UNIVERSIDADE DO ESTADO DA BAHIA

DEPARTAMENTO DE CIÊNCIAS EXATAS E DA TERRA

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

SAMUEL SANTIAGO DE CARVALHO

AVALIAÇÃO DA EXATIDÃO E PRECISÃO DOS RELÓGIOS DE PLACAS MICROCONTROLADORAS ACOPLADAS AO ARDUINO UNO.

SALVADOR

2020

SAMUEL SANTIAGO DE CARVALHO

AVALIAÇÃO DA EXATIDÃO E PRECISÃO DOS RELÓGIOS DE PLACAS MICROCONTROLADORAS ACOPLADAS AO ARDUINO UNO.

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito à obtenção do grau de Bacharel em Sistemas de Informação. Área de Concentração: Ciência da Computação

Orientador: Cláudio Alves De Amorim

SALVADOR

2020

FICHA CATALOGRÁFICA

Sistema de Bibliotecas da UNEB
Dados fornecidos pelo autor

C331a Carvalho, Samuel Santiago de

Avaliação da exatidão e precisão dos relógios de placas microcontroladoras acopladas ao Arduino UNO. / Samuel Santiago de Carvalho.-- Salvador, 2020.

42 fls.

Orientador(a): Cláudio Alves de Amorim.

Inclui Referências

TCC (Graduação - Sistemas de Informação) - Universidade do Estado da Bahia. Departamento de Ciências Exatas e da Terra. Câmpus I. 2020.

1.Exatidão. 2.Precisão. 3.Tempo. 4.Arduino. 5.Microcontrolador.

CDD: 003

SAMUEL SANTIAGO DE CARVALHO

AVALIAÇÃO DA EXATIDÃO E PRECISÃO DOS RELÓGIOS DE PLACAS MICROCONTROLADORAS ACOPLADAS AO ARDUINO UNO.

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito à obtenção do grau de Bacharel em Sistemas de Informação. Área de Concentração: Ciência da Computação

Aprovada em:

BANCA EXAMINADORA

Cláudio Alves De Amorim (Orientador)
Universidade do Estado da Bahia – UNEB

Diego Gervásio Frias Suarez
Universidade do Estado da Bahia – UNEB

Maria Inés Valderrama Restovic
Universidade do Estado da Bahia – UNEB

É sempre assim o curso dos fatos que movem as rodas do mundo: as mãos pequenas os realizam porque precisam, enquanto os olhos dos grandes estão voltados para outros lugares.

(J.R.R. Tolkien)

RESUMO

Atualmente diversos experimentos físicos são realizados e analisados utilizando como facilitadores os microcontroladores – circuitos integrados capazes de serem programados para desempenhar tarefas específicas – facilitando assim a averiguação das variáveis relacionadas a esses. Neste contexto, o tempo desempenha uma relevância significativa, repercutindo em uma necessidade por exatidão e precisão para proporcionar resultados próximos da realidade. O presente trabalho tem como objetivo por meio de dois experimentos diferentes encontrar as variações dos Arduinos, realizando então um comparativo com os mesmo e também verificando o impacto destes resultados em um experimento já existente. Os resultados dessa pesquisa revelaram que a variação encontrada não foi tão grande quanto o esperado, mas que não obstante, apresenta um resultado importante para futuras pesquisas e até mesmo para aplicações que necessitam entender essas variações.

Palavras-chave: Exatidão, Precisão, Tempo, DS3231, Microcontrolador, Arduino

ABSTRACT

Currently several physical experiments are performed and analyzed using as facilitators microcontrollers - integrated circuits that can be programmed to perform specific tasks - therefore facilitating the investigation of the variables related to these. In this context, time plays a significant relevance, reflecting a need for accuracy and precision to provide results close to reality. This work aims to find the Arduino variations by means of two different experiments, making a comparison with them and also verifying the impact of these results in an existing experiment. The results of this research revealed that the variation found was not as great as expected, but that nonetheless, it presents an important result for future research and even for applications that need to understand these variations.

Keywords: Accuracy, Precision, Time, DS3231, Microcontroller, Arduino

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CPU	Unidade Central de Processamento
EEPROM	Memória Programável Somente de Leitura Apagável Eletricamente
IDE	*Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado
LED	*Light-Emitting Diode*, ou Diodo Emissor de Luz
ppm	partes por milhão
PWM	*Pulse Width Modulation*, ou Modulação de Largura por Pulso
RTC	*Real Time Clock*, ou Relógio de Tempo Real
SCL	Clock Serial
SDA	Dados Seriais
USB	*Universal Serial Bus*, ou Porta Universal

SUMÁRIO

1 INTRODUÇÃO

A metodologia experimental de aquisição de dados por computador representa uma possibilidade real, pois dessa forma traz uma melhoria na precisão dos resultados obtidos, redução no tempo de coleta de dados e a rápida representação dos mesmos em gráficos, facilitando assim na comparação de resultados (CAVALCANTE; TAVOLARO; MOLISANI, 2011).

Diante disso é possível visualizar a grande quantidade de experimentos didáticos de física que são realizados e analisados utilizando placas Arduíno que são capazes de medir intervalos de tempos na faixa de mili e microssegundos, o que não é viável por meios manuais (JUNIO; BORGES; NASCIMENTO, 2018).

Muitas pesquisas do âmbito da física utilizando esses microcontroladores dependem das medições de tempo, logo, é necessário validar se realmente é entendido o funcionamento das funções internas de temporização dos microcontroladores.

Para a avaliação dessas funções, primeiramente apresentou-se necessário entender como funciona um temporizador de um microcontrolador, detalhando e especificando suas funcionalidades para compreendermos as variações que possam ocorrer e os motivos associados a essas anomalias.

Em seguida, analisou-se dois aspectos importantes: a exatidão e a precisão. O conceito de exatidão se dá pelo quão mais próximo o resultado está do valor referencial, ou seja, em um experimento, o tempo deve ser o mais próximo possível do esperado para ser exato. Já a precisão, se dá por repetidas análises, os resultados serem semelhantes mas não necessariamente próximos do valor de referência. Neste contexto, as analises dos relógios internos dos microcontroladores, fundamentou-se na averiguação dos resultados obtidos em relação aos dados reais e na verificação de conformidade em uma grande sequência de testes, garantindo assim uma precisão e exatidão. (THOMSEN, 1997)

Neste trabalho, foram analisados todos os fatores previamente citados, a partir de experimentos e avaliações do tempo utilizando os conceitos de medidas e incertezas para a avaliação do temporizador do microcontrolador. Para a apreciação foi utilizado como referência

inicial o *Real Time Clock*, ou Relógio de Tempo Real (RTC) DS3231[1].

Após todos os dados serem coletados foram analisadas a exatidão e a precisão utilizando o RTC como padrão, possibilitando assim uma avaliação de confiabilidade dos trabalhos que se baseiam no temporizador dos microcontroladores.

Para o objetivo acima ser cumprido foi necessário avaliar experimentalmente, os temporizadores de placas microcontroladoras amplamente utilizadas em experimentos didáticos de física. Inicialmente verificando a viabilidade do RTC 3231 como referência para a pesquisa; realizando experimentos com os microcontroladores coletando os dados, tendo o RTC como referência; coletando e analisando os dados obtidos e, por fim, a avaliação da exatidão e precisão dos sistemas internos de temporizações dos microcontroladores.

Esse trabalho então, vem com o desafio de entender os relógios internos dos microcontroladores e como eles funcionam. É também necessário avaliar quão precisos e exatos eles são para poder de fato demostrar o quão confiável é utiliza-los em experimentos. Sua relevância para área da física consiste em indicar o possível impacto dos temporizadores internos sobre a precisão e exatidão dos experimentos didáticos.

[1] Relógio de tempo real de alta precisão e baixo consumo de energia. Em sua placa vem embutido um sensor de temperatura e um cristal oscilador para melhorar sua exatidão

2 PLACAS DE DESENVOLVIMENTO ARDUINO

A plataforma Arduino desenvolvida na Itália, sobre a modalidade *open source*[1], se difundiu rapidamente pelo mundo e desde cedo se percebeu o seu enorme potencial para a educação, tornando-se uma alternativa economicamente viável frente aos laboratórios didáticos automatizados comercializados atualmente. Seu design permite ligá-lo a diversos sensores como, por exemplo: temperatura, luz, velocidade, pressão barométrica, unidade do ar entre outros. Também temos como possibilidade de saída controle de *Light-Emitting Diode*, ou Diodo Emissor de Luz (LED), motores, auto-falante e *display*. A interface com o computador pode ser via entrada/saída *Universal Serial Bus*, ou Porta Universal (USB)/WiFi/Bluetooth. É possível, também, armazenar os dados em um cartão de memória MicroSD e, a posteriori, recolher os dados. Dessa forma, exclui-se a necessidade da placa estar ligada a um microcomputador (MCROBERTS, 2011).

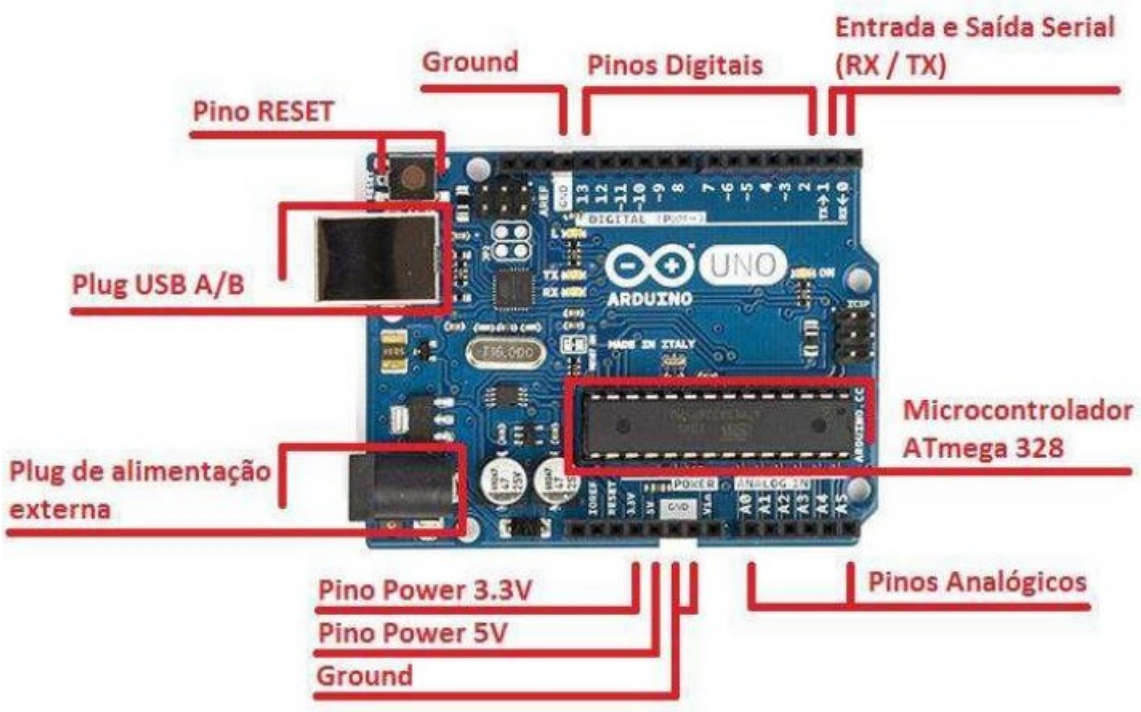
Com o advento da plataforma Arduino, é possível visualizar a facilidade que os microcontroladores trazem para a área computacional e seu potencial para tornar essa área acessível. Alicerçado neste conhecimento, muitos trabalhos científicos, sobretudo os didáticos, utilizam microcontroladores para facilitar a obtenção dos dados e análise dos mesmos. Deste fator nasce a importância de compreender em detalhes o funcionamento dos microcontroladores uma vez que trabalhos didáticos em grande parte contemplam pesquisas relacionadas com análises de experimentos em que o tempo é fator crítico.

Para o desenvolvimento desse trabalho foi especificado um modelo especifico do Arduino comum no mercado. O Arduino UNO que possui um preço médio de 55 reais (ELETROGATE, 2020), e que possui uma boa quantidade de portas digitais e analógicas, uma boa relação custo benefício e facilidade para realização da compra em sites onlines, por esses motivos ele é tão utilizado atualmente e se torna um ótimo objeto de estudo.

O Arduino UNO possui um microcontrolador ATmega 328P que é responsável pelos controles de sinais digitais e analógicos nos pinos e a comunicação serial com os equipamentos através da porta USB. A placa pode ser alimentada pelo cabo USB A/B, ou até mesmo através

[1] Open source é um termo em inglês que significa código aberto. Isso diz respeito ao código-fonte de um software, que pode ser adaptado para diferentes fins.

Figura 1 – Disposição dos principais componentes da placa Arduino UNO.



Fonte: (BORGES, 2011)

de uma fonte externa que pode ser conectada ao seu *plug*. Dentro dos 14 pinos digitais existentes na placa Arduino UNO, 6 deles possuem função *Pulse Width Modulation*, ou Modulação de Largura por Pulso (PWM), que é a mudança de duração do pulso, ou seja, tempo que um sinal digital permanece em nível lógico 1 (indicado por uma tensão de 5 volts) ou em nível lógico 0 (indicado por uma tensão de 0 volts) (NERI, 2014).

2.1 PROCESSADOR ATMEGA 328P

Segundo Pereira (2013) a principal característica do microcontrolador esta em reunirem um só chip, todos os periféricos necessários para o projeto e a fabricação de dispositivos eletrônicos do mais diversos tipos, desde simples sinalizadores e luzes pisca-pisca até equipamentos médicos sofisticados. Ele além de ser uma unidade central de processamento, possui memória e periféricos programáveis de entrada e saída que o tornam capaz de exercer múltiplas funções.

O ATmega 328P é um microcontrolador de arquitetura avançada, de alto desempenho, baixo consumo e otimizado para compiladores, possuindo 131 instruções, a maioria podendo ser executada em apenas um único ciclo de relógio, por exemplo, um microprocessador

utilizando um relógio de 20 MHz terá um ciclo relativo a 50 nanosegundos executando então as instruções nesse tempo, um banco de 32x8 registros de uso geral, possui até 20 MIPS (Milhões de instruções por segundo) a 20 MHz, memória FLASH de 32 KB, programável dentro do sistema, 2 temporizadores de 8 bits e um de 16 bits (ATMEL, 2014).

Este microcontrolador usa a arquitetura Harvard, sendo ela mais recente que a de Von Neumann e que se distingue das outras por possuir duas memórias diferentes e independentes em termos de barramento e ligação ao processador. A Unidade Central de Processamento (CPU) usa um pipeline de um nível, ou seja, enquanto está executando uma instrução, a próxima está sendo acessada da memória para poder ser executada posteriormente, fazendo com que o microcontrolador execute as instruções em um único ciclo de relógio (ATMEL, 2014).

Como foi dito anteriormente o ATmega 328P possui 3 temporizadores, sendo dois deles de 8 bits (TIMER0 e TIMER2) e um de 16 bits (TIMER1). Eles são usados para contagem de eventos externos, geração de sinais PWM, interrupções periódicas e medidas de intervalos de pulsos. Cada temporizador possui características próprias e a biblioteca Arduino utiliza muito delas em funções como *millis(), micros(), delay()*. Na tabela abaixo é possível verificar os temporizadores e seus respectivos detalhes e diferenças.

Tabela 1 – Tabela comparativa entre temporizadores

TEMPORIZADORES (ATmega 328P)			
Temporizadores	**Tamanho**	**Interrupção**	**Funções e pinos no Arduino UNO que utilizam o temporizador**
TIMER0	8 bits (0 - 255)	Compare Match, Overflow	delay(), millis(), micros(), analogWrite(), pinos 5 e 6
TIMER1	16 bits (0 - 65535)	Compare Match, Overflow, Input Capture	Funções da biblioteca servo, analogWrite(), pinos 9 e 10
TIMER2	8 bits (0-255)	Compare Match, Overflow	tone(), analogWrite(), pinos 3 e 11

Fonte: O autor

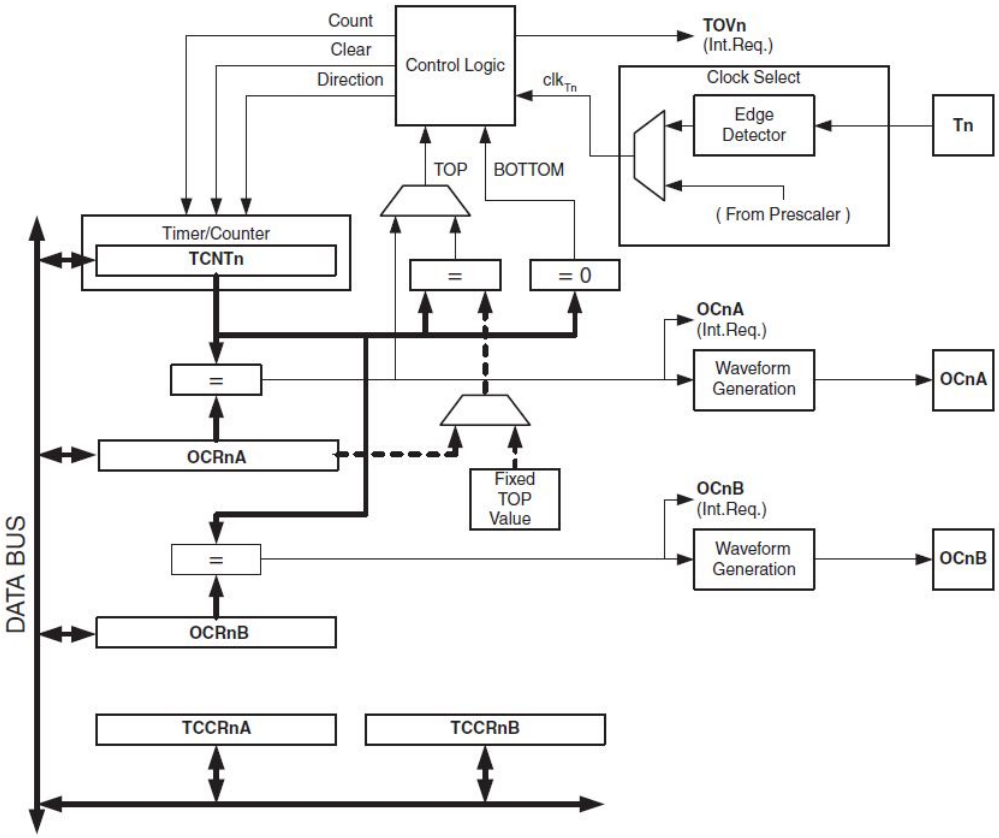
Os temporizadores possuem detalhes e especificações como pode ser visto, e um ponto importante a destacar é como funcionam as interrupções desses temporizadores. Interrupção é um evento que obriga o microprocessador a suspender suas atividades temporariamente para atender uma rotina associada ao evento.

A primeira forma possível de gerar interrupção é pelo modo de comparação (*Com-*

pare Match) na medida que o temporizador estiver incrementando com o ciclo de máquina, será necessário atribuir um valor ao registrador do microcontrolador e se a comparação entre o registrador e o temporizador for igual, irá gerar um vetor de interrupção. Outra forma de gerar é por estouro (*Overflow*) que ocorre quando o temporizador passa do valor máximo do tamanho respectivo a ele e retorna a 0. E por ultimo o método por mudança de nível lógico (*Input Capture*), que apenas o TIMER1 possui, e pode ser utilizado em algum momento específico da contagem do temporizador (ATMEL, 2014).

Na Figura 2 é possível visualizar o diagrama de bloco do TIMER0, onde podemos visualizar o modo de comparação que irá verificar continuamente se o registro TCNTn é igual os registros OCRnA ou OCRnB, caso esses registros sejam iguais o comparador irá sinalizar uma igualdade acionando o registro OCnA ou OCnB. E o modo por estouro que é o normal e o mais simples, o qual no diagrama pode ser visualizado que o TCNTn irá ser comparado com o valor programado e caso ele seja igual irá ativar o registro TOVn de transbordamento.

Figura 2 – Diagrama de bloco do temporizador TIMER0

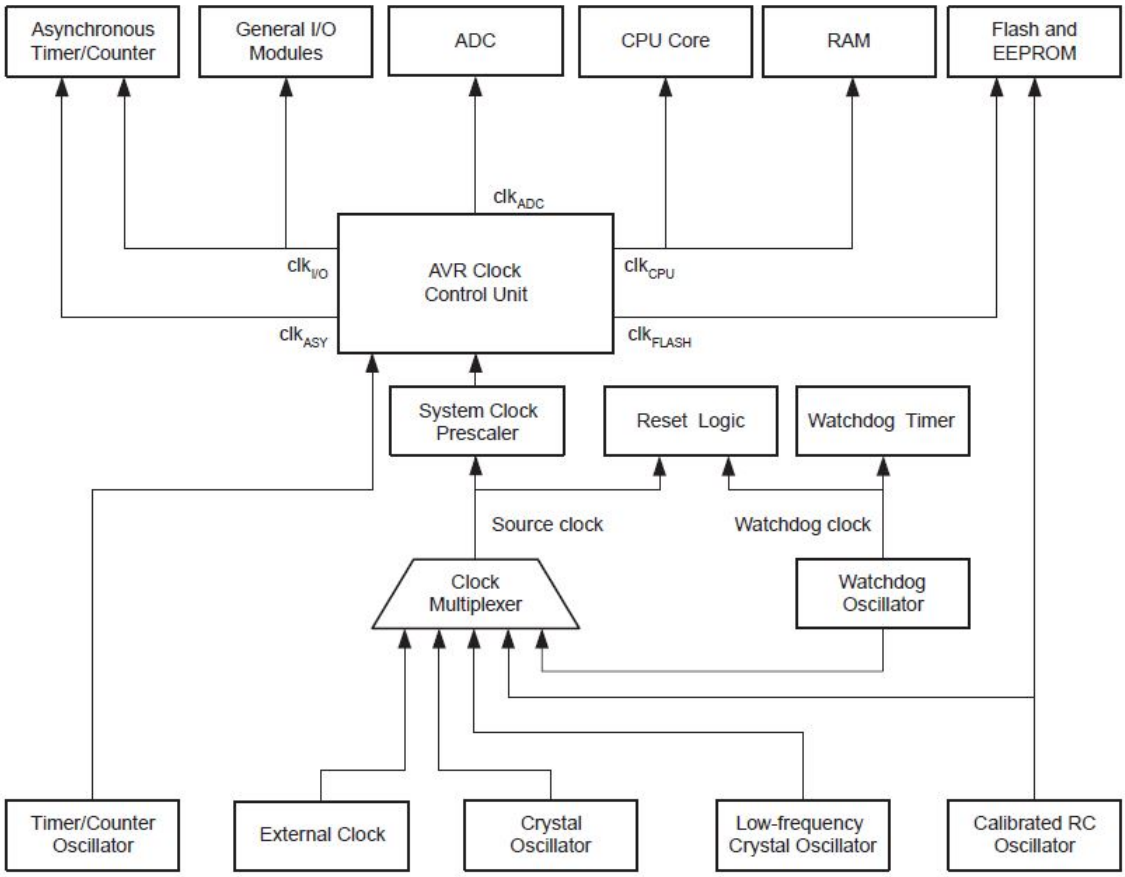


Fonte: (ATMEL, 2014)

Os temporizadores necessitam de uma peça importante do microcontrolador que são

seus relógios. É possível visualiza-los na parte inferior da Figura 3 e sua distribuição, sabendo que nem todos eles necessitam estarem ativados no mesmo momento e para reduzir o consumo, os módulos que não estão sendo usados podem ser interrompidos.

Figura 3 – Relógios do microcontrolador e sua distribuição



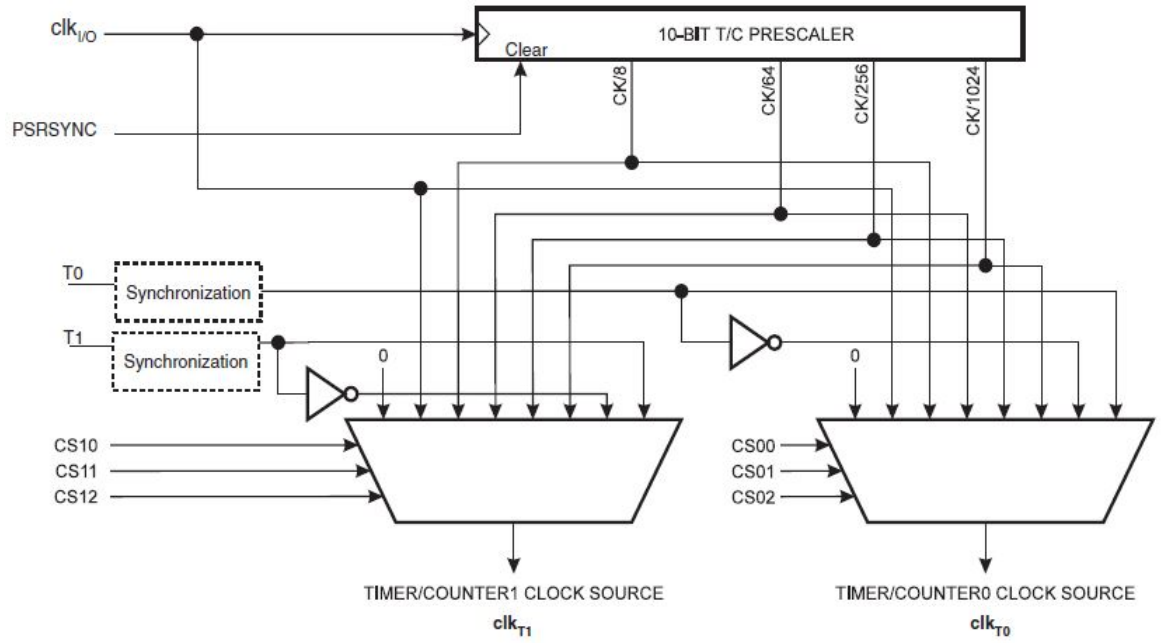
Fonte: (ATMEL, 2014)

Gadre (2001) diz que essa distribuição suporta as seguintes opções de *clock*: cristal ou ressonador cerâmico externo, cristal externo de baixa frequência e sinal de *clock* externo. Sendo estes cristais componentes eletrônicos que utilizam a ressonância de um cristal em vibração para criar sinal elétrico com uma frequência.

Como esse trabalho irá contemplar as funções mais utilizadas para temporização do Arduino como por exemplo as funções *millis(), micros() e delay()*, é preciso se atentar a análise para o TIMER0 responsável por essas funções. Esse temporizador pode ser ajustado diretamente no relógio do sistema fornecendo uma operação mais rápida, com uma frequência do temporizador igual a do relógio do sistema, pode ser escolhido o relógio pré-calibrado (*prescaler*) com diferentes frequências ou um pino externo. Quando o temporizador é externo é sincronizado

com a frequência do oscilador da CPU para garantir uma amostragem adequada do relógio, o tempo mínimo entre os dois e as transações devem ter pelo menos um período do *clock* interno da CPU (GADRE, 2001), na Figura 4 é possível visualizar o TIMER0 e TIMER1 com suas possibilidades de *clocks* incluindo o *prescaler*.

Figura 4 – *Prescaler* para o TIMER0 e TIMER1



Fonte: (ATMEL, 2014)

2.2 INTERRUPÇÕES

Para se tornar mais eficiente as solicitações de operações enviadas pelos periféricos para os microcontroladores assincronamente ou até mesmo periodicamente, existe um recurso denominado interrupção. Utilizando essa técnica de comunicação, os dispositivos periféricos podem informar a ocorrência de um dado evento externo que será tratado prioritariamente.

Em geral os microprocessadores apresentam sinais de interrupção e é necessário um circuito externo, ao qual por meio de sinais de entrada de interrupção, o circuito possa enviar um evento externo que será tratado por meio das sub-rotinas de tratamento de interrupção. Pois dessa forma será possível receber e tratar as informações externas sem alterar a execução do programa principal do Arduino.

A requisição de interrupção pode ocorrer a qualquer momento de forma assíncrona, a qual o microprocessador irá perceber essa requisição por meio de uma “flag” e quando houver

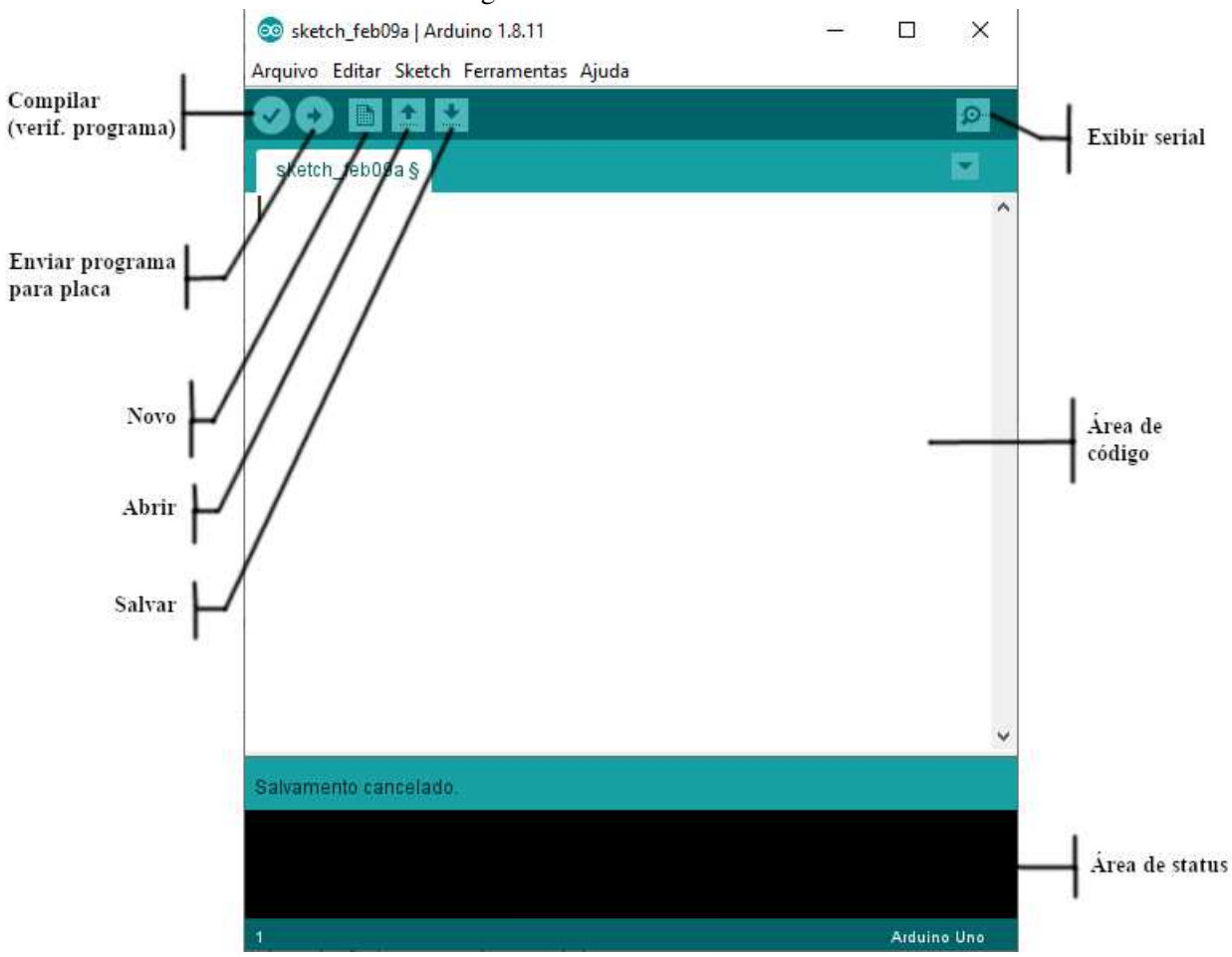
completado a execução relacionada a interrupção, por meio da “flag” será possível informar que foi tratada possibilitando assim novas instruções serem executadas (ANDRADE et al., 2005).

De acordo com (ATMEL, 2014), como método de segurança, para cada interrupção existe um vetor de registradores único, pois dessa forma é possível evitar a execução de uma instância da mesma interrupção quando outra estiver sendo executada. Outro fato importante é a existência de prioridade na execução de interrupções, ou seja, caso a prioridade de uma interrupção seja maior que outra, o seu endereço será escolhido para ser adicionado ao vetor de registros sendo executada primeiro.

2.3 ARDUINO IDE

Além da transferência de dados entre o Arduino e o RTC, é necessário entender sobre a *Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado (IDE) do Arduino. Segundo Schmidt (2015) o Arduino pode ser previamente programado através do software de desenvolvimento, um sistema simples que facilita na programação e é um dos motivos da grande utilização do mesmo. Schmidt (2015) diz que essa IDE consiste em um editor de texto simples, um compilador, um carregador e um monitor de porta serial. É possível visualizar na imagem abaixo a IDE e suas funcionalidades.

Figura 5 – Arduino IDE



Fonte: O autor

Desenvolver o código nessa IDE é simples, primeiro é necessário escrever o código baseado na linguagem C dentro da área de código e após finalizar é possível realizar a compilação para verificar se está tudo correto e pronto para execução, ou executar diretamente enviando para placa, caso não apresente erros o código será compilado e carregado diretamente na placa Arduino. Essa comunicação com a placa de desenvolvimento Arduino ocorre por um cabo USB.

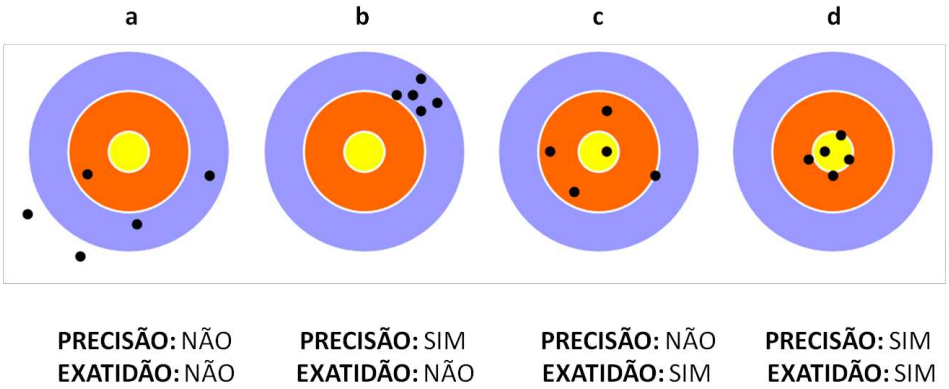
3 ANÁLISE DA EXATIDÃO E PRECISÃO DOS RELÓGIOS

A partir do entendimento e análise do que se está utilizando para os resultados das pesquisas, encontrou-se uma grande lacuna a ser estudada, trata-se da análise do tempo recebido pelas placas Arduíno e se elas são precisas e exatas.

Em metrologia, esse dois termos possuem significados muito diferentes, mas muito importantes para o processo de medição. O conceito de exatidão se dá em relação a proximidade do resultado em referência ao valor verdadeiro. Já a precisão, quando repetidas medidas apresentam resultados semelhantes, não necessariamente sendo próximo do valor verdadeiro.

De acordo com Thomsen (1997) a precisão de uma série de medições é uma medida da concordância entre determinações repetidas. Ela é usualmente quantificada como o desvio padrão de uma série de medidas e exatidão de uma medida (ou da média de um conjunto de medidas) é a distância estimada entre a medida e um valor "verdadeiro", "nominal", "tomado como referência", ou "aceito". Geralmente é expressa como um desvio ou desvio percentual de um valor conhecido. É possível visualizar esses dois conceitos na Figura 6 que representa quatro possíveis resultados em um teste de tiro.

Figura 6 – Diferença entre exatidão e precisão a partir de analogia com tiro ao alvo.



Fonte: (UNITYINSTRUMENTOS, 2015)

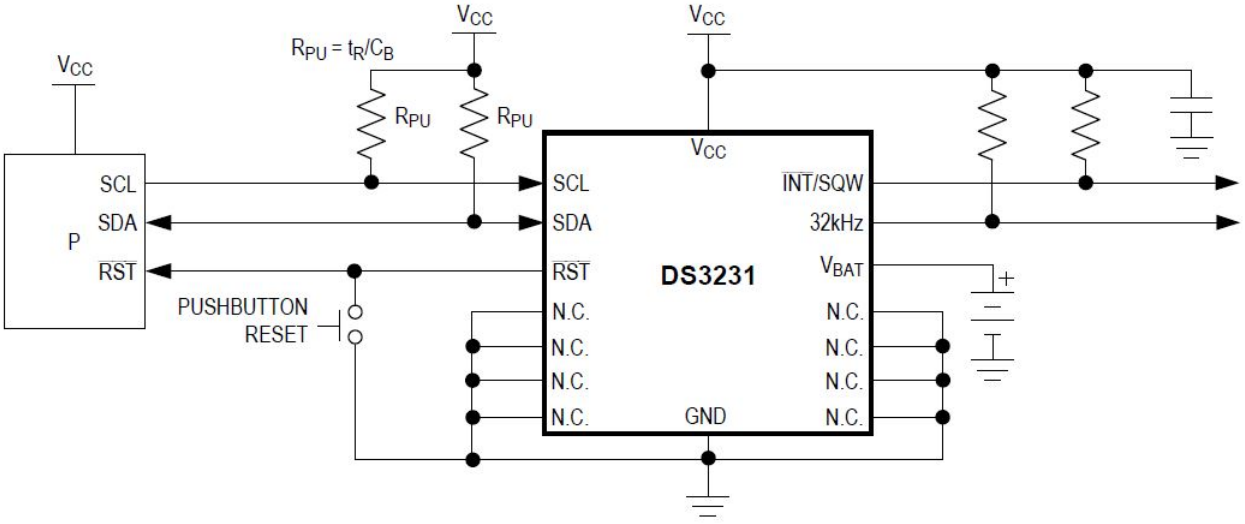
3.1 MÓDULO 3231

Os RTCs são pequenos dispositivos eletrônicos usados para incorporar dados de data e hora a dispositivos eletrônicos, estes utilizam cristais de quartzo e circuitos integrados em um chip para medir o tempo.

Assim, possuindo dois canais(pinos) de comunicação: Clock Serial (SCL) e Dados Seriais (SDA). O protocolo I2C sinaliza o início e fim da transmissão de dados por meio de mudanças no estado lógico dos pinos SCL e SDA. Em resumo, se o SCL estiver em nível alto e houver uma transição de alto para baixo no SDA, o DS3231 reconhece como o início de comunicação. Entretanto, se o SCL estiver em nível alto e houver uma transição de baixo para alto no SDA, o DS3231 reconhece como o fim da comunicação.

O módulo DS3231 é capaz de fornecer informações como segundo, minutos, dia, data, mês e ano. Meses com menos de 31 dias e anos bissextos são corrigidos automaticamente e pode operar tanto no formato 12 horas como 24 horas. Em sua placa vem embutido um sensor de temperatura para melhorar sua exatidão, e em caso de falha de energia o DS3231 automaticamente aciona a bateria que acompanha o módulo para evitar perda de dados, e o endereço e informações são transferidas via protocolo I2C. O Real Time Clock (RTC) DS3231 é um relógio de tempo real de alta precisão com Tensão de operação: 3,3-5V, consome menos de 500nA no modo bateria com oscilador em funcionamento, a faixa de temperatura: 0 a 40°C, dimensões: 38 x 22 x 14 mm. A Figura 7 representa o circuito básico de funcionamento do DS3231 (FERREIRA; AMARAL; GARDEZANI, 2016).

Figura 7 – Circuito básico do RTC DS3231.



Fonte: (MAXIMINTEGRATED, 2015)

De acordo com o *datasheet* (MAXIMINTEGRATED, 2015) o RTC possui uma precisão de mais ou menos 2 partes por milhão (ppm) para 0 até 40 graus, e para casos de -40 até 85 graus ele passa a ter uma precisão de 3,5ppm, ou seja, o tempo que será coletado a partir do RTC pode sofrer essa variação de acordo com a temperatura. Além disso ele possui uma saída programável de onda quadrada e uma interface simples com a maioria dos microcontroladores. Por isso o RTC passa a ser um bom modelo de precisão e exatidão para as análises que foram realizadas nesse trabalho.

Utilizando o módulo DS3231 como referência foi possível por meio de contagens de pulsos entre ele e o Arduino, avaliar a precisão e exatidão das funções internas de temporização dos microcontroladores.

4 METODOLOGIA

O RTC foi definido como o padrão para comparação dos resultados e para isso ao inicio dos experimentos foi disposto de três unidades do dispositivo da mesma marca e modelo, e entre eles foi necessário realizar uma medição de suas frequências. Utilizando o Arduino foi realizada uma contagem de interrupções que foram enviadas do RTC para o microcontrolador e realizou-se um comparativo entre os diferentes dispositivos. Após cinco análises consecutivas realizadas no mesmo dia com intervalos de na média 1 minuto, foi verificado que um dos DS3231 apresentaram um quantidade de interrupções muito distinto dos outros, decidindo assim eliminar este e escolhendo um dos outros dois para os experimentos pois eles apresentaram valores idênticos.

Após a escolha do RTC como o modelo para o testes, foram planejados os experimentos para tratar da comparação do relógio interno do microcontrolador e o tempo esperado levando em conta o mesmo. O primeiro experimento irá verificar a quantidade de interrupções contadas pelo Arduino UNO, ou seja, sabendo-se quantas interrupções são enviadas do RTC, é analisada quantas foram contadas pelo Arduino e realizada a comparação. Já o segundo experimento leva para outra perspectiva, o RTC dessa vez é utilizado como um meio de acionar uma interrupção em determinado intervalo de tempo, e o Arduino ficará responsável por internamente contar o tempo que levou para receber essa interrupção, após receber essa interrupção é comparado o tempo esperado pelo o que foi realmente contado por ele.

4.1 PRIMEIRO EXPERIMENTO

Como foi dito anteriormente o primeiro experimento analisou a quantidade de interrupções captadas pelo Arduino utilizando o RTC como padrão de comparação. Inicialmente foi necessário codificar na IDE do Arduino o código necessário para verificar a contagem de interrupções, para realizar essa análise foram necessárias duas interrupções sendo uma delas com o *attachInterrupt*[1] à qual foi passado o pino a ser utilizado nele e a função que seria executada ao receber uma mudança de estado. Essa função então irá possuir a contagem de pulsos recebidos

[1] Numa interrupção temos dois pontos chaves: a condição da interrupção e a função que será executada. Dessa forma, o comando attachInterrupt é usado para informar ao programa esses dados

pelo módulo DS3231. No código abaixo é possível visualizar essa codificação.

```
1 unsigned long int IRQcount;
2 int pin = 2;
3
4 void setup() {
5   Serial.begin (9600);
6   attachInterrupt(digitalPinToInterrupt(pin), IRQcounter, RISING);
7   ...
8 }
9
10 void IRQcounter() {
11   IRQcount++;
12 }
```

Código-fonte 1 – Código para verificar interrupção do RTC e realizar contagem de pulso

A segunda interrupção foi feita pelo TIMER1, que é um temporizador de 16 bits que permite a contagem de eventos, geração de sinal PWM, medida de pulsos, etc. Com ela foi possível configurar o Arduino para realizar a contagem de um segundo para poder visualizar a quantidade de pulsos por segundo recebidos pelo Arduino para isso foi necessário configurar o TIMER1 para operação normal, limpar os registradores e iniciar com valor para que o estouro ocorra em um segundo, ou seja, após esse tempo o TIMER1 irá verificar o estouro e acionar a interrupção para utilizar a função determinada que no caso do experimento é a visualização da quantidade de interrupções que pode ser visualizado a seguir.

```
1 unsigned long int IRQcount;
2 int countEvents = 0;
3
4 void setup() {
5   ...
6   TCCR1A = 0; // configura timer para os pinos OC1A e OC1B
      desconectados
7   TCCR1B = 0; // limpa registrador
8   TCCR1B |= (1<<CS10)|(1 << CS12);  // configura prescaler
9   // incia timer com valor para que estouro ocorra em 1 segundo
10  // 65536-(16MHz/1024/1Hz) = 49911 = 0xC2F7
```

```
11    TCNT1 = 0xC2F7;
12    TIMSK1 |= (1 << TOIE1);
13  }
14
15  ISR(TIMER1_OVF_vect){ // interrupcao do TIMER1
16    TCNT1 = 0xC2F7; // Renicia TIMER
17    cli(); //desativar interrupcoes
18    unsigned long int result = IRQcount;
19    countEvents++;
20    IRQcount = 0;
21    sei();//ativar interrupcoes
22    Serial.print(F("Evento: "));
23    Serial.print(countEvents);
24    Serial.print(F(" Pulsos Contados = "));
25    Serial.println(result);
26  }
```

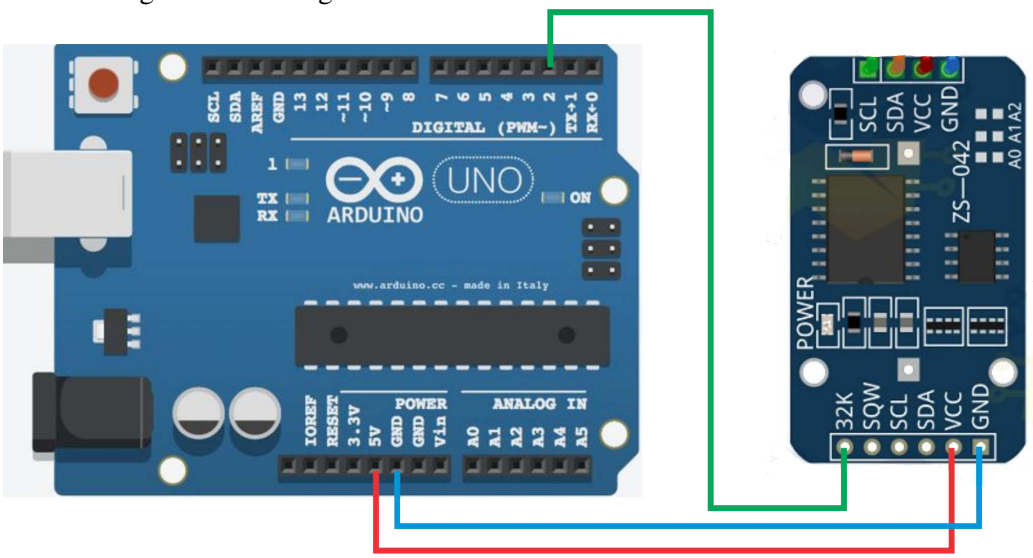
Código-fonte 2 – Configuração da interrupção do TIMER1 e função para leitura de pulsos contados

Para essa modelagem foi necessário estabelecer a comunicação do Arduino UNO com o RTC. Foi utilizada a saída de 32Khz (32.768 pulsos por segundo) do DS3231 em conexão com o pino 2 do Arduino UNO, alimentação de 5 Volts e o terra, dessa forma a modelagem foi feita para o RTC enviar os 32.768 pulsos para o Arduino em um segundo. A montagem do modelo para os testes pode ser visto na Figura 8.

Dessa forma foi possível com o DS3231 enviar os pulsos para o Arduino e com ele ser realizada a contagem de interrupções que foram recebidas, armazenando então na memória para ser visualizada futuramente. O experimentos foram feito em laboratório com uma temperatura em torno de 24 graus, foram feitos consecutivamente, sendo realizadas 4 baterias de testes com 4 contagens para cada Arduino com intervalo de na média 1 minuto entre eles, oferecendo assim uma média de pulsos para ser utilizada como comparativo para o valor referencial.

Após a finalização de cada bateria de teste, foi coletado a média do valores obtidos nas quatro contagens e armazenada numa planilha. Esse método foi utilizado para todos os 12 Arduinos testados, e após todas as baterias de testes foi possível traçar o comparativo deles com

Figura 8 – Montagem do Arduino UNO com o módulo RTC DS3231



Fonte: O autor

o RTC DS3231.

O tempo como uma grandeza física experimental deverá ser determinada a partir de medição e o resultado aproximado para o valor verdadeiro da grandeza, utilizando a teoria de erros, determinar o melhor valor possível para a grandeza e comparar com o valor lido do RTC, esperando assim que o valor do mensurando seja o mais próximo possível do valor de referência (VUOLO, 1992). No final foi possível obter uma avaliação da exatidão e precisão dos relógios microcontroladores utilizando como base o RTC previamente testado.

4.2 SEGUNDO EXPERIMENTO

Para o segundo experimento foi utilizada a mesma forma de bateria de testes, realizando 4 baterias de testes para cada Arduino, o testes foram feitos em uma sala fechada com temperatura aproximada de 24 graus, o experimentos foram realizados no mesmo dia com intervalos de na média 1 minuto entre eles.

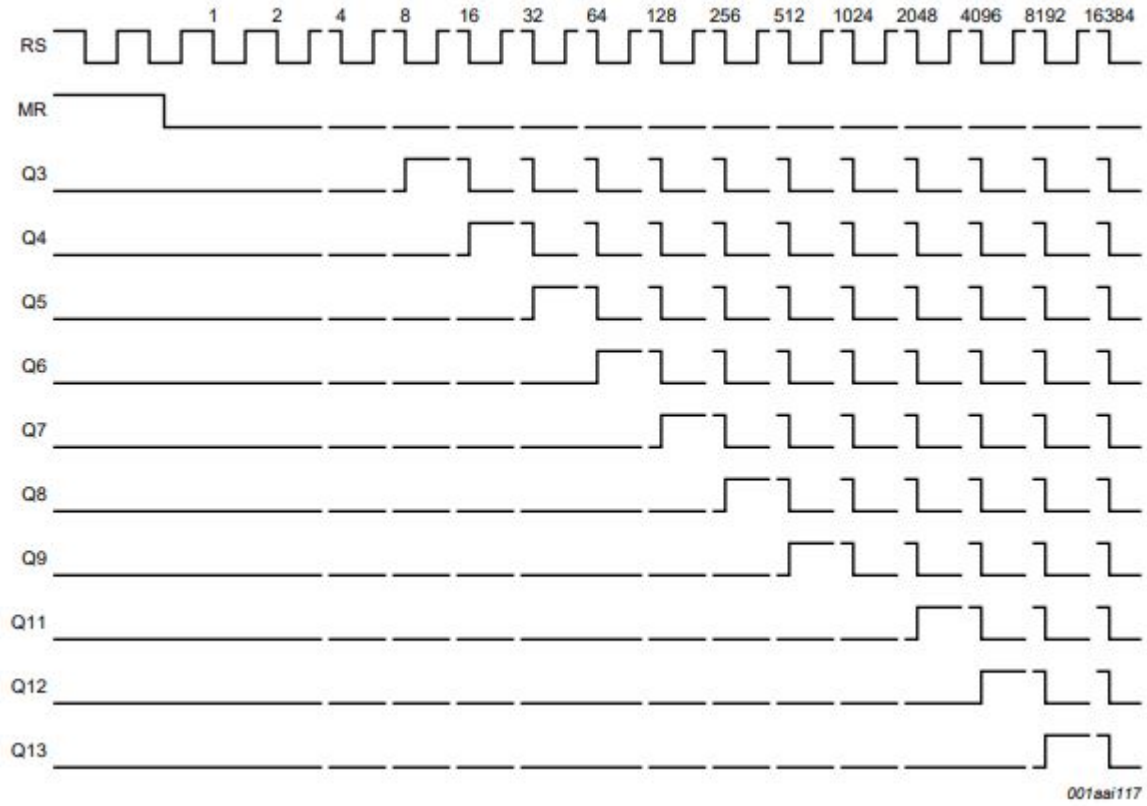
Neste experimento foi definido ao invés de realizar uma contagem de pulsos pelo Arduino, ser feito uma contagem do tempo. Ou seja, o RTC deverá enviar uma interrupção a cada período de tempo para o Arduino possibilitando assim uma comparação entre eles. Para realizar a contagem do tempo a partir do RTC, foi necessário utilizar dois contadores binários do modelo 74HC4060, pois com eles foi possível receber uma interrupção do Arduino com o tempo estabelecido entre os contadores e o RTC.

O contador binário trabalha com saídas representas por "Qn", sendo o "n"parte do cálculo (2^{n+1}), que informa número de "descidas"necessárias para um pulso, ou seja, se utilizarmos a saída "Q3", serão necessárias 16 descidas para completar um pulso, e assim em diante.

Foi decidido para comparação uma contagem de 16 segundos para cada interrupção pois estabelece uma larga escala comparativa para a comparação dos resultados. Para conseguir esse tempo foi necessário conectar o RTC ao primeiro contador binário pela entrada do *clock input* pois assim o contador receberá todos os pulsos. A saída do primeiro foi feita pela saída Q13 que vai dividir o pulso em 16.384, ou seja, serão necessários 16.384 pulsos para o primeiro contador emitir uma saída, dessa forma, como o relógio de tempo real trabalha com 32.768 pulsos por segundo, o contador irá transpor para dois pulsos por segundo.

Com a configuração do primeiro contador realizada, ele então foi conectado a entrada *clock input* do segundo contador, que foi pela Q4, que irá enviar uma interrupção para o Arduino a cada 32 pulsos, ou seja, como o primeiro contador vai está enviando dois pulsos por segundo, serão necessários 16 segundos para ele poder alcançar a quantidade necessária de pulsos para enviar ao Arduino, na Figura 9 é possível visualizar como funciona o diagrama de tempo do contador binário e como é feita essa quebra de pulsos.

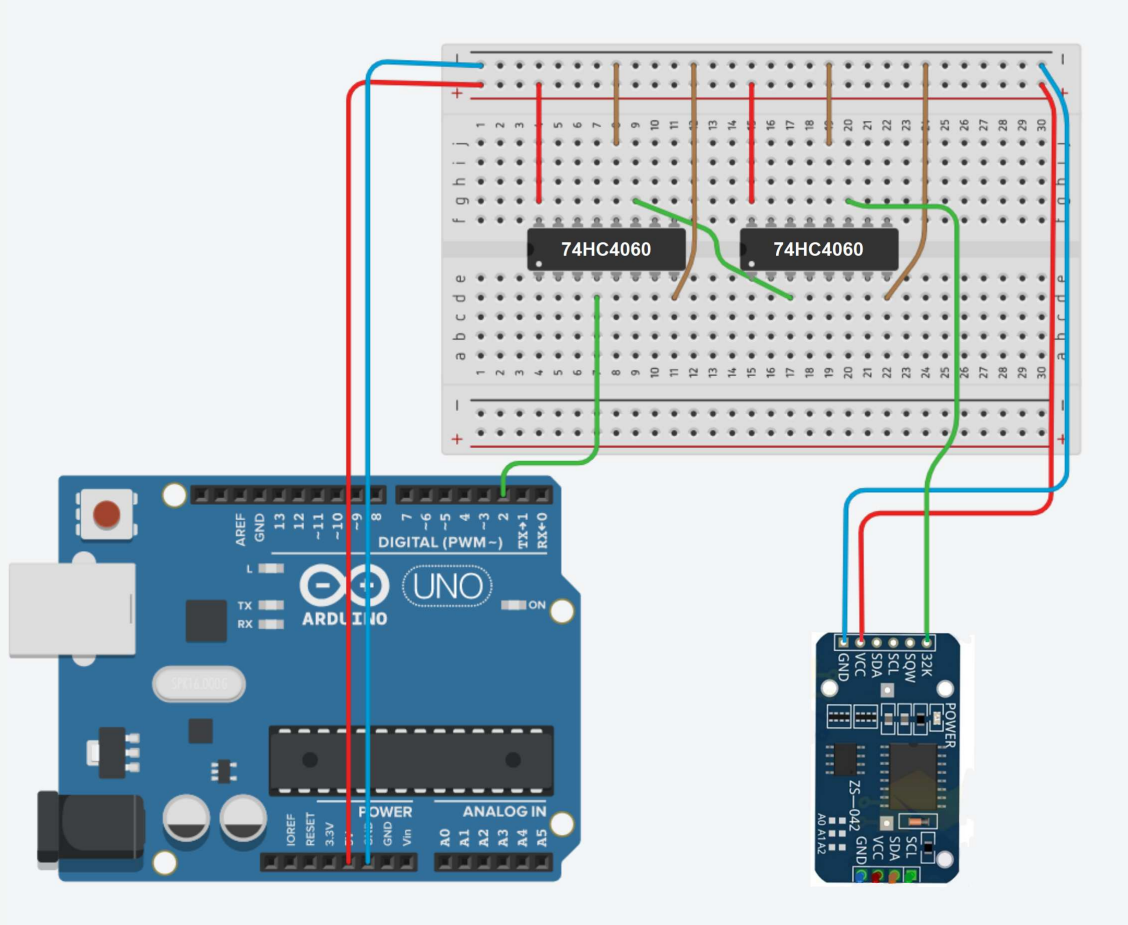
Figura 9 – Diagrama de tempo do contador binário 74HC4060



Fonte: (NEXPERIA, 2013)

Após toda configuração realizada para dividir os pulsos do RTC, foi facilitada a consulta de tempo do Arduino, pois foi possível identificar que para cada pulso recebido, com o relógio de tempo real funcionando corretamente, ele deverá levar 16 segundos para enviar, e o Arduino deverá ter contado esse mesmo tempo para o recebimento da interrupção. Dessa forma basta coletar a interrupção do segundo contador pelo pino 2 do Arduino UNO para receber o pulso a cada 16 segundos e realizar os devidos testes, na Figura 10 é possível visualizar como foi realizada a montagem do Arduino em conjunto com os contadores binários.

Figura 10 – Montagem do Arduino UNO com o módulo RTC DS3231 utilizando contadores



Fonte: O autor

Com a configuração da interrupção com Arduino realizada, é necessária realizar o recebimento da interrupção dentro do Arduino com o *attachInterrupt* o qual foi passado o PINO e a função a ser executada a cada interrupção. Essa função chamada por ele será encarregada apenas de alterar uma variável booleana, pois dessa forma não irá alterar em outras interrupções que possam ocorrer.

Dentro do *loop* do Arduino, é verificada a variável de controle, e caso seja válida é examinada a contagem de tempo do Arduino para ter recebido aquela interrupção por meio da função *micros*[2] e então é salva na Memória Programável Somente de Leitura Apagável Eletricamente (EEPROM). No código-fonte abaixo é possível verificar a função *loop* e a função para armazenamento do tempo.

```
1 . . .
```

[2] Função *micros* retorna o número de microssegundos passados desde que a placa Arduino começou a executar o programa atual.

```
2 unsigned long time_now = 0; // Tempo atual subtraido pelo anterior
3 unsigned long time_prev = micros(); // Tempo anterior
4
5 void armazenarTempo(unsigned long valor) {
6   byte b[4];
7   b[0]= (int) valor;
8   b[1]= (int) (valor>>8);
9   b[2]= (int) (valor>>16);
10  b[3]= (int) (valor>>24);
11  for (int i=0 ; i< 4; i++) {
12    EEPROM.write(endereco, b[i]);
13    endereco++;
14  }
15 }
16
17 void loop() {
18  if (flagIRQ) {
19      time_now = micros() - time_prev;
20      time_prev = micros();
21      armazenarTempo(time_now);
22      numContagens++;
23      flagIRQ = false;
24  }
25  ...
26 }
```

Código-fonte 3 – Parte do código para armazenar contagens na EEPROM

Após quatro interrupções o número de contagens limite é alcançado e chamada a função para ler os tempos armazenados na EEPROM, podendo assim realizar a comparação com o tempo que deveria ser coletado. No código-fonte abaixo é possível verificar a função para visualizar as contagens de tempo.

```
1 void lerTempo(){
2   byte b[4];
3   unsigned long valorMedia = 0;
4   for (int i=0; i<5; i++) {
5     unsigned long valor = 0;
```

```
6       for (int j=1; j<5; j++) {
7         b[j-1] = EEPROM.read((i*4)+j);
8       }
9       valor += (unsigned long)b[3] << 24;
10      valor += (unsigned long)b[2] << 16;
11      valor += (unsigned long)b[1] << 8;
12      valor += (unsigned long)b[0];
13      Serial.print("EEPROM LEITURA CONTAGEM ");
14      Serial.print(i+1);
15      Serial.println(":");
16      Serial.println(valor);
17      if (i != 0) valorMedia += valor;
18    }
19    Serial.println("_____________________________");
20    Serial.println("MEDIA DE VALORES DA LEITURA: ");
21    Serial.println(valorMedia/4);
22  }
23
24  void loop() {
25    ...
26    if (numContagens == NUMEROLIMITECONTAGENS && print_eeprom) {
27       print_eeprom = false;
28       lerTempo();
29    }
30  }
```

Código-fonte 4 – Função para visualização aa contagem armazenada na EEPROM

A cada bateria de teste os resultados foram salvos em uma planilha e esse método foi replicado para todos experimentos seguintes, após essa análise ser realizada para os 12 Arduinos foi então possível comparar o tempo esperado e o que foi realmente coletado de cada dispositivo a partir da função *micros*.

5 RESULTADOS

Neste capitulo será possível visualizar os resultados obtidos utilizando os diferentes experimentos apresentados anteriormente e uma discussão sobre os mesmos.

5.1 RESULTADOS DO PRIMEIRO EXPERIMENTO

Na tabela 2 apresentamos os resultados encontrados para cada Arduino.

Tabela 2 – Tabela de resultados em laboratório, 26/08/2019

Arduino 1	
Teste	**Pulsos/Seg**
1	32799
2	32798
3	32798
4	32798
Média	**32798**

Arduino 2	
Teste	**Pulsos/Seg**
1	32784
2	32784
3	32783
4	32784
Média	**32784**

Arduino 3	
Teste	**Pulsos/Seg**
1	32779
2	32778
3	32777
4	32779
Média	**32779**

Arduino 4	
Teste	**Pulsos/Seg**
1	32801
2	32800
3	32801
4	32800
Média	**32801**

Arduino 5	
Teste	**Pulsos/Seg**
1	32720
2	32720
3	32719
4	32719
Média	**32720**

Arduino 6	
Teste	**Pulsos/Seg**
1	32784
2	32784
3	32783
4	32783
Média	**32784**

Arduino 7	
Teste	**Pulsos/Seg**
1	32796
2	32796
3	32795
4	32795
Média	**32796**

Arduino 8	
Teste	**Pulsos/Seg**
1	32780
2	32779
3	32780
4	32779
Média	**32780**

Arduino 9	
Teste	**Pulsos/Seg**
1	32809
2	32807
3	32808
4	32807
Média	**32808**

Arduino 10	
Teste	**Pulsos/Seg**
1	32809
2	32809
3	32809
4	32809
Média	**32809**

Arduino 11	
Teste	**Pulsos/Seg**
1	32920
2	32919
3	32920
4	32919
Média	**32920**

Arduino 12	
Teste	**Pulsos/Seg**
1	32790
2	32789
3	32788
4	32789
Média	**32789**

Fonte: O autor

Após coletar os dados relacionados com a contagem de pulsos, calculou-se a média de valores coletados anteriormente do Arduíno (32.784 pulsos por segundo) e com ele foi possível calcular a variação percentual da média dos valores como mostrado na Tabela 3. O próximo passo foi verificar a exatidão dos resultados coletados, precisando assim de um valor referencial como base para a variação percentual. Para o experimento o valor referencial utilizado foi a quantidade de pulsos que o RTC envia para o Arduino (32.768 pulsos por segundo). O resultado é apresentado na Tabela 4.

Tabela 3 – Tabela de resultados em laboratório, 26/08/2019

VARIAÇÃO PERCENTUAL COM A MÉDIA DOS VALORES (32.793)	
Arduino	**Variação**
1	0,017%
2	-0,026%
3	0,041%
4	0,026%
5	-0,222%
6	-0,026%
7	0,011%
8	-0,038%
9	0,047%
10	0,050%
11	0,389%
12	-0,011%

Fonte: O autor

Tabela 4 – Tabela de resultados em laboratório, 26/08/2019

VARIAÇÃO PERCENTUAL COM VALOR REFERENCIAL (32.768)	
Arduino	**Variação**
1	0,092%
2	0,049%
3	0,034%
4	0,101%
5	-0,146%
6	0,049%
7	0,085%
8	0,037%
9	0,122%
10	0,125%
11	0,464%
12	0,064%

Fonte: O autor

5.2 RESULTADOS DO SEGUNDO EXPERIMENTO

Para o segundo experimento o RTC foi configurado para enviar uma interrupção a cada 16 segundos. Verificando então pelo Arduino qual foi a média de tempos coletados para o recebimento das interrupções, a um todo de quatro tempos coletados, e o comparativo com o esperado. Realizando então uma variação percentual com o valor referencial. Na Tabela 5 é apresentado todos os tempos coletados e a média de tempo por Arduino, sendo eles os mesmos Arduinos utilizados no experimento anterior.

Tabela 5 – Tabela de resultados em laboratório, 13/11/2019

Arduino 1	
Teste	**Segundos**
1	15975578
2	15975638
3	15975685
4	15975723
Média	**15975662**

Arduino 2	
Teste	**Segundos**
1	15982946
2	15982943
3	15982955
4	15982909
Média	**15982945**

Arduino 3	
Teste	**Segundos**
1	15984043
2	15985885
3	15985811
4	15985773
Média	**15985792**

Arduino 4	
Teste	**Segundos**
1	15974802
2	15974840
3	15974738
4	15974431
Média	**15974770**

Arduino 5	
Teste	**Segundos**
1	16014453
2	16014397
3	16014372
4	16014325
Média	**16014385**

Arduino 6	
Teste	**Segundos**
1	15983689
2	15983284
3	15983504
4	15983489
Média	**15983497**

Arduino 7	
Teste	**Segundos**
1	15977615
2	15978244
3	15977413
4	15977382
Média	**15977514**

Arduino 8	
Teste	**Segundos**
1	15985952
2	15985514
3	15985635
4	15985346
Média	**15985575**

Arduino 9	
Teste	**Segundos**
1	15971744
2	15971687
3	15971609
4	15971534
Média	**15971648**

Arduino 10	
Teste	**Segundos**
1	15971255
2	15971234
3	15970913
4	15970824
Média	**15971074**

Arduino 11	
Teste	**Segundos**
1	15917414
2	15917515
3	15917252
4	15917233
Média	**15917333**

Arduino 12	
Teste	**Segundos**
1	15980789
2	15980609
3	15980582
4	15980606
Média	**15980608**

Fonte: O autor

Com o teste dos 12 Arduinos, foi coletado uma média de todos valores sendo ele

igual a 15.979.061 e utilizá-lo como referencia para calcular a variação percentual entre os valores coletados por cada Arduino e a média visualizados na Tabela 6. No entanto na Tabela 7 foi utilizado o valor referência igual a 16.000.000 o equivalente aos 16 segundos da interrupção do RTC.

Tabela 6 – Tabela de resultados em laboratório, 13/11/2019

VARIAÇÃO PERCENTUAL COM A MÉDIA DOS VALORES (15.979.061)	
Arduino	**Variação**
1	-0,021%
2	0,024%
3	0,042%
4	-0,027%
5	0,221%
6	0,028%
7	-0,010%
8	0,041%
9	-0,046%
10	-0,050%
11	-0,388%
12	0,010%

Fonte: O autor

Tabela 7 – Tabela de resultados em laboratório, 13/11/2019

VARIAÇÃO PERCENTUAL COM VALOR REFERENCIAL (16.000.000)	
Arduino	**Variação**
1	-0,152%
2	-0,107%
3	-0,089%
4	-0,158%
5	0,090%
6	-0,103%
7	-0,141%
8	-0,090%
9	-0,177%
10	-0,181%
11	-0,519%
12	-0,121%

Fonte: O autor

5.3 COMPARATIVO ENTRE OS EXPERIMENTOS

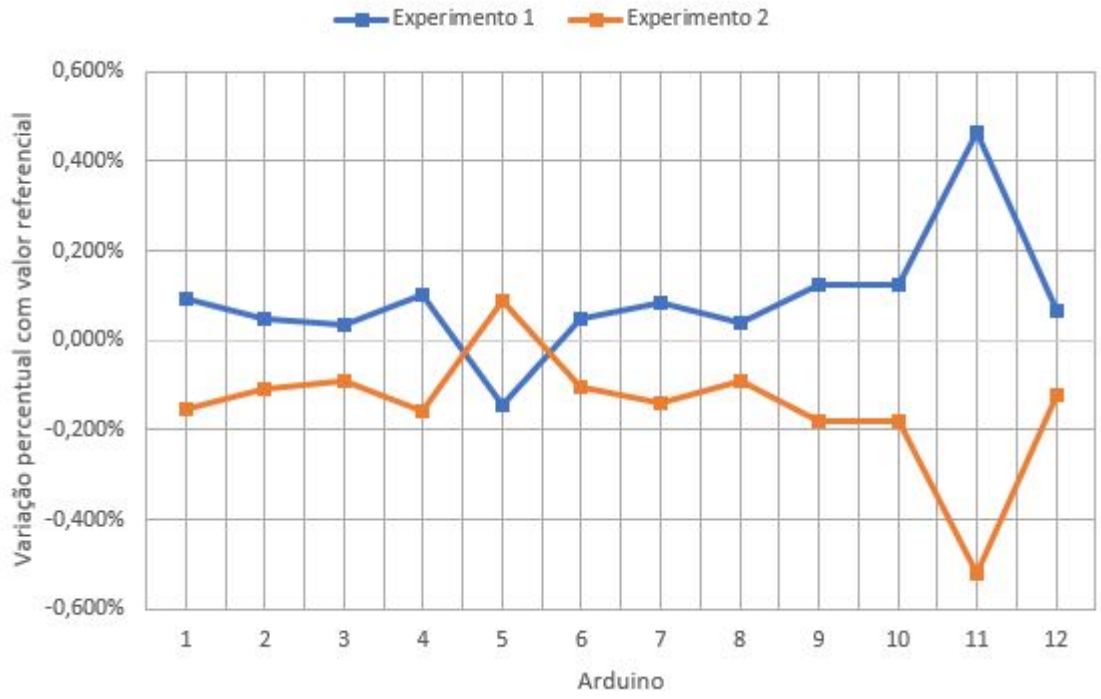
Com os resultados obtidos dos dois experimentos realizados, foi traçado um comparativo entre eles verificando assim a relação entre os mesmos. Validando que quanto menor a

quantidade de pulsos contada pelo Arduino no primeiro experimento, maior o tempo calculado por ele para receber a interrupção no segundo experimento. Por exemplo, o Arduino 5 contou uma média de 32720 pulsos por segundo, sendo ele menor que o referencial esperado de 32768, já no segundo experimento, esse mesmo Arduino contou uma média de 16.014.385 segundos para receber uma interrupção. Ao outro extremo, temos o Arduino 11 que teve a maior média de pulsos contados no primeiro experimento, sendo ele de 32920 pulsos e no segundo experimento o mesmo contou uma média de 15.917.333 microssegundos.

Foi verificado que os valores obtidos no primeiro e segundo experimento são inversamente proporcionais, ou seja, se um Arduino teve a contagem de pulsos maior que o referencial, quer dizer que o relógio interno do Arduino foi mais lento pois ele levou mais tempo para a contagem de pulsos contando mais do que realmente deveria ter recebido. Já se a contagem for menor que o esperado, o relógio interno do Arduino vai ser mais rápido.

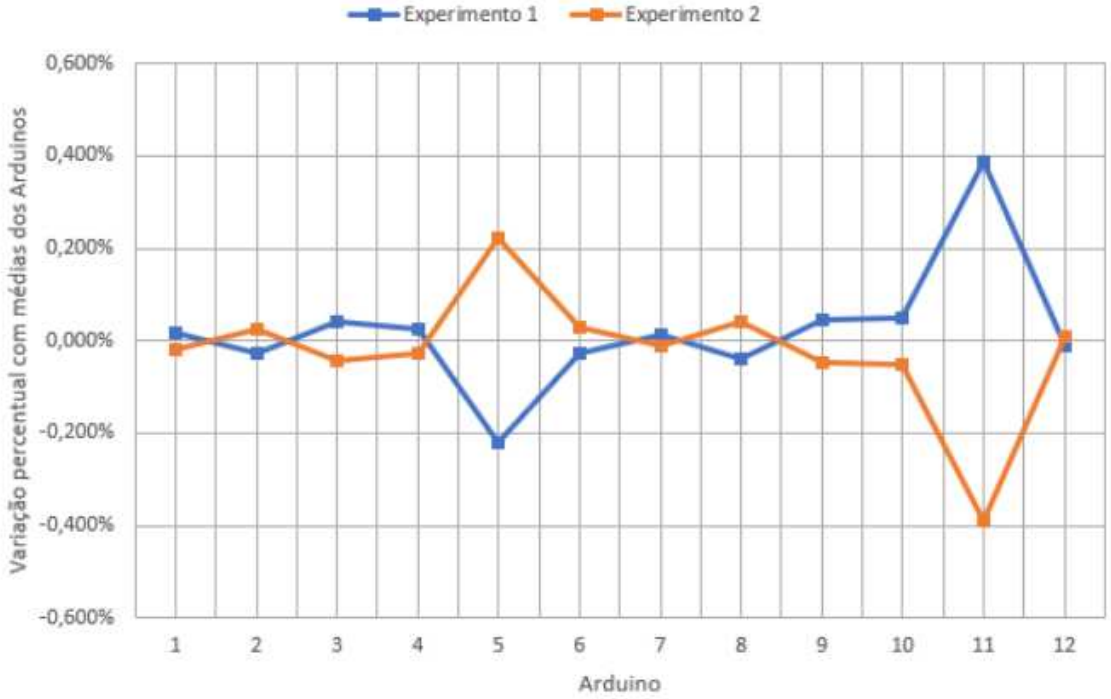
Como foram realizados dois comparativos em cada experimento, um realizando a variação percentual em relação ao referencial e outro ao valor médio coletado dos Arduinos, foram traçados dois gráficos que mostram essa relação entre os experimentos.

Figura 11 – Comparativo entre os experimentos utilizando valor referencial



Fonte: O autor

Figura 12 – Comparativo entre os experimentos utilizando média dos Arduinos



Fonte: O autor

A partir dos dois experimentos realizados foi analisado que os dois apresentam resultados diferentes e dessa forma é possível realizar um comparativo entre eles, para compreender a diferença em percentual entre seus valores, verificando assim a diferença que o método de codificação e experimento pode trazer nos resultados obtidos pelo microcontroladores.

Com isso foi necessário decidir uma métrica comparativa entre eles. O primeiro experimento possui a contagem de pulsos em 1 segundo, logo se a quantidade de pulsos contados pelo Arduino foi maior que o referencial, quer dizer então que o relógio interno dele foi mais lento que o esperado. Sabendo disso foi possível realizar um cálculo de quantos segundos o Arduino ficou mais lento ou mais rápido que o normal. Esse comparativo utilizou o resultados do experimentos comparados com o referencial e não com a média pois foi necessário possuir uma base constante para verificar a diferença entre eles.

A partir do primeiro experimento é possível com a variação percentual calcular o tempo do Arduino, pela formula abaixo:

$$t = \frac{16.000.000}{((v/100)+1)}. \tag{5.1}$$

Sendo V a variação percentual do experimento, dividindo por 100 e somando 1 é possível verificar o tempo que esse Arduino levou para a contagem de pulsos. Após verificar esse tempo é dividido

os 16 segundos (referência do segundo experimento) pelo tempo do Arduino conseguindo assim o tempo esperado do primeiro experimento com parâmetros do segundo. Após verificar esse tempo esperado é possível realizar então uma diferença percentual entre os segundos coletados do segundo experimento e o que seria esperado calcular no primeiro. Na tabela abaixo é possível verificar esse comparativo.

Tabela 8 – Tabela comparativa entre os dois experimentos

VARIAÇÃO PERCENTUAL ENTRE OS EXPERIMENTOS		
Arduino	**Tempo para o primeiro experimento**	**Variação**
1	15.985.294	0,060%
2	15.992.164	0,058%
3	15.994.562	0,055%
4	15.983.856	0,057%
5	16.023.394	0,056%
6	15.992.164	0,054%
7	15.986.412	0,056%
8	15.994.082	0,053%
9	15.980.504	0,055%
10	15.980.025	0,056%
11	15.926.103	0,055%
12	15.989.767	0,057%

Fonte: O autor

Com a tabela apresentada informando a variação percentual entre os experimentos, mesmo os resultados apresentando uma variação pequena é possível verificar que possuem valores aproximados entre todos os Arduinos, mostrando que essa variação ocorreu em todos apresentadas pelo mesmo motivo, a forma como foi codificado e desenvolvido o experimento.

5.4 VARIAÇÃO APLICADA A EXPERIMENTO DE MEDIAÇÃO DA ACELERAÇÃO GRAVITACIONAL

Após encontrar a variação percentual presente nas placas microcontroladoras por meio de experimentos e demostrar que seu valor entre dois experimentos a partir de codificação foi pequeno e constante, realizou-se o próximo passo da pesquisa, encontrar artigos experimentais didáticos de física que possuam o tempo como grande relevante para o resultado, levando em conta as formulas e aplicações expostas, visando então aplicar as variações encontradas neste trabalho.

Para confirmar a influência da variação, utilizou-se como metodologia, 3 variações presentes nos Arduinos no primeiro experimento. Assim sendo, foram utilizados o melhor caso (menor variação), pior caso (maior variação) e caso médio (média das variações encontradas). Dessa forma verificou-se a influência causada nas três hipóteses e a variação percentual originada no experimento.

O experimento escolhido para essa análise se propõe a medir a aceleração da gravidade com a placa Arduino em um experiência simples de queda livre. Segundo Cordova (2016) medir a aceleração gravitacional g é deixar cair verticalmente um corpo a partir de uma altura predeterminada h e medir a duração do seu tempo de queda, a formula pode ser vista a seguir:

$$g = \frac{2h}{t^2}, \tag{5.2}$$

em que t é a duração da queda. Apesar de ser um fundamento simples e uma das maneiras mais fáceis de medir g, a velocidade com que a queda livre ocorre pode tornar o experimento frustrante para novatos e por isso a inserção do Arduino para realizar a verificação da duração da queda facilita o experimento.

Ao verificarmos como o tempo tem relevância dentro do experimento e como ele utiliza o Arduino para realizar a contagem do mesmo, é possível então aplicar a variação percentual e comparar os resultados. Para essa análise foi utilizado um dos resultados encontrados pelo autor, onde h = 0,943m com δh = 0,001m e t = 0,439s com δt = 0,001s . Após a coleta dos dados é possível realizar o cálculo a partir da fórmula:

$$g = \frac{g_{máx} + g_{mín}}{2}, \tag{5.3}$$

onde

$$g_{máx} = \frac{2(h+\delta h)}{(t-\delta t)^2}; \tag{5.4}$$

e

$$g_{mín} = \frac{2(h-\delta h)}{(t+\delta t)^2}. \tag{5.5}$$

Após a análise das fórmulas e dos valores que seriam utilizados para a aplicação, foi necessário definir as variações percentuais que seriam utilizadas para aplicação sendo elas: 0,034% (melhor caso), 0,464% (pior caso) e 0,075% (média das variações). Possuindo então as

variações foi possível aplicar no tempo que foi utilizado para obter os resultados do experimento a partir da seguinte fórmula:

$$t_{var} = t * ((\frac{v}{100}) + 1), \tag{5.6}$$

onde v será a variação percentual e t_{var} o tempo com variação que será utilizado para realizar os novos cálculos. A partir da tabela a baixo é possível verificar os resultados tanto do experimento do autor (sem variação), quanto os resultados aplicando as variações encontradas nessa pesquisa.

Tabela 9 – Tabela com resultados da aplicação da variação

RESULTADOS	
Variação Percentual	***g***
Sem variação	9,78637
Melhor Caso	9,77972
Pior Caso	9,69618
Caso Médio	9,77180

Fonte: O autor

Denota-se que o pior caso apresentou uma variação percentual de 0,93% em relação ao resultado do autor, ou seja, dependendo do grau de exatidão que queira ser alcançado na pesquisa, essa variação se torna relevante para o resultado. Ressaltamos que a variação atribuída ao pior caso resultou do Arduino 11, que não faz parte do mesmo lote que os demais utilizados neste trabalho, o caso médio que apresenta a média de todas variações encontradas nessa pesquisa apresentou uma variação percentual de 0,15%, em que, mesmo menor que o pior caso continua apresentando relevância a depender da aplicação.

6 CONSIDERAÇÕES FINAIS

O trabalho teve como objetivo geral verificar a exatidão e precisão dos relógios internos de placas microcontroladoras acopladas ao Arduino UNO. Essa avaliação foi realizada por meio de dois experimentos efetuados com diferentes codificações e utilizando os mesmos dispositivos. As duas análises apresentaram uma variação muito próxima com uma média de 0,056%, ou seja, não só a variação encontrada nos resultados dos dois experimentos foi pequena como também não foram muito distantes, sendo algo positivo aos resultados, indicando que os resultados foram consistentes.

As variações encontradas nos experimentos foram menores do que as esperadas. Porém mesmo uma variação pequena pode trazer um impacto expressivo para determinados experimentos didáticos ou até mesmo projetos em larga escala. Se verificarmos, por exemplo, o Arduino 11 no primeiro experimento, o mesmo teve uma variação de 0,464% em relação ao valor referencial e por meio da aplicação dele em um experimento existente, foi possível visualizar um variação considerável do valor referencial, salientando assim que as variações encontradas, dependendo do experimento e do grau de exatidão requerido, se tornam relevantes.

Visando melhorar e alcançar mais resultados foram traçados trabalhos futuros importantes, sendo eles, trabalhar a variação de temperatura a fundo, utilizando medidor de temperatura e levando os dispositivos para temperaturas altas e baixas e visualizar como essa variável pode afetar no cristal oscilador do microcontrolador e a variação que pode ocorrer por conta disso. Outro aspecto importante também é a análise de outros modelos de Arduino utilizados atualmente em experimentos e verificar suas variações em comparação com as encontradas nesse trabalho.

REFERÊNCIAS

ANDRADE, M. T. C. de et al. *IMPLEMENTAÇÃO DE UM CRONÔMETRO COM INTERRUPÇÃO*. [S.l.], 2005.

ATMEL. 2014. Disponível em: <https://www.waveshare.com/w/upload/9/93/ATmega328P_datasheet_Complete.pdf>.

BORGES, E. S. *Carro robô em Arduino com comunicação sem fio*. [S.l.], 2011.

CAVALCANTE, M. A.; TAVOLARO, C. R. C.; MOLISANI, E. Física com arduino para iniciantes. *Rev. Bras. Ensino Fís.*, v. 33, n. 4, 2011.

CORDOVA, A. C. T. H. Medida de g com a placa arduino em um experimento simples de queda livre. *Rev. Bras. Ensino Fís.*, v. 38, n. 2, 2016.

ELETROGATE. 2020. Disponível em: <https://www.eletrogate.com/uno-r3-cabo-usb-para-arduino>.

FERREIRA, G. F. d. A. F.; AMARAL, L. H.; GARDEZANI, R. Acionamento de circuitos elétricos via web utilizando arduino. *Revista Caribeña de Ciencias Sociales*, 2016.

GADRE, D. V. *Programming And Customizing the AVR Microcontroller*. 1nd. ed. [S.l.]: McGraw-Hill, 2001.

JUNIO, J. F. N.; BORGES, V. E. S.; NASCIMENTO, R. M. M. F. Descrição temporal de forças de colisão: um modelo didático para laboratório de física assistido por sistema embarcado. *Rev. Bras. Ensino Fís.*, v. 41, n. 3, 2018.

MAXIMINTEGRATED. 2015. Disponível em: <https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>.

MCROBERTS, M. Arduino básico. 2011.

NERI, H. G. F. *Utilização da plataforma arduino para controle de experimentos remotos de física*. [S.l.], 2014.

NEXPERIA. 2013. Disponível em: <https://assets.nexperia.com/documents/data-sheet/74HC_HCT4060_Q100.pdf>.

SCHMIDT, M. *Arduino: A Quick-Start Guide*. 2nd. ed. [S.l.]: Pragmatic Bookshelf, 2015.

THOMSEN, V. Precision and the terminology of measurement. *The Physics Teacher*, v. 35, p. 15–17, 1997.

UNITYINSTRUMENTOS. *Você conhece a diferença entre Precisão e Exatidão?* 2015. Disponível em: <https://unityinstrumentos.com.br/voce-conhece-a-diferenca-entre-precisao-e-exatidao/>.

VUOLO, J. H. *Fundamentos da Teoria de Erros*. 2nd. ed. São Paulo, SP, Brazil: Editora Edgard Blucher, 1992.