. Cada Sprint tem sua lista de tarefas(*Task List*) própria.

  As responsabilidades da Equipe Scrum na Fase 3, são as seguintes:

  3.1) Criar os "entregáveis"do SPRINT.

  3.2) Realizar reuniões diárias rápidas de no máximo 15 minutos para discutir problemas e progresso.

  3.3) Atualizar o quadro Scrum de (A FAZER , FAZENDO, FEITO) a cada dia de trabalho.

- **Fase 4 - Revisão e Retrospecto** :

  A fase de *Revisão e Retrospecto* é realizada ao final de cada Sprint. As responsabilidades da equipe Scrum na Fase 4 são"

  4.1) Os entregáveis do Sprint são mostrados para os Stakeholders.

  4.2) Essas reuniões são feitas para garantir a aprovação do produto desenvolvido e fazer os ajustes necessários o mais rápido possível.

  4.3) O Scrum Master e o time Scrum se reunem para discutir o que foi aprendido no Sprint.

  4.4) Informação é documentada para futuros Sprints.

Após a conclusão dessas 4 fases, o projeto Scrum chega a sua finalização. As informações colhidas a cada Sprint são guardadas para projetos Scrum futuros.

Tabela 2.2: Pontos positivos e negativos da metodologia de desenvolvimento de projetos *Scrum*

Pontos positivos	**Pontos negativos**
Início rápido, entrega de produto realizada incrementalmente com revisão de clientes e feedback frequentes	Muitas vezes, o planejamento Scrum pode ser interpretado como mau-planejado e indisciplinado
Verificação frequente da evolução dos requerimentos do cliente	Necessita de um time altamente qualificado e pronto a interagir com clientes diretamente
Resposta rápida a mudanças	É necessário um alto nível de envolvimento dos clientes no projeto.
Menos retrabalho a ser feito, por causa do envolvimento do cliente, testes contínuos e feedback frequente.	Falta de planejamento de longo prazo detalhado
Comunicação de tempo real entre time de desenvolvimento e clientes	Pouca documentação produzida

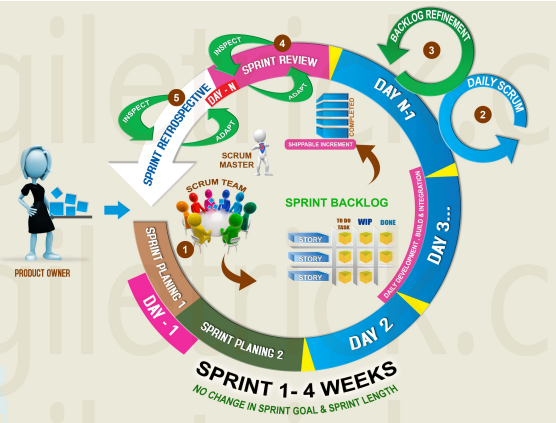

Figura 2.21: Fluxo de um Sprint dentro da metodologia Scrum

A tabela 2.2 faz um resumo dos pontos positivos e negativos próprios de metologias ágeis de desenvolvimento de projeto, como a metodologia Scrum.

2.3 Proposta de curso

Para se construir um projeto Scrum, como exposto na seção 2.2, para o contexto educacional da teoria de masterização de habilidades proposta por Bloom, os seguintes passos são necessários:

1. Realizar adaptação dos princípios e fluxo da metodologia Scrum, em conjunto com os princípios da metodlogia de *masterização de habilidades* para o contexto educacional desejado.

2. Definir os objetivos educacionais na forma *de características de produto.* utilizando a taxonomia de Bloom para criar as *User Stories* e *Epics.*

3. Criar atividades com a placa Galileo de acordo com o contéudo programático estabelecido

Cada uma das atividades listadas acima será tratada, neste trabalho, em uma seção em separado.

2.3.1 Adaptação dos princípios e fluxo metodologia Scrum à disciplina *Algoritmos e Programação de Computadores*

Nesta seção é apresentada uma proposta de adaptação da metodologia Scrum para aplicação na disciplina *Algoritmos e Programação de Computadores.*

A tabela 2.3 mostra um resumo de todo fluxo mostrado na figura 2.21 e, ao lado, a proposta de adaptação:

Tabela 2.3: Scrum aplicado à *Algoritmos e Programação de Computadores*

	Scrum	**Scrum + Masterização de habilidades**
Por que utilizar Scrum?	-Uso de ciclo curtos de desenvolvimento e revisão(Sprints) -Revisão frequente dos requisitos de projeto a cada Sprint - Melhora contínua -Valor agredado acima de obediciência a cronograna e processos - Entrega contínuade Valor	1)Verificação,de níveis de aprendizagem frequentes 2)Aumento, a cada Sprint, da responsabilização pessoal dos alunos com relação a seus estudos

	Scrum	Scrum + Masterização de habilidades
Papéis	- Product Owner Representante da voz do cliente -Scrum Master Gerente do time Scrum responsável por certificar-se que os princípios Scrum estão sendo seguidos pela equipe e também responsável por coletar os requisitos de projeto junto ao Product Owner -Time Scrum ->Equipe responsável por desenvolver os requisitos de projeto descritos, inclusive, por meio de User Stories	Product Owner - Professor Scrum Master - Professor (enquanto os alunos não estiverem familiarizados com a metodologia Scrum e prontos para auto-gerência de seus,estudos) Time Equipe Scrum - Equipe de até 4 estudantes responsáveis(sem contar o professor) por concluir, conjuntamente, os objetivos de aprendizagem descritos por meio da taxonomia de Bloom. A equipe Scrum deve eleger um representante novo a cada Sprint para se comunicar diretamente com o professor (Scrum Master) e planejar as atividades e meios de verificação conjuntamente

	Scrum	Scrum + Masterização de habilidades
Fase 1 - Início Definição de projeto	1) Criar visão de projeto 2) Identificar Scrum Master e Skateholders 3) Formar Time Scrum 4) Desenvolver Epics (Conjunto de user stories) 5) Criar conjunto prioritário de características do produto 6) Criar um plano de desenvolvimento do produto em Sprints sequenciais	1),Criar objetivos educacionais com a taxonomia de Bloom 2)Scrum Master = Professor Stakeholders =,alunos, familias, governo,etc 3) Formar grupos de 3 ou 4 alunos , preferencialmente não amigos e com características complementares 4) Dividir os objetivos educacionais em módulos 5) Criar plano de ascenção dos alunos nos níveis cognitivos de bloom para os conteúdos especificados 6) Criar especificação de Sprint com os objetivos educacionais
Fase 2 - Plano de Projeto e Estimação	1)Critérios de aceitação são criados (a ser feito pelo Product Owner juntamente com o Scrum Team) 2) Scrum Master e time estimam o esforço necessário para desenvolver as User Stories 3)Scrum master e time se comprometem a desenvolver o produto de acordo com as Epics (Conjunto de User Stories) e critérios de aceitação 4)User Stories são divididas em subtarefas para criar uma Task List 5) Time Scrum se reune para decidir quais tarefas serão realizadas na Sprint (Sprint Backlog)	1) O professor, a cada Sprint, deve deixar claro quais serão os critérios de masterização de habilidades preferencialmente utilizandoa taxonomia de Bloom 2)Professor é responsável, a cada Sprint, por estimar o esforço necessário por parte da turma para as masterizações desejadas. A participação dos representantes de equipes com relação à definição de planos Sprint é crescente ao longo dos Sprints.

	Scrum	Scrum + Masterização de habilidades
Fase 3 - Implementação	1)Criar os "entregáveis" do Sprint 2) Realizar reuniões diárias rápidas de no máximo 15 minutos para discutir problemas e progresso 3) Atualizar o quadro Scrum antes de iniciar o dia de trabalho	1)Estudar o conteúdo especificado emgrupo com a ajuda do Scrum Master(Professor) 2) Realizar reuniões diárias rápidas de no máximo 15 minutos para discutir problemas e progresso 3)A equipe Scrum decide os meios de aprendizagem que usarão com auxílio do professor. A cada Sprint, a equipe Scrum deve ficar mais livre para decidir os meios mais eficientes de,aprendizagem 4) Atualizar o quadro Scrum de (A FAZER , FAZENDO, FEITO)
Fase 4 - Fim de Sprint; Revisão e Retrospecto	1)Os,entregáveis são mostrados aos Stakeholders 2) Essas reuniões são feitas para garantir a aprovação do produto desenvolvido e fazer os ajustes necessários o mais rápido possível 3) O Scrum Master e o time Scrum se reunem para discutir o que foi aprendido no Sprint 4) Informação é documentada para futuros Sprints	1)A equipe Scrum desenvolve testes rápidos(para serem resolvidos em no máximo 20 minutos) para que as outras equipes resolvam do conteúdo estudado no Sprint. O gabarito deve ser entregue ao professor antes da aplicação dos testes para que este verifique-os e decida quais testes serão aplicados 2) A equipe resolve, no tempo estipulado, os testes compartilhados 3) Após os testes de unidade, o professor e os alunos tem uma aula para revisar o que foi aprendido de acordo com os objetivos educacionais traçados

2.3.1.1 Definição de objetivos educacionais na forma da metologogia Scrum

Para realizar a definição dos objetivos educacionais, foi utlizada a plataforma web scrumdo ® sendo cadastrados os seguintes objetivos educacionais próprios da disciplina *Algoritmos e Programação de Computadores*:

1. O histórico da computação
2. Organização básica de um computador
3. Fundamentos para linguagens de programação
4. Introdução ao conceito de algoritmo
5. Pseudocódigo e Fluxograma
6. Tipos de variáveis de memória
7. Operadores e expressões
8. Algoritmos sequenciais
9. Algoritmos com alternativas (simples, compostas, aninhadas e de múltipla escolha)
10. Algoritmos com repetição (com teste no início, com teste no fim e com variável de controle)
11. Algoritmos com vetores e matrizes
12. Subalgoritmos, passagem de parâmetros,
13. Ponteiros
14. Recursividade
15. Registros
16. Arquivos
17. Ordenação e busca

Aos objetivos listados, devem ser incluídos os seguintes objetivos educacionais relativos a placa Intel® Galileo e eletrônica básica:

1. Leis básicas de eletrônica(Leis de Kirschoff)
2. Sistema da placa Galileo
3. Ferramentas básicas de prototipação eletrônica
4. Fundamentos de programação em Galileo
5. Introdução a sensores
6. Interação com sensores
7. Uso de displays
8. Motores

9. Tópico mais avançado em eletrônica a ser escolhido com a Turma (Comunicação Ethernet ou Internet, Comunicação sem fio, Uso de placa SD de armazenamento de dados, uso de interrupções, Circuitos integrados periféricos, controle de altas cargas, etc)

A Figura 2.22 mostra o plataforma web scrumdo ® para cadastramento de quadros scrum pertinentes.

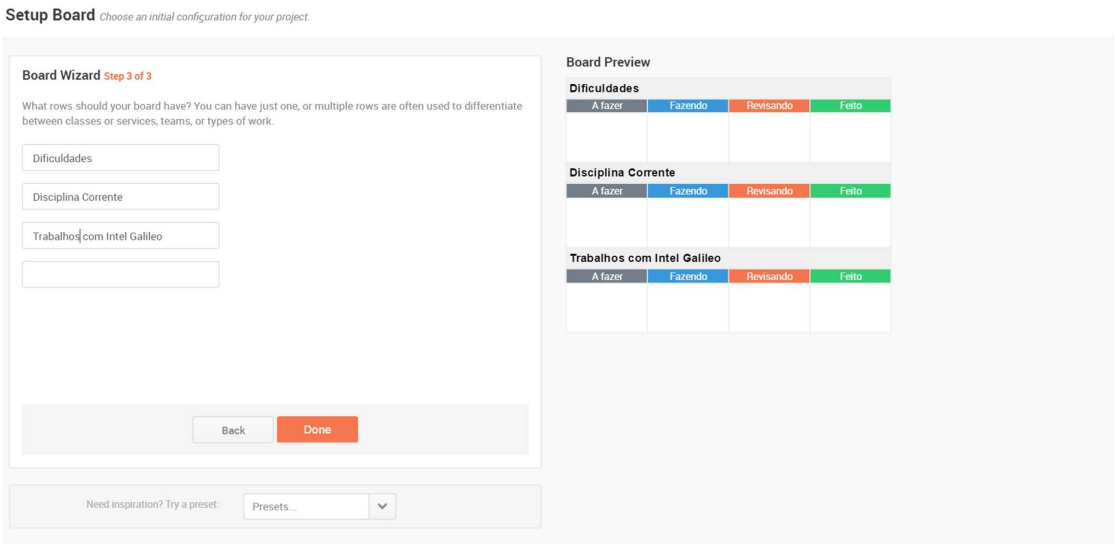

Figura 2.22: Plataforma scrumdo: Formato dos quadros Scrum planejados para a disciplina *Algoritmos e Programação de Computadores*

A Figura 2.23 mostra os quadros *Dificuldades Algoritmos e Programação de Computadores*, *Aplicação com Galileo*. A disposição dos 3 quadros deve refletir a prioridade da turma. As reuniões de Sprint devem servir para resolver questões pendentes posta no quadro *Dificuldades*.

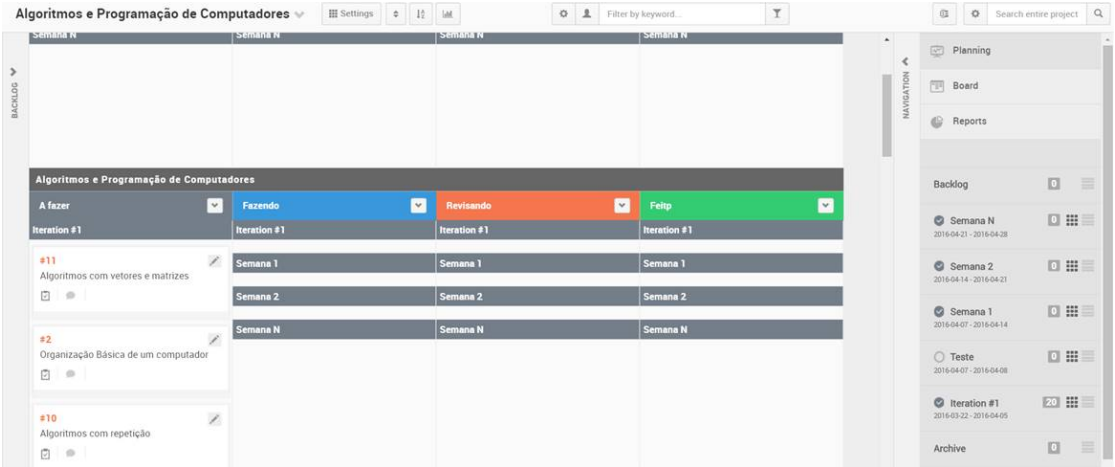

Figura 2.23: Plataforma scrumdo: Quadros Scrum criados e objetivos educacionais, segundo a Taxonomia de Bloom escritos *Algoritmos e Programação de Computadores*

Pode-se, por meio da plataforma, selecionar sub-objetivos. No caso deste trabalho, para cada tópico listado para as práticas com a placa Galileo, foram criados objetivos educacionais para os três primeiros níveis cognitivos identificados por Bloom(Nível de Conhecimento, Compreensão a Aplicação).

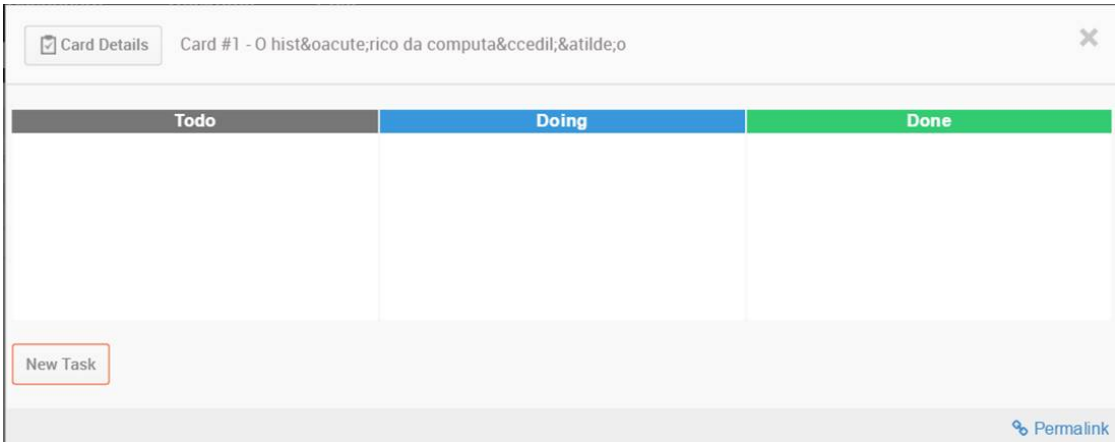

Figura 2.24: Plataforma scrumdo: Definicação de sub-objetivos de um objetivo listado no *Product Backlog Algoritmos e Programação de Computadores*

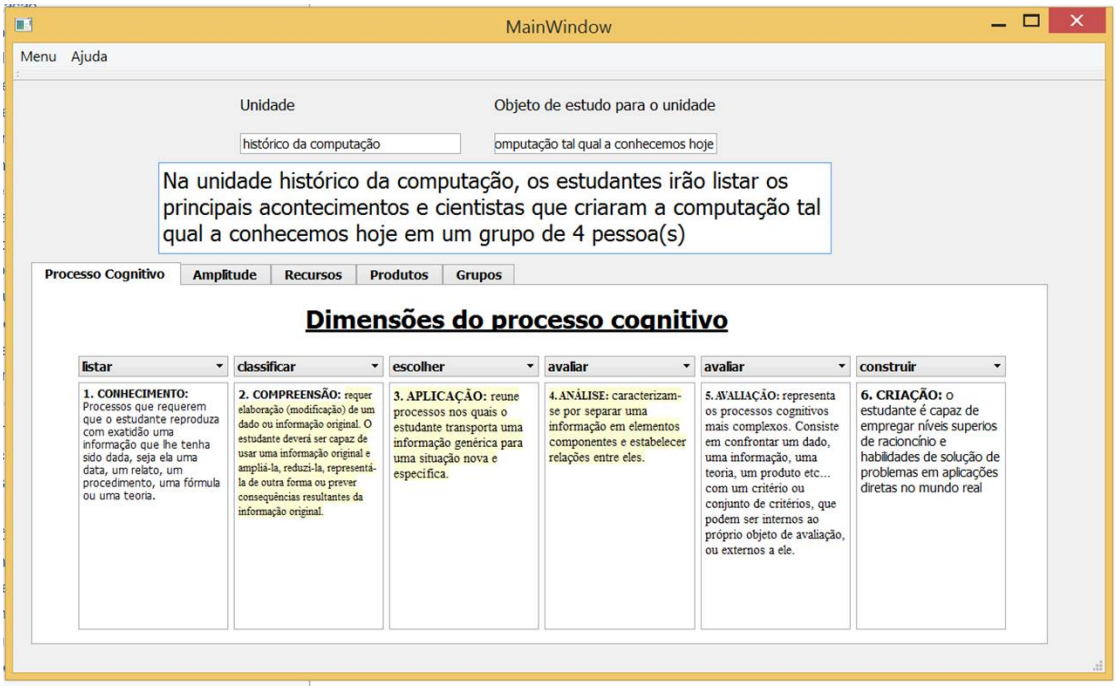

Figura 2.25: Uso do aplicativo desenvolvido em QT para definir os objetivo educacional para o nível *Conhecimento*

2.3.1.2 Definição de objetivos educacionais para a Disciplina *Algoritmos e Programação de Computadores*

Para especificar cada objetivo educacional, foi utilizado o aplicativo desenvolvido em QT, como descrito anteriomente e como mostrado na Figura 2.25.

A seguinte enumeração mostra uma possível especificação de cada um dos módulos de aprendizagem definidos anteriormente seguindo a *Taxonomia de Bloom*:

1. O histórico da computação

   Nível 1 Conhecimento: Na unidade **histórico da computação**, os estudantes irão listar os principais acontecimentos e cientistas que criaram a computação tal qual a conhecemos hoje em um grupo de 4 pessoa(s).

   Nível 2 Compreensão: Na unidade histórico da computação, os estudantes irão explicar a relevância da computação no mundo moderno em um grupo de 4 pessoa(s).

   Nível 3 Aplicação: Na unidade histórico da computação, os estudantes irão escolher, dentre certos sistemas computacionais exposto, qual seria aquele(s) que mais se adequariam ao contexto exposto em um grupo de 4 pessoa(s).

2. Organização básica de um computador

   Nível 1 Conhecimento: Na unidade organização básica de um computador, os estudantes irão enumerar e explicar o funcionamento de todos os sistemas que compõem um computador em um grupo de 4 pessoa(s).

   Nível 2 Compreensão: Na unidade organização básica de um computador, os estudantes irão classificar computadores quanto a parâmetros de desempenho entre outros em um grupo de 4 pessoa(s).

   Nível 3 Aplicação: Na unidade organização básica de um computador, os estudantes irão ilustrar a dinâmica de funcionamento de um computador e criarão um diagrama em um grupo de 4 pessoa(s).

3. Fundamentos para linguagens de programação

   Nível 1 Conhecimento: Na unidade fundamentos para linguagens de programação, os estudantes irão definir a noção geral sobre padrões de linguagens de programação em um grupo de 4 pessoa(s).

   Nível 2 Compreensão: Na unidade fundamentos para linguagens de programação, os estudantes irão explicar a noção geral das principais diferenças entre os diversos tipos de linguagens de programação em um grupo de 4 pessoa(s).

   Nível 3 Aplicação: Na unidade fundamentos para linguagens de programação, os estudantes irão escolher para diversas situações e sistemas diferentes, linguagens de programação mais adequadas em um grupo de 4 pessoa(s).

4. Introdução ao conceito de algoritmo

Nível 1 Conhecimento: Na unidade introdução ao conceito de algoritmo, os estudantes irão definir conceitualmente o que são algoritmos em um grupo de 4 pessoa(s).

Nível 2 Compreensão: Na unidade introdução ao conceito de algoritmo, os estudantes irão identificar vários tipos diferentes algoritmos em um grupo de 4 pessoa(s).

Nível 3 Aplicação: Na unidade introdução ao conceito de algoritmo, os estudantes irão empregar um determinado tipo de algoritmo para cada situação dada em um grupo de 4 pessoa(s).

5. Pseudocódigo e Fluxograma

Nível 1 Conhecimento: Na unidade pseudocódigo e fluxograma, os estudantes irão duplicar por meio das ferramentas citadas, situações do cotidiano em um grupo de 4 pessoa(s).

Nível 2 Compreensão: Na unidade pseudocódigo e fluxograma, os estudantes irão selecionar por meio dos pseudocódigos e fluxogramas já construídos quais são os mais eficientes para cada situação dada em um grupo de 4 pessoa(s).

Nível 3 Aplicação: Na unidade pseudocódigo e fluxograma, os estudantes irão programar os primeiros algoritmos em um grupo de 4 pessoa(s).

6. Tipos de variáveis de memória

Nível 1 Conhecimento: Na unidade tipos de variáveis de memória, os estudantes irão listar todos os tipos de variáveis em um grupo de 4 pessoa(s).

Nível 2 Compreensão: Na unidade tipos de variáveis de memória, os estudantes irão explicar as diferenças de usabilidade entre todos os tipo de variáveis em um grupo de 4 pessoa(s).

Nível 3 Aplicação: Na unidade tipos de variáveis de memória, os estudantes irão escolher dentre todos os tipos de variável de memória, aquelas que melhor se aplicam às situações propostas e explicar o motivo em um grupo de 4 pessoa(s).

7. Operadores e expressões

Nível 1 Conhecimento: Na unidade operadores e expressões, os estudantes irão enumerar os vários tipos de operadores e expressões na linguagem C e demonstrar seu uso em um grupo de 4 pessoa(s).

Nível 2 Compreensão: Na unidade operadores e expressões, os estudantes irão descrever o resultado de várias expressões escritas, explicitando os passos desenvolvidos em um grupo de 4 pessoa(s).

Nível 3 Aplicação: Na unidade operadores e expressões, os estudantes irão desenvolver programas para solução de diversas situações propostas em um grupo de 4 pessoa(s).

8. Algoritmos sequenciais

Nível 1 Conhecimento: Na unidade algoritmos sequenciais, os estudantes irão definir o conceito, todas expressões e operadores utilizados em um grupo de 4 pessoa(s).

Nível 2 Compreensão: Na unidade algoritmos sequenciais, os estudantes irão definir o resultado esperado na utilização de um algoritmo sequencial em um grupo de 4 pessoa(s).

Nível 3 Aplicação: Na unidade algoritmos sequenciais, os estudantes irão empregar algoritmos sequencias na solução de situações diversas em um grupo de 4 pessoa(s).

9. Algoritmos com alternativas (simples, compostas, aninhadas e de múltipla escolha)

   Nível 1 Conhecimento: Na unidade algoritmos com alternativas, os estudantes irão definir os vários de operadores condicionais utlizados em C quanto a sua dinâmica e utilização em um grupo de 4 pessoa(s).

   Nível 2 Compreensão: Na unidade algoritmos com alternativas, os estudantes irão identificar o resultado de uma série de condicionais aplicados num algoritmo sequencial em um grupo de 4 pessoa(s).

   Nível 3 Aplicação: Na unidade algoritmos com alternativas, os estudantes irão construir programas complexos com alternativa condicionais em um grupo de 4 pessoa(s).

10. Algoritmos com repetição (com teste no início, com teste no fim e com variável de controle)

    Nível 1 Conhecimento: Na unidade algoritmos com repetição, os estudantes irão lembrar a sintaxe correta de todas formas de laços de repetição em C em um grupo de 4 pessoa(s).

    Nível 2 Compreensão: Na unidade algoritmos com repetição, os estudantes irão explicar a diferença de usubilidade das formas de laços de repetição em C em um grupo de 4 pessoa(s).

    Nível 3 Aplicação: Na unidade algoritmos com repetição, os estudantes irão construir programas complexos com laços de repetição em um grupo de 4 pessoa(s).

11. Algoritmos com vetores e matrizes

    Nível 1 Conhecimento:

    Nível 2 Compreensão:

    Nível 3 Aplicação: Na unidade algoritmos com vetores e matrizes, os estudantes irão construir programas complexos com utilizando tais conceitos em um grupo de 4 pessoa(s).

12. Subalgoritmos e passagem de parâmetros

    Nível 1 Conhecimento: Na unidade subalgoritmos e passagem de parâmetros, os estudantes irão expor por meio de exemplo e aplicações, a forma sintaticamente correta de escrita de subalgoritmos e passagem de parâmetros em um grupo de 4 pessoa(s)

    Nível 2 Compreensão: Na unidade subalgoritmos e passagem de parâmetros, os estudantes irão identificar todos parâmetros e requisitos de para correta escrita de tais rotinas computacionais em um grupo de 4 pessoa(s).

    Nível 3 Aplicação: Na unidade subalgoritmos e passagem de parâmetros, os estudantes irão construir programas complexos, construídos sob diversos subalgoritmos, em um grupo de 4 pessoa(s)

13. Ponteiros

    Nível 1 Conhecimento: Na unidade ponteiros, os estudantes irão listar todas características que definem por completo um ponteiro em um grupo de 4 pessoa(s).

    Nível 2 Compreensão: Na unidade ponteiros, os estudantes irão identificar os resultados de diversas operações utilizando ponteiros em um grupo de 4 pessoa(s).

    Nível 3 Aplicação: Na unidade ponteiros, os estudantes irão construir programas complexos utilizando ponteiros em um grupo de 4 pessoa(s).

14. Recursividade

    Nível 1 Conhecimento: Na unidade recursividade, os estudantes irão denominar o conceito associado, a usabilidade e aplicações de recursividade em um grupo de 4 pessoa(s).

    Nível 2 Compreensão: Na unidade recursividade, os estudantes irão reconhecer o resultado final da aplicação da recursividade em diversas situações propostas em um grupo de 4 pessoa(s).

    Nível 3 Aplicação: Na unidade recursividade, os estudantes irão construir programas complexos utilizando tal conceito em um grupo de 4 pessoa(s).

15. Registros

    Nível 1 Conhecimento: Na unidade registros, os estudantes irão definir todos conceitos e aplicações relacionadas a registros em um grupo de 4 pessoa(s).

    Nível 2 Compreensão: Na unidade registros, os estudantes irão selecionar formas de registros mais adequadas para cada situação dada em um grupo de 4 pessoa(s).

    Nível 3 Aplicação: Na unidade registros, os estudantes irão construir programas complexos com laços de repetição em um grupo de 4 pessoa(s).

16. Arquivos

    Nível 1 Conhecimento: Na unidade registros, os estudantes irão definir todos conceitos e aplicações relacionadas a registros em um grupo de 4 pessoa(s).

    Nível 2 Compreensão: Na unidade registros, os estudantes irão selecionar formas de registros mais adequadas para cada situação dada em um grupo de 4 pessoa(s).

    Nível 3 Aplicação: Na unidade registros, os estudantes irão construir programas complexos com laços de repetição em um grupo de 4 pessoa(s).

17. Ordenação e busca

    Nível 1 Conhecimento: Na unidade ordenação e busca, os estudantes irão enumerar e definir diversas algoritmos de enumeração e busca em um grupo de 4 pessoa(s).

    Nível 2 Compreensão: Na unidade ordenação e busca, os estudantes irão descrever todos passos dos algoritmos de ordenação e busca estudados anteriomente em um grupo de 4 pessoa(s).

    Nível 3 Aplicação: Na unidade algoritmos com repetição, os estudantes irão construir programas complexos com laços de repetição em um grupo de 4 pessoa(s)

2.3.1.3 Definição de objetivos educacionais para a Disciplina *Algoritmos e Programação de Computadores* com relação às atividades práticas

Nesta seção, define-se um possível planejamento de módulos de aprendizagem relacionadas a eletrônica básica utlizando a placa Galileo em paralelo à aprendizagem de programação estruturada na linguagem C.

1. Leis básicas de eletrônica (Leis de Kirschoff)

   Nível 1 Conhecimento: Na unidade leis básicas de eletrônica(Leis de Kirschoff), os estudantes irão enumerá-las todas e com dados exemplos e situações, escreve-lás propriamente em um grupo de 4 pessoa(s).

   Nível 2 Compreensão: Na unidade leis básicas de eletrônica(Leis de Kirschoff), os estudantes irão reconhecer todos parâmetros das leis de Kirschoff, nos circuitos elétricos exemplificados, em um grupo de 4 pessoa(s).

   Nível 3 Aplicação: Na unidade leis básicas de eletrônica(Leis de Kirschoff), os estudantes irão empregar diferentes dispositivos eletrônicos em circuitos desenvolvidos em simulações e observar as relações das lesi de Kirschoff estudadas anteriormente em um grupo de 4 pessoa(s).

2. Sistema da placa Galileo

   Nível 1 Conhecimento:

   Nível 2 Compreensão:

   Nível 3 Aplicação:

3. Ferramentas básicas de prototipação eletrônica

   Nível 1 Conhecimento:

   Nível 2 Compreensão:

   Nível 3 Aplicação:

4. Fundamentos de programação em Galileo

   Nível 1 Conhecimento:

   Nível 2 Compreensão:

   Nível 3 Aplicação:

5. Introdução a sensores

   Nível 1 Conhecimento:

   Nível 2 Compreensão:

   Nível 3 Aplicação:

6. Interação com sensores

   Nível 1 Conhecimento:

   Nível 2 Compreensão:

   Nível 3 Aplicação:

7. Uso de displays

   Nível 1 Conhecimento:

   Nível 2 Compreensão:

   Nível 3 Aplicação:

8. Motores

   Nível 1 Conhecimento:

   Nível 2 Compreensão:

   Nível 3 Aplicação:

9. Tópico mais avançado em eletrônica a ser escolhido com a Turma (Comunicação Ethernet ou Internet, Comunicação sem fio, Uso de placa SD de armazenamento de dados, uso de interrupções, Circuitos integrados periféricos, controle de altas cargas, etc)

   Nível 1 Conhecimento:

   Nível 2 Compreensão:

   Nível 3 Aplicação:

2.4 Placa Intel ® Galileo

A placa Intel® Galileo é uma *placa de desenvolvimento* com microcontrolador baseado processador Intel® Quark SoC X1000[17]. A placa Galileo possui software e hardware compatível com a placa *Arduino* com relação aos pinos digitais e analógicos. Um programa escrito para Arduino pode ser usado no Galileu por causa dessa compatibilidade. As Figura 2.26 e 2.27 mostram a placa Intel ® em suas visão frontal e traseira.

Nesta seção são apresentadas, enumeradas e explicadas todas características da placa Intel® Galileo.

Primeiramente são apresentados os pinos da placa Galileo juntamente com uma breve descrição de seu uso. Após isso são descritas as enumeradas e explicadas todas características eletro-eletrônicas da placa. Para cada tecnologia na placa é reservada uma pequena sub-seção neste capítulo para sua devida elucidação.

2.4.1 Pinagem da placa Intel ® Galileo

Os pinos da placa Galileo nas partes frontal e traseira são mostrados nas figuras 2.26 e 2.27

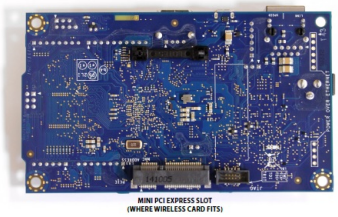

Figura 2.27: Descrição dos pinos da placa Galileo - parte traseira[1]

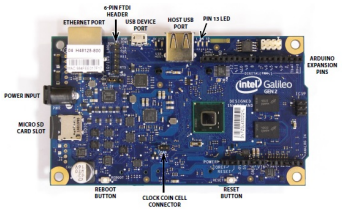

Figura 2.26: Descrição dos pinos da placa Galileo - parte frontal[1]

A descrição de cada um desses pinos é a descrita na tabela 2.4:

2.4.2 Características Elétricas e Eletrônicas da placa Intel ® Galileo

As características elétricas e eletrônicas da placa são enumeradas a seguir. As características que têm uma sub-seção para explicação mais aprofundada estão marcadas com *itálico* e **negrito**:

- Clock de 400 MHz
- Arquitetura Intel® 32 bits
- 14 pinos digitais para entrada e saída, 6 das quais podem ser usadas para saída ***PWM***
- 6 pinos para entrada analógica utilizando o ***conversor analógico-digital AD7298***
- Barramento Serial ***I2C***
- Comunicação serial com periféricos ***SPI***
- Porta Serial ***UART***
- 16KBytes de memória ***L1 Cache***
- 512KBytes de memória ***SRAM***
- Clock de tempo real integrado ***(RTC)***
- Barramento ***PCI Express***
- Conexão para ***USB Host e USB Client***

Tabela 2.4: Pinos Galileo[1]

Pino	**Descrição**
Micro SD Card Slot	Pino no qual se pode um SD Card para permitir ao Galileo a execução de uma versão de Linux com mais recursos
Arduino Expansions Pins	Pinos de entrada e saída da placa Galileo. Esses pinos são compatíveis com os pinos do Arduino e Shields relacionadas.
USB Device Port	Pino para conectar um cabo USB do Galileo ao computador para carregar o Galileo com um programa Arduino
Host USB Port (como webcam, caixa de som, etc)	Pino para conectar um dispositivo periférico
6-Pin FTDI Header - Linux instalado no Galileo	Adaptador para comunicação serial computador
Power Input	Conexão para bateria de 12V. ATENÇÃO, a bateria sempre deve ser conectada ao Galileo antes de conectar um cabo USB do Galileo ao computador para evitar danos a placa.
Ethernet Port	Pino para conectar o Galileo à Internet pelo cabo Ethernet
Mini PCI Express Slot	Pino para conectar um cartão WiFi
Clock Baterry Power	Conexão para uma bateria de relógio de 3V de forma a fazer com Galileo guarde informações de data e hora
Reboot Button	Botão para realizar a placa, inclusive o sistema operacional
Reset Button	Botão para resetar o código que foi carregado no Galileo

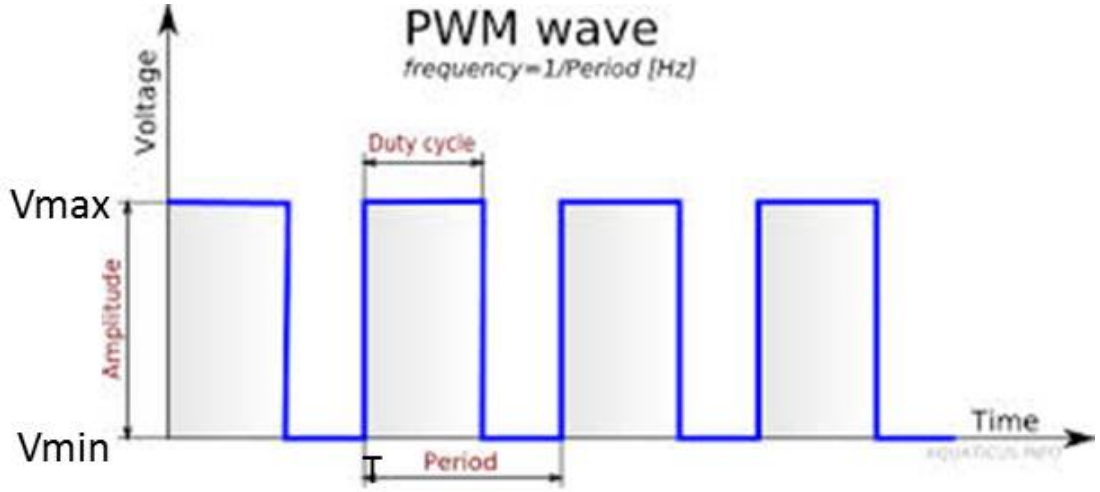

Figura 2.28: Sinal PWM

- 10 pinos padrões ***JTAG*** para debug
- 256 MBytes de memória ***DRAM***
- 11 KBytes de memória ***EEPROM***

2.4.2.1 Sinal PWM

Pulse Width Modulation (PWM) ou Modulação por Largura de Pulso é uma técnica que modulação de impulso utilizada principalmente para codificar uma mensagem num sinal pulsante[18]

Para o caso do Intel Galileo, as aplicações do sinal PWM são principalmente relacionadas ao controle da tensão DC fornecida a um circuito.

O sinal PWM é gerado com ondas quadradas, de período T de ciclo. Durante parte do período, o sinal terá amplitude *Vmax*. O intervalo de tempo no qual no sinal tem amplitude *Vmax* é chamado *Duty-Cicle* como mostra a Figura 2.28. O valor DC de um sinal periódico é calculado como a média aritmética da amplitude do sinal no período. O valor da tensão DC fornecida ao circuito pelo Sinal PWM é calculado com a equação 2.1:

$$V_{dc} = 1/T() \int_0^{DutyCicle} V_{max} dt + \int_{DutyCicle}^{T} V_{min} dt \tag{2.1}$$

$$V_{dc} = 1/T(DutyCicle * V_{max} + T * V_{min} - DutyCicle * V_{min}) \tag{2.2}$$

Se a tensão mínima (Vmin) for igual a zero, o valor DC do sinal PWM é dado por:

$$V_{dc} = \frac{DutyCicle * V_{max}}{T} \tag{2.3}$$

A equação 2.3 mostra que quanto maior for o tempo que o sinal permanecer no seu valor máximo (Vmax), mais próximo de Vmax será o valor DC fornecido ao circuito.

O sinal PMW é gerado na placa Galileo utilizando o clock interno máximo de 400 MHz e registradores de Timer específicos para contagem de pulsos do clock.

Como exemplo para a geração do sinal PWM, digamos que o clock da placa foi setado para a frequência 1kHz. Isso significa que a cada 1ms, o clock gerará um pulso, como indicado na equação 2.4.

$$f = 1Khz -> T = 1ms -> 1000\ pulsos\ de\ clock\ por\ segundo \tag{2.4}$$

Como a tensão de operação da placa Galileo é 5 V, então:

$$V_{max} = 5V \tag{2.5}$$

Caso se queria gerar um sinal PWM cujo componente DC seja 2.5, é necessário então que durante metade do ciclo do sinal PWM, a amplitude do cinal seja 5 V e durante a outra metade do ciclo, a amplitude seja 0V. Para criar tal sinal, o microcontrolador realiza contagem de pulsos de clock.

Para gerar 2.5 V, o microcontrolador(para a frequência exemplo de 1kHz) realiza a contagem de 500 pulsos de clock no intervalo de *DutyCicle* e realiza, após isso, a contagem de 500 pulsos no período no qual a amplitude será de 0 V.Dessa forma, é gerado digitalmente o sinal PWM na placa Galileo.

Como dito no ínicio desta seção, placa Galileo é compatível com a placa Arduíno, tanto a nível de hardware quanto a nível de software. Daí, para executar a criação de um sinal PWM na placa galileu deve-se chamar a função *analogWrite(int porta, int valor)*.

A função *analogWrite(int porta, int valor)* recebe como parâmetros dois inteiros. O inteiro *porta* indica quais dos pinos digitais, habilitados para saída PWM, foi selecionado. O inteiro *valor* deve ser um inteiro entre 0 e 255.

```
1 //Comando para setar na porta digital 5 o valor 5*(127/255) = 2.5 Volts
2 analogWrite(5, 127);
```

A tensão DC que estará presente no pino digital segue a formula: 2.6

$$V_{dc} = \frac{5 * valor}{255} \tag{2.6}$$

2.4.2.2 Conversão analógico-digital

A placa Galileo utiliza para a conversão analógico-digital o circuito integrado *AD7298* [19]. O conversor analógico-digital AD7298 é um conversor de 12 bits e usa para a conversão a técnica de

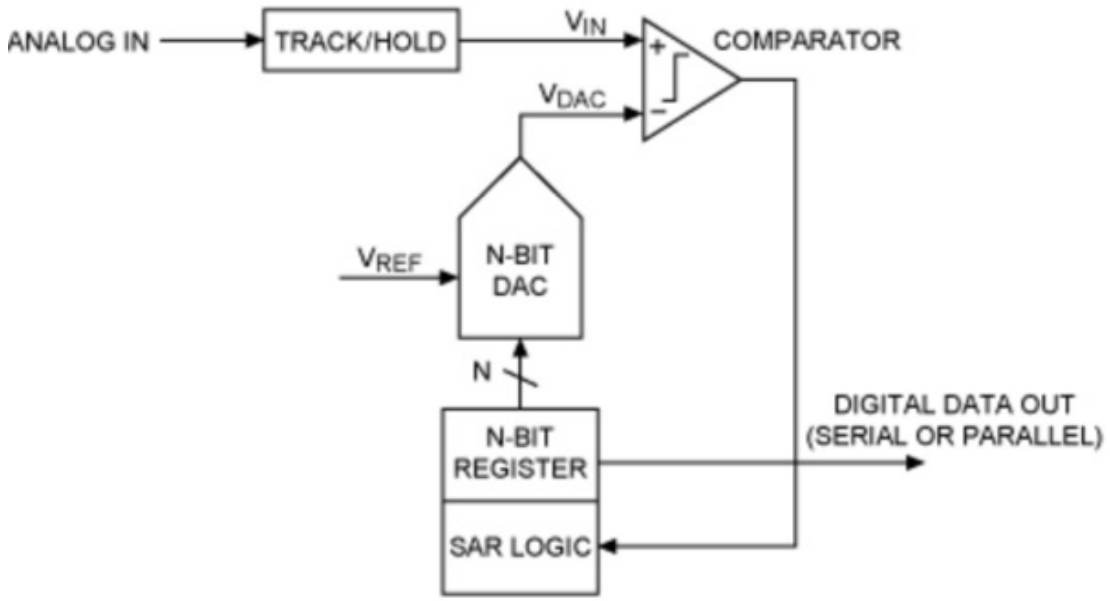

Figura 2.29: Figura esquemática do processo de conversão analógico para digital

aproximações sucessivas.

A figura 2.29 mostra uma figura esquemática para o processo de conversão analógico-digital utlizando a técnica de *aproximações sucessivas* e os termos chave para essa técnica são os seguintes:

- Registrador de aproximação sucessiva (SAR)
- Circuito de amostragem e retenção (Track and Hold)
- Tensão de entrada V_{IN}
- Tensão de referência V_{REF}
- Registrador de N bits (N-BIT REGISTER)
- Conversor digital para analógico de N bits(N-BIT DAC)
- Circuito Comparador

Num primeiro instante, o bit mais significante do conversor D/A é setado para 1, enquanto os outros N-1 bits são setados para 0. Essa configuração inicial dos N bits do conversor D/A força com que na saída exista 1/2 da tensão de referência V_{REF}, ou seja $V_{DAC} = 1/2\ V_{REF}$.

Caso a tensão V_{DAC} seja maior que a tensão de entrada V_{IN}, o bit mais significativo será mantido, caso o contrário, esse bit será setado no valor 0. Depois disso, registrador SAR grava o resultado obtido no comparador no bit avaliado. O processo se repete para os N bits,sempre

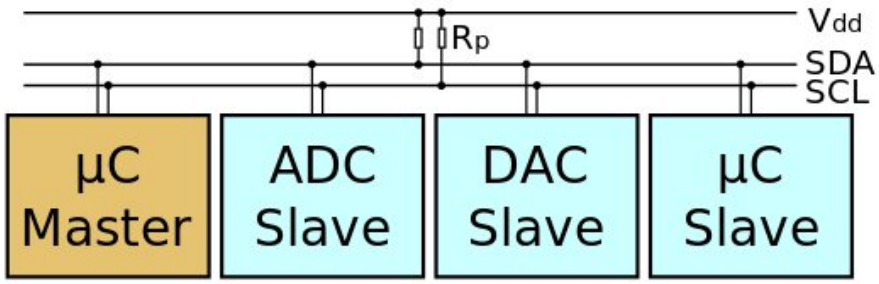

Figura 2.30: Figura esquemática do barramento serial I2C

comparando a tensão de entrada V_{IN} com a tensão de conversão V_{DAC}, fazendo com que a saída da conversão se aproxime progresivamente a cada iteração[20].

A eficiência do processo de conversão analógico-digital está intimamente ligada ao processo interno de conversão digital-analógico. Há diversos processos de conversão digital-analógico, entretanto, para todos, a quantidade de bits a serem convertidos influencia diretamente na linearidade do processo. Por isso se escolhe, em geral, um conversor digital-analógico de 12 bits.

O conversor A/D utilzado na placa Galileo, segundo seu respectivo datasheet [19], possui características implementadas que, entre outras incluem:

- Sensor de temperatura integrado para devidos ajustes às variações de parâmetros causados pela variação de temperatura
- Taxa de saída de conversões completadas superior a 1 MSPS (Million Samples Per Second)

2.4.2.3 Barramento Serial I2C

I2C(Inter-Intergrated-Circuit) é um protocolo de comunicação serial desenvolvida originalmente pela *NXP Semiconductor*. Ela permite a comunicação direta entre diversos componentes utilizando apenas três barramentos: um barramento para transmissão de bits dados - *Serial Data Line(SDA)* - um barramento para o sinal de clock - *Serial Clock Line(SCL)* e um barramento para o uso de um resistor de *pull-up* ligado diretamente uma tensão V_{dd} de 5V ou 3.3V. O endereçamento no protocolo I2C pode ser de 7 ou 10 bits. A velocidade de transmissão de dados variam de 10kbits/s - para o modo *low speed*- 400 kbits/s - para o modo *Fast mode* - e 3.4Mbit/s para o *modo Fast mode plus* [21].

O resistor *pull-up* serve para ter como valor alto de tensão(lógico 1) tanto o barramento de clock como o barramento de dados, Fig.2.30. Para trocar o valor lógico enviado nos barramentos, os dispositivos devem chavear suas respectivas conexões com os barramentos.

No protocolo I2C, sempre existem os dispositivos que agem como *dispositivos mestres*(Masters) e os dispositivos que agem como *dispositivos escravos*(Slaves).

Um *dispositivo mestre* pode escolher com qual dos *dispositivos escravos* ele deseja se comunicar

realizando a "mensagem de início".Após isso mandando os bits de endereço do *dispositivo escravo* são enviados no barramento de dados. É enviada, juntamente com uma mensagem do endereço, uma indicação, por parte do *dispositivo mestre* mostrando se ele deseja escrever ou ler do *dispositivo escravo*. Após isso, o *dispositivo escravo* deve enviar uma mensagem ACK para completar o estabelecimento da comunicação.Para enviar uma mensagem ACK, o *dispositivo escravo* seta o barramento de dados para o valor 0.

Tendo sido estabelecida a comunicação entre *dispositivo mestre* e *dispositivo escravo*, é incubência do *dispositivo escravo* enviar, a cada 8 bits recebidos, uma mensagem ACK.

Os *dispositivos mestres* sempre retem o controle do barramento de clock. Quando um *dispositivo mestre* faz com que o barramento de clock tenha o valor lógico 0, é indicado para os *dispositivos escravos* que eles devem setar o barramento de dados com um bit 0 ou 1.

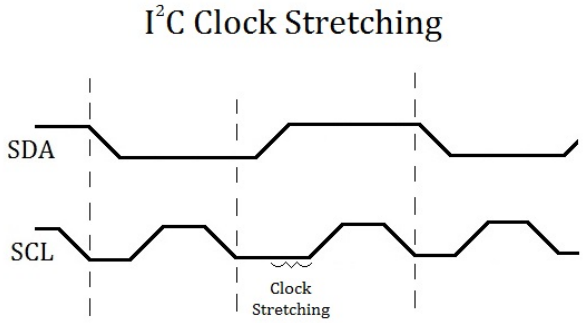

Figura 2.31: I2C - Clock Stretching

Para assegurar o recebimento dos dados, os *dispositivo escravos* podem, possivelmente, realizar o chamado *Clock Stretching*,, Fig 2.31, o qual consiste em manter o barramento de clock no nível 0, mesmo que o mestre o tenha setado para o nível. Isso é feito para, ampliar o tempo do processo de recebimento dos dados, por parte dos *dispositivo escravos*, para assegurar o sucesso de tal processo.

I2C oferece um bom suporte para a comunicação entre dispositivos eletrônicos que são acessados de forma ocasional. A vantagem competitiva da I2C sobre outros protocolos de comunicação de curta distância de baixa velocidade é que seu custo e complexidade não aumenta com o número de dispositivos no barramento.

Por outro lado, a complexidade dos componentes de software I2C de suporte pode ser significativamente mais elevada do que a de vários protocolos concorrentes (SPI e MicroWire, por exemplo) com uma configuração muito simples. Entretanto, seu modelo de endereçamento próprio, aliado com a forma de transferência simples de bytes para necessidades de comunicação simples.

O protocolo I2C é muito utilizado em projetos no placas Galileo utilizando a biblioteca: *Wire.h*

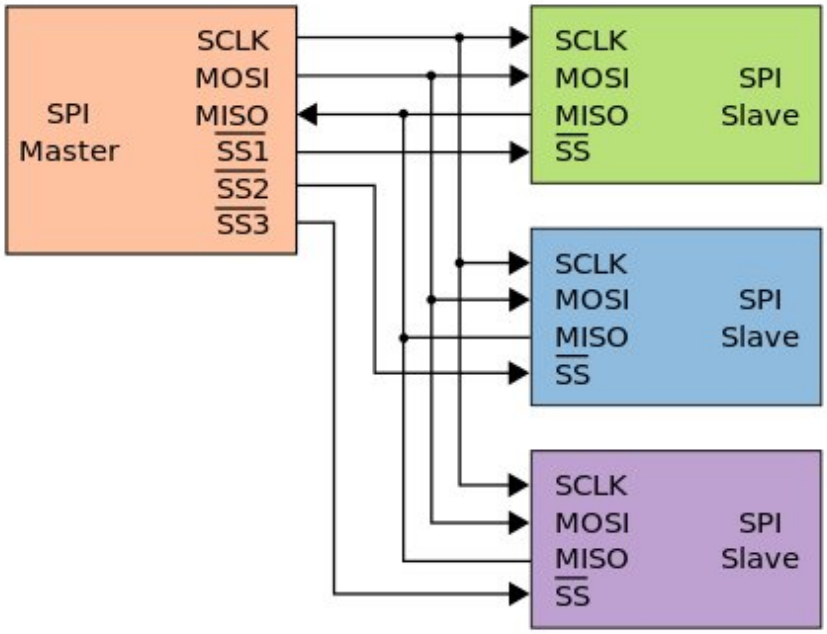

Figura 2.32: Barramento SPI - Um *dispositivo mestre* para *dispositivos escravos*

2.4.2.4 Comunicação serial SPI

Assim como o protocolo I2C, o protocolo de Interface Serial com Periféricos - Serial Peripheral Interface (SPI) - tem como utilidade a comunicação de curta distância entre dispositivos eletrônicos[22].

No protocolo SPI, há apenas um *dispositivo mestre* para vários *dispositivo escravos*. A comunincação entre os *dispositivo mestre* e os *dispositivos escravos* é *full-duplex*, ou seja, os dispositivos citados podem se comunicar entre si em ambas direções. Os pinos *dispositivo mestre* e nos *dispositivos escravos*, como mostrados na Figura 2.32, são os seguintes:

- SCLK: Barramento Serial para o sinal de Clock originado no *dispositivo mestre*
- MOSI: *Master Output Slave Input*; sinal originado no *dispositivo mestre*
- MISO: *Master Input Slave Output*;sinal originado em um dos *dispositivos escravos*
- SS: *Slave Select*; sinal originado no *dispositivo mestre*

O pino SS é utilizado pelo *dispositivo mestre* para selecionar com qual dos *dispositivos escravos* ele se comunicará (seja para receber mensagens ou enviar). Usualmente, quando um dos *dispositivos escravos* é selecionado, todos outros, pela lógica *tri-state* das entradas SS, assumem altas impedância de entrada - o que significa que virtualmente tais escravos estão desconectados do circuito com o *dispositivo mestre*.

Primeiramente, no começo da comunicação com o dispositivo selecionado, é configurado no *dispositivo mestre* a frequência do clock que sai da pino SCLK.

Após isso, começa a ocorrer a troca de bits entre os dispositivos. Para cada bit que o pino MOSI recebe, é também recebido um bit no pino MISO.

Comparado a outros protocolos de inter-comunicação, o protocolo SPI oferece uma das maiores taxas de saída de bits. Isso se deve, dentre outros fatores, a não limitação do tamanho da palavra binária transmitida. As taxas de transmissão são, em geral, da ordem de MHz, entretanto tal taxa é intimamente ligada a velocidade do clock no *dispositivo mestre*, podendo portanto ser livremente aumentada. Além disso, os *dispositivos escravos* não necessitam de um endereço único como no protocolo I2C, daí, todas fase de reconhecimento e estabelecimento de comunicação é facilitada. Entretanto, tais facilidades tornam o protocolo com dificil deburação de erros e não há controle de fluxo nem nos *dispositivos escravos*.

SPI é utilizado em muitas aplicações. Dentre elas, por exemplo:

- Aplicações com sensores:
  - Comunicação com sensores de temperatura
  - Comunicação com sensores de pressão
  - Comunicação com sensores de toque
- aplicações com tipos específicos de mémoria
  - Flash
  - EEPROM

Para fazer projetos com SPI na placa Galileo deve ser utilizada a biblioteca *SPI.h*

2.4.2.5 Porta Serial UART

UART significa *Universal Asynchronous Receiver/Transmitter* (Receptor/Transmissor Universal Assíncrono). Um dispositivo UART é um microchip que tem como responsabilidade controlar a comunicação de um computador ou microcontrolador conectados serialmente. Essencialmente, um dispositivo UART é a dispositivo intermediário entre interfaces seriais e paralelas[23].

A Figura 2.33 mostra um modelo simplicado do que consiste um dispositivo UART. Na parte esquerda da figura, são mostrados os pinos de comunicação paralela pelo barramento de dados (Data Bus). O pinos R/W é utilizado para setar entre modos de leitura e escrita (Read/Write). O pino CLK é o pino do sinal de clock. O pino INT é o pino usado para interrupção de software para avisar o sistema que há dados para serem lidos/escritos no dispositivo UART.

A Figura 2.34 mostra o chamado *frame* de dados da placa UART. O *frame* é composto de 10 bits. O primeiro bit é o bit de *start* utilizado para indicar o ínicio do envio ou recebimento de um byte (8 bits) de dados. O bit *stop* indica o fim do frame.

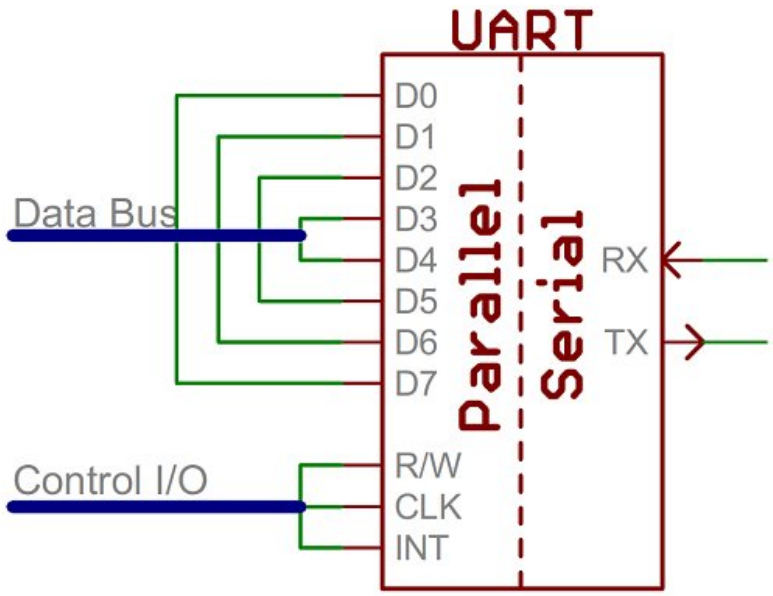

Figura 2.33: Modelo simplificado de um dispositivo UART

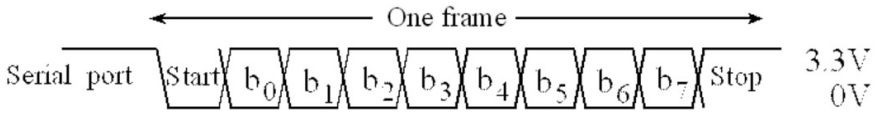

Figura 2.34: Frame UART para transmissão de 1 byte

Já a Figura 2.35 mostra em detalhes o processo que ocorre num dispositivo UART. Na figura, `UART_DR_D` é o *registrador de dados (Data Register)*, o qual é preenchido pela dispositivo que deseja realizar a comunicação utilizando o dispositivo UART. FIFO é a fila de recebimento(RX) ou transmissão de dados(TX). Ambas as filas tem 16 bits de tamanho. No caso da fila de recebimento de dados, 4 dos 16 bits são bits de flags para indicar erros na transmissão.

RXFE é uma flag que indica se que a fila de recebimento está vazia e RXFF é outra flag que indica que a fila de recebeminto está vazia. Quanto as filas de transmissão, TXEF indica que a fila está vazia e TXFF indica que a fila está cheia. UOTX e UORX são *shift register* são os responsáveis pela transformação da comunicação em série para paralela e vice-versa.

O processo de transmissão de dados é o seguinte:

- 1) Dados armazenados no registrador de dados são enviados para a fila
- Caso a fila esteja vazia(flag TX), a fila recebe os bits do registrador de dados
- 2) Os bits são enviados para o shift register UOTX, começando no b0 e sendo "shiftados"até o bit b7.
- 3) Os bits armazenados no UOTX são enviados de forma serial para o shift register receptor

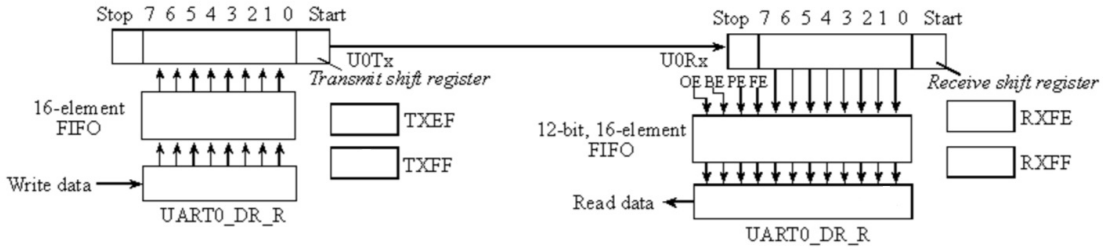

Figura 2.35: Modelo completo de um dispositivo UART

UORX.

- 4) Caso a fila de recepção esteja vazia (flag RX), os dados são colocados na pilha e lá permanecem até serem lidos.

UART é muito utilizado para projetos que requerem comunicação serial com Galileo ou projeto de Multiplas Entradas e Saída Única (MISO) ou projeto com Entrada Única e Saída Múltipla (SIMO). Para trabalhar com UART, deve usar a biblioteca *SoftwareSerial.h*

2.4.2.6 Memória Cache

Dentre as operações num sistema computacional, a operação mais demorada é o acesso à memória. Para evitar tais operações, é usada a chamada memória cache[24].

A memória cache faz parte da organização da memória de um sistema computacional. A memória num sistema computacional é organizada da seguinte forma, Fig.2.36:

- Memória de armazenamento(Storage Device - Memória ROM): Este nível de memória é o que possível mais espaço, entretanto é a memória que demanda mais tempo para ser modificada, por isso, em geral, nesse nível ficam armazenados sistemas operacionais, arquivos de BOOT do sistema, firwares, etc. A memória nesse nível não-volátil, o que significa que ela não é perdida ao se desligar o sistema.

- RAM(Random Access Memory): Este nível de memória é utilizado como memória principal. A memória RAM é de leitura e escrita. Essa memória é utilizada pelo CPU para armazenar e ler dados, arquivos e programas que estão sendo utilizados no momento. A memória RAM é uma memória volátil, o que significa que o conteúdo armazenado nela é perdido após o desligamento do sistema.

- A memória cache é a parte da memória utilizada pela unidade de processamento central (CPU) de um computador para reduzir o tempo médio necessário para ler ou escrever aos dados a partir da memória principal. A memória cache é uma memória menor, mais rápida que armazena cópias dos dados de localizações de memória principais utilizados com

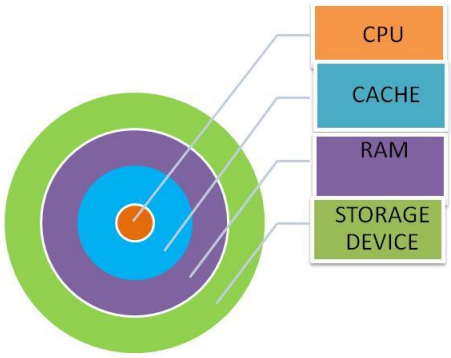

Figura 2.36: Organização da memória num sistema computacional

frequência para evitar a repetição de acessos lentos. A maioria dos processadores têm diferentes caches independentes, incluindo instruções e dados caches, onde o cache de dados é normalmente organizadas como uma hierarquia de níveis mais cache (L1, L2, etc).

- CPU: Na CPU está armazenada toda arquitetura de instruções do sistema computacional. A CPU é responsável pela gerência de todos processos que ocorrem no computador e ela utiliza a memória cache para realizar a maior parte de suas operações

Para toda operação que a CPU executa a qual necessita de certo dado da memória, é sempre verificadp, primeiramente, se o dado já se encontra na memória cache. Caso o dado não se encontre na cache, é solicitado dos níveis mais baixos da memória o dado em questão. Caso o dado já se encontre na cache, ele é lido e processado rapidamente pela CPU.

Microcontroladores simples, em geral, não possuem a memória cache, visto que toda sua estrutura é simplificada. No caso da placa Galileo e placas similares, a memória cache é necessária, visto que tais sistemas podem, inclusive, executar sistemas operacionais e tem grande quantidade de memória de armazenamento.

Atualmente, vem-se dividindo a mémoria cache em níveis: cache L1, cache L2, cache L3, etc. Tal divisão é feita para amplificar o efeito de manter na cache os dados de memória usualmente acessados. A cache L1 contêm os dados acessados mais frequentemente, a cache L2 contêm os dados acessados frequentemente, mas não tanto quanto os dados na cache L1, etc.

No caso da placa Galileo, há apenas um nível de cache: a cache L1 com 16 KBytes, como mostrado na seção 2.4.2.

2.4.2.7 Memória SRAM

Memória SRAM (Static Random Access Memory) é o tipo de memória de acesso aleatório geralmente utilizado no nível de memória cache. Ser de de acesso aleatório significa que qualquer porção da memória é acessada num tempo igual. SRAM é uma memória estática, o que significa

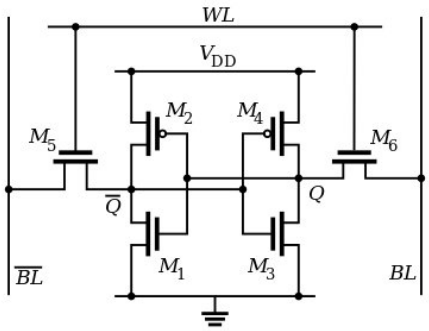

Figura 2.37: Estrutura da célula de memória SRAM

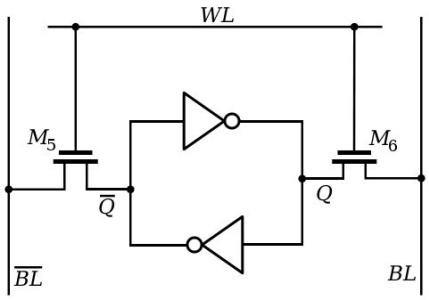

Figura 2.38: Estrutura da célula SRAM com dois inversores

que o dados se manterá armazenado durante um largo intervalo de tempo depois do desligamento do sistema.

A memória SRAM é construída utilizando com flip-flops com transistores MOSFETs. A Figura 2.37 mostra a estrutura básica de armazenamento de uma célula de um bit da memória SRAM.

Resumidamente, os transistores M1, M2, M3 e M4 são os responsáveis por guardar o bit[25]. A estrutura do circuito formada por M1, M2, M3 e M4 realiza a realimentação do bit, sendo responsável pelo qualidade de memória estática que a SRAM possui. A Figura 2.38 mostra como os transistores M1, M2, M3 e M4 podem ser vistos como um par de emissores. Quando um Q, na Figura 2.38, é igual a 1, seu oposto, $\overline{Q} = 0$, é criado na saida do inverso e o sinal Q = 1 é realimentado pelo segundo inversor.

Dessa maneira, a estrutura da célula de um bit de memória SRAM torna desnecessário *re-carregamento* do dado armazenado. Os transistores M5 e M6 são usados para ler ou escrever da célula de memória por meio das linhas de bit BL e $\overline{BL}$. Tal processo de leitura ou escrita pode ser realizado, em média, em 2ns, velocidade a qual é bastante alta para sistemas computacionais.

A memória SRAM é utilizada nós mais variados ambientes como: computadores pessoais, microcontroladores, FPGAs, etc. Na placa Galileo, existem 512 Kbytes de SRAM integrados, tornando a placa Galileo altamente eficiente no tocante ao acesso e atualização da memória.

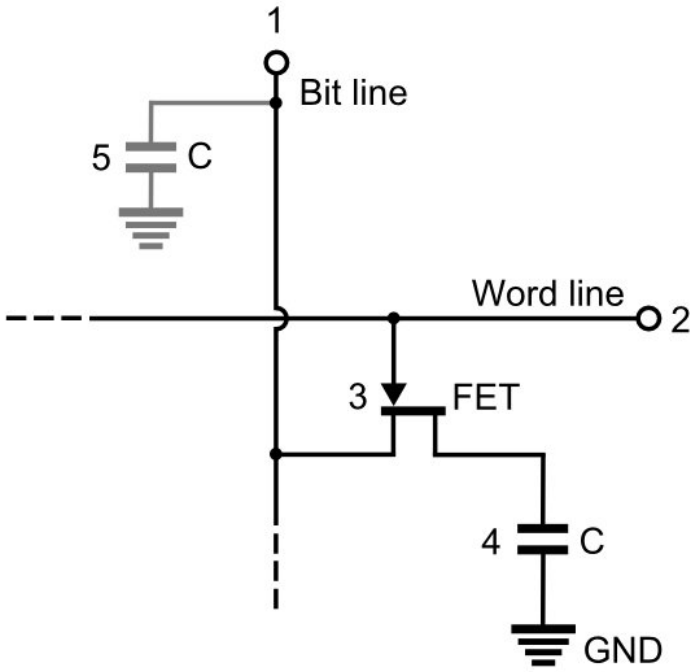

Figura 2.39: Estrutura da célula de memória DRAM

2.4.2.8 Memória DRAM

Dynamic Random Access Memory (DRAM) é uma memória de acesso aleatório como a SRAM. Ao contrário da memória SRAM, a memória DRAM é uma memória *dinâmica*, o que significa que os dados armazenados precisam ser periodicamente recarregados.

Os bits, na mémoria DRAM, são armazenados numa estrutura de um capacitor juntamente com um transistor. O capacitor estar carregado significa o bit 1, e o capacitor está descarregado significa o bit 0. Na Figura 2.40 é mostrada a estrutura de uma célula de memória DRAM. O capacitor marcado pelo número 4 é onde o bit é armazenado.

O processo de escrita no bit é da seguinte forma:

- 1) Na linha marcada pelo número 1(Bit Line) é escrito um bit lógico 0 ou 1(0 ou +Vcc Volts)
- 2) A linha marcada pelo número 2 ativa o transistor conectando a Bit line com o capacitor C (marcado pelo número 4)

O processo de leitura é feita da mesma forma que o processo de escrita, entretanto, a Bit line possui capacitância parasita apreciável. Na figura, essa capacitância é marcada pelo número 5. Tal capacitância parasita diminui a velocidade do processo de leitura por tomar parte da carga armazenada no capacitor marcado pelo número 4 para si.

O descarregamento natural dos capacitores, ainda que em circuito aberto, e a existência de

capacitores parasitas na célula de memória trazem a necessidade de circuitos responsáveis por recarregar, a cada leitura, as células DRAM.

O tempo médio de leitura na memória DRAM é de 64 ns. Pelo tempo de leitura e pela necessidade de recarregamento, em geral, a memória DRAM é usada para memórias menos acessadas.

A placa Galileo possui 256 MByte de memória DRAM gerenciados pelo sistema operacional.

2.4.2.9 Memória EEPROM

EEPROM, Electrically Erasable Programmable Read-Only Memory é uma memória não volátil, o que significa que os dados não são apagados após o desligamento do sistema. A memória EEPROM é similar à memória FLASH. Assim como ela, a memória EEPROM escrita aproximadamente 100.000 vezes. A principal vantagem que a memória EEPROM apresenta em relação a memória FLASH é que ela deve escrever em bytes individualmente, enquanto na memória FLASH é necessário escrever um setor inteiro para alterar bytes individuais. Tal característica torna a memória FLASH mais rápida e com vida-útil menor que a memória EEPROM.

Na placa Galileo há 11 Kbytes memória EEPROM. A EEPROM pode ser programada na placa Galileo com a biblioteca *EEPROM.h*.

2.4.2.10 Clock de tempo real - RTC

Um clock de tempo (RTC) é um clock comum de um sistema computacional com a funcionalidade de ter armazenado nele tempo atual, mesmo que o sistema esteja desligado. Quase todos equipamentos eletrônicos atuais, como computadores, celulares, etc, possuem um clock de tempo real integrado. O tempo atual pode ser adquirido com outros equipamentos além do RTC, entretanto, o RTC têm as seguintes vantagens:

- Baixo consumo de energia
- O fato de ser um sistema indepedente do sistema central, faz com este tenha seu processamento livre para outras tarefas

2.4.2.11 Barramento Mini PCI-Express

Mini PCI-Express é um barramento de alta velocidade de transmissão de dados com 52 pinos. Por meio desses 52 pinos, existem as seguintes conexões:

- Conexões para o barramento PCI Express x1
- Conexões para USB 2.0
- Conexões para SMBus
- Conexões para LEDs de diagnostico de conexões wireless

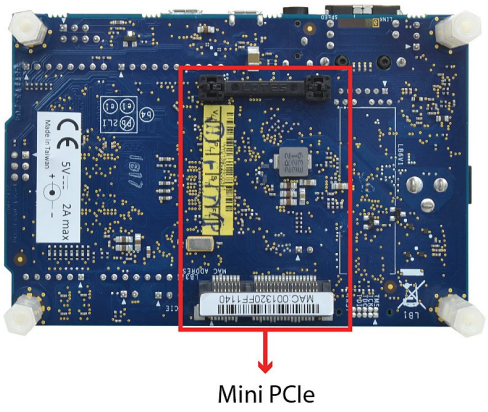

Figura 2.40: Estrutura da célula de memória DRAM

- Conexões para SIM Card
- Conexões para outras extensões PCI
- Saída de 1.5V e 3.3 V

2.4.2.12 USB

USB ou *Universal Serial Bus* é um padrão cabos, conectores e protocolos. O propósito do USB é padronizar a comunicação com equipamentos periféricos como teclados, cameras, impressoras, telefones, etc. USB já passou por três padronizações:

- USB 1.0 com velocidade máxima de transmissão de dados de 12 Mbits/s
- USB 2.0 com velocidade máxima de transmissão de dados 480 Mbits/s
- USB 3.1 com velocidade máxima de transmissão de 10 dados Gbits/s

Para comunicação com periféricos, USB já tem conseguido subsituir com sucesso a comunicação serial e paralela.

A topologia USB é assimétrica em formatado de estrela com um dispositivo central (Host), como mostrado no exemplo da Figura 2.41.

Quando um novo equipamento é conectado, o sistema operacional do dispositivo central, a placa galileo por exemplo, detecta a nova conexão e solicita o driver do equipamento para possibilitar a comunicação. Como mostrado na Figura 2.42, os cabos e conexões USB obedecem os padrões de duas classes: a classe A e a classe B.

Quando o dispositivo central é ligado, é definido para cada dispositivo conectado um endereço. Tal processo inicial é chamado de *enumeração*. Durante a *enumeração*, é também solicitado a cada equipamento o tipo de transferência de dados a ser realizado com ele:

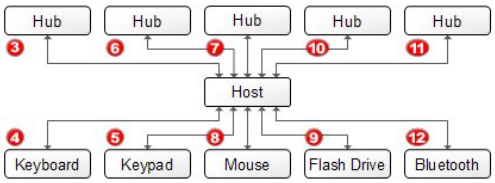

Figura 2.41: Exemplo: Topologia Estrela USB

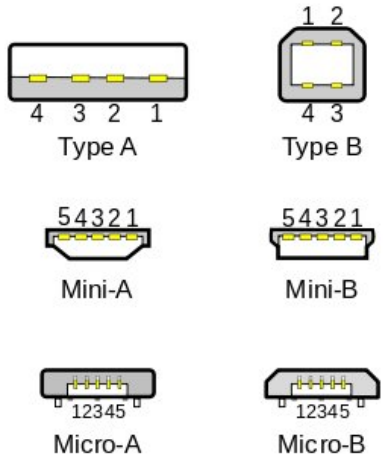

Figura 2.42: Pinos USB

- Transferência de dados por meio de interrupção: Transferência de dados pouco frequente e de baixa quantidade, como transferência com teclado e mouse. Nesse caso, vale a pena interromper o sistema operacional.
- Transferência de dados por meio de pacotes: Transferência de dados pouco frequentes e de grande quantidade de dados, como, por exemplo, a transferência realizada para impressoras. Nesse caso, um bloco de dados e transferido de uma vez só pela porta USB.
- Transferência de dados isócrono(tempo real): Transferência de dados frequente e contínua, como as necessárias num alto falante.

Para cada uma das formas de transferência de dados supracitadas, é reservado pelo USB a largura de banda necessária em frames de largura de banda.

2.4.2.13 JTAG

JTAG(Joint Test Action Group) é a padronização IEEE-1149.1 usada para testes de circuitos impressos. JTAG foi criada para ajudar no problema da crescente dificuldade de testar circuitos associada com a crescente diminuição dos tamanhos do circuitos. Como mostrado na Figura 2.43, a implementação mais simples de JTAG requer 4 fios para sinalização:

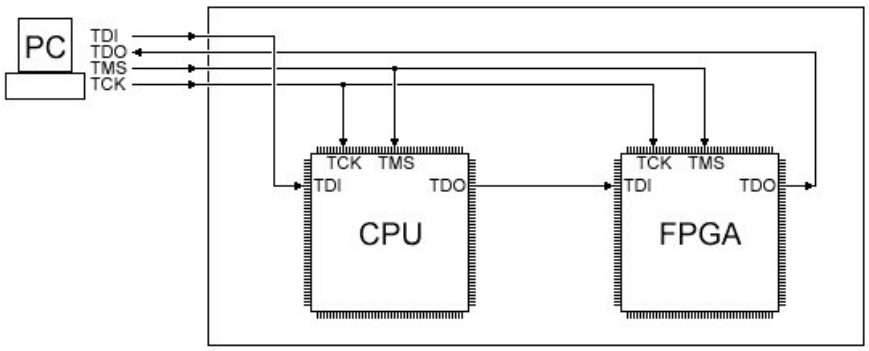

Figura 2.43: JTAG monitorando a conexão de um CPU com uma FPGA

Fonte: Source of the image.

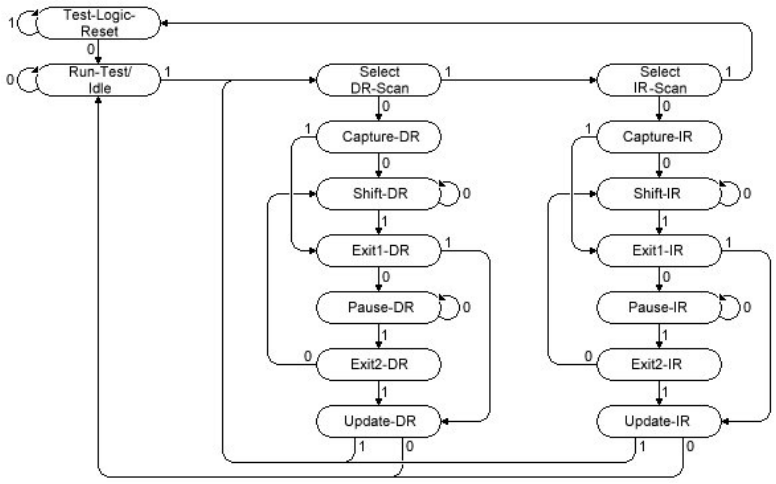

Figura 2.44: Máquina de estados - JTAG

- TDI: Pino para sinal de entrada para a query de teste.
- TDO: Pino para sinal de saída para a query de teste.
- TCK: Sinal do relógio de sicronização do JTAG. Todos outros sinais(TDI, TDO, TMS) são síncronos a esse sinal.
- TMS: Sinal para controlar o estado da máquina de estados interna ao JTAG, a qual tem 16 estados distintos, como mostrado na figura 2.44.

Na máquina de estados mostrada em 2.44, geralmente a JTAG é levada para os estados *Shift-DR*, em primeira instância, e, após isso, levada para o estado *Shift-IR*, onde o dado é coletado. *Shift-DR* e *Shift-IR* tem o mesmo tamanho N de bits. Por exemplo, se *Shift-DR* e *Shift-IR* tiverem 6 bits de tamanho, após 6 clock realizados no TCK, o dado que chegou no *Shift-IR* chega no *Shift-DR*.

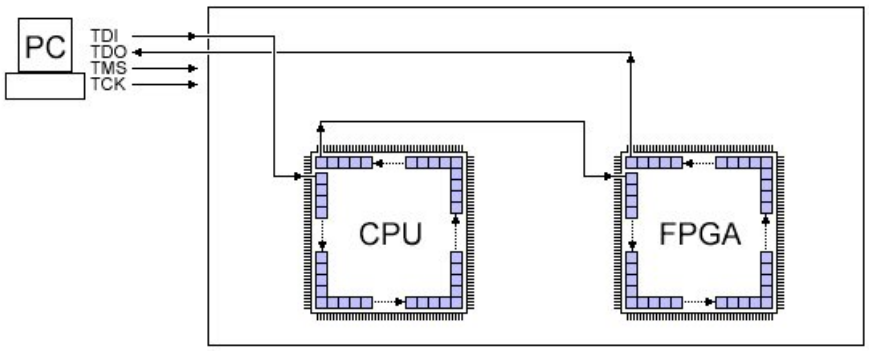

Figura 2.45: Fluxo de dados num debug JTAG

A figura 2.45 mostra fluxo de dados de ida e volta num debug JTAG: JTAG -> CPU -> FPGA. O dado saí pelo pino TDI, percorre a CPU e a FPGA e volta no pino TDO sendo tudo isso controlado pelo pino TMS tendo todos esses pinos sincronizados pelo pino TCK.

A verificação de entrada e saída, incluíndo o tempo de tais eventos, com testes JTAG tornou possível testes complexos em circuitos integrados.

A placa Galileo, como grande partes dos circuitos integrados atuais, possui 9 pinos próprio para debug com JTAG.

Capítulo 3

Desenvolvimento

3.1 Introdução

Neste capítulo, é apresentado um modelo de curso prático para a disciplina *Algoritmos e Programação de Computadores* tendo como base a teoria de aprendizagem por masterização de habilidades e como base metodológica a metodologia ágil de desenvolvimento de projetos *Scrum*.

3.2 Proposta de Curso

REFERÊNCIAS BIBLIOGRÁFICAS

[1] S. BURFOOT J., G. D. e. H. C. B. *A Teacher's Guide to Intel Galileo*. Buildind C5B, Macquarie University, North Ryde, NSW, 2109, 2015.

[2] SOVIC, A.; JAGUST, T.; SERSIC, D. How to teach basic university-level programming concepts to first graders? In: *Integrated STEM Education Conference (ISEC), 2014 IEEE*. [S.l.: s.n.], 2014. p. 1–6.

[3] COTO, M.; MORA, S.; ALFARO, G. Giving more autonomy to computer engineering students: Are we ready? In: *IEEE Global Engineering Education Conference, EDUCON 2013, Berlin, Germany, March 13-15, 2013*. [s.n.], 2013. p. 618–626. Disponível em: <http://dx.doi.org/10.1109/EduCon.2013.6530170>.

[4] CELETI, F. R. Origem da educação obrigatória: Um olhar sobre a prússia. *Revista Saber Acadêmico*, v. 1, n. 1, p. 29–33, June 2012.

[5] OLIVER, J.; TOLEDO, R. On the use of robots in a pbl in the first year of computer science / computer engineering studies. In: *Global Engineering Education Conference (EDUCON), 2012 IEEE*. [S.l.: s.n.], 2012. p. 1–6. ISSN 2165-9559.

[6] F. de O. V. Crescimento, evolução e o futuro dos cursos de engenharia. *Revista de Ensino de Engenharia*, v. 24, n. 2, p. 3–12, December 2005.

[7] R., W. *Alta taxa de desistência na universidade causa déficit de engenheiros*. Setembro 2013. [Online; posted 4-Setembro-2013].

[8] LAHTINEN, E.; ALA-MUTKA, K.; JäRVINEN, H.-M. A study of the difficulties of novice programmers. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 37, n. 3, p. 14–18, jun. 2005. ISSN 0097-8418. Disponível em: <http://doi.acm.org/10.1145/1151954.1067453>.

[9] ESCUDERO, M. R.; HIERRO, C. M.; PABLO, A. Pérez de Madrid y. Using arduino to enhance computer programming courses in science and engineering. In: *EDULEARN13 Proceedings*. [S.l.]: IATED, 2013. (5th International Conference on Education and New Learning Technologies), p. 5127–5133. ISBN 978-84-616-3822-2. ISSN 2340-1117.

[10] TUCKER, P. *Linking teacher evaluation and student learning*. Alexandria, VA: Association for Supervision and Curriculum Development, 2005. ISBN 1-4166-0032-9.

[11] C., D. Defining a 21st century education. *The Center for Public Education*, v. 1, n. 1, p. 1–79, July 2009.

[12] GOEL, S. et al. Collaborative teaching in large classes of computer science courses. In: *Eighth International Conference on Contemporary Computing, IC3 2015, Noida, India, August 20-22, 2015*. [s.n.], 2015. p. 397–403. Disponível em: <http://dx.doi.org/10.1109/IC3.2015.7346714>.

[13] K.R, S. Competências do século 21. *Revista Pesquisa e Debate em Educação*, v. 4, n. 2, p. 15–30, August 2014.

[14] HUITT, W. Bloom et al.'s taxonomy of the cognitive domain. *Educational psychology interactive*, v. 22, 2004.

[15] KALISH, S.; MAHAJAN, V.; MULLER, E. Waterfall and sprinkler new-product strategies in competitive global markets. *international Journal of research in Marketing*, Elsevier, v. 12, n. 2, p. 105–119, 1995.

[16] MELNIK, G. *Agile 2008 August 4-8, 2008, Toronto, Ontario, Canada*. Los Alamitos, Calif: IEEE.Compurter Society, 2008. ISBN 978-0-7695-3321-6.

[17] INTEL. *DataSheet Intel Galileo Gen 2 Development Board*. [S.l.], 2014.

[18] SEDRA., A. S.; SMITH, K. C. *Microeletronics Cicuits*. [S.l.]: Oxford University Press, 2004.

[19] DEVICES, A. *DATASHEET AD7298*. One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A., 2011.

[20] BAKER, R. J. *CMOS Circuit Design, Layout, and Simulation, 3rd Edition (IEEE Press Series on Microelectronic Systems)*. [S.l.]: Wiley-IEEE Press, 2010.

[21] HIMPE, V. *Mastering the IšC bus*. Susteren: Elektor International Media, 2011. ISBN 978-0-905705-98-9.

[22] RUSSELL, R. C. J. *Serial peripheral interface bus*. Place of publication not identified: Book On Demand Ltd, 2012. ISBN 5513504936.

[23] OSBORNE, A. *An introduction to microcomputers*. Berkeley, Calif: Osborne/McGraw-Hill, 1980. ISBN 0-931988-34-9.

[24] HENNESSY, J. *Computer architecture : a quantitative approach*. Waltham, MA: Morgan Kaufmann, 2012. ISBN 978-0-12-383872-8.

[25] ISHIBASHI, K. *Low power and reliable SRAM memory cell and array design*. Berlin New York: Springer, 2011. ISBN 978-3-642-19567-9.

ANEXOS