

# Projeto Final de Curso

**André Rodrigues, Germano Leite, Juliana Nunes**

## 1 Introdução

Na medida em que o uso de sistemas computacionais aumenta na sociedade atual, aplicações com exigências de tempo real tornam-se cada vez mais comuns. Essas aplicações variam muito em relação à complexidade e às necessidades de garantia no entendimento de restrições temporais. Existem sistemas de tempo real simples, como os controladores inteligentes embutidos em utilidades domésticas, tais como lavadora de roupa e videocassetes e existem sistemas de complexidade maior, como os sistemas militares de defesa, sistemas de controle de plantas industriais (químicas e nucleares) e controle de tráfego aéreo e ferroviário.

Em aplicações de tempo real, também é importante ressaltar que existem sistemas que apresentam restrições de tempo mais rigorosas do que outras, por exemplo, os sistemas responsáveis pelo monitoramento de pacientes em hospitais, sistemas embarcados em robôs e veículos (de automóveis até aviões) e há também sistemas que não apresentam restrições tão rigorosas, por exemplo, videogames, teleconferências através da internet e aplicações de multimídia. Todas essas aplicações que apresentam a característica adicional de estarem sujeitas a restrições temporais são agrupadas no que é usualmente identificado como Sistema de Tempo Real.

Diante dessa premissa, com o intuito de aplicar os conhecimentos adquiridos na disciplina de Sistemas de Tempo Real desenvolvemos neste projeto um robô que possui tarefas a serem executadas em tempo real. Utilizamos a plataforma Arduino, diversos sensores e uma biblioteca para auxiliar no escalonamento de tarefas. Neste documento são detalhadas todas as etapas do desenvolvimento do

robô. O documento foi organizado da seguinte forma: A seção 2 explica como o robô foi construído, a seção 3 apresenta as tarefas do robô e mostra como elas foram implementadas, a seção 4 expõe os resultados obtidos e por fim, a seção 5 apresenta a conclusão do trabalho.

## 2 Construção do Robô

Para a construção estrutural do robô utilizamos materiais fornecidos gentilmente pelo Professor Douglas Macharet (UFMG/DCC) e também materiais adquiridos pelos alunos integrantes deste trabalho. Nas subseções 2.1 e 2.2 descrevemos os materiais e métodos utilizados para o desenvolvimento do robô

### 2.1 Materiais

Para arquitetar a estrutura do robô utilizamos peças de Lego (Figura 1). Um kit de peças de lego como esse têm como uma de suas finalidades a construção de robôs, oferecem uma grande quantidade de peças que facilitam o projeto e tornam mais prática a sua execução.



Figura 1: Kit de Lego.

Para permitir que o robô realize as tarefas é necessário uma plataforma que permita integrar os sensores (Figuras 2, 3, 4 e 5) que precisamos para a execução das mesmas. A plataforma escolhida foi o Arduino (Figura 6), uma plataforma eletrônica de código aberto.

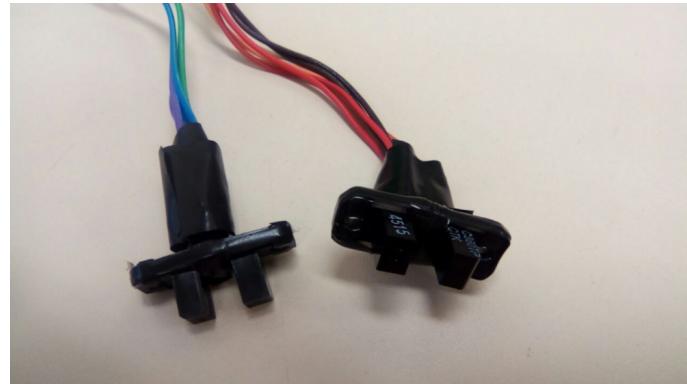


Figura 2: Sensores Break Beam



Figura 3: Sensores Óptico Reflexivos

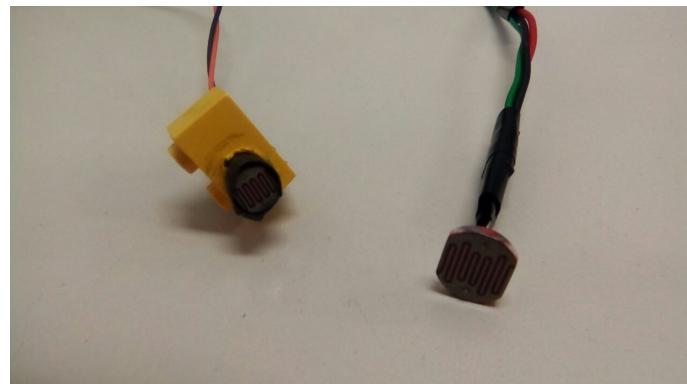


Figura 4: Sensores LDR

Para realizar o controle de motores com nossa placa Arduino, adquirimos um Motor Shield - Driver Ponte H (Figura 7).



Figura 5: Sensores Ultrassônicos

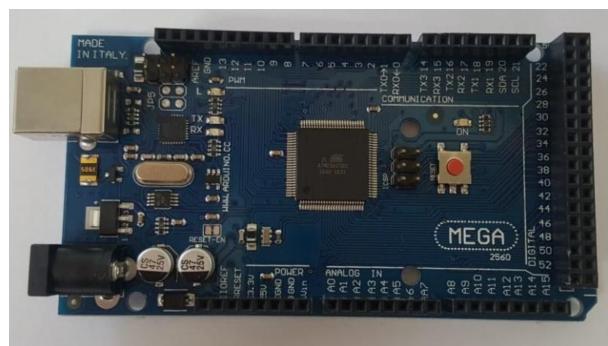


Figura 6: Plataforma Arduino

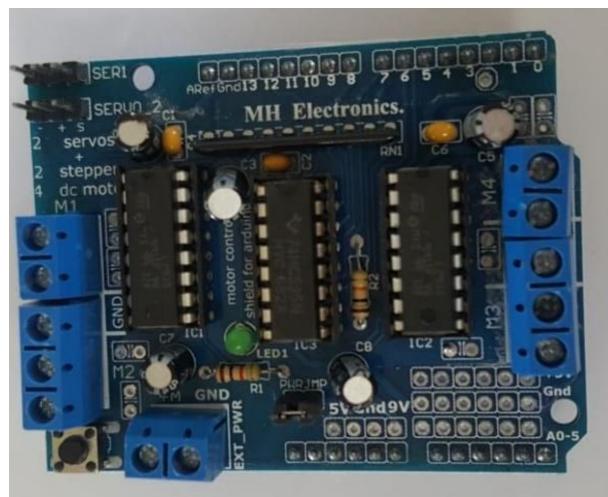


Figura 7: Motor Shield - Driver Ponte H

Após a montagem da estrutura e alocação dos sensores o robô ficou como mostrado na Figura 8.

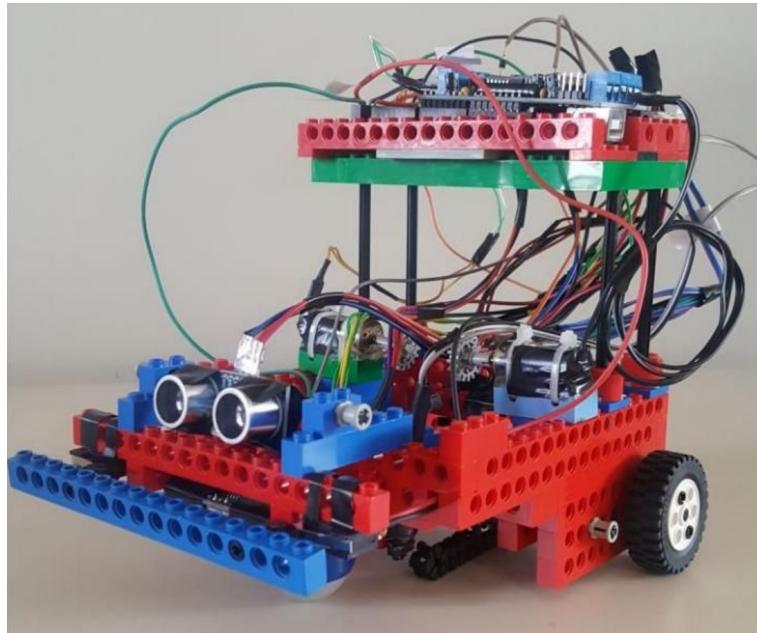


Figura 8: Robô com estrutura finalizada.

## 2.2 Métodos

Para a montagem da estrutura do robô utilizamos o kit de Legos de forma a deixar o Arduíno, o Motor Shield e os sensores bem adaptados. Para a montagem e soldagem dos sensores nós utilizamos tutoriais ilustrados que podem ser facilmente obtidos na internet. Todos os sensores foram testados e apresentaram boa performance.

// Melhorar.

## 3 Tarefas do Robô

Nas duas subseções seguintes nós detalhamos as tarefas que o robô executará e como elas foram implementadas.

### 3.1 Tarefas e Periodicidade

Uma tarefa de tempo real, além da correção lógica, deve satisfazer seus prazos e restrições temporais. As restrições temporais, as relações de precedência e de exclusão usualmente impostas sobre tarefas são determinantes na definição de um modelo de tarefas que é parte integrante de um problema de escalonamento.

Para que se tenha um comportamento temporal adequado, todas as tarefas de tempo real tipicamente estão sujeitas a prazos (deadlines). A princípio, uma tarefa deve ser concluída antes do seu deadline. As consequências de uma tarefa ser concluída após o seu deadline define dois tipos de tarefas de tempo real:

- **Tarefa Crítica**, quando ao ser completada depois de seu deadline pode causar falhas catastróficas no sistema de tempo real e em seu ambiente.
- **Tarefa Não-crítica**, quando ao ser completada depois de seu deadline no máximo implica numa diminuição de desempenho do sistema.

Outra característica temporal de tarefas em sistemas de tempo real está baseada na periodicidade de suas atividades. Os modelos comportam dois tipos de tarefas neste aspecto, são eles:

- **Tarefa Periódica**: Tarefa que possui periodicidade de ativação (e não de execução) constante;
- **Tarefa Aperiódica**: Tarefa cujo intervalo de tempo entre ativações consecutivas não tem nenhum mínimo definido. Para poder ser considerada a utilização de tarefas aperiódicas em sistemas de tempo real, torna-se necessário impor um limite superior à sua utilização de recursos computacionais.

Na Tabela 1 apresentamos as tarefas a serem implementadas juntamente com seus requisitos temporais.

Tabela 1: Tarefas e seus requisitos temporais

Nº	Tarefa	Modelo	Tempo de Execução (segundos)	Prioridade
1	Andar	Não crítica Periódica		
2	Detectar e seguir uma linha	Não crítica Periódica		
3	Verificar obstáculo	Não crítica Periódica		
4	Manutenção da velocidade utilizando um controle PID	Não crítica Periódica		
5	Verificar superfície	Não Crítica Periódica		
6	Detectar contato com objetos	Não crítica Periódica		
7	Desviar de obstáculos	Crítica Aperiódica		
8	Evitar queda de superfícies	Crítica Aperiódica		
9	Em caso de contato com objeto, afastar do mesmo	Crítica Aperiódica		

### 3.2 Implementação e Escalonamento das Tarefas

Nesta seção mostraremos como as tarefas apresentadas na seção 3.1 foram implementadas. A primeira etapa foi verificar se existia algum algoritmo de escalonamento presente na literatura que se adaptasse ao nosso projeto. O algoritmo *Rate Monotonic*, bastante conhecido, é um algoritmo que possui características que se adequam ao nosso projeto, portanto, usamos este algoritmo como base para implementação do nosso código.

As principais características do algoritmo *Rate Monotonic* são:

- As tarefas são periódicas e independentes;
- O deadline de cada tarefa coincide com o seu período;
- O tempo de computação de cada tarefa é conhecido e constante (*Worst Case Computation Time*);
- O tempo de chaveamento entre tarefas é assumido nulo;
- Dá maior prioridade às tarefas de menor período.

Nossas tarefas são em sua maioria periódicas, porém, três destas tarefas são aperiódicas. Para contornar esta situação e poder fazer uso do algoritmo *Rate Monotonic* nós criamos um servidor de tarefas aperiódicas. Este servidor, é tratado como uma tarefa periódica.

O algoritmo funciona da seguinte maneira: Todas as tarefas periódicas são executadas por tempo pré-definido, inclusive a tarefa referente ao servidor de aperiódicas. Para controle de execução das tarefa aperiódicas têm-se uma variável booleana global que auxilia para execução ou não de uma tarefa aperiódica. Este controle funciona da seguinte forma: Quando as tarefas **Verificar obstáculo**, **Detectar contato** e **Verificar superfície** são executadas elas retornam um valor. De acordo com o valor retornado, a variável booleana global recebe o valor *True* (por exemplo, a variável booleana está setada como *False*, se ao executar a tarefa de **Verificar obstáculo** o sensor apresentar um valor que é um valor de risco, esta variável recebe o valor *True*), se ao ser executada a tarefa referente ao servidor de aperiódicos, a variável booleana estiver com valor *True*, a tarefa aperiódica **Desviar de obstáculos** é executada. Ao final da execução da tarefa aperiódica, a variável booleana recebe o valor *False* novamente.

//Melhorar //Colocar pseudo-algoritmo

## 4 Resultados

Após o desenvolvimento do robô foram necessários vários testes para refinamento das tarefas, nós criamos uma pista para treino utilizando cartolinhas e fita isolante. Isso possibilitou que as linhas pudessem ser trocadas com facilidade, o que nos permitiu ver o desempenho do robô em vários cenários. Uma das pistas que montamos pode ser vista na Figura 6, esta pista possui 94 cm de largura e 195 cm de altura.

// Melhorar.

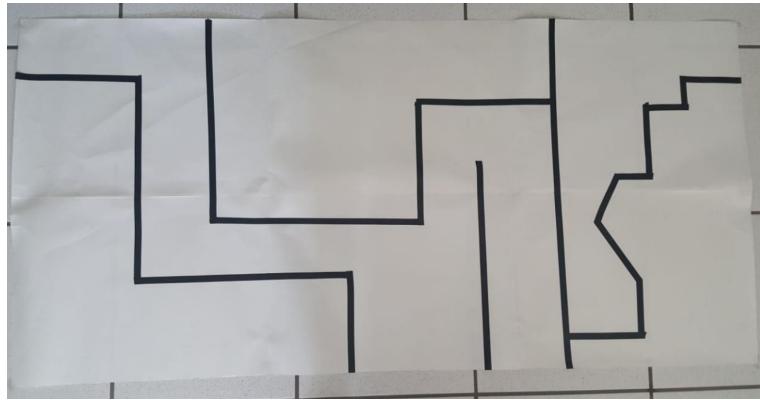


Figura 9: Pista para testes.

## 5 Conclusão

Com este trabalho conseguimos aplicar nossos conhecimentos adquiridos durante o semestre e pudemos adquirir conhecimentos adicionais com os materiais e métodos utilizados para o desenvolvimento do projeto. Sistemas de Tempo real geralmente são complexos devido às suas características de restrição temporal e sensibilidade à falhas, porém, devido ao grande planejamento que fizemos referente a implementação das tarefas e escalonamento, conseguimos desenvolver o robô de forma fluente, sem muitos contratemplos. O robô apresentou resultado satisfatório apresentando bom desempenho na execução das tarefas propostas.

// Melhorar.

## **Referências**

- [1] Farines, Jean-Marie, Joni da Silva Fraga, and RS de Oliveira. "Sistemas de tempo real." Escola de Computação 2000 (2000): 201.
- [2] Controle PID em sistemas embarcados. Carlos Márcio Freitas. Disponível em: [www.embarcados.com.br/controle-pid-em-sistemas-embarcados](http://www.embarcados.com.br/controle-pid-em-sistemas-embarcados). Acesso em: 23 ago. 2017.