
AutoPilot

**Report on Configuration
DefaultConfig**

PACKAGES

AutoPilot

For my personal project, I plan to make an automatic, solar powered nose hair trimmer. Just kidding!! I actually plan to design a basic autopilot system for a model airplane that will enable a plane (in this case a person) to get from point A to B. This project will form a jumping point in which more sophisticated measurements and inputs may be added in the future. This project will aim to take the GPS input in serial form with data inputs of longitude, latitude and heading and compute the direction one must move to get to the desired point. A user will then be able to input a desired coordinate (longitude/latitude pair). This will be done using the computer keyboard through the means of a serial cable to the board (See note). This will simulate sending an actual coordinate to the plane through a ham radio/serial interface. The system will tell by means of arrows the direction the user must move to reach the desired coordinates. In the future this will be translated into moving corresponding flight control surfaces. For example, if the user has entered that they would like to move to coordinate B, which happens to be 20 feet directly to the left of where they are standing, a left arrow would appear on the PIC LCD display board until the user turned to the direction of the point (coordinate). When this happens, a straight arrow would appear until the user walked to the desired coordinates. The user would never really be allowed to get to the designated coordinate because it simulates a plane circulating around the designated point. The user would therefore keep receiving arrows that would effectively direct the user circularly around the point (coordinate).

Based on the coordinates of the plane, point A, (from the GPS) the system will tell the person to 'fly' to another set of coordinates, point B. My algorithm will continually compute where the plane needs to go based on GPS data passed to the system. As the system may not be completely accurate, I will use some sort of weighted equation that takes into account the coordinates and heading of the previously calculated position and the current calculated heading and position.

Overall, the user will be able to input the following commands to the plane (user).

- Input coordinates to fly to*
- Fly straight based on current heading*
- Fly in circle (around the coordinate you are currently at- keep trying to get to current coordinate)*

Note: I am not sure if it is possible to send data from the PIC board (such as the plane's current coordinate) to the computer monitor. I know in lab today we performed the opposite as I was able to print out data on both the display and the LCD on the PIC board.

GLOBALS:

USE CASES:

Compute heading

Takes several consecutive readings and determines the direction plane is traveling by calculating planes last location compared to the current location.

Relations:
itsGPS

Fly To

This will represent an output, for now in the form of arrows on the display, telling the user what direction they must travel to achieve the desired mode.

For straight flight, computer will take the heading information based on several data readings and store it in memory. It will then continue to take readings as the plane is moving and output directions to help keep the plane lined up with the heading stored in memory.

To fly in a circle, computer will store the current longitude/latitude pair in memory and than output directional arrows working to fly as close to the point as possible.

To fly to a destination, the computer will first determine the heading the plane is flying in. Next it will find out

the desired coordinates and calculate where they are in relation to these coordinates. Finally, the necessary heading will be determined and appropriate turning arrows will be outputted to the display for the user to change their direction to the desired heading.

Set Desired Coordinate Parameters

User inputs the desired coordinates in which they would like to navigate. The user will input these coordinates on the keyboard of the computer in longitude/latitude format. This information will only be required if in destination mode.

Relations:

itsPilot

Set Operational Mode

User can specify the following modes of flight:

- 1. Fly straight- based on current heading*
- 2. Fly in circle- take the current location and continually try to get to that point*
- 3. Fly to specified coordinates (destination mode)*

The three modes will be selected from the keyboard of the computer. This will help to simulate being in a remote location, away from the plane. The user will be able to select the mode from a list and will always be able to return to the list at any point in flight. If this happens, the plane will automatically resort to circular flight.

Relations:

itsPilot

Default

GLOBALS:

CLASSES:

Altitude

This will be the height above ground.

Relations:

itsGPS_Data

Association with GPS_Data, Multiplicity of 1, Uni-directional

Attributes:

height

Type of double, Public

Control

This class, based on user input from the keyboard manages the mode the system currently is in and possibly the desired coordinates the user would like to fly to (if in that particular mode). Also determines the correct 'rudder' output based on mode and current heading to get to desired location.

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsSpeed

Association with Speed, Multiplicity of 1, Uni-directional

itsDirection

Association with Direction, Multiplicity of 1, Uni-directional

itsAltitude

Association with Altitude, Multiplicity of 1, Uni-directional

itsPosition

Association with Position, Multiplicity of 1, Uni-directional

itsKeyboard

Association with keyboard, Multiplicity of 1, Uni-directional

Attributes:

mode

Type of int, Public

rudder

Represents the amount of rudder deflection where 0 is the center and + and - values represent the angle at which the rudder deflects from center.

Type of int, Public, Initial Value: 0

Direction

Oggie has said that the GPS we use will calculate heading for me as long as the GPS is moving. If not, I will have to do the calculations myself and they will be accomplished here.

Relations:

itsGPS_Data

Association with GPS_Data, Multiplicity of 1, Uni-directional

Attributes:

heading

Type of unsigned int, Public

GPS_Data

Data will be accumulated every second through the use of Global Positioning Sattelites and stored in a data structure. It will be available for the different functions to use.

keyboard

This class will read user input and make it available to Control.

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Attributes:

keyHit

Type of char, Public

MonitorDisplay

Displays the mode options available to the user based on the control class. Also prints out the current coordinates, heading, altitude, and velocity.

Relations:

itsControl

Association with Control, Multiplicity of 1, Uni-directional

Position

This data pair will represent the coordinates on earth and will be in a long/lat. combination.

Relations:

itsGPS_Data

Association with GPS_Data, Multiplicity of 1, Uni-directional

Attributes:

lattitude

Type of double, Public

longitude

Type of double, Public

Speed

Returns the velocity. May be used eventually to control throttle.

Relations:

itsGPS_Data

Association with GPS_Data, Multiplicity of 1, Uni-directional

Attributes:

velocity

Type of double, Public

ACTORS:

GPS

The GPS will output serial data to the computer.

Relations:

itsCompute heading

Association with Compute heading, Multiplicity of 1, Bi-directional

Pilot

The pilot will activate the autopilot, select the desired mode and input the desired coordinates when necessary.

Relations:

itsSet Desired Coordinate Parameters

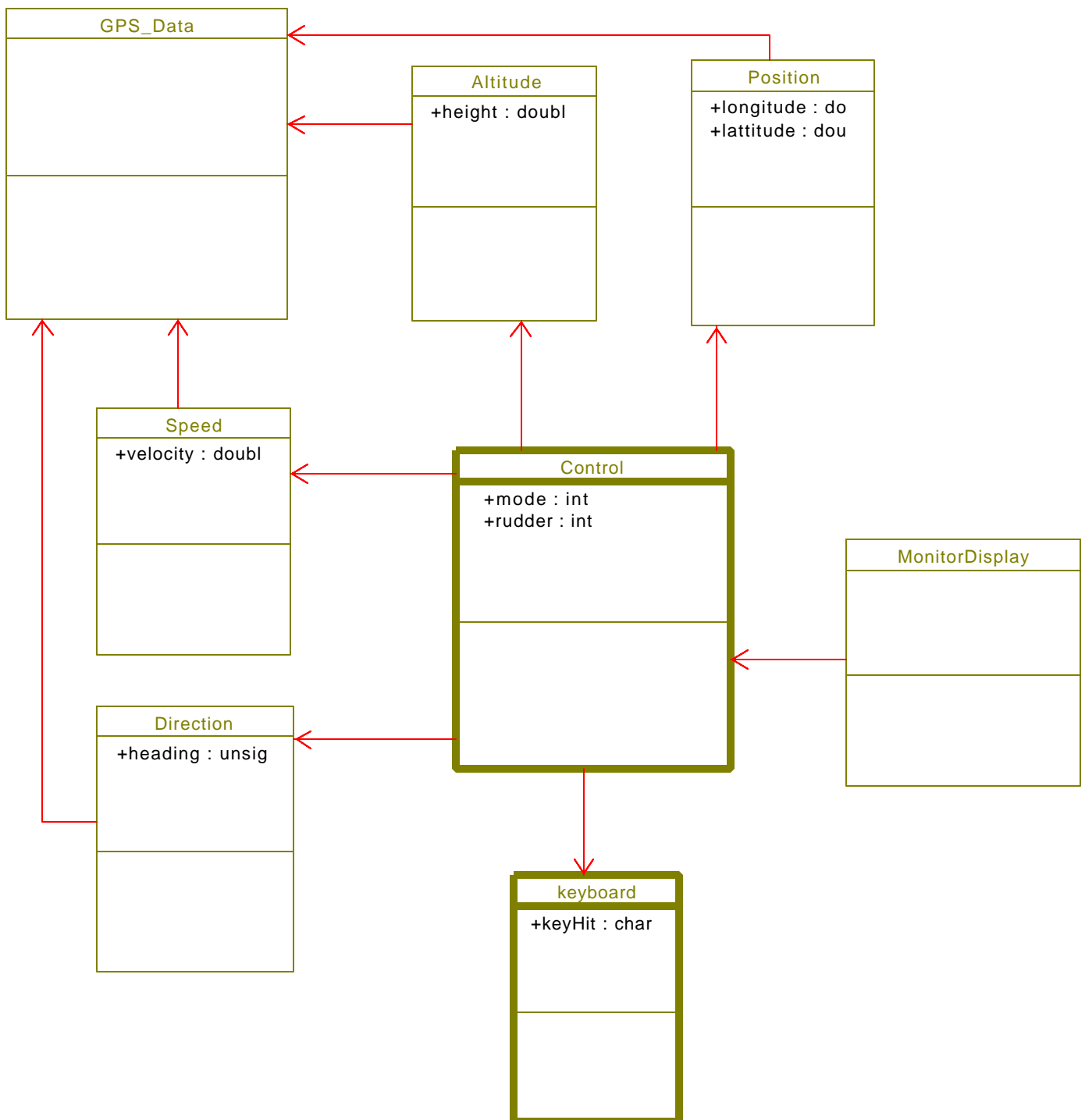
Association with Set Desired Coordinate Parameters, Multiplicity of 1, Bi-directional

itsSet Operational Mode

Association with Set Operational Mode, Multiplicity of 1, Bi-directional

OBJECT MODEL DIAGRAMS

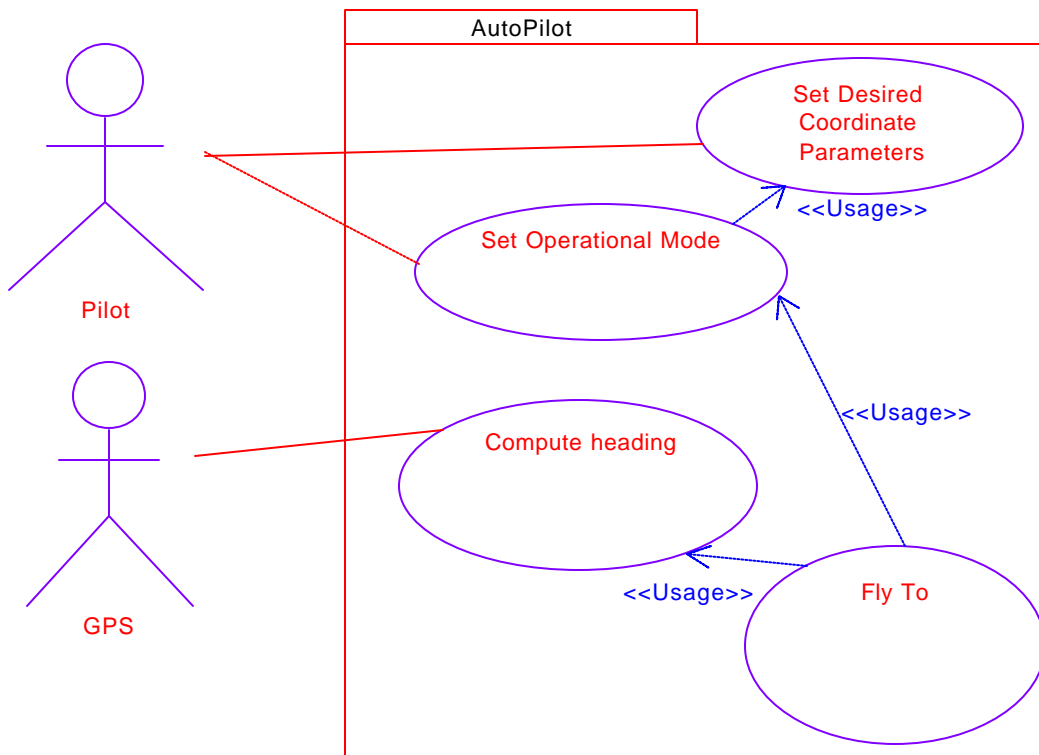
Model1



Looks fine to me.

USE CASE DIAGRAMS

AutoPilot



COMPONENTS

DefaultComponent

COMPONENT SETTINGS:

Build type: Executable

CONFIGURATIONS:

DefaultConfig

Scope type: Explicit

Instrumentation type: None

Time-model type: Real-time

Statechart generation type: Flat

FILES AND FOLDERS: