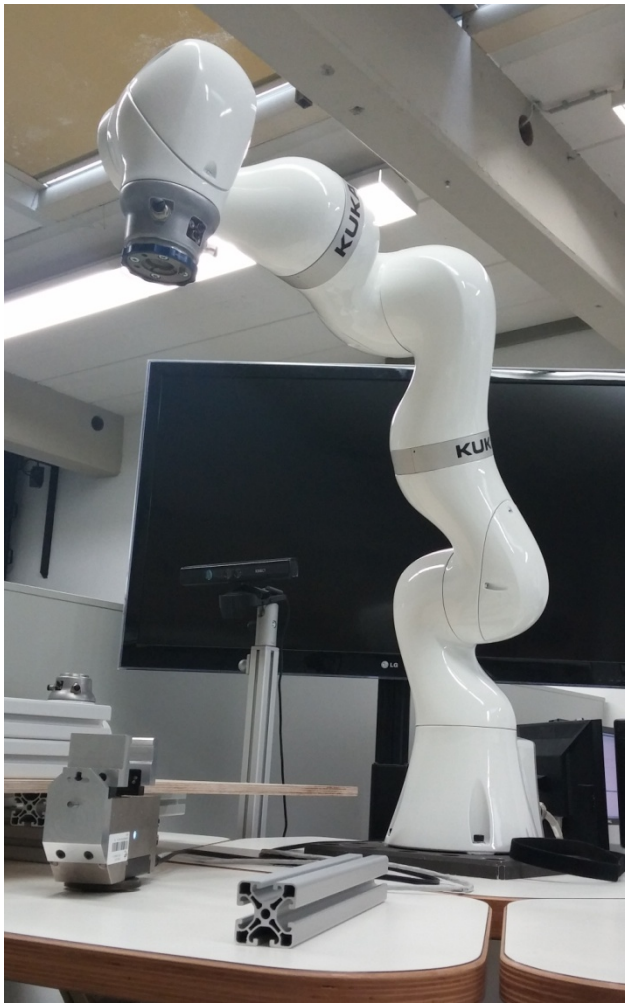
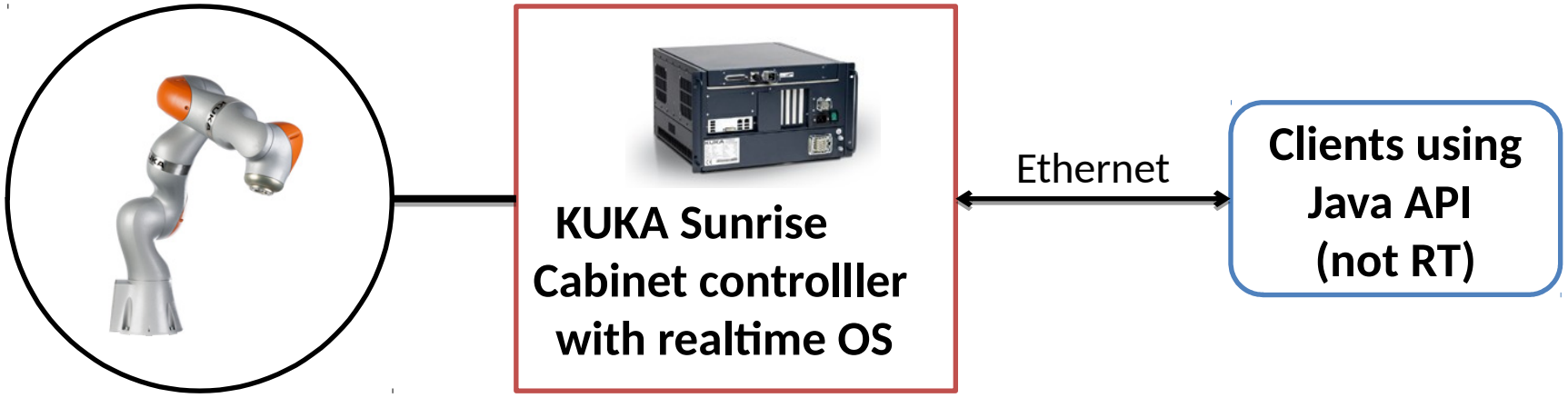
Programming with KUKA Sunrise.OS

KUKA LBR IIWA 7 R800

- 7 DOF lightweight robot
- Weight: 22.5 kg
- Load: 7 kg
- Height: 1.25 m
- Range: 0.8 m



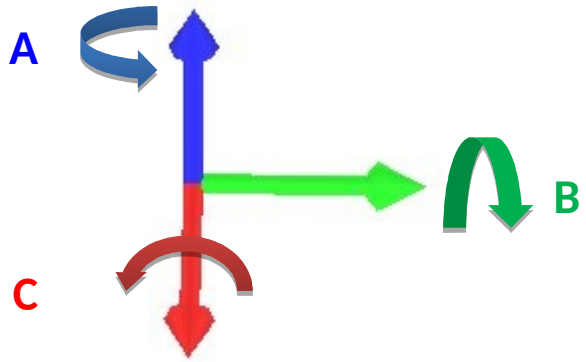
System Setup

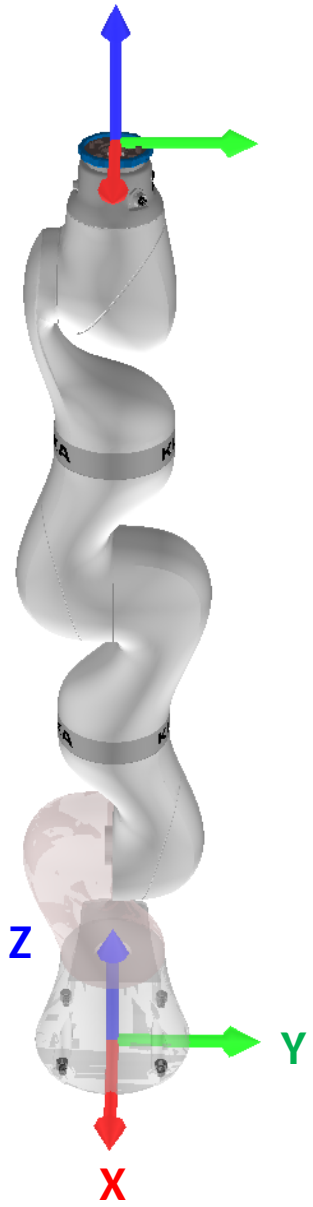


- Prepared Eclipse-Workspace provided
- API useable in regular Java programs

Coordinate Systems

- World CS located in base of robot
- KUKA convention:
  - Distances in mm
  - Rotations in degree (frames) or radians (joint positions)





Basic Usage

- Important classes:
  - **LBR**: communication with robot, created at startup
  - **JointPosition**: 7D vector of motor positions [rad]

```
j = new JointPosition(0,Math.Pi,0,0,0,0,0)
```

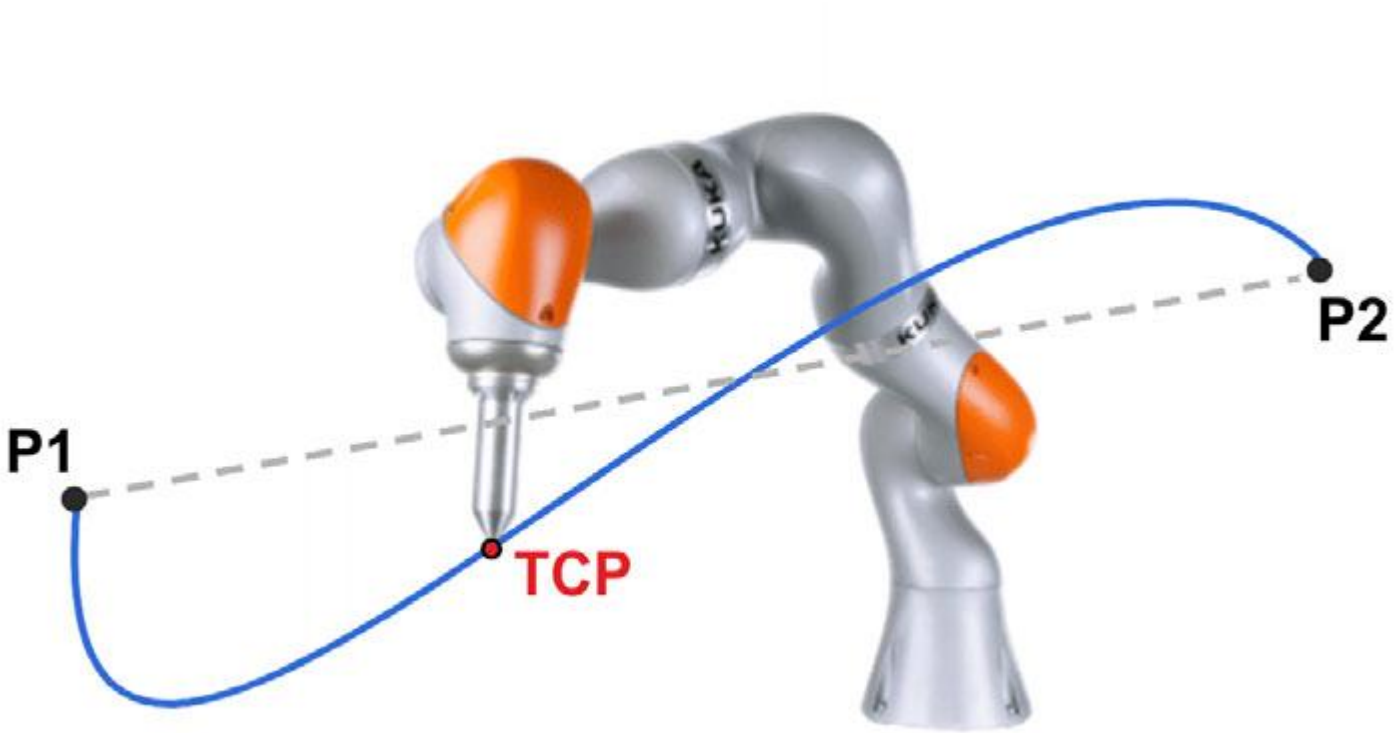
  - **Frame**: describes spatial displacements in space, contains redundancy parameters

```
f = new Frame (x,y,z,a,b,c)
```

  - **IMotion**: Interface for motion commands

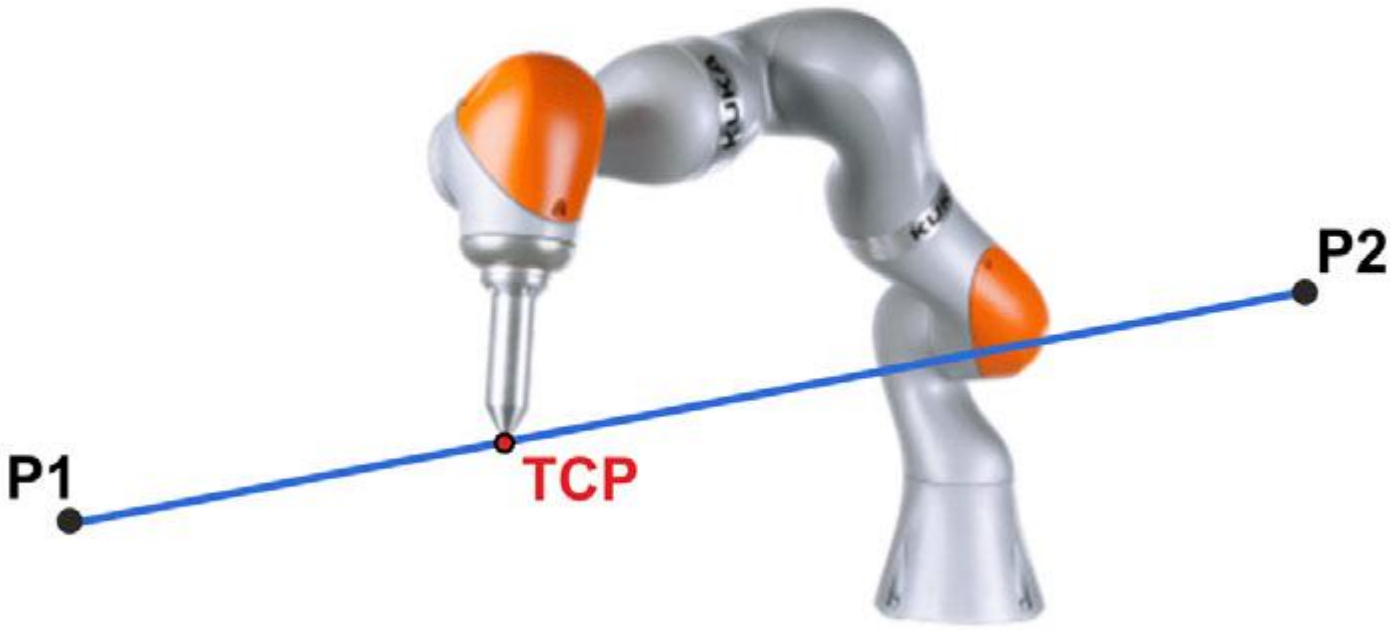
Basic Motions: PTP

```
BasicMotions.ptp(JointPosition);
BasicMotions.ptp(Frame);
```



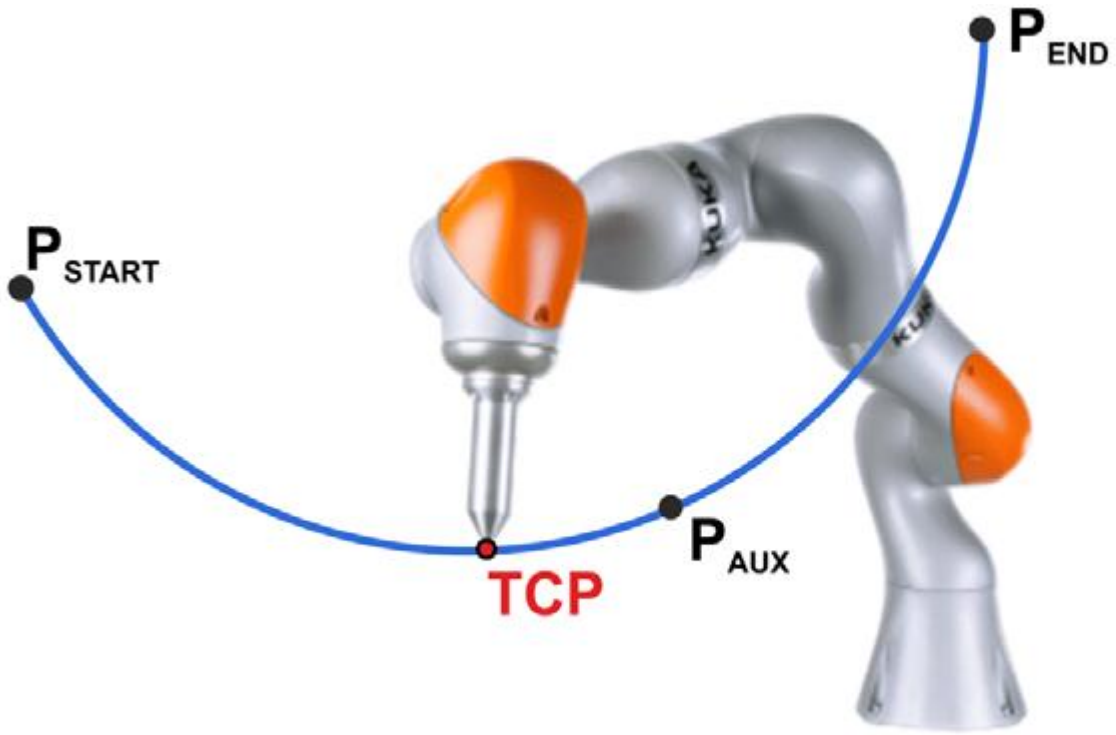
Basic Motions: LIN

```
BasicMotions.lin(Frame);
BasicMotions.linRel(x, y, z);
```



Basic Motions: CIRC

```
BasicMotions.circ(Frame aux, Frame end);
```



Basic Motions

- Execute motions synchronously/async.:

```
LBR.move(IMotion); // blocking call
LBR.moveAsync(IMotion); // non-blocking call
```

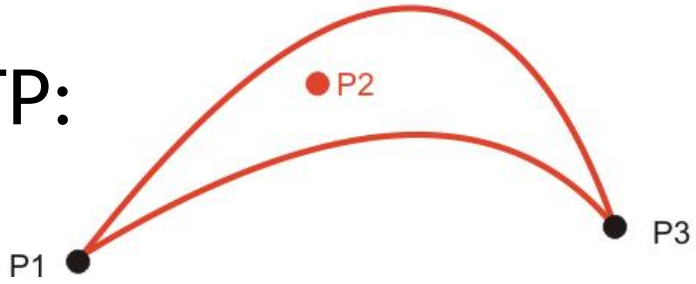
- Important parameters:
  - Velocity
  - Acceleration

```
IMotion.setJointAccelerationRel(0.5).setJointVel
ocityRel(0.3);
```

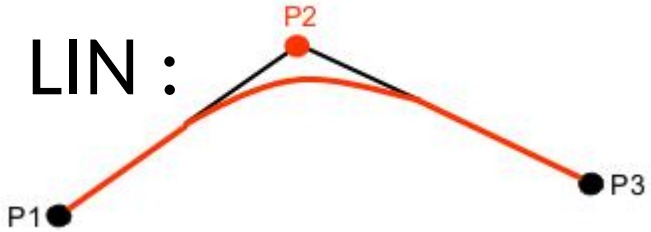
Blending

```
IMotion.setBlendingRel(0.2); // relative value
IMotion.setBlendingCart(20); // in millimeters
```

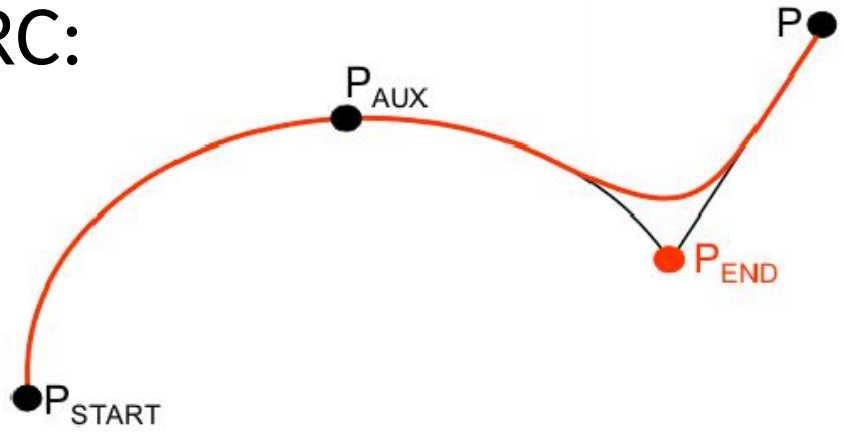
PTP:



LIN :



CIRC:



Combine Motions

- Create motion batches to combine motions

```
Vector<RobotMotion<?>> l = [...];
l.add(IMotion m1);
l.add(IMotion m2);
...
MotionBatch mb = new MotionBatch(l);
```

- Set parameters per motion or for complete motion batch
- Blending of motion commands (for motion batches or async movements)

Read Sensor Values

- Joint positions and tcp frame

```
LBR.getCurrentJointPosition();
LBR.getCurrentCartesianPosition(LBR.getFlange());
```

- Torque sensors

```
LBR.getSensorForMeasuredTorque().getSensorData();
LBR.getSensorForExternalTorque().getSensorData();
LBR.getSensorForExternalForce().getSensorData();
```

Conditions

- Place break conditions on motions:
  - JointTorque Condition
  - Force Condition

```
IMotion.breakWhen(new JointTorqueCondition(
JointEnum, minTorque, maxTorque))
```

- Combine conditions with and, or, xor..

```
ICondition cond = ICondition.and(ICondition);
```

Compliant Control

- Joint impedance and Cartesian impedance controller available

```
IMotion.setMode(ImpedanceControlMode);
```

- Joint impedance controller:

```
JointImpedanceControlMode JICM = new Joint…();
JICM.setStiffness(<7 values>);
JICM.setDamping(<7 values>);
...
```

Compliant Control

- Cartesian impedance controller:
  - DOF-dependent (stiffness, damping , maximum TCP force, …)

```
CartesianImpedanceControlMode CICM = new Cart…();
CICM.parametrize(CartDOF.X).setStiffness(3000);
CICM.parametrize(CartDOF.ALL).setDamping(0.7);
CICM.setMaxControlForce(maxForceX, maxForceY, ...);
```

  - DOF-independent (nullspace stiffness, -damping)

```
CICM.setNullSpaceStiffness(10.0);
CICM.setNullSpaceDamping(0.7);
```