Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica

Automação de um sistema de medição de coeficiente piezoelétrico

Igor Nazareno Soares

São Carlos
2015

Igor Nazareno Soares

Automação de um sistema de medição de coeficiente piezoelétrico

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia da Computação

Orientador: Prof. Dr. Ruy Alberto Corrêa Altafim

São Carlos
2015

Soares, Igor Nazareno
S676a Automação de um sistema de medição de coeficiente piezoelétrico / Igor Nazareno Soares; orientador Ruy Alberto Corrêa Altafim. São Carlos, 2015.

Monografia (Graduação em Engenharia de Computação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2015.

1. Sistema de medida. 2. Piezoeletricidade. 3. Eletreto. 4. Automação. 5. Arduino. 6. Qt. I. Título.

FOLHA DE APROVAÇÃO

Nome: Igor Nazareno Soares

Título: "Automação de um sensor de piezoeletricidade"

Trabalho de Conclusão de Curso defendido em 23/06/2015.

Comissão Julgadora:	**Resultado:**
Prof. Titular Ruy Alberto Corrêa Altafim (Orientador) - SEL/EESC/USP	Aprovado
Mestre Yuri Andrey Olivato Assagra Doutorando - SEL/EESC/USP	Aprovado
Mestre Daniel Augusto Pagi Ferreira Doutorando - SEL/EESC/USP	Aprovado

Coordenador do Curso Interunidades - Engenharia de Computação:

Prof. Associado Evandro Luís Linhari Rodrigues

Dedicado a todos aqueles que, direta ou indiretamente, se beneficiem dos frutos deste trabalho, um pequeno tijolo nos alicerces da Grande Obra

Agradecimentos

A Deus, pela sua Luz, Vida e Amor.

A minha família, pelo constante apoio, amor e carinho que incondicionalmente recebo: minha Mãe, presença constante, doce e amorosa em minha vida, sempre a me guiar, grande Companheira; meu Pai, grande Mestre, Guia e Companheiro; e meu Irmão, eterno Companheiro de Jornada.

Ao Prof. Dr. Ruy Alberto Corrêa Altafim, pelo apoio, incentivo e orientação na concretização deste trabalho.

Ao mestre eng. Yuri Andrey Olivato Assagra e ao eng. Felipe de Sousa, pela atenção, suporte e apoio durante este trabalho.

À Universidade de São Paulo (Escola de Engenharia de São Carlos - Departamento de Engenharia Elétrica), pela oportunidade de realização deste trabalho.

"O Conhecimento é o reflexo mais sublime na escuridão do Paraíso Terrestre"

Lourivaldo Alcides Soares, O Lendário

Resumo

SOARES, I. N. Automação de um sistema de medição de coeficiente piezoelétrico. 2015. Trabalho de Conclusão de Curso - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2015.

Este trabalho consiste na automação de um sensor de piezoeletricidade em piezoeletretos utilizando um microcontrolador Arduino interfaceado por um computador pessoal. O sistema de medição, composto por um eletrômetro responsável pela medida da quantidade de carga elétrica presente no eletreto e um sistema pneumático responsável por sensibilizar mecanicamente a amostra, tem o seu controle realizado através de uma aplicação com interface gráfica que por sua vez é responsável pela comunicação com o microcontrolador, que controla o sistema pneumático, assim como a coleta e processamento de dados obtidos pelo eletrômetro.

Palavras-Chave: Sistema de Medida, piezoeletricidade, eletreto, automação, arduino, Qt

Abstract

SOARES, I. N. Automação de um sistema de medição de coeficiente piezoelétrico. 2015. Trabalho de Conclusão de Curso - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2015.

This work consists of an automation of a piezoelectret's piezoelectricity sensor using an Arduino microcontroller with a PC interface. The measurement system, consisting of an electrometer responsible for measuring the amount of electric charge present in the electret and a pneumatic system responsible for mechanically stimulate the sample, has its control performed by an application GUI that is responsible for the communication with the microcontroller, which controls the pneumatic system, as well as the collection and processing of data obtained by the electrometer.

Keywords: Measurement System, piezoelectricity, electret, automation, arduino, Qt

Sumário

1 Introdução

Como necessidade básica para um futuro estudo e pesquisa mais amplos na construção e utilização de piezoeletretos, o objetivo deste trabalho é construir uma plataforma automatizada para teste de amostras piezoelétricas, ou seja, o desenvolvimento de um sistema que execute de forma rápida, precisa e autônoma todo o processo de estimulação mecânica de uma amostra piezoelétrica, a coleta e o processamento dos dados obtidos em cada ensaio.

O processo de medida do coeficiente piezoelétrico d_{33} das amostras termo-formadas produzidas pelo Grupo de Alta Tensão e Medidas (GATM) da Escola de Engenharia de São Carlos, que será descrito no capítulo 2 do presente trabalho, tem sido realizado através de um processo direto quase-estático em que a coleta dos dados necessários para a obtenção do coeficiente é executada de forma manual, obtendo-se a densidade superficial de carga diretamente do eletrômetro e realizando de forma não automatizada os cálculos necessários.

Esse processo de coleta e processamento manual dos dados motiva este trabalho de automação do sistema de medição, tornando o processo de medida mais fácil, acelerando-o e automatizando-o de forma a reduzir o tempo gasto na realização dos ensaios em cada amostra e melhorando a repetibilidade do processo. Além disso, estabelece uma plataforma de medida automatizada que servirá de base para o desenvolvimento de processos automatizados de medida e análise de diversos outros aspectos de um material piezoelétrico, como por exemplo, a caracterização de seu comportamento no domínio da frequência, sua frequência de ressonância, etc.

Neste ensaio inicial, tem-se o objetivo de determinar o coeficiente piezoelétrico d_{33} da amostra, aqui denominado de *cp* (elucidado em sua forma de obtenção e significado no capítulo 2 deste trabalho), dado pela equação: [1]

$$cp = \frac{\Delta\sigma}{\Delta p}$$

sendo σ a densidade superficial de carga elétrica (pC/m^2) e *p* a pressão sobre a amostra (N/m^2).

Para isso, é necessária a determinação da quantidade de carga de uma amostra em função da força exercida sobre a mesma. Assim, o sistema de teste das amostras aqui apresentado, que foi desenvolvido pelo GATM, consiste de um suporte para a amostra conectado aos terminais da mesma, de forma a permitir a medida do sinal elétrico gerado ao se pressionar o piezoeletreto; um pistão móvel responsável por manter a amostra em seu lugar; e de um sistema pneumático

que controlará o pistão e o ar que exercerá uma pressão sobre a amostra e que gerará a deformação necessária para a realização da medida. O controle do ar pressionado contra a superfície da amostra é realizado por uma válvula eletropneumática (uma válvula cuja parte elétrica é composta por um solenóide que controla eletricamente o seu conjunto pneumático); o controle da pressão do ar é realizado manualmente; e o controle do pistão feito também através do sistema pneumático utilizando mais duas válvulas eletropneumáticas.

Portanto, a automação deste sistema consiste em controlar o pistão e o ar comprimido com um microcontrolador, e durante cada medida, obter e processar os dados lidos pelo eletrômetro, tudo isso através de uma interface gráfica de um aplicativo em um computador pessoal, comunicando sincronamente com o microcontrolador e o eletrômetro.

2 Revisão Bibliográfica

2.1 Piezoeletricidade

Define-se piezoeletricidade como a propriedade que alguns materiais possuem de gerar cargas elétricas em sua superfície quando estimulados mecanicamente (neste caso denominado de efeito piezoelétrico direto), assim como de apresentarem deformações mecânicas quando uma tensão elétrica é sobre eles aplicada (efeito piezoelétrico inverso).

Existem materiais naturalmente piezoelétricos, como alguns cristais, dentre os quais podemos citar o cristal de quartzo, assim como algumas cerâmicas, como o titanato zirconato de chumbo (PZT). Esses materiais possuem uma geometria molecular polarizada, na forma de dipolos, sendo justamente essa polarização natural de suas moléculas a principal responsável pela indução de cargas superficiais. As cargas elétricas induzidas na superfície desses materiais são de sinal contrário ao sentido da polarização. Assim, quando uma força externa é aplicada sobre eles, ocorre uma variação no vetor polarização alterando, por conseguinte, as cargas na superfície que se reorganizam numa nova conformação.

Além desses materiais naturalmente piezoelétricos, há a possibilidade de induzir a formação de dipolos em polímeros que não necessariamente possuem uma polarização molecular natural, de tal forma a gerar um material polimérico com propriedades piezoelétricas. Esses polímeros polarizados, capazes de induzir cargas elétricas superficiais, são denominados eletretos. Pode-se citar os polímeros porosos e celulares, assim como os ditos termo-formados [2].

2.1.1 Caracterização do efeito piezoelétrico

A equação a seguir descreve a relação entre a polarização do material e o efeito piezoelétrico

$$\vec{D} = \varepsilon_0 \vec{E} + \vec{P}$$

em que $\vec{D}$ é o vetor densidade de fluxo elétrico, ε_0 a permissividade do vácuo, $\vec{E}$ o campo elétrico e $\vec{P}$ o vetor de polarização [2].

A partir dela, obtêm-se a matriz dos coeficientes piezoelétricos, constantes que caracterizam as propriedades eletromecânicas de materiais piezoelétricos [3]:

$$\begin{pmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} & d_{16} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} & d_{26} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} & d_{36} \end{pmatrix}$$

em que o coeficiente d_{ij} representa a razão entre a pressão aplicada na direção do eixo j e o campo elétrico aplicado na direção do eixo i quando todo o estresse externo é mantido constante, isto é, i é a direção da polarização do material quando o campo elétrico externo é zero.

As direções X, Y, e Z são representadas pelos índices 1, 2 e 3, respectivamente, e o cisalhamento em relação a um desses eixos é representado pelos índices 4, 5, e 6, respectivamente [4]. As direções consideradas e seus respectivos índices podem ser vistos na figura 1.

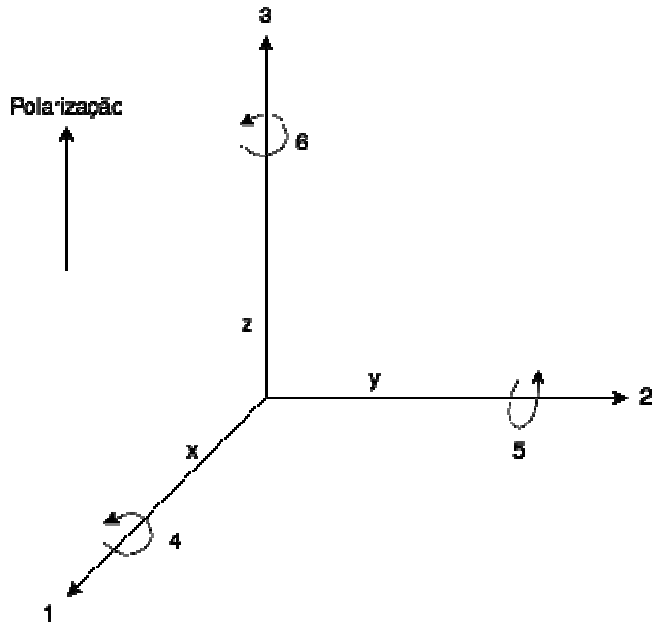

Figura 1: Diferentes direções de aplicação de força em um material piezoelétrico

No ensaio realizado neste trabalho, buscamos determinar o coeficiente d_{33} de amostras de materiais piezoelétricos termo-formados que representa a razão entre a polarização induzida na direção 3 (paralela a direção na qual o piezoeletreto foi polarizado) e a pressão aplicada na mesma direção sobre o material, ou seja, é a tensão induzida na direção 3 por unidade de campo elétrico aplicado na mesma direção.

2.1.2 Principais métodos de medida da piezoeletricidade

Os métodos de medida da piezoeletricidade podem ser classificados em dois grupos: métodos diretos e métodos inversos. Os métodos ditos diretos são aqueles que medem o efeito piezoelétrico direto, ou seja, medem a resposta elétrica do material quando uma determinada pressão mecânica é aplicada sobre ele. Os métodos ditos inversos são aqueles que medem o efeito piezoelétrico inverso, ou seja, medem a resposta mecânica do material para um determinado estimulo elétrico.

2.1.2.1 Métodos diretos de medida de piezoeletricidade

Os métodos diretos de medida de piezoeletricidade são ainda classificados, com base na frequência da pressão externa aplicada sobre o material, como sendo quase-estáticos ou dinâmicos.

O método de medida quase-estático é utilizado para determinar qualitativamente a existência do efeito piezoelétrico em um determinado material. Este método consiste em aplicar uma força constante sobre uma determinada área da amostra e medir o sinal elétrico gerado pela mesma. Uma das formas de se implementar esse método consiste em posicionar a amostra a ser medida entre dois eletrodos que por sua vez encontram-se conectados a um capacitor em paralelo C, como capacitância pelo menos 100 vezes maior do que a capacitância da amostra, e aplicar uma pressão sobre ela através de um corpo de massa *m* [2]. Medindo-se a tensão elétrica no capacitor, é possível obter o coeficiente piezoelétrico d_{33}, dado pela razão entre a carga elétrica Q e a força F aplicada sobre a amostra, através da equação:

$$d_{33} = \frac{Q}{F} = \frac{C.V}{F}$$

sendo *V* a tensão elétrica e *C* a capacitância medida do capacitor. O coeficiente piezoelétrico é expresso em pico Coulomb por Newton (pC/N).

O método quase-estático não determina a resposta em frequência das amostras, sendo necessária para a determinação do comportamento do material em relação à frequência a utilização de métodos diferentes, denominados dinâmicos.

O método direto dinâmico de medida do coeficiente piezoelétrico diferencia-se do método quase-estático somente quanto à forma em que a pressão é exercida sobre a amostra, sendo ela, ao invés de quase-estática, aplicada por um mecanismo de pressão capaz de aplicar em diferentes frequências a força sobre o material.

2.1.3 Processo de medida do sistema de medição desenvolvido (GATM)

O método utilizado pelo sistema em questão neste trabalho para medir o coeficiente piezoelétrico de uma amostra piezoelétrica termo-formada é o método direto (aplica-se uma pressão mecânica sobre o material, medindo a sua resposta elétrica) e quase-estático (que consiste em aplicar uma força constante sobre a amostra e medir o sinal elétrico gerado pela mesma). A figura 2 apresenta uma amostra piezoelétrica termo-formada produzida no GATM.

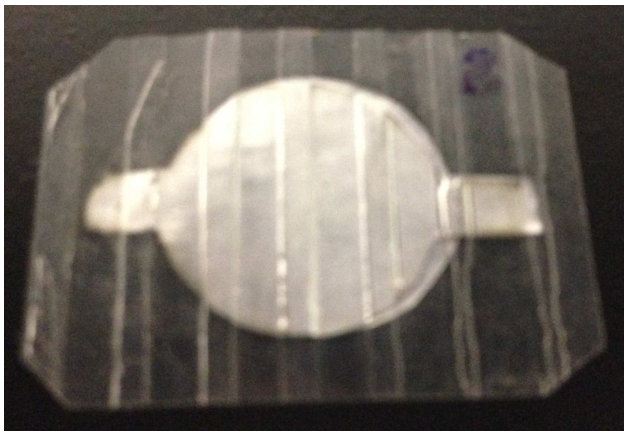

Figura 2: Amostra piezoelétrica termo-formada (GATM)

O sistema é composto por um par de eletrodos, um superior e um inferior. O eletrodo inferior é parte de um suporte metálico fixo de geometria anelar com abertura central de 1,5mm de diâmetro. O eletrodo superior é parte de um pistão metálico móvel de forma a obter um encaixe perfeito sobre o eletrodo inferior. A amostra piezoelétrica é colocada sobre o eletrodo inferior do suporte, de forma que, quando o pistão é acionado, o eletrodo superior é pressionado contra a amostra que permanece fixa entre os dois eletrodos. Além disso, o eletrodo inferior possui uma abertura central circular de área $1{,}77.10^{-4}$ m^2, cujo objetivo é permitir a excitação mecânica da amostra piezoelétrica por meio de um fluxo constante de ar comprimido sobre a amostra através dessa abertura. A figura 3 apresenta o sistema de medição de piezoeletricidade. A figura 4 mostra o par de eletrodos (suporte metálico inferior e pistão superior). A figura 5 apresenta uma vista superior do suporte da amostra.

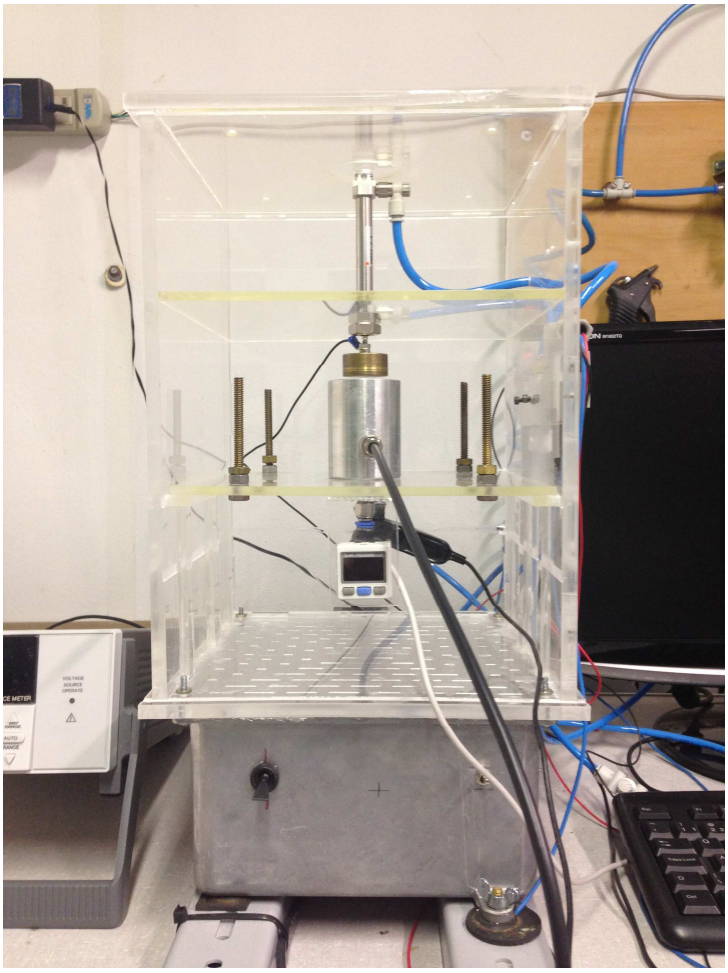

Figura 3: Sistema de medição de piezoeletricidade (GATM)

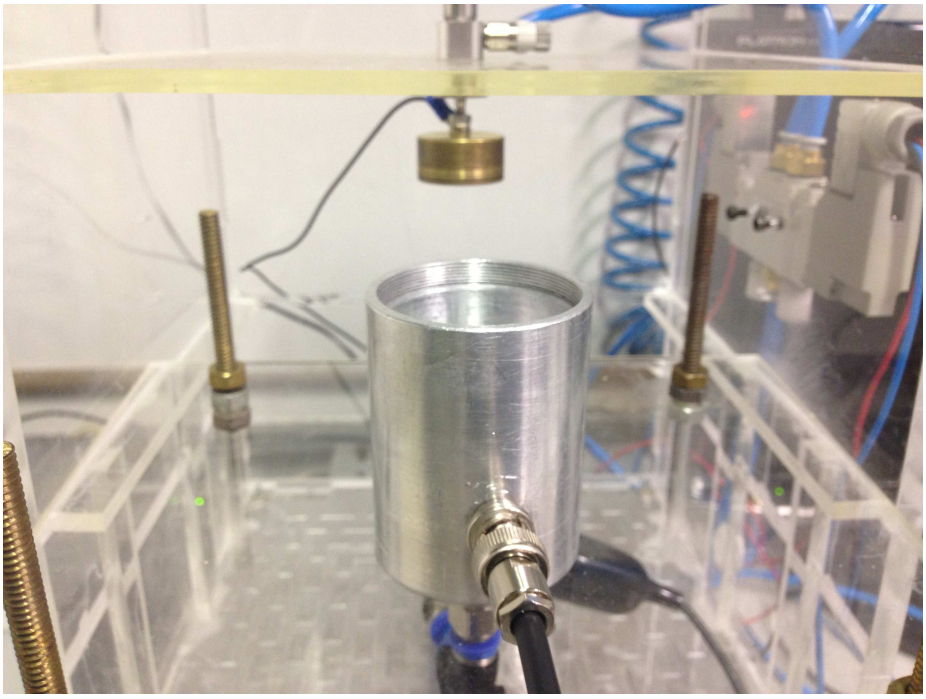

Figura 4: Par de eletrodos do sistema de medição de piezoeletricidade

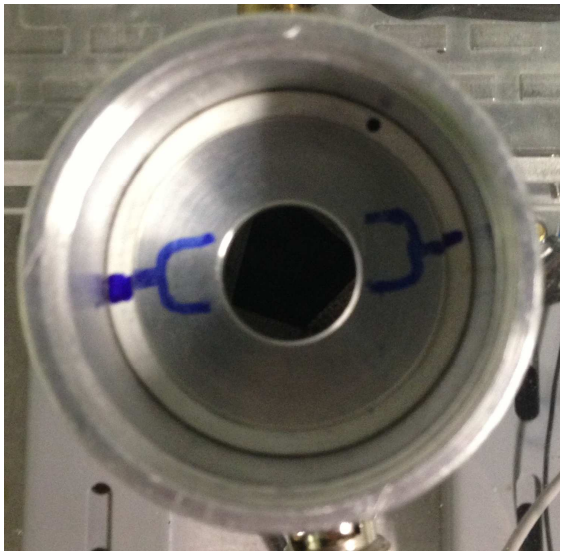

Figura 5: Vista superior do suporte da amostra

Assim, após o posicionamento da amostra sobre o eletrodo inferior, o pistão é acionado de forma a pressionar o eletrodo superior contra a amostra. A seguir, ar comprimido, de pressão conhecida e controlada, é liberado e conduzido até a amostra pelo orifício do eletrodo inferior de forma a comprimi-la e excitá-la mecanicamente.

Para medir a pressão que está sendo aplicada sobre a amostra, um pressostato do fabricante SMC, modelo ISE30, foi colocado na abertura inferior. Para controlar a pressão do fluxo de ar na amostra, foi utilizado um registro mecânico, o qual foi colocado entre o pressostato e a válvula elétrica. O pressostato utilizado pode ser visto na figura 6 e, conectado ao sistema, pode ser visto na figura 7. O registro mecânico utilizado para controlar a pressão do fluxo de ar na amostra pode ser visto na figura 8.

Figura 6: Pressostato SMC ISE30

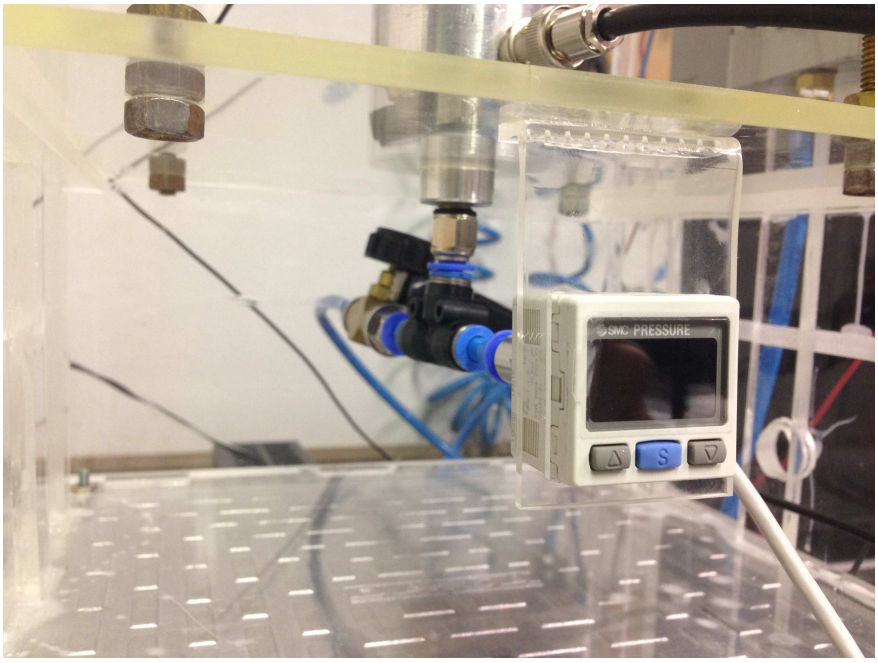

Figura 7: Pressostato conectado ao sistema de medida

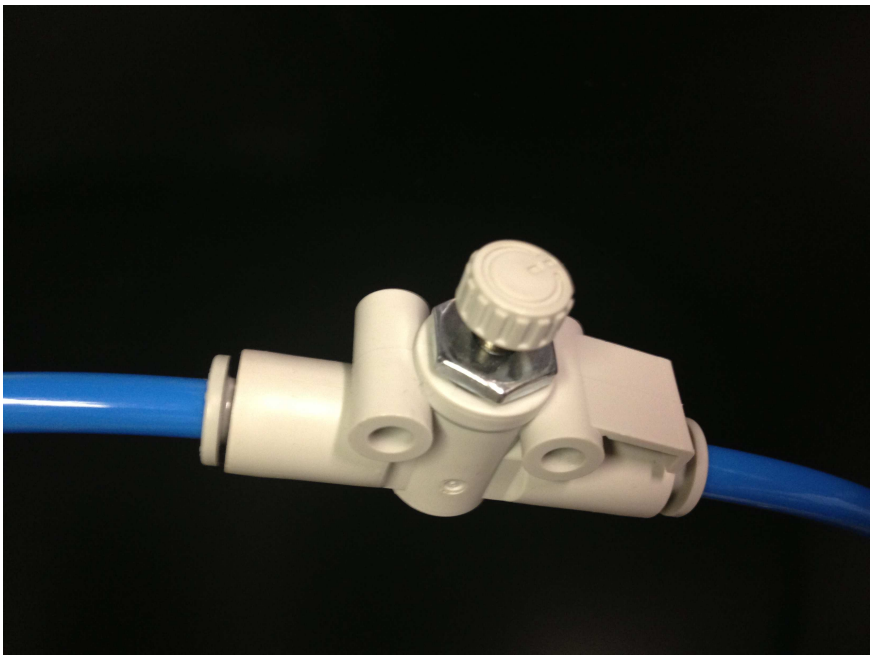

Figura 8: Registro mecânico utilizado para controlar a pressão do fluxo de ar na amostra

Para liberação e contenção do ar comprimido foi utilizada uma válvula elétrica, cujo acionamento era realizado através de um temporizador astável projetado com um circuito integrado 555. A medição foi realizada com ar comprimido sobre a amostra durante 5 segundos e sem o ar por mais 5 segundos, sendo esse tempo de medição passível de ser modificado por meio de um potenciômetro instalado.

A pressão de ar utilizada foi ajustada em 20 kPa, a qual representa uma força de 3,54 N. A força foi calculada por meio da equação abaixo, onde a área da amostra corresponde a abertura do eletrodo inferior ($1{,}77.10^{-4}$ m^2):

$$pressão = \frac{força}{área}$$

As cargas elétricas originadas pela deformação da amostra são medidas com o auxílio de um eletrômetro Keithley, modelo 6517B, o qual está conectado aos eletrodos superior e inferior do sistema. A figura 9 apresenta o eletrômetro utilizado.

Figura 9: Eletrômetro 6517B - Keithley Instruments (Fonte: [5])

O esquema elétrico do sistema funcionando somente com o 555, antes da automação, pode ser visualizado na figura 10.

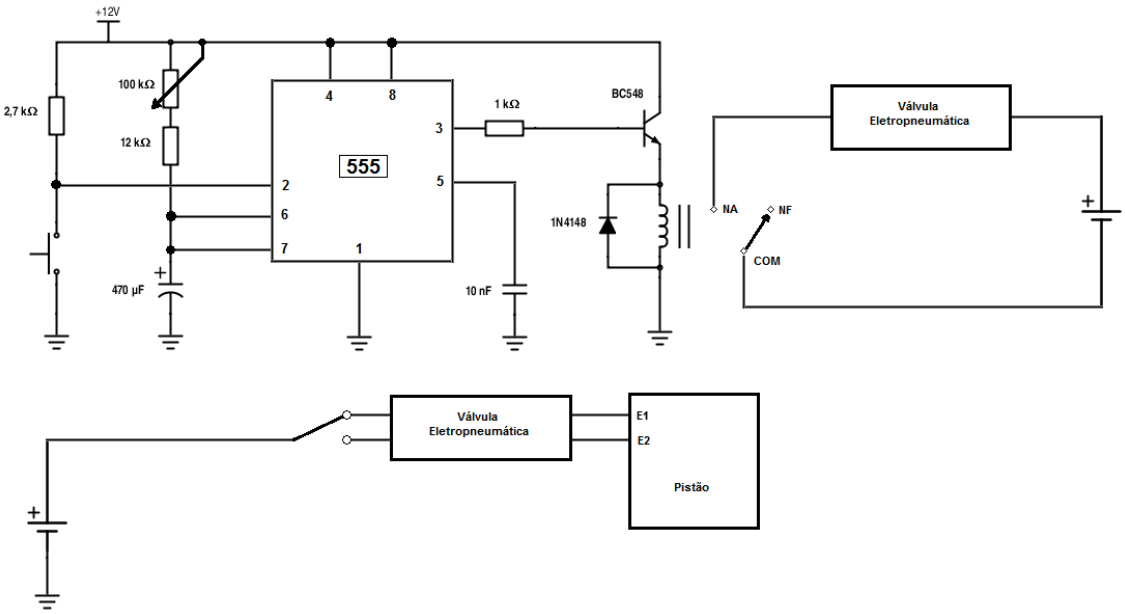

Figura 10: Esquema elétrico do sistema de medição de piezoeletricidade antes da automação

O sistema pneumático para medida do coeficiente piezoelétrico de piezoeletretos aqui descrito e automatizado, desenvolvido no GATM, não possui, até o presente momento, sistema similar para a realização deste método de medida, inovando na forma de realizar o processo de medição direta quase-estática ao utilizar ar comprimido para realizar a excitação mecânica da amostra piezoelétrica e um sistema pneumático na atuação do pistão móvel.

2.2 Arduino

Arduino é uma plataforma física de computação, de código aberto, baseada em uma placa microcontroladora e um ambiente de desenvolvimento para escrita de software para essa placa. Com um ambiente de programação simples e uma arquitetura poderosa, possibilita o rápido e eficiente desenvolvimento de softwares de controle e conexão com o mundo físico. Além disso, o programa de desenvolvimento para Arduino é multi-plataformas (rodando em sistemas operacionais Windows, Macintosh OSX e Linux), gerando uma imensa flexibilidade na execução de projetos que o utilizam. Além de disso, é uma opção de baixo custo em relação a outras plataformas microcontroladoras.

No site do fabricante [6] está disponível uma rica documentação a cerca de como utilizar a plataforma, com diversos exemplos práticos de soluções dos mais variados problemas que se pode encontrar em um projeto.

Ainda no site do fabricante, nota-se a existência de diferentes placas controladoras com diferentes características. No projeto desenvolvido neste trabalho, escolheu-se utilizar a placa Arduino Uno, a mais simples das placas disponíveis que atende perfeitamente todos os requisitos do projeto. Esta placa, em sua terceira revisão (Arduino Uno R3), tem seu esquemático reproduzido no apêndice [7].

2.2.1 Arduino Uno: visão geral

O Arduino Uno é uma placa microcontroladora baseada no microcontrolador ATmega328. O *datasheet* do ATmega328 encontra-se em [8]. O Arduino Uno possui 14 pinos de entrada e saída (E/S) digitais (dos quais 6 podem ser utilizados para enviar sinais de saída modulados por largura de pulso, PWM), 6 pinos de entrada analógicos, um ressoador cerâmico de 16MHz, um conector USB, um conector de alimentação, um *ICSP header* e um botão de *reset*, como pode-se ver na figura 11.

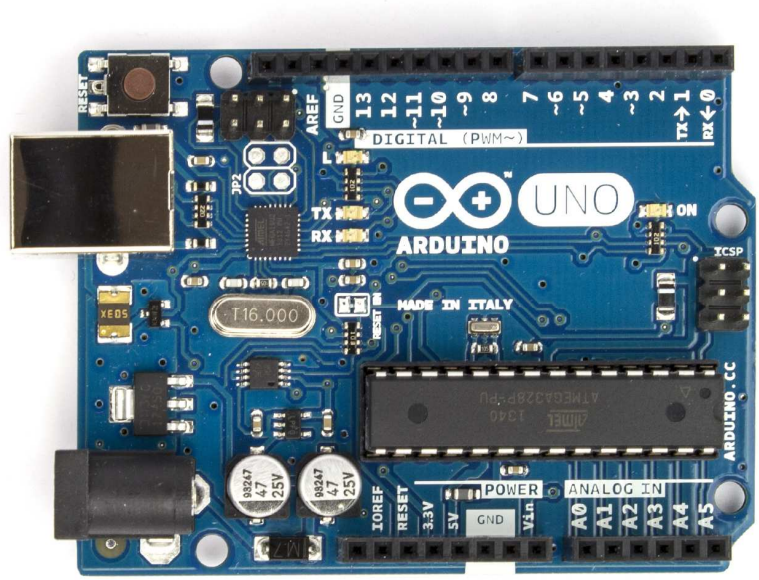

Figura 11: Arduino Uno R3 (Fonte: [6])

Assim, esta placa contém tudo o que é necessário para dar suporte ao microcontrolador dela, bastando conectá-la a um computador através de um cabo USB ou alimentá-la com um adaptador AC/DC ou bateria para colocá-la em funcionamento.

A seguir, é fornecido um sumário da placa microcontroladora Arduino Uno R3 [9]:

- Microcontrolador: ATmega 328
- Voltagem de Operação: 5V
- Voltagem de alimentação (recomendada): 7-12V
- Voltagem de alimentação (limites): 6-20V
- Pinos de Entrada e saída digitais: 14, dos quais, 6 fornecem saída PWM
- Pinos de entrada analógicos: 6
- Corrente DC por pino de E/S: 40mA
- Corrente DC do pino de 3,3V: 50mA
- Memória Flash: 32 kB (ATmega328), dos quais 0,5kB são usados pelo *bootloader*
- SRAM: 2 kB (ATmega328)
- EEPROM: 1 kB (ATmega328)
- Velocidade de *Clock*: 16MHz
- Comprimento: 68,6 mm
- Largura: 53,4 mm
- Massa: 15g

2.2.2 Alimentação

A placa Arduino Uno pode ser alimentada pela conexão USB ou ainda por uma fonte de alimentação externa. A escolha da fonte de alimentação é realizada de forma automática pela placa. No caso de se utilizar uma fonte externa de alimentação que não seja USB, pode ser usado uma bateria ou um conversor AC/DC conectado a placa por um conector de 2.1mm de centro positivo. A placa pode operar com uma fonte externa de 6 a 20 V. No entanto, caso seja alimentada com menos de 7 V, os pinos da placa que fornecem 5 V podem não conseguir fornecer a voltagem necessária e a placa pode ficar instável. Caso seja fornecido mais que 12 V, o regulador de tensão da placa pode sobreaquecer e danificar a mesma, sendo, portanto, recomendado utilizar tensões de alimentação entre 7 e 12 V apenas [9].

O circuito regulador para a entrada externa, destacado do esquemático [7], é mostrado na figura 12. Nele, pode-se ver a utilização do CI NCP1117, da OnSemi, para a regulagem da tensão e a presença do diodo D1 para proteger o circuito no caso de uma fonte com tensão invertida seja ligada à placa [10].

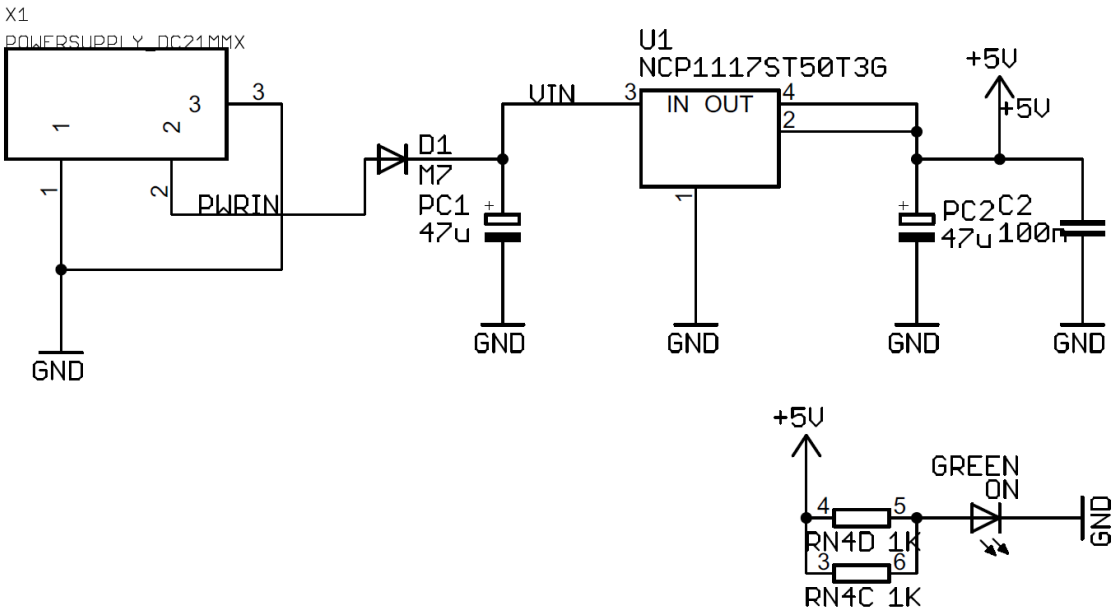

Figura 12: Circuito regulador para a entrada externa

Ao conectar o cabo USB, na ausência de outra fonte externa de alimentação, a placa é alimentada diretamente pela USB. O circuito de seleção da fonte de alimentação pode ser visto na figura 13. Caso haja uma tensão no conector de alimentação DC, a placa será alimentada pela fonte externa enquanto que a conexão USB será utilizada somente para comunicação com o computador. Pode-se observar nesse mesmo circuito a existência do componente LP2985

responsável por fornecer uma tensão continua de 3,3 V para a alimentação de circuitos, sendo a corrente máxima que esse regulador pode fornecer de 50 mA.

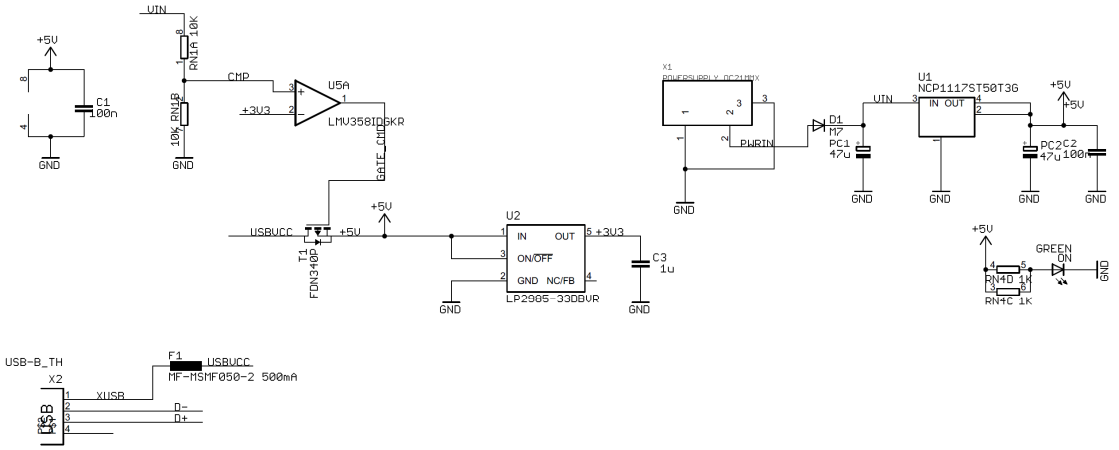

Figura 13: Circuito de seleção da fonte de alimentação da placa Arduino Uno

O Arduino Uno possui um fusível reinicializável (*resettable ployfuse*, em inglês) que protege a porta USB do computador de sub e sobrecorrentes. Apesar da maioria dos computadores possuírem sua própria proteção interna, a placa possui essa camada extra de proteção. Caso uma corrente maior que 500 mA seja aplicada na porta USB, o fusível automaticamente irá interromper a conexão enquanto a sub ou sobrecarga estiver presente. Pode-se ver destacado do esquemático o circuito de proteção da USB da placa Arduino Uno na figura 14.

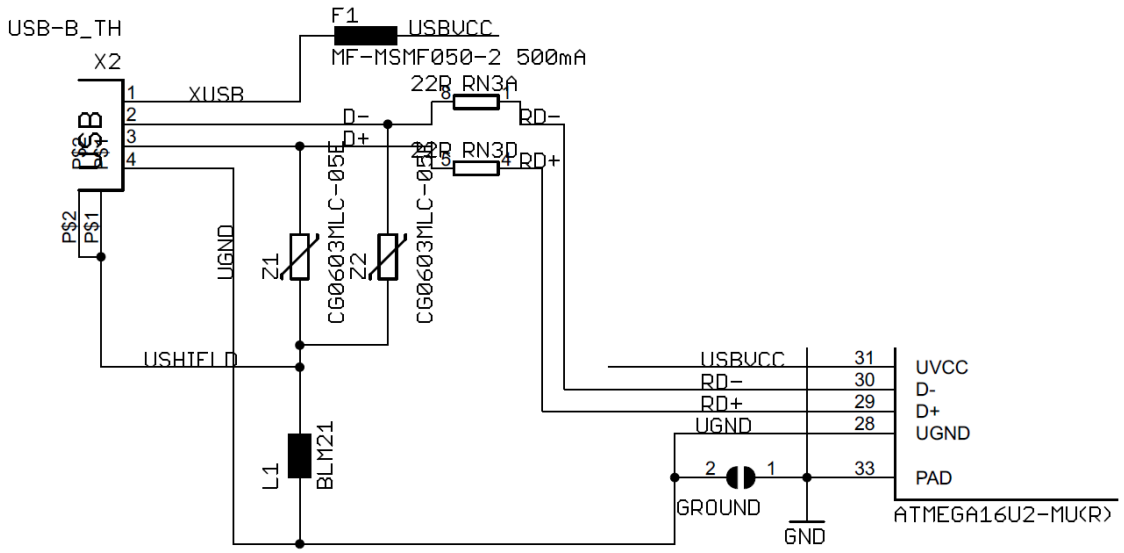

Figura 14: Circuito de proteção da USB da placa Arduino Uno

Os pinos de alimentação são os seguintes, como podemos ver na figura 15:

- VIN: Pino para alimentação externa da placa. Fornece a voltagem de alimentação do Arduino quando se utiliza uma fonte externa através do conector de alimentação (7-12V). Observar que fornecer tensão pelo pino de 5V ou pelo pino de 3,3V ignora o regulador de tensão e pode danificar a placa. Assim, utilizar este pino VIN, o conector USB ou o conector de alimentação para fornecer energia à placa.
- 5V: Pino cuja saída constante é de 5V regulada pelo regulador de tensão da placa. A placa pode estar alimentada através do conector DC (7-12V), através do conector USB (5V) ou através do pino VIN da placa (7-12V).
- 3,3V: Fornece tensão constante de 3,3V regulada pelo regulador de tensão da placa e máxima corrente de 50mA
- GND: Pinos de referência (terra)
- IOREF: Fornece a tensão de referência com a qual o microcontrolador opera.

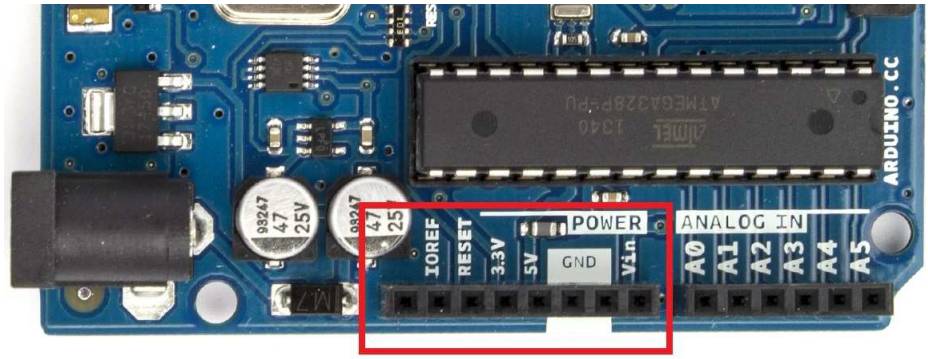

Figura 15: Pinos de alimentação da placa Arduino Uno, em destaque

2.2.3 Microcontrolador: ATmega328

O microcontrolador utilizado no Arduino Uno é o ATMEL ATmega328 de 8 bits da família AVR possuindo uma arquitetura RISC e encapsulamento DIP28, mostrado na figura 16. Ele possui memória flash de 32kB, dos quais 512 Bytes são utilizados pelo *bootloader*; 2kB de SRAM e uma EEPROM de 1kB. Podendo operar em até 20 MHz, no Arduino Uno ele opera a 16MHz, a frequência do cristal oscilador externo conectado aos pinos 9 e 10 do microcontrolador.

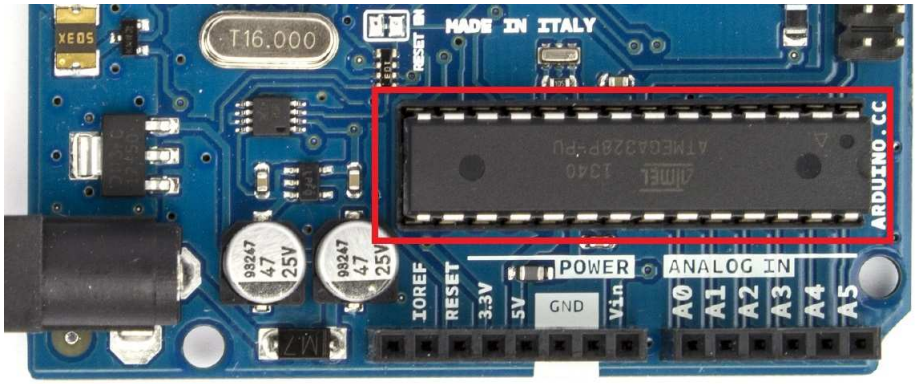

Figura 16: Microcontrolador Atmega328 na placa Arduino Uno, em destaque

O ATmega328 possui vários periféricos internos, os quais a plataforma Arduino aproveita e disponibiliza a maioria dos pinos do microcontrolador para serem utilizados.

Com 28 pinos, 23 deles podem ser usados para entradas e saídas digitais. Pode operar com tensões tão baixas quanto 1,8V, porém com um *clock* máximo de 4MHz para essa tensão.

Possui um conversor analógico-digital com 10 bits de resolução e 8 opções de entrada, sendo normalmente usada apenas 6 no encapsulamento DIP no Arduino. Possui três opções de referência: a tensão de alimentação, fornecida separadamente pelo pino AVcc; uma referência interna de 1,1V e uma tensão externa fornecida através do pino ARef.

Possui uma USART (*Universal Synchronous/Asynchronous Receiver/Transmitter*) que opera em *full-duplex*, com 5 a 9 bits de dados e 1 ou 2 bits de parada e paridade, uma SPI que opera em até 5MHz e um módulo TWI (Two *Wire Interface*) que implementa o padrão I2C e pode operar em até 400kHz.

O ATmega328 possui também um módulo comparador analógico interno e três temporizadores: Timer0, de 8 bits; Timer1, de 16 bits; e Timer2 de 32 bits, podendo os três temporizadores serem usados para geração de sinais PWM (*Pulse Width Modulation*).

2.2.4 Entrada e Saída

Cada um dos 14 pinos digitais do Arduino Uno, em destaque na figura 17, podem ser utilizados como pino de entrada ou de saída, utilizando as funções *pinMode()*, *digitalWrite()* e *digitalRead()*. Eles operam a uma tensão de 5V e podem fornecer ou receber no máximo 40mA, possuindo cada pino um resistor *pull-up* interno de 20 a 50kΩ, desconectado por padrão e que pode ser habilitado via software.

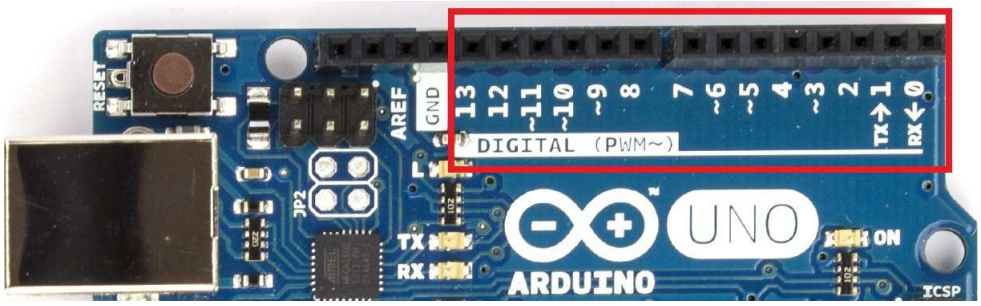

Figura 17: Pinos digitais de entrada e saída do Arduino Uno, em destaque

Alguns desses pinos possuem, além disso, funções especializadas:

- Comunicação Serial: Pinos 0 (RX) e 1 (TX). Utilizados para receber (RX) e transmitir (TX) dados seriais TTL. Esses pinos estão conectados aos pinos correspondentes do chip ATmega8U2 *USB-to-TTL* serial.
- Interrupções Externas: pinos 2 e 3. Esses pinos podem ser configurados para disparar uma interrupção em um valor lógico baixo, em uma borda de subida ou descida, ou em uma mudança de estado (através da função *attachInterrupt()*).
- Modulação por Largura de Pulso (PWM): pinos 3, 5, 6, 9, 10 e 11. Fornece uma saída PWM de 8 bits através da função *analogWrite()*.
- SPI: pinos 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Esses pinos suportam comunicação SPI utilizando a biblioteca SPI.
- LED: pino 13. A placa possui um LED conectado ao pino digital 13. Quando o valor lógico desse pino é ALTO, o LED ascende, quando é BAIXO, ele permanece apagado. O LED conectado ao pino 13 pode ser visto em destaque na figura 18.

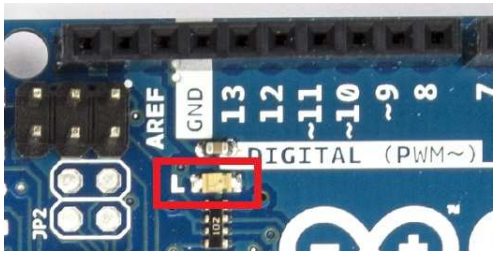

Figura 18: LED conectado ao pino digital 13 da placa, em destaque

Além desses pinos digitais, o Uno possui 6 pinos de entrada analógica, denominados A0 até A5, cada um possuindo uma resolução de 10 bits (1024 valores diferentes). Por padrão, eles medem os valores de GND a 5V, mas é possível alterar a referência superior do intervalo utilizando o pino AREF e a função *analogReference()*. Os pinos de entrada analógica podem ser vistos em destaque na figura 19.

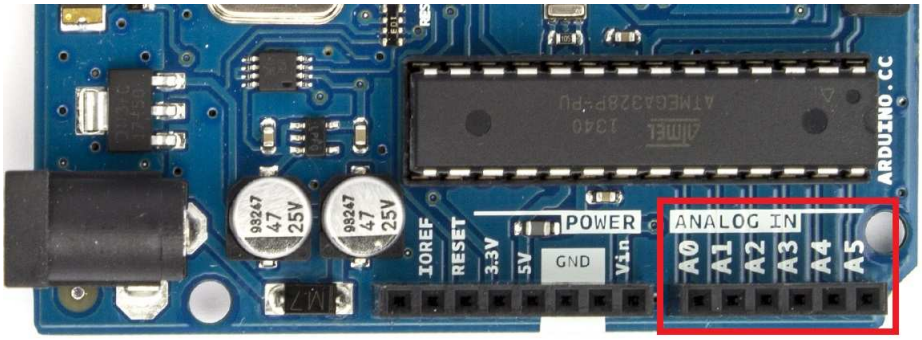

Figura 19: Pinos de entrada analógica do Arduino Uno, em destaque

Além disso, alguns pinos analógicos também possuem funções especializadas:

- TWI: pinos A4 (SDA) e A5 (SCL). Dão suporte a comunicação TWI utilizando a biblioteca *Wire*.
- AREF: Voltagem de referência para as entradas analógicas. (Utilizado com a função *analogReference()*)
- Reset: Quando assume valor lógico BAIXO, reinicia o microcontrolador.

2.2.5 Comunicação

Seja a comunicação com um computador, outro arduino ou microcontrolador, a interface de comunicação é bem flexível e fácil de usar. O microcontrolador ATmega328 fornece uma comunicação serial através de uma UART TTL, disponível através dos pinos 0 (RX) e 1 (TX). Como interface USB para comunicação com o computador, há o microcontrolador ATmega16U2 na placa (como podemos ver na figura 20, em destaque), o qual conduz a comunicação serial através da USB de forma que ela aparece no computador como uma porta de comunicação serial virtual (*COM Port*). Assim, o ATmega16U2 é responsável pela transparência do processo de comunicação, utilizando um *firmware* que usa os *drivers* de comunicação USB padrões, não necessitando instalar nenhum *driver* externo, possibilitando carregar o código binário gerado pelo programa do usuário de forma simples e transparente.

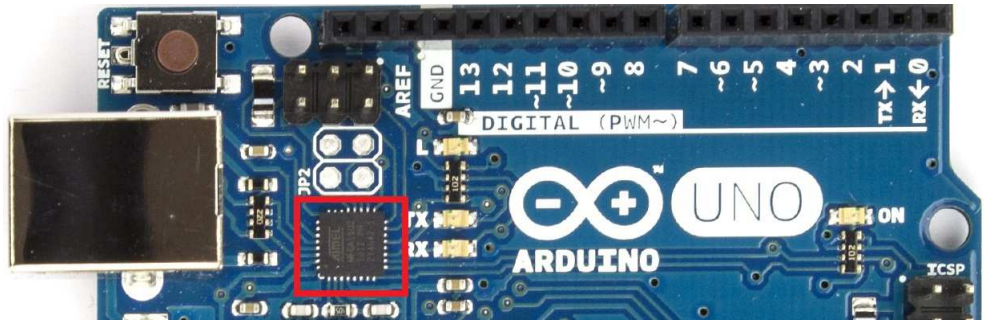

Figura 20: Chip USB-to-serial ATmega16U2 em destaque

No ATmega16U2 também estão conectados dois LEDs (RX e TX), controlados por esse microcontrolador, os quais sinalizam o recebimento e a transmissão de dados entre a placa e o computador. Os LEDs sinalizadores RX/TX serial podem ser vistos em destaque na figura 21.

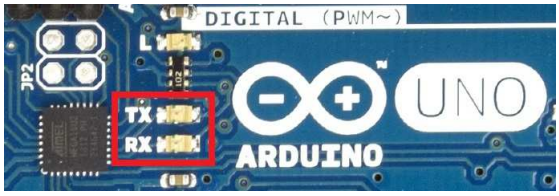

Figura 21: LEDs sinalizadores RX/TX serial, em destaque

O software do Arduino inclui um monitor serial que permite enviar e receber dados de texto simples entre o computador e a placa.

O esquemático do circuito responsável pela comunicação serial é mostrado na figura 22.

Além disso, a biblioteca *SoftwareSerial* permite a comunicação serial com qualquer um dos pinos digitais do Arduino Uno, expandindo assim as possibilidades de comunicação.

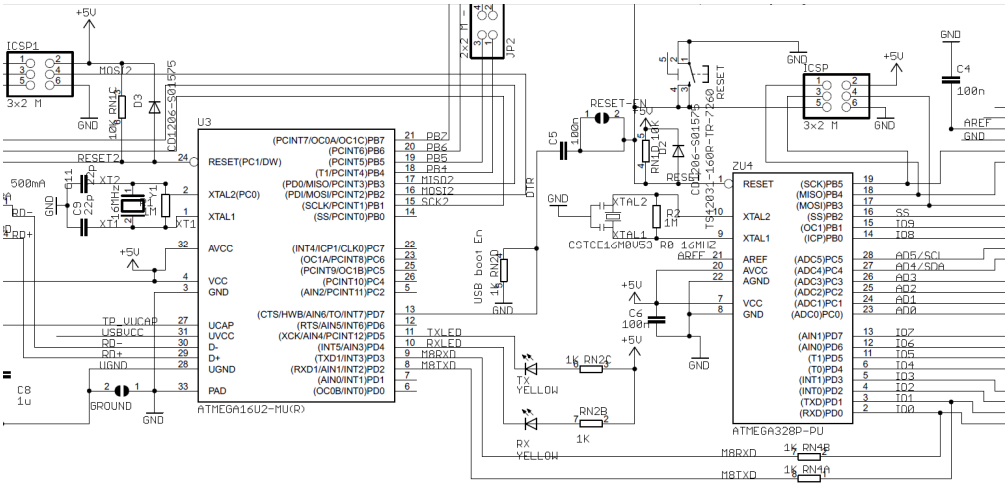

Figura 22: Circuito de comunicação serial

2.2.6 Programação

O Arduino Uno pode ser programado através da interface gráfica do software do Arduino, sendo a comunicação entre a placa e o computador realizada através de uma comunicação serial. Isso é possível uma vez que o ATmega328 presente no Arduino Uno já vem com um *bootloader* carregado em sua memória, o que possibilita o *upload* de novo código para o microcontrolador sem a necessidade de uso de um hardware programador externo.

Alternativamente, pode-se programar o microcontrolador através de um ICSP (*In-Circuit Serial Programming*).

Na figura 23 é apresentada a interface do software de programação para o Arduino.

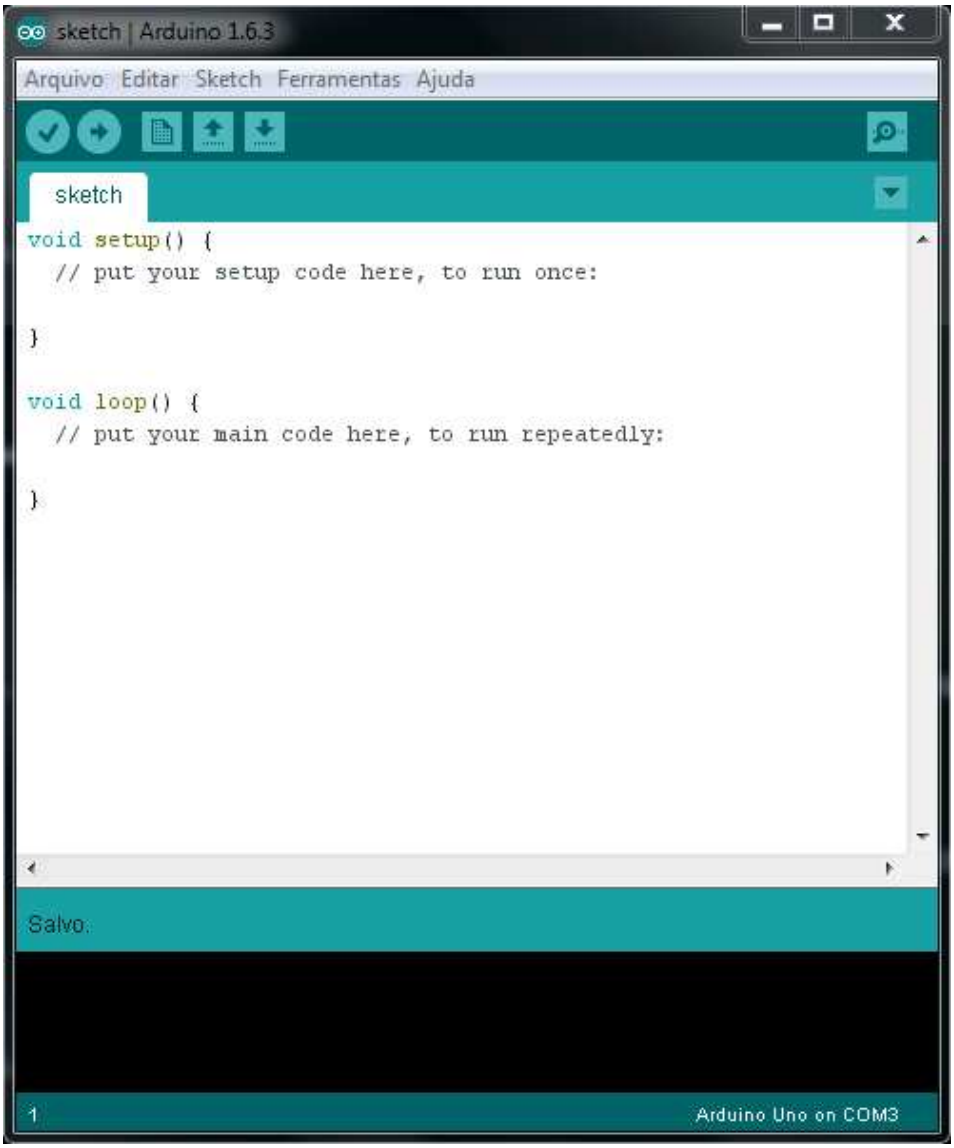

Figura 23: Interface do software de programação para o Arduino

2.3 Qt

Qt é um *framework* multi-plataforma [11], de código aberto, que possibilita a criação de aplicativos capazes de rodar em várias plataformas de hardware e software com pouca ou nenhuma alteração no código-fonte, além de fornecer um arcabouço de soluções integradas para software que torna possível em relativamente poucas linhas de código realizar aplicações robustas e poderosas.

Como um framework, ele fornece uma plataforma para representar elementos gráficos da interface do usuário, como botões e menus, e tratar todas as interações do usuário com esses elementos, possibilitando ainda definir as relações entre os elementos do programa de forma hierarquizada. Tudo isso é fornecido através de uma plataforma que lida automaticamente e de forma transparente com os detalhes de baixo nível da programação, permitindo ao usuário se concentrar nas funcionalidades e arquitetura de sua aplicação. O ambiente de desenvolvimento integrado (IDE) do Qt pode ser visto na figura 24

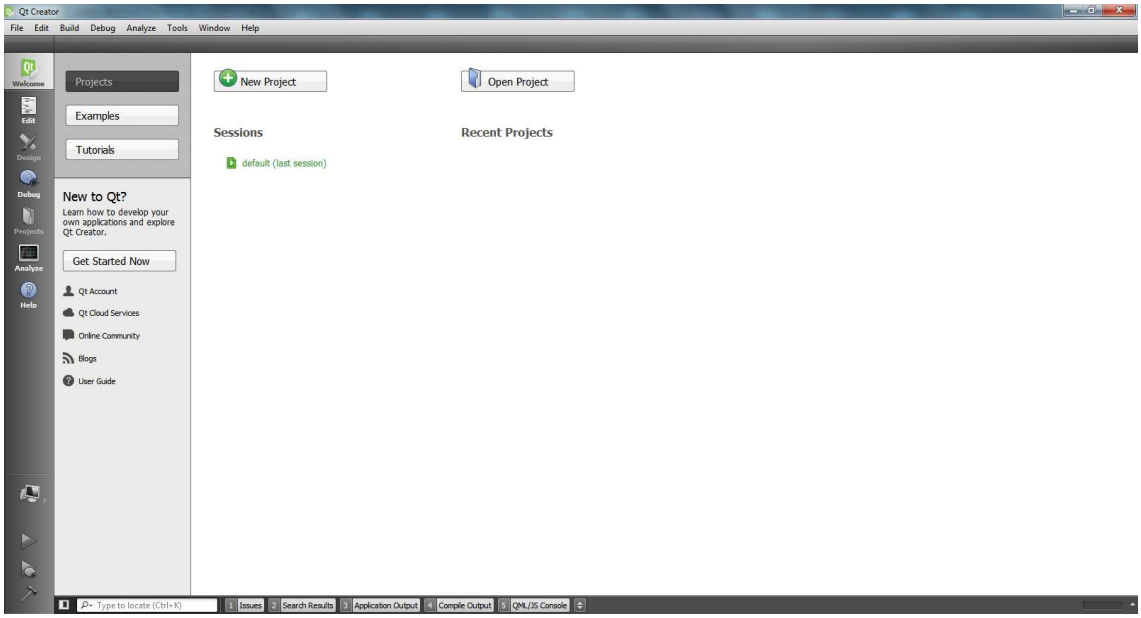

Figura 24: Ambiente de desenvolvimento do Qt

Qt usa a linguagem C++ padrão com extensões e, assim como no caso do Arduino, no site do desenvolvedor [11] está disponível uma rica documentação a cerca de como utilizar o *framework*, com diversos exemplos práticos de soluções dos mais variados problemas que se pode encontrar em um projeto de software.

2.3.1 Implementação de bibliotecas para GUI multi-plataformas

A implementação de bibliotecas para interfaces gráficas (GUI) multi-plataformas pode ser realizada de diversas maneiras em relação à API (*Application Programming Interface*) nativa do sistema operacional em que o programa será executado [12].

Uma maneira possível é a chamada API em camadas (*API Layering*), que consiste em implementar sua própria API sobre a API nativa do sistema. Isso significa que precisaria existir uma implementação de seu framework para cada sistema diferente. A vantagem desse método está na relativa facilidade de escrita de tais bibliotecas, sendo elas completamente compatíveis em aparência e comportamento em relação a programas nativos do sistema em questão. No entanto, programas desenvolvidos em ferramentas que utilizam essa técnica são normalmente mais lentos do que aqueles que usam a API nativa diretamente pelo fato de cada chamada ao sistema terá que ser encaminhada por uma camada a mais de processamento.

Outra forma possível é a chamada emulação de API (*API Emulation*), que consiste em emular a API de um sistema em todos os outros em que desejamos portar nosso programa. Além do fato de que as diversas plataformas são muito diferentes umas das outras, o que torna a emulação de API não muito prática, os programas que são executados em plataformas emuladas são mais lentos do que aqueles executados em plataformas não emuladas por causa da camada adicional.

A terceira forma de se implementar uma ferramenta multi-plataformas é a chamada emulação de interface gráfica (*GUI Emulation*). Ferramentas que implementam a emulação de GUI, como é o caso do Qt, utilizam diretamente as funções gráficas primitivas de cada plataforma, além de realizar todo o processamento gráfico dentro da própria ferramenta. Assim, tem-se que Qt e outros emuladores de GUI são mais rápidos do que outras ferramentas multi-plataformas, além de que, como todo o desenho da interface é realizado dentro da ferramenta, torna-se simples criar uma aplicação com um estilo gráfico de um sistema em outro completamente diferente. Por exemplo, criar um aplicativo com a aparência nativa do Windows em um sistema operacional baseado em Linux, e vice-versa. A desvantagem é a possibilidade da emulação não ser exata, de uma pequena diferença no aspecto visual do programa e em seu comportamento existir dependendo do sistema operacional utilizado, porém, essa diferença é normalmente imperceptível ao usuário. Outra desvantagem, mais relevante, é a cada novo *widget* (um widget é um componente de uma interface gráfica do usuário (GUI), como por exemplo: janelas, botões, menus, ícones, etc) introduzido na plataforma pelo desenvolvedor da ferramenta, deve-se codificar especificamente o comportamento desse *widget* para cada estado, assim como para

uma emulação completa, toda a parte de gráfica que lida com desenhos deve ser reescrita para todas as plataformas suportadas.

Temos, portanto, que apesar das desvantagens, a técnica de emulação de GUI é a mais interessante de ser utilizada devido a sua velocidade, por gerar uma resposta mais rápida da interface, sendo justamente essa agilidade e fluidez na execução da GUI muito mais importante na percepção do usuário.

2.3.2 *Signals and Slots*

Qt possui um foco em programação de interfaces gráficas do usuário (GUI). Portanto, a estrutura central de controle do fluxo de funcionamento dos programas codificados em Qt é na forma de laço de eventos (*event loop*), isto é, o chamado laço principal (*main loop*) aguarda eventos e mensagens e as redireciona para os objetos associados a eles para serem tratados. Ou seja, ao se programar uma GUI, deseja-se que objetos de diferentes naturezas comuniquem-se uns com os outros. Como exemplo desse tipo de comunicação desejada, se o usuário clica no botão “Fechar” da janela da aplicação, é provável que desejemos chamar a função *close()* dessa janela.

Uma maneira de se conseguir essa comunicação é através dos chamados “retornos” (*callbacks).* Um *callback* é um ponteiro para uma função. Assim, caso deseje-se que uma função ativa, sendo processada, notifique sobre determinado evento que ocorreu, passa-se de antemão um ponteiro de uma outra função (o chamado *callback*) para essa função ativa, que chamará essa função de retorno (*callback function*) quando apropriado. Os problemas fundamentais de se usar *callbacks* são: Eles não são *type-safe*, isto é, não há a possibilidade de se ter certeza de que a função sedo processada irá chamar o *callback* com os argumentos corretos.; além disso, o *callback* é fortemente acoplado a função ativa que o chama, uma vez que a função sendo processada no momento necessariamente precisa saber qual *callback* chamar [12].

No Qt é usada uma alternativa a técnica de callback: é o sistema de *Signals and Slots* para a comunicação entre objetos, o que simplifica o tratamento de eventos. Um objeto emite um sinal indicando que determinado evento ocorreu, sem saber que objeto receberá este sinal. O sinal simplesmente é emitido e fica disponível para todos os *widgets* que estiverem a ele associados utilizá-lo [11].

Sinais são emitidos por objetos quando eles alteram seu estado de forma que interessa a outros objetos. Essa é a única forma de comunicação que um objeto possui: ele não sabe se alguém está

recebendo seu sinal, ele simplesmente emite. Assim, o sistema de *Signals and Slots* possibilita o encapsulamento de informação, garantindo que cada objeto poderá ser utilizado como um componente de software.

Um S*lot* é justamente a definição de uma função que espera por um sinal, independentemente se este sinal existe ou se foi emitido. Ou seja, *Slots* podem ser utilizados para receberem sinais, mas também são funções normais de um objeto. Assim como um objeto não sabe se algum outro receberá seu sinal emitido, um *Slot* não sabe se há algum sinal conectado a ele. Isso permite a criação de componentes completamente independentes.

Podem-se conectar vários sinais a um único *Slot*, isto é, vários sinais diferentes ativarão o mesmo procedimento; assim como um único sinal pode ser associado a vários *Slots*. Ainda, é possível conectar um sinal a outro sinal, sendo emitido o segundo sinal assim que o primeiro for emitido.

O mecanismo de *Signals and Slots* é *type-safe* uma vez que a assinatura do sinal deve coincidir com a assinatura do *slot* que o recebe, ou pelo menos que os argumentos do *slot* coincidam com os primeiros argumentos do sinal, já que um *slot* pode ignorar argumentos extras. Com a compatibilidade de assinaturas, o compilador é capaz de auxiliar no processo de checagem de incongruência de tipos. Além disso, esse mecanismo é fracamente acoplado: uma vez emitido o sinal por uma classe, ela não possui vínculo nenhum com qualquer objeto que venha a utilizar o sinal por ela emitido. O mecanismo garante que ao conectar um sinal com um *slot*, o *slot* associado será chamado com os parâmetros do sinal emitido no momento certo. Sinais e *slots* podem conter quantos argumentos forem necessários, sendo completamente *type-safe*.

Na figura 25 tem-se um esquemático do funcionamento do sistema de sinais e *slots*.

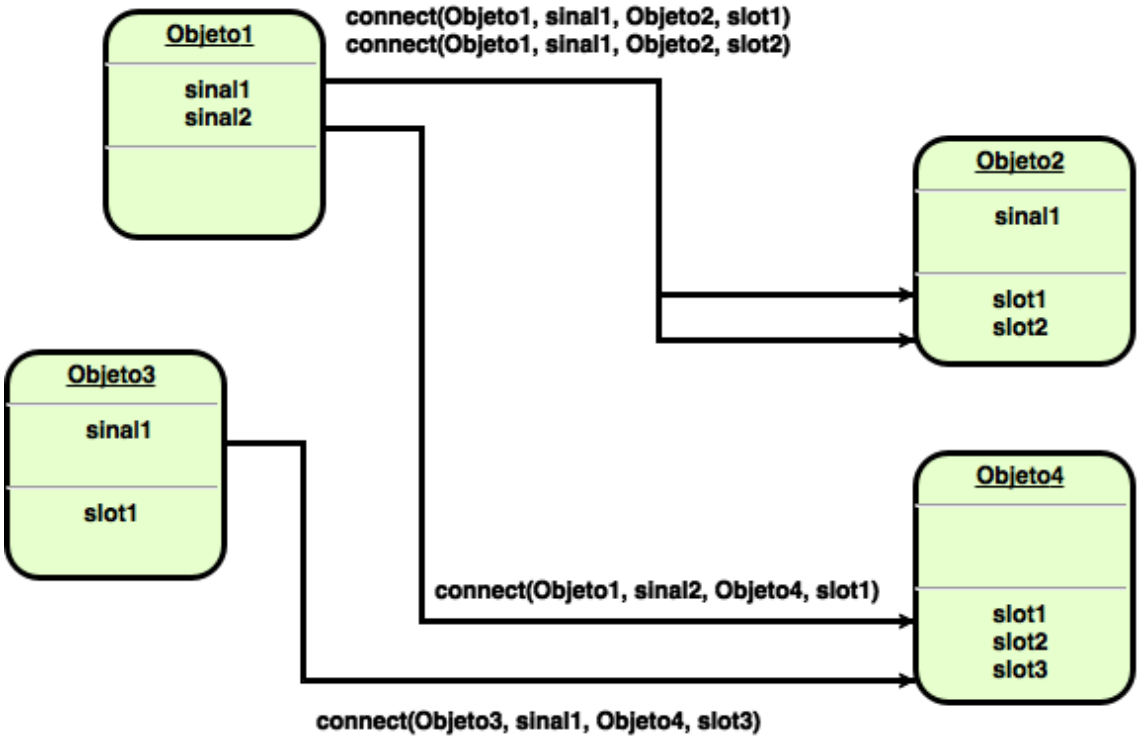

Figura 25: Esquema do funcionamento do sistema de *Signals and Slots*

Assim vê-se que juntos, sinais e *slots* formam um poderoso mecanismo para programação de componentes, sendo esse um dos fatores considerados para a escolha desse *framework* neste projeto.

2.3.3 *MOC*

Outra característica da arquitetura de Qt é o compilador de meta-objetos , *moc*, uma ferramenta executada sobre o código-fonte de uma programa Qt. Ele interpreta certas macros do código, em C++, e as utiliza para gerar código adicional C++ com meta-informações sobre as classes utilizadas no programa. Essas informações são utilizadas para prover funcionalidades não disponíveis nativamente em C++, como *signals and slots* e chamadas assíncronas de funções.

3 Descrição do Trabalho

Devido a sua robustez, poder e simplicidade, foi escolhida a plataforma Arduino para controle do sistema em questão. A interface de comunicação serial com o computador via conector USB, destaca-se pela sua conveniência, simplicidade e praticidade. Da mesma forma, a alimentação da placa através do conector USB mostra-se prática e simples. Além disso, a ampla disponibilidade de pinos de E/S e seu uso extremamente simples e direto foram fatores marcantes na escolha da placa microcontroladora. Também, um fator importante levado em consideração foi o ambiente de desenvolvimento do software para a placa: claro e intuitivo, além de possibilitar tanto a codificação quanto a gravação do código na placa de uma maneira extremamente rápida. Quanto ao desenvolvimento do código, além de possuir uma linguagem simples e poderosa, a ampla documentação e disponibilidade de exemplos diversos, assim como a existência de bibliotecas prontas, torna a plataforma Arduino um meio muito eficiente de desenvolvimento de soluções de controle de diversos tipos.

Para a interface gráfica com o usuário através do computador, foi escolhido o *framework* Qt pela sua versatilidade em termos de plataforma (multi-plataforma), e principalmente por possibilitar um desenvolvimento extremamente ágil de uma interface amigável, atraente e completamente funcional. O mecanismo de tratamento de eventos *signals and slots* foi um fator que chamou muito a atenção durante a escolha do framework adequado ao projeto, pois possibilitou uma solução eficiente no que diz respeito ao sincronismo das mensagens e eventos entre o microcontrolador e o eletrômetro, assim como quanto ao processamento dos dados obtidos em cada ensaio. Além disso, o fato de ser baseada em uma linguagem robusta como C++, colaborou em sua escolha, mostrando-se ser um *framework* poderoso, versátil e muito flexível.

3.1 Hardware

A automação do sistema de medição consistiu em substituir o temporizador astável e o acionamento manual tanto do pistão quanto do ar comprimido sobre a amostra pelo Arduino, que passaria a controlar a válvula eletropneumática do ar comprimido que é jogado sobre a amostra e os as duas válvulas eletropneumáticas que movem o pistão verticalmente através de um sistema também pneumático. Além disso, o Arduino passa a ser uma interface entre o computador e o sistema de medição, recebendo e enviando dados de controle e sincronismo. A automação também incluiu a conexão do eletrômetro ao computador e sua comunicação com o mesmo, assim como o desenvolvimento de uma aplicação com interface gráfica no computador

responsável pela comunicação entre o usuário e o sistema assim como o controle e sincronismo do Arduino e do eletrômetro, a leitura dos dados do eletrômetro e processamento desses dados. Depois de automatizado o sistema, o circuito manual antigo de acionamento do pistão e do ar foi colocado em paralelo com o novo circuito contendo o Arduino de tal maneira que, através de uma chave manual, pode-se escolher realizar o acionamento manualmente como se vinha fazendo antes da realização desse trabalho, ou utilizar o sistema aqui desenvolvido, completamente automatizado.

A figura 26 apresenta o esquemático do controle do sistema de medida e a figura 27 a visão geral do sistema desenvolvido e finalizado funcionando do GATM.

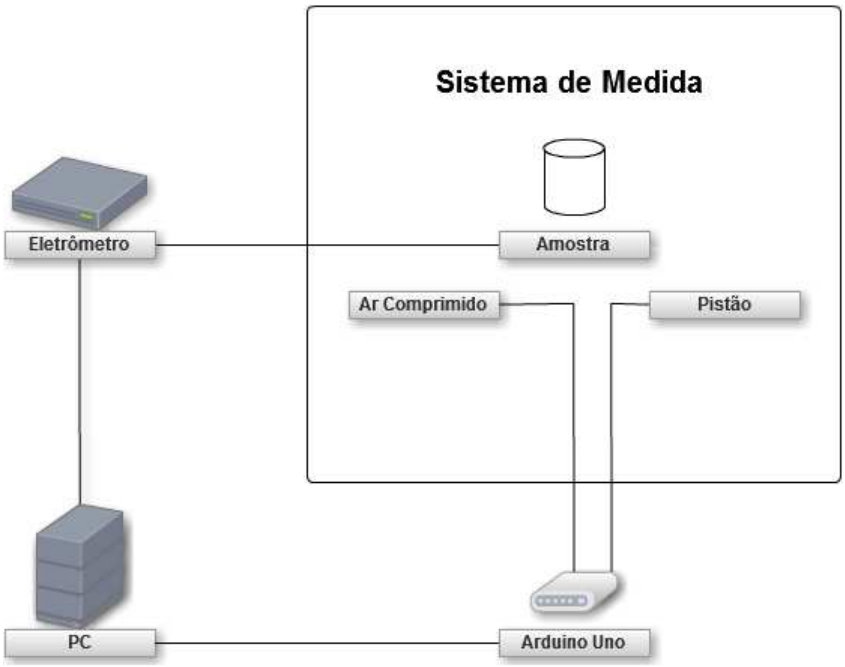

Figura 26: Esquemático do controle do sistema de medida

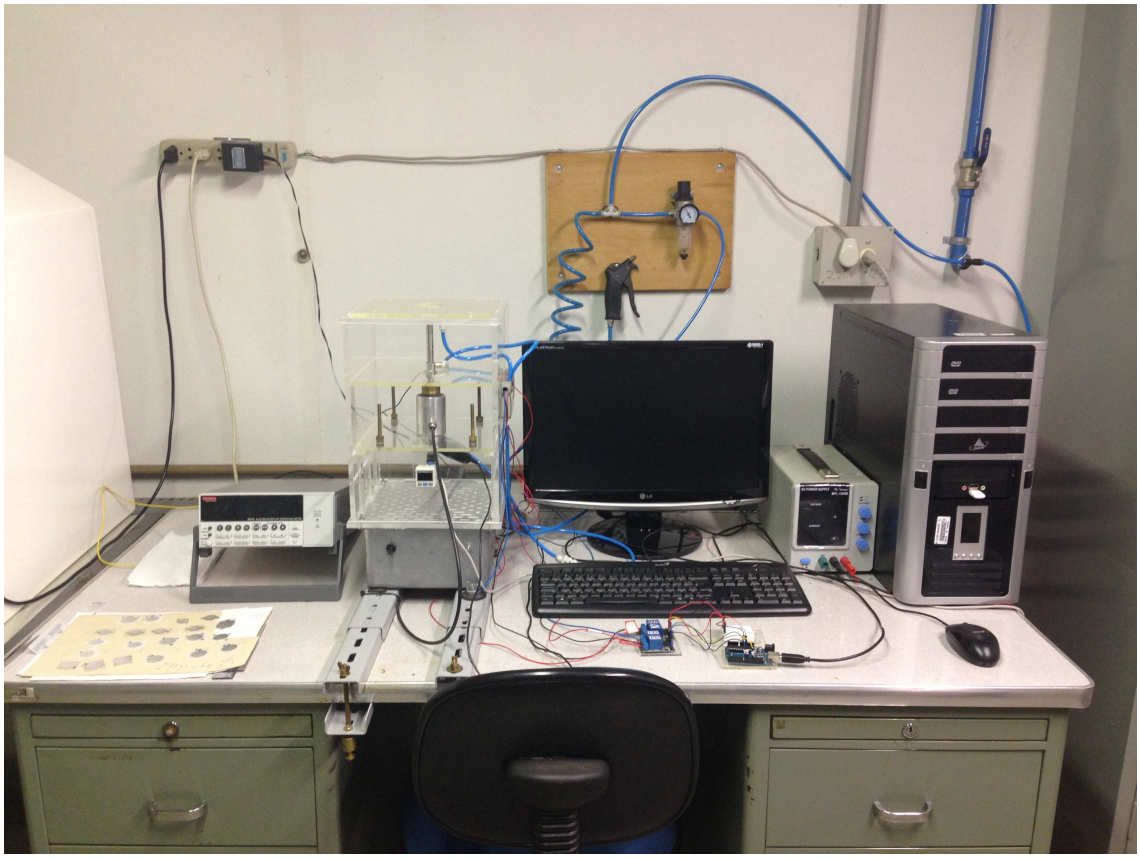

Figura 27: Visão geral do sistema de medidas automatizado e finalizado

O acionamento do pistão é realizado por um sistema pneumático contendo dois canais condutores de ar comprimido, cada um acionado por uma válvula eletropneumática. O primeiro canal está ligado a uma fonte de ar comprimido e ao pistão. Sua função é acionar o mecanismo de movimento do pistão, forçando-o a movimentar-se verticalmente para baixo. O segundo está conectado ao pistão juntamente com o primeiro canal, encontrando-se conectado em sua outra extremidade a uma válvula de escape para o ar de forma a servir para liberar a pressão do ar sobre o pistão. Esses canais, para movimentar o pistão, trabalham de forma complementar. Para descer o pistão, enquanto o primeiro é acionado, jogando ar no pistão e fazendo-o descer sobre a amostra, o segundo encontra-se desativado, mantendo esse segundo duto de ar fechado, impedindo que o ar comprimido escape por ele. Para subir o pistão, o primeiro canal é desativado, suprimindo o fornecimento de ar comprimido, enquanto o segundo é acionado, liberando o ar ainda presente nos dutos, retirando a pressão exercida no pistão, fazendo-o subir verticalmente. A figura 28 mostra o sistema pneumático acoplado ao pistão.

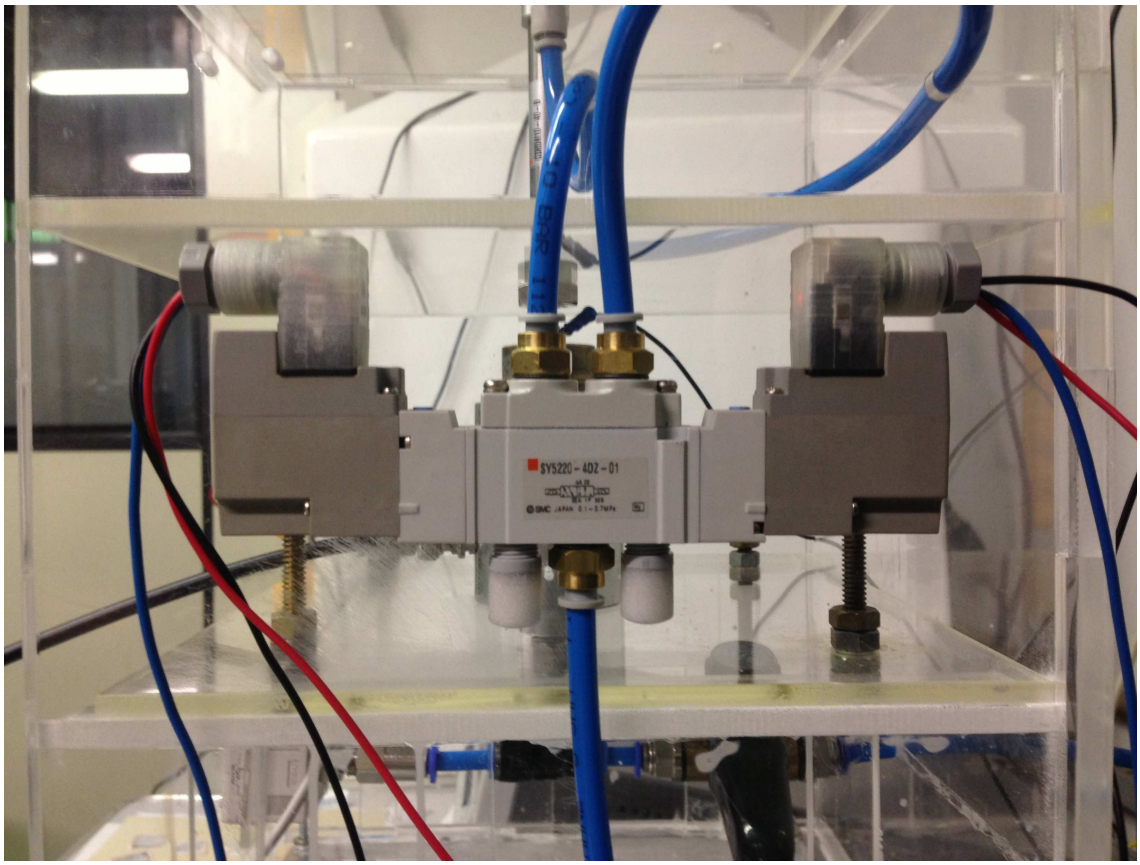

Figura 28: Sistema pneumático de acionamento do pistão

No Arduino, os pinos 8 e 9 foram utilizados para gerar os sinais de controle do pistão. Definidos através do software desenvolvido e carregado no Arduino, eles são acionados com sinais complementares: enquanto um pino possui sinal lógico ALTO, o outro possui sinal lógico BAIXO, e vice-versa. Cada um desses pinos foi conectado a um relé, ativo com sinal lógico BAIXO e alimentado com uma tensão de 5V fornecida pela placa Arduino através do pino 5V. Os relés foram utilizados para isolar o circuito do Arduino, de baixa tensão, das válvulas eletropneumáticas, que operam com uma tensão elevada. O pino 8 foi conectado ao relé com a etiqueta Ch1, que por sua vez está ligado ao duto que injeta ar no pistão. O pino 9 foi conectado ao relé com a etiqueta Ch2, que por sua vez está ligado ao duto que libera o ar no pistão. Ambas as válvulas são alimentados com 12V cada.

O pino 12 do Arduino foi utilizado para gerar o sinal de controle do ar comprimido. Ele foi também conectado a um relé, ativo com sinal lógico BAIXO e alimentado com uma tensão de 5V fornecida pela placa Arduino através do pino 5V. Por sua vez, este relé (com a etiqueta B) foi ligado a válvula eletropneumática responsável pelo acionamento do ar comprimido sobre a amostra através da abertura inferior do suporte da amostra, alimentada com uma tensão de 12V.

A figura 29 mostra os relés utilizados, suas etiquetas e conexões. A figura 30 mostra o Arduino e suas conexões. A figura 31 apresenta as ligações feitas entre o Arduino e os relés.

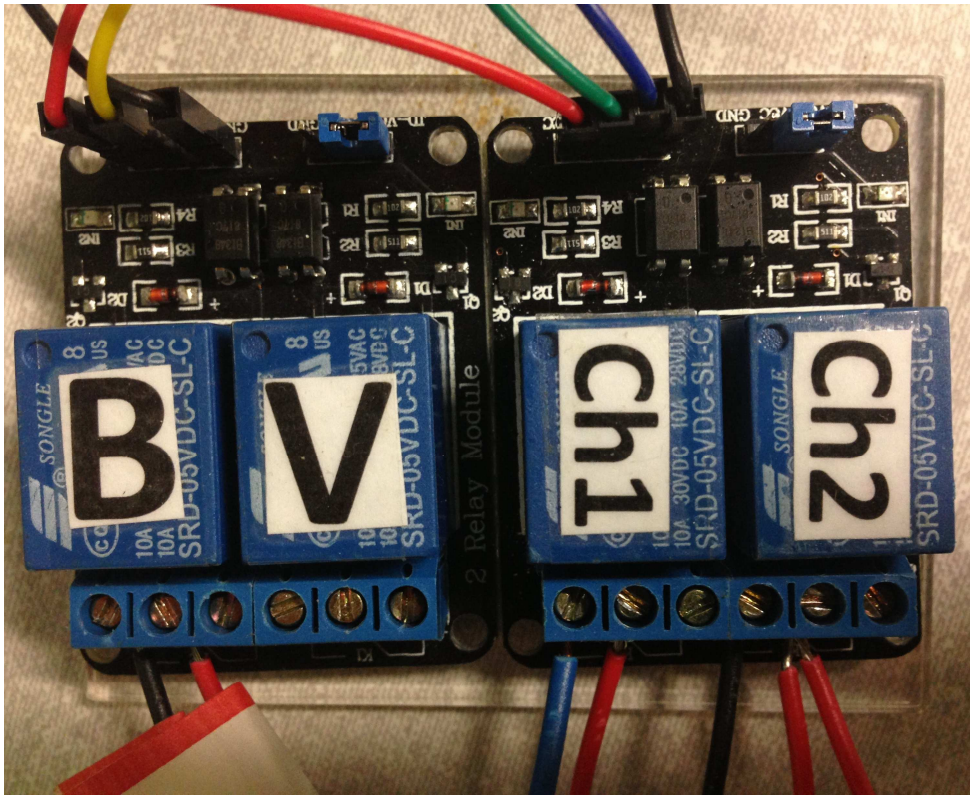

Figura 29: Relés utilizados, suas etiquetas e conexões

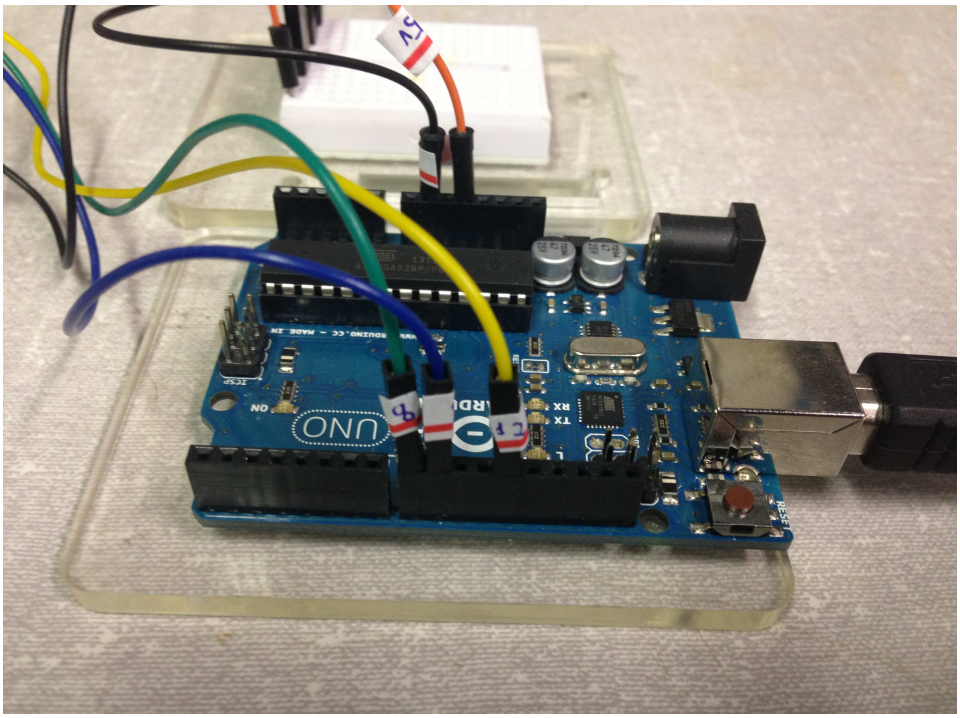

Figura 30: Arduino e suas conexões

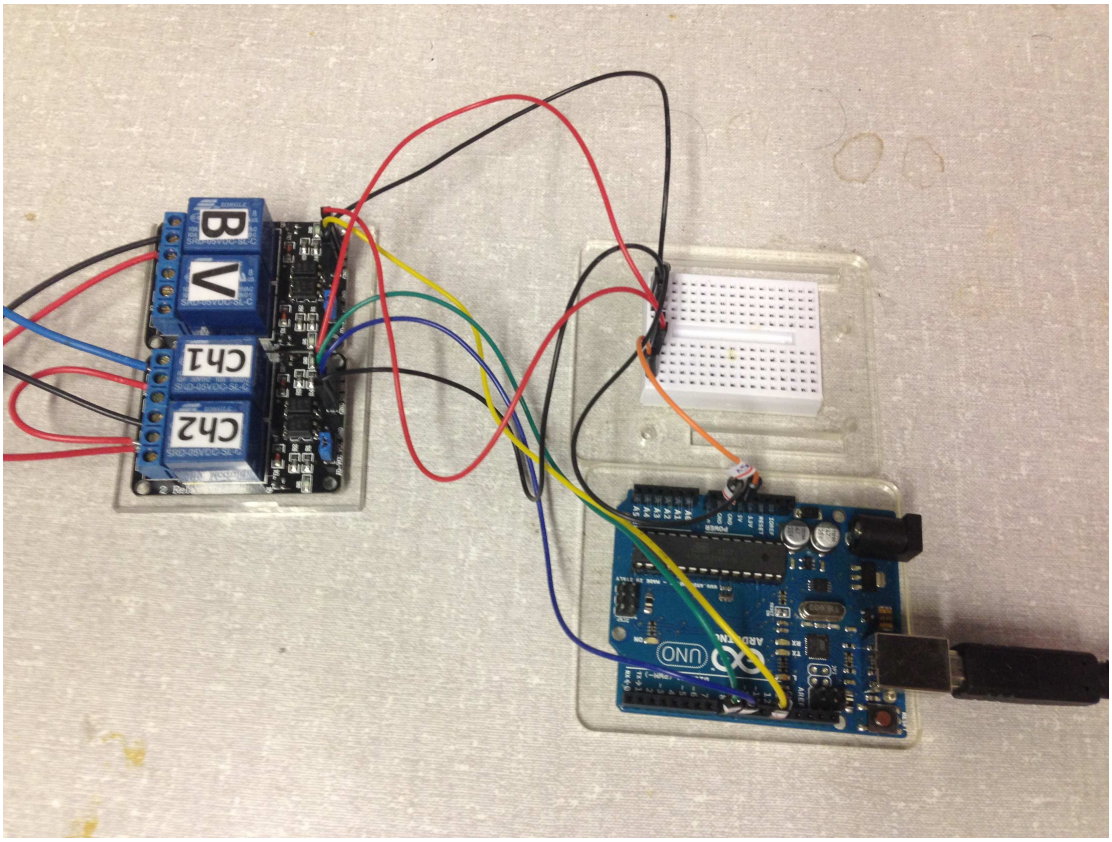

Figura 31: Ligações realizadas entre o Arduino e os relés

Observar que o pino 13 não foi utilizado por possuir ligado a ele um LED, o que poderia afetar o nível do sinal.

Os relés utilizados foram da fabricante Songle, modelo SRD-5VDC-SL-C (voltagem nominal da bobina de 5V DC, estrutura selada, sensibilidade da bobina de 0,36W), cuja especificação encontra-se em [13], podendo ser visto na figura 32.

Figura 32: Módulo de 2 relés Songle SRD-5VDC-SL-C (Fonte: [14])

O eletrômetro utilizado, 6517B da Keithley Instruments, é manualmente ligado. Ele foi conectado ao computador através de sua interface serial (RS-232) utilizando um adaptador serial-USB, sendo os parâmetros da conexão definidos de forma manual diretamente no eletrômetro: BaudRate de 9600 Bd, marcador de final de *string* <CR+LF> e controle de fluxo XON-XOFF. Além disso, a definição da referência para medição das cargas superficiais da amostra é realizada a cada ensaio, manualmente, na escala de nC. A entrada do eletrômetro foi conectada aos eletrodos ligados a amostra, de forma a possibilitar a leitura dos sinais elétricos gerados pela mesma. A figura 9 apresenta o eletrômetro utilizado.

O Arduino, por sua vez, foi conectado ao computador através de seu conector USB. Assim, tanto a alimentação do Arduino quanto a sua comunicação com o computador foi realizada pela conexão USB.

O pressostato utilizado, do fabricante SMC, modelo ISE30, foi conectado a abertura inferior do suporte da amostra para medir a pressão do ar que é jogado sobre a mesma. Ele é alimentado por uma fonte de tensão DC de 13V que pode ser visualizada na figura 33. Como esse modelo de pressostato não possui uma interface que permita a leitura digital dos valores de pressão medidos, não houve a possibilidade de conectá-lo ao computador, de forma que a pressão do ar utilizada deve ser fornecida manualmente a interface gráfica da aplicação desenvolvida. O pressostato utilizado pode ser visto na figura 6 e, conectado ao sistema, pode ser visto na figura 7.

Figura 33: Fonte de tensão DC para alimentação do pressostato.

As especificações do hardware do computador utilizado são:

- Sistema Operacional: Windows 8 Enterprise 32 bits
- Processador: Intel Core 2 Duo CPU E7200 @2,53Ghz (64bits)
- Memória RAM: 2,00 GB
- Placa Mãe: Intel DP35DP
- Placa de Vídeo: NVIDIA GeForce 8500 GT

3.2 Software

Para o controle do sistema de medidas, o que envolve a movimentação do pistão e o acionamento do ar comprimido sobre a amostra, foi desenvolvido um software para o Arduino que realiza esse controle de forma síncrona com a coleta de dados do eletrômetro realizada por uma aplicação no computador. Além disso, esse software carregado na placa microcontroladora é responsável pela comunicação entre o Arduino e o computador.

Além desse software para o Arduino, foi desenvolvida uma aplicação em Qt que tem a finalidade de servir de interface gráfica para o usuário, coletar e processar os dados do eletrômetro de forma sincronizada com o processo de medida, além de comunicar-se com o Arduino.

O código fonte de ambos os softwares desenvolvidos encontra-se disponível no relatório interno do GATM a cerca do sistema desenvolvido [15]

A seguir, tem-se a descrição do funcionamento do software do sistema.

3.2.1 Arduino

Ao ser ligado, o software carregado no microcontrolador do Arduino define os 3 sinais de saída (um para o controle do ar sobre a amostra e outros 2 para controle pneumático do pistão), levantando o pistão, e aguarda a conexão serial do computador ser realizada. Uma vez conectado ao computador, aguarda-se o sinal de início da série de medidas e a quantidade que deverá ser realizada. Recebido o número de medidas a serem realizadas, abaixa-se o pistão, envia um sinal para o computador começar a ler os dados do eletrômetro e liga-se o ar comprimido sobre a amostra, começando assim o ciclo de medidas.

Uma medida consiste em medir a quantidade de carga na amostra durante um intervalo de tempo definido, estando esta comprimida, e, durante igual intervalo de tempo, medir a quantidade de carga na amostra sem pressioná-la. Assim, após o intervalo de tempo estipulado (5 segundos), o Arduino corta o ar injetado sobre a amostra, esperando mais 5s enquanto o computador coleta as informações do eletrômetro, para que no final deste período, envie um sinal ao computador informando o término da medida atual, reiniciando o ciclo.

Ao final da execução do número de medidas recebido do computador, o arduino levanta o pistão, e espera novo sinal para reiniciar outro ciclo de medidas.

Este processo está descrito graficamente no fluxograma da figura 34.

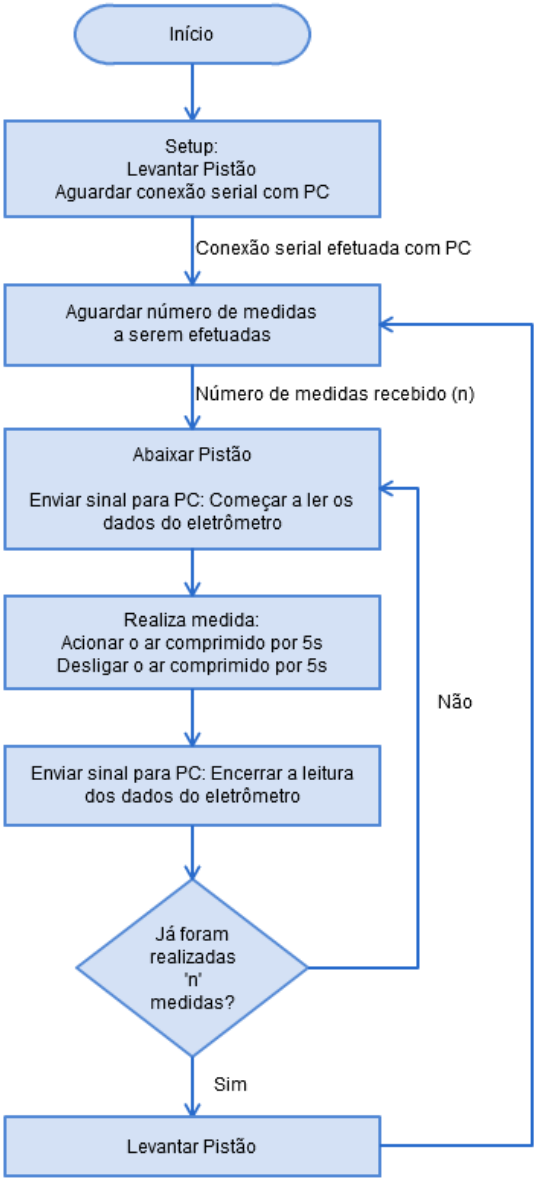

Figura 34: Fluxograma do software executado no Arduino

3.2.2 Computador

Ao iniciar o programa, a interface é apresentada como mostrada na figura 35. Ela permite ao usuário configurar as conexões seriais com o arduino e com o eletrômetro (ambas são independentes uma da outra) assim como conectar e desconectar o computador deles. Uma vez conectado o computador (pelo menos ao arduino) é possível realizar medidas, fornecendo os dados necessários através da interface, sendo possível salvar os dados gerados em arquivo.

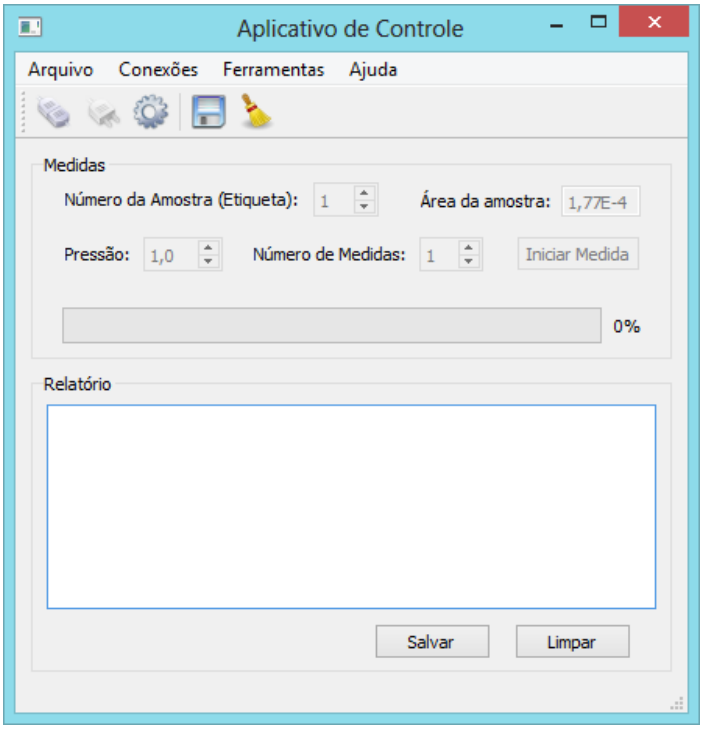

Figura 35: Interface gráfica para interação e controle do sistema

Na figura 36 e 37 é apresentada a tela de configuração das conexões. Nela são listadas todas as portas disponíveis do computador, sendo fornecidas para cada uma as informações a respeito do hardware a elas conectado. Além disso, definem-se especificamente os parâmetros de cada conexão.

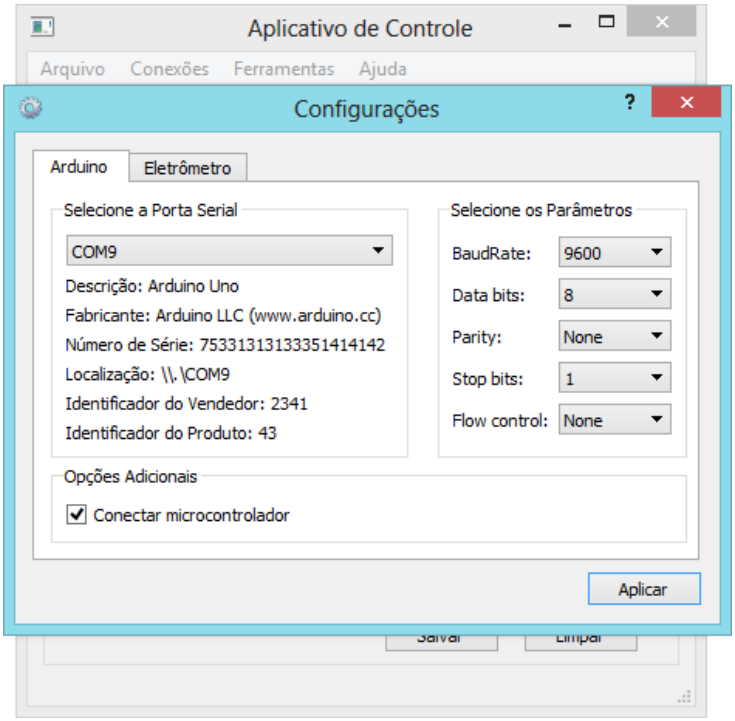

Figura 36: Janela de configuração das conexões com microcontrolador e com eletrômetro – Guia Arduino

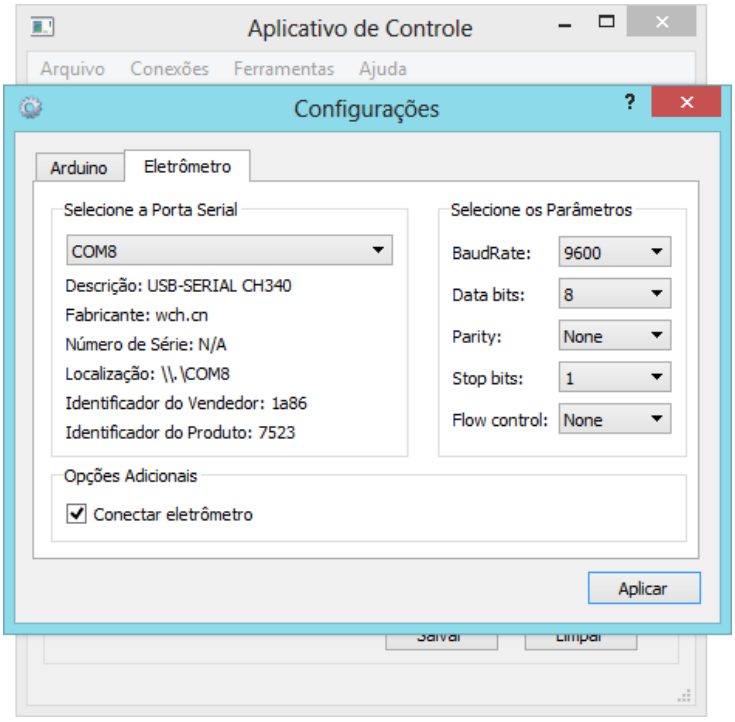

Figura 37: Janela de configuração das conexões com microcontrolador e com eletrômetro – Guia Eletrômetro

Ao clicar em *conectar*, uma vez a conexão estabelecida, é possível editar os parâmetros das medidas a serem realizadas: definição de uma etiqueta para a amostra de piezoeletreto a ter suas propriedades medidas; da pressão aplicada sobre a amostra, e do número de medidas a serem

realizadas no ensaio atual. A área da amostra é fixada fisicamente no projeto do suporte que contém a amostra ($1{,}77.10^{-4}$ m^2).

Uma vez clicado em *Iniciar Medida*, é enviado ao arduino o número de medidas a serem realizadas, aguardando assim o sinal de sincronismo do mesmo para iniciar a leitura dos dados do eletrômetro.

Uma vez recebido esse sinal, o computador envia ao eletrômetro as requisições de leitura, recebendo os dados lidos pelo mesmo, processando e mostrando na parte destinada ao relatório para o usuário os dados coletados, como se pode ver na figura 38.

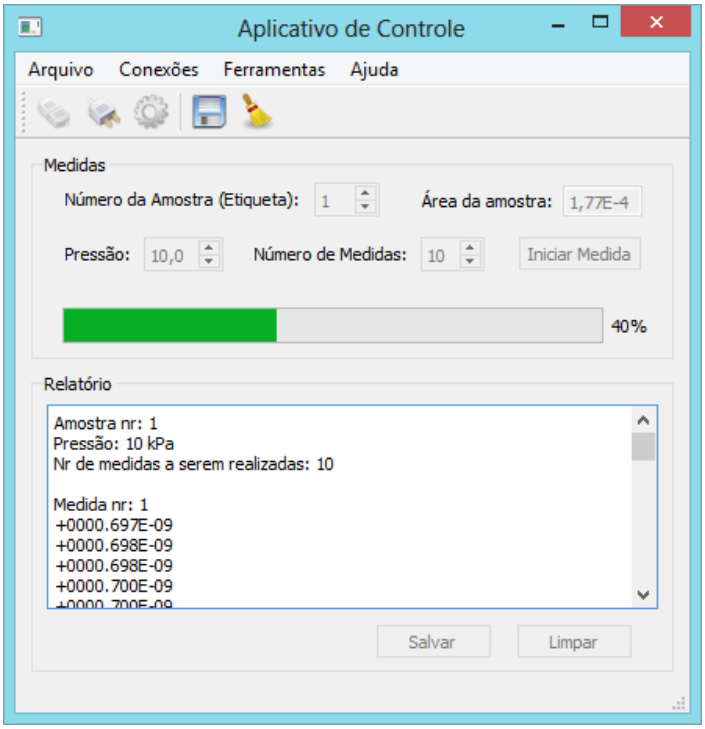

Figura 38: Interface gráfica durante realização de medidas

Terminada a medida atual, recebe-se o sinal do arduino para encerrar a leitura do eletrômetro, aguardando assim o sinal para reiniciar a leitura do mesmo em uma nova medida.

Ao término do número de medidas realizadas sobre a amostra atual, é calculado o coeficiente piezoelétrico e disponibilizado o seu valor ao usuário, na área de relatórios do programa.

O usuário agora tem a possibilidade de salvar o relatório ou executar novo conjunto de medidas.

Ao clicar em *desconectar* ou ao fechar o programa, as conexões com o microcontrolador e com o eletrômetro são encerradas.

Caso o usuário necessite, existe no menu *Ajuda* um guia rápido de utilização, como pode ser visualizado na figura 39.

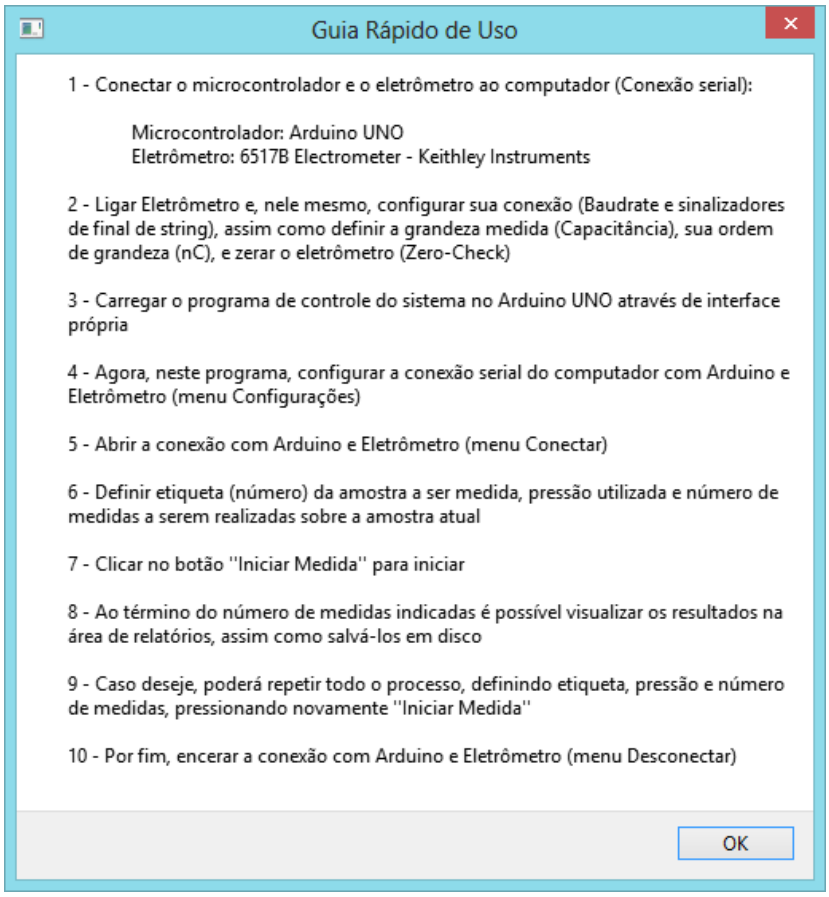

Figura 39: Guia de uso simplificado do sistema

Este processo principal está descrito graficamente no fluxograma da figura 40.

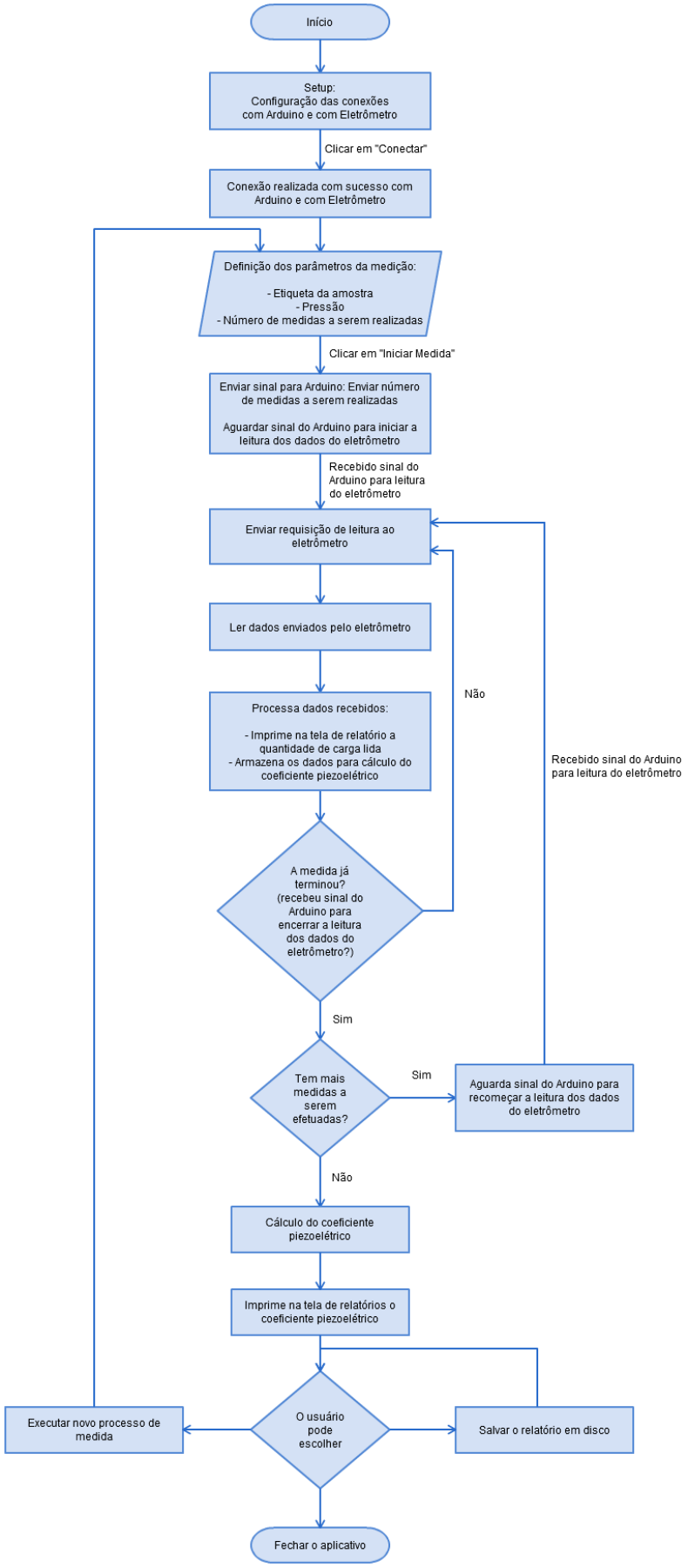

Figura 40: Fluxograma principal do software executado no computador (Interface Qt)

Na tabela 1 pode-se observar alguns resultados de medidas realizadas por este sistema em uma amostra piezoelétrica produzida no GATM. A quantidade de carga superficial, ΔQ, gerada devido à pressão sobre a amostra é calculada pela diferença entre as cargas do piezoeletreto quando sobre pressão externa e na ausência de pressão. Os ensaios descritos na tabela 1 foram realizados sob uma pressão, P, de 10kPa. Assim, para uma área, A, da amostra de $1{,}77.10^{-4} m^2$, a força F aplicada sobre a amostra é: F = P.A. O coeficiente piezoelétrico é calculado como a razão entre o valor médio de ΔQ das medidas realizadas ($\sim\Delta Q$) e a força aplicada. Como pode-se ver da tabela 1, para esse conjunto de medidas $\sim\Delta Q$ vale 0,6989 pC, o que resulta em um coeficiente piezoelétrico de 0,395 pC/N.

Medida	ΔQ (pC)
1	0,673
2	0,688
3	0,690
4	0,703
5	0,705
6	0,705
7	0,707
8	0,706
9	0,705
10	0,707

Tabela 1: Resultados de medidas realizadas pelo sistema desenvolvido

4 Conclusões

A automação do sistema de medidas de piezoeletricidade em piezoeletretos foi, portanto, concluída com êxito, atendendo as necessidades de um sistema com a finalidade do mesmo: controle do pistão, do ar injetado sobre a amostra, coleta e processamento dos dados obtidos, sendo tudo isso apresentado numa interface gráfica única e intuitiva ao usuário. Para futuras implementações, seria útil implementar a definição do tempo de medida pelo usuário através da interface gráfica e a geração de gráficos a partir dos dados obtidos pelo eletrômetro.

Com a automação do sistema, as medidas são agora realizadas de maneira ágil, facilitando o processo de obtenção dos dados e seu processamento. Saber o coeficiente d_{33} de uma amostra requer agora um tempo mínimo, irrisório se comparado ao tempo que o processo de medição e cálculo do coeficiente levava antes.

Durante a realização deste projeto, foi possível perceber a extensão do poder e versatilidade das ferramentas e plataformas utilizadas. O Arduino mostrou-se uma placa controladora simples, flexível, poderosa e de extensa utilidade no controle de sistemas. Da mesma forma, o framework Qt é uma plataforma de desenvolvimento extremamente robusta, possibilitando o desenvolvimento rápido de softwares bem estruturados, funcionais e com uma interface igualmente amigável.

Tem-se, assim, o objetivo deste trabalho atingido de forma satisfatória, atendendo a proposta inicial do mesmo. Este trabalho, contudo, é a base para futuros projetos na área de caracterização de piezoeletretos termo-formados. A automação desse sistema é um passo inicial para o desenvolvimento e expansão das pesquisas realizadas em piezoeletricidade pelo GATM, sendo este trabalho futuramente submetido para um congresso de sistemas de medição.

5 Referências

1. ALTAFIM, R. A. P. **Novos piezoeletretos: desenvolvimento e caracterização**. 2010. Tese de Doutorado - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

2. ALTAFIM, R. A. P. **Análise e implementação de métodos para a caracterização de eletretos termo-formados**. São Carlos, 2005. Dissertação de Mestrado - Escola de Engenharia de São Carlos, Universidade de São Paulo.

3. REZA MOHEIMANI, S. O.; FLEMING, A. J. **Piezoelectric Transducers for Vibration Control and Damping**. Springer-Verlag London, 2006.

4. PIEZOELECTRIC Constant | APC International Ltd. **APC International Ltd**. Disponivel em: <https://www.americanpiezo.com/knowledge-center/piezo-theory/piezoelectric-constants.html>. Acesso em: 01 jun. 2015.

5. TEST & Precision Measurement Tools. **Keithley Instruments Inc.** Disponivel em: <http://www.keithley.com/>. Acesso em: 01 jun. 2015.

6. **Arduino**. Disponivel em: <http://www.arduino.cc/>. Acesso em: 01 jun. 2015.

7. ARDUINO Uno R3 Schematic. **Arduino**. Disponivel em: <https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf>. Acesso em: 01 jun. 2015.

8. ATMEGA48PA/88PA/168PA/328P Datasheet. **Atmel Corporation - Microcontrollers, 32-bit, and touch solutions**. Disponivel em: <http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf>. Acesso em: 01 jun. 2015.

9. ARDUINO Board Uno. **Arduino**. Disponivel em: <https://www.arduino.cc/en/Main/arduinoBoardUno>. Acesso em: 01 jun. 2015.

10. ARDUINO Uno. **Embarcados**. Disponivel em: <http://www.embarcados.com.br/arduino-uno/>. Acesso em: 01 jun. 2015.

11. **Qt | Cross-platform application & UI development framework**. Disponivel em: <http://www.qt.io/>. Acesso em: 01 jun. 2015.

12. DALHEIMER, M. K. **Programming with Qt:** Writing Portable GUI applications on Unix and Win32. 2ª Edição revisada.O'Reilly Media, Inc., 2002.

13. SONGLE Relay Module SRD-5VDC-SL-C Datasheet. **GHI Electronics**. Disponivel em: <https://www.ghielectronics.com/downloads/man/20084141716341001RelayX1.pdf>.

Acesso em: 01 jun. 2015.

14. SONGLE Relay Module SRD-5VDC-SL-C. **Art of Circuits**. Disponivel em: <http://artofcircuits.com/wp-content/uploads/2014/05/2-ch-relay-module-1.jpg>. Acesso em: 01 jun. 2015.

15. SOARES, I. N. **Medidor automatizado de coeficiente piezoelétrico: Código Fonte**. Universidade de São Paulo, Escola de Egenharia de São Carlos. 2015.

6 Apêndice: Esquemático Arduino Uno R3