UMA ABORDAGEM COMPARATIVA ENTRE MICROCONTROLADORES: ARDUINO MEGA X ARDUINO DUE APLICADOS NO CONTROLE DE SOCCER ROBOTS

Jusoan Mór*, Everson Siqueira*, Cristiano Steffens*, Paulo Jefferson Dias de Oliveira Evald*, Vinícius Menezes de Oliveira*, Silvia Silva da Costa Botelho*, Rodrigo Azzolin*

* *Universidade Federal do Rio Grande - FURG*
Km 8 - Av. Itália - Carreiros, 96203-900, Rio Grande, Rio Grande do Sul, Brasil

Emails: jmor@furg.br, everson.brumm@furg.br, cristiano@furg.br, paulo.evald@gmail.com, vinicius@furg.br, silviacb@furg.br, rodrigoazzolin@furg.br

Abstract— Improving the positioning, power consumption and team performance of small size soccer robots is a laborious task. Otherwise, stationary robots or robots attached to rack and pinion structures, soccer robots can run freely on a match field and are susceptible to slips and collisions that impact their actuation in terms of speed and positioning. High performance closed loop controling alternatives become essential in such scenarios. We implement and evaluate the feasibility of the standard Proportional-Integral (PI), Proportional-Integral with Recursive Least Squares (PI-RLS) and Robust Model Reference Adaptive Control (RMRAC) on two models of Arduino prototiping boards. Given time constraints imposed by the soccer game dynamics, it was showed that Arduino DUE can be used to implement any of aforementioned controllers, while Arduino MEGA does not show sufficient processing capabilities to execute the calculi for PI-RLS and RMRAC controllers with sampling rate of $1ms$.

Keywords— Robot Soccer, Control, PI, PI-RLS, RMRAC

Resumo— Melhorar o posicionamento, consumo de energia e desempenho dos robôs de futebol da classe *small size* é uma tarefa trabalhosa. Diferente de robôs estacionários ou ligados a cremalheiras ou pinhões estruturais, robôs de futebol correm livremente no campo de jogo e são suscetíveis a deslizamentos e colisões, que impactam na sua atuação em termos de velocidade e posicionamento. Alternativas de controle de alto desempenho em malha fechada são essenciais nesse cenário. Portanto, neste trabalho foi implementada e avaliada a viabilidade da utilização de três controladores diferentes, aplicados em dois modelos de placas de prototipagem *Arduino*. Os controladores aplicados foram: O controlador Proporcional-Integral (PI), O controlador Proporcional-Integral com Recursividade por Mínimos Quadrados (PI-RLS) e o Controle Adaptativo Robusto por Modelo de Referência (RMRAC). Dadas as limitações de tempo inerentes a dinâmica do jogo de futebol, foi mostrado que o *Arduino DUE* pode ser utilizado para a implementação de qualquer um dos controladores avaliados, enquanto o *Arduino MEGA* não apresentou capacidade de processamento suficiente para executar os controladores PI-RLS e RMRAC em uma janela de tempo de $1ms$.

Palavras-chave— Futebol de Robôs, Controle, PI, PI-RLS, RMRAC

1 Introdução

O futebol de robôs na categoria *small size* é composto por duas equipes de seis robôs em cada equipe, que se enfrentam em um campo de 6,05 por 4,05 metros. Cada robô não deve possuir mais de 180 milímetros de diâmetro e não deve ultrapassar 150 milímetros de altura. A movimentação dos robôs e da bola é capturada por um par de câmeras localizadas a 4 metros de altura sobre o campo. Todas as informações são processadas por um computador e enviadas aos robôs via rádio, como observado na Figura 1. Nenhuma interferência externa de humanos é permitida, apenas em substituições de robôs (2012 ROBOCUP.ORG.BR, 2016). A Universidade Federal do Rio Grande - FURG possui uma equipe de futebol de robôs dessa categoria, conhecida pelo nome de *Furgbol*. O *hardware* e *software* dos robôs foram desenvolvidos na própria instituição. A estrutura dos robôs é composta por uma carcaça de Alumínio, quatro motores *Brushless Direct Current Motor* - BLDC encarregados pela movimentação das rodas omnidirecionais do robô, conforme pode ser visualizado na Figura 2. O robô possui diversos módulos eletrônicos, tais como: os *drivers*, microcontrolador e o receptor de rádio. Os *drivers* são utilizados para o acionamento dos motores, enquanto que o microcontrolador atua no controle dos diversos componentes do sistema e o receptor de rádio adquire as referências calculadas pelo computador central.

Os motores descritos são construídos com três sensores de efeito *Hall* embutidos em sua carcaça, responsáveis pela medição de velocidade e sentido de giro dos motores (Simpkins and Todorov, 2010). A utilização desses sensores para aquisição de velocidade implica na não linearidade do sistema (Siegwart et al., 2011). Uma das restrições do projeto é a necessidade de utilização de uma placa de prototipagem rápida, que deve executar todo o sistema do robô no tempo de $1ms$ afim de que a medida de velocidade seja válida e permita a ação de correção.

Para o controle dos motores do *Furgbol* é utilizado um controlador do tipo Proporcional e Integral - PI (Srisabye et al., 2009; Wasuntapichaikul et al., 2010). Além disso, os controladores adap-

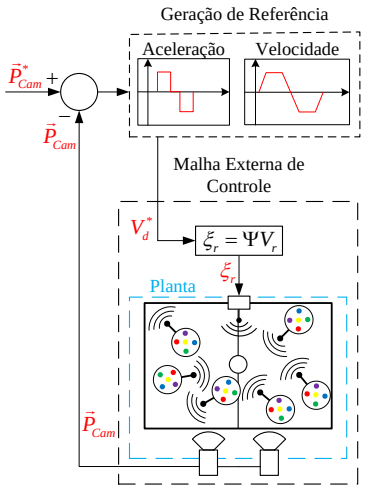

Figura 1: Geração de referência e malha externa do controle de futebol de robôs - *Furgbol*.

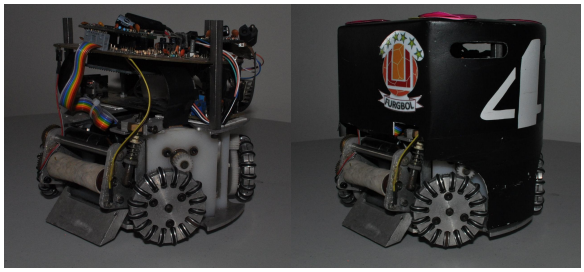

Figura 2: Estrutura do robô do *Furgbol*.

tativos do tipo PI (*Recursive Least Square*) - PI-RLS (Bernat and Stepien, 2011; Lin, 1996) e Controle Adaptativo Robusto por Modelo de Referência (*Robust Model Reference Adaptive Controller*) - RMRAC (Ioannou and Tsakalis, 1986; Ioannou and Sun, 2012), são utilizados como alternativas para melhorar o desempenho do sistema.

Este trabalho tem o objetivo de comparar o desempenho dos microcontroladores *Arduino MEGA* (Site oficial ©2015 Arduino, 2015b) e *Arduino DUE* (Site oficial ©2015 Arduino, 2015a), aplicados como controladores nos robôs do *Furgbol*. Inicialmente, o *Arduino MEGA* foi utilizado no controle de movimentação dos robôs. Entretanto, para a implementação dos controladores adaptativos visando melhorar o desempenho dos robôs em competições, mostrou-se necessário realizar a substituição do *Arduino MEGA* pelo *Arduino DUE*, visto que a implementação de controladores adaptativos incitou o comportamento de saturação (o *Arduino* não consegue processar todos os dados dentro do tempo pré-determinado) no *Arduino MEGA*.

2 Controladores

Inicialmente, para o controle das rodas do robôs do *Furgbol*, um controlador do tipo PI foi implementado, entretanto, o sistema apresentou baixo desempenho frente à dinâmicas não modeladas, como por exemplo o escorregamento da roda. Visando melhorar o desempenho é proposto a implementação dos controladores PI-RLS e RMRAC apresentados nessa seção. Em todos os controles citados anteriormente é necessário a modelagem completa do motor descrita em (Siqueira et al., 2015), onde foi observado a equação que representa a função de transferência do motor BLDC, mostrado em (1). A função de transferência descrita nesse sistema relaciona o sistema mecânico e o sistema elétrico. O sistema mecânico contempla o conjunto de variáveis torque geral T_g, torque de distúrbio T_d, torque total T_t, velocidade angular ω, constante de inércia J, constante de torque K_m e o coeficiente de atrito mecânico do motor b. O sistema elétrico demonstra a tensão de entrada V_r, corrente da armadura i_a, a resistência da armadura R_a, a indutância da armadura L_a e a constante de força contra-eletromotriz K_b.

$$G(s) = \frac{K_m}{s^2 J L_a + s(L_a b + R_a J) + R_a b + K_b K_m} \quad (1)$$

2.1 Controlador PI

No controlador PI, a velocidade do motor é medida e comparada com a velocidade de referência, gerando um erro. Esse erro é utilizado como entrada do bloco PI que produz uma ação de controle aplicado no motor. A estrutura da planta de controle pode ser observada na Figura 3.

O controlador utiliza a ação de dois ganhos $K_P = 12$ e $K_I = 250$ modelados de acordo com o método de alocação de pólos (Ogata, 2001).

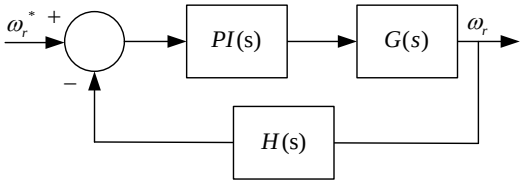

Figura 3: Estrutura do controlador PI

2.2 Controlador PI-RLS

O controlador PI-RLS, conforme pode ser visualizado na Figura 4, possui um algoritmo recursivo que estima os parâmetros do motor. Isso permite calcular dinamicamente os ganhos K_P e K_I do controlador PI (Siqueira et al., 2015).

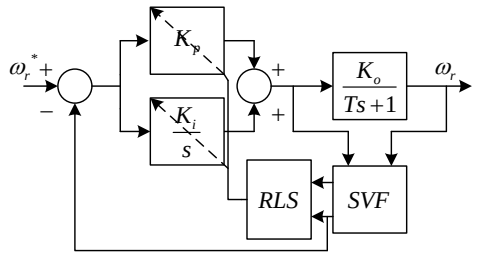

Figura 4: Estrutura do controlador PI-RLS.

O custo computacional do RLS é elevado devido ao número de equações necessárias para execução do algoritmo. Visando reduzir o custo, considerou-se a função de transferência do motor de primeira ordem conforme (2), onde K_o e T representam respectivamente o ganho do sistema e constante do tempo.

$$G(s) = \frac{K_o}{Ts+1} \tag{2}$$

O algoritmo RLS é um algoritmo de estimação paramétrica que requer o modelo da planta (motor) em tempo discreto (Zhang et al., 2013) e na forma de regressão linear (Regazzi, 2015). O modelo da regressão linear está presente na Equação (3), onde k indica a amostragem atual no tempo discreto.

$$\hat{Y}_{(k)} \;=\; C_{(k)}\hat{\theta}_{(k)} \tag{3}$$

As variáveis $\hat{Y}_{(k)}$, $C_{(k)}$, $\hat{\theta}_{(k)}$ são o vetor de predição, matriz de regressão linear e vetor de parâmetros, respectivamente. O algoritmo de identificação recursiva pode ser obtido através da relação presente nas Equações (4), (5) e (6).

$$\hat{\theta}_{(k)} \;=\; \hat{\theta}_{(k-1)} \;+\; K_{(k)}e_{(k)} \tag{4}$$

$$K_{(k)} \;=\; \frac{P_{(k-1)}C_{(k)}}{I + C_{(k)}^T P_{(k-1)}C_{(k)}} \tag{5}$$

$$P_{(k)} \;=\; [I - K_{(k)}C_{(k)}]P_{(k-1)} \tag{6}$$

Sendo que K e P representam, respectivamente, a matriz de ganhos e a matriz de covariância do RLS. A variável e é a diferença dos valores medido e estimado.

2.3 Controlador RMRAC

O controlador RMRAC, apresentado na Figura 5, utiliza um modelo de referência com o mesmo grau da planta desejada, sendo esse o fator que define a resposta dinâmica desejada para a saída da planta. Um erro e_1 é utilizado pelo algoritmo de adaptação para ajustar os parâmetros do controlador, adquirido a partir da saída do modelo de referência e da saída da planta. Este ajuste acontece de um certo modo que o erro entre a saída do modelo de referência y_m e a saída da planta y_p tende a um valor muito pequeno, sendo na média quase nulo (Ioannou and Tsakalis, 1986; Ioannou and Sun, 2012; Lozano-Leal et al., 1990; Mór et al., 2013). Também estão presentes no modelo a entrada da planta u_p, a entrada de referência de velocidade r e a saída dos filtros ω_1 e ω_2.

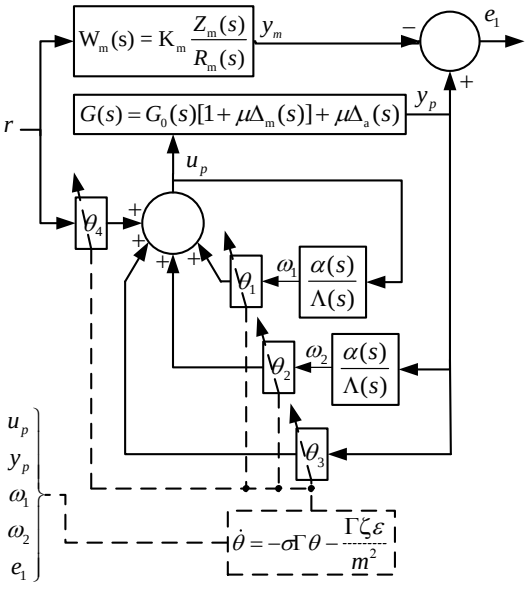

Figura 5: Estrutura do controlador RMRAC.

Além disso, a Equação (7) apresenta uma função de transferência de segunda ordem comumente utilizada em modelos de controle.

$$W(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{7}$$

Ao atribuir na Equação (7) os valores $\xi = 0.7$, $\omega_n = 50$ pode-se aferir o modelo de referência $W_m(s)$.

$$W_m(s) = \frac{2500}{s^2 + 70s + 2500} \tag{8}$$

Os parâmetros do controlador RMRAC podem ser encontrados nas Equações (9) até (16).

$$\delta_0 = 0.7, \quad \delta_1 = 1, \quad \theta(0) = [0, 0, 0, 0.2] \tag{9}$$

$$m(0) \geq \frac{\delta_1}{\delta_0} = 1.4286, \quad \Gamma = 2, \quad \sigma_0 = 0.1 \tag{10}$$

$$\theta^T = [\theta_1, \theta_2, \theta_3, \theta_4], \quad \omega^T = [\omega_1, \omega_2, y_m, y_p] \tag{11}$$

$$\Lambda(s) = s + 1, \quad M_0 = 0.5 \tag{12}$$

$$\alpha(s) \triangleq \alpha_{n-2} = [s^{n-2}, s^{n-3}, ..., s, 1]^T \quad para\ n \geq 2 \tag{13}$$

$$\alpha(s) \triangleq 0 \quad para\ n = 1$$

$$\varepsilon = e_1 + \theta^T\zeta - W_m(s)\theta^T\omega \tag{14}$$

$$\dot{m} = -\delta_0 m + \delta_1(|u_p| + |y_p| + 1) \quad (15)$$

$$\sigma = \begin{cases} 0 & \text{if} \quad \|\theta\| < M_0 \\ \sigma_0(\frac{\|\theta\|}{M_0} - 1) & \text{if} \quad M_0 \leq \|\theta\| < 2M_0 \\ \sigma_0 & \text{if} \quad \|\theta\| \geq 2M_0 \end{cases} \quad (16)$$

onde $M_0 > \|\theta^*\|$ e $\sigma_0 > 2\mu^{-2}/R^2 \in \Re_+$ são parâmetros de projeto.

3 Arquitetura dos Microcontroladores

3.1 Arduino MEGA

Arduino MEGA 2560 é um microcontrolador baseado no *ATMEGA 2560*, possui 54 pinos digitais dos quais 15 são pinos PWM (*Pulse-Width Modulation*), 16 pinos analógicos de entrada, 4 UARTs (*Universal Asynchronous Receiver/Transmitter*), um oscilador de $16MHz$, um ICSP *header* e um botão *reset*.

Para maximizar a performance e o paralelismo, o *Atmel AVR* utiliza a arquitetura *Harvard*, ou seja, possui duas memórias separadas, um para programa e outra para os dados. Instruções na memória do programa são executadas com apenas um nível de *pipeline*. Enquanto uma instrução está sendo executada, a próxima instrução é realocada para a memória do programa. Esse conceito permite que mais de uma instrução seja executada a cada ciclo de *clock*.

Os registradores de acesso rápido de propósito geral trabalham com apenas um acesso por ciclo, permitindo a execução de um ciclo de operação da Unidade Lógica Aritmética (ULA). Em uma típica operação da ULA, dois operandos geram uma saída para o registrador e o resultado é armazenado no registrador de entrada em apenas um ciclo de operação da ULA.

Dos 32 registradores, apenas 6 podem ser usados como 3 registradores indiretos que apontam para um dado no espaço de endereçamento. Um desses apontadores de endereço pode ser utilizados para procurar programas na memória *flash*.

A ULA suporta operações lógicas e aritméticas entre os registradores e entre uma constante e um registrador. Após uma operação aritmética, o *status* do registrador é atualizado para obter a informação a respeito do resultado da operação.

3.2 Arduino DUE

O *Arduino Due* é uma placa de microcontrolador baseado no processador*Atmel SAM3X8E ARM Cortex-M3 CPU*. É a primeira placa *Arduino* baseada em um microcontrolador *ARM* de 32 *bits*. Tem 54 pinos de entrada e saída digital, dos quais 12 podem ser utilizados como saídas PWM, 12 entradas analógicas, 4 UARTs, *clock* de $84MHz$, uma conexão USB OTG (o*n the go*), 2 DAC (*digital to analog converter*), 2 TWI (*two wire interface*), uma entrada de alimentação, um barramento SPI, um barramento JTAG, um botão de *reset*, e um botão de *erase*.

O processador fornece um grande poder de processamento devido a seu *design* eficiente e otimizado. O processador pode realizar multiplicação de 32×32 em um único ciclo de processamento com *hardware* dedicado. O acesso a memória é feito com a tecnologia *Advanced Microcontroller Bus Architecture* (AMBA) (ARM Ltd. Copyright 1995-2016, 2016), que fornece baixa latência e grande velocidade. Uma das vantagens desse processador é a possibilidade de acesso desalinhado a memória, além do controle rápido de periféricos usando a manipulação de *bit* atômico e de *spinlocks* do sistema e manipulação de dado booleano *thread-safe*. O processador tem uma unidade de proteção a memória (MPC), que fornece controle de memória de granulação fina, permitindo que aplicações implemente privilégios de níveis de segurança, ou seja, separando código, dados e pilha em uma base tarefa-por-tarefa.

4 Resultados

Nesta seção são apresentados os resultados práticos, obtidos por leituras diretas em um osciloscópio, onde uma porta do microcontrolador foi utilizada como um *flag* de controle. No início da rotina de controle, o *flag* indica sinal alto, após a execução de todo o código o *flag* passa para sinal baixo. A Figura 6, mostra a bancada de teste, com o osciloscópio, um robô *Furgbol*, e os microcontroladores *Arduino MEGA* e *DUE*. Uma placa de adaptação é utilizada para a captação dos dados, em que, um pino *GND* e o pino de *flag* são de fácil acesso. Além disso, uma função de interrupção é utilizada nos controladores do *Furgbol*, onde o controle deve ser implementado a cada $1ms$ para a correta comunicação via rádio e medição de velocidade.

O tempo de processamento de execução do controlador PI aplicado no *Arduino MEGA* e o *Arduino DUE* pode ser observado nas Figuras 7 e 8, respectivamente. Observa-se, que ambos os microcontroladores conseguem processar o controle dentro do tempo da interrupção, entretanto, é possível notar que o *Arduino MEGA* necessita praticamente do triplo do tempo que o *Arduino DUE*.

O tempo de processamento de execução do controlador PI-RLS aplicado ao *Arduino MEGA* e *DUE* pode ser visualizado nas Figuras 9 e 10, respectivamente. Observa-se, que o *Arduino MEGA* não conseguiu processar completamente o controle PI-RLS dentro do tempo da interrupção estipulado. Entretanto, como esperado, o *Arduino DUE* obteve um desempenho satisfatório e com uma janela tempo restante equivalente a $630\mu s$.

Nas Figuras 11 e 12 pode-se observar o tempo de processamento do controlador RMRAC aplicado no *MEGA* e no *DUE*, respectivamente.

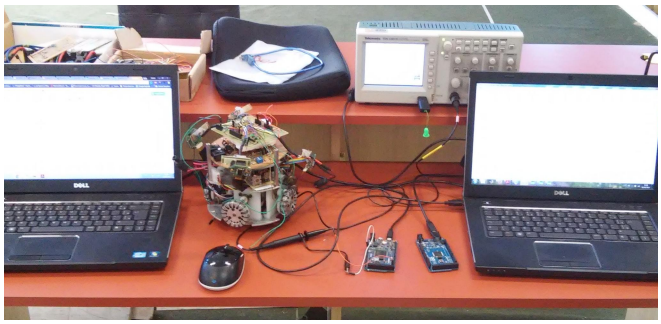

Figura 6: Bancada de testes.

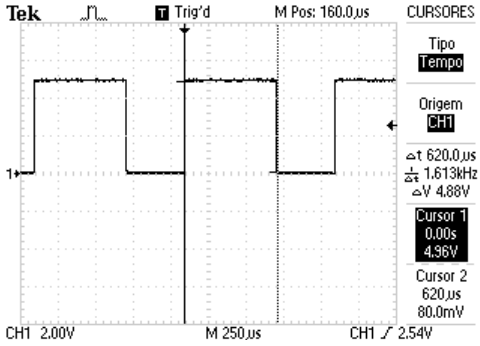

Figura 7: Controle PI aplicado no *Arduino MEGA*.

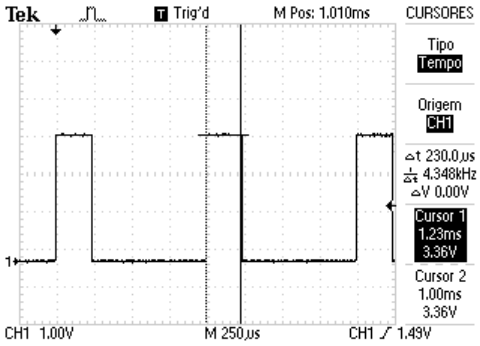

Figura 8: Controle PI aplicado no *Arduino DUE*.

Observa-se, que o *MEGA* não processou o controlador RMRAC no tempo de interrupção e que o *DUE* conseguiu processar, mas com uma janela de tempo restante de $170\mu s$.

5 Conclusão

Nesse trabalho foram apresentados os principais microcontroladores e técnicas de controle utilizadas pela equipe de Futebol de Robôs da FURG - *Furgbol*. Os aspectos abordados nesse trabalho foram quanto a estrutura dos robôs, padronizações para participações de competições e controle dos robôs. O controlador atualmente implementado nos robôs é o PI no microcontrolador *Arduino MEGA*. Para melhorar a performance do robô diante de distúrbios consequentes de derrapagens e deslizamentos, foram implementados os

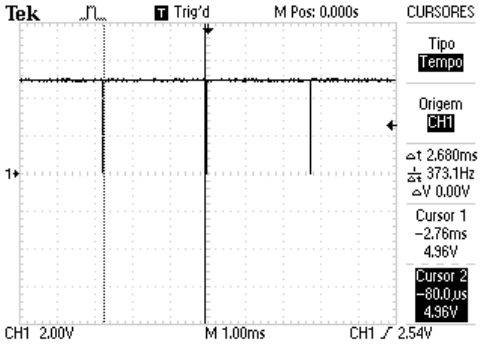

Figura 9: Controle PI-RLS aplicado no *Arduino MEGA*.

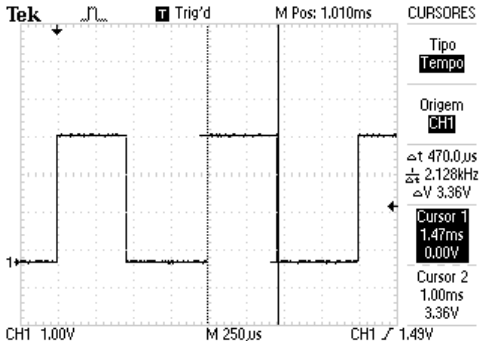

Figura 10: Controle PI-RLS aplicado no *Arduino DUE*.

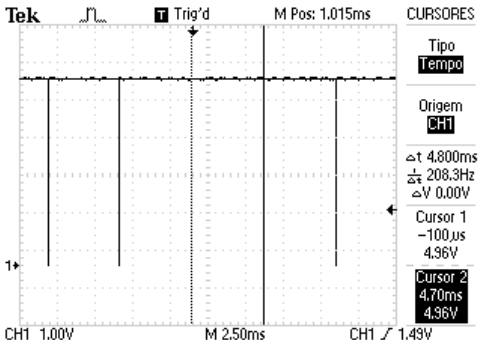

Figura 11: Controle RMRAC aplicado no *Arduino MEGA*.

controladores PI-RLS e o RMRAC no *Arduino MEGA*. Visando detectar os problemas apresentados, foram realizados testes utilizando um os-

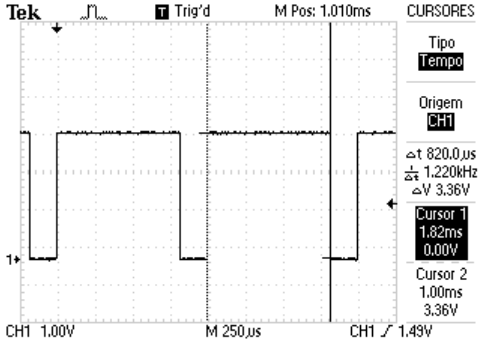

Figura 12: Controle RMRAC aplicado no *Arduino DUE*.

ciloscópio para verificar o tempo de execução de cada controlador aplicado aos *Arduinos MEGA* e *DUE*. Como resultado, verificou-se que o *MEGA* não conseguia processar os controladores PI-RLS e RMRAC. A solução encontrada foi a de substituir o *Arduino MEGA* pelo *Arduino DUE*. Como consequência, os controladores propostos puderam ser implementados e a performance do robô pode ser melhorada.

6 Agradecimentos

Os autores gostariam de agradecer ao apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes).

Referências

2012 ROBOCUP.ORG.BR (2016). RoboCup Brasil, `http://www.robocup.org.br`. [Online; acessado 10-março-2016].

ARM Ltd. Copyright 1995-2016 (2016). AMBA Specifications, `http://www.arm.com/products/system-ip/amba-specifications.php`. [Online; acessado 10-março-2016].

Bernat, J. and Stepien, S. (2011). Application of optimal current driver for the torque control of bldc motor, *Archives of Electrical Engineering* **60**(2): 149.

Ioannou, P. A. and Sun, J. (2012). *Robust adaptive control*, Courier Corporation.

Ioannou, P. A. and Tsakalis, K. S. (1986). A robust direct adaptive controller, *Automatic Control, IEEE Transactions on* **31**(11): 1033–1043.

Lin, F.-J. (1996). Robust speed-controlled induction-motor drive using ekf and rls estimators, *Electric Power Applications, IEE Proceedings-*, Vol. 143, IET, pp. 186–192.

Lozano-Leal, R., Collado, J. and Mondie, S. (1990). Model reference robust adaptive control without a priori knowledge of the high frequency gain, *Automatic Control, IEEE Transactions on* **35**(1): 71–78.

Mór, J. L., Siqueira, E. B., Oliveira, V. M. d. and Azzolin, R. Z. (2013). Controle de velocidade de motor bldc utilizando rmrac, *7th Seminário de Eletrônica de Potência e Controle* .

Ogata, K. (2001). *Modern control engineering*, Prentice Hall PTR.

Regazzi, A. J. (2015). Teste para verificar a igualdade de parâmetros e a identidade de modelos de regressão não-linear/test for parameter equality in nonlinear regression models, *Ceres* **50**(287).

Siegwart, R., Nourbakhsh, I. and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*, MIT Press.

Simpkins, A. and Todorov, E. (2010). Position estimation and control of compact bldc motors based on analog linear hall effect sensors, *American Control Conference (ACC), 2010*, IEEE, pp. 1948–1955.

Siqueira, E. B., Mor, J. L., Azzolin, R. Z. and de Oliveira, V. M. (2015). Algorithm to identification of parameters and automatic re-project of speed controller of bldc motor, *IFAC-PapersOnLine* **48**(19): 256–261.

Site oficial ©2015 Arduino (2015a). Arduino DUE, `http://www.arduino.cc/en/Main/arduinoBoardDue`. [Online; acessado 05-março-2016].

Site oficial ©2015 Arduino (2015b). Arduino MEGA, `http://www.arduino.cc/en/Main/arduinoBoardMega2560`. [Online; acessado 05-março-2016].

Srisabye, J., Wasuntapichaikul, P., Onman, C., Sukvichai, K., Damyot, S., Munintarawong, T., Phuangjaisri, P. and Tipsuwan, Y. (2009). Skuba 2009 extended team description, *Proceedings of RoboCup* .

Wasuntapichaikul, P., Srisabye, J., Onman, C., Damyot, S., Areeprasert, C. and Sukvichai, K. (2010). Skuba 2010 extended team description, *Proceeding of Robocup* .

Zhang, H., Gong, S.-j. and Dong, Z.-z. (2013). Online parameter identification of induction motor based on rls algorithm, *Electrical Machines and Systems (ICEMS), 2013 International Conference on*, IEEE, pp. 2132–2137.