UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPTO. DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Trabalho de Conclusão de Curso

Implementação de um simulador Arduino baseado em Android

Autor: Kollins Gabriel Lima, n^o. USP 9012931

Orientador: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2018

Resumo

Texto em **UM PARÁGRAFO** apenas - deve conter tudo resumidamente (introdução, método(s), resultados e conclusões), de tal forma que seja possível compreender a proposta e o que foi alcançado.

Palavras-Chave: palavra1, palavra2, palavra3, palavra4, palavra5.

Abstract

Abstract, abstract.

Keywords: keyword1, keyword2, keyword3, keyword4, keyword5.

Lista de Figuras

(Se houver...)

Lista de Tabelas

(Se houver...)

Siglas (Se houver...)

MVC	*Model-View-Controller* - Modelo-Visão-Controlador
POO	Programação Orientada a Objetos
UI	*User Interface* - Interface do Usuário
UML	*Unified Modeling Language* - Linguagem de Modelagem Unificada

Sumário

Capítulo 1

Introdução

Os dispositivos móveis vem ganhando cada vez mais espaço no cotidiano das pessoas por trazer funcionalidades diversas em um dispositivo cada vez mais barato e portátil. Seja para fazer uma simples operação matemática, ou para navegar na *web*, tirar fotos, telefonar, etc., os *smartphones* e *tablets* tem evoluído cada vez mais tanto em questões de *hardware* quanto em *software*.

Em se tratando de *hardware*, os dispositivos móveis hoje carregam um grande poder computacional. Segundo uma matéria do Olhar Digital [1], o *desktop* top de linha em 2001 possuía 80 GB de HD, 128 MB de memória primária e um processador single core de 1,53 GHz. Hoje, um *smartphone* top de linha possui 8 núcleos de processamento, com frequências de até 2,8 GHz, memória primária de 4 GB e armazenamento interno de 256 GB (com possibilidade de expansão) [2], tudo isso em um *design* compacto que cabe no bolso. Graças à essa evolução, é possível realizar tarefas cada vez mais complexas em um dispositivo móvel.

Quanto a *software*, hoje existe uma maior padronização dos sistemas mobile, o que facilita o desenvolvimento de aplicações. O sistema operacional líder de mercado é o Android [3]. Tendo sua primeira versão lançada em 2008, diversos foram os motivos pela sua grande popularidade, tais como o fato de ser gratuito, *open-source* (sob a licença Apache 2.0, principalmente [4]), desenvolvimento em conjunto com empresas interessadas (*Open Handset Alliance* - OHA) e o uso do Java para o desenvolvimento de aplicativos, uma vez que essa é a linguagem de programação mais utilizada mundialmente [5], além de entregar ao usuário um sistema moderno, elegante e cheio de recursos.

Devido a constante presença dos dispositivos móveis (com sistema Android) e sua crescente capacidade de *hardware*, esta plataforma tem sido cada vez mais explorada por desenvolvedores. Uma prova disso é a loja oficial de aplicativos do Android (*Google Play*), que oferece uma infinidade de aplicativos (de calculadoras a jogos com gráficos realistas), muitos deles disponibilizados gratuitamente. É por este motivo também que esta plataforma foi escolhida para a realização deste trabalho,

que visa desenvolver um simulador das funcionalidades de um Arduino UNO.

A utilização do poder computacional de *smartphones* / *tablets* em substituição à sistemas tradicionais não é uma ideia nova. Pode-se citar trabalhos como o de Junior [6], que utilizou um dispositivo Android para a implementação de um osciloscópio de baixo custo. Utilizando um microcontrolador ARM Cortex M4F para aquisição dos sinais e comunicação *Bluetooth* com o *smartphone*, foi possível construir um osciloscópio com um custo de projeto de US$35, com erro médio de 0,2% no eixo do tempo, 0,02V no eixo da tensão e taxa máxima de aquisição de 150 mil amostras por segundos, o que, segundo o autor, torna este um sistema "aceitável para o uso do projeto no ambiente de aprendizado", considerando a diferença de preço com osciloscópios comerciais.

Em uma abordagem semelhante, Nwokorie [7] utiliza um *smartphone*, junto à uma placa de Arduino UNO, para a criação de um microscópio. O chamado *SmartScope* utiliza uma estrutura de suporte com uma lente plano-convexa para adaptar a câmera do *smartphone* a captura de imagens microscópicas. A placa de Arduino controla o LED que serve como fonte de luz e permite o ajuste de intensidade do brilho por meio de botões, mostrando em um display LCD a configuração atual. O sistema tem funções de captura de imagem ou vídeo e permite o armazenamento dos dados coletados em um banco de dados (Microsoft Access). Assim como o trabalho de Junior, o grande objetivo é criar um sistema de baixo custo como alternativa aos microscópios comerciais e ajudar alunos sem experiência a aprender a realizar leituras no equipamento.

Também é possível citar o trabalho de Lin [8] que vai além do nível de aplicativo, fazendo modificações no kernel Linux para leitura de dados em um barramento CAN. Lin e sua equipe desenvolveram drivers e bibliotecas para realizar a leitura de dados de sensores em um automóvel por meio do barramento CAN. Dados relativos à velocidade, faróis, temperatura, chaves e alarme foram lidos diretamente das unidades de controle do veículo e mostrados na tela do *smartphone* em um aplicativo de instrumentação próprio, simulando o painel do carro. Seu trabalho evidencia que as possibilidades de uso dos sistemas Android podem se estender também para o nível de sistema.

Procurando na *Google Play*, é possível encontrar diversos aplicativos relacionados com Arduino. Muitos deles se encontram na forma de "aplicativo-tutorial", mostrando exemplos de código e diagramas para o ensino da plataforma, mas também é possível encontrar ambientes de desenvolvimento integrado (IDE) com capacidade de gravação das placas físicas, geradores de código automático, módulos para serem utilizados em projetos de automação (como controles *Wireless* que se integram às *Shields* do Arduino), etc.

Na parte de simuladores, vários foram os aplicativos de simulação/emulação de processadores e microcontroladores encontrados. Um destaque fica para o aplicativo *MCU Prototype Board Simulator*, que simula um kit de desenvolvimento (com botões, leds, LCD, etc.) e permite a execução de códigos

assembly do microcontrolador 68705.

Quanto à simulação de placas de Arduino, apenas um aplicativo foi encontrado. O *CircSim Circuit Simulator* apresenta a interface de um simulador convencionar de circuitos eletrônicos para PC, entanto seu uso é praticamente impossibilitado dado que sua interface não se ajusta adequadamente em dispositivos móveis (fato que pode ser comprovado pelos comentários dos usuários na página do aplicativo).

Sendo assim, o presente trabalho surge também para preencher uma lacuna existente e oferecer mais uma possibilidade para explorar o Arduino, seja para estudantes, hobistas ou qualquer pessoa que se interesse pelo assunto.

1.1 Motivação

O Arduino é uma plataforma de *hardware* e *software* aberto, destinada à desenvolvimentos de projetos na área de eletrônica. Suas aplicações vem crescendo a cada dia e vão desde o uso educacional para ensino de robótica em escolas de ensino fundamental, até aplicações em IoT (Internet of Things - Internet das Coisas) nas universidades.

Uma das principais características do Arduino é sua simplicidade. Em suas versões mais básicas, utilizam microcontroladores de 8-bits, o que torna o sistema menos complexo e mais barato quando comparada com outras plataformas concorrentes (tais como Raspberry Pi e BeagleBone) [9]. Com isso, se tornou uma plataforma ideal para prototipagem e para aprendizado, englobando um publico alvo de artistas, sem conhecimento prévio de eletrônica e programação, à engenheiros experientes, que usam a plataforma para prototipagem.

O desenvolvimento para essa plataforma demanda o uso de placas físicas de Arduino. Essas placas foram desenvolvidas com o objetivo de apresentarem baixo custo e facilitar a prototipagem, se integrando facilmente à módulos externos (*shields*) que adicionam ao sistema funções de interface, sensoriamento, etc. A desvantagem de se utilizar placas eletrônicas é, além do custo para adquiri-las, a necessidade de outros componentes externos, como leds, *protoboards*, multímetros, etc.

Uma opção às placas de Arduino são os simuladores. Para um usuário iniciante, o simulador representa a possibilidade de iniciar seus estudos sem a necessidade de gastos com equipamentos eletrônicos e sem o risco de perder estes equipamentos por uso indevido. Para usuários experientes, um simulador permite explorar diferentes ambientes/situações de funcionamento, além de outros benefícios como o monitoramento interno do sistema, vasta gama de equipamentos eletrônicos virtuais, equipamentos de medição (voltímetro, osciloscópio, etc.), entre outros.

Diversas são as opções de simuladores existentes, cada qual com características próprias. Alguns

que merecem destaque são:

- Proteus: Um dos mais completos encontrados. Possui recursos para montagem e simulação de circuitos eletrônicos analógicos e digitais, bem como desenvolvimento de layouts PCB. Tem como desvantagem o fato de ser um *software* comercial, com a versão básica para simulação Arduino custando US$248,00 [10]

- Virtual Breadboard: Permite a simulação de circuitos eletrônicos digitais em um ambiente virtual, bem como a programação de microcontroladores dentro do próprio sistema. Possui uma versão gratuita, no entanto, para simulação de Arduino é preciso comprar um módulo separadamente pelo valor de US$49,00 [11].

- Simuino: Simulador de Arduino UNO e MEGA para terminal. Apesar de gratuito e *Open-Source*, possui apenas versões para Linux e é distribuído apenas em formato de código fonte, sendo necessário fazer a compilação antes de usá-lo [12].

- CodeBlocks: Possui uma versão com ferramentas próprias para escrever e simular códigos Arduino, permitindo também o *upload* de código para placas físicas, com suporte à diversos modelos de *hardware*. A desvantagem fica por conta de não possuir uma interface gráfica para simulação, que ocorre toda em terminal apenas com textos indicando os estados de entrada e saída [13].

- Autodesk Tinkercad: Certamente, a melhor ferramenta encontrada. É um ambiente de aprendizado online gratuito que permite tanto a criação de projetos eletrônicos quanto de desenhos 3D. Possui um ambiente para codificação e depuração do código na própria ferramenta. A única desvantagem encontrada é o fato de funcionar *on-line*, sendo necessário fazer um cadastro para utilizá-la [14].

Apesar da grande quantidade de simuladores já existentes, como foi mostrado anteriormente, há uma grande falta deste tipo de *software* para dispositivos móveis. A grande motivação deste trabalho é poder contribuir com a comunidade que deseje aprender sobre desenvolvimento para Arduino, fornecendo uma opção de simulador para dispositivos móveis.

1.2 Objetivo

O objetivo principal do projeto é permitir a execução de programas desenvolvidos para a plataforma Arduino UNO em um aplicativo Android, permitindo assim a realização de testes sem a necessidade de uma placa de Arduino ou qualquer outro componente externo.

Também faz parte dos objetivos do trabalho a integração do simulador com a IDE oficial do projeto Arduino.

1.3 Justificativa

Este trabalho se justifica por atuar de forma a criar uma ferramenta *Open-Source* que beneficiará todos aqueles que desejam aprender/praticar o desenvolvimento para Arduino.

1.4 Organização do Trabalho

Este trabalho está distribuído em XXX capítulos, incluindo esta introdução, dispostos conforme a descrição que segue:

Capítulo 2: Descreve .....

Capítulo 3: Discorre sobre .....

Capítulo 4: Apresenta .....

Capítulo 2

Embasamento Teórico

Neste capítulo serão descritos o funcionamento e as características de cada módulo presente no microcontrolador usado no Arduino UNO (ATmega328P) e que será implementado no simulador.

2.1 Visão Geral

O ATmega328P é um microcontrolador RISC de 8-bits e arquitetura Harvard (memória de dados separada da memória de programa). Possui 28 pinos (encapsulamento PDIP), sendo 23 programáveis e pode tralhar com frequência máxima de operação de 20MHz.

Entre os periféricos que estão integrados neste dispositivo, pode-se listar:

- Dois temporizadores de 8-bits com *prescaler* separados;
- Um temporizador de 16-bits;
- 6 canais de PWM;
- Conversor Analógico-Digital de 10-bits (8 canais multiplexados);
- Duas interfaces de comunicação serial SPI;
- Uma USART (*Universal Synchronous Asynchronous Receiver Transceiver*) serial;
- Uma interface serial TWI (*2-wire Serial Interface*), compatível com I^2C da Philips;
- *Watchdog Timer* programável com oscilador separado;
- Entre outros.

A figura 2.1 apresenta um diagrama de blocos da organização interna do microcontrolador

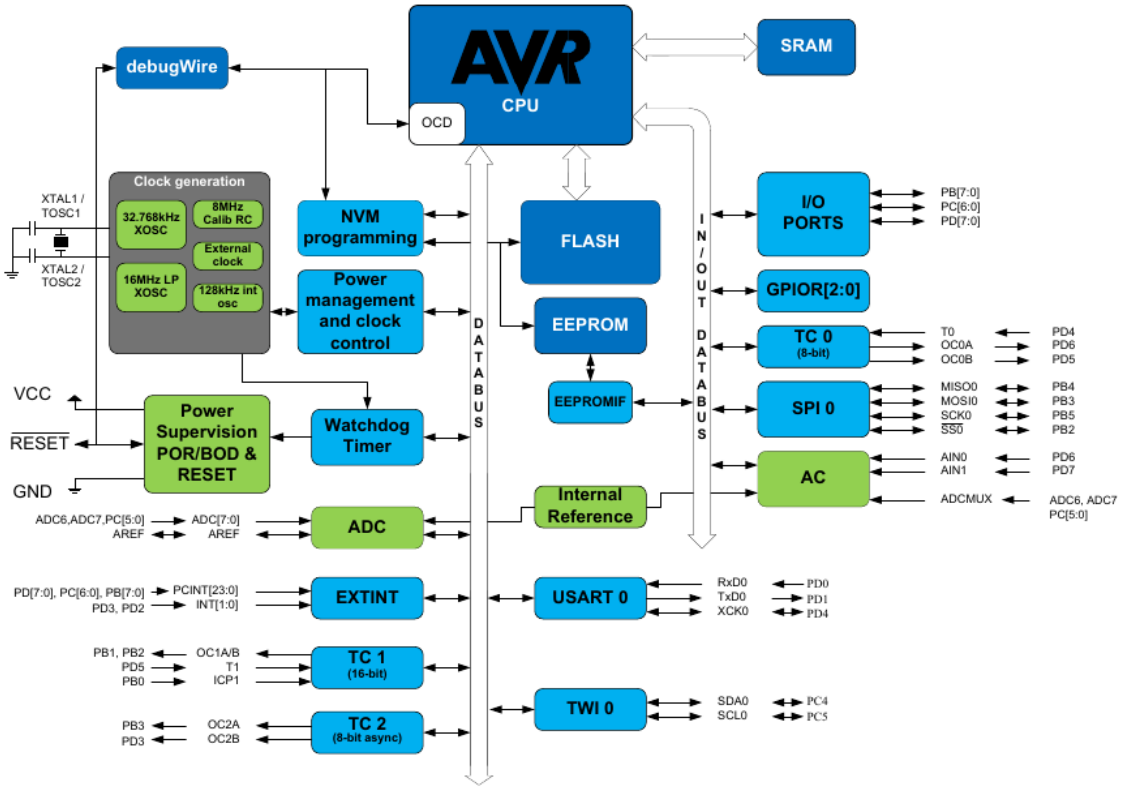

Figura 2.1: Diagrama de blocos da organização interna do ATmega328P

Fonte: Folha de dados ATmega328P

2.2 CPU

A CPU do ATmega328P é apresentada na figura 2.2. Ela possui um banco de 32 registradores de 8-bits, com os 6 últimos podendo ser utilizados como registradores de 16-bits (chamados de registrador X (R27:R26), Y(R29:R28) e Z(R31:R30)); PC de 14-bits; Registrador de *status* (8-bits) que armazenam as *flags* geradas por cada operação aritmética/lógica (zero, *carry*, *overflow*, etc); *Stack Pointer* de 16-bits e demais registradores auxiliares.

A CPU utiliza um *pipeline* de um estágio o que, junto com a arquitetura Harvard, permite que o sistema atinja uma velocidade máxima de 1 MIPS/MHz.

Em chamadas de sub-rotinas e interrupções, a CPU utiliza uma pilha implementada diretamente na memória SDRAM, cujo topo é apontado pelo registrador *Stack Pointer*. Esta estrutura de dados cresce do endereço mais alto da memória para o endereço mais baixo, de forma que o *Stack Pointer* deve ser corretamente inicializado para o último endereço da memória SDRAM.

As interrupções no ATmega328P são organizadas segundo sua prioridade. A tabela 2.1 mostra o vetor de interrupção contendo o endereço de desvio para cada tipo de interrupção. Quanto mais baixo o endereço, maior é a prioridade, de forma que o *RESET* é a interrupção de maior prioridade no

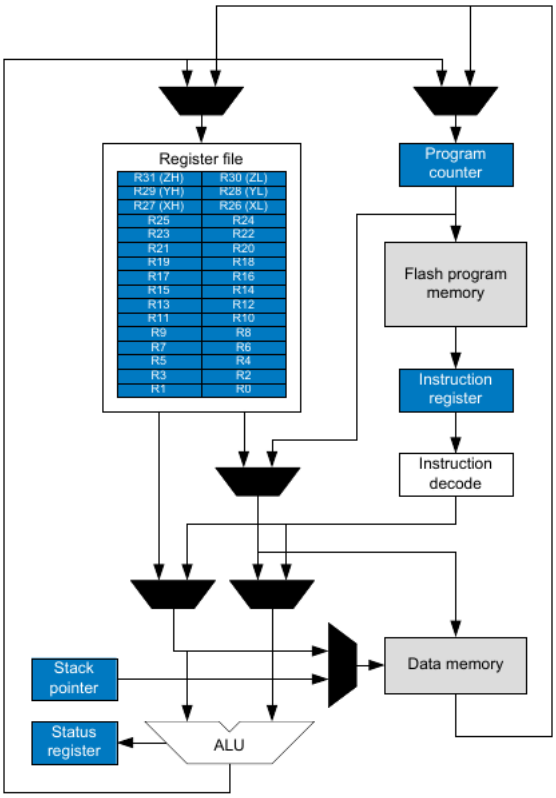

Figura 2.2: Diagrama de blocos da organização da CPU

Fonte: Folha de dados ATmega328P

sistema.

Importante resaltar que as interrupções são desabilitadas automaticamente ao iniciar o tratamento de uma interrupção (e reabilitadas ao terminar), no entanto, este comportamento pode ser alterado por *software* reabilitando as interrupções no começo da rotina.

As interrupções são classificadas em duas classes: as disparadas por envento e as disparadas por uma condição.

Quando as interrupções são disparadas por eventos, é habilitada uma *flag* indicando a ocorrência do evento. Se a interrupção estiver ativada para aquele evento, a interrupção será tratada ou enfileirada para execução posterior. Ou seja, em interrupções por evento, os eventos que não foram tratados são lembrados e serão executados em ordem de prioridade assim que possível.

Quando o disparo ocorre por uma condição, a chamada para a rotina de interrupção permanece ativa enquanto a condição estiver presente. Este tipo de interrupção não necessariamente habilita *flags* de modo que, se a condição for removida antes que a CPU possa tratar a interrupção correspondente, a interrupção não ocorrerá.

Tabela 2.1: Vetor de interrupções ATmega328P

Endereço de Desvio	**Interrupção**	**Descrição**
0x00	RESET	Interrupção de Reset
0x02	INT0	Interrupção Externa 0
0x04	INT1	Interrupção Externa 1
0x06	PCINT0	Interrupção de mudança de estado 0
0x08	PCINT1	Interrupção de mudança de estado 1
0x0A	PCINT2	Interrupção de mudança de estado 2
0x0C	WDT	Estouro do *Watchdog Timer*
0x0E	TIMER2_COMPA	Comparação *Timer* 2 canal A
0x10	TIMER2_COMPB	Comparação *Timer* 2 canal B
0x12	TIMER2_OVF	Estouro do *Timer* 2
0x14	TIMER1_CAPT	Captura de evento *Timer* 1
0x16	TIMER1_COMPA	Comparação *Timer* 1 canal A
0x18	TIMER1_COMPB	Comparação *Timer* 1 canal B
0x1A	TIMER1_OVF	Estouro do *Timer* 1
0x1C	TIMER0_COMPA	Comparação *Timer* 0 canal A
0x1E	TIMER0_COMPB	Comparação *Timer* 0 canal B
0x20	TIMER0_OVF	Estouro do *Timer* 0
0x22	SPI STC	Transferência SPI completa
0x24	USART_RX	Recepção USART completa
0x26	USART_UDRE	Registrador de dados vazio (USART)
0x28	USART_TX	Transmissão USART completa
0x2A	ADC	Conversão analógico-digital completa
0x2C	EE READY	EEPROM pronta
0x2E	ANALOG COMP	Comparador analógico
0x30	TWI	Interface serial I^2C
0x32	SPM READY	Armazenamento na memória de programa

2.3 Memória de Programa

A memória de programa é uma FLASH de 32kB x 8-bits, que está organizada da forma 16kB x 16-bits pois cada instrução do microcontrolador é de 16 ou 32-bits. Assim, o registrador PC de 14-bits pode fazer um endereçamento a palavra na memória de programa.

A figura 2.3 mostra a organização da memória de programa. Pode-se notar que o *Boot Loader* está posicionado em uma seção separada do restante da memória e isso ocorre por questões de segurança.

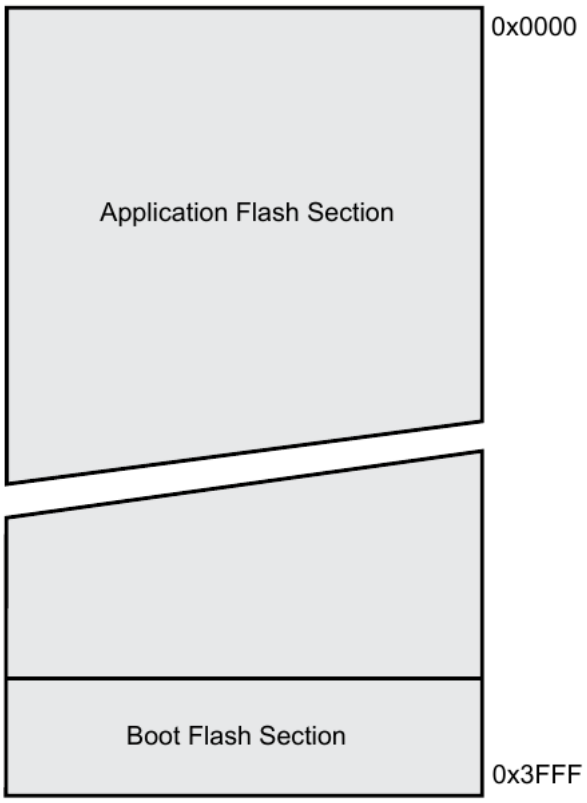

Figura 2.3: Memória de programa ATmega328P

Fonte: Folha de dados ATmega328P

2.4 Memória de Dados

O ATmega328P possui 2kB de memória de dados SDRAM, além do espaço de dados reservado aos registradores.

Apesar dos registradores não estarem fisicamente implementados na memória de dados, o microcontrolador faz um mapeamento linear da memória de modo a se obter, na prática, uma memória como mostrado na figura 2.4.

Existem diferentes modos de endereçamento que são aplicados à memória de dados. Todo o

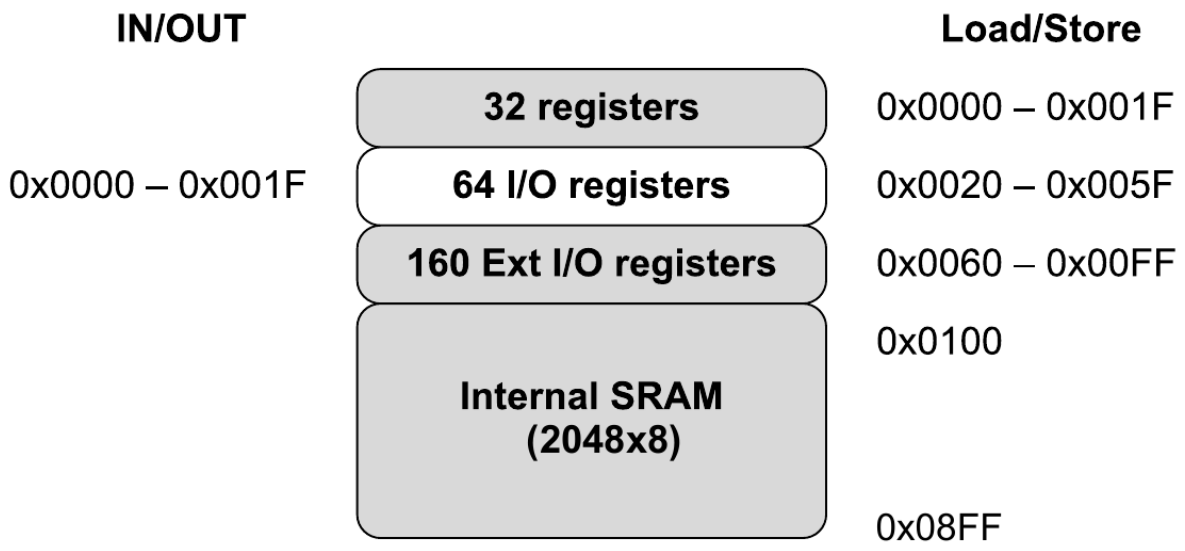

Figura 2.4: Memória de dados ATmega328P

Fonte: Folha de dados ATmega328P

espaço de endereçamento suporta qualquer um dos modos listados, são eles:

- Direto: Acesso direto ao endereço desejado;
- Indireto com deslocamento: Acesso à 63 endereços deslocados a partir do endereço base, dado pelo registrador Y ou Z.
- Indireto: Acesso ao endereço dado pelos registradores X, Y ou Z.
- Indireto com pré-decremento: Registradores X, Y ou Z são decrementados antes de serem utilizados como ponteiro para endereçamento.
- Indireto com pós-incremento: Registradores X, Y ou Z são incrementados depois de serem utilizados como ponteiro para endereçamento.

2.5 Módulo de Entrada e Saída Digital

Capítulo 3

Material e Métodos ou Desenvolvimento do Projeto

3.1 Material

Material utilizado no projeto.

3.2 Métodos

Métodos utilizados no projeto.

Capítulo 4

Resultados e Discussões

Resultados e discussões sobre o trabalho.

Capítulo 5

Conclusão ou Conclusões

Conclusões do trabalho de conclusão de curso.

Trabalhos futuros

Isso é para a Monografia Final de defesa.....

Referências

[1] Microsoft relembra produtos que eram sucesso na época do xp. https://olhardigital.com.br/noticia/microsoft-relembra-produtos-que-eram-sucesso-na-epoca-do-xp/40602, Acesso em: 11 de março de 2018.

[2] Samsung galaxy s9. https://comparador.tecmundo.com.br/samsung-galaxy-s9/, Acesso em: 18 de março de 2018.

[3] R.R. Lecheta. *Google Android 4ª edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. Novatec Editora, 2015.

[4] Content license. https://source.android.com/setup/licenses, Acesso em: 18 de março de 2018.

[5] Paul Deitel, Harvey Deitel, and Abbey Deitel. *Android for Programmers: An App-Driven Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition, 2012.

[6] José Ernesto Almas de Jesus JUNIOR. Implementação de um osciloscópio de baixo custo com exibição gráfica em aplicativo para android, 2016.

[7] A. C. Eberendu, B. O. Omaiye, and E. C. Nwokorie. Using android application to turn smart device into digital microscope on arduino and window platform. In *2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON)*, pages 508–513, Nov 2017.

[8] H. F. Teng, M. J. Wang, and C. M. Lin. An implementation of android-based mobile virtual instrument for telematics applications. In *2011 Second International Conference on Innovations in Bio-inspired Computing and Applications*, pages 306–308, Dec 2011.

[9] J. Overman, L. Pool, L. Kreuk, and T. Ketel. Arduino - the open-source ide. https://delftswa.gitbooks.io/desosa-2017/content/arduino/chapter.html, Acesso em: 08 de abril de 2018.

[10] Create your package. https://www.labcenter.com/buy-vsm/, Acesso em: 15 de abril de 2018.

[11] Vbb software licenses. http://www.virtualbreadboard.com/DocView.html?doc=WebShop/WebShop, Acesso em: 15 de abril de 2018.

[12] Arduino uno/mega simulator. http://web.simuino.com/home-1, Acesso em: 15 de abril de 2018.

[13] Codeblocks arduino ide. http://arduinodev.com/codeblocks/, Acesso em: 15 de abril de 2018.

[14] O tinkercad é um aplicativo simples e on-line de projeto e impressão 3d para todos os usuários. https://www.tinkercad.com/, Acesso em: 15 de abril de 2018.