JULIANO D. BARBOZA

DESENVOLVIMENTO DE UMA PLATAFORMA DE TESTES PARA DISPOSITIVOS MICROFLUÍDICOS COM ÊNFASE EM APLICAÇÕES BIOMÉDICAS

Texto apresentado à Escola Politécnica da Universidade de São Paulo como requisito para a conclusão do curso de graduação em Engenharia Mecatrônica, junto ao Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

São Paulo
2012

JULIANO D. BARBOZA

DESENVOLVIMENTO DE UMA PLATAFORMA DE TESTES PARA DISPOSITIVOS MICROFLUÍDICOS COM ÊNFASE EM APLICAÇÕES BIOMÉDICAS

Texto apresentado à Escola Politécnica da Universidade de São Paulo como requisito para a conclusão do curso de graduação em Engenharia Mecatrônica, junto ao Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

Área de Concentração:

Engenharia Mecatrônica

Orientador:

Prof. Dr. Ricardo Cury Ibrahim

São Paulo
2012

FICHA CATALOGRÁFICA

Barboza, Juliano Dawid

Desenvolvimento de uma plataforma de testes para dispositivos microfluídicos com ênfase em aplicações biomédicas/ J. D. Barboza. –São Paulo, 2012.

169 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Mecatrônica 2. Tecnologias da saúde 3. Mecânica dos fluídos I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas MecânicosII. t.

AGRADECIMENTOS

Agradeço a minha família por sempre me apoiar em meus empreendimentos, aos meus amigos por todos os momentos de diversão que me revigoraram para seguir lutando e ao meu amor, por me ensinar a sonhar.

RESUMO

Aparelhos de diagnóstico portáteis estão revolucionando o universo dos cuidados com a saúde. Se antes era necessário um longo e tedioso processo de retirada de amostras, espera para que fossem processadas e demora para que as informações chegassem ao prestador de saúde, agora é possível que o próprio cuidador, com apenas uma gota de sangue, seja capaz de obter informações importantes em apenas alguns minutos ou segundos. E em relação a saúde, quanto mais rápido se tem informações, mais vidas podem ser salvas. As tecnologias dos diagnósticos portáteis, bem como outros avanços que permitem a detecção e, tal vez, o tratamento de enfermidades o mais cedo possível, de forma decentralizada e próxima ao paciente são as chamadas *point-of-care technologies* (POCT) .

Dentro desse contexto, o presente trabalho propôs desenvolver uma plataforma de testes para dispositivos de diagnóstico portáteis. Essa plataforma pode ser utilizada no desenvolvimento de dispositivos dessa natureza, nas validações e testes após o desenvolvimento e até em pesquisa e comparação entre tecnologias ou soluções. Como a microfluídica é a tecnologia de escolha para portabilidade em POCT, uma plataforma de testes para aparelhos de diagnóstico portáteis deve também ser aplicável a outros dispositivos microfluídicos.

ABSTRACT

Portable diagnostic devices are changing the universe of healthcare. If a long and tedious process of sampling, analysis and delivery of information was once necessary, now it can be done almost instantly by the caregiver, at the point of contact with the patient. And in terms of health, quicker information gathering means more lifes saved. The technologies of portable diagnostic devices, as well as other advances that enable the earlier detection and even an earlier treatment of diseases, in a decentralized manner and as near the patient as posible are called point-of-care technologies (POCT) .

In this context, this work presents the project, construction and validation of a testing platform for portable diagnostic devices. This platform can be used in the stage of development, validation, tests and even research on new technologies and solutions. As microfluidics is the enabling technology for portability in POCT, a testing platform for portable diagnostic devices can also be used for other microfluidic devices.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

USP	Universidade de São Paulo
NIBIB	*National Institute of Biomedical Imaging and Bioengineering*
DID	*Data Item Descriptions*
CMMI	*Capability Maturity Model Integration*
PDCA	*Plan - Do - Check - Act*
TPS	*Toyota Production System*
PDMS	Polidimetilsiloxano
POCT	*Point-of-care Technologies*
PCR	*Polimerase Chain Reaction*
ELISA	*Enzyme-Linked Immunoabsorbent Assay*
MDF	placa de fibra de madeira de média densidade
PWM	*Pulse-width-modulation*
CI	Circuito Integrado
FTDI	*Future Technology Devices International Ltd.*
UART	*Universal asynchronous receiver/transmitter*
TTL	*TTL compatible logical levels*
IDE	*Integrated Development Environment*
MER	Modelo Entidade Relacionamento
PI	Proporcional Integrativo

LISTA DE SÍMBOLOS

Kn	Número de Knudsen
ρ	Massa específica
t	Tempo
v	Velocidade
p	Pressão
T_c	Forças de cisalhamento
f	Forças de campo por unidade de massa
e	Energia específica
Q	Calor transferido
μ	Coeficiente de atrito dinâmico
Re	Número de Reynolds
σ_{lg}	Tensão superficial na interface líquido/gás
θ	Ângulo de contato com a parede do canal
r_0	Raio do canal
L_e	Comprimento de entrada
T	Temperatura
Ra_L	Número de Rayleigh
$\overline{h}$	Coeficiente de transferência de calor médio
$\dot{Q}$	Fluxo de calor

g	Aceleração da gravidade
Nu_L	Número de Nusselt
ω_c	Frequencia de corte
R	Resistência
C	Capacitância
ϕ	Fluxo volumétrico
Kn	Número de Knudsen
ω_n	Velocidade de rotação
τ	Constante de tempo em sistema dinâmico de primeira ordem
K	Ganho de um sistema dinâmico

1 INTRODUÇÃO

1.1 Apresentação

A maneira como se pensa em assistência médica está sofrendo uma revolução. Existe um grande desenvolvimento em torno do conceito "point-of--care", em que o cuidado a um paciente deve ser dado o mais próximo a ele possível, e o mais cedo possível. Uma pesquisa no site google.com pelo termo "point-of-care"retornou em torno de 971 milhões de resultados (em junho de 2012), sendo os mais notáveis empresas e desenvolvimentos tecnológicos. De acordo com o Point-of-Care Technologies Research Network (POCTRN) criado pelo The National Institute of Biomedical Imaging and Bioengineering (NIBIB) nos Estados Unidos, "Testes rápidos e exatos no ponto de contato inicial com o paciente irão melhorar a assistência médica e diminuir custos. Tecnologias Point-of-care podem aumentar acesso à medicina moderna e permitir diagnósticos e tratamentos mais precoces"(tradução nossa).

Um produto muito importante para o modelo point-of-care é o dispositivo de diagnóstico portátil. Com este, é possível realizar diagnósticos fora de um laboratório, o que representa um enorme avanço. Pode-se, por exemplo, combater doenças em locais pobres ou inacessíveis, levando o dispositivo às pessoas ao invés de deslocá-las a algum centro de diagnóstico, ou ter que retirar amostras e enviar a pessoal especializado. Outras aplicações são a melhora

do tempo de resposta em casos clínicos críticos e a diminuição de tempo até o tratamento efetivo em unidades de saúde convencionais. Atualmente existem diversos aparelhos de diagnóstico portáteis que usam tecnologias microfluídicas, ou seja, lidam com volumes muito pequenos de fluidos para garantir sua portabilidade. Foram encontradas centenas de empresas que trabalham com essa tecnologia. No Brasil, no entanto, não foi encontrada nenhuma aplicação comercial desenvolvida nacionalmente.

Uma etapa crítiica no desenvolvimento de produtos é a de testes e verificações. No ramo da microfluídica, isso se torna mais crítico, uma vez que as métricas são específicas da área, envolvendo, por exemplo, mensuração de volumes na escala de picolitros, ou fluxos de microlitros por segundo. Assim, a produção nacional de dispositivos microfluídicos, inclusive equipamentos de diagnóstico portáteis, possui a barreira da falta de ferramentas para lidar com esse tipo de medida. Nesse contexto, o presente trabalho apresenta o desenvolvimento de uma plataforma de testes para dispositivos microfluídicos que possa preencher essa necessidade. Seus usos pretendidos são na fase de desenvolvimento de aparelhos diagnósticos portáteis, na fase de validação em relação aos requisitos, em pesquisas dentro da área microfluídica ou que utilizem microfluídica e como ferramenta de comparação entre aparelhos microfluídicos comerciais.

1.2 Objetivos

O objetivo do presente trabalho é apresentar o projeto e a construção de uma plataforma de testes adequada para microfluídica, com especial ênfase em aparelhos de microfluídica biomédicos, como os de diagnóstico portáteis. Assim, foi projetada e construída uma bancada de testes cuja caracterís-

tica mais importante é manipular fluxos e pressões pertinentes ao desenvolvimento em microfluídica. A plataforma também contém sensores de temperatura e pressão que atuam na faixa e precisão pertinentes a essa aplicação.

Para testar a bancada de testes, foi utilizado um protótipo fictício de aparelho de diagnóstico microfluídico. É de se esperar que a bancada obtida também seja útil em outras aplicações microfluídicas. Foi documentada a validade da bancada através do teste de um dispositivo fictício construído em PDMS.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Revisão Bibliográfica

A etapa de testes no desenvolvimento de um produto é essencial. Desde as normas militares norteamericanas, como a MIL-STD-498 isso é requerido no ciclo de projeto. Esta norma é dividida em 22 DIDs (Data Item Descriptions) e 3 delas formam a área denominada *Qualification/test products*. Com o tempo, outras normas surgiram na área de gestão de projetos, aliadas ao que passou a ser chamado de qualidade. Algumas das mais conhecidas são PMBOK, ISO9000 e CMMI. Todas fazem menção a alguma forma de métrica ou verificação de requisitos muitas vezes quantitativos. Mais recentemente, os sistemas de produção *Lean* ou TPS estão em harmonia com o ciclo PDCA (*plan do check act*) (ROTHER, 2010), em que a parte *"check"* está intimamente ligada a métricas e testes, e suas comparações com as expectativas da etapa de planejamento (*Plan*).

Dessa forma, testes são importantes no projeto de qualquer produto. Na área microfluídica, existem artigos que exploram os testes automáticos em aparelhos microfluídcios digitais, ou seja, na chamada *Microelectronic fluidic* (KERKHOFF, 2007), (KERKHOFF et al., 2005). Todavia, na microfluídica convencional, ou "analógica", testes são realizados reunindo-se componentes necessários para cada verificação, como por exemplo um detector de fótons,

uma bomba de seringa programável e um software de análise (LOK; KWOK; NGUYEN, 2012), uma bomba de seringa e um regulador de pressao (SHAEGH; NGUYEN; CHAN, 2012), uma microválvula regulada por pressão, software Lab-View e uma bomba de seringa no modo de extração (KIM et al., 2011) e um microscópio invertido, uma câmera e um software em Matlab para análise de imagens (PHAN et al., 2012) . Como pode-se identificar alguns elementos comuns ou frequentes nesses testes, a construção de uma plataforma de testes que reune tais elementos e, por tanto, pode ser utilizada em diversos experimentos se justifica.

2.2 Microfluídica

No intuito de se desenvolver uma plataforma que teste dispositivos microfluídicos é vital compreender o que é microfluídica, quais suas aplicações mais comuns e as tendências para o futuro.

2.2.1 Definição

Ao contrário do que se pode imaginar, um dispositivo de microfluídica não necessariamente tem dimensões reduzidas. Isso porque a definição de microfluídica é a ciência que lida com o comportamento e manipulação de fluídos em uma escala onde efeitos diferentes dos da escala macro são observados (NGUYEN, 2002). Nesse caso, o dispositivo que comporta o fluído não necessita ser pequeno, apenas manipular um volume diminuto, ou um escoamento com dimensões típicas reduzidas.

As vantagens de se trabalhar com fluidos na escala micro são: melhor tempo de resposta, maior segurança, menor consumo de energia, menor

volume morto, menor fadiga do sistema, menos dejetos, menor uso de reagentes, maior simplicidade, menor custo, maior sensibilidade na detecção de compostos, tamanho do dispositivos da mesma ordem de grandeza que as estruturas utilizadas, escalabilidade e portabilidade (GRAVESEN; BRANEBJERG; JENSEN, 1993) (CHANG; NAGEL; ZAGHLOUL, 2008) (GRAYSON et al., 2004).

2.2.2 Hipótese do Contínuo

Uma das hipóteses fundamentais da mecânica dos fluídos convencional é a hipótese do contínuo. Essa hipótese enuncia que um fluído pode ser tratado como um meio contínuo, ou seja, suas propriedades (como densidade, velocidade e pressão) estão bem definidas em todos os pontos do espaço e variam continuamente de ponto a ponto (NGUYEN, 2002). Quando se trabalha com uma escala reduzida, já não é tão trivial a validade dessa hipótese, uma vez que o fluído é composto por unidades discretas: moléculas. No caso de gases, o número adimensional de Knudsen (Kn) ajuda a avaliar se essa hipótese continua válida, porém não há equivalente para líquidos. Felizmente, segundo Nguyen, é de se esperar que a hipótese do contínuo seja válida para a maioria das aplicações de microfluídica. Entre seus argumentos está uma discussão de que é provável que a continuidade não seja uma boa aproximação apenas para dimensões típicas menores que 10 nm.

Como a mecânica dos fluídos tradicional utiliza a hipótese do contínuo, o fato de essa hipótese continuar válida em dispositivos de microfluídica significa que podem ser usados os resultados teóricos da primeira. Assim, pode-se basear o estudo de microfluídica nas mesmas equações regentes que as da mecânica dos fluídos: as equações de conservação de massa, energia e quantidade de movimento linear (equações de Navier-Stokes).

2.2.3 Equações Regentes

As equações regentes na mecânica dos fluídos contínua representam a conservação de três grandezas físicas: massa, quantidade de movimento linear e energia. Na forma apresentada nas equações (2.1), (2.2) e (2.3), ρ representa a massa específica do fluído, p a pressão, t o tempo, v a velocidade, T_c as forças de cisalhamento, f as forças de campo por unidade de massa, e a energia específica, Q o calor transferido. O símbolo $\frac{\partial A}{\partial B}$ representa a derivada parcial de A em relação a B, ∇ o operador del (ou seja ∇f é o gradiente de f, $\nabla \cdot f$ é o divergente de f e $\nabla \times f$ é o rotacional de f) e $\frac{D}{Dt}()$ a derivada material ($\frac{D}{Dt}(f) = \frac{\partial f}{\partial t} + v \cdot \nabla f$, v é a velocidade do ponto). Sua solução fornece uma descrição completa de um escoamento. No entanto, essas equações não constituem um sistema fechado, e são necessárias equações constitutivas ou simplificações para se obter soluções. Além, é claro, de condições de contorno e condições iniciais.

$$\frac{\partial \rho}{\partial t} + \nabla(\rho v) = 0 \tag{2.1}$$

$$\rho \frac{Dv}{Dt} = -\nabla p + \nabla T_c + \rho f \tag{2.2}$$

$$\frac{D}{Dt}(\rho e) = -p\nabla v + T_c \nabla v + \nabla Q \tag{2.3}$$

2.2.4 Condições de Contorno

Quando se estuda a interface entre um fluido escoando e suas fronteiras, é comum a utilização da hipótese do não escorregamento (no-slip condition) (NGUYEN, 2002), apesar de haver hipóteses diferentes (LAUGA; BRENNER; STONE, 2007). Além disso, também se utilizam hipóteses semelhantes para ou-

tras grandezas físicas, como a condição de variação contínua da temperatura. Por causa das dimensões próximas às distâncias moleculares, é possível que essa hipótese não seja mais válida; que exista uma diferença na velocidade da camada de moléculas mais próxima do contorno e a velocidade deste. Felizmente, na maioria das aplicações na escala micro, a condição de não escorregamento deve ser uma boa aproximação (NGUYEN, 2002). Uma exceção importante é o caso de um líquido movendo-se em um canal hidrofílico por capilaridade, onde o ponto de contato entre o líquido, o canal e a atmosfera deve mover-se relativamente ao canal. Para completar as condições de contorno nesse caso é preciso utilizar o conceito de tensão superficial, discutido mais adiante.

2.2.5 O Número de Reynolds em Escoamentos Microfluídicos

Na forma adimensional da equação de conservação de momentum linear, assumindo um fluído newtoniano e isotrópico, expressa na equação (2.4), surge um número adimensional importante no estudo de fluídos: o número de Reynolds, definido na equação (2.5). Nessas expressões, D representa uma dimensão típica do escoamento (o diâmetro de um canal, por exemplo), v uma velocidade representativa do escoamento (velocidade média, em geral), μ o coeficiente de atrito dinâmico, ρ a massa específica do fluído, f as forças de campo por unidade de massa (como a gravidade, por exemplo) e Re o número de Reynolds. As grandezas com um asterísco estão adimensionalizadas da

seguinte forma:

$$
\begin{aligned}
v^* &= \frac{(\text{velocidade do ponto})}{v} \\
t^* &= \frac{D(\text{tempo})}{v} \\
x^* &= \frac{(\text{posição } x \text{ do ponto})}{D} \\
p^* &= \frac{D(\text{pressão do ponto})}{\mu v}
\end{aligned}
$$

$$
\frac{\rho v D}{\mu}\left(\frac{\partial v^*}{\partial t^*} + v^* \frac{\partial v^*}{\partial x^*} - \frac{fD}{v^2}\right) = \frac{-\partial p^*}{\partial x^*} + \frac{\partial^2 v^*}{\partial x^{*2}} \tag{2.4}
$$

$$
Re = \frac{\rho v D}{\mu} \tag{2.5}
$$

O número de Reynolds está associado à relação entre as forças inerciais e a forças viscosas em um escoamento, e é importante em determinar se este se dá em regime laminar ou turbulento. Usualmente a transição entre os dois regimes ocorre para números de Reynolds entre 1000 e 2000 (NGUYEN, 2002), sendo que abaixo deste valor ocorre um escoamento laminar e acima turbulento.

Aplicações típicas de microfluídica possuem canais com diâmetro na ordem de centenas de micrômetros (10^{-4} m) e velocidade de até alguns decímetros por segundo ($0,1$ m/s). A água, mesmo não sendo o fluído de trabalho do dispositivo, em geral está presente em grande quantidade e pode-se, assim, utilizar seus parâmetros para estimar o número de Reynolds. A densidade da água é da ordem de 1000 kg / m^3, e sua viscosidade dinâmica em torno de 10^{-3} $Pa \cdot s$ a 20 ºC . Assim, o número de Reynolds em aplicações microfluídicas será tipicamente da ordem de 10.

Com número de Reynolds abaixo da transição de escoamento laminar para turbulento, dispositivos microfluídicos devem lidar com escoamentos laminares. Além disso, se o número de Reynolds for pequeno o suficiente (em torno de 1 ou menos), pode-se simplificar as equações de conservação tornando-as lineares e, por tanto, mais simples de serem resolvidas. As equações de conservação simplificadas estão expressas nas equações (2.6) e (2.7), representando conservação de massa e momentum linear, respectivamente. Os índices i e j representam a direção em que está sendo feito o cálculo, por exemplo $i = 1$ representa a direção x do problema, então v_1 é a velocidade de um ponto na direção x, e assim por diante.

$$\frac{\partial v_i^*}{\partial x_i^*} = 0 \tag{2.6}$$

$$\frac{\partial p^*}{\partial x_i^*} + \frac{\partial^2 v_i^*}{\partial x_j^{*2}} = 0 \tag{2.7}$$

2.2.6 Comprimento de Entrada

Ao estudar o escoamento em um canal, pode-se supor que as propriedades do escoamento serão constantes em todas as seções do canal, já que as características de todas essas seções transversais são as mesmas. As propriedades encontradas dessa forma são ditas "em regime". O que ocorre na prática, no entanto, é que ao entrar em um canal ou passar por uma variação de seção transversal, o escoamento não atinge os valores de regime imediatamente, mas deve desenvolver-se gradualmente. A distância entre a entrada de um canal e o ponto em que os valores em regime são alcançados é chamada de comprimento de entrada.

O comprimento de entrada pode ser previsto pela equação (2.8) (FOX; MCDONALD, 1999), onde L_e representa o comprimento de entrada, D uma dimensão representativa do escoamento e Re o número de Reynolds.

$$\frac{L_e}{D} = 0,05Re \tag{2.8}$$

No entanto, para números de Reynolds pequenos a expressão da equação (2.9) se mostra mais adequada (SHAH; LONDON, 1978).

$$\frac{L_e}{D} = \frac{0,6}{1 + 0,035Re} + 0,056Re \tag{2.9}$$

Além disso, em canais com geometria plana (comprimento com ordens de grandeza a mais que altura) o comprimento de entrada deve ser ainda menor que o previsto pela equação (2.9) (em torno de metade), já que na direção maior as propriedades se desenvolvem mais rapidamente (NGUYEN, 2002).

2.2.7 Tensão Superficial

A tensão superficial é um fenômeno que ocorre na interface entre um líquido e outro meio, como um gás ou um sólido. Pela diferença de espaçamento no líquido e no gás, a força atrativa em direção ao líquido deve ser maior que em direção contrária, criando uma tendência do líquido a formar gotas (NGUYEN, 2002). Já a interação com o sólido é difícil de se prever, mas pode ocorrer a atração do líquido em direção ao sólido (diz-se que a superfície do sólido é hidrofílica nesse caso) ou a repulsão (diz-se que a superfície do sólido é hidrofóbica nesse caso).

Em um canal pequeno, o efeito da tensão próximo às paredes deste se tornam mais relevantes. A diferença de pressão causada pela tensão superfi-

cial em um canal pode ser prevista pela equação (2.10), onde Δp é a queda de pressão, σ_{lg} é a tensão superficial na interface líquido/gás , θ é o ângulo de contato com a parede do canal e r_0 é o raio do canal.

$$\Delta p = \frac{2\sigma_{lg}\cos(\theta)}{r_0} \tag{2.10}$$

2.2.8 Exemplos de Equacionamento

2.2.8.1 Solução das Equações de Navier-Stokes para Canais Longos de Sessão Circular

Em um escoamento que ocorre em um canal longo, pode-se supor que a velocidade se dará puramente na direção axial, e que essa velocidade somente depende da posição em relação às paredes do canal (direções radiais). Nessas condições, as equações (2.1) e (2.2) se simplificam para as equações (2.11) e (2.12). A direção axial adotada é x e y e z são as direções radiais.

$$\frac{\partial p}{\partial y} = \frac{\partial p}{\partial z} = 0 \tag{2.11}$$

$$\frac{-\partial p}{\partial x} + \mu\left(\frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) = 0 \tag{2.12}$$

Existem soluções para essas equações e diversas condições de contorno (geometrias da sessão transversal). Para o caso da sessão circular, a solução é dada pela equação (2.13). Pode-se rearranjar a solução na forma da equação (2.14). Nessas equações, $\dot{Q}$ é o fluxo volumétrico na sessao do canal, r_0 é o raio do canal, L seu comprimento, d seu diâmetro, x a direção axial do canal e as outros grandezas como definidas anteriormente.

$$\dot{Q} = \frac{\pi r_0^4}{8\mu}\left(\frac{-dp}{dx}\right) \tag{2.13}$$

$$\Delta p = f\frac{L}{d}\frac{\rho v^2}{2} = 32\frac{\mu L v}{d^2}, \text{ onde } f = \frac{64}{Re} \quad (2.14)$$

2.2.8.2 Aproximação para a Solução das Equações de Navier-Stokes para Canais Longos de Sessão Qualquer

Apesar de haver soluções analíticas para diversas geometrias, normalmente obtém-se uma aproximação boa utilizando-se a solução para canal circular e o conceito de diâmetro hidráulico. O diâmetro hidráulico é definido pela equação (2.15), onde A representa a área da sessão transversal e P_{wet} o perímetro em contato com o fluído. O diâmetro hidráulico estabelece uma equivalência entre canais de geometria diversa e um canal circular com diâmetro igual ao diâmetro hidráulico.

$$D_h = \frac{4A}{P_{wet}} \quad (2.15)$$

Com as técnicas de fabricação de litografia, litografia macia e micro manufatura normalmente se obtém canais com sessão transversal aproximadamente retangular. Para esse tipo de geometria, o diâmetro hidráulico é, em função da altura (a) e largura (b) da sessão transversal, dado pela equação (2.16).

$$D_h = \frac{4(ab)}{2(a+b)} = \frac{2ab}{a+b} \quad (2.16)$$

Substituindo-se esse valor na expressão (2.14), obtém-se que a queda de pressão Δp é dada por pela equação (2.17).

$$\Delta p = 32\frac{\mu L v}{(\frac{2ab}{a+b})^2} \quad (2.17)$$

Figura 1: Distribuição geográfica de empresas nas áreas de lab-on-a-chip, microfluídica e biomems

2.2.9 Dispositivos Microfluídicos Comerciais

O site fluidicmems.com mantém uma lista de empresas representativas de tecnologia lab-on-a-chip, microfluídica e biomems. Em 6 de Junho de 2012 a lista abrangia 238 companhias. Também disponibilizam um mapa com a distribuição geográfica dessas empresas, reproduzido na figura 1. Como pode-se observar, a maioria das empresas encontra-se na América do Norte e na Europa, e não há nenhuma na América do Sul.

As áreas de atuação dessas empresas podem ser resumidas a: diagnósticos point-of-care, terapias (administração de drogas) e pesquisa (principalmente em biologia e estudos de drogas).

3 DESENVOLVIMENTO DE UMA PLATAFORMA DE TESTES

Uma etapa muito importante no desenvolvimento de um produto é a de testes e verificações. É nela que se assegura que os requisitos de um dispositivo foram atendidos, e também que fornece informações para pontos críticos na melhoria.

Nesse sentido, dispositivos de Microfluídica têm requisitos específicos e, por tanto, demandam medidas adequadas, com valores, em alguns casos, diferentes de outras aplicações fluídicas. Além disso, mais de uma atuação e medida são necessárias ao mesmo tempo. É adequada, por tanto, a construção de uma bancada de testes que possibilite o controle de todos esses parâmetros.

No contexto desse trabalho, tal bancada também será útil para validar as simplificações e hipóteses utilizadas.

Outra vantagem da bancada de testes é que esta poderá ser reaproveitada em projetos futuros de microfluídica. Seu projeto levará em conta a possibilidade de que seja melhorada ou modificada para incluir novas formas de atuação e/ou sensoriamento, ou mesmo aumentar o número de atuadores e sensores.

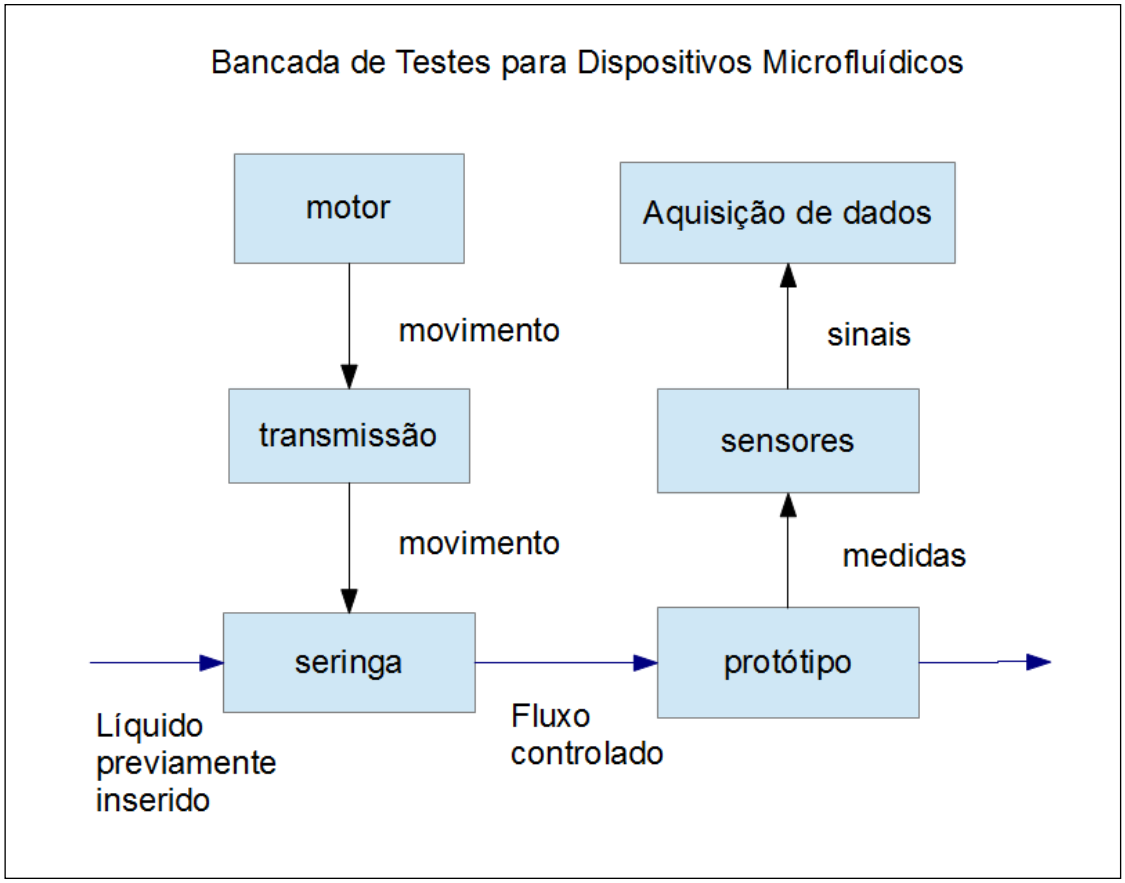

Figura 2: Desenho conceitual da bancada de testes para dispositivos de microfluídica.

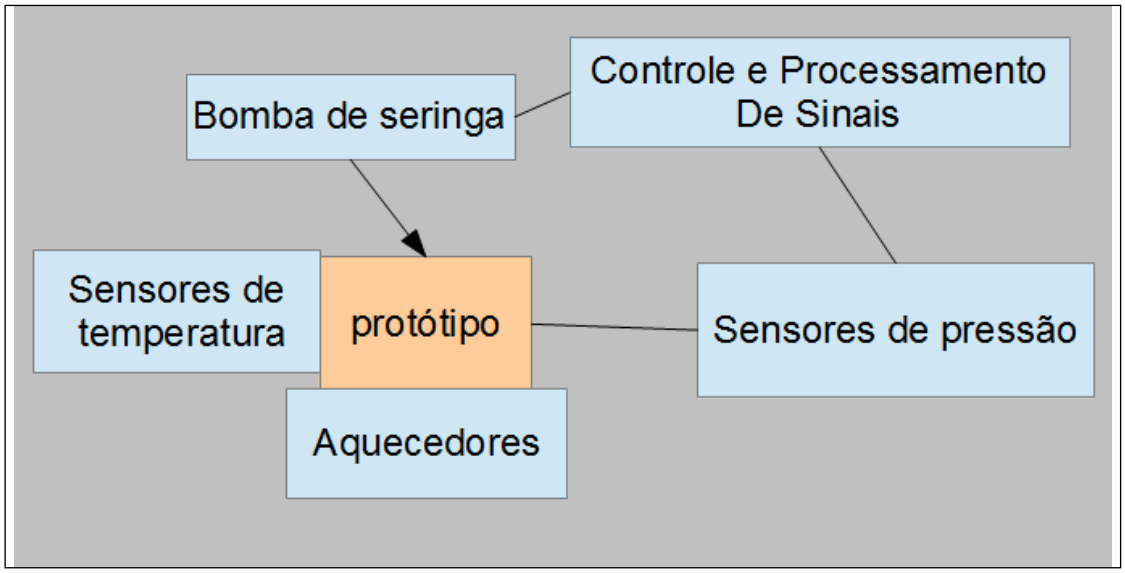

Figura 3: Layout da bancada de testes para dispositivos de microfluídica.

As figuras 2 e 3 mostram conceitualmente como se espera que a bancada seja.

3.1 Requisitos

Os requisitos da bancada de testes serão levantados a partir dos limites das grandezas físicas envolvidas na aplicação. Inicialmente são feitas estimativas para esses limites e, após a construção da bancada, esses limites são verificados.

3.1.1 Tamanho dos protótipos e do Dispositivo

Determinou-se que o tamanho máximo exposto na máquina de litografia disponível no laboratório de sistemas integráveis (LSI) da Escola Politécnica da USP é em torno de 6 cm x 6 cm. Assim, escolheu-se utilizar um tamanho padrão para os protótipos de 5 cm x 5 cm.

3.1.2 Ligações Hidráulicas

Tanto as seringas utilizadas quanto o sensor de pressão são adequados para serem utilizados com tubos de 5 mm de diâmetro. Dessa forma, decidiu-se utilizar esse tamanho de tubo para todas as ligações, a menos que não aplicável.

3.1.3 Limites de Temperatura

Uma das reações tipicamente utilizadas em laboratórios clínicos se chama Polimerase Chain Reaction (PCR), que consiste de duplicar um trecho de DNA (ou RNA, em alguns casos) diversas vezes, a ponto de sua quantidade ser detectável. Com esse tipo de reação, que é calibrada para uma sequencia específica de nucleotídeos, pode-se fazer testes do tipo tem/não tem para algumas doenças. Na reação PCR, deve haver ciclos de temperatura, sendo que a

maior delas chega a 94 °C . Essa é a estimativa inicial para a temperatura máxima que o sistema deve atingir.

Outra reação importante em laboratórios clínicos é a ELISA (Enzyme--Linked Immunoabsorbent Assay), em que substâncias são detectadas (tipicamente anticorpos) através de uma reação imune (ligação com algum tipo de anticorpo, como de coelhos). Nesse tipo de teste, a substância-alvo é quantificada através da mensuração de quantas reações imunes ocorrem, tipicamente transformando essa quantidade em um sinal luminoso. Em todos os protocolos de ELISA estudados as reações ocorrem na temperatura ambiente, então esse tipo de reação não exigirá maior flexibilidade na temperatura do sistema.

3.1.4 Potência dos Aquecedores

Com o requisito de controlar a temperatura do sistema microfluídico, surge a necessidade de estimar a potência que será necessária para tal controle. A potência para aquecimento será calculada como a necessária para manter o sistema em sua máxima temperatura (em torno de 100 °C) em uma temperatura ambiente de 0 °C, ou seja,manter uma diferença de temperatura com o ambiente de $\Delta T = 100$. Essa potência de aquecimento deve compensar a perda de calor para o ambiente e para o próprio fluído. A perda de calor para o ambiente pode ser estimada como a perda de calor para o ar através de uma área de $A = 0,0025m^2$ ($5cm$ x $5cm$) por convecção natural. Supondo-se as propriedades do ar a $50C$ ($323K$), tem-se um número de Rayleigh de $Ra_L = 8287$, de acordo com a equação 3.1 (INCROPERA et al., 2008). O coeficiente de transferência de calor médio pode ser estimado como $\overline{h} = 10,84 \frac{W}{m^2 \cdot K}$, de acordo com a equação 3.2 (INCROPERA et al., 2008). Dessa forma, a potência necessária

para manter a diferença de temperatura é $\dot{Q} = 2,71W$ (equação 3.3.

$$Ra_L = \frac{g\beta(T_s - T_\infty)L^3}{\nu\alpha} = \frac{9,8\frac{1}{323}(50)(\frac{0,0025}{0,2})^3}{15,89 \cdot 10^{-6} \cdot 22,5 \cdot 10^{-6}} \tag{3.1}$$

$$\overline{Nu_L} = \frac{\overline{h}L}{k} = 0,54Ra_L^{\frac{1}{4}} \tag{3.2}$$

$$\dot{Q} = \overline{h}A\Delta T = 10,84 \cdot 0,0025 \cdot 100 \tag{3.3}$$

A perda de calor para o líquido no interior do dispositivo, por outro lado, pode ser estimada como a potência necessária para aquecer a massa de líquido que entra no sistema por unidade de tempo. Adotando as propriedades da água (massa específica $\rho = 1000kg/m^3$ e calor específico $c = 4,1813J/g.K$) e um fluxo de $\phi = 10^{-10}m^3/s$ (vide seção 3.1.5), e de acordo com a equação 3.4, a potência necessária para manter a diferença de temperatura é de $4.10^{-5}W$.

$$\dot{Q} = \dot{m}c\Delta T = (\rho\phi)c\Delta T \tag{3.4}$$

Como a perda de calor para o fluído é mais de 4 ordens de grandeza menor que a perda para o ambiente, será considerada apenas essa última. Dessa forma, a potência para aquecer o sistema deve poder fornecer uma potência máxima de $2,71W$.

3.1.5 Faixa de Atuação de Pressão, Fluxo e Velocidade

Para se estimar a pressão necessária para ativar um sistema de canais microfluídicos utilizando a equação 2.17, é necessário estabelecer valores para a viscosidade do líquido (μ), o comprimento a ser percorrido (L), a velocidade (v) e as dimensões da sessão transversal ($\frac{2ab}{a+b}$). Utilizando-se como aproxima-

ção inicial o valor de viscosidade da água $\mu = 8,90.10^{-4} Pa.s$, um comprimento total de $L = 1m$, uma sessão transversal quadrada de aresta $a = b = 100\mu m$ e impondo-se uma velocidade do fluído de $v = 1cm/s = 0,01m/s$, chega-se ao valor de pressão de $\Delta p = 28,5kPa$. Assim, o sistema de medição deve ser capaz de medir pressões próximas a esse valor.

Pode-se estimar o fluxo de líquido no dispositivo pela equação 3.5. Utilizando-se as aproximações anteriores, chega-se ao valor de fluxo de $\phi = 10^{-10} m^3/s$. Com esse fluxo, espera-se que uma seringa de $10mL$ se esvazie em $10^5 s$. Se o percurso dessa seringa é de $100mm$, então a velocidade de sua haste deve ser de $1\mu m/s$. Essa deve ser a velocidade de trabalho média do acionamento da seringa.

$$\phi = vA = vab \tag{3.5}$$

3.1.6 Faixa de Comprimentos de Luz a serem Detectados

Os reagentes luminosos utilizados nas reações laboratoriais se chamam fluoroforos. Os fluoroforos pesquisados emitem luz na faixa de comprimento de onda entre $386nm$ (hidroxicumarina) e $767nm$ (conjugados de aloficocianina-Cy7). O composto fluoresceína, por exemplo, emite no comprimento de onda de $521nm$. O sistema de medição deve, por tanto, detectar luz na faixa de comprimentos de onda de $386nm$ até $767nm$.

3.1.7 Suporte a Modificações

Como cada projeto tem necessidades únicas, é de se esperar que sejam necessárias pequenas modificações na plataforma de testes para sua utilização com êxito. Assim, um dos requisitos da plataforma é que a sua expansão

ou modificação sejam previstas no projeto. Os pontos que podem ser alterados são:

- Número de sensores: em algumas aplicações são necessárias mais medidas que em outras.
- Tipo de sensores: a bancada prevê sensores de pressão, temperatura e luminosidade, porém deve suportar qualquer outra medida que seja adicionada, como po rexemplo resistência elétrica. Outro ponto nesse quesito é que possam ser utilizados sensores de diversos modelos.
- Quantidade de bombas de seringa: em algum projeto pode ser essencial a criação de dois ou mais escoamentos independentes. Nesse caso, a plataforma deve suportar a inclusão de novas bombas de seringa.

3.2 Projeto

3.2.1 Bomba de Seringa

Para fornecer fluxo de líquidos em níveis compatíveis com a microfluídica, optou-se por equipar a plataforma de testes com uma bomba de seringa. Uma bomba de seringa é basicamente um sistema que pressiona uma seringa para ser esvaziada a uma velocidade pré-determinada. A bomba de seringa consiste basicamente de três elementos: atuador, transmissão e suportes.

O atuador deve fornecer potência suficiente para movimentar o êmbolo. A potência requerida para esse movimento é dada pelo fluxo e pressão máximos ($28,5kPa$ x $10^{-10}m^3/s$), e vale aproximadamente $3.10^{-6}W$. Além disso, devem ser compensadas as perdas no sistema de transmissão, que devem superar em várias ordens de grandeza a potência necessária para apenas

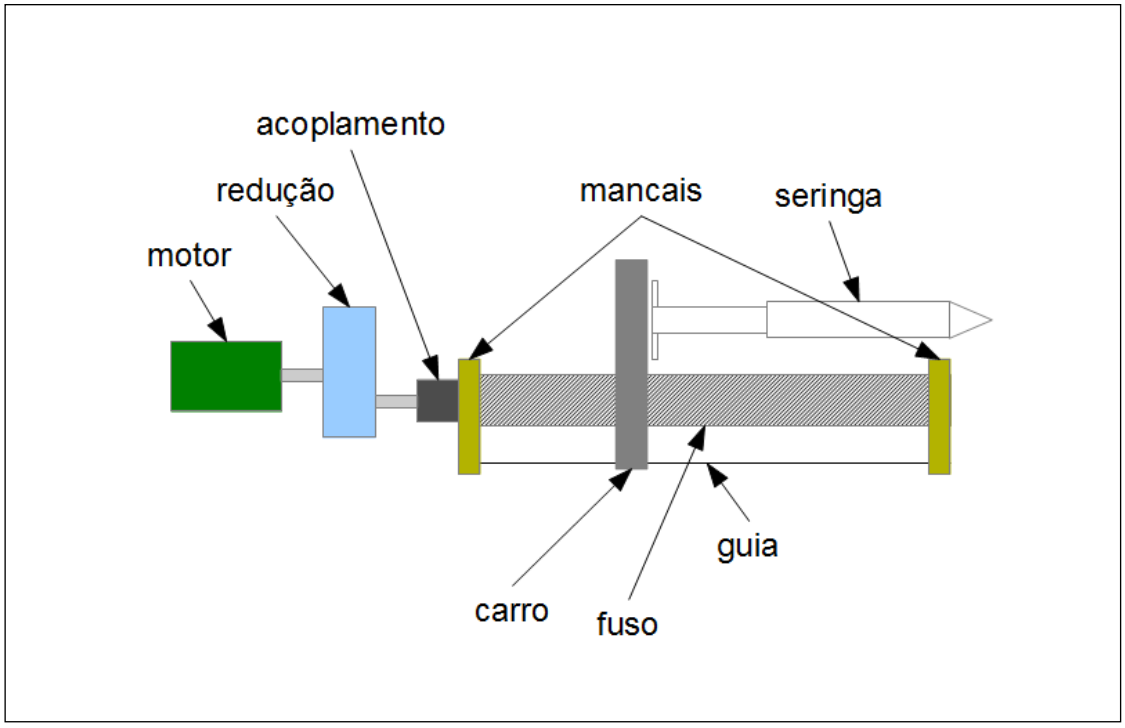

Figura 4: Sistema de transmissão para a bomba de seringa.

promover o escoamento. Pela disponibilidade, optou-se por utilizar um motor de corrente contínua com encoder embutido da Buehler. A potência do atuador será limitada pelo circuito elétrico de acionamento, que pode fornecer até 12 W (vide 3.2.4). Para o controle da velocidade do motor determinou-se um modelo dinâmico do mesmo, discutido com mais detalhes na seção 3.2.5.

Além de gerar a potência necessária ao movimento do fluído, também é necessário convertê-la em sua forma útil: potência hidráulica. Para isso, sera utilizada uma seringa convencional (sem agulha) e uma transmissão que converta o movimento circular do motor adotado a um movimento linear de acionamento da seringa.

Por questão de disponibilidade, optou-se por utilizar uma redução e uma transmissão por fuso. A Figura 4 mostra um esboço do sistema de transmissão.

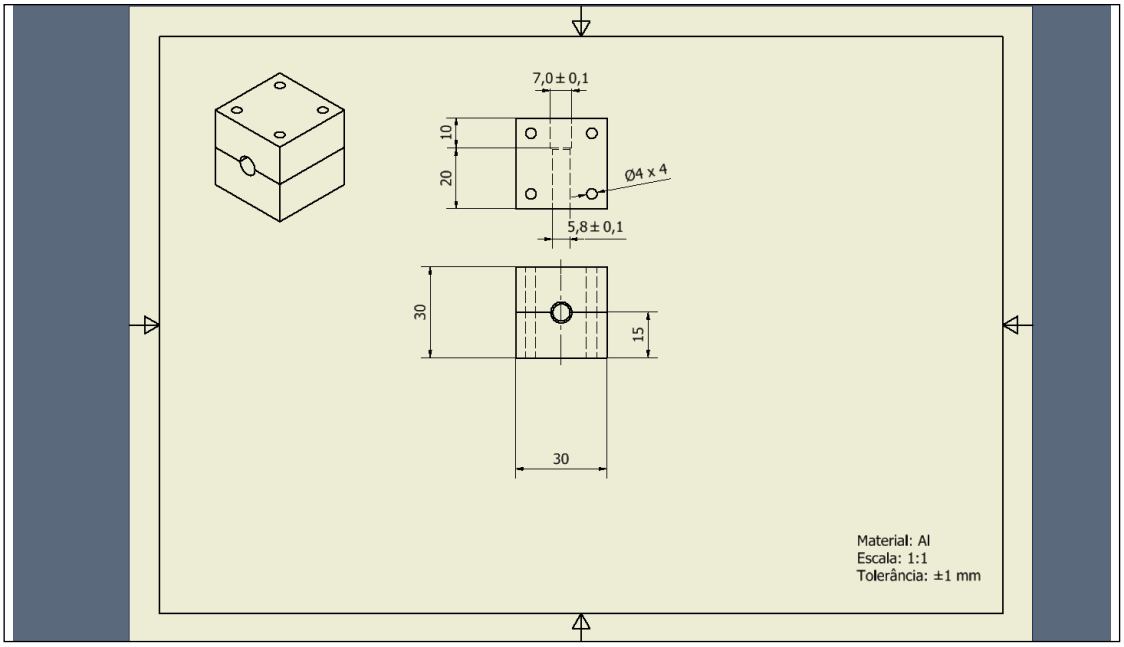

Figura 5: Acoplamento entre o eixo da redução e o eixo do fuso.

A redução foi retirada de um motor síncrono e adaptada para ser acoplada ao motor utilizado. A engrenagem de entrada da redução foi fixada na engrenagem de saída do motor por um furo e posteriormente fixada com cola (cianoacrilato). A taxa de redução foi calculada pelo número de dentes das engrenagens, e corresponde a $R_0 = \frac{38}{8}\frac{55}{10}\frac{62}{10} = 162$.

O acoplamento entre a redução e o fuso será feito por uma união rígida, ilustrada na figura 5. O fuso será sustentado por dois mancais de escorregamento em nylon. Há uma guia em paralelo ao fuso e fixado nos mancais. O carro terá um furo guia, uma rosca e um dispositivo de encaixe para a parte móvel da seringa. Esta, por sua vez, estará fixada no mancal mais próximo. A figura 6 mostra o projeto dos mancais (à esquerda) e do carro (à direita).

Os suportes da bomba de seringa serão feitos em MDF devido à disponibilidade e facilidade de trabalho. O motor a ser utilizado foi modelado utilizando-se o software comercial Inventor ® da Autodesk. A partir desse modelo pôde

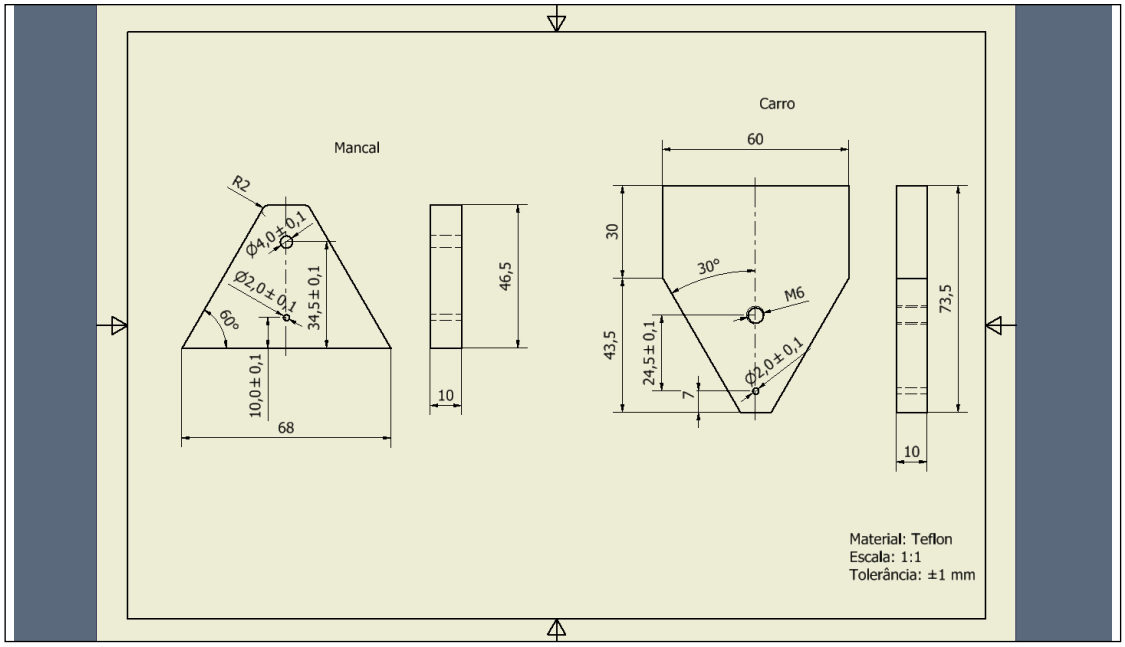

Figura 6: Projeto dos mancais e do carro do sistema de transmissão.

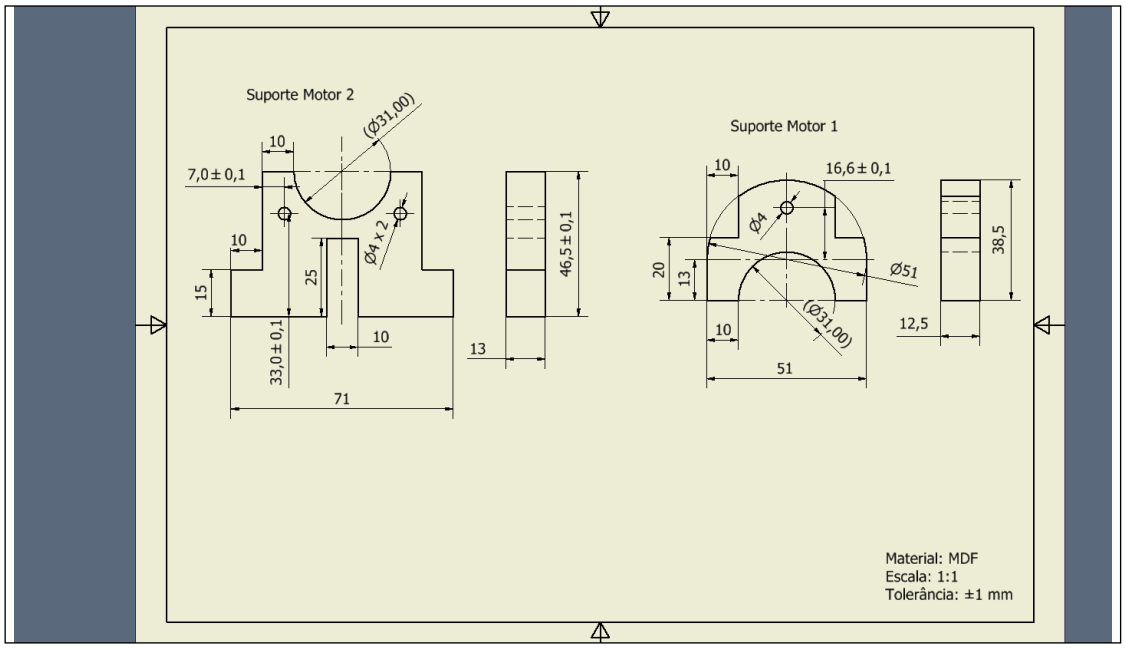

Figura 7: Suporte do Motor da bomba de seringa.

ser projetado um suporte. Na figura 7 tem-se o suporte projetado.

Um procedimento semelhante foi adotado para projetar o suporte da redução. Na figura 8 tem-se o projeto de tal suporte.

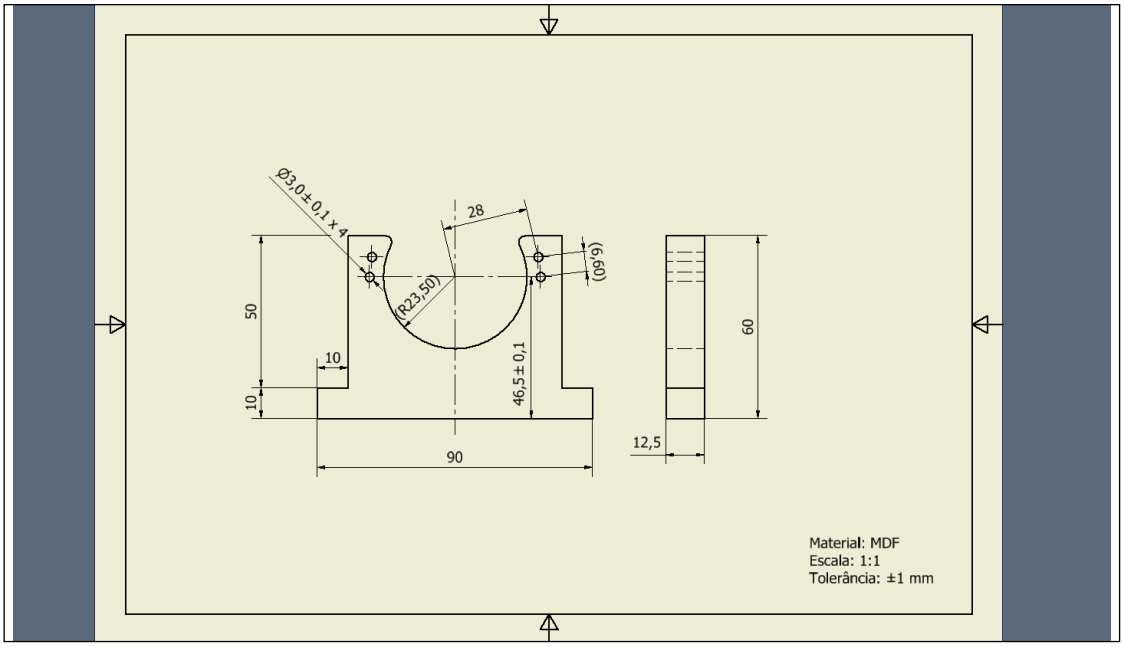

Figura 8: Suporte da redução da bomba de seringa.

3.2.2 Sensores

Os sensores são selecionados de acordo com os requisitos e, então, é definido o suporte necessário a eles, como alimentação e filtros.

3.2.2.1 Temperatura

O sensor escolhido para aferição de temperatura é o LM35, fabricado pela antiga National Semiconductor, agora parte da Texas Instruments. O LM35 possui limite de temperatura de 150 ºC, o que o torna adequada à aplicação. O *datasheet* deste componente sugere a montagem do filtro da figura 9 para desacoplar a saída do sensor de cargas capacitivas altas (maiores que $50pF$).

Para definir se o filtro seria ou não utilizado, foi realizado um experimento em que houve uma quantificação da melhoria no sinal do sensor. Foi utilizada a placa de desenvolvimento Arduino™Diecimila para realização do ex-

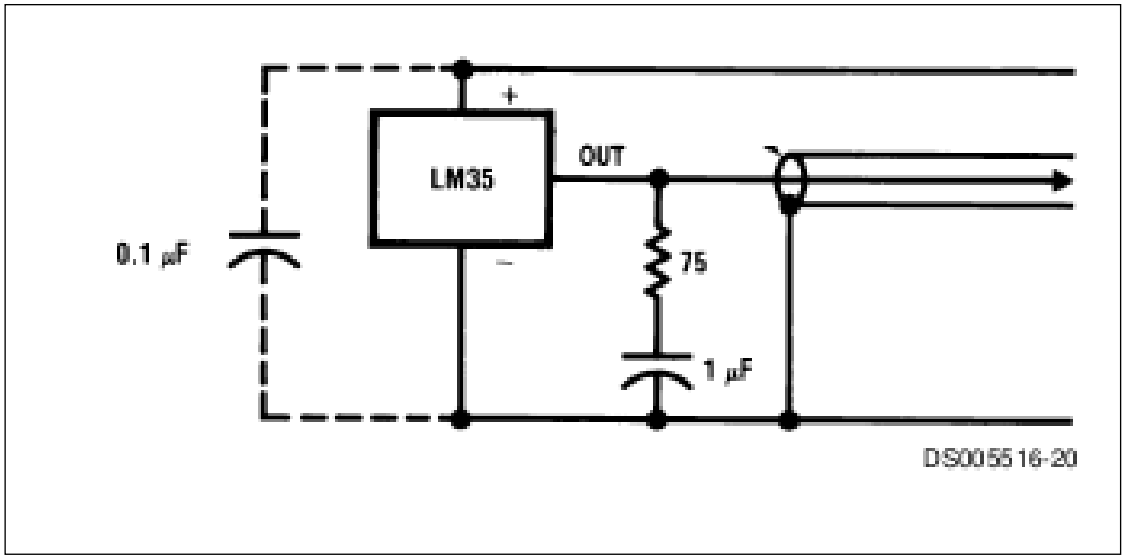

Figura 9: Filtro para o sensor de temperatura LM35.

perimento. A medida utilizada foi a de intervalo máximo para variação de 2 unidades AD (cada unidade AD equivale, no caso da placa Arduino™Diecimila, à variação de temperatura necessária para gerar uma variação de Vref/1024, onde Vref é a tensão de referência do sensor) durante cinco minutos. O critério de aceitação foi o de intervalo de no mínimo um segundo nas condições da sala MZ-12 do prédio da Mecatrônica, com o ar-condicionado ligado. O código do programa utilizado encontra-se no apêndice B.1.

Sem a utilização de filtro, o intervalo máximo para variação de uma unidade AD obtido esteve na faixa de 17 a 44 milissegundos. Com o filtro da figura 11, o intervalo máximo esteve entre 4106 e 53099 milissegundos. Dessa forma, decidiu-se utilizar o filtro.

3.2.2.2 Pressão

Para medição de pressão, foi selecionado um sensor de 50 kPa da fabricante Freescale, com saída escalonada (0 a 5 V) e encaixe para tubos de 5 mm chamado MPX5050DP. Para esse sensor, foram encontradas dois circui-

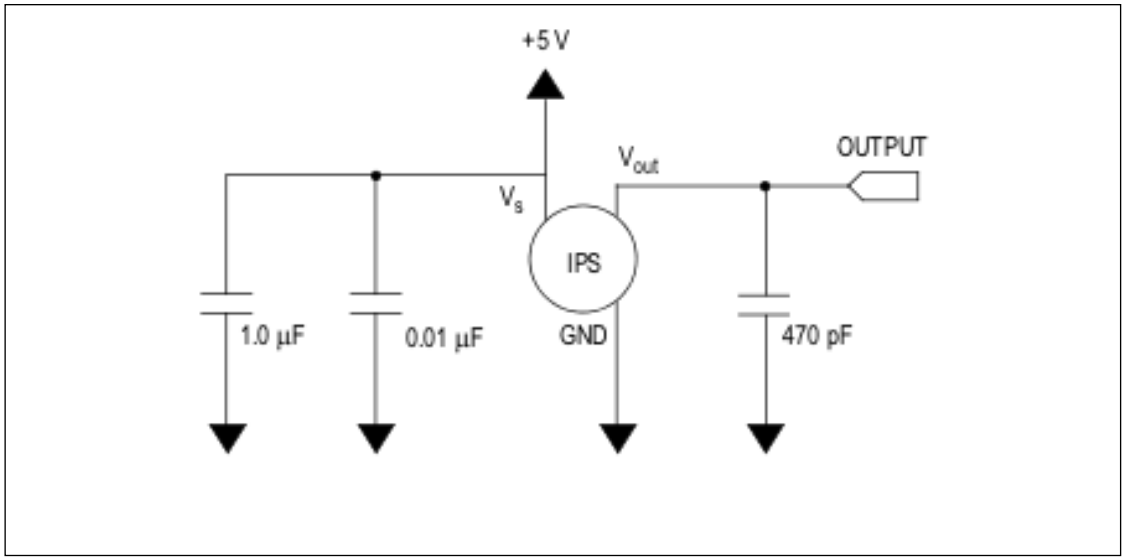

Figura 10: Filtro para o sensor de pressão MPX5050, retirado de seu *datasheet*.

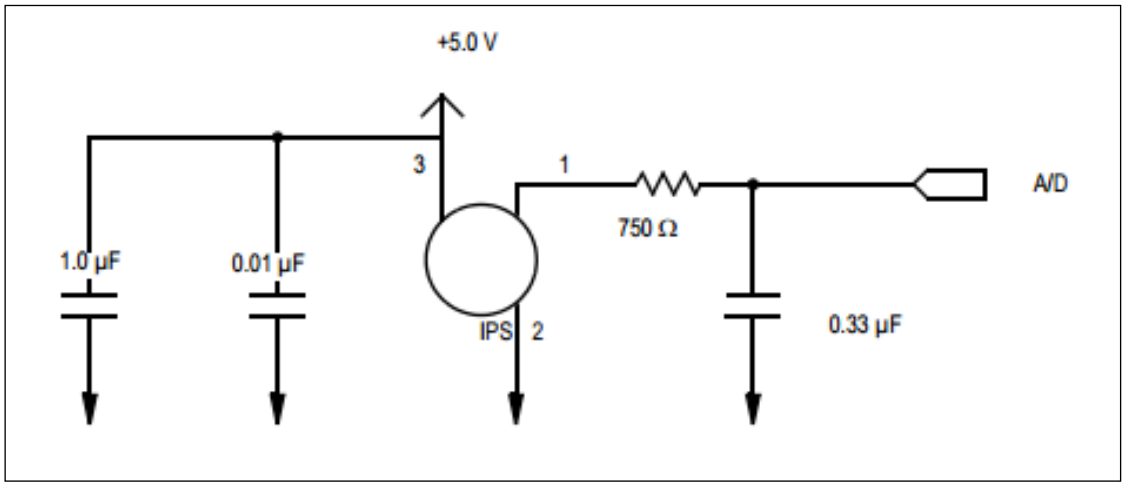

Figura 11: Filtro para o sensor de pressão MPX5050, retirado da *application note* AN1646 da Freescale.

tos distintos recomendados pela fabricante, o da figura 10 e o da figura 11.

Para definir se algum dos dois filtros encontrados seria ou não utilizado, foi realizado um experimento semelhante ao feito com o sensor de temperatura. A medida utilizada também foi a de intervalo máximo para variação de 2 unidades AD durante cinco minutos. O critério de aceitação foi o de intervalo de no mínimo um segundo com as entradas abertas para o ambiente, nas condi-

ções da sala MZ-12 do prédio da Mecatrônica, com o ar-condicionado ligado. O código do programa utilizado encontra-se no Apêndice B.2.

Sem utilizar filtros, o intervalo máximo para variação de duas unidade AD obtido foi de 64 milissegundos. Com o filtro da figura 10, o intervalo máximo foi de 61 milissegundos. Com o filtro da figura 11, o intervalo máximo estava na faixa de 11671 a 22666 milissegundos. Dessa forma, decidiu-se utilizar o filtro da Figura 11. Nota-se que essa montagem corresponde a um filtro RC (passa-baixa), com $R = 750\Omega$, $C = 0,33\mu F$ e frequência de corte de $\omega_c = 643Hz$, de acordo com a equação 3.6.

$$\omega_c = \frac{1}{2\pi RC} \tag{3.6}$$

3.2.2.3 Luminosidade

Devido à limitação de tempo e orçamento optou-se por não equipar a versão inicial da plataforma e testes com sensores de luz.

3.2.3 Aquecedores

Devido à limitação de tempo e orçamento optou-se por não equipar a versão inicial da plataforma e testes com aquecedores.

3.2.4 Eletrônica

Devido à disponibilidade e facilidade de uso, foi utilizada a placa Arduino™Diecimila para controlar a parte eletrônica e interfacear com o computador. O software utilizado para desenvolver os circuitos impressos foi o Eagle ®.

Para o projeto eletrônico decidiu-se criar uma placa para lidar com alta potência (ou seja, o acionamento do motor). Como o cabo do motor também possui a saída do *encoder*, essa placa, doravante denominada placa de potência, também faz a ligação destes com o microcontrolador. As demais partes - sensores de pressão, sensores de temperatura e outros sensores - estão conectadas à placa de sensores. Para fazer a ligação entre a placa do microcontrolador e as placas foi feita uma placa adaptadora e dois cabos, o CABO_SEN e o CABO_POT, que se conectam a placa adaptadora à placa de sensores e à de potência, respectivamente. O diagrama de blocos da figura 12 ilustra todas essas partes. Além dos cabos já mencionados foram incluidos os cabos CABO_MOTOR, já incluso com o motor, CABO_LM, para colocação do sensor de temperatura próximo ao experimento e o CABO_POW, para fornecer 12 V para o acionamento do motor.

A forma escolhida para se dividir a eletrônica facilita uma futura expansão, já que a placa de sensores pode ser modificada para obter sinais de outros sensores ou a placa de potência também ser modificada para mover outro motor ou até serem incluídos itens de acionamento ou sensoriamento em uma placa adaptadora intermediária sem outras modificações. O software também foi implementado considerando-se que o número de sensores e atuadores e seu mapeamento nos canais da placa Arduino™ podem ser modificados.

A placa de sensores necessita receber alimentação (Vcc=5V e GND) da placa adaptadora e fornece as saídas dos sensores de temperatura (LMout) e de pressão (MPXout). Já a placa de potência necessita de dois sinais de pwm (pwm1 e pwm2) para acionar o motor, alimentação (Vcc=5V e GND) para acionar o *encoder* e fornece como saída dois sinais digitais (CNA e CNB) vindos do encoder. Por fim, o LM35 necessita de alimentação (Vcc=5V e

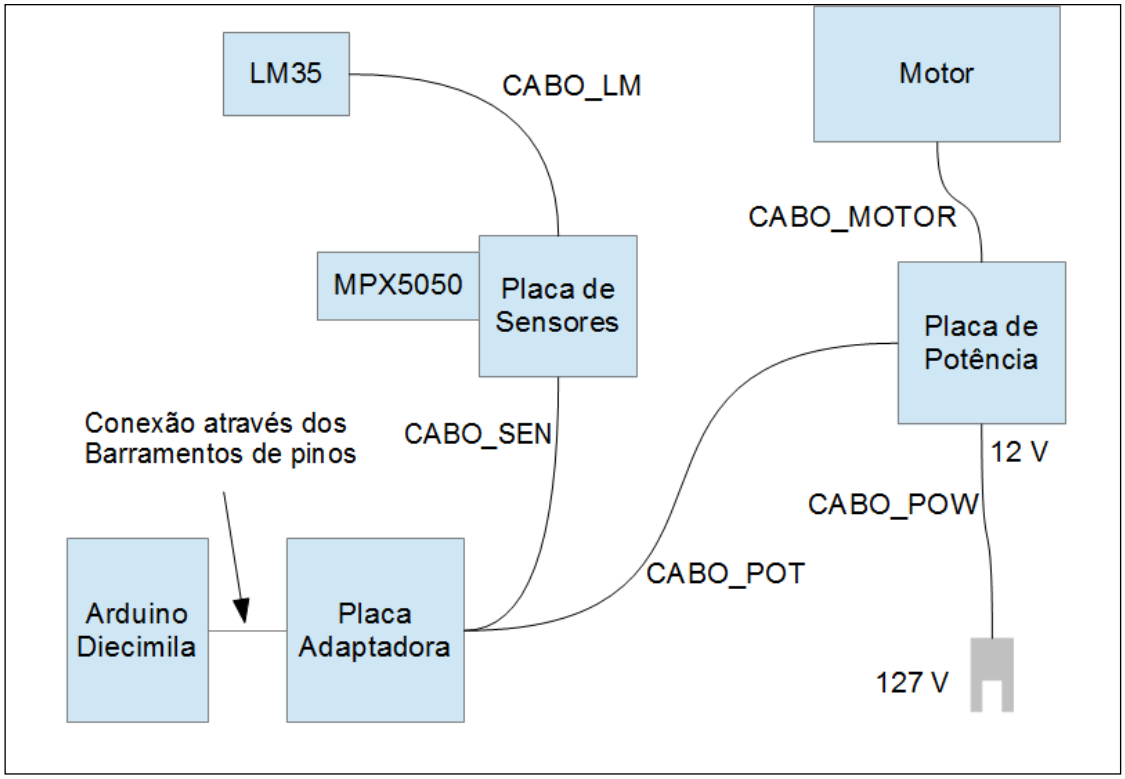

Figura 12: Esquema elétrico da plataforma de testes.

GND) e fornece uma saída analógica (LMout). Com esse mapeamento dos sinais, foi definida a pinagem de cada cabo, conforme ilustra a figura 13. O CABO_MOTOR teve seus sinais descobertos por experimentação.

Para o projeto das placas, foram adotadas as larguras de 40 mil para trilhas de sinal e 60 mil para o Vcc. Esses valores foram obtidos através de consecutivos aumentos e redesigns do *layout* até o maior valor que pudesse ser feito com apenas um lado de trilhas e utilizando o menor número de *jumpers*.

3.2.4.1 Placa de Potência

Para acionar o motor foi escolhido o circuito integrado TA7267B da Toshiba. De acordo com as especificações deste componente e dos cabos que se conectam à placa de potência, foi feito o esquema elétrico da figura 14 e seu respectivo *layout*, ilustrado na figura 15.

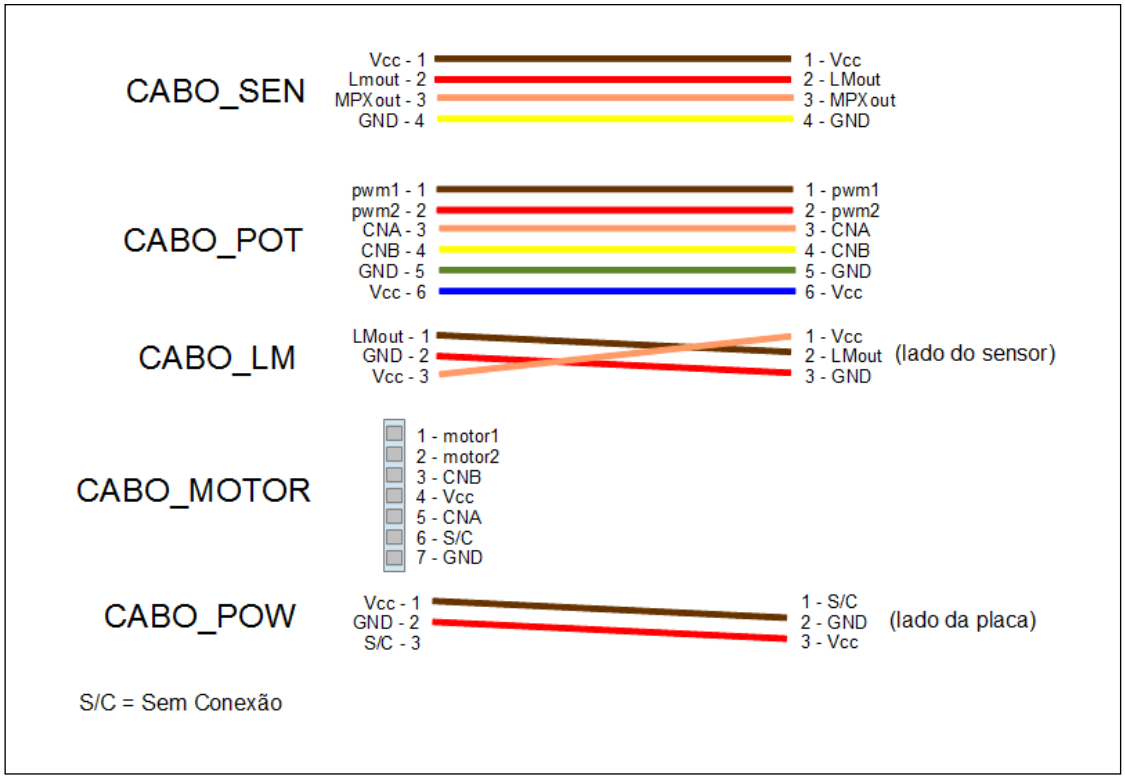

Figura 13: Especificação dos cabos a serem usados na plataforma de testes.

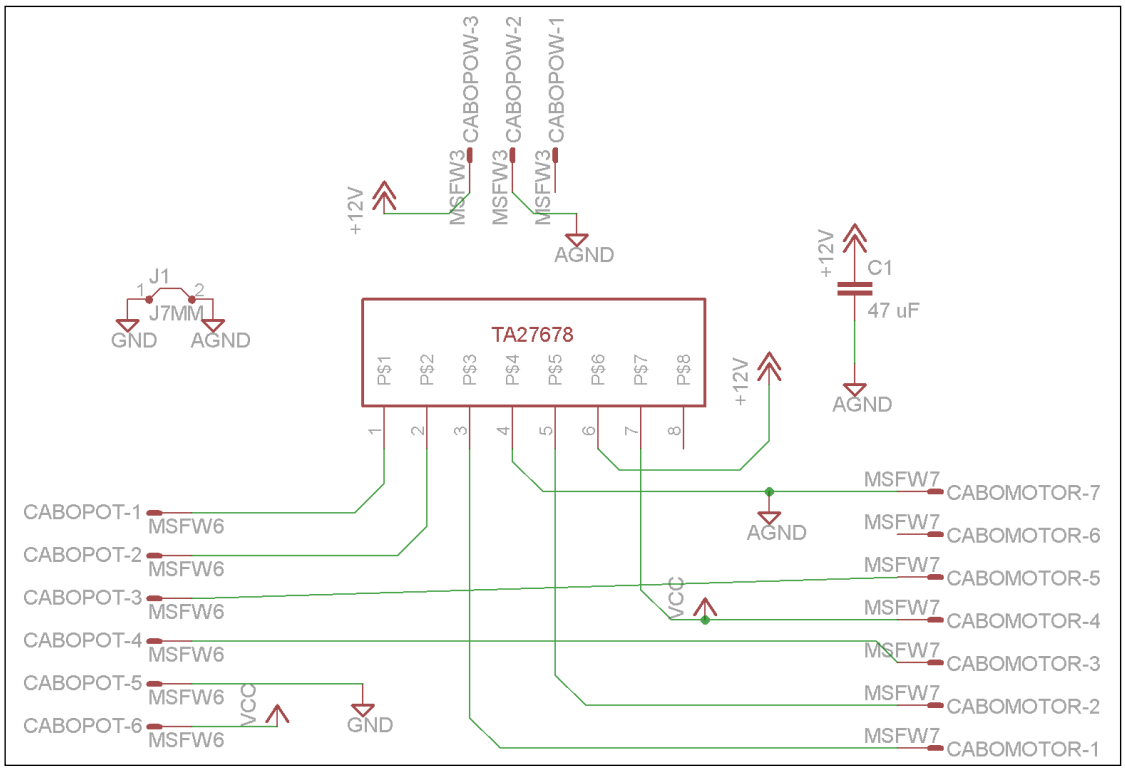

Figura 14: Esquema elétrico da Placa de Potência.

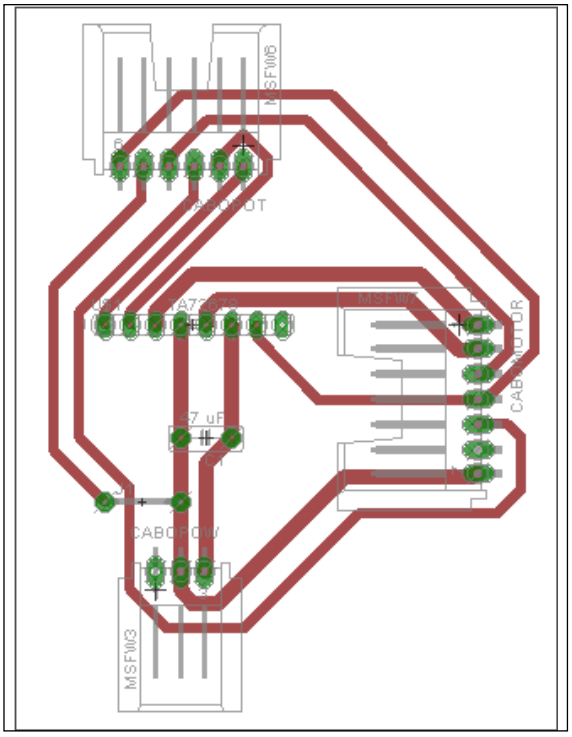

Figura 15: *Layout* da Placa de Potência.

O motor foi testado com um multímetro e registrou-se um valor máximo de corrente de 600 mA. Assim, a fonte de 12 V deverá ser limitada a 1 A e, por tanto, fornecerá uma potência máxima de 12 W.

3.2.4.2 Placa de Sensores

Baseando-se nas especificações dos sensores e dos cabos que se conectam à placa de sensores e incluindo os filtros escolhidos, foi feito o esquema elétrico da figura 16, e seu correspondente *layout*, ilustrado na figura 17. Esse esquema elétrico e esse *layout* são a terceira versão feita para essa placa, já que as primeiras possuiam conexões trocadas com os cabos dos sensores, que após uma verificação, ainda na fase de design, foram corrigidas.

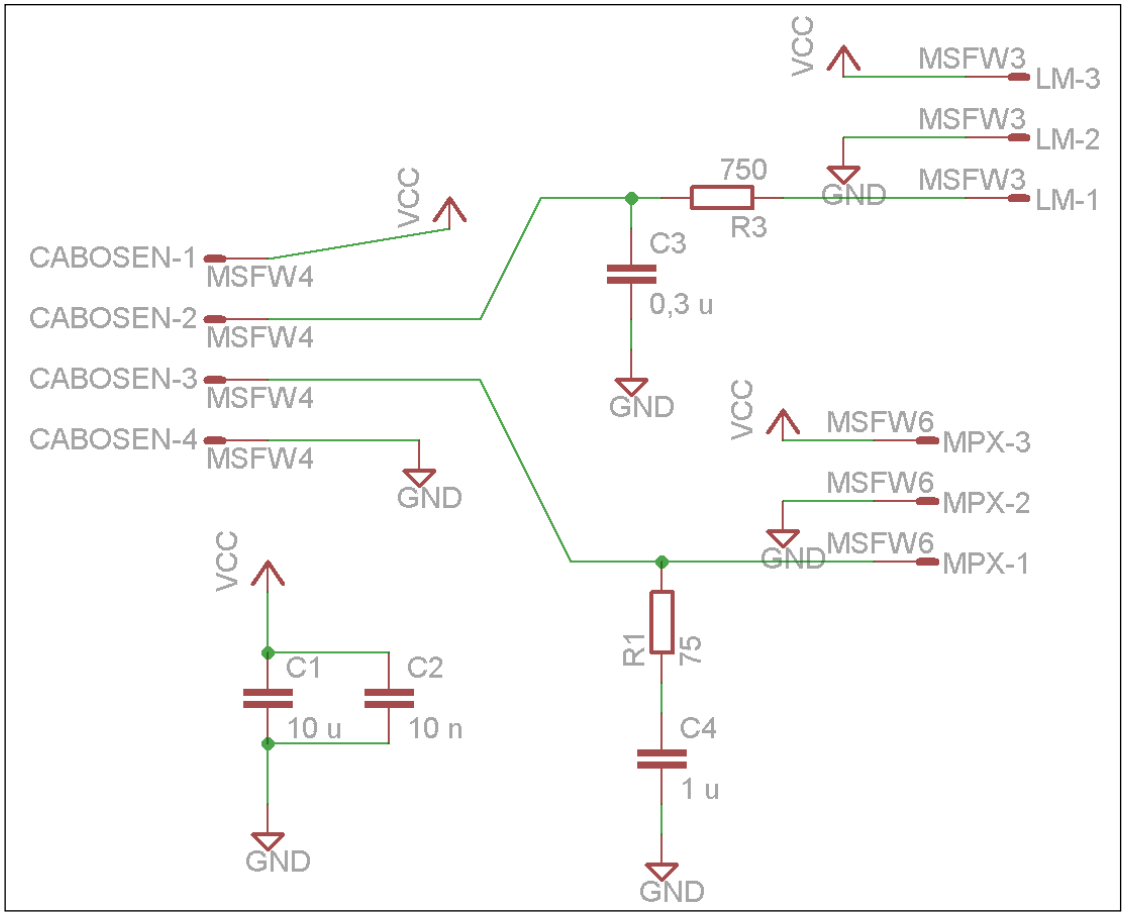

Figura 16: Esquema elétrico da Placa de Sensores.

3.2.4.3 Placa Adaptadora

Com base nas especificações das ligações que devem ser feitas na placa adaptadora e nos pinos da placa Arduino™Diecimila, foi feito o esquema elétrico da figura 18 e seu respectivo layout, ilustrado na figura 19. Essa versão desta placa é a segunda, pois primeiramente houve uma inversão na numeração dos pinos que se conectam à placa Arduino™. Esse erro foi corrigido após a primeira fabricação.

3.2.4.4 Comunicação

Para configurar a plataforma e acompanhar o funcionamento da mesma, será feita uma aplicação em computador. Desta forma, é necessária a in-

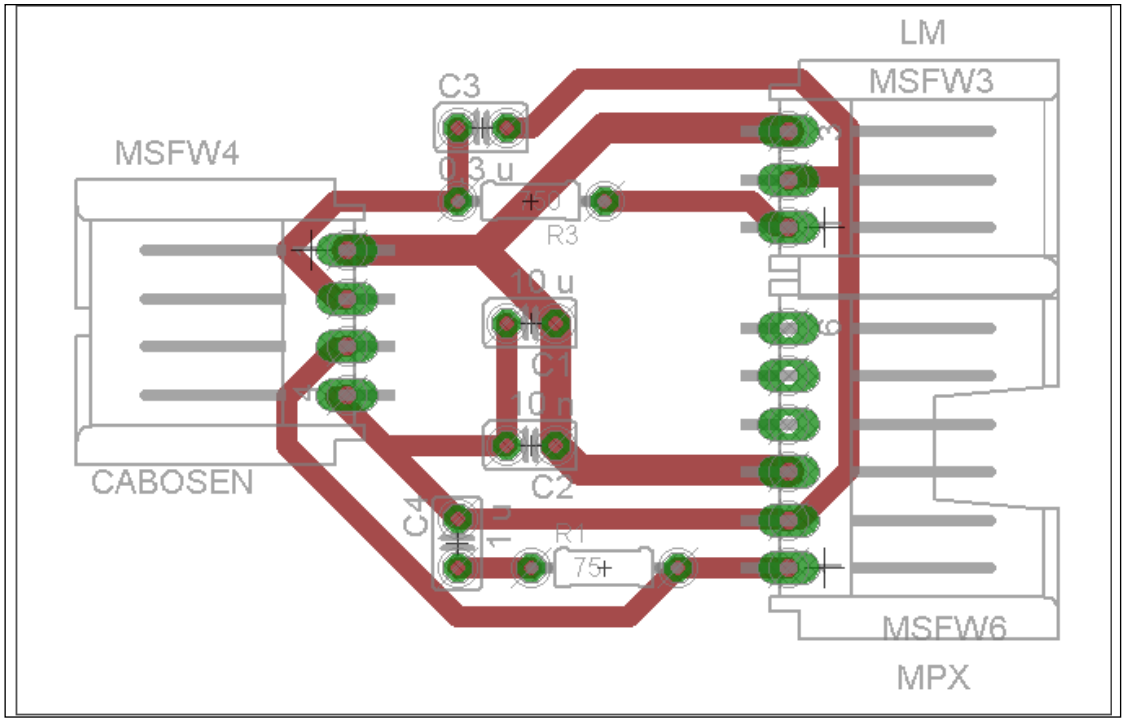

Figura 17: *Layout* da Placa de Sensores.

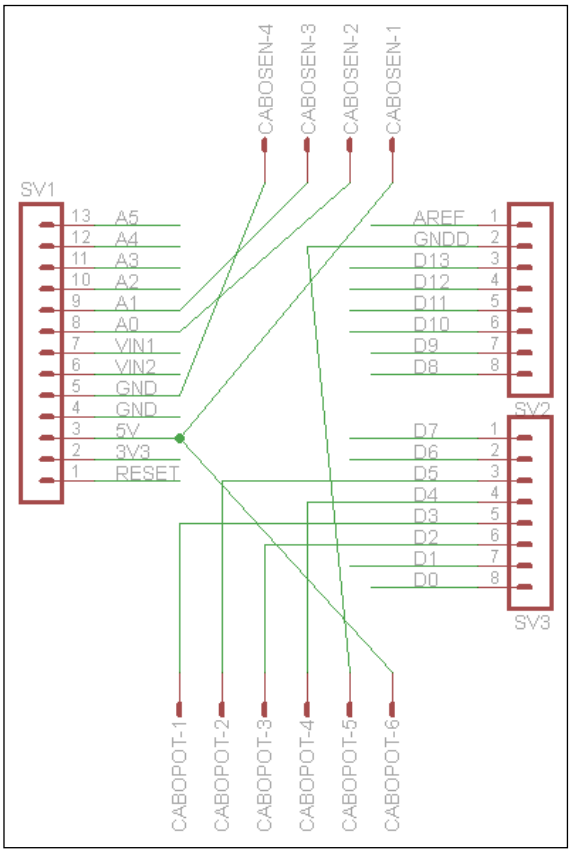

Figura 18: Esquema elétrico da Placa Adaptadora.

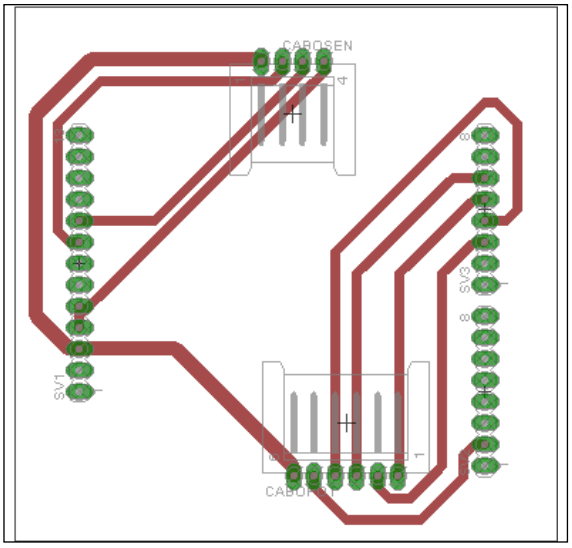

Figura 19: *Layout* da Placa Adaptadora.

tegração da placa Arduino™Diecimila com o computador. Essa placa possui interface USB, implementada conforme o esquema da figura 20. Dessa forma, há uma integração do microcontrolador ATmega168, que contém suporte para comunicação serial UART TTL, com a comunicação serial USB através do CI FT232RL da FTDI. Esse CI implementa USB 2.0 e provê uma porta com virtual para o computador. No modelo OSI/ISO, isso significa que a camada de comunicação 1 (física) já está implementada. Resta, por tanto, desenvolver as outras camadas até se obter uma interface com a aplicação. Para cumprir esse papel, foi escolhido o protocolo *Modbus*. Pode-se entender que o protocolo *Modbus* engloba duas camadas do modelo OSI/ISO: a camada 2 (*data link*), através do *Modbus Serial Line Protocol* e a camada 7 (aplicação), através do *Modbus Protocol*. Assim, obtém-se recursos no nível de aplicação para utilizar a comunicação.

O objetivo dessa seção é expor o projeto da implementação do protocolo *Modbus* tanto em uma placa Arduino™Diecimila quanto em um computador. Por parte do computador, foi escolhido desenvolver a interface com o usuário em linguagem java, na IDE Netbeans. A comunicação entre o aplicativo e

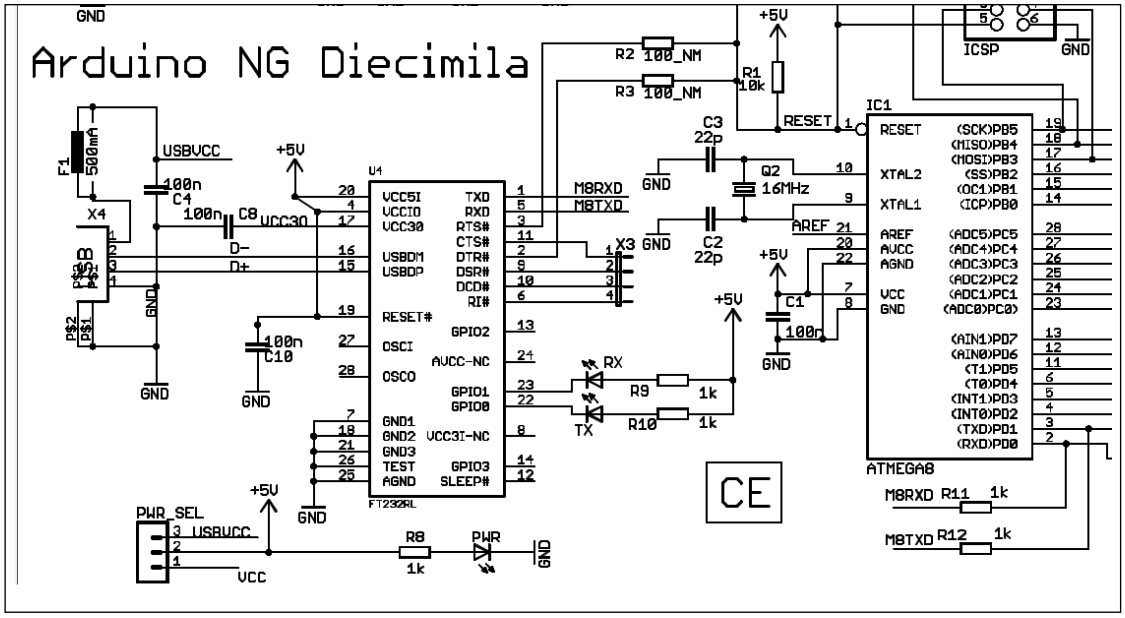

Figura 20: Comunicação USB no Arduino™Diecimila (retirado de http://www.arduino.cc/en/Main/arduinoBoardDiecimila)

o hardware do computador é feita através dos drivers instalados, e entre o aplicativo e o driver pela biblioteca RXTXcomm. Essa biblioteca foi escolhida pois já havia sido utilizada em trabalhos anteriores e, por tanto, demanda menos tempo de implementação. Restava, por tanto, a criação de uma classe java que de fato implemente o protocolo *Modbus*. Essa classe foi chamada CommunicationController. Já na parte da placa Arduino™, a ponte entre o software e o hardware de comunicação é feita através de uma biblioteca chamada SoftwareSerial. Para implementar o protocolo Modbus escolheu-se adaptar uma biblioteca existente, chamada ModbusSlave (https://sites.google.com/site/jpmzometa/arduino-mbrt/arduino-modbus-slave) e integrá-la à aplicação. A figura 21 mostra os componentes de Software que foram desenvolvidos.

Como as trocas de dados entre a plataforma de testes e o computador são de adquirir valor de sensores e ajustar valores de configurações, somente são necessárias as funções 3 (0x03): *read holding registers* e 6 (0x06): write

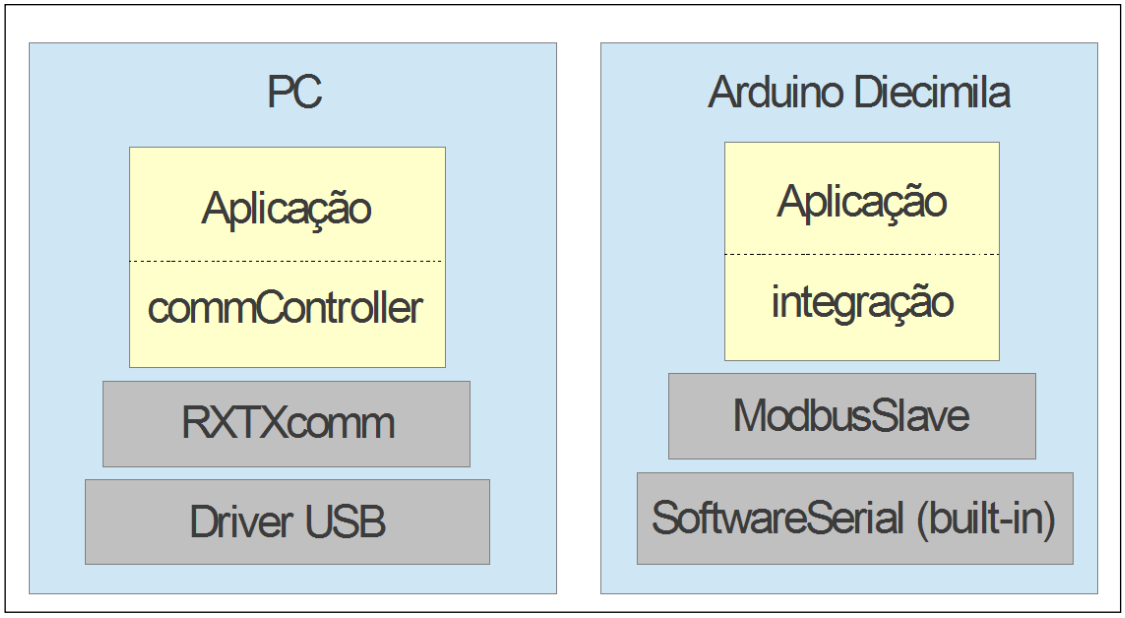

Figura 21: Elementos de software disponíveis previamente (cinza claro) e desenvolvidos (amarelo claro) para a comunicação.

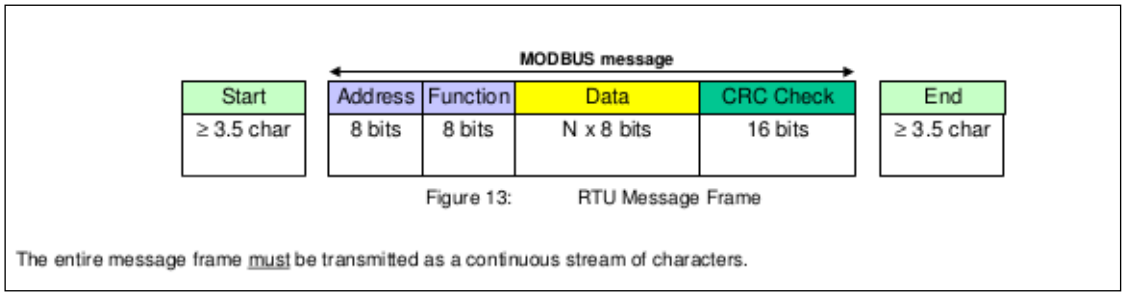

Figura 22: Composição dos pacotes de mensagens em Modbus RTU. (obtido em http://www.modbus.org/specs.php).

single register do protocolo *Modbus*. O protocolo define que é obrigatória a implementação de Modbus RTU, onde os endereços e valores são representados em formato binário, em oposição a Modbus ASCII, onde cada byte é transmitido por 2 bytes que representa o caracter em ASCII do byte a ser transmitido, que é opcional. Dessa forma, somente será feita a implementação de Modbus RTU. O *frame* dos caracteres utilizado é de 8 bits, paridade par (*even*) e 1 *stop bit*. O formato dos pacotes de dados está ilustrado na figura 22.

A latência máxima aceitável é de 100 ms, para que esteja abaixo do tempo médio de resposta de um ser humano de 190 ms para estímulos visuais (KOSINSKI, 2012). Com isso espera-se que o atraso não seja evidente a um operador leigo.

A classe CommunicationController em java deverá conter os seguintes métodos públicos:

- *configure*(baud,parity) : configura a baud rate e a paridade utilizadas.
- *init*(port) : inicia uma conexão serial na porta *port*, com os parâmetros definidos em *configure* ou os default: baud = 9600, parity = even.
- *end*() : encerra uma conexão ativa
- *readRegister*(slave_id, register, n_of_registers) : envia uma mensagem ao escravo com id *slave_id*, através da conexão atualmente ativa, requisitando a leitura de n_de_registers registradores a partir do registro register. O retorno deve ser um objeto que indique o sucesso ou não da comunicação, o sucesso ou não da execução, os valores requisitados e outras informações da mensagem recebida. O pacote de dados da função de leitura de registrador está ilustrado na figura 23.
- *writeRegister*(slave_id, register, value) : envia uma mensagem ao escravo com id *slave_id*, através da conexão atualmente ativa, requisitando a escrita do valor value no registro register. O retorno deve ser um objeto que indique o sucesso ou não da comunicação, o sucesso ou não da execução e outras informações da mensagem recebida. A figura 24 mostra a especificação do pacote de dados da função de escrita de um único registrador.

Request

Function code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	**0x03**
Byte count	1 Byte	2 x **N***
Register value	**N*** x 2 Bytes	

***N** = Quantity of Registers

Error

Error code	1 Byte	**0x83**
Exception code	1 Byte	01 or 02 or 03 or 04

Here is an example of a request to read registers 108 – 110:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**03**	Function	**03**
Starting Address Hi	**00**	Byte Count	**06**
Starting Address Lo	**6B**	Register value Hi (108)	**02**
No. of Registers Hi	**00**	Register value Lo (108)	**2B**
No. of Registers Lo	**03**	Register value Hi (109)	**00**
		Register value Lo (109)	**00**
		Register value Hi (110)	**00**
		Register value Lo (110)	**64**

Figura 23: Especificação da função 3 (0x03): read holding registers. (obtido em http://www.modbus.org/specs.php).

Request

Function code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 to 0xFFFF

Response

Function code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 to 0xFFFF

Error

Error code	1 Byte	**0x86**
Exception code	1 Byte	01 or 02 or 03 or 04

Here is an example of a request to write register 2 to 00 03 hex:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**06**	Function	**06**
Register Address Hi	**00**	Register Address Hi	**00**
Register Address Lo	**01**	Register Address Lo	**01**
Register Value Hi	**00**	Register Value Hi	**00**
Register Value Lo	**03**	Register Value Lo	**03**

Figura 24: Especificação da função 6 (0x06): write single register. (obtido em http://www.modbus.org/specs.php).

A parte de integração no software da placa Arduino™deve configurar os parâmetros de comunicação e periodicamente atualizar os registros de acordo com as mensagens recebidas. Deve fazer uso dos métodos *configure* e *refresh* da biblioteca ModbusSlave.

Os testes que verificaram se há algum problema com alguma das soluções utilizadas para essas especificações foram feitos com o modo "TESTE_SIMPLES"acionado da classe CommunicatorController. Neste modo, tanto as mensagens enviadas quanto as mensagens recebidas são exibidas na tela. Os testes consistiram de:

1. Teste do uso simples: Verificar se todas as mensagens enviadas e recebidas estão de acordo com a especificação do protocolo e se todas as tarefas são executadas quando são enviadas mensagens corretas.

2. Teste do uso repetido: Verificar se todas as mensagens enviadas e recebidas estão de acordo com a especificação do protocolo e se todas as tarefas são executadas ao utilizar um registrador repetidamente.

3. Teste do uso incorreto: Verificar se todas as mensagens enviadas e recebidas estão de acordo com a especificação do protocolo e se a execução falha em todos os casos quando se enviam mensagens incorretas.

3.2.5 Controle

3.2.5.1 Modelagem do Motor

Para o projeto do controlador do motor, primeiramente foi feita a modelagem do mesmo. A relação entre a posição angular do eixo de um motor e a tensão de entrada pode ser modelada como nas equações 3.7 e 3.8, onde L_a é a indutância da armadura do motor, R_a a resistência da mesma, i_a é a

corrente de armadura, K_v é a constante de velocidade do motor, e_a é a tensão de armadura, J_0 é a inércia do rotor, θ é a posição angular do mesmo, b_0 é o coeficiente de atrito viscoso, K_t é a constante de torque do motor. Aplicando-se a transformada de Laplace em ambas as equações e eliminando i_a, chega-se à relação 3.9, onde $E_a(s)$ é a transformada de Laplace de $e_a(t)$ e $\Theta(s)$ é a transformada de Laplace de $\theta(t)$. Pode-se considerar a simplificação de que a indutância da bobina do motor (L_a) é desprezível, chegando-se a um modelo smplificado de um sistema de segunda ordem, expresso na equação 3.10. Nesta equação, $K_0 = K_t/R_a$, $B = b_0 + K_v K_t/R_a$ e $\tau = J_0/B$

$$L_a \frac{di_a}{dt} + R_a i_a + K_v \frac{d\theta}{dt} = e_a \tag{3.7}$$

$$J_0 \frac{d^2\theta}{dt^2} + b_0 \frac{d\theta}{dt} = K_t i_a \tag{3.8}$$

$$\frac{\Theta(s)}{E_a(s)} = \frac{K_t}{s(L_a s + R_a)(J_0 s + b_0) + K_t K_v s} \tag{3.9}$$

$$\frac{\Theta(s)}{E_a(s)} = \frac{K_0}{s(J_0 s + B)} = \frac{K_0/B}{s^2 J_0/B + s} = \frac{K}{\tau s^2 + s} \tag{3.10}$$

Foram feitos experimentos para descobrir o valor de K e τ do motor. Eles foram baseados no fato de que a relação entre a velocidade angular ω_n e a tensão de armadura e_a é um sistema de primeira ordem. Assim, ao aplicar-se um degrau de tensão E, a saída deve ter a forma da equação 3.11. Dessa forma, o valor de regime será KE, e o tempo necessário para atingir $(1 - e^{-1}) \approx 63\%$ desse valor é numericamente igual a τ.

$$\omega_n(t) = (1 - e^{-\frac{t}{\tau}})KE \tag{3.11}$$

O detalhamento dos experimento pode ser visto na subseção 3.3.2, e os resultados no apêndice C.1. Os valores obtidos foram de $K = 37/255\ ticks/ms$ e $\tau = 14\ ms$, sendo cada $tick$ igual a um milésimo de volta.

Deve-se ressaltar que essa modelagem não leva em conta a carga do motor. Essa carga implica em uma inércia maior a ser vencida pelo torque do motor. No entanto, essa inércia é reduzida pela taxa de redução da transmissão, que é de 162. Pode-se esperar, assim, um efeito marginal da carga do motor no comportamento dinâmico, desde que essa carga não seja muito elevada. Como também pode haver variação desta carga, optou-se por excluí-la do modelo. Seu efeito aparecerá, então, como um ruído. Devido a isso, o controle deve incluir um elemtno integrador para anular o erro em regime.

3.2.5.2 Especificação do Controle

O objetivo do controle é manter a velocidade de movimentação do êmbolo da seringa constante e igual a um valor pre-ajustado. Para isso, será feito controle de malha fechada sobre a rotação do eixo do motor. O diagrama da figura 25 ilustra essa situação.

Os requisitos para o sistema controlado são de que o tempo de acomodação seja menor que 100 ms (critério de 2%), e que não haja sobressinal.

3.2.6 Software

3.2.6.1 Interface com o operador

Ao utilizar a plataforma de testes, o operador deve ter como objetivo ou produzir um fluxo controlado de algum fluido ou acompanhar uma ou mais medidas ou ambos. Dessa forma, a interface do computador deve disponibilizar:

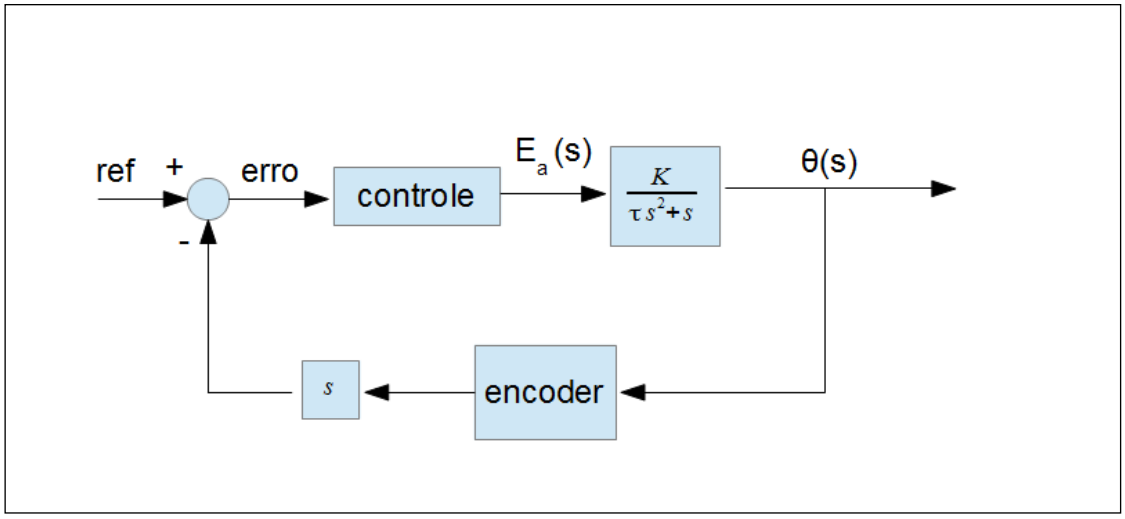

Figura 25: Diagrama de blocos do sistema de controle.

1. Uma forma de inicializar o sistema, incluindo a configuração que será utilizada, sensores que estão conectados e a posição inicial da seringa.

2. A possibilidade de ajustar a velocidade (ou fluxo) e o curso total (ou volume) produzidos pelo movimento do motor.

3. Uma maneira de acompanhar o valor instantâneo dos sensores conectados.

4. Uma maneira de programar a taxa de amostragem dos sensores e gerar arquivos de tendência com esses valores ao longo de um tempo ajustável.

Como a placa Arduino™é alimentada pelo cabo USB, não é esperado que essa siga funcionando no caso de uma acidental desconexão. No entanto, a interface deve detectar tal desconexão e alertar o operador. Após a reconexão o sistema não deve mover-se.

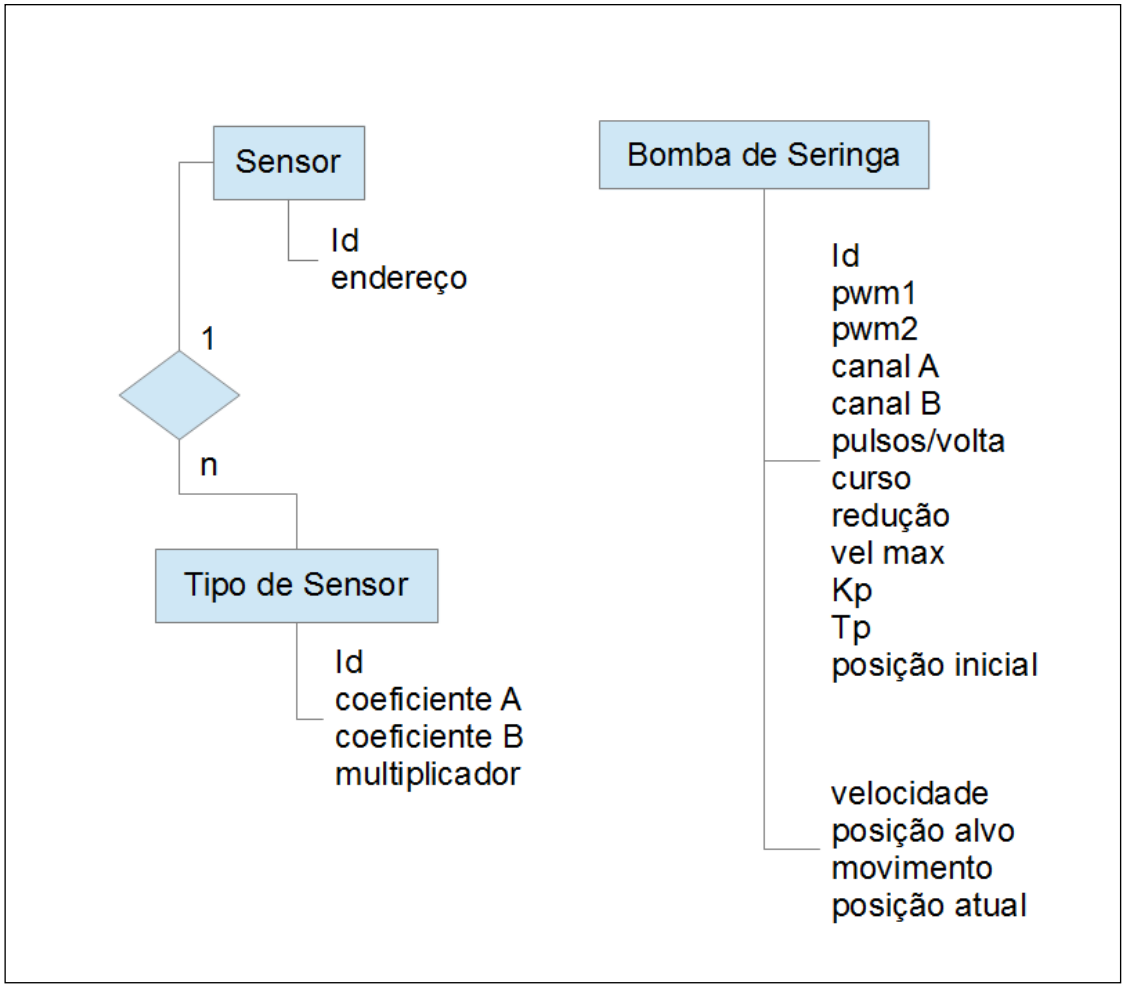

Figura 26: Modelo entidade relacionamento das informações necessárias à operação da plataforma de testes microfluídicos.

3.2.6.2 Estrutura de dados

Para realizar as funções de teste, a placa Arduino™deve conter algumas informações sobre o sistema que está controlando. A figura 26 mostra um diagrama MER dessas informações.

As escolhas do que seria fixo e do que seria variável foram baseadas no requisito de que podem ser adicionados novos sensores, ou os atuais trocados ou até haver a adição de mais bombas de seringa no futuro, no intuito de aperfeiçoar e adequar a plataforma a algum projeto no qual seja utilizada. Dessa forma, cada sensor possui um "tipo", que contém as informações de

coeficientes da curva do sensor e multiplicador utilizado para comunicar o valor, e esse "tipo"pode ser utilizado por outros sensores acoplados ao sistema, ou até serem criados novos tipos para outros sensores. No caso da bomba de seringa, todas as informações que caracterizam o motor e a bomba podem ser ajustados, e podem ser programados novos valores no caso da inclusão de uma nova bomba. Todas essas medidas garantem a versatilidade da plataforma, já que não será necessário modificar seu software para se adequar a um novo *layout*. O oposto disto seria programar os valores utilizados para todas as grandezas no próprio software, deixando-os "*hard coded*".

3.3 Construção

3.3.1 Bomba de Seringa

Os suportes para o motor e para a redução foram construídos de acordo com os desenhos das figuras 7 e 8. Foi utilizada uma serra manual e uma furadeira de bancada. Como a tolerância é relativamente larga em várias cotas, pôde-se observar uma certa variação com o desenho. No entanto, essa variação não compromete o funcionamento da plataforma.

Os mancais e o carro da transmissão foram construidos de acordo com o desenho da figura 6, e a união rígida de acordo com o desenho da figura 5. Foi utilizada uma serra manual e uma furadeira de bancada.

3.3.2 Circuitos Impressos e Cabos

Para a construção dos circuitos impressos optou-se por um processo manual, com impressão dos *layouts* em folha de papel glossy, transcrição para a placa por meio de uma fonte de calor e posterior corrosão do cobre.

Foram feitas três tentativas até alcançar o resultado final. Na primeira vez, a impressão ocorreu corretamente porém não havia sido levada em conta a inversão da placa. Na segunda tentativa o papel foi superaquecido e derreteu, inutilizando a placa. Na terceira vez tudo ocorreu conforme o planejado.

Após a corrosão e a soldagem, foi feito um teste de conexões em todos os pinos de todas as placas, que não indicou nenhum defeito. Assim, foi aplicada uma camada de verniz geral para proteção e os circuitos impressos estavam prontos. Os cabos foram construidos com barramentos de pinos, testados para curto-circuitos e aprovados.

Com todos os componentes eletrônicos prontos, foram feitos dois testes de leitura dos sensores para validar o circuito. Primeiramente, utilizou-se o programa do apêndice B.3 para verificar a leitura do sensor de pressão. Uma seringa foi acoplada à entrada do sensor, e pôde-se observar uma variação proporcional ao aperto desta seringa. Em seguida, utilizou-se o programa do apêndice B.4 para verificar a leitura do sensor de temperatura. Pôde-se observar uma variação positiva da temperatura ao manter o sensor em contato com a pele humana, e uma variação negativa ao soltá-lo. Dessa forma, o circuito cumpre sua função corretamente.

Também foi realizado um teste para o acionamento do motor da bomba de seringa, aproveitando-se para medir seu desempenho dinâmico. O experimento consiste de executar o programa do apêndice B.5, que induz a movimentação do motor com amplitude máxima (pwm de 100 %) e grava a leitura da posição angular e número de voltas. Após um certo número de voltas (cada experimento utilizou um número diferente), os dados são enviados para o computador. Devido às limitações de memória, em alguns casos

a informação recebida pelo computador estava incompleta. Assim foram experimentados casos em busca de maior número de dados sem perdas. Os resultados se ajustam bem a uma curva do tipo $\omega_n(t) = (1 - e^{-\frac{t}{\tau}})KE$ (correlação maior que 0,95 em todos os testes), então foram registrados os valores de KE (valor final da curva) e tempo necessário para atingir $63,2\%$ (τ), $86,5\%$ (2τ), $95,0\%$ (3τ), e $98,2\%$ (4τ) de KE. Esses valores estão no apêndice C.1.

3.3.3 Montagem

Todos os elementos foram fixados em uma base de MDF para suporte. Durante a montagem, priorizou-se o alinhamento dos eixos da transmissão (motor-redução e redução-fuso). As placas de circuito impresso foram dispostas da seguinte forma: a placa de potência próxima ao motor, a placa adaptadora ao lado desta e a placa de sensores ao lado da adaptadora, fazendo com que as placas formem um "L". A disposição final pode ser vista na figura 27.

3.3.4 Software

3.3.4.1 Comunicação

A versão final da classe CommunicatorController, gerada para o computador, encontra-se no apêndice B.6.

Para a realização do teste de uso simples, foi utilizado o programa do apêndice B.7. Os resultados encontram-se no apêndice C.2. Nota-se que no início duas mensagens não são respondidas, e no resto dos testes tudo ocorre bem. O problema das duas mensagens iniciais já foi observado inclusive quando se grava a placa, e notou-se que ocorre somente no início de uma comunicação. Pode estar relacionado aos pinos RTS e DTR do FT232RL, que estão

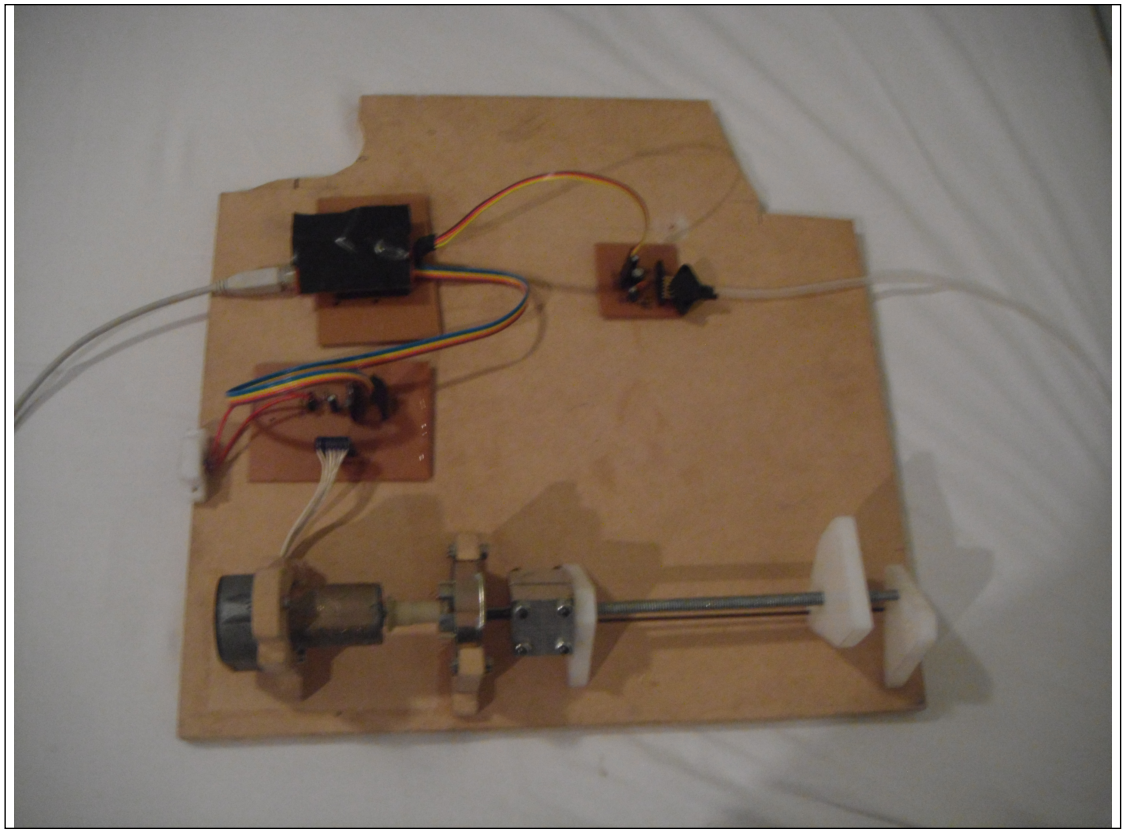

Figura 27: Plataforma de testes pronta.

conectados ao RESET do ATmega168.

Para a realização do teste de uso repetido, foi utilizado o programa do apêndice B.8. Os resultados encontram-se no apêndice C.3. Novamente foi observado o problema das duas primeiras mensagens, porém fora isso todos os resultados estão conformes. Nesse momento, a classe foi modificada para que a função de inicialização mande uma mensagem inicial, assim fazendo com que o problema de não responder as duas primeiras requisições não afete o resto da comunicação. Também foram incluídas as variáveis booleanas que controlam o modo de DEBUG, a sabotagem do CRC e a sabotagem do endereço do registro, para realização dos testes de uso incorreto.

Para a realização do último teste, o de uso incorreto, foi utilizado o programa do apêndice B.9.Os resultados encontram-se no apêndice C.4. Na primeira tentativa, todos os casos retornavam *"commSuccess=false"*. Esse erro foi corrigido, para que esse valor somente seja falso se não hover resposta do escravo. Quando há uma resposta, porém com erro, o valor de *commSuccess* é verdadeiro, porém o de *executionSuccess* é falso. Após essa modificação o teste foi realizado novamente e foram obtidos os resultados que estão no apêndice C.4.

Como, na versão final, todos os casos de teste obtiveram êxito, os programas criados podem ser utilizados para realizar a comunicação entre o computador e a placa Arduino™.

3.3.4.2 Programa do microcontrolador

O programa para o microcontrolador encontra-se no apêndice B.10. Este programa basea-se no mapeamento das informações utilizadas pela plataforma de testes para funcionar em registradores, de acordo com a tabela do apêndice A.

Como discutido na sessão 3.2.5.1, é necessário um componente integrador no controle. Optou-se, então, por um controle PI, conforme a figura 28, com Kp=1 e Tp=0,014. Com esses valores, chega-se a um valor de largura de banda de $\omega_{LB} = 1,41K_p$. Utilizando o critério de frequencia de amostragem maior que 50 vezes a largura de banda, chega-se ao valor $T_a = 0,089K_p$. Foi utilizada a biblioteca *Arduino PID Library* (disponível em http://playground.arduino.cc/Code/PIDLibrary) para cálculo do esforço de controle.

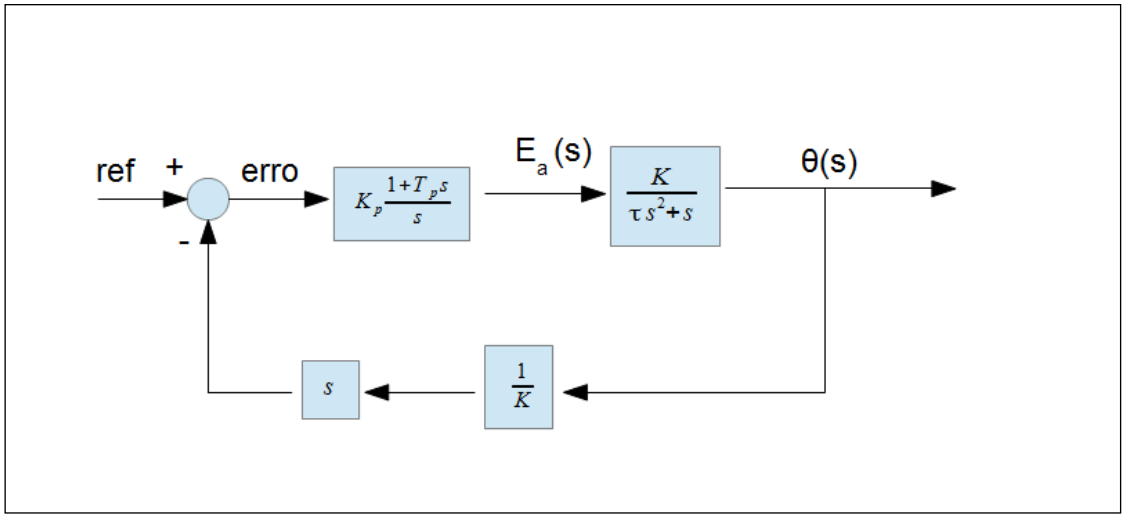

Figura 28: Diagrama de blocos do sistema final.

3.3.4.3 Aplicativo para o computador

O programa da interface com o operador encontra-se no apêndice B.11. Todas as informações da tabela do apêndice A podem ser enviadas ao microcontrolador através dessa interface.

3.4 Teste da Plataforma

Para testar se a plataforma desenvolvida pode ser utilizada em verificações e experimentos com dispositivos microfluídicos, foi construído um protótipo fictício utilizando a técnica de fabricação de litografia macia em polidimetilsiloxano (PDMS), um polímero utilizado para prototipagem rápida na área de microfluídica (MCDONALD; WHITESIDES, 2002). O molde, usinado em alumínio de maneira convencional, gerou dois canais de 1 mm de expessura, 1 mm de profundidade e 5 cm de comprimento. Para selar os canais, foi utilizada uma tampa também de PDMS. A união das duas metades foi feita com a técnica de cura parcial (EDDINGS; JOHNSON; GALE, 2008).

A plataforma foi carregada com as configurações dos sensores e do motor obtidas neste trabalho, e pôde-se observar tanto o controle de velocidade da bomba de seringa quanto a coleta dos valores dos sensores ocorrendo simultaneamente. Desta forma, a plataforma de testes funcionou em um teste real.

Durante os testes, foi observado o fenômeno de *stick-slip* na seringa, o que gera descontinuidades no fluxo. Isso pode ser amenizado com uma fixação mais rígida da seringa, ajuste de velocidades maiores e utilização de seringas com menor atrito entre o êmbolo e as paredes da seringa. Um outro ponto que pode ser melhorado em futuras versões da plataforma é a introdução da gravação dos valores obtidos nos sensores com o tempo, afim de se utilizar esses dados em pesquisas.

4 CONCLUSÕES

Foi apresentado o projeto de uma plataforma de testes para dispositivos de microfluídica com ênfase em aparelhos biomédicos. Essa plataforma se insere no contexto de facilitadora para o avanço da tecnologia no País. Apesar de a área de testes e verificações em alguns casos ser negligenciada, ela é essencial na busca por qualidade e, por tanto, competitividade e avanço.

Além de uma verificação individual de cada componente que compõe a plataforma de testes, foi possível simular o teste de um dispositivo microfluídico em que todos os elementos funcionaram corretamente em conjunto.

As características finais da plataforma de testes são: sensor de pressão de até 50 kPa e precisão de 2,5%, sensor de temperatura de até 150 ºC com precisão de 0,5 ºC, bomba de seringa com velocidade máxima de $400\ \mu m/s$, interface no computador para controle e monitoramento.

Alguns pontos de melhoria foram observados, como a possibilidade de gravação dos valores lidos nos sensores com o tempo. Mesmo assim, a plataforma já pode ser utilizada com o suporte para pesquisas, desenvolvimentos ou testes na área de microfluídica.

REFERÊNCIAS

CHANG, C.-P.; NAGEL, D. J.; ZAGHLOUL, M. E. Formation and status of the mems microfluidcs industry. *ICSE2008 Proc.*, 2008.

EDDINGS, M. A.; JOHNSON, M. A.; GALE, B. K. Determining the optimal pdms-pdms bonding technique for microfluidic devices. *J. Micromech. Microeng.*, v. 18, 2008.

FOX; MCDONALD. *Introduction to Fluid Mechanics*. [S.l.]: New York: Wiley, 1999.

GRAVESEN, P.; BRANEBJERG, J.; JENSEN, O. S. Microfluidics-a review. *J. Micromech. Microeng.*, v. 3, p. 168–182, 1993.

GRAYSON, A. C. R. et al. A biomems review: Mems technology for physiologically integrated devices. *PROCEEDINGS OF THE IEEE*, VOL. 92, 2004.

INCROPERA, F. P. et al. *Fundamentos de transferÃªncia de calor e de massa*. [S.l.]: LTC, 2008.

KERKHOFF, H. Testing microelectronic biofluidic systems. *Design Test of Computers, IEEE*, v. 24, n. 1, p. 72 –82, jan.-feb. 2007. ISSN 0740-7475.

KERKHOFF, H. et al. Vhdl-ams fault simulation for testing dna bio-sensing arrays. In: *Sensors, 2005 IEEE*. [S.l.: s.n.], 2005. p. 4 pp.

KIM, M. S. et al. Quantitative proteomic profiling of breast cancers using a multiplexed microfluidic platform for immunohistochemistry and immunocytochemistry. *Biomaterials*, v. 32, n. 5, p. 1396 – 1403, 2011. ISSN 0142-9612. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0142961210013475>.

KOSINSKI, R. J. A literature review on reaction time. *Clemson University.*, 2012.

LAUGA, E.; BRENNER, M. P.; STONE, H. A. Microfluidics: The no-slip boundary condition. *Cap. 19 em: Handbook of Experimental Fluid Dynamics*, 2007.

LOK, K. S.; KWOK, Y. C.; NGUYEN, N.-T. Double spiral detection channel for on-chip chemiluminescence detection. *Sensors and Actuators B: Chemical*, v. 169, n. 0, p. 144 – 150, 2012. ISSN 0925-4005. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925400512004078>.

MCDONALD, J. C.; WHITESIDES, G. M. Poly(dimethylsiloxane) as a material for fabricating microfluidic devices. *Accounts of Chemical Research*, v. 35, p. 491–499, 2002.

NGUYEN, N.-T. *Fundamentals and Applications of Microfluidics*. [S.l.]: Norwood, MA, USA, 2002.

PHAN, V.-N. et al. Fabrication and experimental characterization of nanochannels. *Journal of Heat Transfer*, ASME, v. 134, n. 5, p. 051012, 2012. Disponível em: <http://link.aip.org/link/?JHR/134/051012/1>.

ROTHER, M. *Toyota Kata*. [S.l.: s.n.], 2010.

SHAEGH, S. A. M.; NGUYEN, N.-T.; CHAN, S. H. Air-breathing microfluidic fuel cell with fuel reservoir. *Journal of Power Sources*, v. 209, n. 0, p. 312 – 317, 2012. ISSN 0378-7753. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0378775312005393>.

SHAH, R. K.; LONDON, A. L. Laminar flow forced convection in ducts. *New York: Academic*, 1978.

APÊNDICE A – TABELA DE REGISTRADORES

Registrador	Informação	Descrição
0	parada de emergência	indica se há funcionamento normal (zero) ou deve haver uma parada de emergência (valor diferente de zero)
1 a 20	valor do sensor	contém o valor do sensor com o respectivo id
21	gravar tipo	indica se já foram preenchidas as informações para o novo tipo de sensor a ser gravado (registradores 22 a 25)
22	id tipo	id do novo tipo de sensor a ser gravado
23	parâmetro A tipo	coeficiente angular da reta de calibração do novo tipo de sensor
24	parâmetro B tipo	*offset* da reta de calibração do novo tipo de sensor
25	multiplicador tipo	multiplicador a ser aplicado para comunicar o valor da grandeza medida pelo novo tipo de sensor
26	não utilizado	reservado para futuras modificações

Tabela 1: Mapeamento de informações e registradores do microcontrolador.

Registrador	Informação	Descrição
27	gravar sensor	indica se já foram preenchidas as informações para o novo sensor a ser gravado (registradores 28, 29 e 30)
28	id sensor	id do novo sensor a ser gravado)
29	tipo sensor	id do tipo do novo sensor a ser gravado)
30	endereço sensor	endereço (porta analógica) do novo sensor a ser gravado
31	não utilizado	reservado para futuras modificações
32	gravar motor	indica se já foram preenchidas as informações para o novo motor a ser gravado (registradores 33 a 44)
33	id motor	id do novo motor a ser gravado
34	pwm1	endereço (porta digital) do sinal de pwm1 do novo motor a ser gravado
35	pwm2	endereço (porta digital) do sinal de pwm2 do novo motor a ser gravado
36	canal A	endereço (porta digital) do sinal do canal A do encoder do novo motor a ser gravado
37	canal B	endereço (porta digital) do sinal do canal B do encoder do novo motor a ser gravado
38	pulsos/ volta	quantidade de pulsos por volta do encoder do novo motor a ser gravado
39	curso	curso total do carro do novo motor a ser gravado (em micrômetros)
40	reducao	reducao total do novo motor a ser gravado (micrômetros/volta)
41	velocidade máxima	velocidade máxima do carro do novo motor a ser gravado (pulsos/milisegundo)
42	Kp	constante proporcional do controle do novo motor a ser gravado
43	Tp	constante que define o controle integrativo do novo motor a ser gravado

Tabela 2: Mapeamento de informações e registradores do microcontrolador (continuação).

Registrador	Informação	Descrição
44	posição inicial	posição inicial do carro do novo motor a ser gravado (em micrômetros)
45	não utilizado	reservado para futuras modificações
46	não utilizado	reservado para futuras modificações
47	gravar velocidade	indica se já foram preenchidas as informações para a nova velocidade de algum motor (registradores 48 e 49)
48	id motor	id do motor que terá sua velocidade modificada
49	velocidade	velocidade nova a ser gravada (0 a 255, sendo 255 equivalente à velocidade máxima)
50	não utilizado	reservado para futuras modificações
51	gravar posição alvo	indica se já foram preenchidas as informações para a nova posição alvo de algum motor (registradores 52 e 53)
52	id motor	id do motor que terá a posição alvo modificada
53	posição alvo	nova posição alvo a ser gravada (em micrômetros)
54	gravar movimentação	indica se já foram preenchidas as informações para o novo valor de movimentação de algum motor (registradores 55 e 56)
55	id motor	id do motor que terá o valor de movimentação modificado
56	movimentação	novo valor de movimentação a ser gravado (zero indica parado, qualquer outro valor indica para ocorrer movimento)
57	não utilizado	reservado para futuras modificações
58	id ler posição	id do motor que terá su aposição lida
59	pronto para ler posição	indica se o valor de posição do motor a ser lido já foi gravado no registrador 60
60	posição	posição do motor (em micrômetros)

Tabela 3: Mapeamento de informações e registradores do microcontrolador (continuação).

APÊNDICE B – PROGRAMAS UTILIZADOS

B.1 Programa utilizado para quantificar a qualidade do sinal do sensor de temperatura LM35

```
#define cteA 0.48828 // (5 / 10,24) ºC/ADU

#define margin 0.5

int sensorPin = A0;     // pino conectado ao sensor
int sensorValue = 0;  // variable to store the value coming from the
    sensor
int timeoff = 200;
float temperature = 0;
float lastTemperature = 1;
unsigned long lastMillis = 0;
unsigned long maxMillis = 0;

void setup() {
   Serial.begin(9600); //Comunicacao serial a 9600bps

 lastMillis = millis();
  Serial.println("start");
}

void loop() {
```

```
21    sensorValue = analogRead(sensorPin);

23    temperature = (float)sensorValue cteA;


25

    if(temperature > lastTemperature + margin || temperature <
        lastTemperature - margin) {
27      lastTemperature = temperature;
      if(millis()-lastMillis > maxMillis)
29        maxMillis = millis()-lastMillis;

31      Serial.print(temperature);
      Serial.print(" : ");
33      Serial.print(millis()-lastMillis);
      Serial.print(" , max: ");
35      Serial.println(maxMillis);
      lastMillis = millis();
37    }

39  }
```

filtrotemp.cpp

B.2 Programa utilizado para quantificar a qualidade do sinal do sensor de pressão MPX5050DP

```
1   #define cteA 0.05425
    #define cteB -2.222
3
    #define margin 0.1
5
    int sensorPin = A0;     // pino conectado ao sensor
```

```
int sensorValue = 0;  // variable to store the value coming from the
    sensor
float pressure = 0;
float lastPressure = 1;
unsigned long lastMillis = 0;
unsigned long maxMillis = 0;

void setup() {
   Serial.begin(9600); //Comunicacao serial a 9600bps

 lastMillis = millis();
  Serial.println("start");
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  pressure = (float)sensorValue cteA + cteB;

  if(pressure > lastPressure + margin || pressure < lastPressure -
     margin) {
    lastPressure = pressure;
    if(millis()-lastMillis > maxMillis)
      maxMillis = millis()-lastMillis;

    Serial.print(pressure);
    Serial.print(" : ");
    Serial.print(millis()-lastMillis);
    Serial.print(" , max: ");
    Serial.println(maxMillis);
    lastMillis = millis();
  }
}
```

filtropress.cpp

B.3 Programa utilizado para testar o circuito do sensor de pressão

```
#define cteA 0.05425 // calculated based on datasheet
#define cteB -2.222 // calculated based on datasheet

#define margin 0.1

int sensorPin = A1;     // input pin
int sensorValue = 0;  // variable to store the value coming from the
    sensor
float pressure = 0;

void setup() {
   Serial.begin(9600); //Comunicacao serial a 9600bps
  Serial.println("start");
}

void loop() {
  sensorValue = analogRead(sensorPin);
  pressure = (float)sensorValue cteA + cteB;

  if(pressure > lastPressure + margin || pressure < lastPressure -
      margin) {
    lastPressure = pressure;

    Serial.print(pressure);
  }
}
```

testsen1.cpp

B.4 Programa utilizado para testar o circuito do sensor de temperatura

```
#define 0.48828 // (5 / 10,24) ºC/ADU calculated based on datasheet

#define margin 10

int sensorPin = A0;     // input pin
int sensorValue = 0;  // variable to store the value coming from the
    sensor
float temperature = 0;

void setup() {
   Serial.begin(9600); //Comunicacao serial a 9600bps
  Serial.println("start");
}

void loop() {
  sensorValue = analogRead(sensorPin);
  temperature = (float)sensorValue cteA;

  if(temperature > lastTemperature + margin || temperature <
      lastTemperature - margin) {
    lastTemperature = temperature;

    Serial.print(temperature);
  }
}
```

testsen2.cpp

B.5 Programa utilizado para modelar o motor dinamicamente

```
#define encoder0PinA  2 // pino do canal A do encoder
#define encoder0PinB  4 // pino do canal B do encoder

#define pulsosPorVolta 1000 // numero de pulsos lidos no encoder por
    volta
#define encoderBreak 50 // apenas exibir valor do encoder de tantos
   em tantos passos

int encoder0Pos = 0; // valor lido pelo encoder
int encoder0LastPos = 1; // ultimo valor do encoder exibido na tela

int voltas = 0; // numero de voltas completas
int voltasADar = 5; // aciona o motor por X voltas quando diferente
   de zero

int pwm1 = 3; // pino in 1 do PWM
int pwm2 = 5; // pino in 2 do PWM

int pwmValue = 255; // valor do pwm (0 a 255)

String dados = "";

void setup() {
  Serial.begin(9600); //Comunicacao serial a 9600bps
  analogReference(DEFAULT); //Fundo de escala para sinal analogico

  pinMode(pwm1, OUTPUT); //Saida pwm para um terminal do motor
  pinMode(pwm2, OUTPUT); //Saida pwm para o outro terminal do motor

  pinMode(encoder0PinA, INPUT);
```

```
  digitalWrite(encoder0PinA, HIGH);       // turn on pullup resistor
  pinMode(encoder0PinB, INPUT);
  digitalWrite(encoder0PinB, HIGH);       // turn on pullup resistor

  attachInterrupt(0, doEncoder, CHANGE);  // encoder pin on
      interrupt 0 - pin 2

  Serial.println("start");
}
void loop() {

  lerEncoder();

    acionarPWM();

  if(voltasADar > voltas){
    Serial.println(dados);
  }
}

void acionarPWM() {


  if(voltasADar==voltas) {

    analogWrite(pwm1,255);
    analogWrite(pwm2,255);

  } else {

    analogWrite(pwm1,0);
    analogWrite(pwm2,0);

```

```
    if(voltasADar > voltas){

      analogWrite(pwm2,pwmValue);

    } else {

      analogWrite(pwm1,pwmValue);

    }
  }

}

void lerEncoder() {

 if(encoder0LastPos < (encoder0Pos-encoderBreak) ||
  encoder0LastPos > (encoder0Pos+encoderBreak)) {

    // Serial.print("Encoder: ");
    // Serial.print(voltas);
    // Serial.print(" - ");
    // Serial.println(encoder0Pos);
  dados += voltas;
  dados += ";";
  dados += encoder0Pos;
  dados += "\n";

    if(encoder0Pos > pulsosPorVolta) {
      voltas++;
      encoder0Pos -= pulsosPorVolta;
    } else if (encoder0Pos < -pulsosPorVolta) {
      voltas--;
      encoder0Pos += pulsosPorVolta;
```

```
93      }

95      encoder0LastPos = encoder0Pos;
    }
97  }

99  void doEncoder() {
    /  If pinA and pinB are both high or both low, it is spinning
101     forward. If they're different, it's going backward.

103     For more information on speeding up this process, see
        [Reference/PortManipulation], specifically the PIND register.
105   /
    if (digitalRead(encoder0PinA) == digitalRead(encoder0PinB)) {
107   encoder0Pos++;
    } else {
109   encoder0Pos--;
    }
111

    }
```

testmot.cpp

B.6 Classe CommunicationController implementada em java

```
    package tcc_gui;
2
    import java.util.EnumMap;
4   import java.util.Map;
    import gnu.io.CommPort;
6   import gnu.io.CommPortIdentifier;
    import gnu.io.SerialPort;
```

```
8
import java.io.InputStream;
10 import java.io.OutputStream;

12 public class CommunicationController {

14     private boolean DEBUG = false;
       private boolean SIMPLE_TEST = false;
16     private boolean SABOTAGE_CRC = false;
       private boolean SABOTAGE_REGISTER_ADDRESS = false;
18
       /
20         @return the DEBUG
         /
22     public boolean isDEBUG() {
           return DEBUG;
24     }

26     /
          @param DEBUG the DEBUG to set
28       /
       public void setDEBUG(boolean DEBUG) {
30         this.DEBUG = DEBUG;
       }
32
       /
34         @return the SIMPLE_TEST
         /
36     public boolean isSIMPLE_TEST() {
           return SIMPLE_TEST;
38     }

40     /
```

```
     @param SIMPLE_TEST the SIMPLE_TEST to set
   */
  public void setSIMPLE_TEST(boolean SIMPLE_TEST) {
      this.SIMPLE_TEST = SIMPLE_TEST;
  }

  /
     @return the SABOTAGE_CRC
   */
  public boolean isSABOTAGE_CRC() {
      return SABOTAGE_CRC;
  }

  /
     @param SABOTAGE_CRC the SABOTAGE_CRC to set
   */
  public void setSABOTAGE_CRC(boolean SABOTAGE_CRC) {
      this.SABOTAGE_CRC = SABOTAGE_CRC;
  }

  /
     @return the SABOTAGE_REGISTER_ADDRESS
   */
  public boolean isSABOTAGE_REGISTER_ADDRESS() {
      return SABOTAGE_REGISTER_ADDRESS;
  }

  /
     @param SABOTAGE_REGISTER_ADDRESS the
      SABOTAGE_REGISTER_ADDRESS to set
   */
  public void setSABOTAGE_REGISTER_ADDRESS(boolean
     SABOTAGE_REGISTER_ADDRESS) {
```

```
72          this.SABOTAGE_REGISTER_ADDRESS = SABOTAGE_REGISTER_ADDRESS;
    }
74
    public enum Register {
76
        REG0,REG1,REG2,REG3,REG4,REG5
78  };
    public static int SUCCESS = 0;
80  public static int CRC_ERROR = -3;
    public static int TIMEOUT_ERROR = -2;
82  private static int NUMBER_OF_RETRIES = 5;
    private static long FRAME_TIMEOUT = 500;
84  private Map<Register, Integer> addressMap = new EnumMap<Register
        , Integer>(Register.class);
    private String commPortName = null;
86  private SerialPort serialPort = null;
    private InputStream inputSerialStream = null;
88  private OutputStream outputSerialStream = null;
    private String sync = new String();
90  private int baudRate = 9600;
    private int parity = SerialPort.PARITY_EVEN;
92  private int databits = SerialPort.DATABITS_8;
    private int stopbits = SerialPort.STOPBITS_1;
94
    public CommunicationController() {
96      addressMap.put(Register.REG0, 0);
        addressMap.put(Register.REG1, 1);
98      addressMap.put(Register.REG2, 2);
        addressMap.put(Register.REG3, 3);
100     addressMap.put(Register.REG4, 4);
        addressMap.put(Register.REG5, 5);
102 } // constructor
```

```
104     public void configure(int baud,int parity, int databits, int
            stopbits) {
            this.baudRate = baud;
106         this.parity = parity;
            this.databits = databits;
108         this.stopbits = stopbits;
        } // configure
110
        public void init(String commPort) throws Exception {
112         this.commPortName = commPort;
            connect(this.commPortName);
114
             this.readRegister(1, Register.REG0, 1); // send something
                to heat up the line.
116                                                  // also, the
                                                        arduino board
                                                        will probably
                                                        reset so its
                                                        good to wait for
                                                         it to
                                                        initialize ?

118     } // init

120     public void end() throws Exception {
            this.disconnect();
122     }

124     public CommReturn writeRegister(int destination, Register
            register, int value) throws Exception {
            CommReturn ret = null;
126         synchronized (sync) {
```

```
            byte[] frame = buildWriteRegisterFrame(destination,
                register, value);
128         ret = sendRequestGetResponse(frame);
        }
130     return ret;
    } // writeRegister
132
    public CommReturn readRegister(int destination, Register
        register, int numberOfRegisters) throws Exception {
134     CommReturn ret = null;
        synchronized (sync) {
136         byte[] frame = buildReadRegisterFrame(destination,
                register, numberOfRegisters);
            ret = sendRequestGetResponse(frame);
138     }
        return ret;
140 } // readRegister

142 private CommReturn sendRequestGetResponse(byte[] frame) throws
        Exception {
        CommReturn ret = new CommReturn();
144     ret.commSuccess = false;
        ret.exceptionCode = TIMEOUT_ERROR; // timeout
146     for (int retries = 0; retries < NUMBER_OF_RETRIES; retries
            ++) {
            sendRequest(frame);
148         ret = getResponse();
            if (ret.commSuccess) {
150             if (isDEBUG()) {
                    System.out.println("RET.SUCCESS = TRUE");
152             }
                if (checkCRC(ret)) {
154                 if (isDEBUG()) {
```

```
                    System.out.println("CRC OK !");
                }
                if (ret.function > 0x80) {
                    ret.executionSuccess = false; // error //[ju
                        :20121106] changed commSuccess to
                        executionSuccess
                }
                break;
            } else {
                ret.commSuccess = false;
                ret.exceptionCode = CRC_ERROR;
            }
        } else {
            if (isDEBUG()) {
                System.out.println("Timeout");
            }
        }
    } // for retries
    return ret;
} // sendRequestGetResponse

private void sendRequest(byte[] frame) throws Exception {
    if (isDEBUG() || isSIMPLE_TEST()) {
        System.out.print("trying to send: ");
        for (int i = 0; i < frame.length; i++) {
            System.out.print(Integer.toHexString(Integer.
                parseInt(Byte.toString(frame[i])) & 0x00FF) + " "
                );
        }
        System.out.println();
    }
    outputSerialStream.write(frame);
    outputSerialStream.flush();
```

```
184         if (isDEBUG()) {
                System.out.println("sent ok !");
186         }
        } // sendRequest
188
        private CommReturn getResponse() throws Exception {
190         CommReturn ret = new CommReturn();
            byte[] buffer = new byte[1024];
192         int idx = 0;
            Timer timer = new Timer(FRAME_TIMEOUT);
194         timer.start();

196         boolean found = false;
            if (isDEBUG()|| isSIMPLE_TEST()) {
198             System.out.println("waiting response !");
            }
200         while ((!found) && (!timer.timedOut())) {
                if (inputSerialStream.available() > 0) {
202                 int chInt = inputSerialStream.read();
                    if (isDEBUG() || isSIMPLE_TEST()) {
204                     System.out.print(" " + Integer.toHexString(chInt
                            ));
                    }
206                 if (chInt != -1) {
                        buffer[idx] = (byte) chInt;
208                     switch (idx) {
                            case 1:
210                             ret.function = buffer[1] & 0x00FF;
                                if (isDEBUG()) {
212                                 System.out.println("function : " +
                                        Integer.toHexString(ret.function)
                                        );
                                }
```

```
214                     break;

216                 case 2:
                        if (ret.function > 0x80) {
218                         ret.exceptionCode = buffer[2];
                        }
220                     break;

222                 case 4:
                        if (ret.function > 0x80) { // error
224                         if (isDEBUG()) {
                                System.out.println("error
                                    function received ");
226                         }
                            ret.size = 3;
228                         ret.data = buffer;
                            ret.value = null;
230                         ret.commSuccess = true;
                            ret.executionSuccess = false;
232                         found = true;
                        }
234                     break;

236                 case 7:
                        if (ret.function == 6) {
238                         ret.size = 6;
                            ret.data = buffer;
240                         ret.value = new int[1];
                            ret.value[0] = ((buffer[4] & 0x00FF)
                                << 8) | (buffer[5] & 0x00FF);
242                         ret.commSuccess = true;
                            ret.executionSuccess = true;
244                         found = true;
```

```
                    }
                    break;

                default:
                    if (ret.function == 3) {
                        if (idx == (buffer[2] & 0x00FF) + 4)
                            {
                            ret.size = idx - 1;
                            ret.data = buffer;
                            int totalFields = (buffer[2] & 0
                               x00FF) / 2;
                            ret.value = new int[totalFields
                               ];
                            for (int numberOfField = 0;
                               numberOfField < totalFields;
                               numberOfField++) {
                                ret.value[numberOfField] =
                                    ((buffer[(numberOfField
                                     2) + 3] & 0x00FF) << 8)
                                    | (buffer[(numberOfField
                                      2) + 4] & 0x00FF);
                            }
                            ret.commSuccess = true;
                            ret.executionSuccess = true;
                            found = true;
                        }
                    }
                    break;
            }
            idx++;


        } else {
```

```
                    throw new Exception("Port closed");
                }
            } else {
                Thread.sleep(10);
                if (isDEBUG()) {
                    System.out.print(" -");
                }
            }
        } // while !found and !timeout


        if (timer.timedOut()) {
            ret.data = null;
            ret.value = null;
            ret.size = 0;
            ret.commSuccess = false;
            ret.executionSuccess = false;
            ret.function = 0;
            ret.exceptionCode = TIMEOUT_ERROR; // timeout
        }
        if(isDEBUG() || isSIMPLE_TEST()) {
            System.out.println();
        }

        return ret;
    } // getResponse

    private byte[] buildReadRegisterFrame(int destination, Register
        register, int numberOfRegisters) {
        byte[] frame = new byte[8];
        int function03 = 3;
        int startingAddress = addressMap.get(register);

```

```
        frame[0] = (byte) (destination & 0x00FF);
302     frame[1] = (byte) (function03 & 0x00FF);
        frame[2] = (byte) ((startingAddress >>> 8) & 0x00FF);
304     frame[3] = (byte) (startingAddress & 0x00FF);
        frame[4] = (byte) ((numberOfRegisters >>> 8) & 0x00FF);
306     frame[5] = (byte) (numberOfRegisters & 0x00FF);
        int crc = calculateCRC(frame, 0, 5);
308     frame[6] = (byte) ((crc >>> 8) & 0x00FF);
        frame[7] = (byte) (crc & 0x00FF);
310
        if(SABOTAGE_REGISTER_ADDRESS) {
312         byte[] sabotaged_frame = new byte[frame.length-1];
            sabotaged_frame[0] = frame[0];
314         sabotaged_frame[1] = frame[1];
            sabotaged_frame[2] = frame[2];
316         // one byte of address forgotten ! oops =)
            sabotaged_frame[3] = frame[4];
318         sabotaged_frame[4] = frame[5];
            crc = calculateCRC(sabotaged_frame, 0, 4);
320         sabotaged_frame[5] = (byte) ((crc >>> 8) & 0x00FF);
            sabotaged_frame[6] = (byte) (crc & 0x00FF);
322         return sabotaged_frame;
        }
324
        return frame;
326 } // buildReadRegisterFrame

328 private byte[] buildWriteRegisterFrame(int destination, Register
        register, int value) {
        byte[] frame = new byte[8];
330     int function06 = 6;
        int startingAddress = addressMap.get(register);
332
```

```
        frame[0] = (byte) (destination & 0x00FF);
334     frame[1] = (byte) (function06 & 0x00FF);
        frame[2] = (byte) ((startingAddress >>> 8) & 0x00FF);
336     frame[3] = (byte) (startingAddress & 0x00FF);
        frame[4] = (byte) ((value >>> 8) & 0x00FF);
338     frame[5] = (byte) (value & 0x00FF);
        int crc = calculateCRC(frame, 0, 5);
340     frame[6] = (byte) ((crc >>> 8) & 0x00FF);
        frame[7] = (byte) (crc & 0x00FF);
342
        if(SABOTAGE_REGISTER_ADDRESS) {
344         byte[] sabotaged_frame = new byte[frame.length-1];
            sabotaged_frame[0] = frame[0];
346         sabotaged_frame[1] = frame[1];
            sabotaged_frame[2] = frame[2];
348         // one byte of address forgotten ! oops =)
            sabotaged_frame[3] = frame[4];
350         sabotaged_frame[4] = frame[5];
            crc = calculateCRC(sabotaged_frame, 0, 4);
352         sabotaged_frame[5] = (byte) ((crc >>> 8) & 0x00FF);
            sabotaged_frame[6] = (byte) (crc & 0x00FF);
354         return sabotaged_frame;
        }
356     return frame;
    } // buildWriteRegisterFrame
358
    private void connect(String portName) throws Exception {
360     CommPortIdentifier portIdentifier = CommPortIdentifier.
            getPortIdentifier(portName);
        if (portIdentifier.isCurrentlyOwned()) {
362         if (isDEBUG()) {
                System.out.println("Error: Port is currently in use"
                    );
```

```
364             }
                //@TODO: retry !
366         } else {
                CommPort commPort = portIdentifier.open(this.getClass().
                    getName(), 2000);
368             if (commPort instanceof SerialPort) {
                    serialPort = (SerialPort) commPort;
370                 serialPort.setSerialPortParams(baudRate, databits,
                        stopbits, parity);
                    inputSerialStream = serialPort.getInputStream();
372                 outputSerialStream = serialPort.getOutputStream();
                } else {
374                 if (isDEBUG()) {
                        System.out.println("Error: Only serial ports are
                            handled.");
376                     //@TODO: retry other names !
                    }
378             }
            }
380     } // connect

382     private void disconnect() {
            serialPort.close();
384     } // disconnect

386     private int calculateCRC(byte[] buf, int start, int end) {
            int i, j;
388         int temp, temp2, flag;

390         temp = 0xFFFF;

392         for (i = start; i <= end; i++) {
                temp = (temp ^ (buf[i] & 0x00FF)) & 0xFFFF;
```

```
394
                for (j = 1; j <= 8; j++) {
                    flag = temp & 0x0001;
396                 temp = temp >>> 1 & 0xFFFF;
                    if (flag != 0) {
398                     temp = (temp ^ 0xA001) & 0xFFFF;
                    }
400             }
            }
402
            /  Reverse byte order.  /
404         if (isDEBUG()) {
                System.out.println(Integer.toHexString(temp));
406         }

408
            temp2 = ((temp >>> 8) & 0x00FF);
410         temp = ((temp << 8) | temp2);
            temp &= 0xFFFF;
412         if (isDEBUG() || isSABOTAGE_CRC()) {
                System.out.println("CRC: " + Integer.toHexString(temp));
414         }
            if (isSABOTAGE_CRC()) {
416             temp--;
                temp &= 0xFFFF;
418             System.out.println("SABOTAGED CRC: " + Integer.
                    toHexString(temp));
            }
420
            return (temp);
422
        } // calculateCRC
424
        private boolean checkCRC(CommReturn ret) {
```

```
426         int calculatedCRC = calculateCRC(ret.data, 0, ret.size - 1);
            int rxhi = (ret.data[ret.size] & 0x00FF) << 8;
428         int rxlo = ret.data[ret.size + 1] & 0x00FF;
            int rxCRC = (rxhi | rxlo);
430         if (isDEBUG()) {
                System.out.println("calculate=" + Integer.toHexString(
                    calculatedCRC) + " rx=" + Integer.toHexString(rxCRC))
                    ;
432         }
            if (calculatedCRC == rxCRC) {
434             return true;
            }
436         return false;
        } // checkCRC
438
} // CommunicationController
```

CommunicationController.java

B.7 Programa utilizado para o teste de uso simples da comunicação

```
1  package tcc_gui;
3  /
5     @author juliano
   */
7  public class Tcc_GUI {
9      private static int MAX_LATENCY = 100;
11     /
```

```
     @param args the command line arguments
    */
    public static void main(String[] args) throws Exception {


        CommunicationController communicationController = new
            CommunicationController();
        communicationController.init("/dev/ttyUSB0");

        System.out.println();
        System.out.println("first test:");
        System.out.println();
        Timer t = new Timer(MAX_LATENCY);
        t.start();
        CommReturn ret = communicationController.writeRegister(1,
            CommunicationController.Register.REG0,1);
        System.out.println("timedout ? "+ t.timedOut());
        System.out.println(ret.toString());

        System.out.println();
        System.out.println("second test:");
        System.out.println();
        t = new Timer(MAX_LATENCY);
        t.start();
        ret = communicationController.writeRegister(1,
            CommunicationController.Register.REG1,0);
        System.out.println("timedout ? "+ t.timedOut());
        System.out.println(ret.toString());

        System.out.println();
        System.out.println("third test:");
        System.out.println();
        t = new Timer(MAX_LATENCY);
```

```
        t.start();
43      ret = communicationController.writeRegister(1,
            CommunicationController.Register.REG2,0xffff);
        System.out.println("timedout ? "+ t.timedOut());
45      System.out.println(ret.toString());

47      System.out.println();
        System.out.println("fourth test:");
49      System.out.println();
        t = new Timer(MAX_LATENCY);
51      t.start();
        ret = communicationController.readRegister(1,
            CommunicationController.Register.REG0,1);
53      System.out.println("timedout ? "+ t.timedOut());
        System.out.println(ret.toString());
55
        System.out.println();
57      System.out.println("fifth test:");
        System.out.println();
59      t = new Timer(MAX_LATENCY);
        t.start();
61      ret = communicationController.readRegister(1,
            CommunicationController.Register.REG1,1);
        System.out.println("timedout ? "+ t.timedOut());
63      System.out.println(ret.toString());

65      System.out.println();
        System.out.println("fifth test:");
67      System.out.println();
        t = new Timer(MAX_LATENCY);
69      t.start();
        ret = communicationController.readRegister(1,
            CommunicationController.Register.REG0,3);
```

```
71          System.out.println("timedout ? "+ t.timedOut());
            System.out.println(ret.toString());
73
            communicationController.end();
75
        }
77  }
```

commtest1.java

B.8 Programa utilizado para o teste de uso repetido da comunicação

```
1   package tcc_gui;
    /
3      @author juliano
      /
5   public class Tcc_GUI {

7       private static int MAX_LATENCY = 100;

9       /
           @param args the command line arguments
11        /
        public static void main(String[] args) throws Exception {
13
            CommunicationController communicationController = new
                CommunicationController();
15          communicationController.init("/dev/ttyUSB0");

17          System.out.println();
            System.out.println("write test:");
19          System.out.println();
```

```
        Timer t = new Timer(MAX_LATENCY);
        CommReturn ret = new CommReturn();

        for (int i = 1; i <= 50; i++) {

            System.out.println(i);
            t = new Timer(MAX_LATENCY);
            t.start();
            ret = communicationController.writeRegister(1,
                CommunicationController.Register.REG0, i);
            System.out.println("timedout ? " + t.timedOut());
            System.out.println(ret.toString());
        }

        System.out.println();
        System.out.println("read test:");
        System.out.println();
        t = new Timer(MAX_LATENCY);
        t.start();
        ret = communicationController.readRegister(1,
            CommunicationController.Register.REG0, 1);
        System.out.println("timedout ? " + t.timedOut());
        System.out.println(ret.toString());

        communicationController.end();

    }
}
```

commtest2.java

B.9 Programa utilizado para o teste de uso incorreto da comunicação

```
1  package tcc_gui;
2
3  /
4
5     @author juliano
6    /
7  public class Tcc_GUI {
8
9      private static int MAX_LATENCY = 100;
10
11     /
12        @param args the command line arguments
13       /
14     public static void main(String[] args) throws Exception {
15
16         CommunicationController communicationController = new
17             CommunicationController();
18         communicationController.init("/dev/ttyUSB0");
19         communicationController.setSIMPLE_TEST(true);
20
21         Timer t = new Timer(MAX_LATENCY);
22         CommReturn ret = new CommReturn();
23
24         System.out.println();
25         System.out.println("first test:");
26         System.out.println();
27         t = new Timer(MAX_LATENCY);
28         t.start();
29         ret = communicationController.writeRegister(2,
30             CommunicationController.Register.REG0, 1);
```

```
29          System.out.println("timedout ? " + t.timedOut());
            System.out.println(ret.toString());
31
            System.out.println();
33          System.out.println("second test:");
            System.out.println();
35          t = new Timer(MAX_LATENCY);
            t.start();
37          ret = communicationController.writeRegister(1,
                CommunicationController.Register.REG5, 1); // obs :
                registrador adicionado na classe communication controller
                 com address 5
            System.out.println("timedout ? " + t.timedOut());
39          System.out.println(ret.toString());

41          communicationController.setSABOTAGE_CRC(true);
            System.out.println();
43          System.out.println("third test:");
            System.out.println();
45          t = new Timer(MAX_LATENCY);
            t.start();
47          ret = communicationController.writeRegister(1,
                CommunicationController.Register.REG0, 1);
            System.out.println("timedout ? " + t.timedOut());
49          System.out.println(ret.toString());
            communicationController.setSABOTAGE_CRC(false);
51
            System.out.println();
53          System.out.println("fourth test:");
            System.out.println();
55          t = new Timer(MAX_LATENCY);
            t.start();
```

```
57          ret = communicationController.readRegister(1,
                CommunicationController.Register.REG5, 1);
            System.out.println("timedout ? " + t.timedOut());
59          System.out.println(ret.toString());

61          System.out.println();
            System.out.println("fifth test:");
63          System.out.println();
            t = new Timer(MAX_LATENCY);
65          t.start();
            ret = communicationController.readRegister(1,
                CommunicationController.Register.REG4, 2);
67          System.out.println("timedout ? " + t.timedOut());
            System.out.println(ret.toString());
69
            communicationController.setSABOTAGE_REGISTER_ADDRESS(true);
71          System.out.println();
            System.out.println("sixth test:");
73          System.out.println();
            t = new Timer(MAX_LATENCY);
75          t.start();
            ret = communicationController.readRegister(1,
                CommunicationController.Register.REG0, 1);
77          System.out.println("timedout ? " + t.timedOut());
            System.out.println(ret.toString());
79          communicationController.setSABOTAGE_REGISTER_ADDRESS(false);

81          communicationController.end();

83      }
}
```

commtest3.java

B.10 Programa final do microcontrolador

```
#include <PID_v1.h>

#include <MFTP_Motores.h>

#include <MFTP_Sensores.h>

#include <ModbusSlave.h>

ModbusSlave mbs;
MFTP_Sensores Sens;
MFTP_Motores Mots;

/ slave registers /
enum {
        EMERGENCY_STOP=0,
        SENSOR1,
        SENSOR2,
        SENSOR3,
        SENSOR4,
        SENSOR5,
        SENSOR6,
        SENSOR7,
        SENSOR8,
        SENSOR9,
        SENSOR10,
        SENSOR11,
        SENSOR12,
        SENSOR13,
        SENSOR14,
        SENSOR15,
        SENSOR16,
        SENSOR17,
```

```
    SENSOR18,
34  SENSOR19,
    SENSOR20,
36  TYPE_RECORD,
    TYPE_ID,
38  TYPE_PARA,
    TYPE_PARB,
40  TYPE_MULT,
    NOT_USED1,
42  SENSOR_RECORD,
    SENSOR_ID,
44  SENSOR_TYPE,
    SENSOR_ADDRESS,
46  NOT_USED2,
    MOTOR_RECORD,
48  MOTOR_ID,
    MOTOR_PWM1,
50  MOTOR_PWM2,
    MOTOR_CNA,
52  MOTOR_CNB,
    MOTOR_PPV,
54  MOTOR_COURSE,
    MOTOR_REDUCTION,
56  MOTOR_VMAX,
    MOTOR_KP,
58  MOTOR_TP,
    MOTOR_POSINI,
60  NOT_USED3,
    NOT_USED4,
62  SPEED_RECORD,
    SPEED_MOTOR_ID,
64  SPEED_VALUE,
    NOT_USED5,
```

```
66          TARGET_RECORD,
            TARGET_MOTOR_ID,
68          TARGET_VALUE,
            MOVE_RECORD,
70          MOVE_MOTOR_ID,
            MOVE_VALUE,
72          NOT_USED6,
            READPOS_MOTOR_ID,
74          READPOS_READY,
            READPOS_VALUE,
76          MB_REGS         /  required by ModbusSlave  /
};
78
int regs[MB_REGS]; // here are all the registers
80 int result = 0; // return of the update function

82 #define ledPin 13 // led

84 void setup()
{
86
  /  the Modbus slave configuration parameters  /
88  const unsigned char SLAVE = 1; // my address

90  const long BAUD = 9600;
  const char PARITY = 'e'; // even
92  const char TXENPIN = 0; // not RS485

94  mbs.configure(SLAVE,BAUD,PARITY,TXENPIN);

96  pinMode(ledPin , OUTPUT);
  digitalWrite(ledPin ,LOW);
98
```

```
  for(int i = 0;i<MB_REGS;i++) {
    regs[i] = 0; // initialize registers as 1
  }

}

void loop()
{
  result = mbs.update(regs, MB_REGS);
  Sens.get_all_values(regs); // be careful
  Mots.refresh();

  if(regs[TYPE_RECORD] != 0) { // ready to record a new type of
     sensor
    Sens.add_type(regs[TYPE_ID],regs[TYPE_PARA], regs[TYPE_PARB],
       regs[TYPE_MULT]);

    regs[TYPE_ID]=0;
    regs[TYPE_PARA]=0;
    regs[TYPE_PARB]=0;
    regs[TYPE_MULT]=0;
    regs[TYPE_RECORD]=0;
  }
  if(regs[SENSOR_RECORD] != 0) { // ready to record a new sensor
    Sens.add_sensor(regs[SENSOR_ID],regs[SENSOR_TYPE], regs[
       SENSOR_ADDRESS]);

    regs[SENSOR_ID]=0;
    regs[SENSOR_TYPE]=0;
    regs[SENSOR_ADDRESS]=0;
    regs[SENSOR_RECORD]=0;
  }
  if(regs[SENSOR_RECORD] != 0) { // ready to record a new sensor
```

```
    Sens.add_sensor(regs[SENSOR_ID],regs[SENSOR_TYPE], regs[
        SENSOR_ADDRESS]);
130
    regs[SENSOR_ID]=0;
132 regs[SENSOR_TYPE]=0;
    regs[SENSOR_ADDRESS]=0;
134 regs[SENSOR_RECORD]=0;
}
136 if(regs[MOTOR_RECORD] != 0) { // ready to record a new motor

138 Mots.add_motor(
        regs[MOTOR_ID],
140     regs[MOTOR_PWM1],
        regs[MOTOR_PWM2],
142     regs[MOTOR_CNA],
        regs[MOTOR_CNB],
144     regs[MOTOR_PPV],
        regs[MOTOR_COURSE],
146     regs[MOTOR_REDUCTION],
        regs[MOTOR_VMAX],
148     regs[MOTOR_KP],
        regs[MOTOR_TP],
150     regs[MOTOR_POSINI]);

152     regs[MOTOR_PWM1]=0;
        regs[MOTOR_PWM2]=0;
154     regs[MOTOR_CNA]=0;
        regs[MOTOR_CNB]=0;
156     regs[MOTOR_PPV]=0;
        regs[MOTOR_COURSE]=0;
158     regs[MOTOR_REDUCTION]=0;
        regs[MOTOR_VMAX]=0;
160     regs[MOTOR_KP]=0;
```

```
            regs[MOTOR_TP]=0;
            regs[MOTOR_POSINI]=0;
            regs[MOTOR_ID]=0;
    }

}
```

controler.ino

B.11 Programa final da interface gráfica

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * IHM.java
 *
 * Created on 04/11/2012, 02:24:25
 */
package tcc_gui;

import java.awt.event.*;
import java.util.logging.Level;
import java.util.logging.Logger;

/*
 *
 * @author juliano
 */
public class IHM extends javax.swing.JFrame {

```

```
    public CommunicationController commController;
24  public boolean connected = false;
    public boolean moving = false;
26  public Refresh refresh;

28  /
       Creates new form IHM
30    /
    public IHM() {
32      commController = new CommunicationController();
        //commController.setSIMPLE_TEST(true);
34      initComponents();
        this.configFrame.setVisible(false);
36      refresh = new Refresh();
        refresh.ihm = this;
38  }

40  /
       This method is called from within the constructor to
        initialize the form.
42     WARNING: Do NOT modify this code. The content of this method
        is always
       regenerated by the Form Editor.
44    /
    @SuppressWarnings("unchecked")
46  // <editor-fold defaultstate="collapsed" desc="Generated Code
       ">//GEN-BEGIN:initComponents
    private void initComponents() {
48
        speedId = new javax.swing.JTextField();
50      speedValue = new javax.swing.JTextField();
        sendSpeed = new javax.swing.JButton();
52      targetValue = new javax.swing.JTextField();
```

```
        targetId = new javax.swing.JTextField();
54      sendTarget = new javax.swing.JButton();
        emergencyStop = new javax.swing.JButton();
56      moveId = new javax.swing.JTextField();
        sendMove = new javax.swing.JButton();
58      readSensorId = new javax.swing.JTextField();
        readSensor = new javax.swing.JButton();
60      readSensorValue = new javax.swing.JTextField();
        commPort = new javax.swing.JTextField();
62      connect = new javax.swing.JToggleButton();
        jScrollPane1 = new javax.swing.JScrollPane();
64      allSensorsValues = new javax.swing.JTextArea();
        configFrame = new javax.swing.JInternalFrame();
66      sendSensor = new javax.swing.JButton();
        sensorAddress = new javax.swing.JTextField();
68      sensorType = new javax.swing.JTextField();
        sensorId = new javax.swing.JTextField();
70      configClose = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
72      jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
74      jSeparator1 = new javax.swing.JSeparator();
        typeId = new javax.swing.JTextField();
76      jLabel4 = new javax.swing.JLabel();
        typeParA = new javax.swing.JTextField();
78      jLabel5 = new javax.swing.JLabel();
        typeParB = new javax.swing.JTextField();
80      jLabel6 = new javax.swing.JLabel();
        typeMult = new javax.swing.JTextField();
82      jLabel7 = new javax.swing.JLabel();
        sendType = new javax.swing.JButton();
84      jSeparator2 = new javax.swing.JSeparator();
        motorId = new javax.swing.JTextField();
```

```
86          jLabel8 = new javax.swing.JLabel();
            motorPwm1 = new javax.swing.JTextField();
88          jLabel9 = new javax.swing.JLabel();
            motorPwm2 = new javax.swing.JTextField();
90          jLabel10 = new javax.swing.JLabel();
            motorCnA = new javax.swing.JTextField();
92          jLabel11 = new javax.swing.JLabel();
            motorCnB = new javax.swing.JTextField();
94          jLabel12 = new javax.swing.JLabel();
            motorPpv = new javax.swing.JTextField();
96          jLabel13 = new javax.swing.JLabel();
            motorCourse = new javax.swing.JTextField();
98          jLabel14 = new javax.swing.JLabel();
            motorReduction = new javax.swing.JTextField();
100         jLabel15 = new javax.swing.JLabel();
            motorVmax = new javax.swing.JTextField();
102         jLabel16 = new javax.swing.JLabel();
            motorKp = new javax.swing.JTextField();
104         jLabel17 = new javax.swing.JLabel();
            motorTp = new javax.swing.JTextField();
106         jLabel18 = new javax.swing.JLabel();
            motorPosIni = new javax.swing.JTextField();
108         jLabel19 = new javax.swing.JLabel();
            sendMotor = new javax.swing.JButton();
110         jSeparator3 = new javax.swing.JSeparator();
            config = new javax.swing.JButton();
112
            setDefaultCloseOperation(javax.swing.WindowConstants.
                DO_NOTHING_ON_CLOSE);
114         setTitle("Plataforma de Testes MicrofluÃ dicos");
            setPreferredSize(new java.awt.Dimension(550, 450));
116         setResizable(false);
            addWindowListener(new java.awt.event.WindowAdapter() {
```

```
118             public void windowClosing(java.awt.event.WindowEvent evt
                    ) {
                    kill(evt);
120             }
            });
122
            speedId.setText("1");
124
            speedValue.setText("40");
126
            sendSpeed.setText("send speed");
128         sendSpeed.setToolTipText("");
            sendSpeed.addMouseListener(new java.awt.event.MouseAdapter()
                {
130             public void mouseReleased(java.awt.event.MouseEvent evt)
                    {
                    sendSpeedMouseReleased(evt);
132             }
            });
134
            targetValue.setText("1000");
136
            targetId.setText("1");
138
            sendTarget.setText("send target");
140         sendTarget.addMouseListener(new java.awt.event.MouseAdapter
                () {
                public void mouseReleased(java.awt.event.MouseEvent evt)
                    {
                    sendTargetMouseReleased(evt);
142             }
            });
144
```

```
146         emergencyStop.setText("emergency");
            emergencyStop.addMouseListener(new java.awt.event.
                MouseAdapter() {
148              public void mouseReleased(java.awt.event.MouseEvent evt)
                     {
                     emergencyStopMouseReleased(evt);
150              }
            });
152
            moveId.setText("1");
154
            sendMove.setText("move");
156         sendMove.addMouseListener(new java.awt.event.MouseAdapter()
                {
                 public void mouseReleased(java.awt.event.MouseEvent evt)
                     {
158                  sendMoveMouseReleased(evt);
                 }
160         });

162         readSensorId.setText("1");

164         readSensor.setText("read pos");
            readSensor.addMouseListener(new java.awt.event.MouseAdapter
                () {
166              public void mouseReleased(java.awt.event.MouseEvent evt)
                     {
                     readSensorMouseReleased(evt);
168              }
            });
170
            readSensorValue.setEditable(false);
172         readSensorValue.setText("value");
```

```
174             commPort.setText("COM3");
176             connect.setText("connect");
                connect.addMouseListener(new java.awt.event.MouseAdapter() {
178                 public void mouseReleased(java.awt.event.MouseEvent evt)
                        {
                        connectMouseReleased(evt);
180                 }
                });
182
                allSensorsValues.setEditable(false);
184             allSensorsValues.setColumns(4);
                allSensorsValues.setRows(2);
186             allSensorsValues.setWrapStyleWord(true);
                jScrollPane1.setViewportView(allSensorsValues);
188
                configFrame.setTitle("config");
190             configFrame.setVisible(true);

192             sendSensor.setText("send sensor");
                sendSensor.addMouseListener(new java.awt.event.MouseAdapter
                    () {
194                 public void mouseReleased(java.awt.event.MouseEvent evt)
                        {
                        sendSensorMouseReleased(evt);
196                 }
                });
198
                sensorAddress.setText("14");
200
                sensorType.setText("1");
202
```

```
        sensorId.setText("1");
204
        configClose.setText("close");
206     configClose.addMouseListener(new java.awt.event.MouseAdapter
            () {
             public void mouseReleased(java.awt.event.MouseEvent evt)
                 {
208              configCloseMouseReleased(evt);
             }
210     });

212     jLabel1.setText("ID");

214     jLabel2.setText("Type");

216     jLabel3.setText("pin");

218     typeId.setText("1");

220     jLabel4.setText("ID");

222     typeParA.setText("1");

224     jLabel5.setText("parA");

226     typeParB.setText("1");

228     jLabel6.setText("parB");

230     typeMult.setText("1");

232     jLabel7.setText("mult");
```

```
234         sendType.setText("send type");
            sendType.addMouseListener(new java.awt.event.MouseAdapter()
                {
236              public void mouseReleased(java.awt.event.MouseEvent evt)
                     {
                     sendTypeMouseReleased(evt);
238              }
            });
240
            motorId.setText("1");
242
            jLabel8.setText("ID");
244
            motorPwm1.setText("3");
246
            jLabel9.setText("pwm1");
248
            motorPwm2.setText("5");
250
            jLabel10.setText("pwm2");
252
            motorCnA.setText("2");
254
            jLabel11.setText("cnA");
256
            motorCnB.setText("4");
258
            jLabel12.setText("cnB");
260
            motorPpv.setText("1000");
262
            jLabel13.setText("ppv");
264
```

```
        motorCourse.setText("10000");
266
        jLabel14.setText("curso");
268
        motorReduction.setText("810");
270
        jLabel15.setText("invK");
272
        motorVmax.setText("45");
274
        jLabel16.setText("Vmax");
276
        motorKp.setText("1");
278
        jLabel17.setText("Kp");
280
        motorTp.setText("0.014");
282
        jLabel18.setText("Tp");
284
        motorPosIni.setText("0");
286
        jLabel19.setText("posIni");
288
        sendMotor.setText("send motor");
290     sendMotor.addMouseListener(new java.awt.event.MouseAdapter()
            {
            public void mouseReleased(java.awt.event.MouseEvent evt)
                {
292             sendMotorMouseReleased(evt);
            }
294     });
```

```
296         org.jdesktop.layout.GroupLayout configFrameLayout = new org.
                jdesktop.layout.GroupLayout(configFrame.getContentPane())
                ;
            configFrame.getContentPane().setLayout(configFrameLayout);
298         configFrameLayout.setHorizontalGroup(
                configFrameLayout.createParallelGroup(org.jdesktop.
                    layout.GroupLayout.LEADING)
300             .add(configFrameLayout.createSequentialGroup()
                    .addContainerGap()
302                 .add(configFrameLayout.createParallelGroup(org.
                        jdesktop.layout.GroupLayout.LEADING)
                        .add(jSeparator3)
304                     .add(jSeparator1)
                        .add(org.jdesktop.layout.GroupLayout.TRAILING,
                            configFrameLayout.createSequentialGroup()
306                             .add(0, 0, Short.MAX_VALUE)
                                .add(configClose))
308                     .add(jSeparator2)
                        .add(configFrameLayout.createSequentialGroup()
310                         .add(configFrameLayout.createParallelGroup(
                                org.jdesktop.layout.GroupLayout.LEADING)
                                .add(motorReduction, org.jdesktop.layout
                                    .GroupLayout.PREFERRED_SIZE, org.
                                    jdesktop.layout.GroupLayout.
                                    DEFAULT_SIZE, org.jdesktop.layout.
                                    GroupLayout.PREFERRED_SIZE)
312                             .add(jLabel15))
                            .add(18, 18, 18)
314                         .add(configFrameLayout.createParallelGroup(
                                org.jdesktop.layout.GroupLayout.LEADING)
                                .add(configFrameLayout.
                                    createSequentialGroup()
```

```
316                     .add(motorVmax, org.jdesktop.layout.
                            GroupLayout.PREFERRED_SIZE, org.
                            jdesktop.layout.GroupLayout.
                            DEFAULT_SIZE, org.jdesktop.layout
                            .GroupLayout.PREFERRED_SIZE)
                    .add(18, 18, 18)
318                 .add(motorKp, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE, 21,
                        org.jdesktop.layout.GroupLayout.
                        PREFERRED_SIZE))
                .add(configFrameLayout.
                    createSequentialGroup()
320                 .add(jLabel16)
                    .addPreferredGap(org.jdesktop.layout
                        .LayoutStyle.UNRELATED)
322                 .add(jLabel17)))
            .add(18, 18, 18)
324         .add(configFrameLayout.createParallelGroup(
                org.jdesktop.layout.GroupLayout.LEADING)
                .add(motorTp, org.jdesktop.layout.
                    GroupLayout.PREFERRED_SIZE, org.
                    jdesktop.layout.GroupLayout.
                    DEFAULT_SIZE, org.jdesktop.layout.
                    GroupLayout.PREFERRED_SIZE)
326             .add(jLabel18))
            .add(18, 18, 18)
328         .add(configFrameLayout.createParallelGroup(
                org.jdesktop.layout.GroupLayout.LEADING)
                .add(configFrameLayout.
                    createSequentialGroup()
330                 .add(jLabel19)
                    .add(0, 0, Short.MAX_VALUE))
```

```
                .add(configFrameLayout.
                    createSequentialGroup()
                    .add(0, 0, Short.MAX_VALUE)
                    .add(motorPosIni, org.jdesktop.
                        layout.GroupLayout.PREFERRED_SIZE
                        , 20, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(org.jdesktop.layout
                        .LayoutStyle.UNRELATED)
                    .add(sendMotor))))
            .add(configFrameLayout.createSequentialGroup()
                .add(configFrameLayout.createParallelGroup(
                    org.jdesktop.layout.GroupLayout.LEADING)
                    .add(configFrameLayout.
                        createSequentialGroup()
                        .add(configFrameLayout.
                            createParallelGroup(org.jdesktop.
                            layout.GroupLayout.LEADING)
                            .add(sensorId, org.jdesktop.
                                layout.GroupLayout.
                                PREFERRED_SIZE, 26, org.
                                jdesktop.layout.GroupLayout.
                                PREFERRED_SIZE)
                            .add(jLabel1))
                        .addPreferredGap(org.jdesktop.layout
                            .LayoutStyle.RELATED)
                        .add(configFrameLayout.
                            createParallelGroup(org.jdesktop.
                            layout.GroupLayout.LEADING)
                            .add(configFrameLayout.
                                createSequentialGroup()
                                .add(jLabel2)
```

```
                .addPreferredGap(org.
                    jdesktop.layout.
                    LayoutStyle.RELATED)
348             .add(jLabel3))
            .add(configFrameLayout.
                createSequentialGroup()
350             .add(sensorType, org.
                    jdesktop.layout.
                    GroupLayout.
                    PREFERRED_SIZE, 22, org.
                    jdesktop.layout.
                    GroupLayout.
                    PREFERRED_SIZE)
                .addPreferredGap(org.
                    jdesktop.layout.
                    LayoutStyle.RELATED)
352             .add(sensorAddress, org.
                    jdesktop.layout.
                    GroupLayout.
                    PREFERRED_SIZE, 26, org.
                    jdesktop.layout.
                    GroupLayout.
                    PREFERRED_SIZE)
                .add(18, 18, 18)
354             .add(sendSensor))))
        .add(configFrameLayout.
            createSequentialGroup()
356         .add(configFrameLayout.
                createParallelGroup(org.jdesktop.
                layout.GroupLayout.LEADING)
                .add(typeId, org.jdesktop.layout
                    .GroupLayout.PREFERRED_SIZE,
                    20, org.jdesktop.layout.
```

```
                GroupLayout.PREFERRED_SIZE)
            .add(jLabel4))
        .addPreferredGap(org.jdesktop.layout
            .LayoutStyle.UNRELATED)
        .add(configFrameLayout.
            createParallelGroup(org.jdesktop.
            layout.GroupLayout.LEADING)
            .add(configFrameLayout.
                createSequentialGroup()
                .add(jLabel5)
                .add(18, 18, 18)
                .add(jLabel6)
                .addPreferredGap(org.
                    jdesktop.layout.
                    LayoutStyle.UNRELATED)
                .add(jLabel7))
            .add(configFrameLayout.
                createSequentialGroup()
                .add(typeParA, org.jdesktop.
                    layout.GroupLayout.
                    PREFERRED_SIZE, 20, org.
                    jdesktop.layout.
                    GroupLayout.
                    PREFERRED_SIZE)
                .add(18, 18, 18)
                .add(typeParB, org.jdesktop.
                    layout.GroupLayout.
                    PREFERRED_SIZE, 20, org.
                    jdesktop.layout.
                    GroupLayout.
                    PREFERRED_SIZE)
                .add(18, 18, 18)
```

```
372                                 .add(typeMult, org.jdesktop.
                                        layout.GroupLayout.
                                        PREFERRED_SIZE, 20, org.
                                        jdesktop.layout.
                                        GroupLayout.
                                        PREFERRED_SIZE)
                                    .add(18, 18, 18)
374                                 .add(sendType))))
                        .add(configFrameLayout.
                            createSequentialGroup()
376                         .add(motorId, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, 20,
                                org.jdesktop.layout.GroupLayout.
                                PREFERRED_SIZE)
                            .addPreferredGap(org.jdesktop.layout
                                .LayoutStyle.RELATED)
378                         .add(motorPwm1, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, 20,
                                org.jdesktop.layout.GroupLayout.
                                PREFERRED_SIZE)
                            .add(18, 18, 18)
380                         .add(motorPwm2, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, 20,
                                org.jdesktop.layout.GroupLayout.
                                PREFERRED_SIZE)
                            .add(18, 18, 18)
382                         .add(motorCnA, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, 20,
                                org.jdesktop.layout.GroupLayout.
                                PREFERRED_SIZE)
                            .addPreferredGap(org.jdesktop.layout
                                .LayoutStyle.UNRELATED)
```

```
                .add(motorCnB, org.jdesktop.layout.
                    GroupLayout.PREFERRED_SIZE, 20,
                    org.jdesktop.layout.GroupLayout.
                    PREFERRED_SIZE)
                .add(18, 18, 18)
                .add(motorPpv, org.jdesktop.layout.
                    GroupLayout.PREFERRED_SIZE, org.
                    jdesktop.layout.GroupLayout.
                    DEFAULT_SIZE, org.jdesktop.layout
                    .GroupLayout.PREFERRED_SIZE)
                .add(18, 18, 18)
                .add(motorCourse, org.jdesktop.
                    layout.GroupLayout.PREFERRED_SIZE
                    , org.jdesktop.layout.GroupLayout
                    .DEFAULT_SIZE, org.jdesktop.
                    layout.GroupLayout.PREFERRED_SIZE
                    ))
            .add(configFrameLayout.
                createSequentialGroup()
                .add(jLabel8)
                .add(18, 18, 18)
                .add(jLabel9)
                .addPreferredGap(org.jdesktop.layout
                    .LayoutStyle.RELATED)
                .add(jLabel10)
                .addPreferredGap(org.jdesktop.layout
                    .LayoutStyle.UNRELATED)
                .add(jLabel11)
                .add(18, 18, 18)
                .add(jLabel12)
                .add(18, 18, 18)
                .add(jLabel13)
                .add(28, 28, 28)
```

```
                                .add(jLabel14)))
                        .add(0, 0, Short.MAX_VALUE)))
                .addContainerGap())
        );
        configFrameLayout.setVerticalGroup(
            configFrameLayout.createParallelGroup(org.jdesktop.
                layout.GroupLayout.LEADING)
            .add(configFrameLayout.createSequentialGroup()
                .add(configFrameLayout.createParallelGroup(org.
                    jdesktop.layout.GroupLayout.LEADING)
                    .add(configFrameLayout.createSequentialGroup()
                        .addContainerGap()
                        .add(configFrameLayout.createParallelGroup(
                            org.jdesktop.layout.GroupLayout.BASELINE)
                            .add(jLabel1)
                            .add(jLabel2)
                            .add(jLabel3)))
                    .add(configFrameLayout.createSequentialGroup()
                        .add(30, 30, 30)
                        .add(configFrameLayout.createParallelGroup(
                            org.jdesktop.layout.GroupLayout.BASELINE)
                            .add(sensorId, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, org.
                                jdesktop.layout.GroupLayout.
                                DEFAULT_SIZE, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE)
                            .add(sensorType, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, org.
                                jdesktop.layout.GroupLayout.
                                DEFAULT_SIZE, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE)
                            .add(sensorAddress, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, org.
```

```
                        jdesktop.layout.GroupLayout.
                        DEFAULT_SIZE, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE)
422                 .add(sendSensor))))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.
                RELATED)
424         .add(jSeparator1, org.jdesktop.layout.GroupLayout.
                PREFERRED_SIZE, 10, org.jdesktop.layout.
                GroupLayout.PREFERRED_SIZE)
            .add(1, 1, 1)
426         .add(configFrameLayout.createParallelGroup(org.
                jdesktop.layout.GroupLayout.BASELINE)
                .add(jLabel4)
428             .add(jLabel5)
                .add(jLabel6)
430             .add(jLabel7))
            .add(2, 2, 2)
432         .add(configFrameLayout.createParallelGroup(org.
                jdesktop.layout.GroupLayout.BASELINE)
                .add(typeId, org.jdesktop.layout.GroupLayout.
                    PREFERRED_SIZE, org.jdesktop.layout.
                    GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                    .GroupLayout.PREFERRED_SIZE)
                .add(typeParA, org.jdesktop.layout.GroupLayout.
434                 PREFERRED_SIZE, org.jdesktop.layout.
                    GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                    .GroupLayout.PREFERRED_SIZE)
                .add(typeParB, org.jdesktop.layout.GroupLayout.
                    PREFERRED_SIZE, org.jdesktop.layout.
                    GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                    .GroupLayout.PREFERRED_SIZE)
                .add(typeMult, org.jdesktop.layout.GroupLayout.
436                 PREFERRED_SIZE, org.jdesktop.layout.
```

```
                GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                .GroupLayout.PREFERRED_SIZE)
            .add(sendType))
438         .addPreferredGap(org.jdesktop.layout.LayoutStyle.
            UNRELATED)
        .add(jSeparator2, org.jdesktop.layout.GroupLayout.
            PREFERRED_SIZE, 10, org.jdesktop.layout.
            GroupLayout.PREFERRED_SIZE)
440     .add(1, 1, 1)
        .add(configFrameLayout.createParallelGroup(org.
            jdesktop.layout.GroupLayout.BASELINE)
442         .add(jLabel8)
            .add(jLabel9)
444         .add(jLabel10)
            .add(jLabel11)
446         .add(jLabel12)
            .add(jLabel13)
448         .add(jLabel14))
        .add(3, 3, 3)
450     .add(configFrameLayout.createParallelGroup(org.
            jdesktop.layout.GroupLayout.BASELINE)
            .add(motorId, org.jdesktop.layout.GroupLayout.
                PREFERRED_SIZE, org.jdesktop.layout.
                GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                .GroupLayout.PREFERRED_SIZE)
452         .add(motorPwm1, org.jdesktop.layout.GroupLayout.
                PREFERRED_SIZE, org.jdesktop.layout.
                GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                .GroupLayout.PREFERRED_SIZE)
            .add(motorPwm2, org.jdesktop.layout.GroupLayout.
                PREFERRED_SIZE, org.jdesktop.layout.
                GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                .GroupLayout.PREFERRED_SIZE)
```

```
454                 .add(motorCnA, org.jdesktop.layout.GroupLayout.
                        PREFERRED_SIZE, org.jdesktop.layout.
                        GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                        .GroupLayout.PREFERRED_SIZE)
                    .add(motorCnB, org.jdesktop.layout.GroupLayout.
                        PREFERRED_SIZE, org.jdesktop.layout.
                        GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                        .GroupLayout.PREFERRED_SIZE)
456                 .add(motorPpv, org.jdesktop.layout.GroupLayout.
                        PREFERRED_SIZE, org.jdesktop.layout.
                        GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                        .GroupLayout.PREFERRED_SIZE)
                    .add(motorCourse, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE, org.jdesktop.
                        layout.GroupLayout.DEFAULT_SIZE, org.jdesktop
                        .layout.GroupLayout.PREFERRED_SIZE))
458             .add(5, 5, 5)
                .add(configFrameLayout.createParallelGroup(org.
                    jdesktop.layout.GroupLayout.BASELINE)
460                 .add(jLabel15)
                    .add(jLabel16)
462                 .add(jLabel17)
                    .add(jLabel18)
464                 .add(jLabel19, org.jdesktop.layout.GroupLayout.
                        PREFERRED_SIZE, 14, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.
                    RELATED)
466             .add(configFrameLayout.createParallelGroup(org.
                    jdesktop.layout.GroupLayout.BASELINE)
                    .add(motorReduction, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE, org.jdesktop.
                        layout.GroupLayout.DEFAULT_SIZE, org.jdesktop
```

```
                    .layout.GroupLayout.PREFERRED_SIZE)
                .add(motorVmax, org.jdesktop.layout.GroupLayout.
                    PREFERRED_SIZE, org.jdesktop.layout.
                    GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                    .GroupLayout.PREFERRED_SIZE)
                .add(motorKp, org.jdesktop.layout.GroupLayout.
                    PREFERRED_SIZE, org.jdesktop.layout.
                    GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                    .GroupLayout.PREFERRED_SIZE)
                .add(motorTp, org.jdesktop.layout.GroupLayout.
                    PREFERRED_SIZE, org.jdesktop.layout.
                    GroupLayout.DEFAULT_SIZE, org.jdesktop.layout
                    .GroupLayout.PREFERRED_SIZE)
                .add(motorPosIni, org.jdesktop.layout.
                    GroupLayout.PREFERRED_SIZE, org.jdesktop.
                    layout.GroupLayout.DEFAULT_SIZE, org.jdesktop
                    .layout.GroupLayout.PREFERRED_SIZE)
                .add(sendMotor))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.
                UNRELATED)
            .add(jSeparator3, org.jdesktop.layout.GroupLayout.
                PREFERRED_SIZE, 10, org.jdesktop.layout.
                GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.
                RELATED, 22, Short.MAX_VALUE)
            .add(configClose)
            .addContainerGap())
    );

    config.setText("config");
    config.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseReleased(java.awt.event.MouseEvent evt)
            {
```

```
                configMouseReleased(evt);
484         }
        });
486
        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.
            layout.GroupLayout(getContentPane());
488     getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
490         layout.createParallelGroup(org.jdesktop.layout.
                GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
492             .addContainerGap()
                .add(layout.createParallelGroup(org.jdesktop.layout.
                    GroupLayout.LEADING)
494                 .add(layout.createSequentialGroup()
                        .add(21, 21, 21)
496                     .add(commPort, org.jdesktop.layout.
                            GroupLayout.PREFERRED_SIZE, org.jdesktop.
                            layout.GroupLayout.DEFAULT_SIZE, org.
                            jdesktop.layout.GroupLayout.
                            PREFERRED_SIZE)
                        .add(18, 18, 18)
498                     .add(connect))
                    .add(layout.createSequentialGroup()
500                     .add(targetId, org.jdesktop.layout.
                            GroupLayout.PREFERRED_SIZE, org.jdesktop.
                            layout.GroupLayout.DEFAULT_SIZE, org.
                            jdesktop.layout.GroupLayout.
                            PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.
                            LayoutStyle.RELATED)
502                     .add(targetValue, org.jdesktop.layout.
                            GroupLayout.PREFERRED_SIZE, org.jdesktop.
```

```
                layout.GroupLayout.DEFAULT_SIZE, org.
                jdesktop.layout.GroupLayout.
                PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.
                LayoutStyle.RELATED)
504         .add(sendTarget))
        .add(layout.createSequentialGroup()
506         .add(moveId, org.jdesktop.layout.GroupLayout
                .PREFERRED_SIZE, org.jdesktop.layout.
                GroupLayout.DEFAULT_SIZE, org.jdesktop.
                layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.
                LayoutStyle.RELATED)
508         .add(sendMove))
        .add(layout.createSequentialGroup()
510         .add(layout.createParallelGroup(org.jdesktop
                .layout.GroupLayout.LEADING)
                .add(layout.createSequentialGroup()
512                 .add(speedId, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE, org.
                        jdesktop.layout.GroupLayout.
                        DEFAULT_SIZE, org.jdesktop.layout
                        .GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(org.jdesktop.layout
                        .LayoutStyle.RELATED)
514                 .add(speedValue, org.jdesktop.layout
                        .GroupLayout.PREFERRED_SIZE, org.
                        jdesktop.layout.GroupLayout.
                        DEFAULT_SIZE, org.jdesktop.layout
                        .GroupLayout.PREFERRED_SIZE))
                .add(readSensorId, org.jdesktop.layout.
                    GroupLayout.PREFERRED_SIZE, org.
                    jdesktop.layout.GroupLayout.
```

```
                DEFAULT_SIZE, org.jdesktop.layout.
                GroupLayout.PREFERRED_SIZE))
516         .addPreferredGap(org.jdesktop.layout.
            LayoutStyle.RELATED)
            .add(layout.createParallelGroup(org.jdesktop
            .layout.GroupLayout.LEADING)
518             .add(layout.createSequentialGroup()
                    .add(config)
520                 .add(18, 18, 18)
                    .add(emergencyStop)
522                 .add(18, 18, 18)
                    .add(jScrollPane1, org.jdesktop.
                        layout.GroupLayout.PREFERRED_SIZE
                        , org.jdesktop.layout.GroupLayout
                        .DEFAULT_SIZE, org.jdesktop.
                        layout.GroupLayout.PREFERRED_SIZE
                        ))
524             .add(layout.createSequentialGroup()
                    .add(layout.createParallelGroup(org.
                        jdesktop.layout.GroupLayout.
                        LEADING)
526                     .add(sendSpeed)
                        .add(layout.
                            createSequentialGroup()
528                         .add(readSensor)
                            .add(26, 26, 26)
530                         .add(readSensorValue, org.
                                jdesktop.layout.
                                GroupLayout.
                                PREFERRED_SIZE, org.
                                jdesktop.layout.
                                GroupLayout.DEFAULT_SIZE,
                                 org.jdesktop.layout.
```

```
                                        GroupLayout.
                                        PREFERRED_SIZE)))
                                .addPreferredGap(org.jdesktop.layout
                                    .LayoutStyle.UNRELATED)
532                             .add(configFrame, org.jdesktop.
                                    layout.GroupLayout.PREFERRED_SIZE
                                    , org.jdesktop.layout.GroupLayout
                                    .DEFAULT_SIZE, org.jdesktop.
                                    layout.GroupLayout.PREFERRED_SIZE
                                    )))))
                .addContainerGap(33, Short.MAX_VALUE))
534         );
            layout.setVerticalGroup(
536             layout.createParallelGroup(org.jdesktop.layout.
                    GroupLayout.LEADING)
                .add(layout.createSequentialGroup()
538                 .addContainerGap()
                    .add(layout.createParallelGroup(org.jdesktop.layout.
                        GroupLayout.LEADING)
540                     .add(layout.createSequentialGroup()
                            .add(layout.createParallelGroup(org.jdesktop
                                .layout.GroupLayout.BASELINE)
542                             .add(config)
                                .add(emergencyStop))
544                         .add(67, 67, 67)
                            .add(layout.createParallelGroup(org.jdesktop
                                .layout.GroupLayout.BASELINE)
546                             .add(readSensor)
                                .add(readSensorId, org.jdesktop.layout.
                                    GroupLayout.PREFERRED_SIZE, org.
                                    jdesktop.layout.GroupLayout.
                                    DEFAULT_SIZE, org.jdesktop.layout.
                                    GroupLayout.PREFERRED_SIZE)
```

```
548                     .add(readSensorValue, org.jdesktop.
                            layout.GroupLayout.PREFERRED_SIZE,
                            org.jdesktop.layout.GroupLayout.
                            DEFAULT_SIZE, org.jdesktop.layout.
                            GroupLayout.PREFERRED_SIZE))
                .add(31, 31, 31)
550             .add(layout.createParallelGroup(org.jdesktop
                    .layout.GroupLayout.BASELINE)
                    .add(speedId, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE, org.
                        jdesktop.layout.GroupLayout.
                        DEFAULT_SIZE, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE)
552                 .add(speedValue, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE, org.
                        jdesktop.layout.GroupLayout.
                        DEFAULT_SIZE, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE)
                    .add(sendSpeed))
554             .add(26, 26, 26)
                .add(layout.createParallelGroup(org.jdesktop
                    .layout.GroupLayout.BASELINE)
556                 .add(targetValue, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE, org.
                        jdesktop.layout.GroupLayout.
                        DEFAULT_SIZE, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE)
                    .add(targetId, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE, org.
                        jdesktop.layout.GroupLayout.
                        DEFAULT_SIZE, org.jdesktop.layout.
                        GroupLayout.PREFERRED_SIZE)
558                 .add(sendTarget))
```

```
                        .add(18, 18, 18)
560                     .add(layout.createParallelGroup(org.jdesktop
                            .layout.GroupLayout.BASELINE)
                            .add(moveId, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, org.
                                jdesktop.layout.GroupLayout.
                                DEFAULT_SIZE, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE)
562                         .add(sendMove))
                        .add(64, 64, 64)
564                     .add(layout.createParallelGroup(org.jdesktop
                            .layout.GroupLayout.BASELINE)
                            .add(connect)
566                         .add(commPort, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE, org.
                                jdesktop.layout.GroupLayout.
                                DEFAULT_SIZE, org.jdesktop.layout.
                                GroupLayout.PREFERRED_SIZE)))
                    .add(layout.createSequentialGroup()
568                     .add(jScrollPane1, org.jdesktop.layout.
                            GroupLayout.PREFERRED_SIZE, 50, org.
                            jdesktop.layout.GroupLayout.
                            PREFERRED_SIZE)
                        .add(18, 18, 18)
570                     .add(configFrame, org.jdesktop.layout.
                            GroupLayout.PREFERRED_SIZE, org.jdesktop.
                            layout.GroupLayout.DEFAULT_SIZE, org.
                            jdesktop.layout.GroupLayout.
                            PREFERRED_SIZE)))
                .addContainerGap(49, Short.MAX_VALUE))
572     );

574     pack();
```

```
    }// </editor-fold>//GEN-END:initComponents

576

    private void sendSensorMouseReleased(java.awt.event.MouseEvent
        evt) {//GEN-FIRST:event_sendSensorMouseReleased
578     int id = Integer.parseInt(this.sensorId.getText());
        int type = Integer.parseInt(this.sensorType.getText());
580     int address = Integer.parseInt(this.sensorAddress.getText())
            ;
        try {
582         this.commController.writeRegister(1,
                CommunicationController.Register.SENSOR_ID, id);
            this.commController.writeRegister(1,
                CommunicationController.Register.SENSOR_TYPE, type);
584         this.commController.writeRegister(1,
                CommunicationController.Register.SENSOR_ADDRESS,
                address);
            this.commController.writeRegister(1,
                CommunicationController.Register.SENSOR_RECORD, 1);
586
        } catch (Exception ex) {
588         Logger.getLogger(IHM.class.getName()).log(Level.SEVERE,
                null, ex);
        }
590
    }//GEN-LAST:event_sendSensorMouseReleased
592
    private void kill(java.awt.event.WindowEvent evt) {//GEN-FIRST:
        event_kill
594     if (connected) {
            try {
596             commController.end();
                refresh.interrupt();
598         } catch (Exception ex) {
```

```
                Logger.getLogger(IHM.class.getName()).log(Level.
                    SEVERE, null, ex);
            }
        }
        System.exit(0);
    }//GEN-LAST:event_kill

    private void sendTypeMouseReleased(java.awt.event.MouseEvent evt
        ) {//GEN-FIRST:event_sendTypeMouseReleased
        int id = Integer.parseInt(this.typeId.getText());
        int parA = Integer.parseInt(this.typeParA.getText());
        int parB = Integer.parseInt(this.typeParB.getText());
        int mult = Integer.parseInt(this.typeMult.getText());
        try {
            this.commController.writeRegister(1,
                CommunicationController.Register.TYPE_ID, id);
            this.commController.writeRegister(1,
                CommunicationController.Register.TYPE_PARA, parA);
            this.commController.writeRegister(1,
                CommunicationController.Register.TYPE_PARB, parB);
            this.commController.writeRegister(1,
                CommunicationController.Register.TYPE_MULT, mult);
            this.commController.writeRegister(1,
                CommunicationController.Register.TYPE_RECORD, 1);
        } catch (Exception ex) {
            Logger.getLogger(IHM.class.getName()).log(Level.SEVERE,
                null, ex);
        }

    }//GEN-LAST:event_sendTypeMouseReleased

    private void readSensorMouseReleased(java.awt.event.MouseEvent
        evt) {//GEN-FIRST:event_readSensorMouseReleased
```

```
624         int id = Integer.parseInt(this.readSensorId.getText());
            try {
626             this.commController.writeRegister(1,
                    CommunicationController.Register.READPOS_MOTOR_ID, id
                    );
                this.commController.writeRegister(1,
                    CommunicationController.Register.READPOS_READY, 0);
628             CommReturn ret = new CommReturn();
                int i = 0;
630
                ret = this.commController.readRegister(1,
                    CommunicationController.Register.READPOS_READY, 1);
632             while (ret.value[0] == 0 && i<CommunicationController.
                    NUMBER_OF_RETRIES) {
                    this.wait(CommunicationController.FRAME_TIMEOUT);
634                 ret = this.commController.readRegister(1,
                        CommunicationController.Register.READPOS_READY,
                        1);
                    i++;
636             }

638             ret = this.commController.readRegister(1,
                    CommunicationController.Register.READPOS_VALUE, 1);
                this.readSensorValue.setText(Integer.toString(ret.value
                    [0]));
640         } catch (Exception ex) {
                Logger.getLogger(IHM.class.getName()).log(Level.SEVERE,
                    null, ex);
642         }
        }//GEN-LAST:event_readSensorMouseReleased
644
```

```
    private void connectMouseReleased(java.awt.event.MouseEvent evt)
        {//GEN-FIRST:event_connectMouseReleased
646     if (!connected) {
            try {
                commController.init(this.commPort.getText());
                connected = true;
                this.connect.setText("disconnect");
            } catch (Exception ex) {
                Logger.getLogger(IHM.class.getName()).log(Level.
                    SEVERE, null, ex);
            }
        } else {
            try {
                connected = false;
                commController.end();
                this.connect.setText("connect");
            } catch (Exception ex) {
                Logger.getLogger(IHM.class.getName()).log(Level.
                    SEVERE, null, ex);
            }
        }
    }//GEN-LAST:event_connectMouseReleased

    private void sendMotorMouseReleased(java.awt.event.MouseEvent
        evt) {//GEN-FIRST:event_sendMotorMouseReleased
        int id = Integer.parseInt(this.motorId.getText());
        int pwm1 = Integer.parseInt(this.motorPwm1.getText());
        int pwm2 = Integer.parseInt(this.motorPwm2.getText());
        int cnA = Integer.parseInt(this.motorCnA.getText());
        int cnB = Integer.parseInt(this.motorCnB.getText());
        int ppv = Integer.parseInt(this.motorPpv.getText());
        int course = Integer.parseInt(this.motorCourse.getText());
```

```
            int reduction = Integer.parseInt(this.motorReduction.getText
                ());
674         int vmax = Integer.parseInt(this.motorVmax.getText());
            int kp = Integer.parseInt(this.motorKp.getText());
676         int tp = (int) (1.0 / Double.parseDouble(this.motorTp.
                getText()));
            int posini = Integer.parseInt(this.motorPosIni.getText());
678         try {
                this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_ID, id);
680             this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_PWM1, pwm1);
                this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_PWM2, pwm2);
682             this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_CNA, cnA);
                this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_CNB, cnB);
684             this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_PPV, ppv);
                this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_COURSE, course
                    );
686             this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_REDUCTION,
                    reduction);
                this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_VMAX, vmax);
688             this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_KP, kp);
                this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_TP, tp);
```

```
690             this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_POSINI, posini
                    );
                this.commController.writeRegister(1,
                    CommunicationController.Register.MOTOR_RECORD, 1);
692         } catch (Exception ex) {
                Logger.getLogger(IHM.class.getName()).log(Level.SEVERE,
                    null, ex);
694         }
        }//GEN-LAST:event_sendMotorMouseReleased
696
        private void sendSpeedMouseReleased(java.awt.event.MouseEvent
            evt) {//GEN-FIRST:event_sendSpeedMouseReleased
698         int id = Integer.parseInt(this.speedId.getText());
            int speed = Integer.parseInt(this.speedValue.getText());
700         try {
                this.commController.writeRegister(1,
                    CommunicationController.Register.SPEED_MOTOR_ID, id);
702             this.commController.writeRegister(1,
                    CommunicationController.Register.SPEED_VALUE, speed);
                this.commController.writeRegister(1,
                    CommunicationController.Register.SPEED_RECORD, 1);
704         } catch (Exception ex) {
                Logger.getLogger(IHM.class.getName()).log(Level.SEVERE,
                    null, ex);
706         }
        }//GEN-LAST:event_sendSpeedMouseReleased
708
        private void sendTargetMouseReleased(java.awt.event.MouseEvent
            evt) {//GEN-FIRST:event_sendTargetMouseReleased
710         int id = Integer.parseInt(this.targetId.getText());
            int target = Integer.parseInt(this.targetValue.getText());
712         try {
```

```
            this.commController.writeRegister(1,
                CommunicationController.Register.TARGET_MOTOR_ID, id)
                ;
            this.commController.writeRegister(1,
                CommunicationController.Register.TARGET_VALUE, target
                );
            this.commController.writeRegister(1,
                CommunicationController.Register.TARGET_RECORD, 1);
        } catch (Exception ex) {
            Logger.getLogger(IHM.class.getName()).log(Level.SEVERE,
                null, ex);
        }
    }//GEN-LAST:event_sendTargetMouseReleased

    private void sendMoveMouseReleased(java.awt.event.MouseEvent evt
        ) {//GEN-FIRST:event_sendMoveMouseReleased
        int id = Integer.parseInt(this.moveId.getText());
        int move = 1;
        this.sendMove.setText("stop");
        if (moving) {
            move = 0;
            this.sendMove.setText("move");
        }
        moving = !moving;
        try {
            this.commController.writeRegister(1,
                CommunicationController.Register.MOVE_MOTOR_ID, id);
            this.commController.writeRegister(1,
                CommunicationController.Register.MOVE_VALUE, move);
            this.commController.writeRegister(1,
                CommunicationController.Register.MOVE_RECORD, 1);
        } catch (Exception ex) {
```

```
            Logger.getLogger(IHM.class.getName()).log(Level.SEVERE,
                null, ex);
        }
    }//GEN-LAST:event_sendMoveMouseReleased

    private void configMouseReleased(java.awt.event.MouseEvent evt)
        {//GEN-FIRST:event_configMouseReleased
        this.configFrame.setVisible(true);
    }//GEN-LAST:event_configMouseReleased

    private void configCloseMouseReleased(java.awt.event.MouseEvent
        evt) {//GEN-FIRST:event_configCloseMouseReleased
        this.configFrame.setVisible(false);
    }//GEN-LAST:event_configCloseMouseReleased

    private void emergencyStopMouseReleased(java.awt.event.
        MouseEvent evt) {//GEN-FIRST:event_emergencyStopMouseReleased
        try {
            this.commController.writeRegister(1,
                CommunicationController.Register.EMERGENCY_STOP, 1);
        } catch (Exception ex) {
            Logger.getLogger(IHM.class.getName()).log(Level.SEVERE,
                null, ex);
        }
    }//GEN-LAST:event_emergencyStopMouseReleased

    /
       @param args the command line arguments
      /
    public static void main(String args[]) {
        /  Set the Nimbus look and feel  /
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
            setting code (optional) ">
```

```
        /* If Nimbus (introduced in Java SE 6) is not available,
           stay with the default look and feel.
           For details see http://download.oracle.com/javase/
            tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.
                swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.
                        getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(IHM.class.getName()).
                log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(IHM.class.getName()).
                log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(IHM.class.getName()).
                log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(IHM.class.getName()).
                log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new IHM().setVisible(true);
```

```
786             }
            });
788     }
        // Variables declaration - do not modify//GEN-BEGIN:variables
790     public javax.swing.JTextArea allSensorsValues;
        private javax.swing.JTextField commPort;
792     private javax.swing.JButton config;
        private javax.swing.JButton configClose;
794     private javax.swing.JInternalFrame configFrame;
        private javax.swing.JToggleButton connect;
796     private javax.swing.JButton emergencyStop;
        private javax.swing.JLabel jLabel1;
798     private javax.swing.JLabel jLabel10;
        private javax.swing.JLabel jLabel11;
800     private javax.swing.JLabel jLabel12;
        private javax.swing.JLabel jLabel13;
802     private javax.swing.JLabel jLabel14;
        private javax.swing.JLabel jLabel15;
804     private javax.swing.JLabel jLabel16;
        private javax.swing.JLabel jLabel17;
806     private javax.swing.JLabel jLabel18;
        private javax.swing.JLabel jLabel19;
808     private javax.swing.JLabel jLabel2;
        private javax.swing.JLabel jLabel3;
810     private javax.swing.JLabel jLabel4;
        private javax.swing.JLabel jLabel5;
812     private javax.swing.JLabel jLabel6;
        private javax.swing.JLabel jLabel7;
814     private javax.swing.JLabel jLabel8;
        private javax.swing.JLabel jLabel9;
816     private javax.swing.JScrollPane jScrollPane1;
        private javax.swing.JSeparator jSeparator1;
818     private javax.swing.JSeparator jSeparator2;
```

```
    private javax.swing.JSeparator jSeparator3;
820 private javax.swing.JTextField motorCnA;
    private javax.swing.JTextField motorCnB;
822 private javax.swing.JTextField motorCourse;
    private javax.swing.JTextField motorId;
824 private javax.swing.JTextField motorKp;
    private javax.swing.JTextField motorPosIni;
826 private javax.swing.JTextField motorPpv;
    private javax.swing.JTextField motorPwm1;
828 private javax.swing.JTextField motorPwm2;
    private javax.swing.JTextField motorReduction;
830 private javax.swing.JTextField motorTp;
    private javax.swing.JTextField motorVmax;
832 private javax.swing.JTextField moveId;
    private javax.swing.JButton readSensor;
834 private javax.swing.JTextField readSensorId;
    private javax.swing.JTextField readSensorValue;
836 private javax.swing.JButton sendMotor;
    private javax.swing.JButton sendMove;
838 private javax.swing.JButton sendSensor;
    private javax.swing.JButton sendSpeed;
840 private javax.swing.JButton sendTarget;
    private javax.swing.JButton sendType;
842 private javax.swing.JTextField sensorAddress;
    private javax.swing.JTextField sensorId;
844 private javax.swing.JTextField sensorType;
    private javax.swing.JTextField speedId;
846 private javax.swing.JTextField speedValue;
    private javax.swing.JTextField targetId;
848 private javax.swing.JTextField targetValue;
    private javax.swing.JTextField typeId;
850 private javax.swing.JTextField typeMult;
    private javax.swing.JTextField typeParA;
```

```
852     private javax.swing.JTextField typeParB;
        // End of variables declaration//GEN-END:variables
854 }
```

IHM.java

```
/
2    To change this template, choose Tools | Templates
     and open the template in the editor.
4   /
package tcc_gui;
6
import java.util.logging.Level;
8 import java.util.logging.Logger;

10 /

12    @author usuario
    /
14 public class Refresh extends Thread {

16     public IHM ihm;
       final int MAX_SENSORS = 2;
18
       public void run() {
20         while (true) {
               try {
22                 this.sleep(500);
               } catch (InterruptedException ex) {
24                 Logger.getLogger(Refresh.class.getName()).log(Level.
                       SEVERE, null, ex);
               }
26             if (ihm.connected) {
                   CommReturn ret;
```

```
                try {
                    ret = ihm.commController.readRegister(1,
                        CommunicationController.Register.SENSOR1,
                        MAX_SENSORS);
                    String values = "";
                    for (int i = 0; i < MAX_SENSORS; i++) {
                        values += Integer.toString(ret.value[i]) + "
                            \n";
                    }
                    ihm.allSensorsValues.setText(values);
                } catch (Exception ex) {
                    Logger.getLogger(Refresh.class.getName()).log(
                        Level.SEVERE, null, ex);
                }
            }
        }
    }
}
```

Refresh.java

```
package tcc_gui;

/

   @author juliano
  /
public class Tcc_GUI {

    /
       @param args the command line arguments
      /
    public static void main(String[] args) throws Exception {

```

```
        final IHM ihm = new IHM();

//          ihm.commController = new CommunicationController();
//          ihm.commController.init("COM3");
//          ihm.commController.setDEBUG(true);


        //CommReturn ret = new CommReturn();

        //ret = communicationController.writeRegister(2,
            CommunicationController.Register.EMERGENCY_STOP, 1);
        //ret = communicationController.readRegister(1,
            CommunicationController.Register.EMERGENCY_STOP, 1);

          /  Set the Nimbus look and feel  /
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
            setting code (optional) ">
        /  If Nimbus (introduced in Java SE 6) is not available,
            stay with the default look and feel.
           For details see http://download.oracle.com/javase/
            tutorial/uiswing/lookandfeel/plaf.html
          /
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.
                swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.
                        getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
```

```
39              java.util.logging.Logger.getLogger(IHM.class.getName()).
                    log(java.util.logging.Level.SEVERE, null, ex);
            } catch (InstantiationException ex) {
41              java.util.logging.Logger.getLogger(IHM.class.getName()).
                    log(java.util.logging.Level.SEVERE, null, ex);
            } catch (IllegalAccessException ex) {
43              java.util.logging.Logger.getLogger(IHM.class.getName()).
                    log(java.util.logging.Level.SEVERE, null, ex);
            } catch (javax.swing.UnsupportedLookAndFeelException ex) {
45              java.util.logging.Logger.getLogger(IHM.class.getName()).
                    log(java.util.logging.Level.SEVERE, null, ex);
            }
47          //</editor-fold>

49          /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
51
                public void run() {
53                  ihm.setVisible(true);
                    ihm.refresh.start();
55              }
            });
57
    }
59  }
```

Tcc_GUI.java

APÊNDICE C – RESULTADOS DE TESTES

C.1 Teste dinâmico do motor CC utilizado

Teste	min Δticks	voltas	K E	$\overline{\tau}$	$\sigma\tau$	correlação com $KE(1-e^{-t/\overline{\tau}})$
1	50	5	37,63	15,2	3,6	0,9727
2	50	5	37,74	14,0	2,6	0,9771
3	40	6	36,85	14,5	1,9	0,9795
4	100	6	38,17	15,7	6,4	0,9559
5	200	6	37,00	10,9	3,1	0,9761

Tabela 4: Resultado do teste dinâmico do motor CC utilizado.

C.2 Teste Simples da Implementação de Modbus entre o Computador e a Placa Arduino™

.

```
run:
trying to send: 1 3 0 0 0 1 84 a
waiting response !

trying to send: 1 3 0 0 0 1 84 a
waiting response !

trying to send: 1 3 0 0 0 1 84 a
waiting response !
```

```
 1 3 2 0 1 79 84

first test:

trying to send: 1 6 0 0 0 1 48 a
waiting response !
 1 6 0 0 0 1 48 a
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=1  Size=6 Exception=0

second test:

trying to send: 1 6 0 1 0 0 d8 a
waiting response !
 1 6 0 1 0 0 d8 a
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=0  Size=6 Exception=0

third test:

trying to send: 1 6 0 2 ff ff 29 ba
waiting response !
 1 6 0 2 ff ff 29 ba
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=65535  Size=6 Excepti

fourth test:
```

```
trying to send: 1 3 0 0 0 1 84 a
waiting response !
 1 3 2 0 1 79 84
timedout ? false
commSuccess=true executionSuccess=true Function=3 Value=1  Size=5 Exception=0

fifth test:

trying to send: 1 3 0 1 0 1 d5 ca
waiting response !
 1 3 2 0 0 b8 44
timedout ? false
commSuccess=true executionSuccess=true Function=3 Value=0  Size=5 Exception=0

fifth test:

trying to send: 1 3 0 0 0 3 5 cb
waiting response !
 1 3 6 0 1 0 0 ff ff 1d 5
timedout ? false
commSuccess=true executionSuccess=true Function=3 Value=1 0 65535  Size=9 Exc
BUILD SUCCESSFUL (total time: 1 second)
```

C.3 Teste do Uso Repetido da Implementação de Modbus entre o Computador e a Placa Arduino™

.

```
run:
trying to send: 1 3 0 0 0 1 84 a
waiting response !

trying to send: 1 3 0 0 0 1 84 a
waiting response !

trying to send: 1 3 0 0 0 1 84 a
waiting response !
 1 3 2 0 1 79 84

write test:

1
trying to send: 1 6 0 0 0 1 48 a
waiting response !
 1 6 0 0 0 1 48 a
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=1  Size=6 Exception=0
2
trying to send: 1 6 0 0 0 2 8 b
waiting response !
 1 6 0 0 0 2 8 b
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=2  Size=6 Exception=0
3
trying to send: 1 6 0 0 0 3 c9 cb
waiting response !
```

```
 1 6 0 0 0 3 c9 cb
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=3  Size=6 Exception=0
4
trying to send: 1 6 0 0 0 4 88 9
waiting response !
 1 6 0 0 0 4 88 9
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=4  Size=6 Exception=0
5
trying to send: 1 6 0 0 0 5 49 c9
waiting response !
 1 6 0 0 0 5 49 c9
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=5  Size=6 Exception=0
6
trying to send: 1 6 0 0 0 6 9 c8
waiting response !
 1 6 0 0 0 6 9 c8
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=6  Size=6 Exception=0
7
trying to send: 1 6 0 0 0 7 c8 8
waiting response !
 1 6 0 0 0 7 c8 8
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=7  Size=6 Exception=0
8
```

```
trying to send: 1 6 0 0 0 8 88 c
waiting response !
 1 6 0 0 0 8 88 c
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=8  Size=6 Exception=0
9
trying to send: 1 6 0 0 0 9 49 cc
waiting response !
 1 6 0 0 0 9 49 cc
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=9  Size=6 Exception=0
10
trying to send: 1 6 0 0 0 a 9 cd
waiting response !
 1 6 0 0 0 a 9 cd
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=10  Size=6 Exception=
11
trying to send: 1 6 0 0 0 b c8 d
waiting response !
 1 6 0 0 0 b c8 d
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=11  Size=6 Exception=
12
trying to send: 1 6 0 0 0 c 89 cf
waiting response !
 1 6 0 0 0 c 89 cf
timedout ? false
```

```
commSuccess=true executionSuccess=true Function=6 Value=12  Size=6 Exception=
13
trying to send: 1 6 0 0 0 d 48 f
waiting response !
 1 6 0 0 0 d 48 f
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=13  Size=6 Exception=
14
trying to send: 1 6 0 0 0 e 8 e
waiting response !
 1 6 0 0 0 e 8 e
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=14  Size=6 Exception=
15
trying to send: 1 6 0 0 0 f c9 ce
waiting response !
 1 6 0 0 0 f c9 ce
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=15  Size=6 Exception=
16
trying to send: 1 6 0 0 0 10 88 6
waiting response !
 1 6 0 0 0 10 88 6
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=16  Size=6 Exception=
17
trying to send: 1 6 0 0 0 11 49 c6
waiting response !
```

```
 1 6 0 0 0 11 49 c6
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=17  Size=6 Exception=
18
trying to send: 1 6 0 0 0 12 9 c7
waiting response !
 1 6 0 0 0 12 9 c7
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=18  Size=6 Exception=
19
trying to send: 1 6 0 0 0 13 c8 7
waiting response !
 1 6 0 0 0 13 c8 7
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=19  Size=6 Exception=
20
trying to send: 1 6 0 0 0 14 89 c5
waiting response !
 1 6 0 0 0 14 89 c5
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=20  Size=6 Exception=
21
trying to send: 1 6 0 0 0 15 48 5
waiting response !
 1 6 0 0 0 15 48 5
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=21  Size=6 Exception=
22
```

```
trying to send: 1 6 0 0 0 16 8 4
waiting response !
 1 6 0 0 0 16 8 4
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=22  Size=6 Exception=
23
trying to send: 1 6 0 0 0 17 c9 c4
waiting response !
 1 6 0 0 0 17 c9 c4
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=23  Size=6 Exception=
24
trying to send: 1 6 0 0 0 18 89 c0
waiting response !
 1 6 0 0 0 18 89 c0
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=24  Size=6 Exception=
25
trying to send: 1 6 0 0 0 19 48 0
waiting response !
 1 6 0 0 0 19 48 0
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=25  Size=6 Exception=
26
trying to send: 1 6 0 0 0 1a 8 1
waiting response !
 1 6 0 0 0 1a 8 1
timedout ? false
```

```
commSuccess=true executionSuccess=true Function=6 Value=26  Size=6 Exception=
27
trying to send: 1 6 0 0 0 1b c9 c1
waiting response !
 1 6 0 0 0 1b c9 c1
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=27  Size=6 Exception=
28
trying to send: 1 6 0 0 0 1c 88 3
waiting response !
 1 6 0 0 0 1c 88 3
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=28  Size=6 Exception=
29
trying to send: 1 6 0 0 0 1d 49 c3
waiting response !
 1 6 0 0 0 1d 49 c3
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=29  Size=6 Exception=
30
trying to send: 1 6 0 0 0 1e 9 c2
waiting response !
 1 6 0 0 0 1e 9 c2
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=30  Size=6 Exception=
31
trying to send: 1 6 0 0 0 1f c8 2
waiting response !
```

```
 1 6 0 0 0 1f c8 2
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=31  Size=6 Exception=
32
trying to send: 1 6 0 0 0 20 88 12
waiting response !
 1 6 0 0 0 20 88 12
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=32  Size=6 Exception=
33
trying to send: 1 6 0 0 0 21 49 d2
waiting response !
 1 6 0 0 0 21 49 d2
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=33  Size=6 Exception=
34
trying to send: 1 6 0 0 0 22 9 d3
waiting response !
 1 6 0 0 0 22 9 d3
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=34  Size=6 Exception=
35
trying to send: 1 6 0 0 0 23 c8 13
waiting response !
 1 6 0 0 0 23 c8 13
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=35  Size=6 Exception=
36
```

```
trying to send: 1 6 0 0 0 24 89 d1
waiting response !
 1 6 0 0 0 24 89 d1
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=36  Size=6 Exception=
37
trying to send: 1 6 0 0 0 25 48 11
waiting response !
 1 6 0 0 0 25 48 11
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=37  Size=6 Exception=
38
trying to send: 1 6 0 0 0 26 8 10
waiting response !
 1 6 0 0 0 26 8 10
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=38  Size=6 Exception=
39
trying to send: 1 6 0 0 0 27 c9 d0
waiting response !
 1 6 0 0 0 27 c9 d0
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=39  Size=6 Exception=
40
trying to send: 1 6 0 0 0 28 89 d4
waiting response !
 1 6 0 0 0 28 89 d4
timedout ? false
```

```
commSuccess=true executionSuccess=true Function=6 Value=40  Size=6 Exception=
41
trying to send: 1 6 0 0 0 29 48 14
waiting response !
 1 6 0 0 0 29 48 14
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=41  Size=6 Exception=
42
trying to send: 1 6 0 0 0 2a 8 15
waiting response !
 1 6 0 0 0 2a 8 15
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=42  Size=6 Exception=
43
trying to send: 1 6 0 0 0 2b c9 d5
waiting response !
 1 6 0 0 0 2b c9 d5
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=43  Size=6 Exception=
44
trying to send: 1 6 0 0 0 2c 88 17
waiting response !
 1 6 0 0 0 2c 88 17
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=44  Size=6 Exception=
45
trying to send: 1 6 0 0 0 2d 49 d7
waiting response !
```

```
 1 6 0 0 0 2d 49 d7
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=45  Size=6 Exception=
46
trying to send: 1 6 0 0 0 2e 9 d6
waiting response !
 1 6 0 0 0 2e 9 d6
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=46  Size=6 Exception=
47
trying to send: 1 6 0 0 0 2f c8 16
waiting response !
 1 6 0 0 0 2f c8 16
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=47  Size=6 Exception=
48
trying to send: 1 6 0 0 0 30 89 de
waiting response !
 1 6 0 0 0 30 89 de
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=48  Size=6 Exception=
49
trying to send: 1 6 0 0 0 31 48 1e
waiting response !
 1 6 0 0 0 31 48 1e
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=49  Size=6 Exception=
50
```

```
trying to send: 1 6 0 0 0 32 8 1f
waiting response !
 1 6 0 0 0 32 8 1f
timedout ? false
commSuccess=true executionSuccess=true Function=6 Value=50  Size=6 Exception=

read test:

trying to send: 1 3 0 0 0 1 84 a
waiting response !
 1 3 2 0 32 39 91
timedout ? false
commSuccess=true executionSuccess=true Function=3 Value=50  Size=5 Exception=
BUILD SUCCESSFUL (total time: 3 seconds)
```

C.4 Teste do Uso Incorreto da Implementação de Modbus entre o Computador e a Placa Arduino™

```
   .

run:

first test:

trying to send: 2 6 0 0 0 1 48 39
waiting response !

trying to send: 2 6 0 0 0 1 48 39
```

```
waiting response !

trying to send: 2 6 0 0 0 1 48 39
waiting response !

trying to send: 2 6 0 0 0 1 48 39
waiting response !

trying to send: 2 6 0 0 0 1 48 39
waiting response !

timedout ? true
commSuccess=false executionSuccess=false Function=0 Value=null Size=0 Excepti

second test:

trying to send: 1 6 0 5 0 1 58 b
waiting response !
 1 86 2 c3 a1
timedout ? false
commSuccess=true executionSuccess=false Function=134 Value=null Size=3 Except

third test:

CRC: 480a
SABOTAGED CRC: 4809
trying to send: 1 6 0 0 0 1 48 9
waiting response !
```

```
trying to send: 1 6 0 0 0 1 48 9
waiting response !

trying to send: 1 6 0 0 0 1 48 9
waiting response !

trying to send: 1 6 0 0 0 1 48 9
waiting response !

trying to send: 1 6 0 0 0 1 48 9
waiting response !

timedout ? true
commSuccess=false executionSuccess=false Function=0 Value=null Size=0 Excepti

fourth test:

trying to send: 1 3 0 5 0 1 94 b
waiting response !
 1 83 2 c0 f1
timedout ? false
commSuccess=true executionSuccess=false Function=131 Value=null Size=3 Except

fifth test:

trying to send: 1 3 0 4 0 2 85 ca
waiting response !
```

```
 1 83 2 c0 f1
timedout ? false
commSuccess=true executionSuccess=false Function=131 Value=null Size=3 Except

sixth test:

trying to send: 1 3 0 0 1 d8 44
waiting response !
 1 83 3 1 31
timedout ? false
commSuccess=true executionSuccess=false Function=131 Value=null Size=3 Except
BUILD SUCCESSFUL (total time: 6 seconds)
```