

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
DEPARTAMENTO DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

RHUAN EDSON CALDINI COSTA

**DESENVOLVIMENTO DE UM OSCILOSCÓPIO DIGITAL
UTILIZANDO AS PLATAFORMAS ARDUINO E ANDROID**

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO

2016

RHUAN EDSON CALDINI COSTA

**DESENVOLVIMENTO DE UM OSCILOSCÓPIO DIGITAL
UTILIZANDO AS PLATAFORMAS ARDUINO E ANDROID**

Proposta de Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso 1, do curso de Engenharia de Computação, Departamento de Computação – DACOM, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof Dr. André Takeshi Endo

Co-orientador: Prof Dr. André Sanches Fonseca
Sobrinho

CORNÉLIO PROCÓPIO

2016

RESUMO

COSTA, Rhuan Edson Caldini. Desenvolvimento de um Osciloscópio Digital Utilizando as Plataformas Arduino e Android. 35 f. Trabalho de Conclusão de Curso – Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

O barateamento dos dispositivos eletrônicos aliado ao surgimento de plataformas acessíveis de prototipagem, como o Arduino, tem contribuído para o crescimento no número de pessoas adeptas ao “Faça Você Mesmo” - *Do it Yourself* (DIY). Em alguns projetos, ferramentas mais avançadas como o osciloscópio são necessárias para análise de seu funcionamento. Porém esta é uma ferramenta relativamente cara se comparada ao custo envolvido nos projetos de DIY. Desta forma, a presente proposta de trabalho tem como objetivo desenvolver um osciloscópio baseado nas plataformas Arduino e Android, bem como avaliar sua eficiência perante um osciloscópio profissional. Espera-se como resultado viabilizar aos praticantes de DIY a construção de um osciloscópio utilizando componentes de fácil acesso, como Arduinos e dispositivos móveis Android.

Palavras-chave: Android, Arduino, Osciloscópio

ABSTRACT

COSTA, Rhuan Edson Caldini. Development of a Digital Oscilloscope Using Arduino and Android Platforms. 35 f. Trabalho de Conclusão de Curso – Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

The cheapening of electronic devices allied to the emergence of affordable prototyping platforms, such as the Arduino, have been contributing to the growth in the number of people adept to Do it Yourself (DIY). In some projects, advanced tools as the oscilloscope are required for the analysis of its operation. However, it is a relatively expensive tool compared to the cost involved in DIY projects. Thus, the present work proposal has as objective to develop an oscilloscope based on the Arduino and Android platforms, as well as to evaluate its efficiency towards a professional oscilloscope. It is expected as a result to make feasible to the DIY practitioners the construction of an oscilloscope using easily accessible components, as Arduinos and Android mobile devices.

Keywords: Android, Arduino, Oscilloscope

LISTA DE FIGURAS

FIGURA 1	– Camadas da Plataforma Google Android	12
FIGURA 2	– <i>Layout</i> e Pinagem do Arduino Uno	16
FIGURA 3	– <i>Layout</i> do Arduino Mega 2560	16
FIGURA 4	– Conexão entre ATmega32u4 e Atheros AR9331 no Arduino Yún	17
FIGURA 5	– <i>Layout</i> do Arduino Yún	18
FIGURA 6	– Esquemático Exemplo	19
FIGURA 7	– IDE Arduino	20
FIGURA 8	– Amostragem e Quantização de um Sinal Analógico	21
FIGURA 9	– Divisão de Tensão Através de uma Série de Resistores	22
FIGURA 10	– Configuração Não Inversora do Amp-Op	23
FIGURA 11	– Esquemático do Circuito de Amplificação com Offset	24
FIGURA 12	– Tela de um Osciloscópio	25
FIGURA 13	– Diagrama de Blocos de um Osciloscópio Digital	27
FIGURA 14	– Diagrama da Arquitetura Proposta	29
FIGURA 15	– Diagrama da Arquitetura Proposta	30

LISTA DE SIGLAS

DIY	Do it Yourself
GUI	Graphical User Interface
ASF	Apache Software Foundation
ART	Android Runtime
AOT	Ahead-of-Time
SDK	Software Development Kit
IDE	Integrated Development Environment
ROM	Read Only Memory
RAM	Random Access Memory
PWM	Pulse Width Modulation
IoT	Internet of Things
PoE	Power over Ethernet
ADC	Analog to Digital Converter
A/D	Analógico/Digital
Amp-Op	Amplificador Operacional

SUMÁRIO

1 INTRODUÇÃO	7
1.1 MOTIVAÇÃO	8
1.2 OBJETIVOS	9
1.3 ORGANIZAÇÃO TEXTUAL	9
2 FUNDAMENTAÇÃO TEÓRICA	10
2.1 SISTEMA OPERACIONAL ANDROID	10
2.2 APLICAÇÕES PARA ANDROID	11
2.3 PLATAFORMA ARDUINO	13
2.3.1 Arduino Uno	15
2.3.2 Arduino Mega 2560	15
2.3.3 Arduino Yún	17
2.3.4 Exemplo de um Projeto com Arduino	18
2.4 CONVERSÃO ANALÓGICO-DIGITAL	19
2.5 PRÉ-CONDICIONAMENTO DO SINAL	21
2.5.1 Circuito de Atenuação	21
2.5.2 Circuito de Amplificação	22
2.5.3 Circuito de Offset	23
2.6 OSCILOSCÓPIO DIGITAL	25
2.7 TRABALHOS RELACIONADOS	27
3 PROPOSTA DE TRABALHO	29
3.1 MONTAGEM DO CIRCUITO DE CONDICIONAMENTO DO SINAL	29
3.2 MONTAGEM DO CIRCUITO DE AQUISIÇÃO DO SINAL	29
3.3 COMUNICAÇÃO ENTRE ARDUINO E DISPOSITIVO MÓVEL	30
3.4 DESENVOLVIMENTO DA APLICAÇÃO ANDROID	30
3.5 EXECUÇÃO DE TESTES E ANÁLISE DOS RESULTADOS	31
4 CRONOGRAMA	32
REFERÊNCIAS	33

1 INTRODUÇÃO

Nas últimas décadas houve uma grande evolução no desenvolvimento de componentes e produtos eletrônicos. Um marco histórico foi a implementação do transistor na década de 40, que substituiu as caras, grandes e pouco duráveis válvulas, possibilitando a construção de equipamentos menores e mais duráveis (LATHI; DING, 2012; BOYLESTAD; NASHELSKY, 2004; MORIMOTO, 2011). Os processos de produção de dispositivos semicondutores estão cada vez mais otimizados, resultando em componentes mais eficientes, menores e mais baratos. Isto possibilitou a construção de equipamentos operados por meio de baterias, tornando-os compactos e portáteis, como por exemplo os dispositivos móveis.

Os *smartphones* se tornaram extremamente populares, atualmente existindo mais de 2 bilhões de aparelhos no mundo, com previsão de ultrapassar a marca dos 2.6 bilhões até 2019 (EMARKETER, 2016). Segundo uma pesquisa da *International Data Corporation IDC* (2015), mais de 80% destes *smartphones* utilizavam o sistema operacional Android em 2015, desenvolvido pelo Google e *open-source*. Estes dispositivos se tornaram parte do dia a dia das pessoas, estando presentes não só como meios de comunicação e entretenimento, mas também como ferramentas na área de saúde, educação e até mesmo militar (FARTO, 2016).

Além do barateamento e miniaturização dos dispositivos eletrônicos, o surgimento de plataformas *open-source* de hardware e software como o Arduino, tem promovido o crescente número de pessoas adeptas ao *do it yourself* (DIY) (KUZNETSOV; PAULOS, 2010; NICULESCU; LITA, 2015). Esta prática consiste em construir equipamentos, eletrônicos ou não, por conta própria, utilizando-se de componentes disponíveis no mercado (*off the shelf*).

Muitos dos projetos desenvolvidos pelos hobbystas utilizam-se de microcontroladores para a implementação da lógica de funcionamento. Alguns deles atingem complexidades maiores, envolvendo sinais de alta frequência, consequentemente exigindo ferramentas de diagnóstico mais aprimoradas para a análise de seu funcionamento. Tais ferramentas, como o osciloscópio, costumam ser relativamente caras se comparadas ao custo dos componentes envolvidos nos projetos.

O osciloscópio é um instrumento utilizado na visualização de sinais elétricos, analógicos ou digitais, caracterizado pela sua alta precisão (KARIM, 2014). Sendo muito utilizado por técnicos, engenheiros, mecânicos e até mesmo na área médica, este aparelho é capaz de interpretar sinais com frequência de até centenas megahertz e mostrá-los em um *display* de modo que a forma do sinal seja humanamente comprehensível (BOYLESTAD; NASHELSKY, 2004).

1.1 MOTIVAÇÃO

Dada a portabilidade dos dispositivos móveis e a elevada capacidade de processamento quando comparados aos microcontroladores, estão surgindo projetos que utilizam ambos os dispositivos trabalhando em conjunto, aproveitando as particularidades de cada um. Um exemplo é a utilização de dispositivos móveis como interface humano-computador devido a sua capacidade de processamento de imagens em duas e três dimensões, além da entrada de comandos via *touch-screen*. Enquanto isso, o microcontrolador é responsável pela aquisição dos sinais de sensores, processamento destes sinais e envio de comandos aos atuadores, além da comunicação com o dispositivo móvel para a interação humana.

As tecnologias mais populares nestes tipos de sistemas são a plataforma Arduino e o sistema operacional móvel Android, ambos por serem dispositivos bem difundidos em meio a comunidade *maker*, como são chamados os praticantes de DIY. Outro ponto importante é a ideologia *open-source* destes dispositivos, que possibilita aos desenvolvedores mais avançados a customização dos mesmos, além da grande quantidade de material técnico disponível na Internet.

No desenvolvimento de projetos que utilizam transmissão de sinais, uma tarefa indispensável é a análise dos sinais envolvidos, sejam eles analógicos ou digitais. Para tal tarefa, é necessário o uso de um osciloscópio, que por ser um equipamento de alto custo, está disponível somente para profissionais da área ou em ambientes específicos, como universidades e empresas do ramo. Alunos de cursos relacionados à eletrônica e sistemas embarcados têm acesso ao osciloscópio das universidades, porém este acesso fica, na maioria das vezes, restrito somente ao momento das aulas práticas, o que pode dificultar o desenvolvimento de projetos pessoais.

Uma forma utilizada para contornar este problema é a construção de osciloscópios simples, de baixo custo, sem as funcionalidades específicas dos instrumentos profissionais como por exemplo trigger e cursor. Nestes projetos são utilizados conversores analógico-digital de microcontroladores, e até mesmo de placas de captura de áudio. Estes dispositivos não são desenvolvidos para tal finalidade, porém conseguem obter uma representação satisfatória do sinal analisado desde que este seja de baixa frequência, geralmente abaixo de 10 kHz (ATMEL, 2015). Nestes projetos, um computador é utilizado para o processamento dos dados e como *Graphical User Interface* (GUI), enquanto o microcontrolador fica responsável apenas pela conversão analógico-digital.

Existem também projetos que utilizam a entrada de áudio (microfone externo) de um dispositivo Android para a captura dos dados analógicos, mostrando o gráfico correspondente

na tela do dispositivo. Esta solução varia em precisão dependendo da capacidade do conversor analógico-digital contido no dispositivo móvel, estando limitada a sinais geralmente abaixo de 20 kHz, uma vez que estes dispositivos são projetados para trabalhar na faixa de frequência onde se encontram os sons audíveis ao ser humano, 20 Hz a 20 kHz (YI, 2010).

Diante dos pontos acima mencionados, o desenvolvimento de um osciloscópio de baixo custo construído a partir de componentes facilmente encontráveis no mercado, com satisfação precisão para utilização em hobbies atenderia às necessidades de pessoas que desejam se aventurar no mundo da eletrônica, porém não pretendem investir em um equipamento caro e considerado de uso profissional.

1.2 OBJETIVOS

Devido à popularidade da plataforma Arduino entre os *makers* (praticantes de DIY), e também à grande quantidade de dispositivos móveis com sistema operacional Android, a proposta do presente trabalho é construir e analisar a eficiência de um osciloscópio baseado nessas duas plataformas.

Para tal análise será considerada a integração entre a plataforma Arduino e o dispositivo móvel com Android, bem como suas funcionalidades em relação a um osciloscópio profissional.

1.3 ORGANIZAÇÃO TEXTUAL

O desenvolvimento deste trabalho está dividido em cinco capítulos. A fundamentação teórica é apresentada no Capítulo 2, subdividido nos principais tópicos abordados nesta proposta: sistema operacional Android, aplicações para Android, plataforma Arduino, conversão analógico-digital, pré-condicionamento do sinal, osciloscópio digital e trabalhos relacionados. Já no Capítulo 3 é exposta a proposta deste trabalho. Por fim, no Capítulo 4 é apresentado o cronograma do mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentado o embasamento teórico relacionado a este trabalho, com o objetivo de auxiliar o leitor na compreensão do mesmo. Também serão mencionados trabalhos relacionados a esta proposta. Serão apresentadas as seguintes seções: Na Seção 2.1 são mostrados os principais aspectos do sistema operacional Android; a Seção 2.2 descreve as características das aplicações para Android; na Seção 2.3 a plataforma Arduino é explicada, detalhando alguns modelos de placa e mostrando um exemplo de projeto com Arduino; a Seção 2.4 demonstra o funcionamento da conversão de um sinal analógico para digital; o pré-condicionamento necessário para a conversão de um sinal é explicado na Seção 2.5; o funcionamento de um osciloscópio é demonstrado na Seção 2.6; e por fim são descritos alguns trabalhos relacionados na Seção 2.7.

2.1 SISTEMA OPERACIONAL ANDROID

Lançado em 2008, o sistema operacional para *smartphones* Android teve rápida aceitação no mercado. Nos Estados Unidos as vendas de telefones com Android aumentaram em mais de sete vezes em 2010 se comparado com o ano anterior (DEITEL et al., 2013). Desde então este sistema operacional vem ganhando mercado, atingindo 82,8% de participação em 2015 (IDC, 2015). Segundo Ableson et al. (2012), Android é uma plataforma de software que vem revolucionando o mercado global de telefones celulares.

O Android teve seu desenvolvimento iniciado pela Android Inc., comprada pelo Google em 2005. Em 2007 foi formado um consórcio com o objetivo de continuar o desenvolvimento do Android de forma que trouxesse um mercado de dispositivos móveis melhor e mais aberto (DEITEL et al., 2013; ABLESON et al., 2012). Composto inicialmente de 34 empresas, atualmente o consórcio conta com 84 companhias dos ramos de semicondutores, software, dispositivos móveis, operadoras de telefonia e comercialização (DEITEL et al., 2013; OPEN HANDSET ALLIANCE, 2016).

Considerado a primeira plataforma móvel *open-source*, o Android possibilita aos fabricantes de dispositivos móveis modificar, adicionar novas funcionalidades e otimizar o sistema operacional para seus aparelhos sem depender do Google (ABLESON et al., 2012; LECHETA, 2015). Sob a licença *Apache Software Foundation* (ASF), é permitido que alterações sejam feitas no código fonte para customização do sistema operacional sem que haja necessidade de compartilhamento destas alterações (LECHETA, 2015).

O Android conta com um *kernel* baseado em Linux, uma interface gráfica rica, aplicações para usuário final, bibliotecas de código, *frameworks* para aplicações, suporte multimídia e muito mais (ABLESON et al., 2012). O sistema de segurança do Android é baseado na segurança do Linux. No Android, cada aplicação é executada em um único processo, e cada processo possui uma *thread* dedicada. Para cada aplicação instalada é criado um usuário no sistema operacional com acesso a sua estrutura de diretórios. Dessa forma, nenhum outro usuário pode ter acesso a essa aplicação (LECHETA, 2015).

As aplicações desenvolvidas para Android rodam em uma máquina virtual chamada Dalvik, otimizada para a execução em dispositivos móveis. A partir do Android 4.4 (KitKat) foi criada a máquina virtual ART (Android Runtime) com o objetivo de substituir a Dalvik. Na versão 5.0 (Lollipop), o ART se tornou a máquina virtual padrão do Android. Uma das melhorias do ART é a compilação *Ahead-of-Time* (AOT), que tem como objetivo otimizar o código para melhorar o desempenho da aplicação. Além disso, houve melhorias no *Garbage Collector*, gerenciador de memória do Android (LECHETA, 2015).

O sistema operacional Android é organizado em cinco camadas projetadas para serem flexíveis, representadas na Figura 1 (ABLESON et al., 2012; VENKATASHAIAH, 2014; CAVALCANTE, 2015):

1. ***Applications***: Aplicações do usuário;
2. ***Application Framework***: Recursos para as aplicações como *activities* e *views*, janelas, serviços baseados em localização e telefonia;
3. ***Libraries***: Bibliotecas com recursos para navegação web (Webkit), banco de dados (SQLite), gráficos avançados (OpenGL), criptografia (SSL), áudio e vídeo;
4. ***Android Runtime***: Máquina virtual Dalvik e bibliotecas centrais do Java;
5. ***Linux Kernel***: Fornece uma camada de abstração do hardware, bem como *core services* (gerenciamento de processos, memória e arquivos). É no *kernel* onde são implementados os drivers de hardware como Bluetooth, Wi-Fi, *touch-screen*, câmera, GPS, acelerômetro, entre outros.

2.2 APLICAÇÕES PARA ANDROID

Enquanto os componentes internos do sistema operacional são escritos em C ou C++, as aplicações de usuário são construídas em Java (ABLESON et al., 2012). A linguagem Java

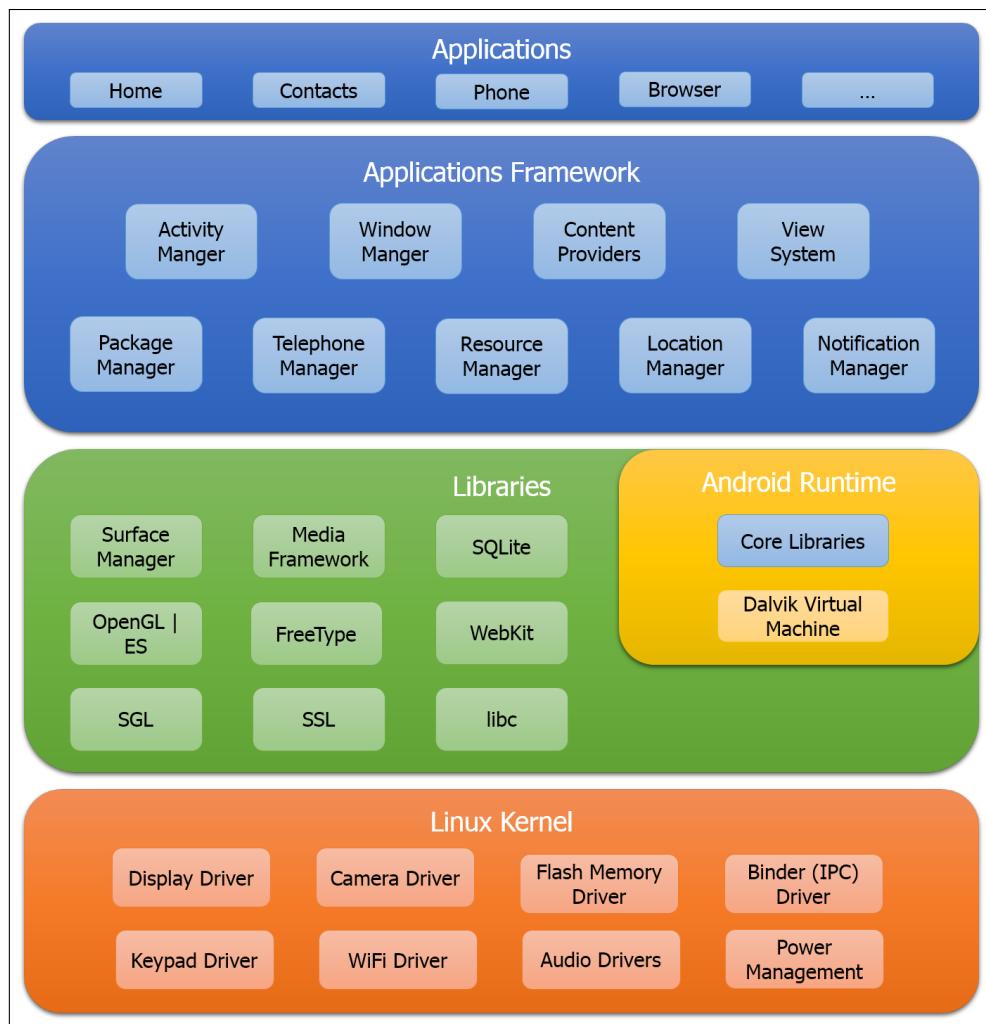


Figura 1: Camadas da Plataforma Google Android

Fonte: (VENKATASHAIAH, 2014)

foi uma escolha lógica para o desenvolvimento de aplicações para a plataforma Android, pois é poderosa, gratuita e de código-fonte aberto. Além disso, as aplicações desenvolvidas nesta linguagem podem ser executadas em uma variedade de dispositivos, sem necessidade de código específico para cada plataforma (DEITEL et al., 2013).

Uma característica da plataforma Android é que não há diferença entre aplicações inclusas no sistema operacional e aplicações criadas com o *Android Software Development Kit* (SDK). Isto significa que aplicações poderosas podem ser escritas utilizando os recursos disponíveis no dispositivo (ABLESON et al., 2012). Os principais recursos incluem sensores (acelerômetro, GPS, proximidade, luminosidade, microfone, câmera, giroscópio, entre outros), redes (Wi-Fi, Bluetooth, NFC, GSM, 3G, etc), bacos de dados e integração entre aplicações.

Para o desenvolvimento de aplicações para o Android é necessária a instalação do

Android SDK, software contendo um emulador para simular dispositivos Android, ferramentas utilitárias e uma API completa para a linguagem Java. O Android Studio é a IDE (Integrated Development Environment) oficial de desenvolvimento, fornecida pelo Google, e já contém o SDK (LECHETA, 2015).

2.3 PLATAFORMA ARDUINO

Segundo o site oficial, Arduino é uma plataforma de prototipagem *open-source* baseada em hardware e software fáceis de utilizar. Ao longo dos anos, o Arduino tem sido o cérebro de milhares de projetos, desde objetos do dia-a-dia até instrumentos científicos complexos. Uma comunidade mundial de *makers* (estudantes, hobbystas, artistas, programadores e profissionais) tem se reunido ao redor desta plataforma *open-source* e suas contribuições têm somado uma incrível quantidade de conhecimento acessível que pode ser de grande ajuda tanto para novatos quanto para especialistas. A plataforma simplifica o processo de trabalhar com microcontroladores, oferecendo algumas vantagens para professores, estudantes e amadores, entre elas (ARDUINO, 2016a):

- **Baixo custo:** placas Arduino são relativamente baratas se comparadas com outras plataformas de microcontroladores;
- **Multi-plataforma:** a IDE Arduino roda em Windows, Mac OS e Linux enquanto a maioria dos sistemas de microcontroladores são limitados ao Windows;
- **Ambiente de programação simples e limpo:** fácil de usar para iniciantes enquanto flexível o suficiente para usuários avançados;
- **Software *open-source* e extensível:** O software Arduino está disponível para aprimoramentos por programadores experientes. A linguagem pode ser expandida por meio de bibliotecas escritas em C++ e usuários mais experientes podem utilizar a linguagem AVR-C, específica para microcontroladores Atmel AVR, diretamente nos programas Arduino;
- **Hardware *open-source* e extensível:** Os esquemáticos das placas Arduino são publicados sob a licença *Creative Commons*, possibilitando a confecção de versões modificadas das mesmas.

Para se aprofundar nas especificações de hardware do Arduino, primeiramente é necessário compreender o que é um microcontrolador. Este é um circuito integrado contendo todas as partes principais de um computador típico, sendo elas (BAYLE, 2013):

- O **processador** é o cérebro do microcontrolador, onde cálculos são realizados e todas as decisões são tomadas;
- As **memórias** são espaços onde o programa e os elementos de usuário (variáveis) estão. São encontradas nos tipos *Read Only Memory* (ROM), mais lentas, porém não voláteis, usadas para armazenar o programa; e *Random Access Memory* (RAM), rápidas porém voláteis, usadas para armazenar os dados em tempo de execução;
- Os **periféricos** possuem a função de auxiliar o processador e estender suas capacidades. Exemplos são conversores analógico-digital, *timers*, interrupções, barramentos de comunicação (serial, SPI, I2C), geradores de PWM (Pulse Width Modulation), entre outros;
- As **entradas e saídas** são as vias de comunicação entre o mundo e o microcontrolador. As entradas são usadas para ler sensores e receber mensagens, enquanto as saídas comandam atuadores e enviam mensagens.

O alto nível de integração fornecido pelos microcontroladores - tudo em um único chip - os fazem ideais para sistemas embarcados, sendo usados desde em controles remotos, passando por *smartphones*, até em sistemas de injeção eletrônica de automóveis (BAYLE, 2013). Os principais aspectos a serem analisados em um microcontrolador são: clock, número de bits do processador, quantidade de memórias e periféricos disponíveis.

Atualmente existem dezenas de modelos de placas Arduino, a maioria delas baseada em microcontroladores Atmel AVR de 8 bits. A primeira placa desenvolvida possuía um ATmega8 rodando a um clock de 16 MHz com 8 KB de memória *flash*. Posteriormente foi adotado o uso do ATmega168, com 16 KB de memória *flash*. As versões mais recentes do Arduino contam com o microcontrolador ATmega328, com 32 KB de memória *flash*. Para projetos que requerem mais memória existe ainda o modelo Arduino Mega 2560, com 256 KB (EVANS et al., 2013).

Placas especialistas denominadas *shields* têm como função expandir as funcionalidades básicas dos Arduinos. Elas podem ser empilhadas, umas em cima das outras, somando múltiplas funcionalidades ao microcontrolador (EVANS et al., 2013). Exemplos de funcionalidades adicionáveis por meio de *shields* são Bluetooth, Wi-Fi, Ethernet, controle de motores elétricos, comunicação GSM, alimentação por bateria, entre outros.

Grande parte das placas Arduino segue um padrão de pinagem para que, independente do modelo, os *shields* sejam compatíveis, além de tornar a elaboração de esquemáticos universal. Este padrão conta com 14 pinos digitais de I/O dos quais seis são capazes de fornecer

saída PWM, seis entradas analógicas, além de protocolos de comunicação como serial, *Serial Peripheral Interface* (SPI) e I2C. Cada placa também possui um conector (*header*) *In-Circuit Serial Programming* (ICSP) e um botão reset (EVANS et al., 2013). Existem modelos de placas que não seguem o padrão de pinagem, desenvolvidos em circuitos menores com o intuito de serem aplicados em projetos do tipo *Internet of Things* (IoT) e *wearables*. No momento do desenvolvimento do presente trabalho, os modelos mais conhecidos devido as suas particularidades são Uno, Mega 2560 e Yún, e serão descritos a seguir.

2.3.1 ARDUINO UNO

Muito utilizada por iniciantes, a versão mais popular do Arduino atualmente é a Uno, lançada em 2010. A principal diferença entre ele e seus antecessores é a presença de, além do microcontrolador principal ATmega328, um microcontrolador ATmega16U2 programado como conversor USB-serial. Esta conversão é necessária para que o Arduino se comunique com o computador através da porta USB, sendo assim programado. O ATmega16U2 pode ser reprogramado, fazendo com que o Arduino se comporte como outro dispositivo USB como por exemplo mouse, teclado ou *joystick* (EVANS et al., 2013; ARDUINO, 2016c). Na Figura 2 são mostrados o *layout* e pinagem do Arduino Uno.

O Arduino Uno pode ser alimentado pela porta USB, a mesma utilizada para programá-lo, ou por uma fonte externa. A fonte de energia é selecionada automaticamente pela placa, que possui um regulador de tensão possibilitando-a ser alimentada com tensões entre 6 e 20 volts, porém recomenda-se tensões entre 7 e 12 volts para evitar instabilidade do microcontrolador ou superaquecimento do regulador (ARDUINO, 2016c).

2.3.2 ARDUINO MEGA 2560

O Arduino Mega 2560 é geralmente utilizado em projetos que demandam uma maior quantidade de pinos de I/O, bem como maior espaço de memória para sua programação, ou ainda múltiplas portas seriais. Além da pinagem padrão compatível com o Arduino Uno, consequentemente com os *shields*, o Mega conta com mais três conjuntos de pinos totalizando 54 pinos de I/O dos quais 15 suportam PWM, 16 entradas analógicas e quatro portas serial UART (ARDUINO, 2016b).

O microcontrolador utilizado na construção desta placa é o ATmega2560, com 256 KB de memória flash, rodando aos mesmos 16 MHz da versão Uno. Este modelo também conta com um ATmega16U2 trabalhando como conversor USB-serial (EVANS et al., 2013). O *layout*

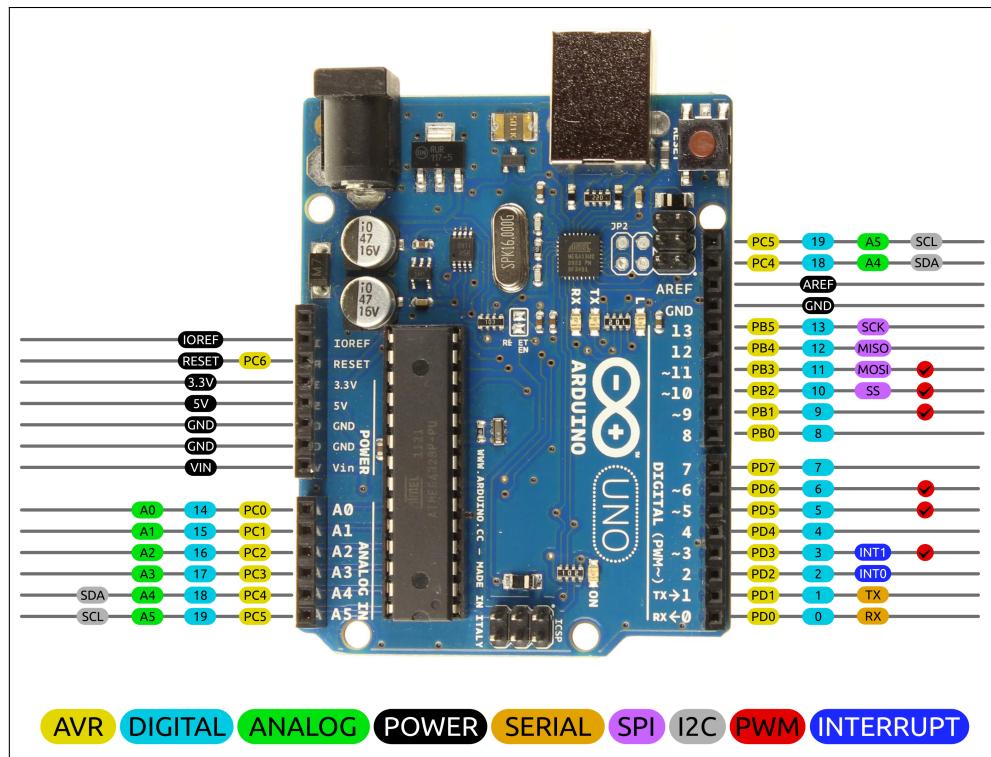


Figura 2: *Layout* e Pinagem do Arduino Uno

Fonte: Adaptado de (BOUNI, 2015)

do Arduino Mega 2560 é mostrado na Figura 3.

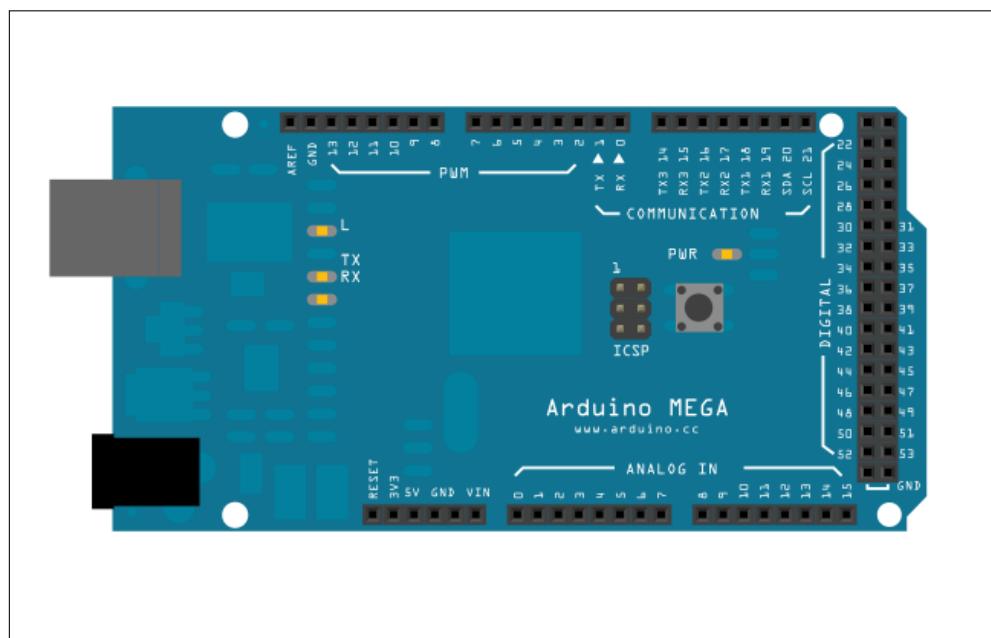


Figura 3: Layout do Arduino Mega 2560

Fonte: Extraído de (ARDUINO, 2016b)

Da mesma forma que o Arduino Uno, é possível alimentar o Mega 2560 pela porta de programação USB com tensão de 5 volts, ou por uma fonte externa entre 6 e 20 volts (recomendado de 7 a 12 volts) por meio do conector *jack* ou pelos pinos Vin e Gnd (ARDUINO, 2016b).

2.3.3 ARDUINO YÚN

Diferente das versões citadas anteriormente, o Arduino Yún possui além de um microcontrolador, um microprocessador com suporte a uma distribuição Linux baseada no OpenWrt chamado OpenWrt-Yun. Este processador fornece ao Yún as funcionalidades Wi-Fi, Ethernet, USB Host e um slot micro-SD (ARDUINO, 2016d). Portanto este modelo é geralmente utilizado em projetos que requerem conexão com a Internet.

O microcontrolador utilizado nesta versão é o ATmega32u4, que possui especificações similares ao ATmega328 utilizado no Uno. Seu maior diferencial encontra-se na comunicação USB integrada, dispensando a necessidade de outro chip dedicado a esta função. O microprocessador Linux modelo Atheros AR9331 é capaz de executar scripts escritos em Python e se comunica com o microcontrolador através de uma conexão serial. Um diagrama representando a conexão entre os dois ambientes pode ser visto na Figura 4.

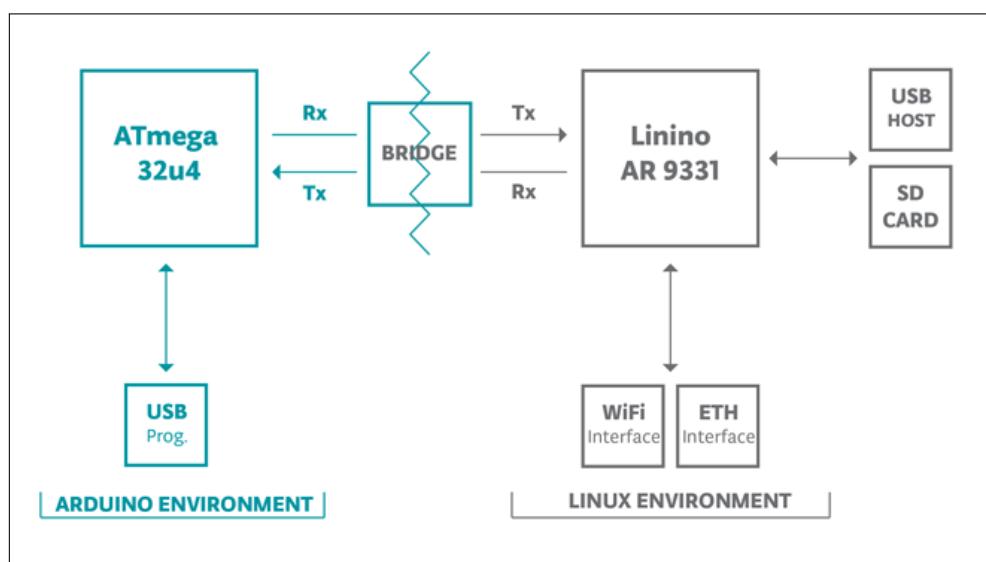


Figura 4: Conexão entre ATmega32u4 e Atheros AR9331 no Arduino Yún

Fonte: Extraído de (ARDUINO, 2016d)

Outra particularidade deste modelo de Arduino é que ele não possui regulador de tensão integrado. Portanto deve ser alimentado pela porta micro-USB ou injetando 5 volts no pino Vin da placa. Outra possibilidade é alimentá-lo por um módulo *Power over Ethernet*

(PoE) vendido separadamente. O *layout* do Arduino Yún é mostrado na Figura 5.

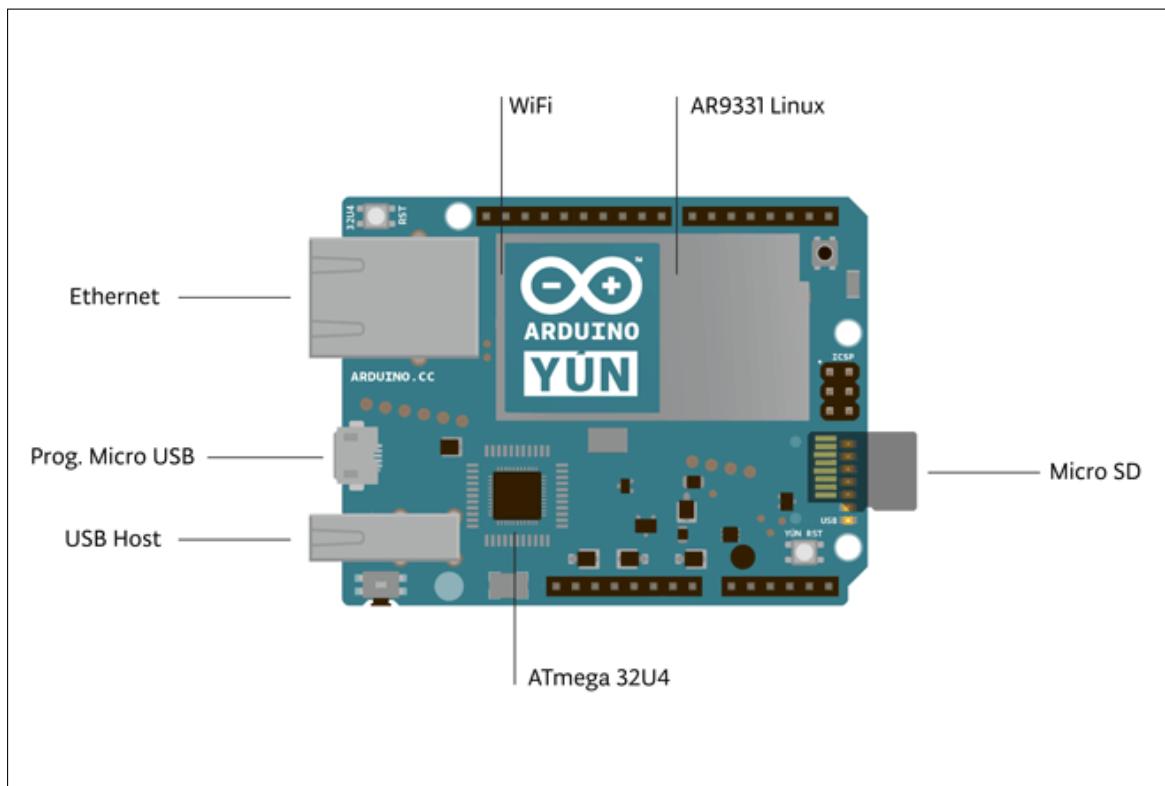


Figura 5: Layout do Arduino Yún

Fonte: Extraído de (ARDUINO, 2016d)

2.3.4 EXEMPLO DE UM PROJETO COM ARDUINO

Um exemplo de projeto desenvolvido para a plataforma Arduino é mostrado na presente seção para fins de demonstração. O projeto consiste em acender um LED enquanto um botão estiver pressionado. O esquemático correspondente à este circuito pode ser visto na Figura 6. O mesmo foi desenvolvido no software Fritzing, de iniciativa *open-source*, que segue o mesmo espírito da plataforma Arduino (FRITZING, 2016).

Uma captura de tela da IDE Arduino contendo o código necessário para o funcionamento do projeto exemplo é mostrado na Figura 7. Os pinos correspondentes ao LED e botão são definidos em constantes no início do código para facilitar sua referência no decorrer do programa. A função `setup()` é executada somente uma vez ao ligar o microcontrolador e é utilizada para a inicialização dos pinos de I/O, configuração de sensores, entre outras atividades que não necessitam ser repetidas. Após sua execução, as instruções contidas na função `loop()` são executadas repetidamente caracterizando o funcionamento contínuo do microcontrolador. Nesta função são lidos sensores, realizado o processamento de acordo com a lógica do

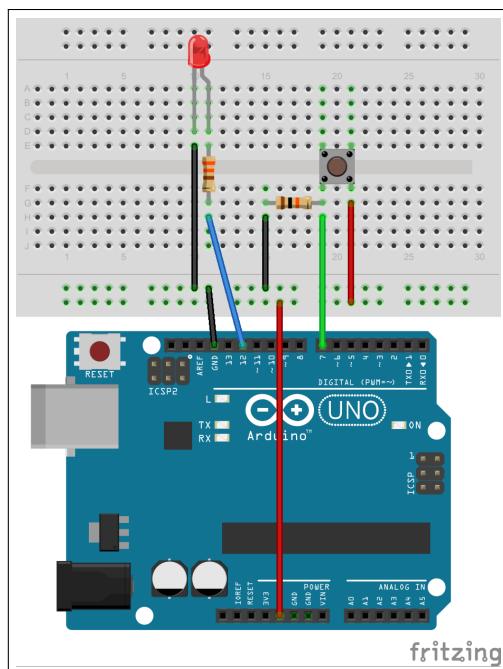


Figura 6: Esquemático Exemplo

Fonte: Autoria Própria

programa, e controlados os atuadores.

2.4 CONVERSÃO ANALÓGICO-DIGITAL

A grande maioria dos sinais no mundo é analógica, podendo assumir qualquer valor em qualquer instante de tempo (SEDRA; SMITH, 2000). Sinais analógicos são contínuos no tempo e em um intervalo de valores, logo têm valores em cada instante de tempo, e esses valores podem ter qualquer amplitude no dado intervalo. Por outro lado, sinais digitais existem apenas em momentos discretos no tempo e podem assumir somente um número finito de valores (LATHI; DING, 2012).

Para trabalhar com sinais analógicos em microcontroladores, que são dispositivos digitais, primeiramente é necessário converter a tensão do sinal no dado momento para um valor em binário que o represente. Esta conversão é feita por meio de conversores analógico-digital ou, em inglês, *Analog to Digital Converter* (ADC). Grande parte dos microcontroladores possuem este periférico incluso em seu circuito integrado, como é o caso dos utilizados nos Arduinos. A conversão analógico-digital (A/D) não é 100% precisa, entretanto, como a percepção humana não exige precisão infinita, a conversão A/D pode capturar a informação necessária de uma fonte analógica de forma satisfatória (LATHI; DING, 2012).

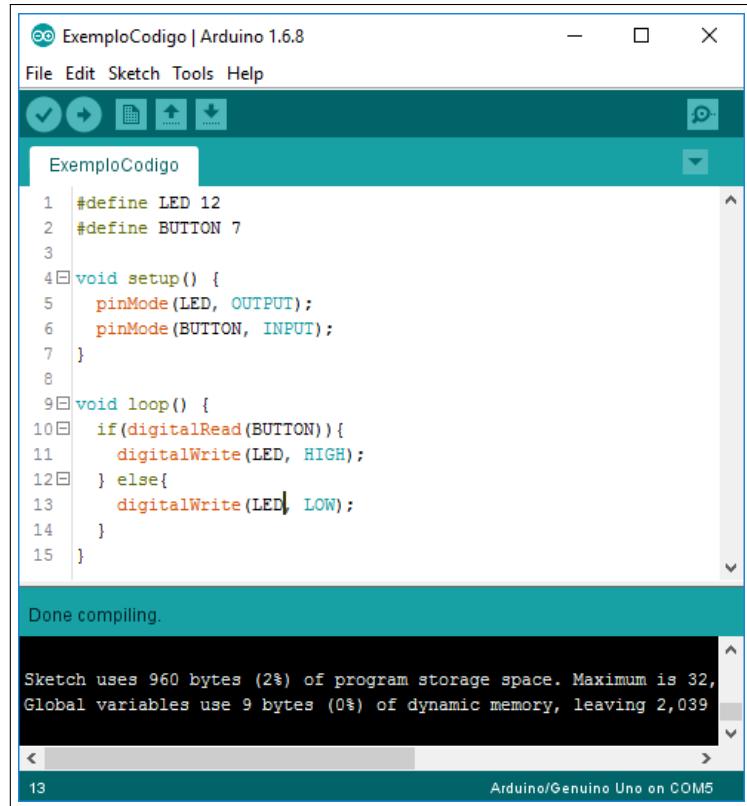


Figura 7: IDE Arduino

Fonte: Autoria Própria

A conversão A/D ocorre em duas etapas: um sinal em tempo contínuo é amostrado produzindo um sinal em tempo discreto cujas amplitudes contínuas são então quantizadas em níveis discretos. O teorema de Nyquist afirma que para que um sinal possa ser reconstruído a partir de suas amostras sem perda de informações, é necessário que a frequência de amostragem seja no mínimo o dobro da maior frequência do espectro do sinal (LATHI; DING, 2012).

Estando o sinal amostrado em intervalos de tempo uniformes, os valores das amostras ainda não se encontram na forma digital, pois assumem valores em um intervalo contínuo. Então a quantização é efetuada, processo no qual cada amostra é aproximada ao nível de quantização mais próximo.

Para um sinal $m(t)$ com valores no intervalo $(-m_p, m_p)$, o quantizador divide este intervalo em L subintervalos. Cada amostra de amplitude é aproximada ao valor médio do subintervalo que a contém, e passa a ser representada por um dos L números. Após estes dois passos, amostragem e quantização, a conversão A/D se conclui. Tais etapas são ilustradas na Figura 8.

A precisão do sinal quantizado pode ser aumentada por meio do incremento do número

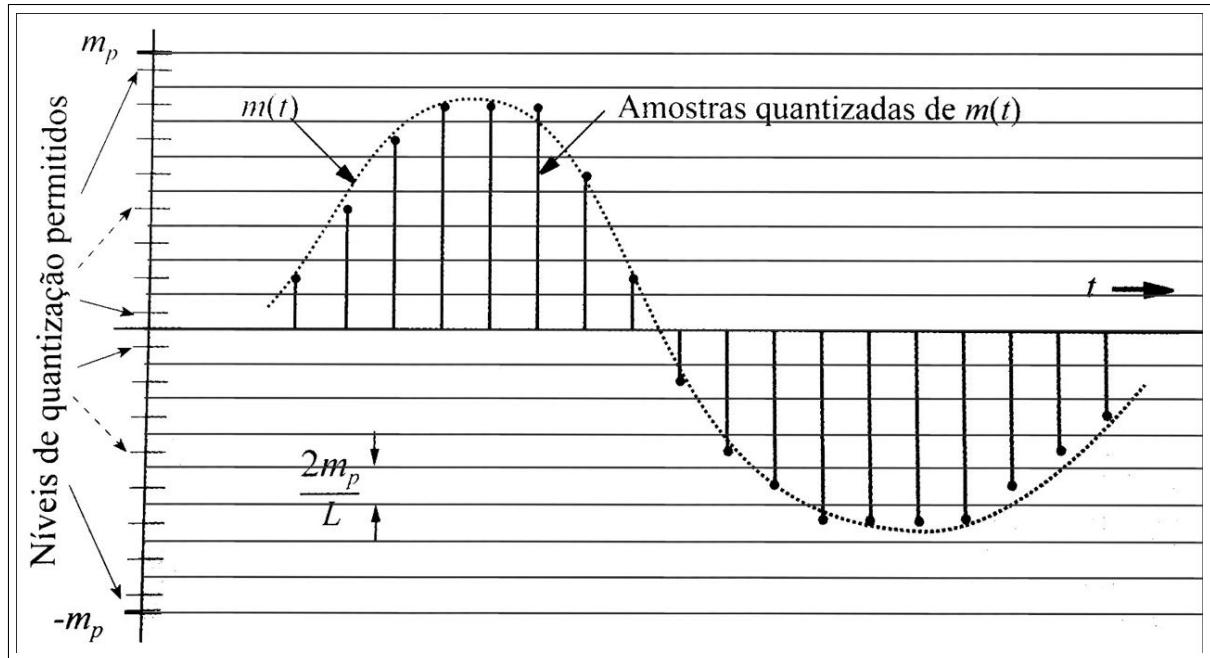


Figura 8: Amostragem e Quantização de um Sinal Analógico

Fonte: Adaptado de (LATHI; DING, 2012)

de níveis L (LATHI; DING, 2012). Na prática, a quantidade de níveis é dada por 2^n , onde n é a quantidade de bits utilizada para representar cada valor de forma digital. Por exemplo, quando temos um ADC de 8 bits, temos 256 níveis (2^8), sendo cada nível representado por uma sequência de 8 bits.

2.5 PRÉ-CONDICIONAMENTO DO SINAL

Para que um conversor A/D possa trabalhar com as mais variadas formas de onda é necessário um tratamento prévio do sinal para que o mesmo se adeque aos valores aceitos pelo conversor. Serão necessários três circuitos de condicionamento do sinal: atenuação, amplificação e *offset*.

2.5.1 CIRCUITO DE ATENUAÇÃO

Para que tensões acima de 5 V possam ser medidas, é necessário abaixar essa tensão para valores entre 0 e 5 V. Este efeito é conseguido através de divisores resistivos. Segundo Boylestad (2007), a tensão através uma série de elementos resistivos se dividirá conforme a magnitude dos níveis de resistência. A proporção de tensão através de resistores em série será a mesma proporção de suas resistências. Uma representação de um divisor resistivo com dois

resistores pode ser visto na Figura 9.

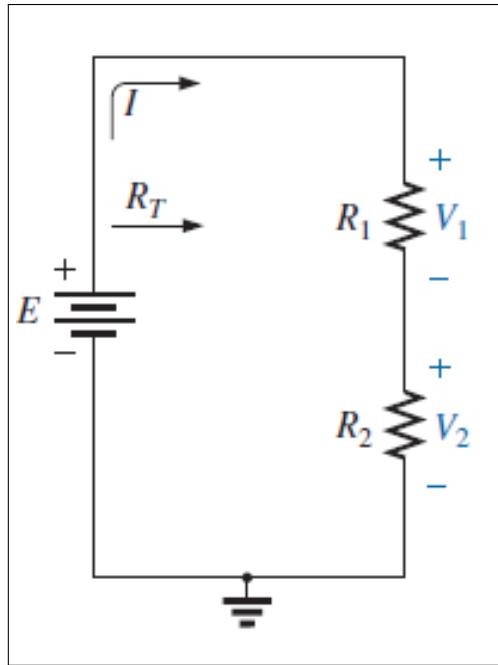


Figura 9: Divisão de Tensão Através de uma Série de Resistores

Fonte: Extraído de (BOYLESTAD, 2007)

A regra do divisor de tensão (Equação 1) permite a determinação da tensão em qualquer resistor de uma série sem necessidade da determinação da corrente através deles. A tensão V_x no resistor x dá-se pela multiplicação de sua resistência (R_x) pela tensão (E) aplicada na série de resistores, dividida pela soma de todas as resistências (R_T).

$$V_x = R_x \frac{V}{R_T} \quad (1)$$

2.5.2 CIRCUITO DE AMPLIFICAÇÃO

Em ocasiões onde se deseja ler um sinal com valor de tensão muito baixo, para que a representação do sinal tenha uma resolução satisfatória é necessário amplificá-lo de forma que ele se estenda pela maior faixa possível dentro da faixa de tensão aceita pelo conversor, usualmente entre 0 e 5 V. Para isso é utilizado o amplificador operacional (Amp-Op) que, de acordo com a combinação de resistores utilizados em sua realimentação, oferece determinados valores de ganho.

Ao amplificar um sinal, deve-se ter cuidado para não modificar a informação contida no mesmo. O intuito é que o sinal de saída do amplificador seja uma réplica exata do sinal

de entrada, exceto sua amplitude (tensão) que deve ser maior. Dá-se à essa propriedade dos amplificadores o nome de linearidade (SEDRA; SMITH, 2000).

A proporção de amplificação do sinal é conhecida como ganho do amplificador e, no caso dos Amp-Ops, pode ser configurado dentro de um limite definido pelas características do componente, de acordo com a combinação de resistores utilizada na realimentação de sua entrada. As duas configurações mais comuns para um Amp-Op são amplificador inversor, o qual tem como saída o sinal amplificado defasado 180º em relação ao sinal de entrada; e amplificador não inversor, que não apresenta tal defasagem (JÚNIOR, 2003). A configuração não inversora é a que melhor atende este projeto pois alterações no sinal não são desejadas.

O circuito de configuração de um Amp-Op como amplificador não inversor pode ser visto na Figura 10. Nesta configuração o sinal de entrada V_{in} é aplicado diretamente ao terminal de entrada positivo (+) do Amp-Op, enquanto um dos terminais de R_2 é conectado ao terra e o outro à entrada não inversora (−). O resistor R_1 realimenta o sinal de saída para a entrada não inversora (−) do amplificador (SEDRA; SMITH, 2000; JÚNIOR, 2003). Nesta configuração, o ganho obtido pelo amplificador dá-se segundo a Equação 2.

$$A = 1 + \frac{R_2}{R_1} \quad (2)$$

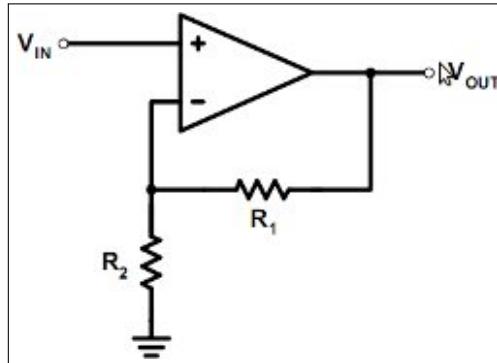


Figura 10: Configuração Não Inversora do Amp-Op

Fonte: Extraído de (ELETRÔNICA, 2016)

2.5.3 CIRCUITO DE OFFSET

Quando se deseja analisar uma forma de onda alternada, onde os valores oscilam acima e abaixo de zero, é necessário que os valores negativos sejam elevados para que fiquem acima de zero pois os conversores A/D alimentados com 5 volts positivos não são capazes de reconhecer valores negativos. Desta forma, o sinal é acrescido de uma tensão contínua fazendo

com que toda a onda fique dentro da faixa de tensão aceita pelo conversor. Este efeito é denominado *offset* e também é conseguido por meio do amplificador operacional (JÚNIOR, 2003; SEDRA; SMITH, 2000). Geralmente, é necessário utilizar o *offset* em conjunto com divisores resistivos para tensões alternadas maiores que a aceita pelo conversor A/D, ou em conjunto com amplificadores para tensões alternadas pequenas, na casa dos milivolts.

Segundo a fabricante de componentes eletrônicos, TEXAS INSTRUMENTS (2008), para se obter um ganho positivo (não inversor) com uma tensão de offset pode-se utilizar o circuito representado na Figura 11.

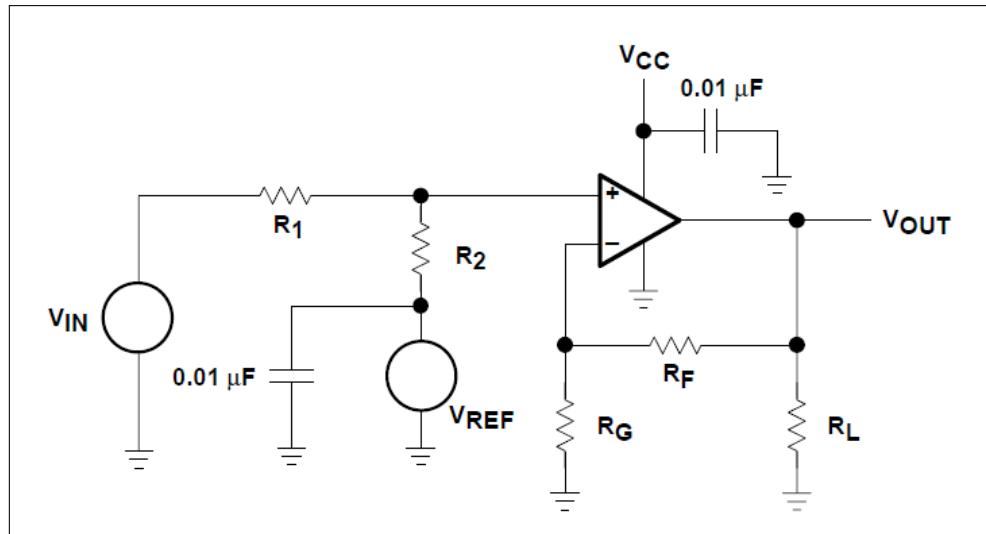


Figura 11: Esquemático do Circuito de Amplificação com Offset

Fonte: Extraído de (TEXAS INSTRUMENTS, 2008)

Dado o ganho linear do amplificador operacional, a equação que representa a saída deste circuito pode ser representada resumidamente por uma equação de reta (Equação 3) onde G é o ganho do circuito e O é a tensão de offset. A equação real do circuito obtida utilizando as regras de divisor de tensão e superposição é mostrada na Equação 4.

$$V_{out} = G \cdot V_{in} + O \quad (3)$$

$$V_{out} = V_{in} \cdot \left(\frac{R_2}{R_1 + R_2} \right) \cdot \left(\frac{R_F + R_G}{R_G} \right) + V_{ref} \cdot \left(\frac{R_1}{R_1 + R_2} \right) \cdot \left(\frac{R_F + R_G}{R_G} \right) \quad (4)$$

Relacionando as Equações 3 e 4 obtém-se as equações que relacionam os valores de ganho (Equação 5) e offset (Equação 6) desejados com os valores dos resistores a serem utilizados.

zados no circuito.

$$G = \left(\frac{R_2}{R_1 + R_2} \right) \cdot \left(\frac{R_F + R_G}{R_G} \right) \quad (5)$$

$$O = V_{ref} \cdot \left(\frac{R_1}{R_1 + R_2} \right) \cdot \left(\frac{R_F + R_G}{R_G} \right) \quad (6)$$

2.6 OSCILOSCÓPIO DIGITAL

O principal propósito de um osciloscópio é demonstrar graficamente sinais elétricos conforme eles variam no tempo. A maior parte dos osciloscópios produz um gráfico de duas dimensões onde o tempo se encontra no eixo x , e a tensão no eixo y (SPARKFUN, 2016). Um exemplo de tela de um osciloscópio é mostrado na Figura 12.

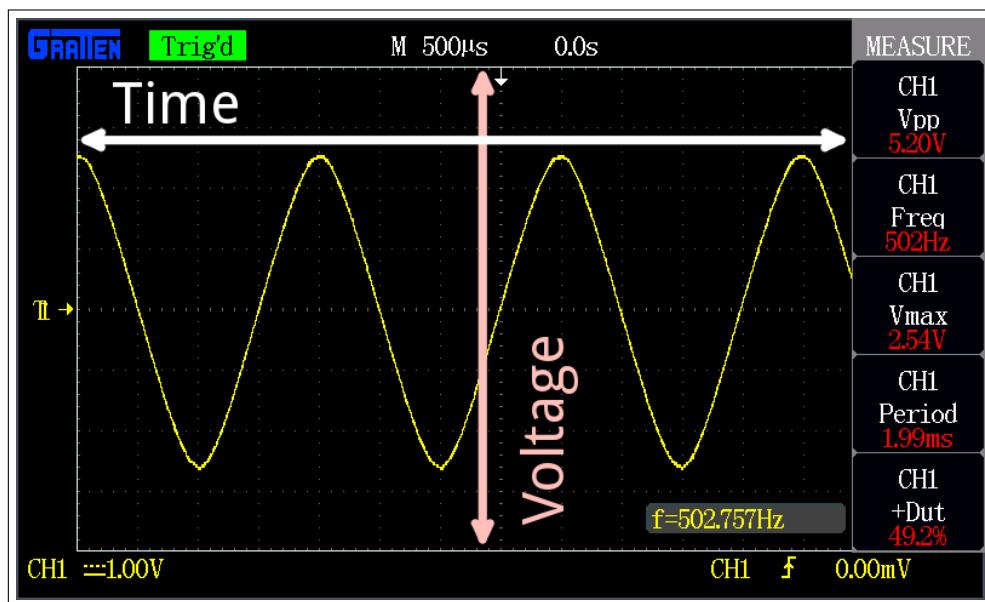


Figura 12: Tela de um Osciloscópio

Fonte: Extraído de (SPARKFUN, 2016)

O instrumento possui controles que permitem ajustar a escala tanto de tempo quanto de tensão. Há também a função *trigger* que ajuda a estabilizar o sinal na tela. Exemplos de grandezas que o osciloscópio consegue medir são frequência, período, *duty cycle*, tempo de subida e descida, amplitude da onda e valores máximo, mínimo e médio de tensão (SPARKFUN, 2016). Além de mostrar a forma de uma onda, com o auxílio do osciloscópio também é possível identificar a presença de ruído no circuito.

Algumas características são fundamentais na análise de um osciloscópio (SPARKFUN, 2016; TEKTRONIX, 2000):

- **Largura de banda:** o quanto rápido um osciloscópio consegue identificar uma mudança de sinal. Especifica o limite de frequência que pode ser confiavelmente lido pelo aparelho;
- **Quantidade de canais:** quantidade de sinais que podem ser lidos e mostrados simultaneamente;
- **Taxa de amostragem:** define quantas vezes por segundo o sinal é lido;
- **Tempo de subida:** determina o pulso de subida do sinal mais rápido que pode ser lido;
- **Tensão máxima de entrada:** a maior tensão que pode ser lida pelo aparelho sem danificá-lo;
- **Resolução:** representa o quanto precisamente a tensão de entrada pode ser lida. Este valor pode mudar conforme a escala vertical é ajustada;
- **Sensibilidade vertical:** representa os valores mínimo e máximo da escala de tensão, listado como volts por divisão;
- **Impedância de entrada:** quando a frequência do sinal é muito alta, mesmo uma pequena impedância adicionada ao circuito pode afetar o sinal. Todo osciloscópio adicionará uma certa impedância ao circuito que está lendo, chamada impedância de entrada.

O sistema de *trigger* de um osciloscópio é dedicado à estabilizar e focar a imagem da onda. O *trigger* indica ao osciloscópio em quais partes do sinal a medida deve ser tomada. Em casos de sinais periódicos, o *trigger* pode ser manipulado para manter a imagem do *display* estática (TEKTRONIX, 2000; SPARKFUN, 2016).

O funcionamento de um osciloscópio digital inicia-se com a amplificação ou atenuação vertical (tensão) do sinal lido para que este fique em conformidade com a faixa de tensão aceita pelo conversor A/D. O próximo passo é a conversão do sinal pelo ADC, no qual amostras do sinal são tomadas em pontos discretos no tempo e suas tensões são convertidas para valores digitais chamados *sample points*. A taxa com que os dados são obtidos é chamada de taxa de amostragem. Os pontos amostrados passam então por um demultiplexador, responsável por

conectar as múltiplas entradas à memória de aquisição, onde são armazenados para posterior processamento. O microprocessador lê os dados da memória, realiza os processamentos necessários e envia os resultados para a memória do *display*, onde finalmente a imagem é exibida (TEKTRONIX, 2000). O diagrama de blocos deste processo é mostrado na Figura 13.

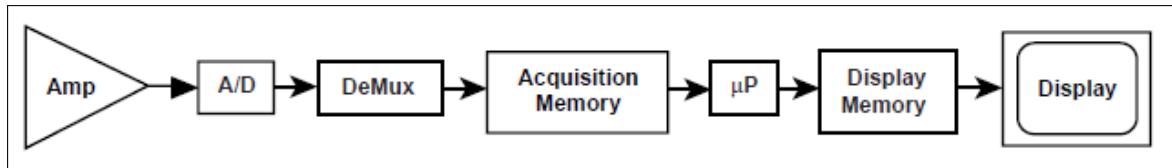


Figura 13: Diagrama de Blocos de um Osciloscópio Digital

Fonte: Extraído de (TEKTRONIX, 2000)

2.7 TRABALHOS RELACIONADOS

Alguns projetos já desenvolvidos apresentam relação com esta proposta. Karim (2014) projetou um osciloscópio de baixo custo com um único canal baseado em Arduino e *display* LCD gráfico. Neste projeto são feitas leituras do sinal e armazenadas na memória RAM até que esta esteja cheia, momento em que os dados são mostrados no *display*. O circuito de condicionamento do sinal utilizado é capaz de receber tensões alternadas (AC) e contínuas (DC) de até 100 volts e convertê-las em tensões entre 0 e 5 volts em corrente contínua (VDC). O autor não implementa a função *trigger* em hardware, mas sim na programação do microcontrolador. Esta solução foi capaz de ler, sem distorção, sinais de frequências entre 10 Hz e 7.7 kHz com uma taxa de amostragem de 10 mil amostras por segundo. O autor destaca que o ADC contido no Atmega328P não é rápido o suficiente para trabalhar com sinais de frequências mais altas.

Em Yi (2010) foi utilizada a placa de som de um computador para a construção de um osciloscópio de dois canais. Com uma taxa de amostragem de 44,1 kHz e 16 bits de resolução, o projeto foi capaz de interpretar sinais entre 10 Hz e 20 kHz. A interface gráfica do osciloscópio foi programada em LabVIEW, contendo todo o processamento do sinal, enquanto a placa de som ficou responsável apenas pela aquisição dos valores por meio da conversão A/D.

No artigo intitulado *Open Source Oscilloscope for Hobby Users*, os autores Niculescu e Lita (2015) afirmam que a taxa de amostragem de 2 MSPS (*mega samples per second*) com 8 bits de resolução é considerada suficiente para utilização em hobbies. Neste trabalho, um osciloscópio de custo menor que 10 dólares é construído baseado em um microcontrolador PIC o qual possui um ADC com taxa de aquisição de 2 MSPS. A exibição dos dados é feita em um computador através de um programa desenvolvido em C# e a comunicação se dá por USB. O

controle dos parâmetros de condicionamento do sinal é realizado através da interface gráfica do programa, sendo interpretado pelo microcontrolador que seleciona a atenuação ou ganho adequado por um multiplexador analógico.

Um sistema de automação residencial integrando as plataformas Arduino e Android foi projetado por Sartori et al. (2015). A comunicação entre os dois dispositivos foi realizada por meio de rede local (LAN), estando o Arduino conectado através de um *shield Ethernet* e o dispositivo Android por Wi-Fi. A solução utilizada foi hospedar uma página PHP em um computador na rede, a qual envia *strings* ao Arduino informando quais equipamentos devem ser ligados ou desligados de acordo com as ações executadas na página. O navegador do dispositivo móvel é utilizado para acessar a página PHP. De forma similar, Lang et al. (2013) desenvolveram um sistema de controle de irrigação gerenciado por um dispositivo móvel através de uma página web hospedada em um computador na rede local. Este computador mantinha sua comunicação com o Arduino por meio de conexão USB.

3 PROPOSTA DE TRABALHO

Nesta seção são descritas as atividades que serão desenvolvidas a fim de alcançar o objetivo deste trabalho sendo subdividida de acordo com as atividades planejadas: montagem do circuito de condicionamento do sinal, montagem do circuito de aquisição do sinal, implementação da comunicação entre o Arduino e o dispositivo móvel, desenvolvimento da aplicação Android, e por fim execução de testes e análise dos resultados. Um diagrama representando a arquitetura da solução proposta pode ser visto na Figura 14.

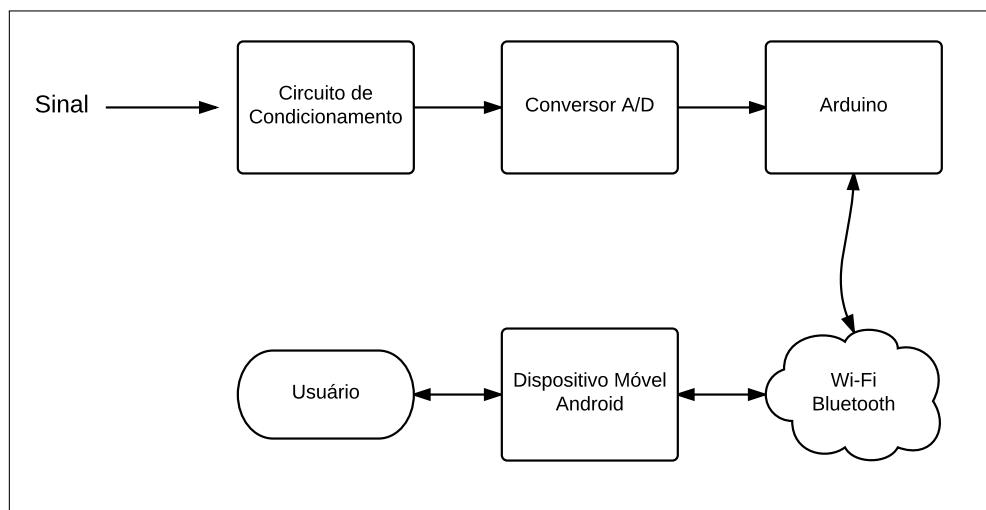


Figura 14: Diagrama da Arquitetura Proposta

Fonte: Autoria própria

3.1 MONTAGEM DO CIRCUITO DE CONDICIONAMENTO DO SINAL

O circuito de condicionamento do sinal a ser desenvolvido efetuará o tratamento do mesmo por meio de combinações entre offset e amplificação ou atenuação, de forma que o sinal de saída deste circuito represente o mesmo formato do sinal de entrada, porém ajustado às condições de tensão de entrada do circuito de aquisição do sinal.

3.2 MONTAGEM DO CIRCUITO DE AQUISIÇÃO DO SINAL

Após o sinal condicionado de acordo com a faixa de tensão suportada pelo conversor analógico-digital, este componente faz a amostragem do sinal, representando o valor analógico lido em um valor digital correspondente. O circuito de aquisição do sinal consiste em um con-

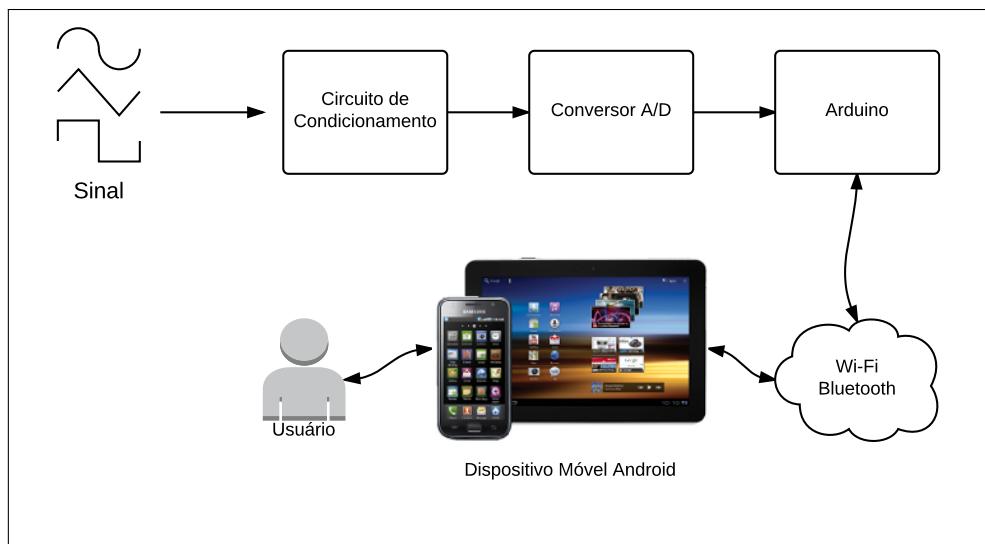


Figura 15: Diagrama da Arquitetura Proposta

Fonte: Autoria própria

versor analógico-digital, podendo este ser um componente discreto, ou interno ao microcontrolador. Serão utilizadas as duas opções para fins de comparação de desempenho e acurácia.

3.3 COMUNICAÇÃO ENTRE ARDUINO E DISPOSITIVO MÓVEL

Os meios de comunicação entre o Arduino e o dispositivo móvel disponíveis são: Bluetooth por meio de um *shield* para o Arduino; Wi-Fi também por meio de um *shield*, ou nativamente para o Arduino YUN; e USB podendo ser de duas formas, na primeira o dispositivo móvel atuando como host por meio de um adaptador USB OTG, e a segunda o Arduino YUN atuando como host por meio de sua porta USB nativa.

Cada opção será estudada e verificada sua viabilidade a fim de encontrar a melhor solução para o projeto. O intuito principal é que o projeto seja portátil, portanto as conexões sem fio Bluetooth e Wi-Fi são preferidas.

3.4 DESENVOLVIMENTO DA APLICAÇÃO ANDROID

Será desenvolvida uma aplicação que possui como função principal a exibição da forma de onda representante do sinal na tela do smartphone. Esta aplicação receberá os dados do microcontrolador, fará o tratamento necessário e os exibirá em forma de gráficos 2D, da mesma forma em que são mostrados em osciloscópios profissionais. A manipulação da forma de exibição do sinal, como por exemplo alteração de escala, também será feita por meio de

botões na interface gráfica que se assemelham a um osciloscópio real.

3.5 EXECUÇÃO DE TESTES E ANÁLISE DOS RESULTADOS

Testes serão executados a fim de verificar a eficiência do canal de comunicação entre o microcontrolador e o smartphone (sem lag, travamento), bem como se as formas de ondas obtidas pelo dispositivo construído são semelhantes às de um osciloscópio verdadeiro.

Um gerador de função será utilizado para gerar ondas de diversos formatos, frequências e amplitudes, sendo assim possível identificar os limites do projeto construído.

4 CRONOGRAMA

Na Tabela 1 é apresentado o cronograma das atividades previstas para este trabalho, sendo este distribuído nos meses de março a dezembro de 2016.

Tabela 1: Cronograma de Atividades

Atividades	Meses (Ano 2016)									
	3	4	5	6	7	8	9	10	11	12
Revisão Bibliográfica	X	X								
Elaboração da Proposta	X	X								
Desenvolvimento dos Circuitos Eletrônicos			X	X	X					
Desenvolvimento da Comunicação Entre Arduino e Dispositivo Móvel						X	X			
Desenvolvimento do Aplicativo Android				X	X	X				
Testes e Análise dos Resultados						X	X			
Escrita da Monografia	X	X	X	X	X	X	X			
Elaboração da Apresentação de Defesa							X	X		

Fonte: Autoria Própria

REFERÊNCIAS

ABLESON, W. F. et al. **Android in Action**. 3. ed. Shelter Island, NY: Manning Publications Co., 2012. ISBN 9781617290503.

ARDUINO. **Arduino - Introduction**. 2016. Disponível em: <<http://www.arduino.cc/en/Guide/Introduction>>. Acesso em: 23 de abril de 2016.

ARDUINO. **Arduino Mega 2560**. 2016. Disponível em: <<https://www.arduino.cc/en/Main/ArduinoBoardMega2560>>. Acesso em: 24 de abril de 2016.

ARDUINO. **Arduino Uno**. 2016. Disponível em: <<http://www.arduino.cc/en/Main/ArduinoBoardUno>>. Acesso em: 24 de abril de 2016.

ARDUINO. **Arduino Yún**. 2016. Disponível em: <<https://www.arduino.cc/en/Main/ArduinoBoardYun>>. Acesso em: 24 de abril de 2016.

ATMEL. **Atmel 8-bit Microcontroller with 4/8/16/32 kbytes in-system Programmable Flash Datasheet**. 1600 Technology Drive, San Jose, CA 95110 USA: Atmel, 2015.

BAYLE, J. **C Programming for Arduino**. Birmingham B3 2PB, UK: Packt Publishing, 2013. ISBN 978-1-84951-758-4.

BOUNI. **Arduino-Pinout**. 2015. Disponível em: <<https://github.com/Bouni/Arduino-Pinout>>. Acesso em: 24 de abril de 2016.

BOYLESTAD, R. L. **Introductory Circuit Analysis**. 11. ed. Upper Saddle River, New Jersey: Pearson, 2007. ISBN 0-13-173044-4.

BOYLESTAD, R. L.; NASHELSKY, L. **Dispositivos Eletrônicos e Teoria de Circuitos**. 8. ed. São Paulo, SP: Pearson, 2004. ISBN 978-85-87918-22-2.

CAVALCANTE, B. M. **Apoio a Localização de Defeitos Durante a Execução de Sequências de Teste para Aplicações Android**. Dissertação (Trabalho de Conclusão de Curso) — Universidade Tecnológica Federal do Paraná, 2015.

DEITEL, P. et al. **Android para Programadores**. Porto Alegre, RS: Bookman, 2013. ISBN 9788540702103.

ELETROÔNICA. **Amplificadores Operacionais AmpOp**. 2016. Disponível em: <<http://www.electronica-pt.com/amplificadores-operacionais-ampop>>. Acesso em: 30 de abril de 2016.

EMARKETER. **Smartphone users worldwide 2014-2019**. 2016. Disponível em: <<http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>>. Acesso em: 30 de abril de 2016.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. **Arduino in Action**. Shelter Island, NY: Manning Publications Co., 2013. ISBN 9781617290244.

FARTO, G. de C. **Uma Contribuição ao Teste Baseado em Modelo no Contexto de Aplicações Móveis**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2016.

FRITZING. **Fritzing: electronics made easy**. 2016. Disponível em: <<http://fritzing.org/home/>>. Acesso em: 4 de maio de 2016.

IDC. **Smartphone OS Market Share, 2015 Q2**. 2015. Disponível em: <<http://www.idc.com/prodserv/smartphone-os-market-share.jsp>>. Acesso em: 22 de abril de 2016.

JÚNIOR, A. P. **Eletrônica Analógica: Amplificadores Operacionais e Filtros Ativos**. 6. ed. São Paulo, SP: Bookman, 2003. ISBN 978-85-363-0190-7.

KARIM, I. A. A low cost portable oscilloscope based on arduino and glcd. In: **Informatics, Electronics Vision (ICIEV), 2014 International Conference on**. [S.l.: s.n.], 2014. p. 1–4.

KUZNETSOV, S.; PAULOS, E. Rise of the expert amateur: Diy projects, communities, and cultures. In: **Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries**. New York, NY, USA: ACM, 2010. (NordiCHI '10), p. 295–304. ISBN 978-1-60558-934-3. Disponível em: <<http://doi.acm.org/10.1145/1868914.1868950>>.

LANG, A. G.; DUTRA, A. F. A.; WAROMBY, P. H. **Desenvolvimento de um Irrigador Microcontrolado Integrado a um Sistema de Supervisão e Controle**. Dissertação (Projeto de Pesquisa) — Universidade Tecnológica Federal do Paraná, 2013.

LATHI, B. P.; DING, Z. **Sistemas de Comunicações Analógicos e Digitais Modernos**. 4. ed. Rio de Janeiro, RJ: LTC, 2012. ISBN 978-85-216-2027-3.

LECHETA, R. R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. 5. ed. São Paulo, SP: Novatec Editora Ltda., 2015. ISBN 9788575224687.

MORIMOTO, C. E. **A História da informática (Parte 6: Sistemas embarcados e supercomputadores)**. aug 2011. Disponível em: <<http://www.hardware.com.br/guias/historia-informatica/transistor.html>>. Acesso em: 1 de maio de 2016.

NICULESCU, V.; LITA, A. I. Open source oscilloscope for hobby users. In: **2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER)**. [S.l.: s.n.], 2015. p. 203–207. ISSN 2068-1038.

OPEN HANDSET ALLIANCE. **Alliance Members**. 2016. Disponível em: <http://www.openhandsetalliance.com/oha_members.html>. Acesso em: 22 de abril de 2016.

SARTORI, G.; MOLINA, L. A.; LIMA, W. C. G. de. **Desenvolvimento de um Sistema Microcontrolado de Baixo Custo Utilizando Smartphone para Aplicações de Automação residencial**. Dissertação (Trabalho de Conclusão de Curso) — Universidade Tecnológica Federal do Paraná, 2015.

SEDRA, A. S.; SMITH, K. C. **Microeletrônica**. 4. ed. São Paulo, SP: Makron Books, 2000. ISBN 85.346.1044-4.

SPARKFUN. **How to Use an Oscilloscope**. 2016. Disponível em: <<https://learn.sparkfun.com/tutorials/how-to-use-an-oscilloscope>>. Acesso em: 30 de abril de 2016.

TEKTRONIX. **XYZs of Oscilloscopes**. Beaverton, Oregon, USA: [s.n.], 2000. Disponível em: <<http://ecee.colorado.edu/mcclurel/txyzscopes.pdf>>.

TEXAS INSTRUMENTS. **Single-Supply Op Amp Design Techniques**. [S.l.: s.n.], 2008.

VENKATASHIAH, L. **Android Architecture: Layers in the Android Stack**. jun 2014. Disponível em: <<https://lokeshv.wordpress.com/tag/android/>>. Acesso em: 22 de abril de 2016.

YI, D. Research and design of virtual oscilloscope based on sound card. In: **Electrical and Control Engineering (ICECE), 2010 International Conference on**. [S.l.: s.n.], 2010. p. 1566–1569.