Modelagem do Veículo Aéreo Não Tripulado QuadRotor Gyrofly 200ED

Alunos: *Daniel Sperry*
Daniela Castellain
Daniela Uez
Renê Oliveira

Disciplina: Projeto e desenvolvimento de sistemas embarcados

Professores: Jean-Marie Farines e Cristian Koliver

Dezembro/2011

Sumário

- Introdução
- Descrição do QuadRotor Gyrofly 200ED
- Etapas do projeto de SE
- Definição de Requisitos
- Modelagem Funcional
- Modelagem Arquitetural
- Desafios encontrados
- Visão crítica
- Conclusão
- Bibliografia

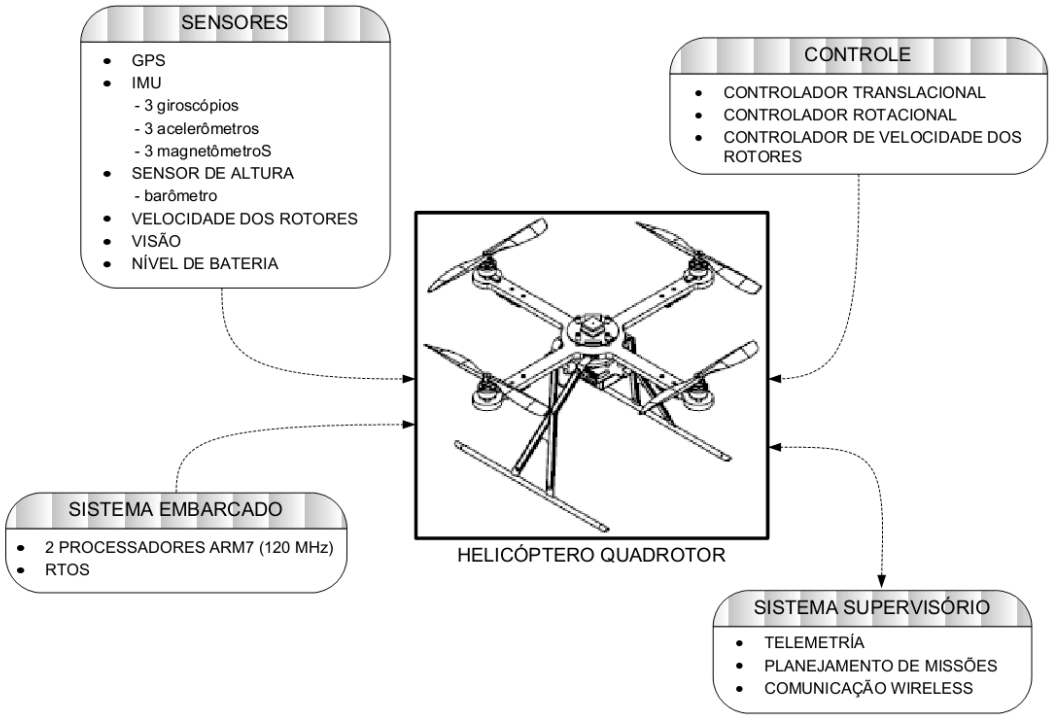
Introdução (1/2)

- O trabalho descreve a modelagem do veículo não tripulado QuadRotor Gyrofly 200ED, tanto do ponto de vista de estrutura como de comportamento.
- Utilizou-se o artigo *Navegation and Guindance of Unmanned Aerial Vehicles An Application to the Gyrofly 200ED QuadRotor*, tendo como autor Guilherme Raffo.
- *Unmanned Aerial Vehicles (UAV)*, são equipamentos conduzidos sem a presença física de tripulantes, e projetados para ambientes de monitoramento, proteção, dentre outros. Ex: apoio militar em atividades de busca e resgate.

Introdução (2/2)

- A principal justificativa para o uso de UAVs está na substituição da presença física em ambientes onde existem riscos à vida.
- A modelagem funcional e a modelagem arquitetural do sistema foram baseadas nas abordagens MDE (engenharia baseado em modelos).
- MDE é uma metodologia de desenvolvimento que foca na criação de modelos, visando elevar o nível de abstração.

Descrição do QuadRotor Gyrofly 200ED



- Quadrotor têm a capacidade de definir sua localização (posição e orientação) em um ambiente desconhecido.
- A localização é estimada através de sensores e pela fusão de todas as informações obtidas através destes dispositivos.
- A velocidade é definida por malhas de controle de translação e rotação.

Etapas do projeto de SE

- Definição de Requisitos
- Modelagem Funcional
- Modelagem Arquitetural
- Mapeamento de SW/HW
- Geração de Código

Definição de Requisitos (1/2)

- Requisitos Funcionais: descrevem o comportamento do sistema, todas as coisas que o sistema deve fazer.

Ex: Quando a aeronave é ativada, todos os sensores devem ser inicializados e verificados.

- Requisitos Não-Funcionais: são restrições que se coloca sobre como o sistema deve realizar seus requisitos funcionais (performance, desempenho do sistema).

Ex: O sistema deve monitorar as coordenadas XYZ através dos sensores GPS, barômetro, e a fusão de dados por todos os outros sensores.

Definição de Requisitos (2/2)

- ***StartStop*:** O quadrotor deve ser ativado pelo usuário para entrar no modo de operação. Quando desligado, o helicóptero deve desligar os rotores.
- ***Communication with groud station*:** Após o quadrotor ser ativado e consequentemente os seus sensores, a comunicação e a telemetria devem ser checados e uma confirmação deve ser devolvida ao sistema.
- ***Localization*:** Quando o quadrotor for ativado, todos os sensores devem ser reinicializados e verificados.
- ***Take-off and Landing:*** O quadrotor pode levantar vôo após os rotores e os controladores de rotação e translação serem inicializados.
- ***Autonomous path tracking:*** Após o quadrotor levantar vôo e obter uma posição inicial, uma trajetória pré-definida é seguida pelo helicóptero.
- ***Control loops:*** Quando o UAV, a localização e as estações estiverem ativadas, todos os controladores devem ser inicializados.

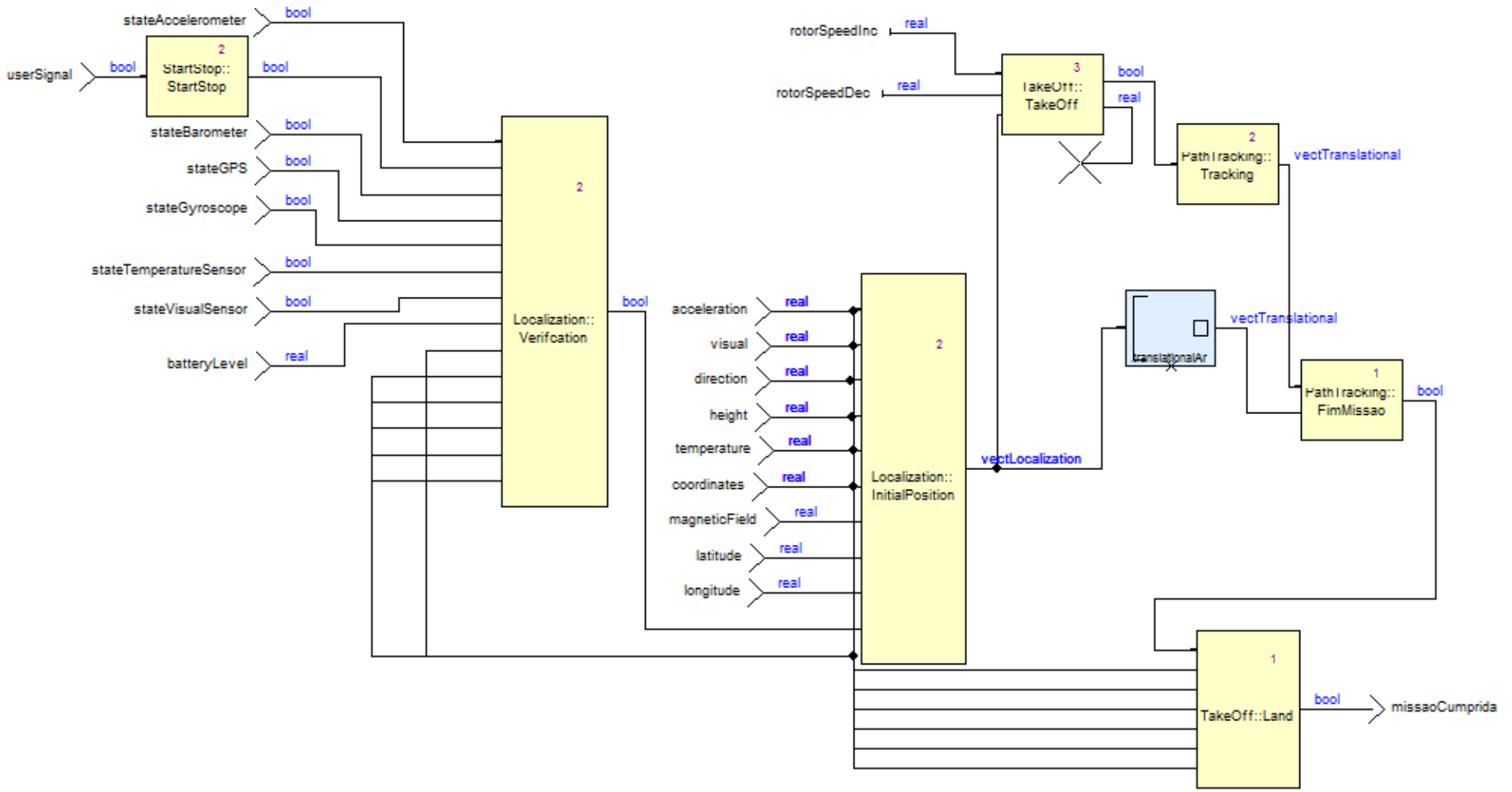
Etapas do projeto de SE

- Definição de Requisitos
- Modelagem Funcional
- Modelagem Arquitetural
- Mapeamento de SW/HW
- Geração de Código

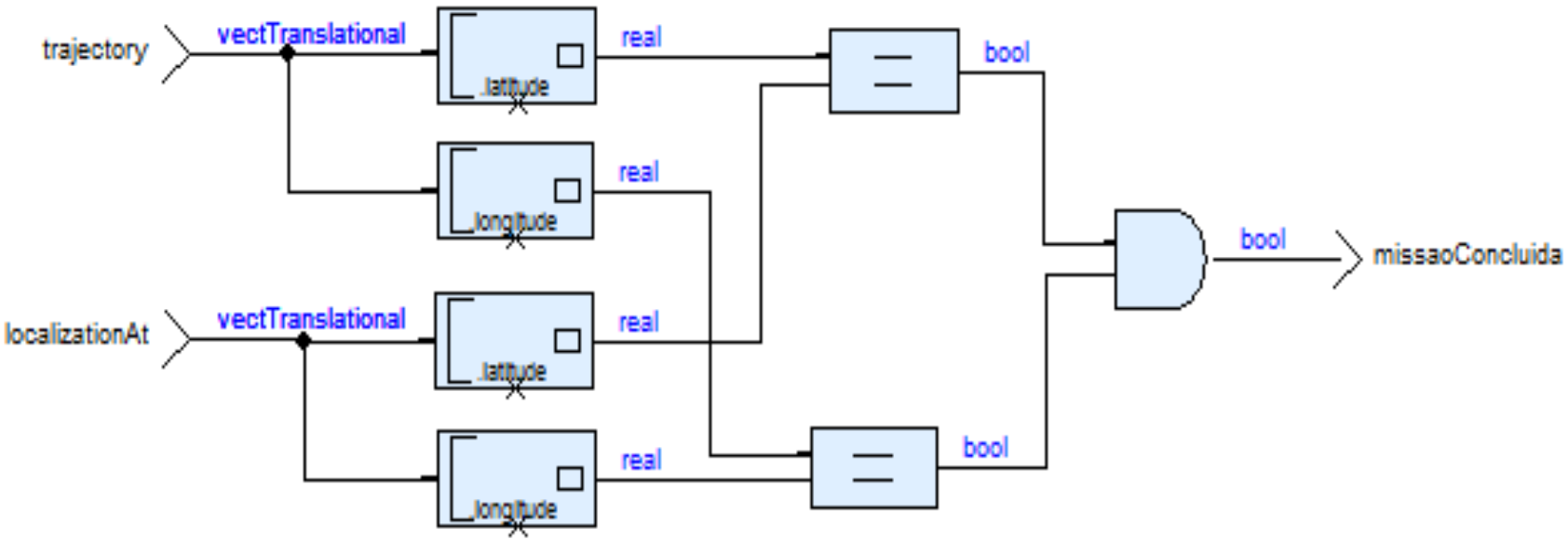
Modelagem Funcional (1/6)

- Objetivo de representar e descrever as funcionalidades do sistema de maneira mais abstrata.
- Deve ser independente de plataforma.
- A modelagem se deu pela utilização do software SCADE (pertence a Esterel Technologies), que utiliza linguagem LUSTRE.
- LUSTRE é uma linguagem síncrona textual e declarativa, do tipo *data-flow*.

Visão geral do sistema (2/6)



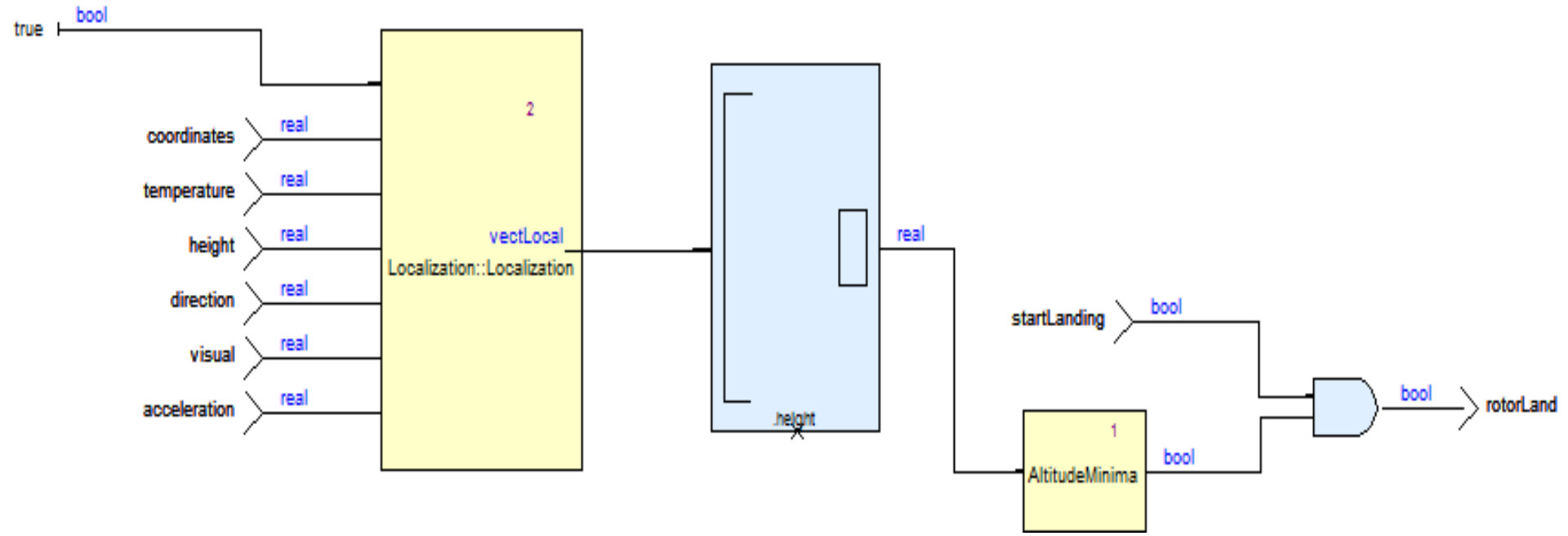
FimMissão (3/6)



FimMissão (4/6)

```
/* $*************** KCG Version 6.1.2 (build i5) ***************
** Command: s2c612 -config E:/Aulas/Embarcados/trabalho/quadRotor/Simulation\kcg_s2c_config.txt
** Generation date: 2011-12-05T17:30:03
*****************************************************************$ */

#include "kcg_consts.h"
#include "kcg_sensors.h"
#include "FimMissao_PathTracking.h"

void FimMissao_reset_PathTracking(outC_FimMissao_PathTracking *outC)
{
}

/* PathTracking::FimMissao */
void FimMissao_PathTracking(
  /* PathTracking::FimMissao::trajectory */vectTranslational *trajectory,
  /* PathTracking::FimMissao::localizationAt */vectTranslational *localizationAt,
  outC_FimMissao_PathTracking *outC)
{
  kcg_copy_vectTranslational(&outC->_L1, trajectory);
  outC->_L4 = outC->_L1.latitude;
  kcg_copy_vectTranslational(&outC->_L2, localizationAt);
  outC->_L7 = outC->_L2.latitude;
  outC->_L3 = outC->_L4 == outC->_L7;
  outC->_L6 = outC->_L1.longitude;
  outC->_L8 = outC->_L2.longitude;
  outC->_L10 = outC->_L6 == outC->_L8;
  outC->_L5 = outC->_L3 & outC->_L10;
  outC->missaoConcluida = outC->_L5;
}

/* $*************** KCG Version 6.1.2 (build i5) ***************
** FimMissao_PathTracking.c
** Generation date: 2011-12-05T17:30:03
*****************************************************************$ */
```

Land (5/6)


true
bool
coordinates
real
temperature
real
height
real
direction
real
visual
real
acceleration
real
2
vectLocal
Localization::Localization
real
height
startLanding
bool
1
AltitudeMinima
bool
bool
rotorLand

Land (6/6)

```
/* $*************** KCG Version 6.1.2 (build i5) ****************
** Command: s2c612 -config E:/Aulas/Embarcados/trabalho/quadRctor/Simulation\kcg_s2c_config.txt
** Generation date: 2011-12-05T17:30:03
*****************************************************************$ */

#include "kcg_consts.h"
#include "kcg_sensors.h"
#include "Land_TakeOff.h"

void Land_reset_TakeOff(outC_Land_TakeOff *outC)
{
  /* 1 */ AltitudeMinima_reset_TakeOff(&outC->Context_1);
  /* 2 */ Localization_reset_Localization(&outC->Context_2);
}

/* TakeOff::Land */
void Land_TakeOff(
  /* TakeOff::Land::startLanding */kcg_bool startLanding,
  /* TakeOff::Land::acceleration */kcg_real acceleration,
  /* TakeOff::Land::visual */kcg_real visual,
  /* TakeOff::Land::direction */kcg_real direction,
  /* TakeOff::Land::height */kcg_real height,
  /* TakeOff::Land::temperature */kcg_real temperature,
  /* TakeOff::Land::coordinates */kcg_real coordinates,
  outC_Land_TakeOff *outC)
{
  outC->_L4 = startLanding;
  outC->_L23 = kcg_true;
  outC->_L25 = coordinates;
  outC->_L24 = temperature;
  outC->_L28 = height;
  outC->_L26 = direction;
  outC->_L27 = visual;
  outC->_L29 = acceleration;
  /* 2 */
  Localization_Localization(
    outC->_L23,
    outC->_L25,
    outC->_L24,
    outC->_L28,
    outC->_L26,
    outC->_L27,
```

Etapas do projeto de SE

- Definição de Requisitos
- Modelagem Funcional
- Modelagem Arquitetural
- Mapeamento de SW/HW
- Geração de Código

Modelagem Arquitetural (1/12)

- Utilizada principalmente para identificar como será realizada a implementação do que foi definido na modelagem funcional.
- Utilizou-se a linguagem AADL (*Architecture Analysis and Design Language*), uma linguagem de descrição de arquitetura.
- Linguagem textual e gráfica usada no domínio de sistemas embarcados e tempo real.
- Utilizou-se o ambiente TOPCASED, que é baseada na plataforma Indigo (Eclipse 3.7).

Modelagem Arquitetural (2/12)

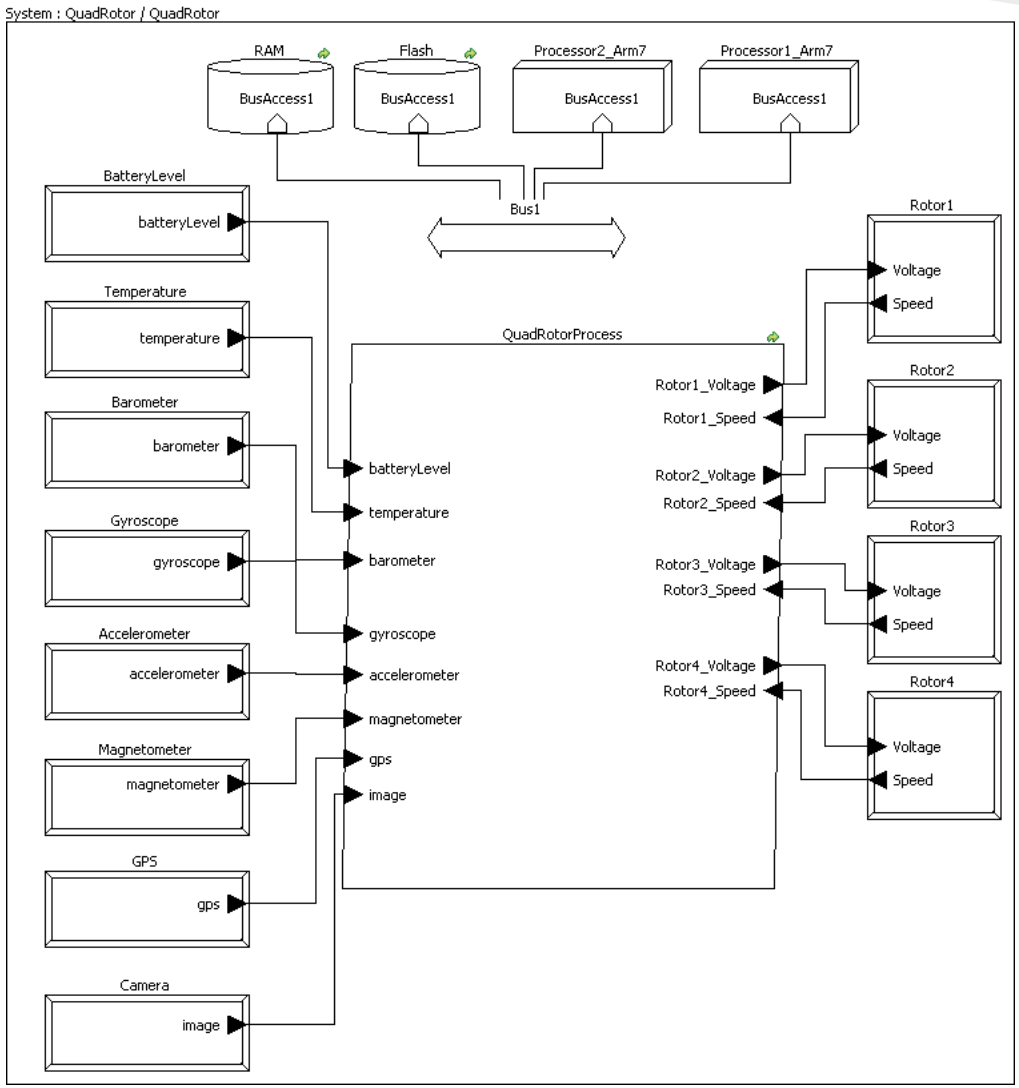
- Componentes representam o software e hardware do sistema. componentes de software, componentes de hardware e componentes do sistema (que permite agrupar componentes de software e hardware).
- Conectores que representam as conexões usadas para ligar os componentes.
- Dois modelos ADELE foram criados:
  - *Package diagram*: contém tipos de dados e definições de tipo dos processadores, rotores e barramento.
  - *System diagram*: depende do diagrama de package, contendo detalhes do sistema.

Visão geral do sistema (3/12)

- A modelagem teve início a partir da visão geral do sistema modelada de acordo com as especificações levantadas na análise de requisitos e com os elementos arquiteturais requeridos para implementação da modelagem funcional.

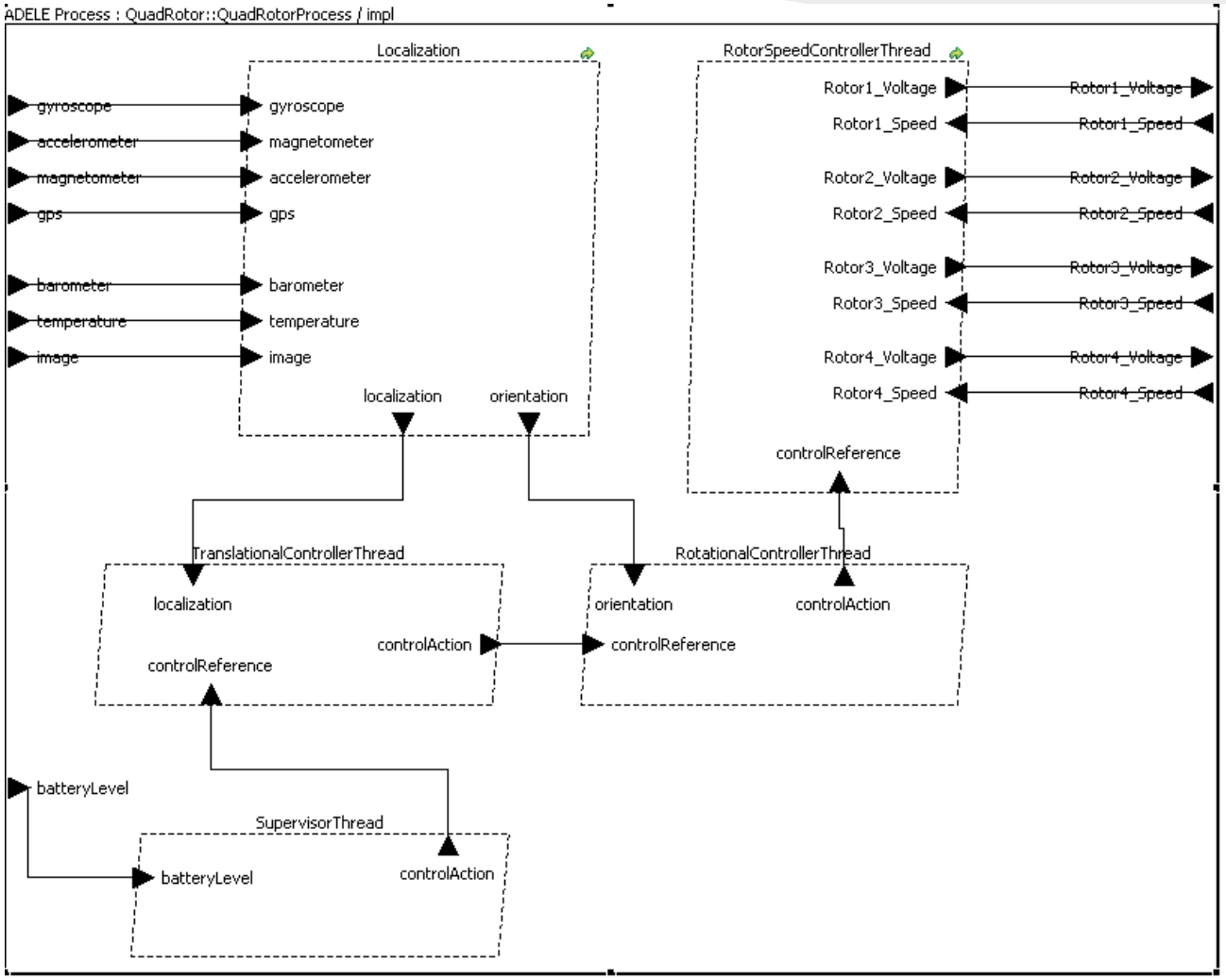
- Foram incluídos os sensores, os rotores, o processo representando o controlador e o hardware disponível no QuadRotor (memórias, barramento e processadores)

System Diagram (4/12)

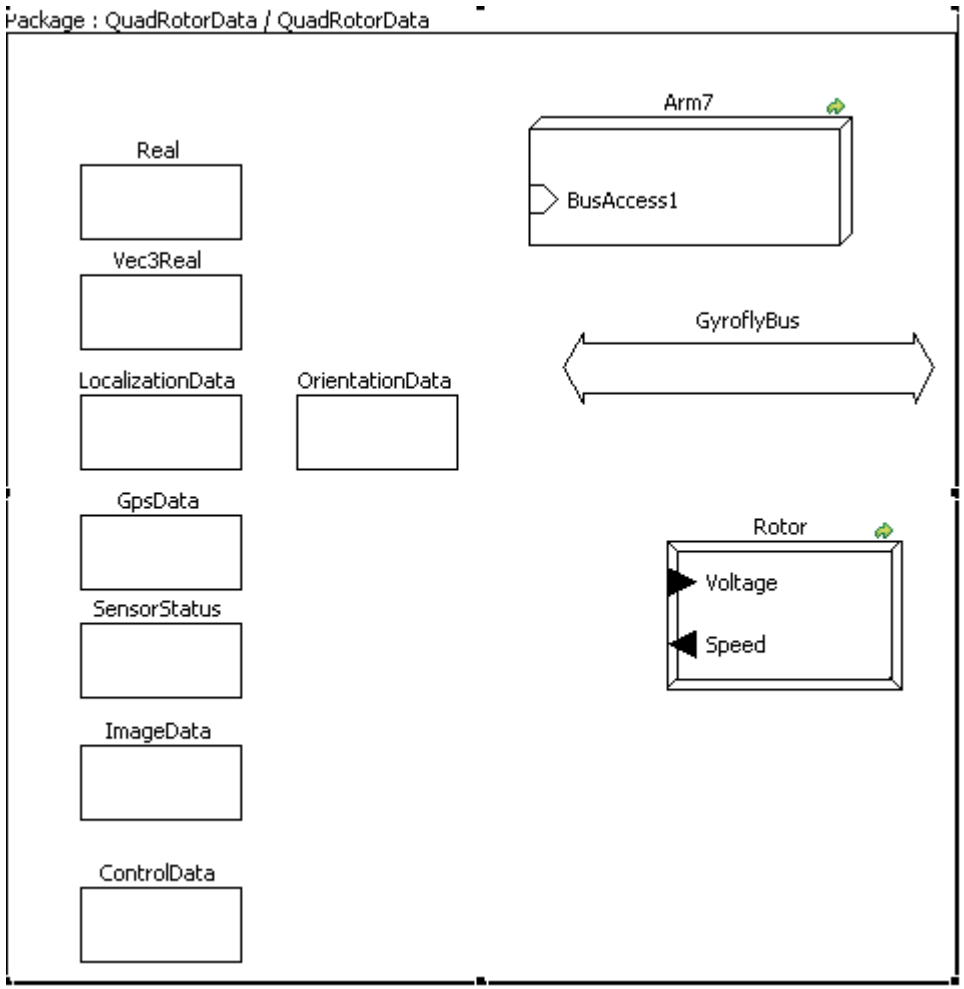



```
SYSTEM QuadRotor
END QuadRotor;

SYSTEM IMPLEMENTATION QuadRotor.others
SUBCOMPONENTS
  Flash : MEMORY Flash.MicroSDCard;
  Gyroscope : DEVICE Gyroscope;
  BatteryLevel : DEVICE BatteryLevel;
  Temperature : DEVICE Temperature;
  Barometer : DEVICE Barometer;
  GPS : DEVICE GPS;
  Accelerometer : DEVICE Accelerometer;
  Magnetometer : DEVICE Magnetometer;
  Camera : DEVICE Camera;
  QuadRotorProcess : PROCESS QuadRotorProcess.others;
  Rotor1 : DEVICE QuadRotorData::Rotor;
  Rotor2 : DEVICE QuadRotorData::Rotor;
  Rotor3 : DEVICE QuadRotorData::Rotor;
  Rotor4 : DEVICE QuadRotorData::Rotor;
  Bus1 : BUS QuadRotorData::GyroflyBus;
  RAM : MEMORY RAM;
  Processor1_Arm7 : PROCESSOR QuadRotorData::Arm7.rtos;
  Processor2_Arm7 : PROCESSOR QuadRotorData::Arm7.rtos;
CONNECTIONS
  DATA PORT Gyroscope.gyroscope -> QuadRotorProcess.gyroscope;
  DATA PORT BatteryLevel.batteryLevel -> QuadRotorProcess.batteryLevel;
  DATA PORT Temperature.temperature -> QuadRotorProcess.temperature;
  DATA PORT Barometer.barometer -> QuadRotorProcess.barometer;
  DATA PORT GPS.gps -> QuadRotorProcess.gps;
  DATA PORT Accelerometer.accelerometer -> QuadRotorProcess.accelerometer;
  DATA PORT Magnetometer.magnetometer -> QuadRotorProcess.magnetometer;
  DATA PORT Camera.image -> QuadRotorProcess.image;
  DATA PORT QuadRotorProcess.Rotor1_Voltage -> Rotor1.Voltage;
  DATA PORT QuadRotorProcess.Rotor2_Voltage -> Rotor2.Voltage;
  DATA PORT QuadRotorProcess.Rotor3_Voltage -> Rotor3.Voltage;
  DATA PORT QuadRotorProcess.Rotor4_Voltage -> Rotor4.Voltage;
  DATA PORT Rotor1.Speed -> QuadRotorProcess.Rotor1_Speed;
  DATA PORT Rotor2.Speed -> QuadRotorProcess.Rotor2_Speed;
```

System Diagram (5/12)



Package Diagram (6/12)




```
PACKAGE QuadRotorData
PUBLIC

  DEVICE Rotor
  FEATURES
    Voltage : IN DATA PORT QuadRotorData::Real;
    Speed : OUT DATA PORT QuadRotorData::Real;
  END Rotor;

  BUS GyroflyBus
  END GyroflyBus;

  PROCESSOR Arm7
  FEATURES
    BusAccess1 : REQUIRES BUS ACCESS QuadRotorData::GyroflyBus;
  END Arm7;

  PROCESSOR IMPLEMENTATION Arm7.rtos
  END Arm7.rtos;

  DATA Real
  END Real;

  DATA Vec3Real
  END Vec3Real;

  DATA LocalizationData
  END LocalizationData;

  DATA GpsData
  END GpsData;

  DATA SensorStatus
  END SensorStatus;

  DATA ImageData
  END ImageData;
```

Data Types (7/12)

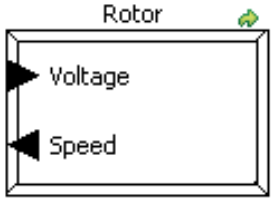
- Os tipos de dados utilizados na descrição do sistema foram modelados numa package separada.
- AADL e ADELE permitem a utilização de package para organizar o sistema.
- Os tipos descritos são baseados nos requisitos e nos tipos identificados na modelagem funcional.

Device (8/12)

- Cada sensor e rotor foram modelados como um *device*.

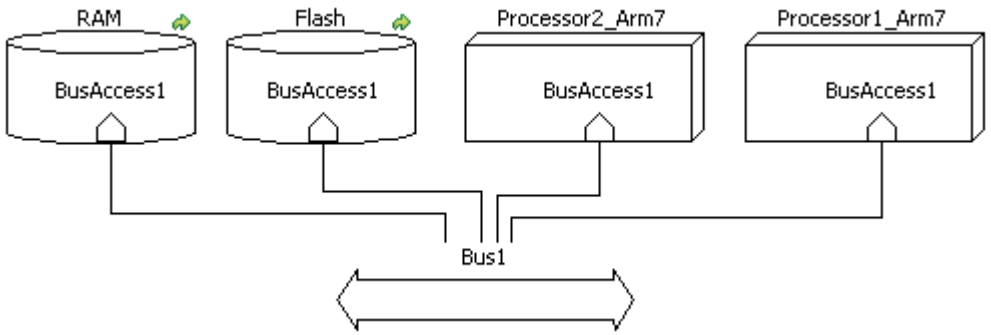
- Os rotores contêm uma porta de saída para leitura de velocidade do rotor, e uma porta de entrada contendo a voltagem a ser aplicada no rotor.

- Os sensores modelados com seu sinal de saída explicitando um dado de saída.



Processor, Memory e Bus (9/12)

- Processador: são responsáveis pelo escalonamento e execução de thread.
- Memória: representam componentes de armazenamento de dados.
- Barramento: comunicação associada a interação entre outros elementos da plataforma (memória, processador e device).



System (10/12)

- Na descrição do *system* QuadRotor foram incluídos os elementos de hardware processadores, memórias, *bus* e *devices* e o elemento de software: QuadRotorProcess.
- QuadRotorProcess descreve a implementação. A interconexão entre os devices e QuadRotorProcess são descritas na *implementation* do sistema.

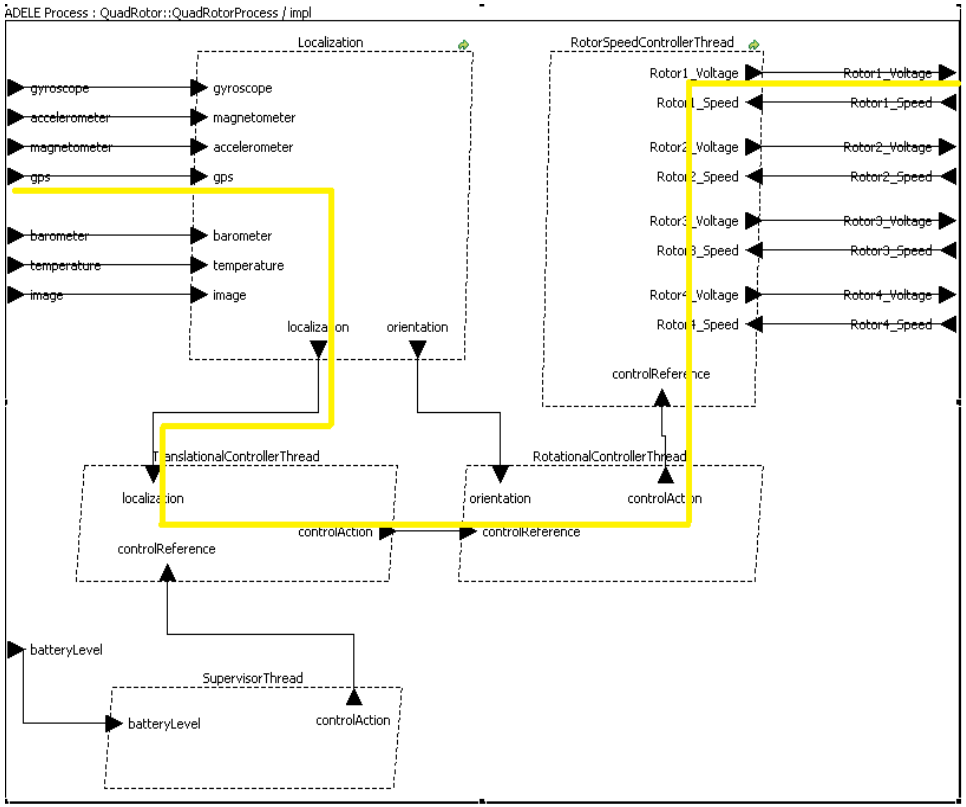
```
SYSTEM QuadRotor
END QuadRotor;

SYSTEM IMPLEMENTATION QuadRotor.others
SUBCOMPONENTS
 Flash : MEMORY Flash.MicroSDCard;
 Gyroscope : DEVICE Gyroscope;
 BatteryLevel : DEVICE BatteryLevel;
 Temperature : DEVICE Temperature;
 Barometer : DEVICE Barometer;
 GPS : DEVICE GPS;
 Accelerometer : DEVICE Accelerometer;
 Magnetometer : DEVICE Magnetometer;
 Camera : DEVICE Camera;
 QuadRotorProcess : PROCESS QuadRotorProcess.others;
 Rotor1 : DEVICE QuadRotorData::Rotor;
 Rotor2 : DEVICE QuadRotorData::Rotor;
 Rotor3 : DEVICE QuadRotorData::Rotor;
 Rotor4 : DEVICE QuadRotorData::Rotor;
 Bus1 : BUS QuadRotorData::GyroflyBus;
 RAM : MEMORY RAM;
 Processor1_Arm7 : PROCESSOR QuadRotorData::Arm7.rtos;
 Processor2_Arm7 : PROCESSOR QuadRotorData::Arm7.rtos;
CONNECTIONS
 DATA PORT Gyroscope.gyroscope -> QuadRotorProcess.gyroscope;
 DATA PORT BatteryLevel.batteryLevel -> QuadRotorProcess.batteryLevel;
 DATA PORT Temperature.temperature -> QuadRotorProcess.temperature;
 DATA PORT Barometer.barometer -> QuadRotorProcess.barometer;
 DATA PORT GPS.gps -> QuadRotorProcess.gps;
 DATA PORT Accelerometer.accelerometer -> QuadRotorProcess.accelerometer;
 DATA PORT Magnetometer.magnetometer -> QuadRotorProcess.magnetometer;
 END QuadRotor.others;
```

Thread (11/12)

- *Thread* É uma unidade escalonável de execução sequencial de um código fonte.
- Foram modeladas cinco threads em função dos componentes arquiteturais e funcionais do sistema.
  - *Localization:* processa as entadas dos sensores e gera informação de localização e orientação do quadrotor.
  - *Supervisor Thread:* controla o monitoramento do estado do sistema.
  - *TranslationalControllerThread:* processa a posição desejada do quadrotor. Sua saída é a entrada de controle da rotação.
  - *RotationalControllerThread:* define as velicidades necessárias de rotação dos rotores para estabeler a orientação desejada.
  - *RotorSpeedThread:* controla a vedlocidade de rotação de cada rotor individual para atingir os objetivos estabelecidos pelo controlador rotacional.

Flows (12/12)

- A especificação de *flows* possibilita a descrição e a análise de um caminho abstrato de informação através de um sistema. Eis um fluxo representando o caminho da informação de localização através do processo de controle.




```
DEVICE GPS
flows
 LocalizationFlow1: flow source gps;
END GPS;


PROCESS QuadRotorProcess
FEATURES
flows
 LocalizationFlow1: flow path gps -> Rotor4_Voltage;
END QuadRotorProcess;


THREAD Localization
flows
 LocalizationFlow1: flow path gps -> localization;
END Localization;


THREAD TranslationalControllerThread
flows
 LocalizationFlow1: flow path localization -> controlAction;
END TranslationalControllerThread;


THREAD RotationalControllerThread
flows
 LocalizationFlow1: flow path controlReference -> controlAction;
END RotationalControllerThread;
// ...
```

Desafios encontrados (1/2)

SCADE:

- A modelagem no SCADE necessita que o sistema seja visto sob outra óptica, até então desconhecida para os membros do grupo;

- Problemas para entender os requisitos da maneira como eles foram a apresentados

- Dificuldades para iniciar a modelagem
  - O que deve ser modelado?
  - Como integrar cada parte do sistema?
  - Como realizar os testes e simulações?

Desafios encontrados (2/2)

AADL:

- Ambiente TOPCASED apresentou problemas quando gerava .adeledi para .aadl

Problema resolvido com a versão 2007 do Eclipse (Eclipse Europa).

- Não encontramos uma forma de gerar o .aadl para .adeledi (de representação textual para gráfica).

- Algumas construções válidas não geram código AADL correto.

Visão crítica

AADL:

- No uso de ferramentas *open source*, no caso TOPCASED, surge o problema de escolher uma distribuição dentre as várias disponíveis. Dentre as distribuições testadas, algumas apresentaram plugins não funcionais ou com erro.

- Transformação de representação textual para gráfica.

- Teve-se melhor entendimento da modelagem em AADL após a modelagem funcional no SCADE.

Bibliografia

Castillo, P., Lozano, R. e Dzul, A. E. (2005). *Modelling and Control of Mini-Flying Machines*, Springer-Verlag, London, UK.

Hoffmann, G. M., Huang, H., Waslander, S. L. e Tomlin, C. J. (2007). *Quadrotor helicopter flight dynamics and control: Theory and experiment, Proc. of the AIAA Guidance, Navigation and Control Conference and Exhibit*, South Carolina, USA, pp. 1{20.

Raffo, G. V. (2011). *Robust Control Strategies for a QuadRotor Helicopter. An Underactuated Mechanical System*, PhD thesis, Universidad de Sevilla, Departamento de Ingeniería de Sistemas y Automática, Sevilla, España.

Obrigada!