Inteligência Artificial - GAC105

Relatório do Trabalho Prático

Repositório GitHub

Alunos:

Pedro Fonseca Rodrigues de Sousa

Lucas Malachias Furtado

Claudio Manoel Dos Reis Junior

Turma: 10A

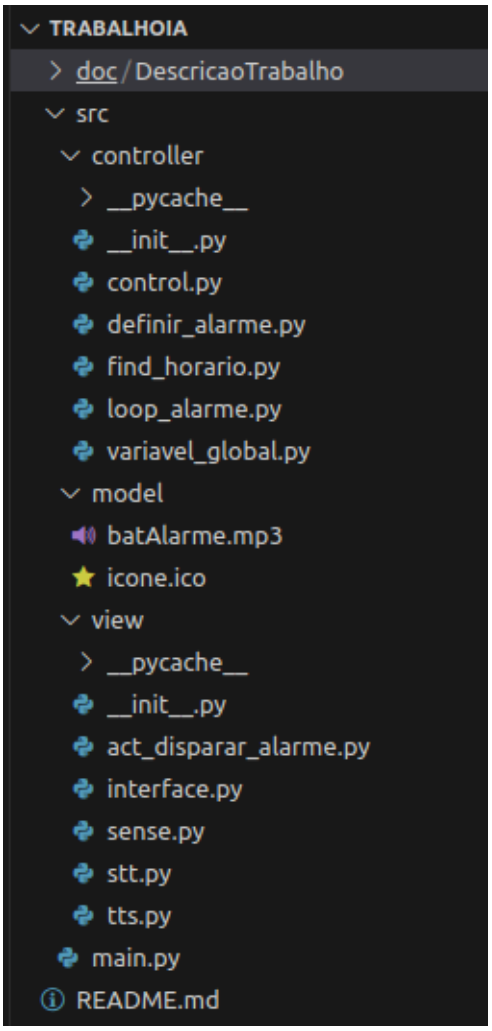
Sumário

Introdução

Este relatório descreve o desenvolvimento de um robô social autônomo e natural, que utiliza o modelo arquitetônico Sentir-Pensar-Agir para interagir com seres humanos. O projeto desafiou os alunos a aplicar conceitos de Inteligência Artificial em um contexto do mundo real, capacitando-os a desenvolver soluções que melhoram a interação entre humanos e robôs. Para isso, foi desenvolvido um “alarme inteligente” que captura informações pela voz dos usuários através da implementação de uma API de reconhecimento de fala e interage com os mesmos usuários por meio de uma tecnologia que converte texto em fala.

O projeto final, portanto, tem o objetivo de interagir com seres humanos de forma natural e eficaz, processar os dados obtidos e tomar decisões de forma autônoma com o objetivo de configurar um alarme de maneira inteligente. Neste relatório, serão apresentados os principais desafios enfrentados durante o desenvolvimento do robô social, as soluções encontradas, os resultados obtidos, bem como a arquitetura do projeto.

Arquitetura



A arquitetura do projeto foi baseada no modelo MVC (Model-View-Controller) e, portanto, foi dividido em três diretórios principais:

- Model: Como não foi necessário estabelecer a conexão com um banco de dados, nosso model contém apenas os arquivos da base de dados do projeto, que são utilizados em ações definidas em outros módulos no robô.

- Control: Neste módulo, é possível estabelecer um paralelo com o Pensar do modelo arquitetônico Sentir-Pensar-Agir, pois estão os arquivos que controlam a lógica implementada para que o robô funcione corretamente. Em “control.py” são ativadas as threads do programa referentes a interface e a interação do robô alarme. Dessa forma, essas duas ações operam paralelamente enquanto o programa está funcionando. Além disso, os arquivos “definir_alarme.py”, “find_horario.py”, “loop_alarme.py” e

“variavel_global.py” implementam funções importantes para o funcionamento do projeto e são divididos de maneira organizada nesse módulo.

- View: Neste módulo, é possível estabelecer um paralelo com o Sentir e Agir do modelo arquitetônico Sentir-Pensar-Agir, visto que aqui está definida as funções que recebem as informações e retornam ao usuário os dados processados durante o funcionamento do robô. Em “act_disparar_alarme.py”, há uma função que é acionada no horário do(s) alarme(s) configurado(s). Nos arquivos “stt.py” e “tts.py” é implementado a integração com as APIs speech-to-text e text-to-speech, respectivamente. Em “sense.py” está a função que define o comportamento principal do robô durante a interação com o usuário. Por fim, em “interface.py” existe uma função responsável por criar uma interface gráfica que será exibida enquanto o robô está ativo, adicionando à interface os horários dos alarmes que serão definidos pelo usuário.

Além dos arquivos organizados nesses diretórios, há um único arquivo “main.py” responsável por ativar o início do funcionamento do robô.

Desafios enfrentados

- Limite de tempo durante o reconhecimento de voz: Ao longo do desenvolvimento, houve uma certa dificuldade em definir o tempo correto que o reconhecedor de voz speech-to-text deveria “escutar” o usuário durante determinados momentos da interação. Em alguns momentos, o tempo não era suficiente para reconhecer o que o usuário desejava. Portanto, após vários testes, foi definido um limite adequado para reconhecer a voz do usuário de acordo com cada etapa da interação com o robô.

- Dificuldade de implementação da interface gráfica: A dificuldade de implementação da interface no projeto foi integrar os componentes visuais de com a lógica do programa, principalmente em relação a lista de horários. O programa deve atualizar constantemente a interface gráfica conforme um alarme é definido ou ativado. A solução encontrada foi inserir a execução da interface gráfica em uma nova thread.

- Organização dos arquivos e diretórios: Para que o projeto siga uma estrutura definida e melhor organizada, foi necessário dividir os arquivos em diretórios baseados na arquitetura MVC. Na tentativa de arranjar os diferentes módulos com seus respectivos arquivos houveram muitos problemas em importar determinados arquivos, uma vez que há funções do programa que dependem de outras funções desses arquivos para funcionar corretamente. Muitas vezes, componentes de alguns arquivos não eram corretamente atualizados em funções de outros arquivos ou mesmo era exibido mensagens de erro provavelmente causadas por “circular import.” Contudo, após estabelecer uma melhor organização desses arquivos em seus diretórios e isolando algumas funções, foi possível alcançar o resultado esperado na disposição dos módulos e componentes do projeto.

Resultados obtidos

Durante a execução deste projeto, alcançamos resultados significativos na criação de um robô social de gerenciamento de alarmes funcional e interativo. A adoção da arquitetura Model-View-Controller em conjunto com os conceitos aprendidos na disciplina de Inteligência Artificial referentes ao modelo Sense-Think-Act permitiu uma organização eficiente do código, facilitando a manutenção e a expansão do programa. A interface gráfica intuitiva oferece uma experiência de usuário agradável, facilitando a visualização dos alarmes adicionados. Além disso, a integração com as APIs speech-to-text e text-to-speech permitiram a criação de um programa que interage com o usuário de maneira compreensível e natural.

O objetivo inicialmente traçado de desenvolver uma aplicação que interage, reconhece e processa os dados de forma autônoma, retornando ao usuário as respostas de forma eficaz e correta foi atingido com sucesso. Em resumo, esse projeto demonstra a viabilidade de um robô social capaz de auxiliar os seres humanos no gerenciamento de alarmes por meio de uma solução implementada com uma interface gráfica e recursos de voz.