ESTABELECIMENTO DE ROTAS PARA AR.DRONE UTILIZANDO DELPHI XE 10

Aluno(a): Rafael Ronaldo Rahn

Orientador: Mauro Marcelo Mattos

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação Teórica
- Trabalhos Correlatos
- Desenvolvimento
- Operacionalidade da Implementação
- Resultados e Discussões
- Conclusões e Sugestões
- Demostração

Introdução

- Drones
  - Denomicação
- Aplicações
  - Uso militar
  - Logística
  - Fotografia
  - Agricultura
- Motivação - Automatização

Objetivos

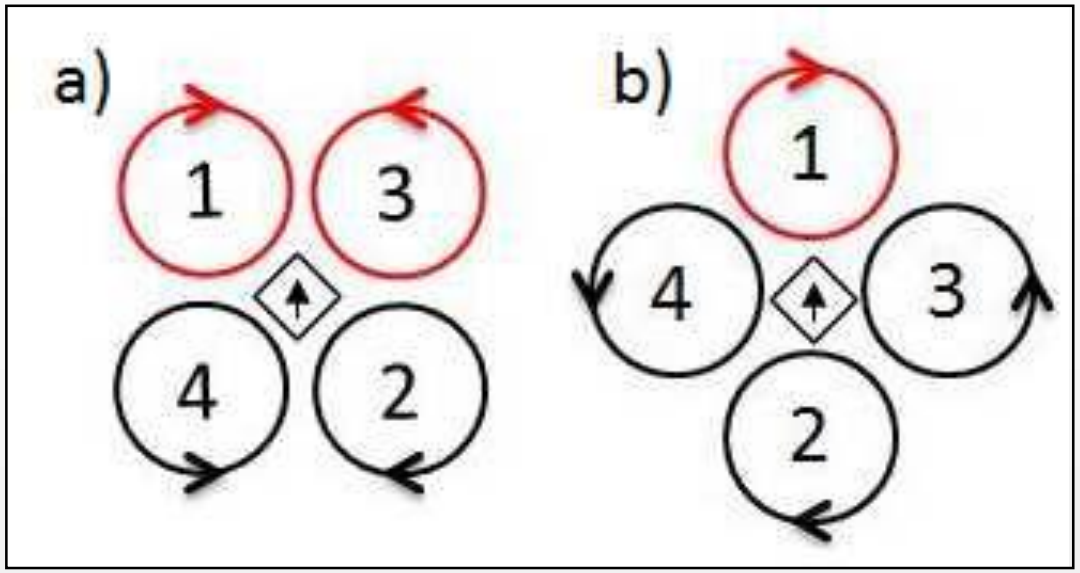
- Desenvolver um aplicativo de suporte ao estabelecimento de rotas
- Desenvolver um conjunto de cenários de teste para validar o aplicativo

Fundamentação Teórica

- AR.Drone 2.0

Fundamentação Teórica

– **DINÂMICA DE FUNCIONAMENTO DE UM QUADRICÓPTERO**



Fundamentação Teórica

- Os recursos disponíveis no AR.Drone 2.0
- Câmeras horizontal e vertical
- Acelerômetro, Giroscópios, Sensor de pressão, Ultrassom
- Magnetômetro, Kernel Linux 2.6.32;
- Processador ARM Cortex A8 32-bit de 1GHz, memória RAM 1GB e *Wifi* nas bandas b/g/n.

Fundamentação Teórica

- GPS
- 1m Precisão

Fundamentação Teórica

- porta 5556- utilizada para envio de comandos para o drone;
- porta 5554- utilizada pelo drone para envio dos dados de navegação;
- porta 5555- utilizada pelo drone para envio de streaming de vídeo;
- porta 5559- porta TCP utilizada para envio de comandos críticos que não podem ser perdidos

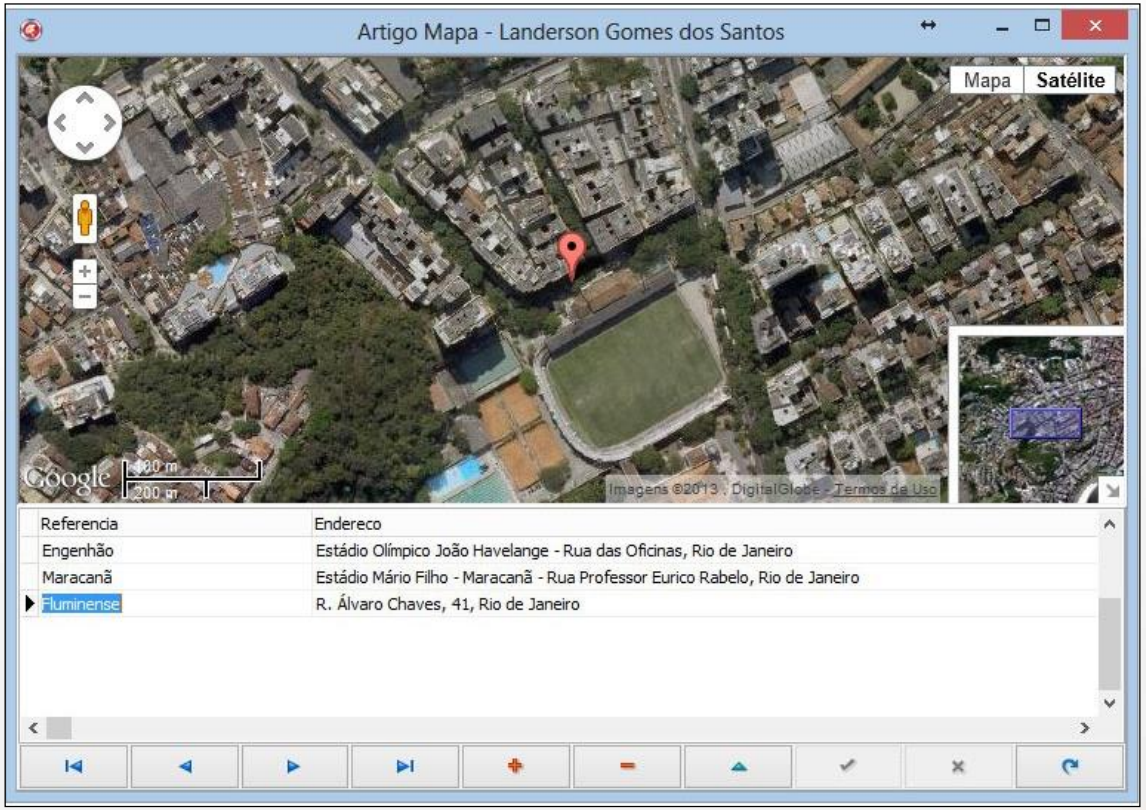
Fundamentação Teórica

- **COMPONENTE TARDRONE PARA DELPHI**
  - Comandos
    - comunicação
    - movimentação
    - configuração

```
begin
  ARDrone.MoveForward(1);

  ARDrone.MoveBackward(1);

  ARDrone.MoveRight(1);

  ARDrone.MoveLeft(1);

  ARDrone.RotateCW(1);

  ARDrone.RotateCCW(1);

end;
```

Fundamentação Teórica

- **COMPONENTE GOOGLEMAPS PARA DELPHI**
  - Mapa
  - Marcadores
  - Coordenadas
  - Linhas
  - Circulos



Fundamentação Teórica

- Equação de Haversine
  - utilizada para realizar o cálculo da distância entre dois pontos.

$$d = 2R\ arcsen\sqrt{sen^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)sen^2\left(\frac{\psi_2 - \psi_1}{2}\right)}$$

Trabalhos Correlatos

- *Robot Operating System* (ROS)
  - Desenvolvido em 2007 pelo laboratório de inteligência artificial de Stanford
  - Fornece abstração de *hardware, device* e *drivers,* bibliotecas
  - Fornece suporte AMIGO, AscTec Quadrotor, Intel Edison, Lego NXT, Softbank Pepper e inúmeros outros

Trabalhos Correlatos

- Portal (2011), desenvolveu um piloto automático para o AR Drone 2.0 o qual tem o objetivo de fazer o *drone* cumprir um itinerário pré-definido por marcadores posicionados no chão.

Trabalhos Correlatos

- Já Afonso (2014) Java Arduino e Node.js

Botões	Funções	Botões	Funções
Power	TakeOff/Landing	Mode	-
Mute	Desconectar/Sair	Play/Pause	Subir/Descer
Voltar	Mover para Trás	Avançar	Mover para a Frente
Eq	Executar missão	Menos(-)	Mover Yaw para Esquerda
Mais (+)	Mover Yaw para Direita	0	Teste de comunicação
Loop	Decrementa Angle em 15	U/SD	Incrementa Angle em 15
1	Executa Missão Quadrática	2	Executa Missão Linear
3	Executa Missão Triangular	4	Executa Missão Circular
5	Mover para a esquerda	6	Move para a direita
7	Decrementa Value em 1	8	Incrementa Value em 1
9	Posiciona o drone no ponto de origem	-	-

Requisitos

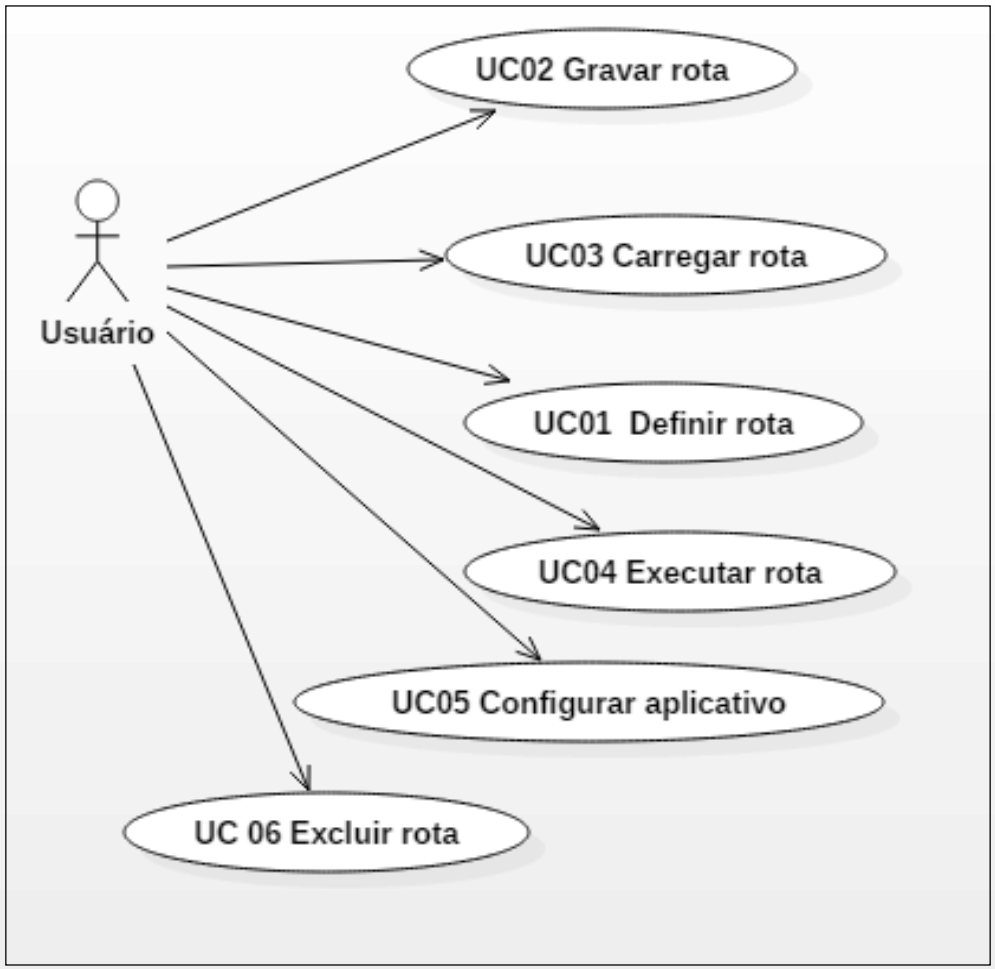
- permitir criar um mapa de navegação (RF);
- permitir estabelecimento de ações em posições específicas do mapa (RF);
- realizar *upload* das tarefas para o *drone* (RF);
- disparar o voo autônomo a partir da rota pré-programada (RF);
- permitir a importação de um mapa 2D com marcadores para ferramenta (RF);

Requisitos

- permitir que após o mapa importado seja possível desenhar uma rota para o *drone* percorrer (RF);
- permitir salvar as rotas do mapa (RF)
- calibrar a rota para corrigir eventuais desvios durante a navegação do *drone* (RF)
- utilizar a IDE Delphi XE 10 (RNF);
- utilizar o Ar.Drone 2.0 (RNF).

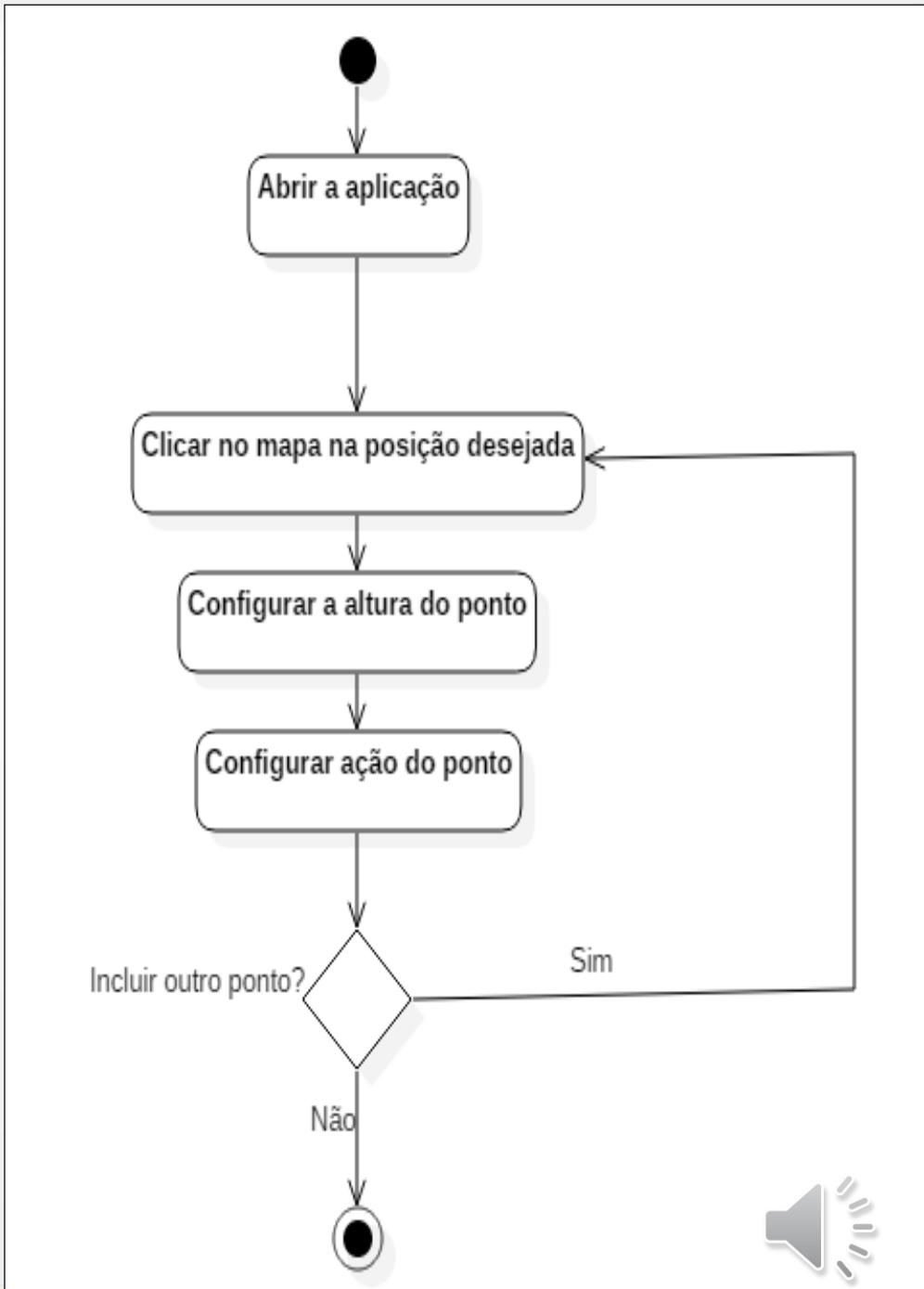
Especificação

- Casos de uso da aplicação



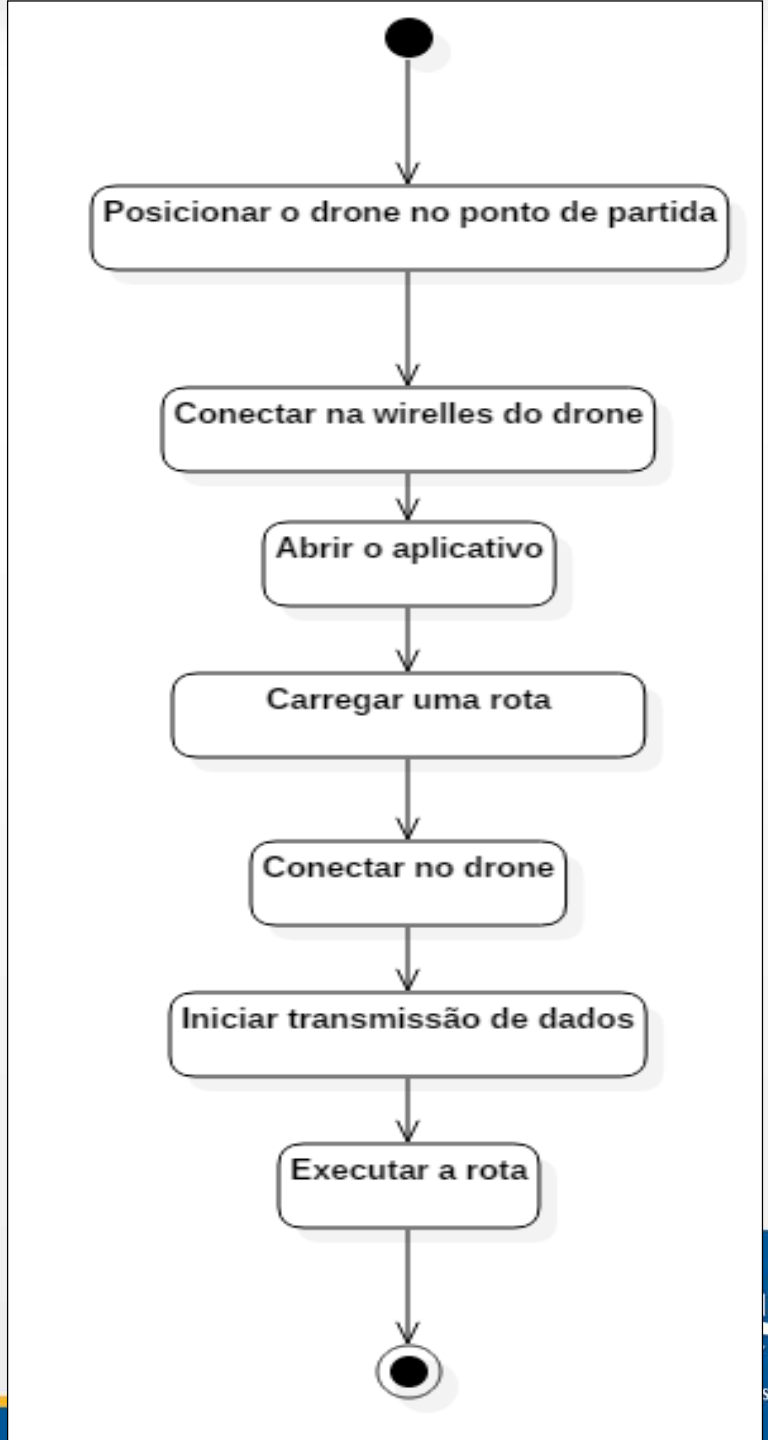
Especificação

- Fluxo da criação de uma rota



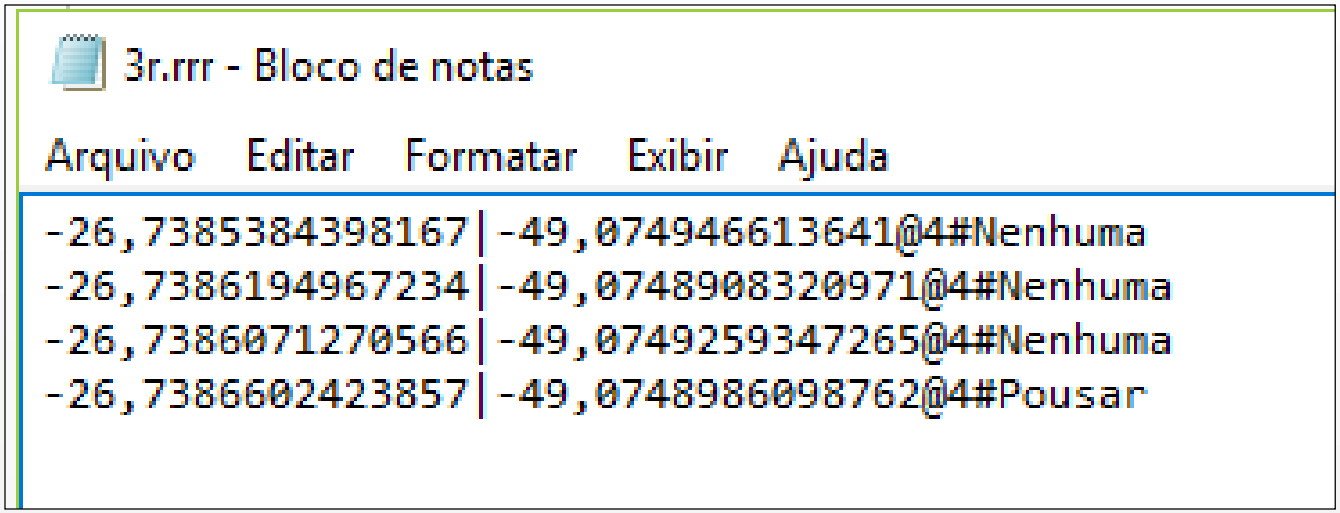
Especificação

- Macro fluxo para execução de uma rota

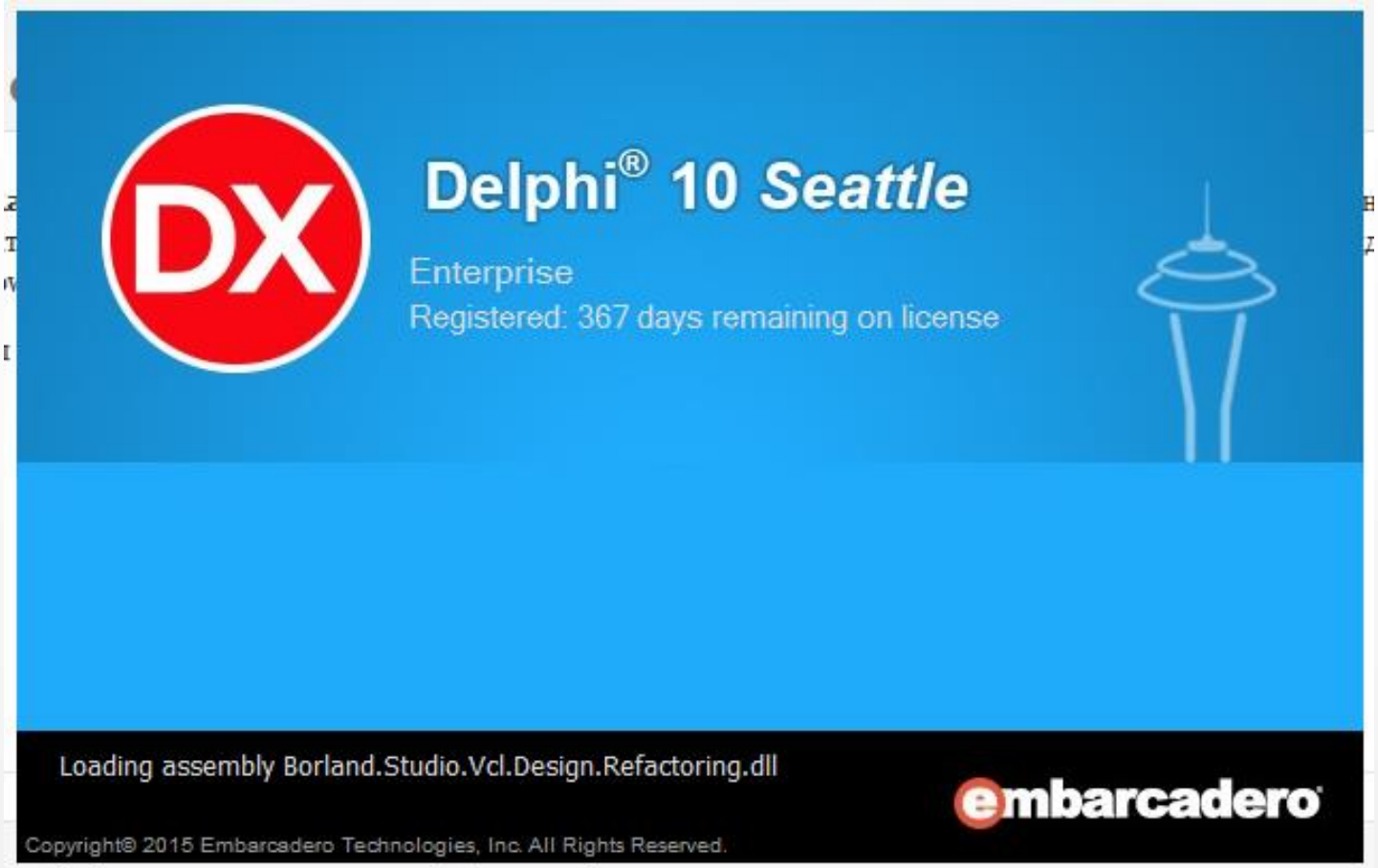


Especificação

- Arquivo de rotas



Implementação



Implementação

- Inicio da transmissão de dados

```
SetLength(Buf, 5);
Buf[0] := 1;
ARDrone.NavUDP.SendBuffer(buf);
ARDrone.Config('general:navdata_demo','TRUE'); //TRUE
ARDrone.Config('general:navdata_options','22'); //22
ARDrone.Config('general:navdata_options','777060865'); //22
Sleep(100);
ARDrone.SendCommand('AT*CTRL','0');
Sleep(100);
FEnviaNovaSolicitacao := True;
//incia a thread
IdThreadDados.Start;

IdUDPServer1.Active := True;
IdUDPServer1.BufferSize := 0;
//IdThreadQuebraRetorno.Start;
TimerNavDat.Enabled := True;
```

Implementação

Rotina de leitura dos dados do drone

```
Procedure TForm2.IdThreadDadosRun(Sender: TIdThreadComponent);
var
  buf: TIdBytes;
begin
  if FEnviaNovaSolicitacao then
  begin
    ARDrone.NavUDP.Send('1');
    SetLength(Buf, 2048);
    ARDrone.navUdp.ReceiveBuffer(buf);
    bufferDados.Add(buf);
  end; end;
```

Implementação

- Mapeamento da estrutura principal

```
NAVDATA_DEMO = record
    tag : Word;
    size : Word;
    ctrl_state: cardinal;
    vbat_flying_percentage: cardinal;
    theta: Single;
    phi: Single;
    psi: Single;
    altitude: Integer;
    vx: Integer;
    vy: Integer;
    vz: Integer;
    num_frames: cardinal;                  // Don't use
    detection_camera_rot: vector31_t;       // Don't use
    detection_camera_trans: vector31_t;     // Don't use
    detection_tag_index: cardinal;         // Don't use
    detection_camera_type: cardinal;       // Don't use
    drone_camera_rot: vector31_t;           // Don't use
    drone_camera_trans: vector31_t;         // Don't use
end;
```

Implementação

Rotina de interpretação dos dados recebidos

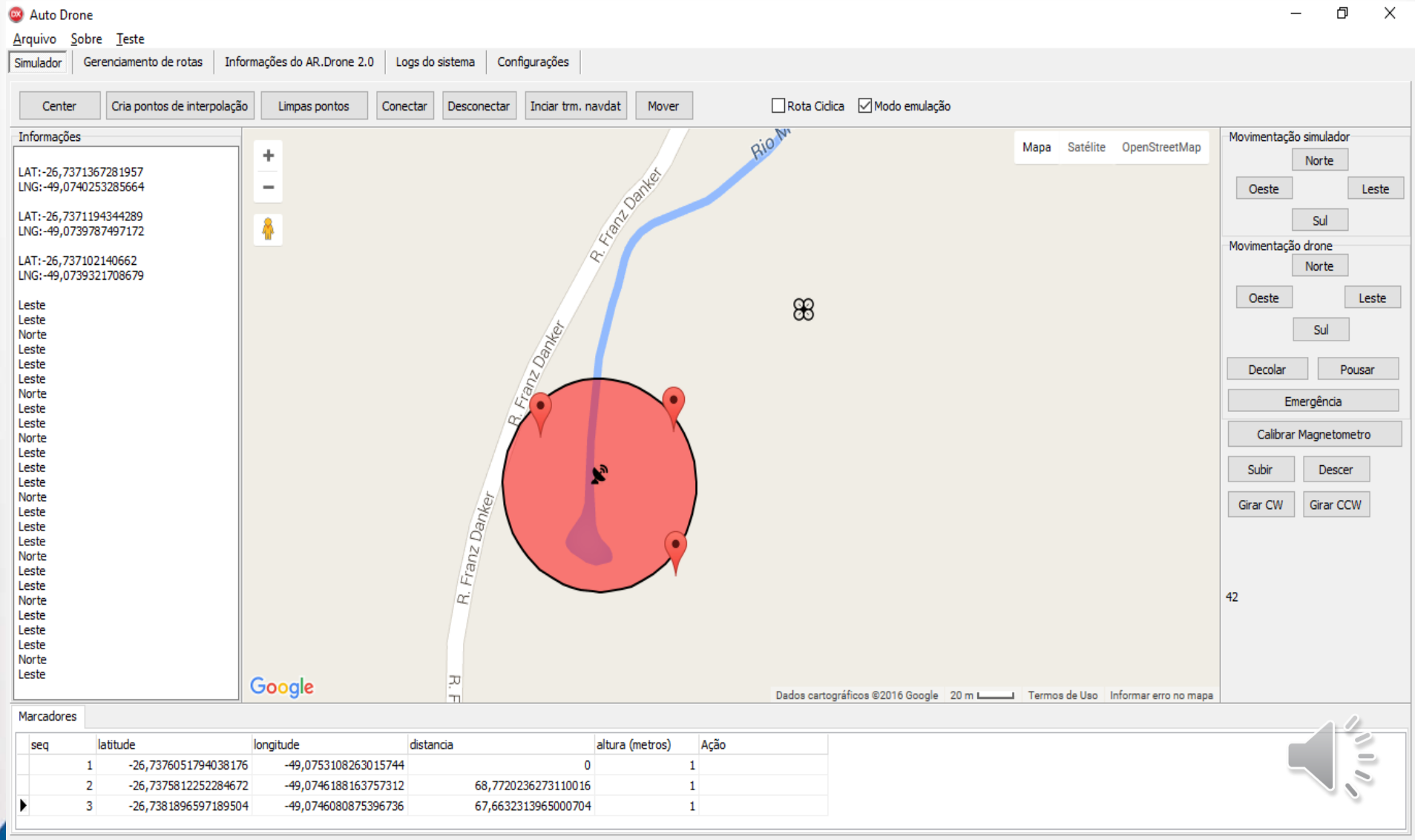
```
Var
  ardrone_navdata_demo: navdata_demo;
begin
  index := 16;
  while (index < TAMANHO ) do
  begin
...
    case tmp_tag of
        ARDRONE_NAVDATA_DEMO_TAG :
        begin
          CopyBuf(buf, dados_mov, index);
          move(dados_mov, ardrone_navdata_demo,
                    MIN(tmp_size, sizeof(NAVDATA_DEMO)));
          if loop then
            Break;
        end;
```

Implementação

- Equação de Haversine

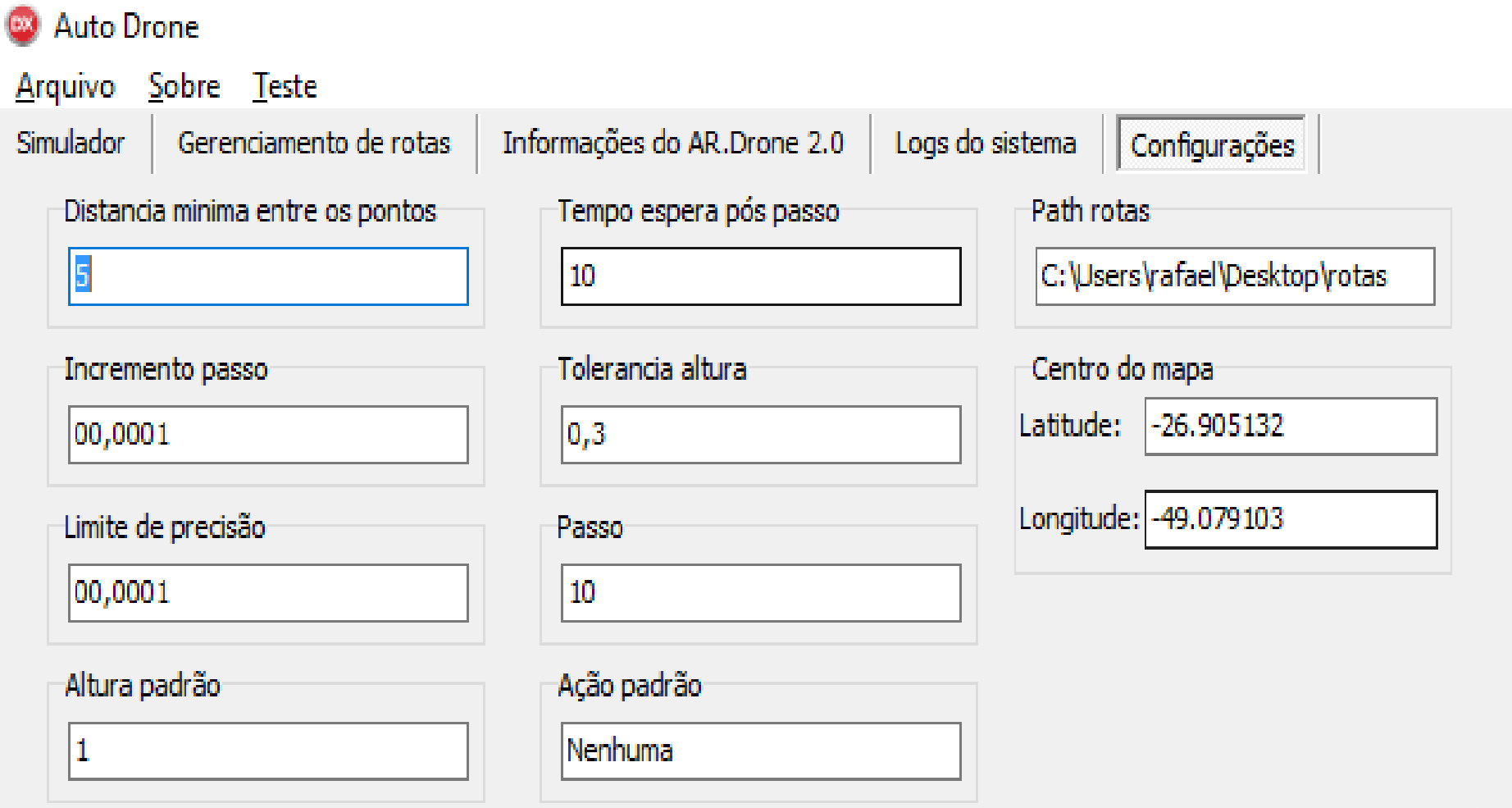
```
function TForm2.CalculaDistancia(Lat1, Lat2, Lng1, Lng2: Real): Real;
begin
  Result := 6371000*ARCCOS(COS(PI()*(90-Lat2)/180)*COS((90-Lat1)*PI()/180)+
  SIN((90-Lat2)*PI()/180)*SIN((90-Lat1)*PI()/180)*COS((Lng1-Lng2)*PI()/180));
end;
```

Operacionalidade da Implementação



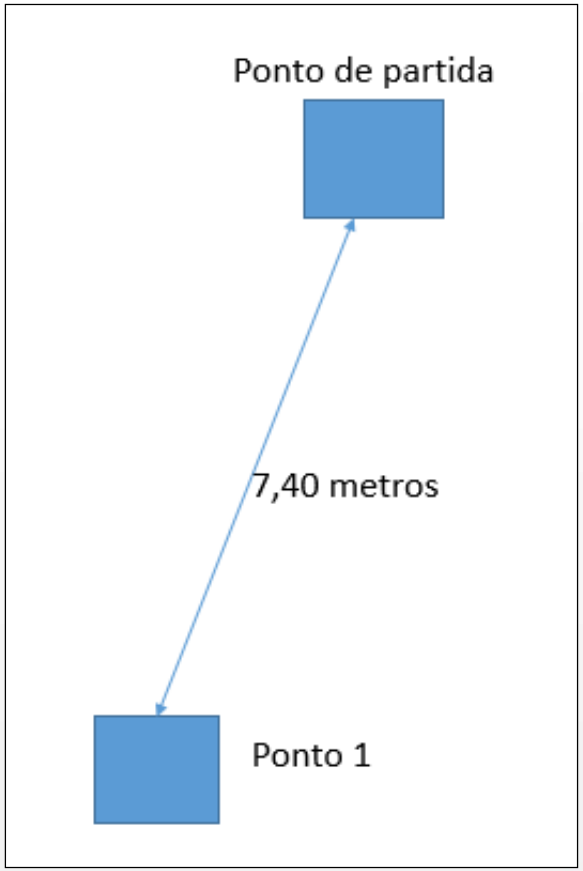
seq	latitude	longitude	distancia	altura (metros)	Ação
1	-26,7376051794038176	-49,0753108263015744	0	1	
2	-26,7375812252284672	-49,0746188163757312	68,7720236273110016	1	
3	-26,7381896597189504	-49,0746080875396736	67,6632313965000704	1	

Operacionalidade da Implementação



Resultados e Discussões

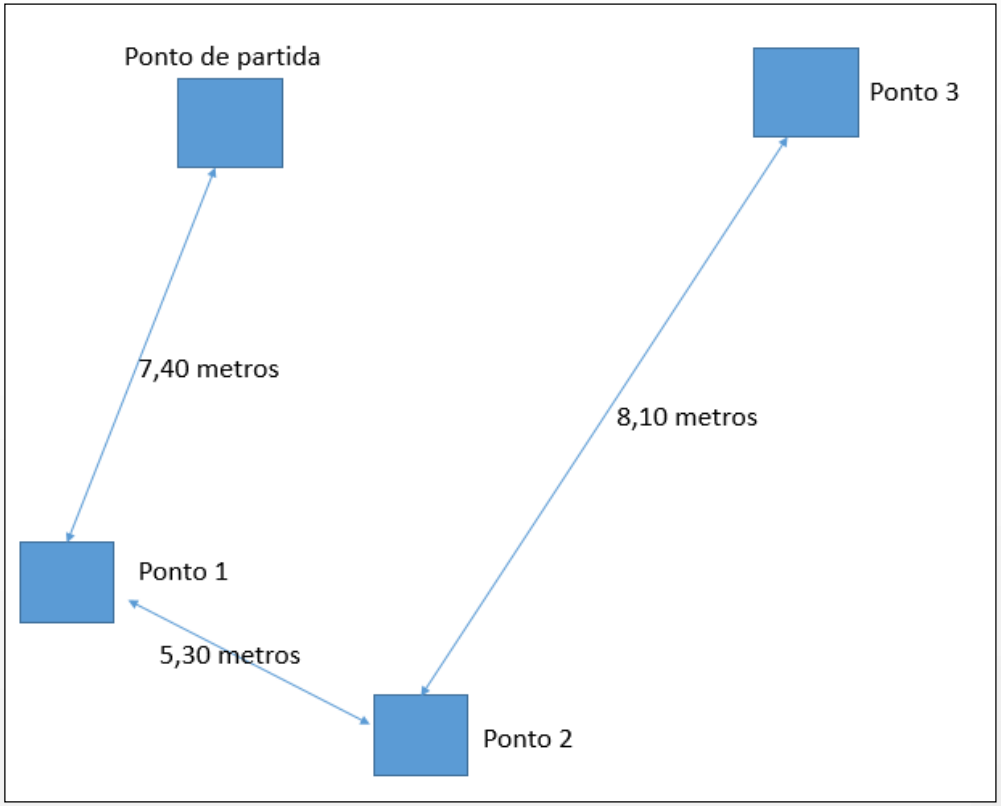
- Média de 1,51 metros em 10 execuções da rota



Resultados e Discussões

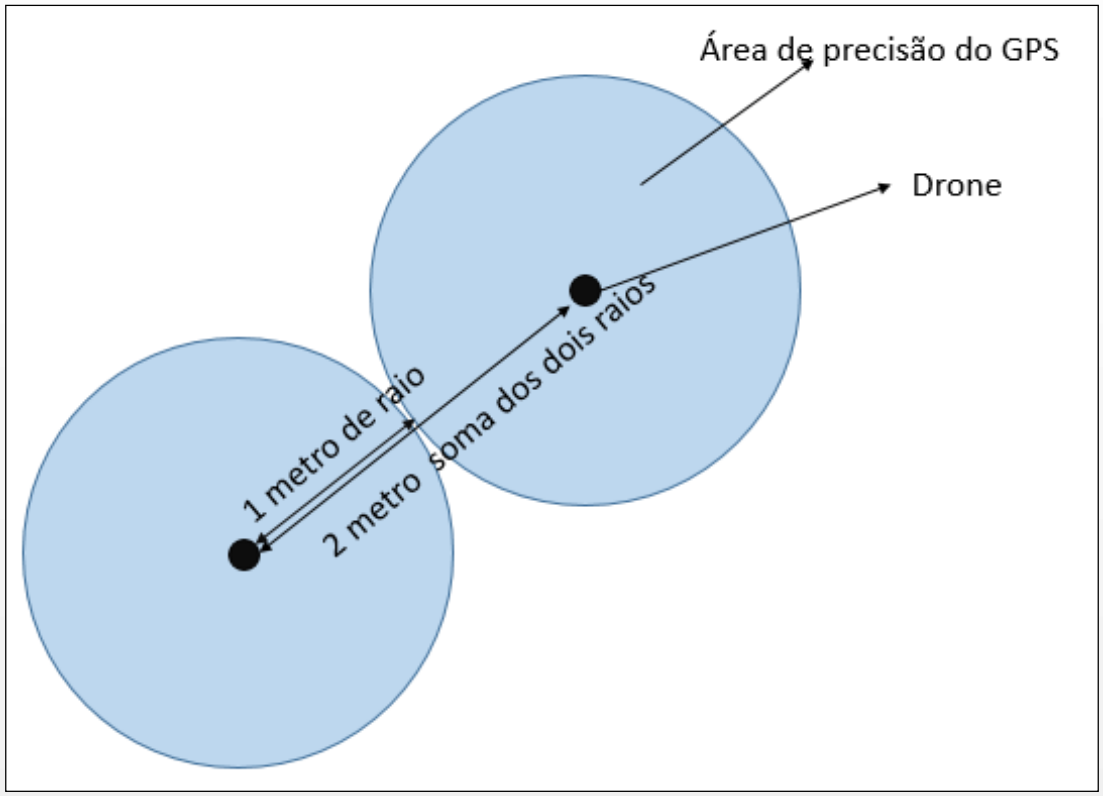
- Média 1,63 m em 10 execuções da rota

- Média 1,65 m em 10 execuções da rota, com alteração de altura



Resultados e Discussões

- Área de precisão



Conclusões e Sugestões

Caracteristica	Portal	ROS	Afonso	Trabalho presente
Utiliza mapas		x		X
Realiza caminhamento	x			x
Utiliza controle remoto		x	x	
Realiza ações		x		x
Permite exportar rotas				x
Utiliza GPS		x		x

Conclusões e Sugestões

- Possíveis Extensões
  - implementar mais ações para o *drone* executar (tirar uma foto, abrir comunicação de video, girar, realizar uma animação, retornar);
  - pesquisar e desenvolver um cálculo melhor para otimizar a precisão, uma alternativa é também fazer o uso dos acelerômetros;

Conclusões e Sugestões

- implementar recursos para utilização das câmeras;
- automatizar para que o *drone* se alinhe com o norte sem a necessidade do usuário fazer isto;
- extender o componente TAr.Drone implementando também as funcionalidades para recebimento das informações do *drone;*
- criação de uma versão para tablets