UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

AULAS GALILEU

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

Prof. Fulano de Tal 2, ENE/UnB _____
Co-orientador

Prof. Fulano de Tal 3, EESC/USP _____
Examinador externo

Prof. Fulano de Tal 4, ENE/UnB _____
Examinador interno

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 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 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 da 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 nos 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 Placa Intel ® Galileo

A placa Intel® Galileo é uma *placa de desenvolvimento* com microcontrolador baseado processador Intel® Quark SoC X1000[10]. 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 2.1 e 2.2 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.

2.1.1 Pinagem da placa Intel ® Galileo

Os pinos da placa Galileo nas partes frontal e traseira são mostrados nas figuras 2.1 e 2.2

A descrição de cada um desses pinos é a descrita na tabela 2.1:

Tabela 2.1: 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

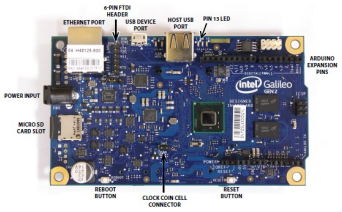

Figura 2.1: Descrição dos pinos da placa Galileo - parte frontal[1]

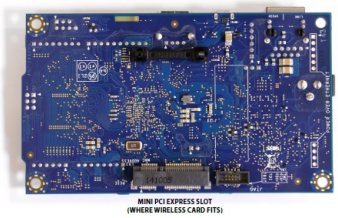

Figura 2.2: Descrição dos pinos da placa Galileo - parte traseira[1]

2.1.2 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***
- 10 pinos padrões ***JTAG*** para debug
- 256 MBytes de memória ***DRAM***
- 11 KBytes de memória ***EEPROM***

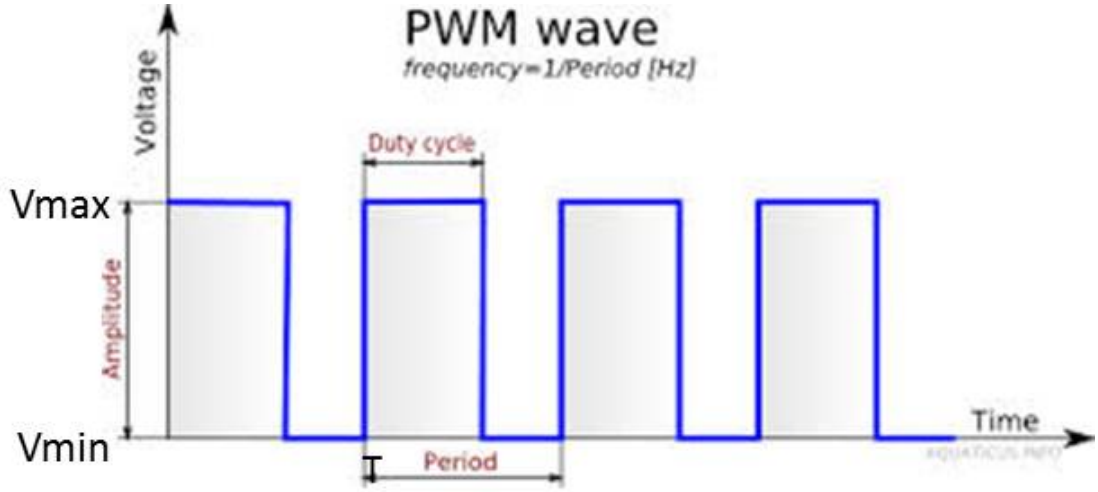

Figura 2.3: Sinal PWM

2.1.2.1 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[11]

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 2.3. 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

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

2.1.2.2 Conversão analógico-digital

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

A figura 2.4 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:

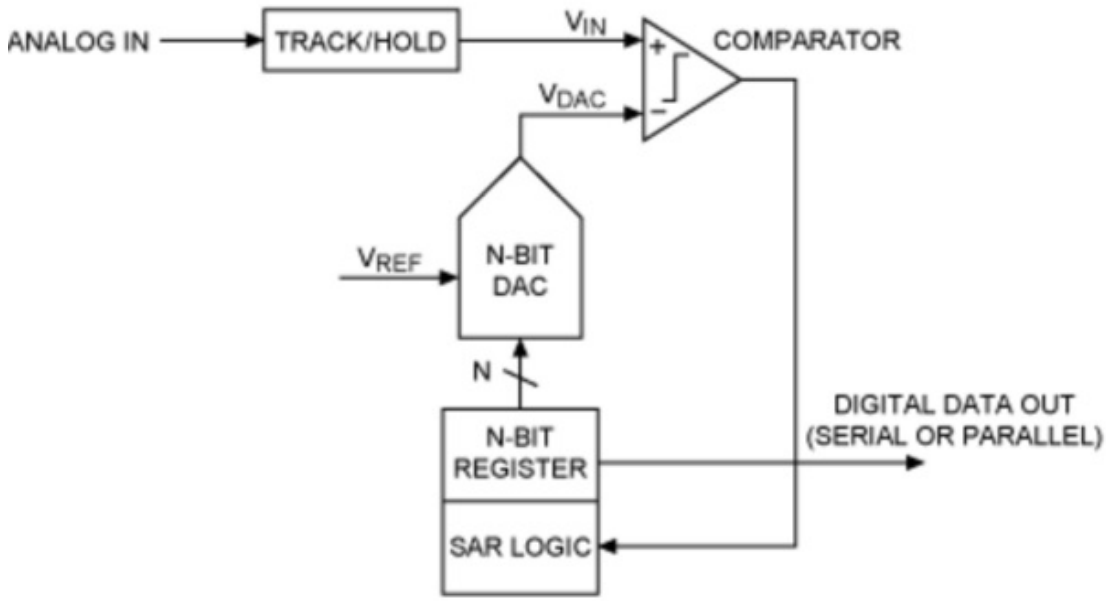

Figura 2.4: Figura esquemática do processo de conversão analógico para digital

- 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 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[13].

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, entre-

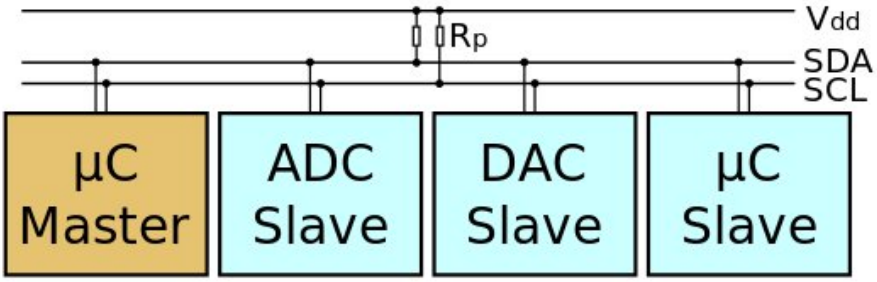

Figura 2.5: Figura esquemática do barramento serial I2C

tanto, 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 [12], 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)

2.1.2.3 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* [14].

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.2.5. 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.

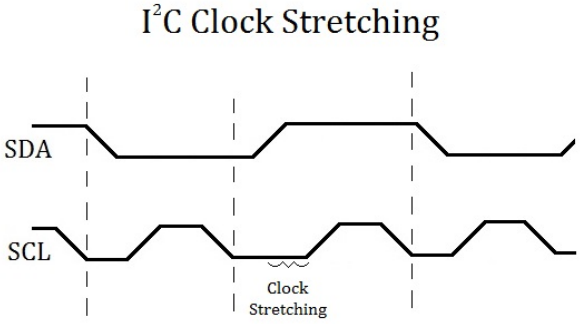

Figura 2.6: I2C - Clock Stretching

Para assegurar o recebimento dos dados, os *dispositivo escravos* podem, possivelmente, realizar o chamado *Clock Stretching*,, Fig 2.6, 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*

2.1.2.4 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[15].

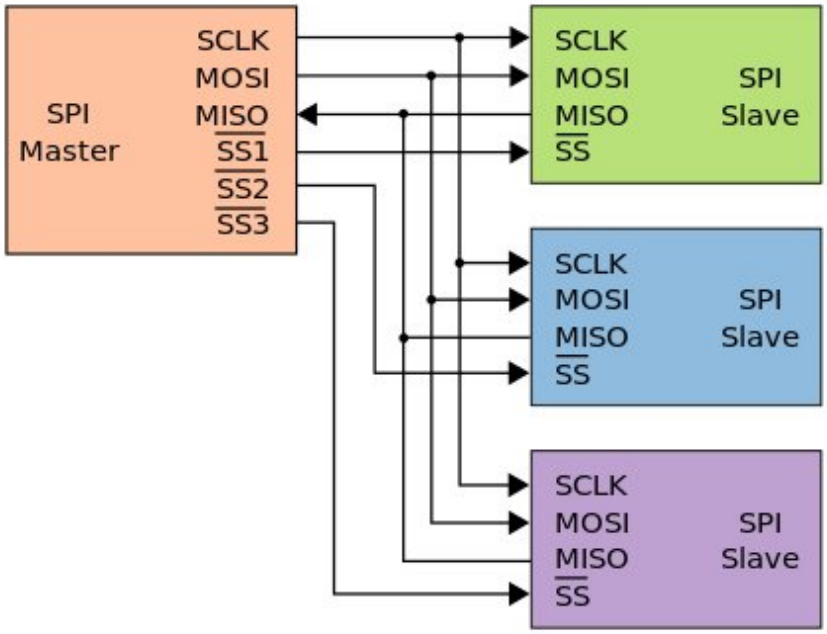

Figura 2.7: Barramento SPI - Um *dispositivo mestre* para *dispositivos escravos*

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 2.7, 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*

2.1.2.5 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[16].

A Figura 2.8 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 2.9 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.

Já a Figura 2.10 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.

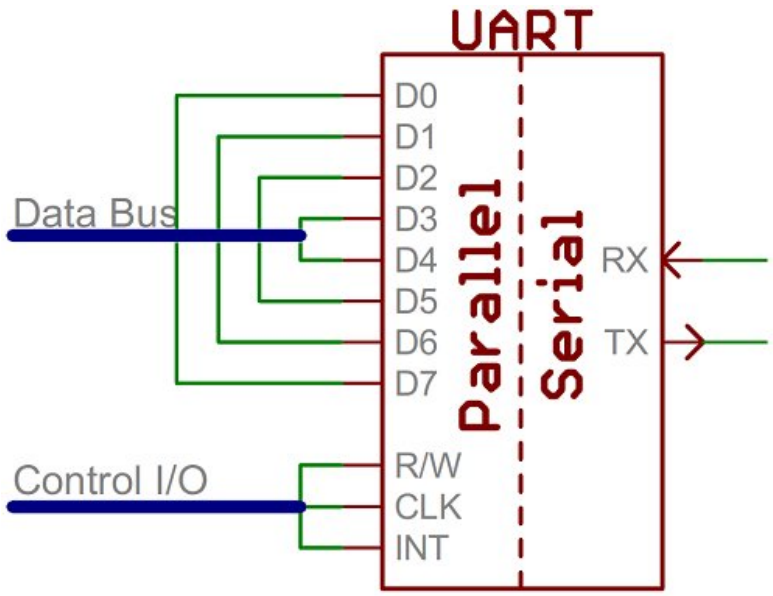

Figura 2.8: Modelo simplificado de um dispositivo UART

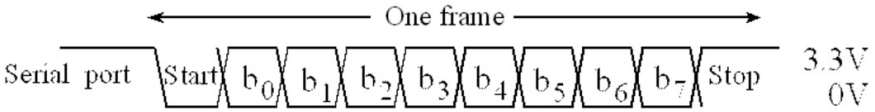

Figura 2.9: Frame UART para transmissão de 1 byte

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 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

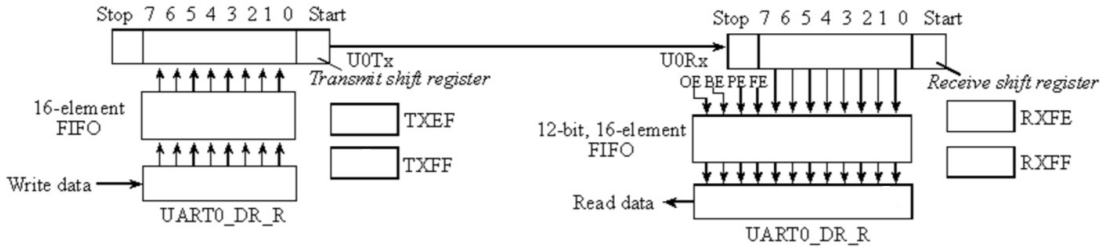

Figura 2.10: Modelo completo de um dispositivo UART

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*

2.1.2.6 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[17].

A memória cache faz parte da organização da memória de um sistema computacional. A memória num sistema computacional é organizada da seguinte forma, Fig.2.11:

- 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 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

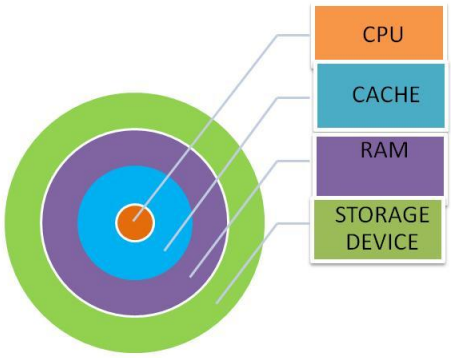

Figura 2.11: Organização da memória num sistema computacional

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 2.1.2.

2.1.2.7 Memória SRAM

Memória SRAM (Static Random Access Memory) é o tipo de memória geralemte utilizado no nível de memória cache. É uma memória volátil, apesar de exibir certar remanescência para o caso de desligamento.

A memória SRAM é construída utilizando transistores MOSFETs. A Figura 2.12 mostra a estrutura básica de armazenamento de bit da memória SRAM.

Resumidamente, os transistores M1, M2, M3 e M4 são os responsáveis por guardar o bit[18]. Os transistores M5 e M6 são usados para ler ou escrever da célula de memória por meio das linhas de bit BL e B^{-}L. Tal processo de leitura ou escrita pode ser realizado, em média, em 70 a 100 ns, velocidade a qual é bastante alta para sistemas computacionais.

A memória SRAM é utilizada nós mais variados ambientes como: computadores pessoais,

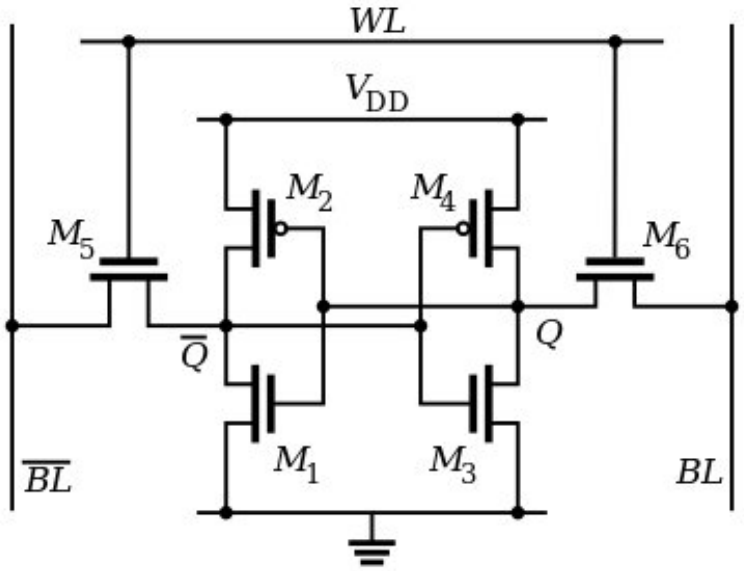

Figura 2.12: Estrutura CMOS SRAM

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.

2.1.2.8 Memória DRAM

2.1.2.9 Memória EEPROM

2.1.2.10 Clock de tempo real - RTC

2.1.2.11 Barramento PCI-Express

2.1.2.12 USB Host e USB Client

2.1.2.13 Pinos JTAG

Capítulo 3

Desenvolvimento

Resumo opcional.

3.1 Introdução

Na introdução deverá ser feita uma descrição geral da metodologia que foi seguida para o desenvolvimento. A seguir, é feita a descição do sistema desenvolvido.

Deve-se ressaltar que equações fazem parte do texto, devendo receber pontuação apropriada e ser numerada. Alguns exemplos são mostrados na seção 3.3.

3.2 Arquitetura geral

3.3 Classificador estatístico de padrões

O classificador automático de padrões utiliza o princípio da menor distância no processo de associação de dados. Assim, sendo $\mathbf{x}$ o vetor de características extraídas de uma imagem e $\mathbf{P_x}$ sua matriz de covariâncias respectiva, utiliza-se

$$d_i = (\mathbf{x} - \mathbf{p}_i)^T (\mathbf{P_x} + \mathbf{P_{p_i}})^{-1} (\mathbf{x} - \mathbf{p}_i) \tag{3.1}$$

como métrica para a distância estatística de $\mathbf{x}$ e um padrão de características $\mathbf{p}_i$ e matriz de covariâncias $\mathbf{P_{p_i}}$. Esta métrica é conhecida também pela denominação "distância de Mahalanobis". A distância definida pela Eq. (3.1) segue distribuição χ^2_n, em que n é a dimensão da base do vetor $\mathbf{x}$. Assim sendo, no caso específico de $n = 3$, o padrão associado ao vetor $\mathbf{x}$ é dito casado com o padrão $\mathbf{p}_i$ com 5% de margem de erro se

$$d_i \leq 7,815. \tag{3.2}$$

A seguir é mostrado como o modelo Latex apresenta os comandos \section \subsection e \subsubsection. Por questão de estilo, o texto deve ser organizado de modo a se evitar o uso de \subsubsection.

3.4 Seção

Meu texto da seção.

3.4.1 Sub-seção

Meu texto da sub-seção.

3.4.1.1 Sub-sub-seção

Meu texto da sub-sub-seção.

Se necessário, use notas de rodapé [1]

[1] Essa é uma nota de rodapé.

Capítulo 4

Resultados Experimentais

Resumo opcional.

4.1 Introdução

Na introdução deverá ser feita uma descrição geral dos experimentos realizados.

Para cada experimentação apresentada, descrever as condições de experimentação (e.g., instrumentos, ligações específicas, configurações dos programas), os resultados obtidos na forma de tabelas, curvas ou gráficos. Por fim, tão importante quando ter os resultados é a análise que se faz deles. Quando os resultados obtidos não forem como esperados, procurar justificar e/ou propor alteração na teoria de forma a justificá-los.

4.2 Avaliação do algoritmo de resolução da equação algébrica de Riccati

O algoritmo proposto para solução da equação algébrica de Riccati foi avaliado em diferentes máquinas. Os tempos de execução são mostrados na Tabela 4.1. Nesta tabela, os algoritmos propostos receberam a denominação CH para Chandrasekhar e $CH + LYAP$ para Chandrasekhar com Lyapunov. As implementações foram feitas em linguagem *script* MATLAB.

Observa-se que o algoritmo $CH + LYAP$ apresenta tempos de execução superiores com relação ao algoritmo CH. Entretanto, era esperado que o algoritmo CH fosse mais rápido. Este resultado

Tabela 4.1: Tempos de execução em segundos para diferentes máquinas

Algoritmo	**Laptop 1.8 GHz**	**Desktop PIII 850 MHz**	**Desktop MMX 233**	**Laptop 600 MHz**
Matlab ARE	649,96	1.857,5	7.450,5	9.063,9
CH	259,44	606,4	2.436,5	2.588,5
$CH + LYAP$	357,86	952,9	3.689,2	3.875,0

se justifica pelo fato de o algoritmo CH fazer uso de funções embutidas do MATLAB. Já o algoritmo $CH + LYAP$ faz uso também de funções *script* externas, aumentando bastante seu tempo computacional.

Capítulo 5

Conclusões

Este capítulo é em geral formado por: um breve resumo do que foi apresentado, conclusões mais pertinentes e propostas de trabalhos futuros.

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

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

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

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

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

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

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

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

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

ANEXOS

I. DIAGRAMAS ESQUEMÁTICOS

II. DESCRIÇÃO DO CONTEÚDO DO CD