

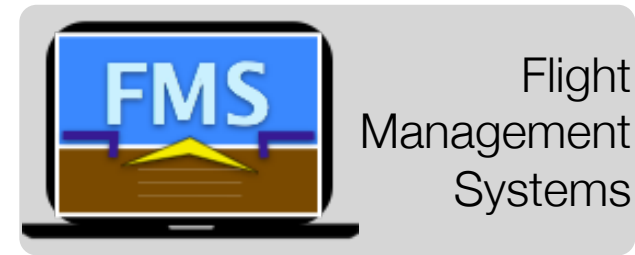
Flight Management Systems

Flight simulator

Writing a simple autopilot

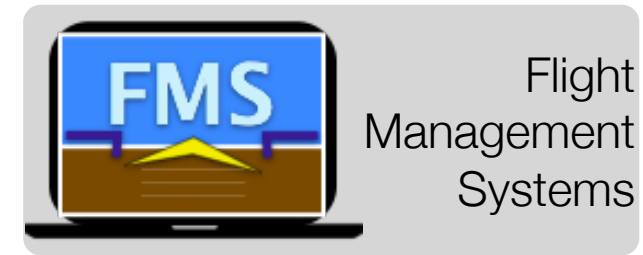
Joan Vila & Ángel Rodas

Building a simple autopilot



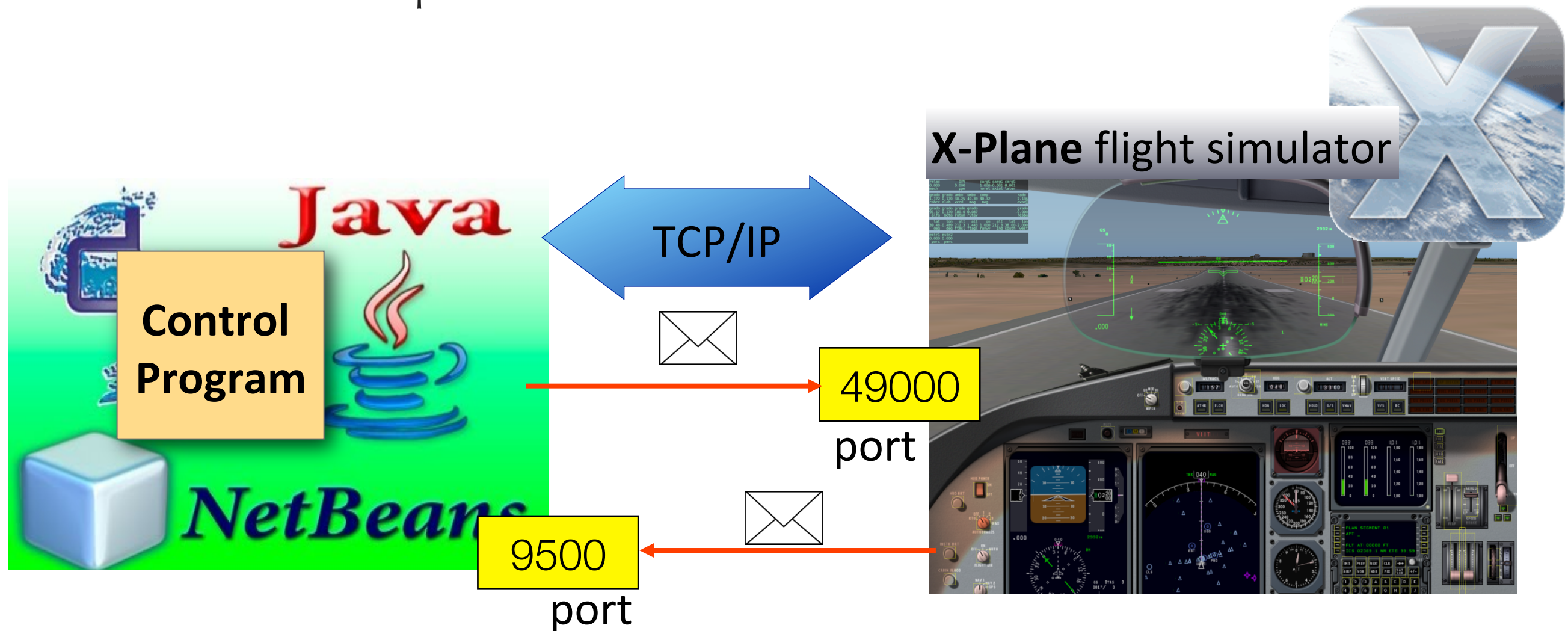
- **Communicating Java and XPlane using UDP**
- **A simple autopilot**

Communicating Java and XPlane



- **UDP communication**

- ✦ Communication ports



Communicating Java and XPlane



Flight
Management
Systems

- The “datarefs” tree

```
sim/aircraft/artstability/acf_ASh_hi_pos=0.000000
sim/aircraft/artstability/acf_ASh_lo_rate=0.000000
sim/aircraft/artstability/acf_AShiV=0.000000
sim/aircraft/artstability/acf_ASloV=0.000000
sim/aircraft/artstability/acf_ASmagh_hi=0.000000
sim/aircraft/artstability/acf_ASmagh_lo=0.000000
sim/aircraft/artstability/acf_ASmagp_hi=0.000000
sim/aircraft/artstability/acf_ASmagp_lo=0.000000
sim/aircraft/artstability/acf_ASmagr_hi=0.000000
sim/aircraft/artstability/acf_ASmagr_lo=0.000000
sim/aircraft/artstability/acf_ASp_hi_pos=0.000000
sim/aircraft/artstability/acf_ASp_lo_rate=0.000000
sim/aircraft/artstability/acf_ASr_hi_rate=0.000000
sim/aircraft/artstability/acf_ASr_lo_rate=0.000000
sim/aircraft/artstability/acf_has_clutch=0
sim/aircraft/bodies/acf_fuse_cd=0.115000
sim/aircraft/bodies/acf_fuse_cd_array=[0/95]:0.115000
sim/aircraft/controls/acf_RSC_idlespeed_prp=57.595871
sim/aircraft/controls/acf_RSC_maxgreen_prp=277.507385
sim/aircraft/controls/acf_RSC_mingov_prp=0.000000
sim/aircraft/controls/acf_RSC_mingreen_prp=198.967560
sim/aircraft/controls/acf_RSC_redline_prp=282.743378
sim/aircraft/controls/acf_ail1_crat=0.220000
sim/aircraft/controls/acf_ail1_dr=15.000000
sim/aircraft/controls/acf_ail1_up=20.000000
sim/aircraft/controls/acf_ail2_crat=0.000000
```


Communicating Java and XPlane



- The “datarefs” tree

- ✦ <http://www.xsquawkbox.net/xpsdk/docs/DataRefs.html>

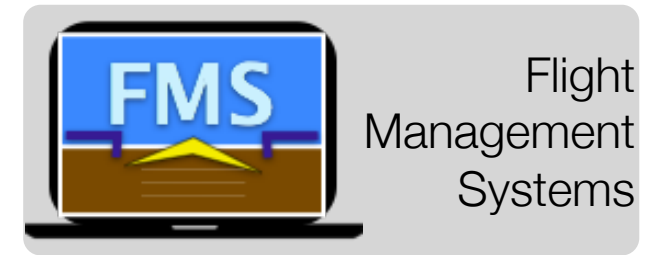
+ [sim/flightmodel/position/](#)

latitude	double	660+	no	degrees	The latitude of the aircraft
longitude	double	660+	no	degrees	The longitude of the aircraft
elevation	double	660+	no	meters	The elevation above MSL of the aircraft
theta	float	660+	yes	degrees	The pitch relative to the plane normal to the Y axis in degrees - OpenGL coordinates
phi	float	660+	yes	degrees	The roll of the aircraft in degrees - OpenGL coordinates
psi	float	660+	yes	degrees	The true heading of the aircraft in degrees from the Z axis - OpenGL coordinates

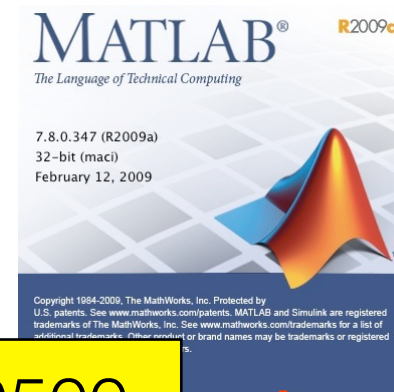
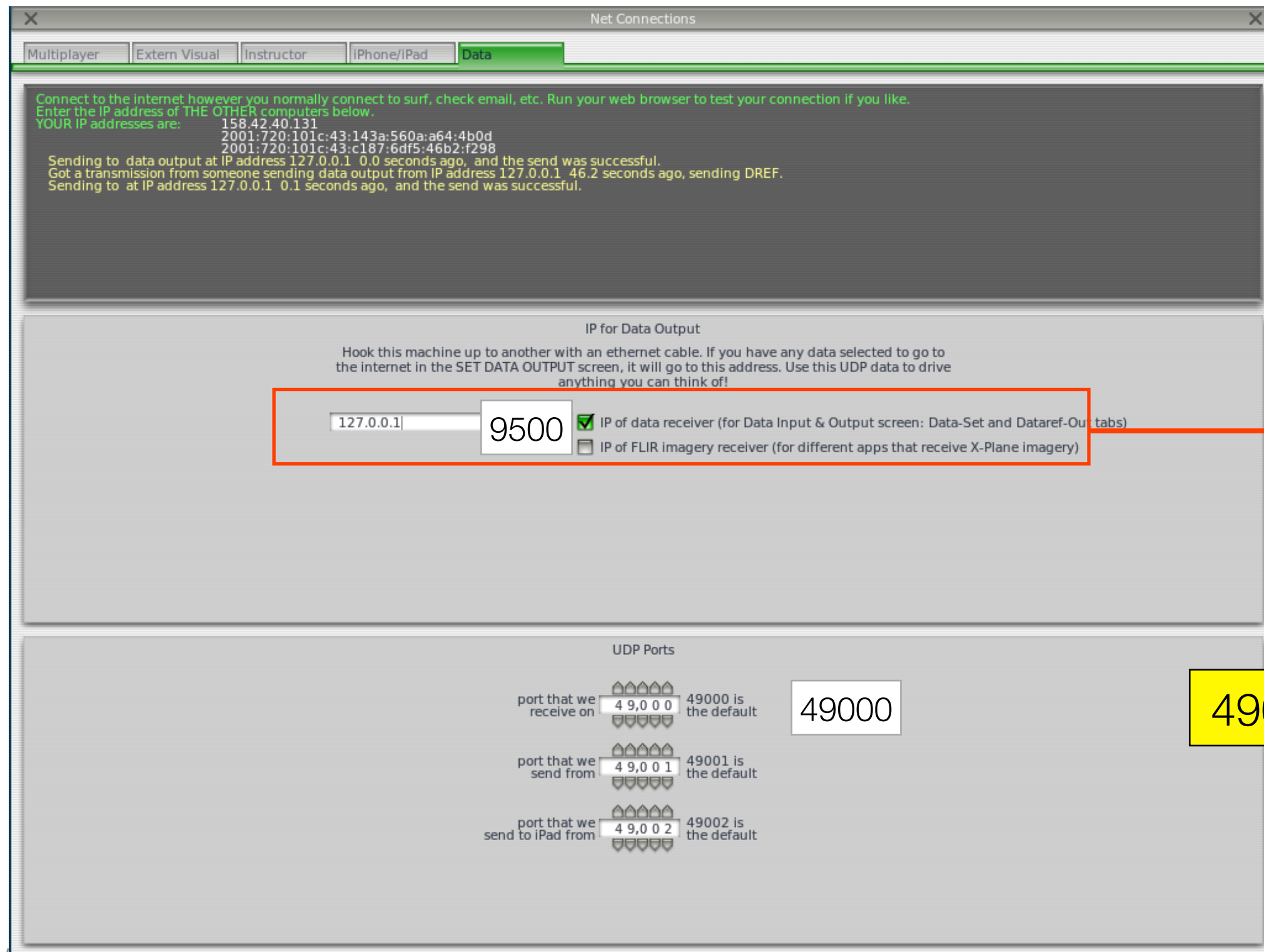
+ [sim/joystick/](#)

yoke_pitch_ratio	float	900+	yes	[-1..1]	The deflection of the joystick axis controlling pitch. Use <code>override_joystick</code> or <code>override_joystick_pitch</code>
yolk_pitch_ratio	float	660+	yes	[-1..1]	Legacy - this spelling error is still present for
yoke_roll_ratio	float	900+	yes	[-1..1]	The deflection of the joystick axis controlling roll. Use <code>override_joystick</code> or <code>override_joystick_roll</code>
yolk_roll_ratio	float	660+	yes	[-1..1]	Legacy - this spelling error is still present for backward compatibility with older plugins.
yoke_heading_ratio	float	900+	yes	[-1..1]	The deflection of the joystick axis controlling yaw. Use <code>override_joystick</code> or <code>override_joystick_heading</code>

Communicating Java and XPlane



- Change the X-plane net configuration



9500

49000

Communicating Java and XPlane



• X-plane data output configuration

Data Input & Output

Data Set | **Data See** | **Dataref-Out** | **Flight-Test**

enable: ☒ internet ☒ disk file ☒ graphical ☒ cockpit display

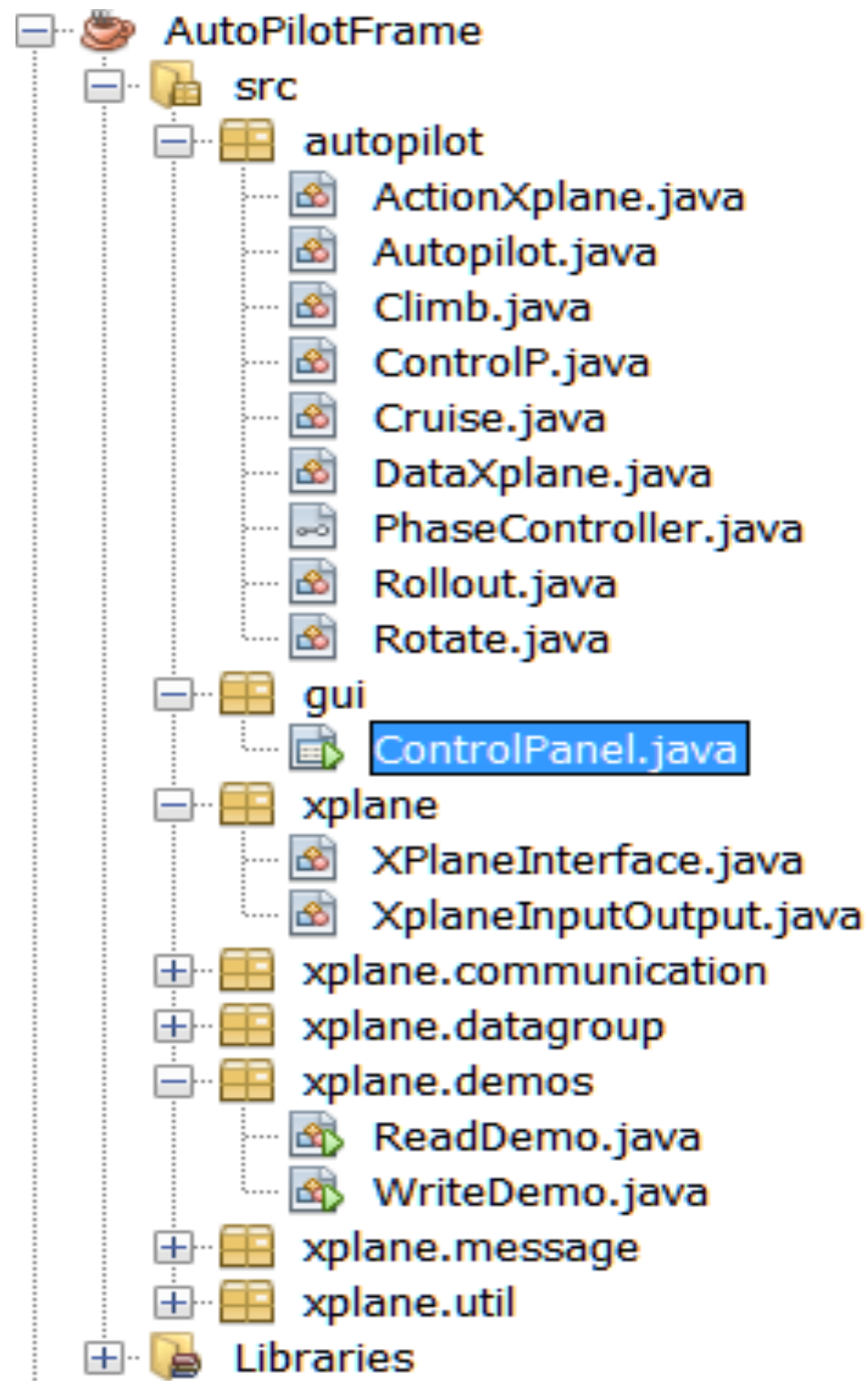
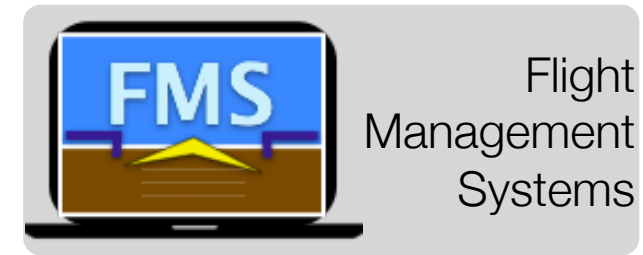
0 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> frame rate	33 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> starter timeout	70 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> defs: ailerons 1	106 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> switches 1:electrical
1 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> times	34 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> engine power	71 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> defs: ailerons 2	107 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> switches 2:EFIS
2 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> sim stats	35 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> engine thrust	72 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> defs: roll spoilers 1	108 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> switches 3:AP/f-dir/HUD
3 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> speeds	36 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> engine torque	73 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> defs: roll spoilers 2	109 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> switches 4:anti-ice
4 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Mach, VVI, G-load	37 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> engine RPM	74 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> defs: elevators	110 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> switches 5:anti-ice/fuel
5 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> atmosphere: weather	38 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> prop RPM	75 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> defs: rudders	111 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> switches 6:clutch/astab
6 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> atmosphere: aircraft	39 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> prop pitch	76 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> defs: yaw-brakes	112 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> switches 7:misc
7 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> system pressures	40 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> propwash/jetwash	77 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> control forces	
8 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> joystick ail/elv/rud	41 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> N1	78 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> TOTAL vert thrust vects	113 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> annunciators: general
9 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> other flight controls	42 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> N2	79 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> TOTAL lat thrust vects	114 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> annunciators: general
10 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> art stab ail/elv/rud	43 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> MP	80 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> pitch cyclic disc tilts	115 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> annunciators: engine
11 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> flight con ail/elv/rud	44 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> EPR	81 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> roll cyclic disc tilts	116 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> autopilot arms
12 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> wing sweep/thrust vect	45 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FF	82 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> pitch cyclic flapping	117 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> autopilot modes
13 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> trim/flap/slat/s-brakes	46 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> ITT	83 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> roll cyclic flapping	118 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> autopilot values
14 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> gear/brakes	47 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> EGT	84 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> grnd effect lift, wings	119 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> weapon status
15 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> angular moments	48 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> CHT	85 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> grnd effect drag, wings	120 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> pressurization status
16 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> angular velocities	49 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> oil pressure	86 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> grnd effect wash, wings	121 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> APU/GPU status
17 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> pitch, roll, headings	50 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> oil temp	87 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> grnd effect lift, stabs	122 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> radar status
18 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> AoA, side-slip, paths	51 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> fuel pressure	88 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> grnd effect drag, stabs	123 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> hydraulic status
19 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> mag compass	52 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> generator amperage	89 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> grnd effect wash, stabs	124 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> elec & solar status
20 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> lat, lon, altitude	53 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> battery amperage	90 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> grnd effect lift, props	125 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> icing status 1
21 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> loc, vel, dist traveled	54 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> battery voltage	91 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> grnd effect drag, props	126 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> icing status 2
22 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> all planes: lat	55 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> elec fuel pump on/off	92 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> wing lift	127 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> warning status
23 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> all planes: lon	56 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> idle speed lo/hi	93 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> wing drag	128 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> flite-plan legs
24 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> all planes: alt	57 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> battery on/off	94 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> stab lift	129 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> hardware options
25 <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> throttle command	58 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> generator on/off	95 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> stab drag	130 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> camera location
26 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> throttle actual	59 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> inverter on/off	96 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> COM 1/2 frequency	131 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> ground location
27 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> feathr-norm-beta-revers	60 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> FADEC on/off	97 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> NAV 1/2 frequency	132 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> climb stats
28 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> prop setting	61 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> igniter on/off	98 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> NAV 1/2 OBS	133 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> cruise stats
29 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> mixture setting	62 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> fuel weights	99 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> NAV 1 deflections	
30 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> carb heat setting	63 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> payload weights and CG	100 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> NAV 2 deflections	
31 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> cowl flap setting	64 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> aero forces	101 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> ADF 1/2 status	
32 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> magneto setting	65 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> engine forces	102 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> DME status	
	66 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> landing gear vert force	103 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> GPS status	
	67 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> landing gear deployment	104 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> XPNDR status	
	68 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> lift over drag & coeffs	105 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> MARKER status	
	69 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> prop efficiency		

Cockpit During Flight
Graphical Display in 'Data See'
Disk file 'data.txt'
Internet via UDP

detail: ☐ rotors ☐ propellers ☐ wings ☐ stabs & misc

UDP rate: 0 5.0 /sec
disk rate: 0 1.1 /sec

Java Project



DataXplane and ActionXplane classes



Flight
Management
Systems

```
public class ActionXplane {  
  
    private int phase; // current phase  
    private float ailerons;  
    private float elevators;  
    private float rudder;  
    private float throttle;  
    private float brake;  
    private float flaps;  
  
    public ActionXplane() {  
        ailerons = 0;  
        elevators = 0;  
        rudder = 0;  
        throttle = 0;  
        brake = 1;  
        flaps = 0;  
    }  
  
    public void setPhase(int phase) {  
        this.phase = phase;  
    }  
}
```

```
public class DataXplane {  
  
    private float lat;  
    private float lon;  
    private float alt;  
    private float roll;  
    private float pitch;  
    private float head;  
  
    private float aoa;  
    private float beta;  
    private float vpath;  
    private float hpath;  
  
    private float ias;  
    private float tas;  
    private float gs;  
    private float vs;  
  
    private float brake;  
  
    public void setLat(float lat) {  
        this.lat = lat;  
    }  
}
```

XplaneInputOutput class



```
public class XplaneInputOutput {

    private XPlaneInterface xpi;
    private DataXplane data;

    public XplaneInputOutput() {

        try {
            xpi = new XPlaneInterface("127.0.0.1", 9500, 49000, "DATAGroupConfig.xml");
            xpi.unregisterDATAMessages("");
            xpi.registerDATAMessages("3,4,8,13,14,16,17,18,20,25"); // 20 lat lon alt replaces 22 23 24
            xpi.startReceiving();
            try {
                Thread.sleep(1000); // wait a few before send actions
            } catch (InterruptedException ex) {
                Logger.getLogger(ControlPanel.class.getName()).log(Level.SEVERE, null, ex);
            }
        } catch (SocketException ex) {
            Logger.getLogger(ControlPanel.class.getName()).log(Level.SEVERE, null, ex);
        } catch (UnknownHostException ex) {
            Logger.getLogger(ControlPanel.class.getName()).log(Level.SEVERE, null, ex);
        }
        data = new DataXplane();
    }

}
```

XplaneInputOutput class



Flight
Management
Systems

```
public DataXplane read() {  
  
    data.setLat(xpi.getValue("position.lat"));  
    data.setLon(xpi.getValue("position.lon"));  
    data.setAlt(xpi.getValue("position.altMsl"));  
    data.setRoll(xpi.getValue("orientation.roll"));  
    data.setPitch(xpi.getValue("orientation.pitch"));  
    data.setHead(xpi.getValue("orientation.headingTrue"));  
    data.setAoa(xpi.getValue("orientation.alfa"));  
    data.setBeta(xpi.getValue("orientation.beta"));  
    data.setVpath(xpi.getValue("orientation.vpath"));  
    data.setHpath(xpi.getValue("orientation.hpath"));  
    data.setBrake(xpi.getValue("controls.brake"));  
    data.setIas(xpi.getValue("position.ias"));  
    data.setTas(xpi.getValue("position.tas"));  
    data.setGs(xpi.getValue("position.gs"));  
    data.setVs(xpi.getValue("position.vs"));  
    return data;  
}  
  
public void write(ActionXplane action) {  
  
    xpi.setValue("controls.aileronsPosition", action.getAilerons());  
    xpi.setValue("controls.elevatorsPosition", action.getElevators());  
    xpi.setValue("controls.rudderPosition", action.getRudder());  
    xpi.setValue("engine.throttleCommand1", action.getThrottle());  
    xpi.setValue("controls.brake", action.getBrake());  
    xpi.setValue("controls.flapHand1", action.getFlaps());  
}
```

Demos. Reading data from X-Plane



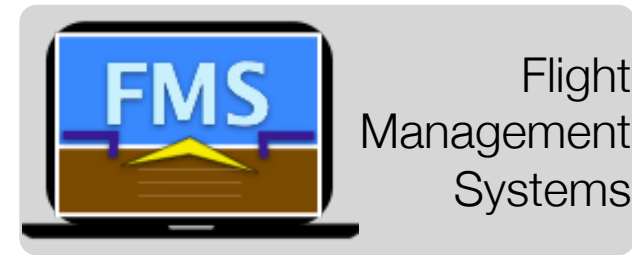
```
public class ReadDemo {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        XplaneInputOutput xplane;  
        DataXplane data;  
  
        xplane = new XplaneInputOutput();  
  
        try {  
            Thread.sleep(1000); // wait a little before reading  
        } catch (InterruptedException ex) {  
        }  
  
        data = xplane.read();  
        System.out.println("Latitude " + data.getLat() + " Longitude " + data.getLon() + " Altitude " + data.getAlt());  
  
        xplane.stop();  
    }  
}
```


Demos. Writing data to X-Plane



```
public class WriteDemo {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        XplaneInputOutput xplane;  
        ActionXplane action;  
  
        xplane = new XplaneInputOutput();  
        action = new ActionXplane();  
        action.setAilerons(-1);  
        action.setElevators(0);  
        action.setRudder(-1);  
        action.setThrottle(1);  
        action.setBrake(0);  
        action.setFlaps(1);  
        xplane.write(action);  
  
        xplane.stop();  
  
    }  
}
```

Building a simple autopilot



- **Communicating Java and XPlane using UDP**
- **A simple autopilot**

Autopilot class



```
public class Autopilot extends Thread {

    public static final int ROLLOUT = 1;
    public static final int ROTATE = 2;
    public static final int CLIMB = 3;
    public static final int CRUISE = 4;

    XplaneInputOutput xplane;
    ControlPanel mygui;
    private boolean running;
    private int sampleRate;

    private DataXplane data;
    private ActionXplane action;

    private PhaseController pController;
    private Rollout rollout;
    private Rotate rotate;
    private Climb climb;
    private Cruise cruise;

    public Autopilot(ControlPanel mygui) {

        this.xplane = new XplaneInputOutput();
        this.mygui = mygui;

        try {
            Thread.sleep(1000); // wait a few before
        } catch (InterruptedException ex) {
            Logger.getLogger(ControlPanel.class.getName());
        }
        rollout = new Rollout(); // initialize phases
        rotate = new Rotate();
        climb = new Climb();
        cruise = new Cruise();

        running = true;
        sampleRate = 200;
        this.start();
    }
}
```

```
@Override
public void run() {

    pController = rollout; // initial phase

    while (running) { // control loop

        data = xplane.read();
        mygui.display(data);
        action = pController.computeAction(data);
        xplane.write(action);
        mygui.display(action);

        switch (action.getPhase()) {
            case ROLLOUT:
                pController = rollout;
                break;
            case ROTATE:
                pController = rotate;
                break;
            case CLIMB:
                pController = climb;
                break;
            case CRUISE:
                pController = cruise;
                break;
        }

        try {
            Thread.sleep(sampleRate);
        } catch (InterruptedException ex) {
            Logger.getLogger(ControlPanel.class.getName());
        }

        xplane.stop();
    }
}
```

PhaseController interface and ControlP



```
public class ControlP {  
  
    float Kp;  
    float minv;  
    float maxv;  
  
    public ControlP(float Kp, float minv, float maxv) {  
        this.Kp = Kp;  
        this.minv = minv;  
        this.maxv = maxv;  
    }  
  
    public float control(float err) {  
        float action;  
        action = Kp * err;  
        action = Math.min(maxv, action);  
        action = Math.max(minv, action);  
        return action;  
    }  
}
```


PhaseController interface and ControlP



```
public interface PhaseController {  
  
    public ActionXplane computeAction(DataXplane data);  
  
}
```

```
public class Rollout implements PhaseController {  
  
    private ActionXplane actions;  
    private ControlP rudder_control;  
  
    public Rollout() {  
        actions = new ActionXplane();  
        rudder_control = new ControlP(0.03f, -1, 1);  
    }  
  
    @Override  
    public ActionXplane computeAction(DataXplane data) {
```

```
public class Rotate implements PhaseController {  
  
    private ActionXplane actions;  
    private ControlP lateral_guidance;  
    private ControlP lateral_control;  
  
    public Rotate() {  
        actions = new ActionXplane();  
        lateral_guidance = new ControlP(0.2f, -20, 20);  
        lateral_control = new ControlP(0.05f, -1, 1);  
    }  
  
    @Override  
    public ActionXplane computeAction(DataXplane data) {
```

```
public class Climb implements PhaseController {  
  
    private ActionXplane actions;  
    private ControlP lateral_guidance;  
    private ControlP lateral_control;  
    private ControlP vertical_guidance;  
    private ControlP vertical_control;  
  
    public Climb() {  
        actions = new ActionXplane();  
        lateral_guidance = new ControlP(0.2f, -20, 20);  
        lateral_control = new ControlP(0.05f, -1, 1);  
        vertical_guidance = new ControlP(1, -15, 15);  
        vertical_control = new ControlP(0.001f, -1, 1);  
    }  
  
    @Override  
    public ActionXplane computeAction(DataXplane data) {
```

```
public class Cruise implements PhaseController {  
  
    private ActionXplane actions;  
    private ControlP lateral_guidance;  
    private ControlP lateral_control;  
    private ControlP vertical_guidance;  
    private ControlP vertical_control;  
  
    public Cruise() {  
        actions = new ActionXplane();  
        lateral_guidance = new ControlP(0.2f, -20, 20);  
        lateral_control = new ControlP(0.05f, -1, 1);  
        vertical_guidance = new ControlP(0.6f, -15, 15);  
        vertical_control = new ControlP(0.006f, -1, 1);  
    }  
  
    @Override  
    public ActionXplane computeAction(DataXplane data) {
```

GUI. ControlPanel class



Autopilot

X-Plane data

Latitude	Longitude	Altitude	Roll	Pitch	Head
39,50	-0,50	226	-0,49	1,84	116,3
	AoA	Beta	Vpath	Hpath	
	1,78	0,05	0,29	116,3	
	las	Tas	Gs	Vs	<input type="radio"/> brake
	42,1	42,3	42,3	4	

X-Plane control

Ailerons	Elevators	Rudder	Throttle	Flaps
0,000000	0,000000	0,081073	1,00	0,00

PHASE: rollOut

```
public class ControlPanel extends javax.swing.JFrame {

    /**
     * Creates new form ControlPanel
     */
    private Autopilot aPilot;

    public ControlPanel() {
        initComponents();
        aPilot=new Autopilot(this);
    }

    public void display(DataXplane dx) {

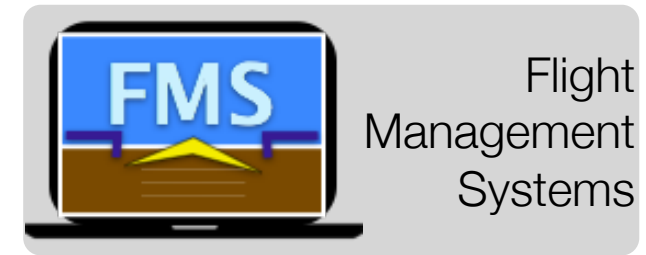
        latText.setText(String.format("%.2f", dx.getLat()));
        lonText.setText(String.format("%.2f", dx.getLon()));
        altText.setText(String.format("%.0f", dx.getAlt()));
        rollText.setText(String.format("%.2f", dx.getRoll()));
        ...

    public void display(ActionXplane ax) {

        String phase="";
        switch (ax.getPhase()) {
            case Autopilot.ROLLOUT:
                phase = "rollOut";
                break;
            case Autopilot.ROTATE:
                phase = "rotate";
                break;
            case Autopilot.CLIMB:
                phase = "climb";
                break;
            case Autopilot.CRUISE:
                phase = "cruise";
                break;
        }
        phaseLabel.setText(phase);
        ailText.setText(String.format("%.6f", ax.getAilerons()));
        eleText.setText(String.format("%.6f", ax.getElevators()));
        rudText.setText(String.format("%.6f", ax.getRudder()));
        thrText.setText(String.format("%.2f", ax.getThrottle()));
        brakeRadioButton.setSelected(ax.getBrake() != 0);
        flapText.setText(String.format("%.2f", ax.getFlaps()));

    }
}
```

Exercise: a simple autopilot



◦ A simple autopilot for the Cessna 172

- ✦ Complete the Netbeans project to take off the Cessna from LEVC RWY 12 runway and take it cruise phase to 2000 ft and heading 120.
- ✦ The program will consist of the following phases:
 - *Rollout*
 - *Rotation*
 - *Climb*
 - *Cruise*

Exercise: a simple autopilot

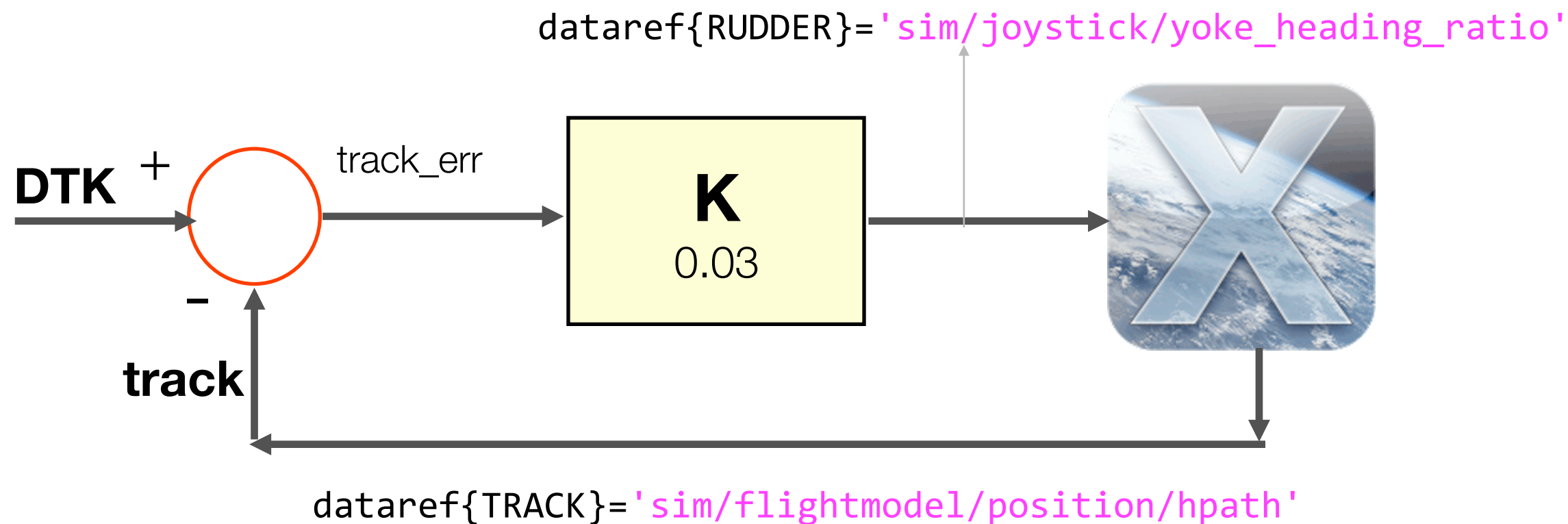


- **Rollout phase**

- ✦ **Throttle:** full
- ✦ **Elevators:** no deflection
- ✦ **Rudder control:** the rudder controls the heading on ground because it acts on the front wheel.
 - **Control policy:** *Proportional control. Based on track error.*
 - ▶ $DTK = 120;$
 - ▶ $error_h = DTK - track;$
 - ▶ $actions(RUDDER) = K_{rud} * error_h;$
 - *Try with different K_{rud} . Suggestion: $K_{rud} = 0.03$*
- ✦ **Ailerons:** no deflection
- ✦ **Transition** to the next phase when the $TAS > 70$ kts.

Exercise: a simple autopilot

- Rollout phase
 - Rudder control



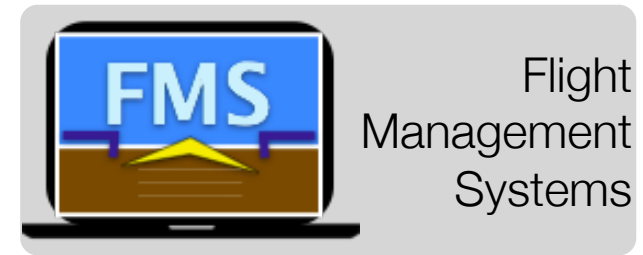
Rollout code



```
public class Rollout implements PhaseController {  
  
    private ActionXplane actions;  
    private ControlP rudder_control;  
  
    public Rollout() {  
        actions = new ActionXplane();  
        rudder_control = new ControlP(0.03f, -1, 1);  
    }  
}
```

```
@Override  
public ActionXplane computeAction(DataXplane data) {  
  
    final float DTK = 119; // references  
    final float TAS_MAX = 70; // termination conditions  
    float tas, track; // inputs  
    float rudder; // outputs  
    float track_error; // errors  
  
    // READ DATA-----  
    tas = data.getTas();  
    track = data.getHpath();  
  
    // LATERAL GUIDANCE-----  
    // CONTROL LOOP  
    track_error = DTK - track;  
    rudder = rudder_control.control(track_error);  
  
    // SET ACTIONS-----  
    actions.setAilerons(0);  
    actions.setElevators(0);  
    actions.setRudder(rudder);  
    actions.setThrottle(1);  
    actions.setBrake(0);  
    actions.setFlaps(0.0f);  
  
    // SET NEXT FLIGHT PHASE-----  
    if (tas > TAS_MAX) {  
        actions.setPhase(Autopilot.ROTATE);  
    } else {  
        actions.setPhase(Autopilot.ROLLOUT);  
    }  
  
    return actions;  
}
```

Exercise: a simple autopilot

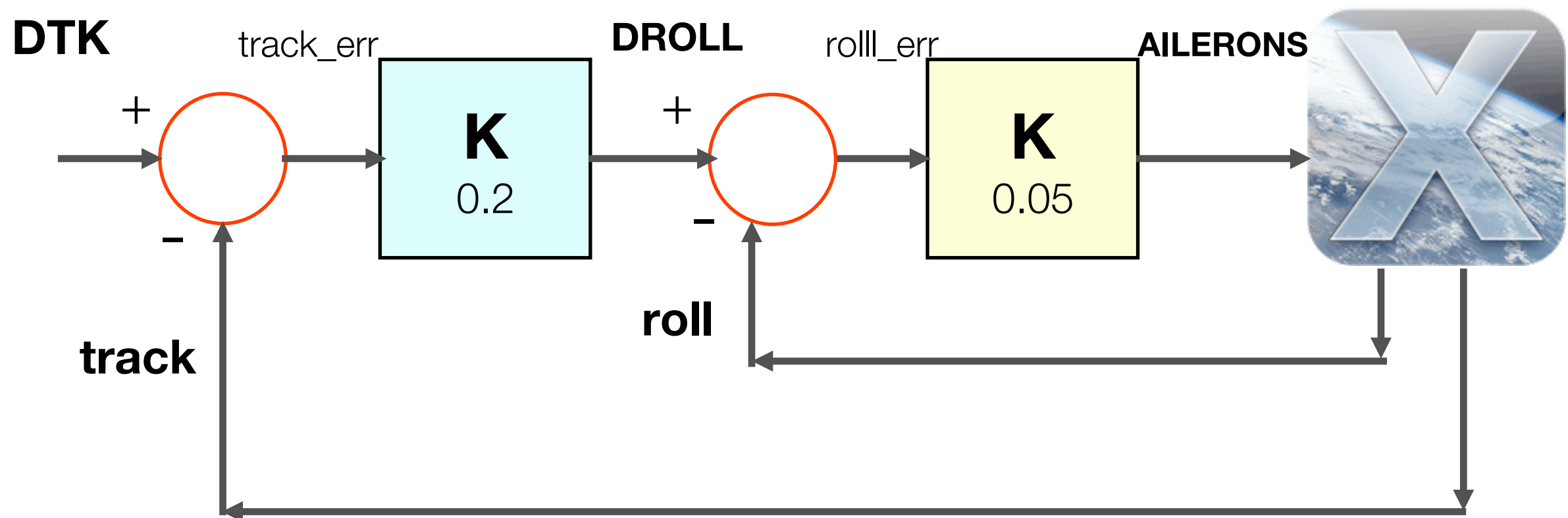


- **Rotate phase**

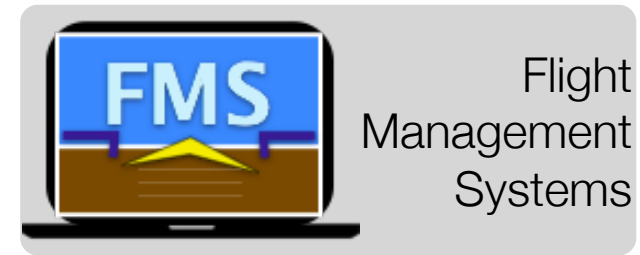
- ✦ **Throttle:** full
- ✦ **Elevators:** a fix value to start climbing.
 - *Suggestion: 0.01*
- ✦ **Rudder:** navigating straight does not require to use rudder actions.
- ✦ **Ailerons:** ailerons are used to control the roll and thus the track angles.
 - **Control policy:** *Proportional control. Double loop:*
 - **Guidance loop:** *compute roll angle based on track error. $K=0.2$*
 - **Control loop:** *control roll angle based on roll error. $K=0.05$*
- ✦ **Transition** to the next phase when altitude > 400 ft.

Exercise: a simple autopilot

- Rotate phase
 - Ailerons control



Exercise: a simple autopilot

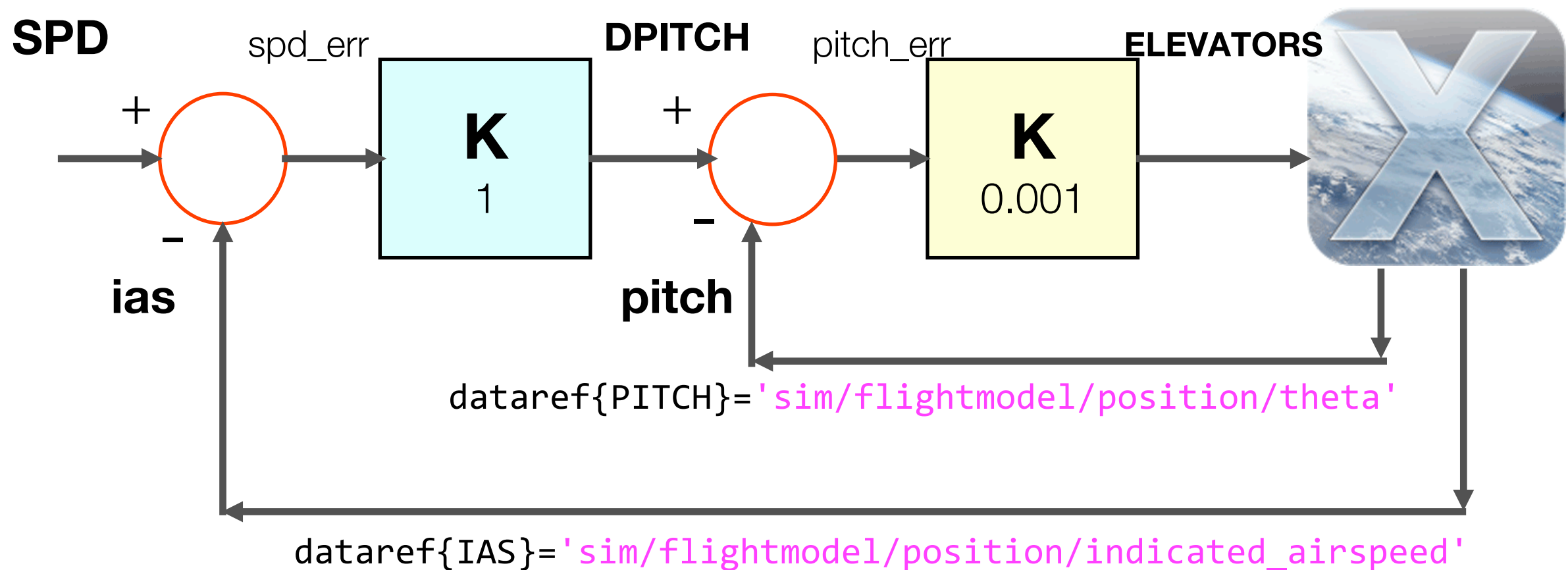


◦ Climb phase

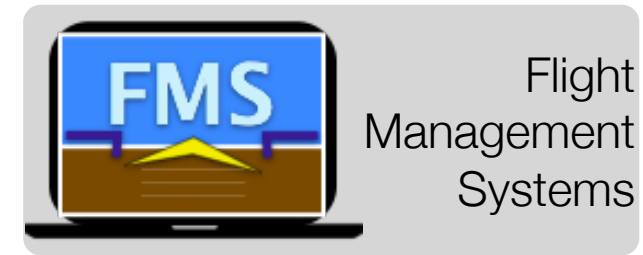
- ✦ **Throttle:** full
- ✦ **Elevators:** Climb after takes off is a constant speed climb (TAS/IAS). The target value for (TAS/IAS) will be 90 kts.
 - ✦ ***Elevators control law:*** *if the speed decreases, the pitch angle will be reduced to increase speed and avoid stall and viceversa.*
 - ***Control policy:*** *Proportional control. Double loop:*
 - ***Guidance loop:*** *compute pitch angle based on speed error. $K=1$*
 - ***Control loop:*** *control pitch angle based on pitch error. $K=0.001$*
- ✦ **Rudder:** navigating in straight does not require to use rudder actions.
- ✦ **Ailerons:** Similar to previous phase.
- ✦ **Transition** to the next phase when altitude > 2000 ft.

Exercise: a simple autopilot

- Climb phase
 - ✦ Elevators control



Exercise: a simple autopilot



◦ Cruise phase

- ✦ The goal of this phase will be keeping a **constant altitude of 2000 ft** and turning to **course 060°**.
- ✦ **Throttle:** reduce to 50%. Watch the tachometer is in the green area to adjust the right value.
- ✦ **Elevators:** keep constant altitude of 2000 ft. Perform a proportional control based on altitude error.
 - **Control policy:** *Proportional control. Double loop:*
 - **Guidance loop:** *compute pitch angle based on altitude error. $K=0.6$*
 - **Control loop:** *control pitch angle based on pitch error. $K=0.006$*
- ✦ **Rudder:** no deflection.
- ✦ **Ailerons:** Similar to previous phase. Now the DTK is 060°.
- ✦ **Transition:** None.