RELATÓRIO DE INICIAÇÃO CIENTÍFICA

PROJETO: PROPOSTA, ANÁLISE E IMPLEMENTAÇÃO DE ALGORITMOS DISTRIBUÍDOS PARA ENXAME DE ROBÔS.

ALUNO: Rafahel Costa dos Santos Matias

Março de 2024.

RESUMO

Neste trabalho, são apresentadas as atividades do projeto de iniciação científica efetuadas no período de 19/10/2023 a 12/03/2024, onde está sendo utilizado um robô e-puck da fabricante gctronic. De início foi efetuada a leitura do manual do usuário, onde constam as especificações técnicas do hardware do robô,de seus sensores e atuadores. Em seguida, foi efetuada a instalação dos softwares necessários para realizar a comunicação com o e-puck. Por fim, iniciaram-se os estudos dos programas de exemplos fornecidos pelo fabricante e logo em seguida a elaboração de algoritmos próprios.

Palavras-chave: *e-puck, robô, iniciação científica.*

SUMÁRIO

1. INTRODUÇÃO

O dispositivo e-puck é um robô criado pela empresa gctronic. Como o nome sugere é a quarta geração do robô da família e-puck. É amplamente utilizado em estudos de graduação e pós-graduação em robótica e eletrônica no desenvolvimento de tarefas em enxames de robôs.

O e-puck é um robô de pequeno porte. Possui câmera, oito sensores infravermelhos, 10 LEDs, três microfones, um alto falante, dois motores de passo e duas portas seriais em seu topo.

2. HARDWARE

2.1. Visão geral

O e-puck está equipado com 8 sensores infravermelhos, três microfones, um autofalante, dois motores de passos, LEDs RGB, um acelerómetro 3D, chave seletora e uma câmera com resolução de 680x480. A comunicação sem fio pode ser feita usando Bluetooth ou por cabo serial, e o processamento ocorre Linux embarcado. O controle do robô é feito por um microcontrolador dsPIC modelo 30F6014A.

e-puck - imagem site do fabricante

3. SOFTWARE

3.1. INSTALAÇÃO

De acordo com as especificações requeridas no manual do robô é necessário um computador com sistema operacional Windows ou Linux; caso a conexão seja feita por Bluetooth, é necessário que o computador tenha tal recurso. No site da gctronic e do e-puck é apresentada uma preferência pelo sistema Windows, diante dessa informação, foi utilizado a preferência do fornecedor para as análises e criações de códigos.

3.1.1. Compilador

O compilador indicado pelo manual é o "Compilador C30", porém, este, não está mais disponível. A versão utilizada atualmente é o "Compilador Xc16", disponível no site da Microchip como versão gratuita "Student Mode".

3.1.2. MPLAB

O software "MPLAB" deve ser instalado por meio do site da Microchip e o usuário deve acessar o MPLAB X IDE.

3.2. Conexão Bluetooth no e-puck e computador

A conexão via Bluetooth foi feita com o acesso às configurações do computador para a busca de um novo dispositivo Bluetooth e colocando a senha de acesso sendo o próprio número do robô, localizado com quatro números na parte frontal do robô.

3.3. Envio dos códigos executáveis para o robô

O envio funciona por meio do "Tiny bootloader", disponível no site do e-puck. Após a criação do arquivo executável no Tiny, é necessário buscar o arquivo, colocar a entrada "COM" correta que o robô está conectado no bluetooth, enviar e logo em seguida clicar no botão azul do robô que indica o reset para ter a conexão.

4. PROGRAMAÇÃO

4.1. Linguagens

A programação e estudo dos algoritmos devem ser feitas na IDE do MPLAB, outra IDE pode ser utilizada, porem a compilação deve ser feita no MPLAB. A programação deve ser feita em "C" ou em "Python", porém no site do fabricante é aconselhado usar a linguagem "C".

4.2. Compilação

A compilação deve ser feita pelo MPLAB clicando no botão direito do mouse no projeto e em "Build" ou "clean and build". É mais aconselhável usar o "clean and build", pois o arquivo executável gerado é com extensão "hex" e não o padrão de compilação em C. Devido essa diferença, caso a compilação não crie do arquivo, o usuário pode achar que arquivo foi gerado e usar um arquivo antigo no momento de enviar para o robô.

4.2.1. Preparatórios para a compilação

No momento da criação do projeto é necessário que coloque o compilador Xc16. Dentro do projeto deve-se seguir o caminho Properties e Building, para acionar o "Execute this line after build" e o "Normalize hex file", clicar em "ImagePath" e logo em seguida em "Apply". Ainda em Properties, é necessário que entre no XC16 e colocar o "Heap size" para 512.

Deve ser adicionado o arquivo p30F6014A.gld ao projeto, que deverá ser criado na primeira entrada no MPLAB. Além desse arquivo, devem ser adicionadas todas as bibliotecas que serão utilizadas no código com as extensões de arquivo C (.c) e de arquivo header (.h). Alguns exemplos das importações que devem ser feitas, retiradas do programa apresentado na seção 4.3 de algoritmos são: '***e_I2C_protocol', 'e_lsm330', 'e_epuck_ports', 'e_micro', 'e_acc', 'e_prox', 'e_ad_conv', 'e_agenda', 'e_motors' e 'e_led'.*** Especificamente o arquivo '***memory.h***' é importado do arquivo de teste disponível no site do e-puck. Dentro dos arquivos importados é preciso modificar as chamadas das bibliotecas, pois essas importações foram feitas originalmente em um diretório diferente do que o usuário está implementando.

4.3. ALGORITMOS

O código apresentado abaixo serve de base para as funções que serão inseridas posteriormente. O programa inicia com algumas chamadas de diretório para a importação de bibliotecas do robô, funções próprias de "C" e o microcontrolador *"p30F6014A.h".* A biblioteca correta é a que se encontra no diretório library dendro do projeto demonstrativo do e-puck.

O "*#define SELECTOR0 _RG6*" define as chaves seletoras que serão utilizados no robô; "*if (selector==0){*", tudo que se encontra dentro desse if será feito depois que o usuário colocar a chave do e-puck na posição 0. Os outros if$_s$ com esse mesmo molde é para o usuário colocar a chave na respetiva posição.

O "*char buffer[BUFFER_SIZE];*" é usado como um ponteiro para algumas funcionalidades do robô, guardando informações especificas.

Este é o código base utilizado para adicionar as funções onde está indicado:

```
#include "p30F6014A.h"

#define IR_RECIEVER

#include <stdio.h>
#include "string.h"
#include "math.h"
#include <time.h>

#include < library\motor_led\e_init_port.h>
```

```
#include < C:\library\motor_led/advance_one_timer/e_led.h>
#include < C:\ library\motor_led/advance_one_timer/e_motors.h>
#include <C: \library\motor_led/advance_one_timer/e_agenda.h>


#include <C:\ library\a_d/advance_ad_scan/e_ad_conv.h>
#include <C:\ library\a_d/advance_ad_scan/e_prox.h>
#include <C:\ library\a_d/advance_ad_scan/e_acc.h>
#include <C:\ library\a_d/advance_ad_scan/e_micro.h>
#include <C:\ library\motor_led/e_epuck_ports.h>
#include <C:\ library\acc_gyro/e_lsm330.h>
#include <C:\ library\I2C/e_I2C_protocol.h>

/* selector on normal extension*/

#define SELECTOR0 _RG6
#define SELECTOR1 _RG7
#define SELECTOR2 _RG8
#define SELECTOR3 _RG9

#include "C:\ MPLABXProjects\e-puck-library-master\program\DemoGCtronic-complete\memory.h"

char buffer[BUFFER_SIZE];

extern int selector;
char c;

#ifdef IR_RECIEVER

#define SPEED_IR 600
#endif


#define PI 3.141592653589793
#define ARRAY_WIDTH 640
#define ARRAY_HEIGHT 480


int main()
{

        e_init_port();
        e_start_agendas_processing();
       e_init_motors();




        // Para selecionar a chave seletora
        selector=getselector();

   int max_motor_speed = 1000;


        if (selector==0) { // ----------------------------------------------------------------------------------------------------------------

    //  Colocar uma função



        } else if (selector==1) { // ----------------------------------------------------------------------------------------------------------

    //  Colocar uma função



   } else if (selector==2){
```

```
    // Colocar uma função

  } else if (selector==3){ // ----------------------------------------------------------------------------------------------------------

// Colocar uma função

  } else if (selector == 4){ // ----------------------------------------------------------------------------------------------------------

// Colocar uma função

  } else if (selector == 5){ // ----------------------------------------------------------------------------------------------------------

// Colocar uma função

  } else if (selector == 6){

// Colocar uma função

  }


      while(1);
      return 0;
}
```

4.3.1.**Anda_robo.** Está função faz o robô andar para frente com velocidade “speed”.

```
int anda_robo(int speed) {

    e_set_speed_right(speed);
    e_set_speed_left(speed);

    return 0;}
```

4.3.2. vira: Essa função faz o robô andar para frente fazendo uma curva com paramentos “speed”, “prop_esq” e “prop__dir”. A prop é para saber o quendo que o usuário deseja que vire e esq e dir no final, determina se quer que vire para a esquerda ou direita, respectivamente**.**

```
int vira(float speed, float prop_esq, float prop_dir) {

    e_set_speed_right(speed * prop_esq);
    e_set_speed_left(speed * prop_dir);

    return 0;}
```

4.3.1.**Anda_para:** Essa função é utilizada para que o e-puck leia os sensores frontais para que o robô para caso tenha uma aproximação de distância pré-determinada pelo usuário. Os parâmetros são: dist (Distancia do objeto que o sensor lê), vel (Velocidade do robô) e temp (Atraso intencional ao final da execução).

```
int anda_para(int sensor, int dist, int vel, int temp){

    int valor_sensor;

        while (1) {

            valor_sensor = e_get_prox(sensor);

            // Se o valor lido pelo sensor for maior que 35, pare o robô
            if (valor_sensor > dist)
            {
                e_set_speed_left(0);
                e_set_speed_right(0);

            } else {
                // Caso contrário, continue movendo o robô para frente
(ou faça o que desejar)
                e_set_speed_left(vel);

                e_set_speed_right(vel);
            }


    wait(temp);
        }

}
```

4.3.2. Anda_sala: Esse código faz o robô andar pela sala desviando das paredes. Nesse programa é possível adicionar outras funcionalidades para o robô, usando a câmera e/ou sensor.

```
int anda_sala(int vel, int temp){

    int valor_sensor;

        while (1) {

            valor_sensor = e_get_prox(sensor);

            if (valor_sensor > dist)
            {
                e_set_speed_left(vel/2);
                e_set_speed_right(-vel/2);
                // aqui pode ser colocada outras funcionalidades para caso
  o robô tenha essa leitura do sensor

            } else {
                e_set_speed_left(vel);

                e_set_speed_right(vel);
            }
```

```
            // É possível também adicionar outros ifs aqui com condicionais
        diferentes utilizando a câmera e/ou os sensores


         wait(temp);
              }

        }
```

4.3.3. **LEDs:** A funções do LED se encontram na biblioteca do e-puck. Alguns exemplos são:

e_set_led (x,y), x é o número do LED no robô e y é o nível logico.
e_set_front_led (y), acende um LED frontal com luz mais forte
e_blink_led (), acende todos os LEDs

4.3.4. **Visão geral:** Os algoritmos foram criados de acordo com o estudo das bibliotecas do robô e do código de exemplo, disponibilizado no site da gctronic.

- Codigo base;
- Anda_robo;
- vira;
- anda_para;
- anda_sala;
- LEDs.

5. CONCLUSÕES

As funções desenvolvidas permitem o robô navegar pela sala de forma autônoma, evitando obstáculos e acionando os LEDs em momentos oportunos. Esse conjunto de instruções proporciona ao robô um comportamento inteligente, atendendo aos requisitos estabelecidos e demonstrando a eficácia do código na execução das tarefas propostas.

6. REFERÊNCIAS

1. E-PUCK. Disponível em: https://e-puck.gctronic.com/index.php
2. E-GCTRONIC. Disponível em: https://www.gctronic.com/
3. MICROCHIP. Disponível em: https://www.microchip.com/
4. MANUAL E-PUCK. Disponível em: https://www.gctronic.com/files/miniDocWeb.pdf
5. LINKS E-PUCK. Disponível em: https://www.gctronic.com/e-puck_links.php
6. REPOSITORIO GITHUB. Disponível em: https://github.com/gctronic
7. STACK OVERFLOW. Disponível em: https://stackoverflow.com/
8. FÓRUM. Disponível em https://forum.coppeliarobotics.com/
9. CYBERBOTICS. Disponível em: https://cyberbotics.com/