Lego Navigation

Darren Liccardo

Mark McKelvin

Outline

- Objectives and Motivation
- Nasty Limitations
- Giotto Structure
  - Ports, drivers, tasks
    - State estimator
    - Controllers
    - Comm
      - Controller mode change
  - Global state change (Giotto mode)
- High level control (asynchronous communication)
  - Java comm (UDP like packet delivery)
- Demo

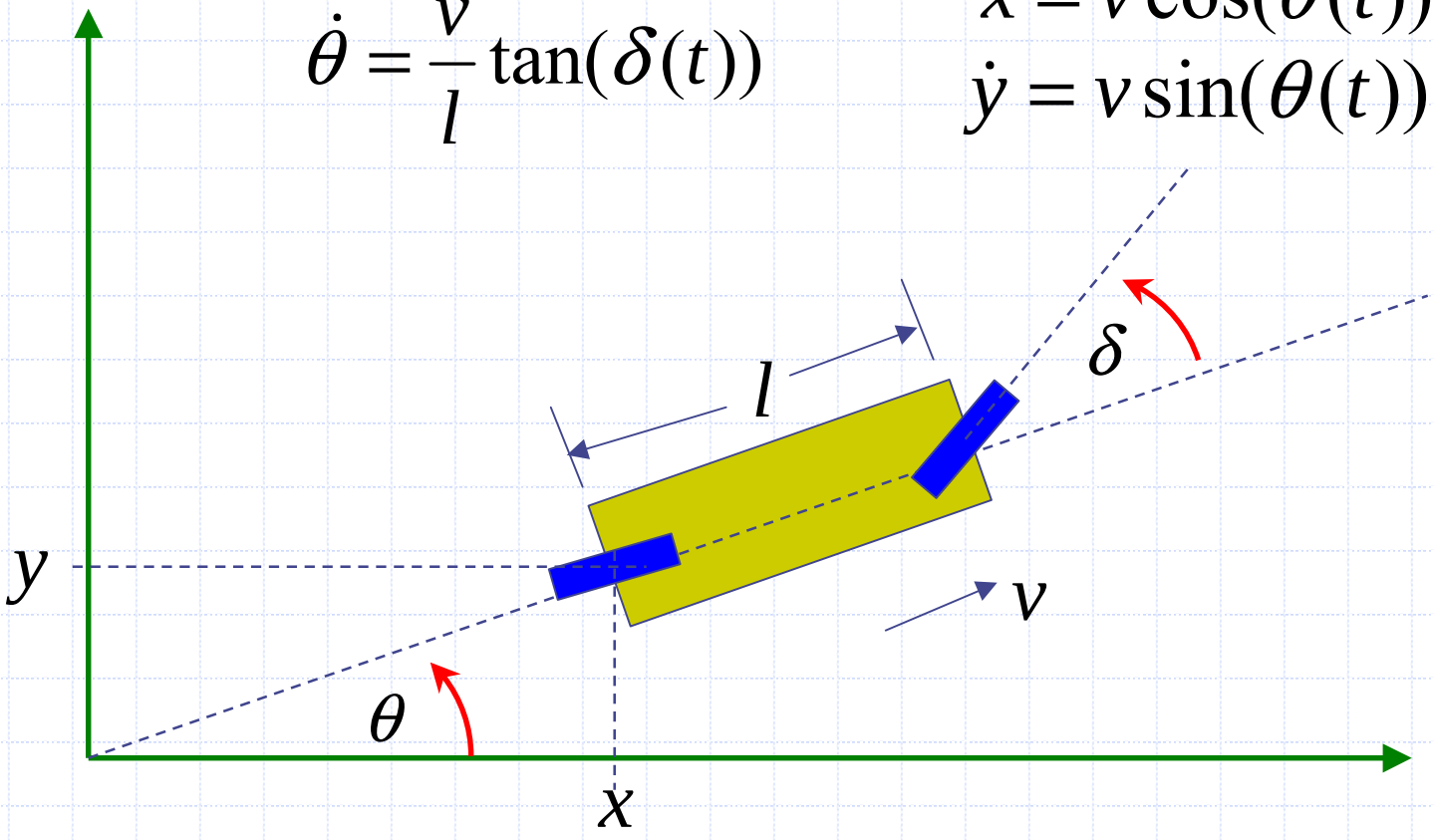
Motivation and Objectives

- Objectives:
  - Desire robot to navigate to arbitrary coordinates
  - Create logical platform abstraction
- Motivation:
  - Allows maximum flexibility in path planning
  - Allows (fast) straight line return paths
  - Manage complexity

Bicycle Steering Model

$$\dot{\theta} = \frac{v}{l}\tan(\delta(t))$$

$$\dot{x} = v\cos(\theta(t))$$
$$\dot{y} = v\sin(\theta(t))$$



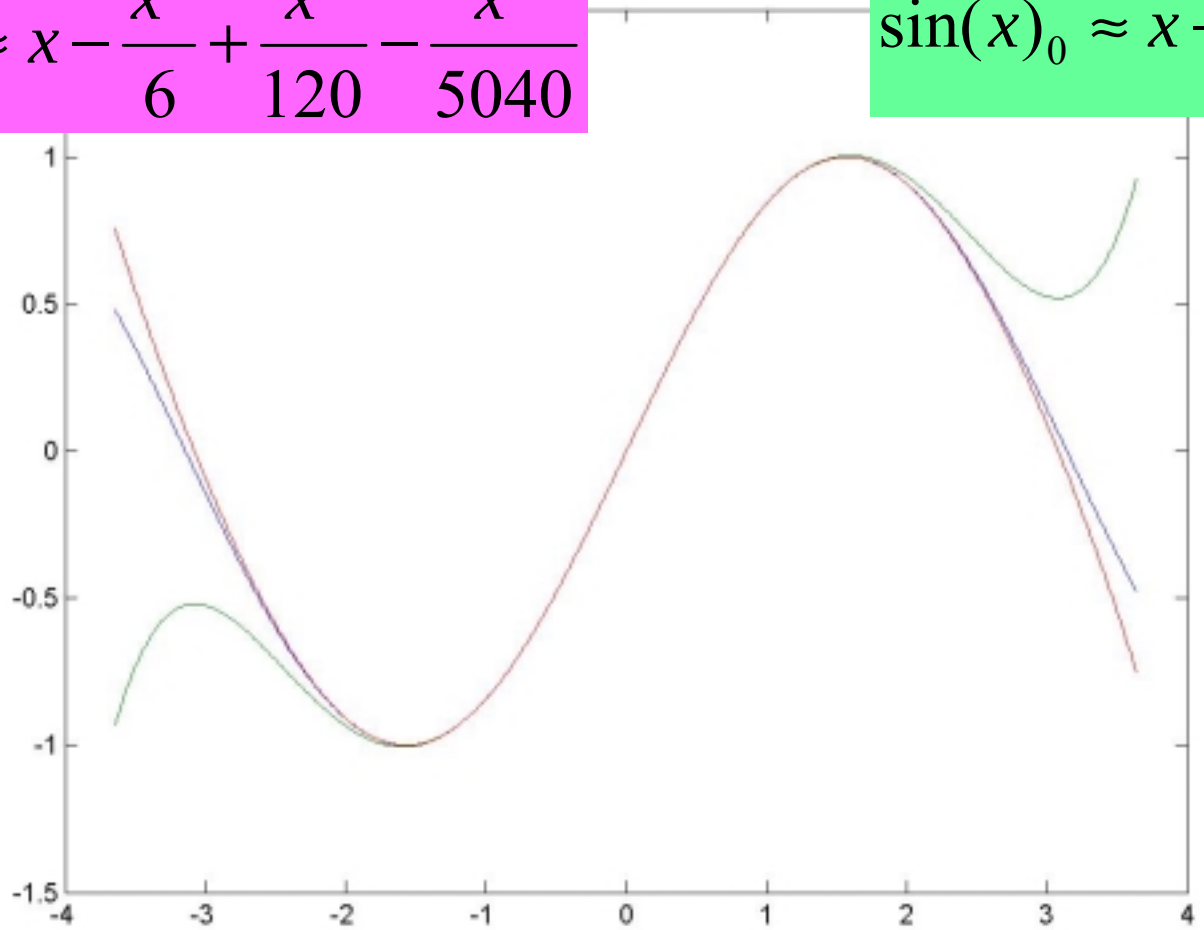
Constrained Platform

- Need trig functions for state estimation
- Constraints:
  - 16-bit registers
  - No floating point unit
  - Low clock speed
- Use software floating point to calculate (low-order) Taylor series expansions

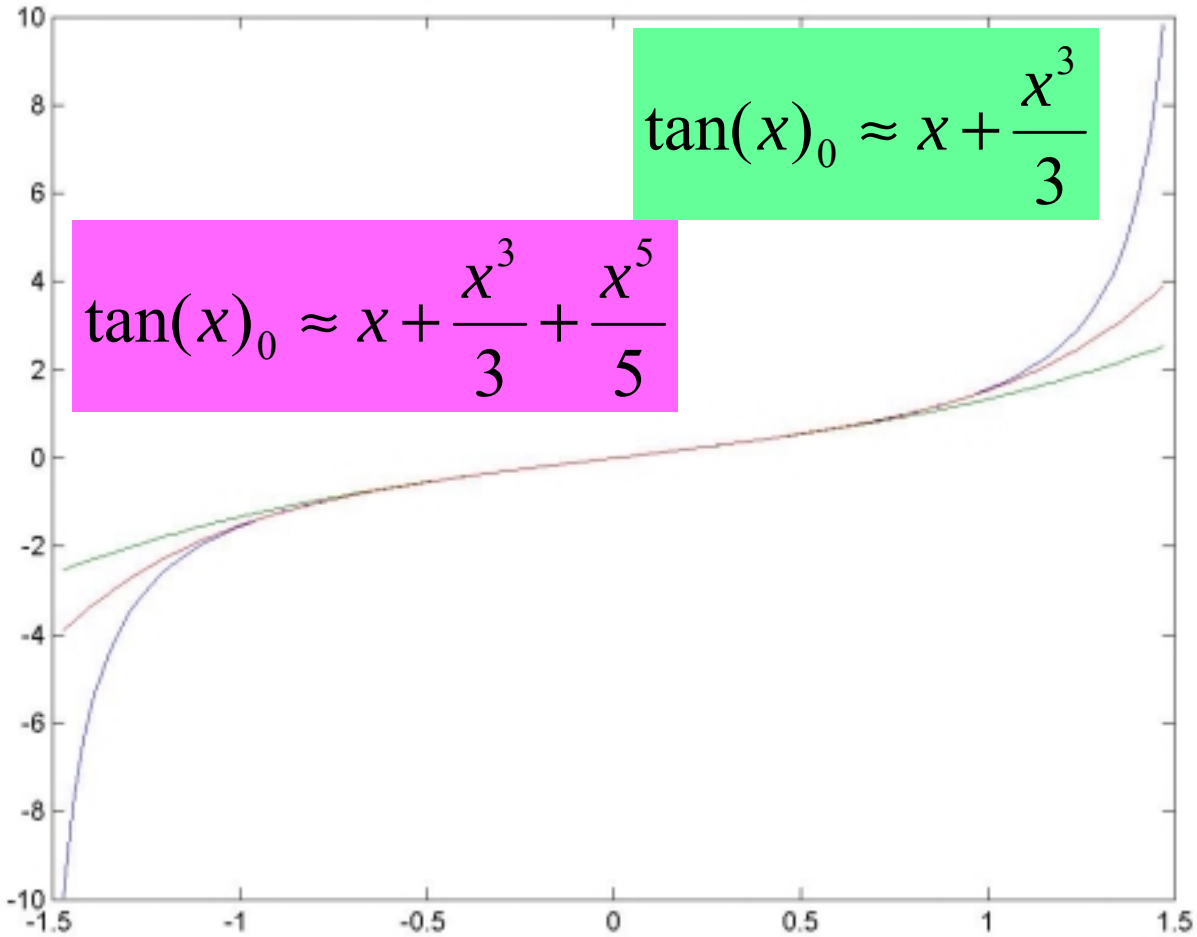
Taylor Series: sin

$$\sin(x)_0 \approx x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040}$$

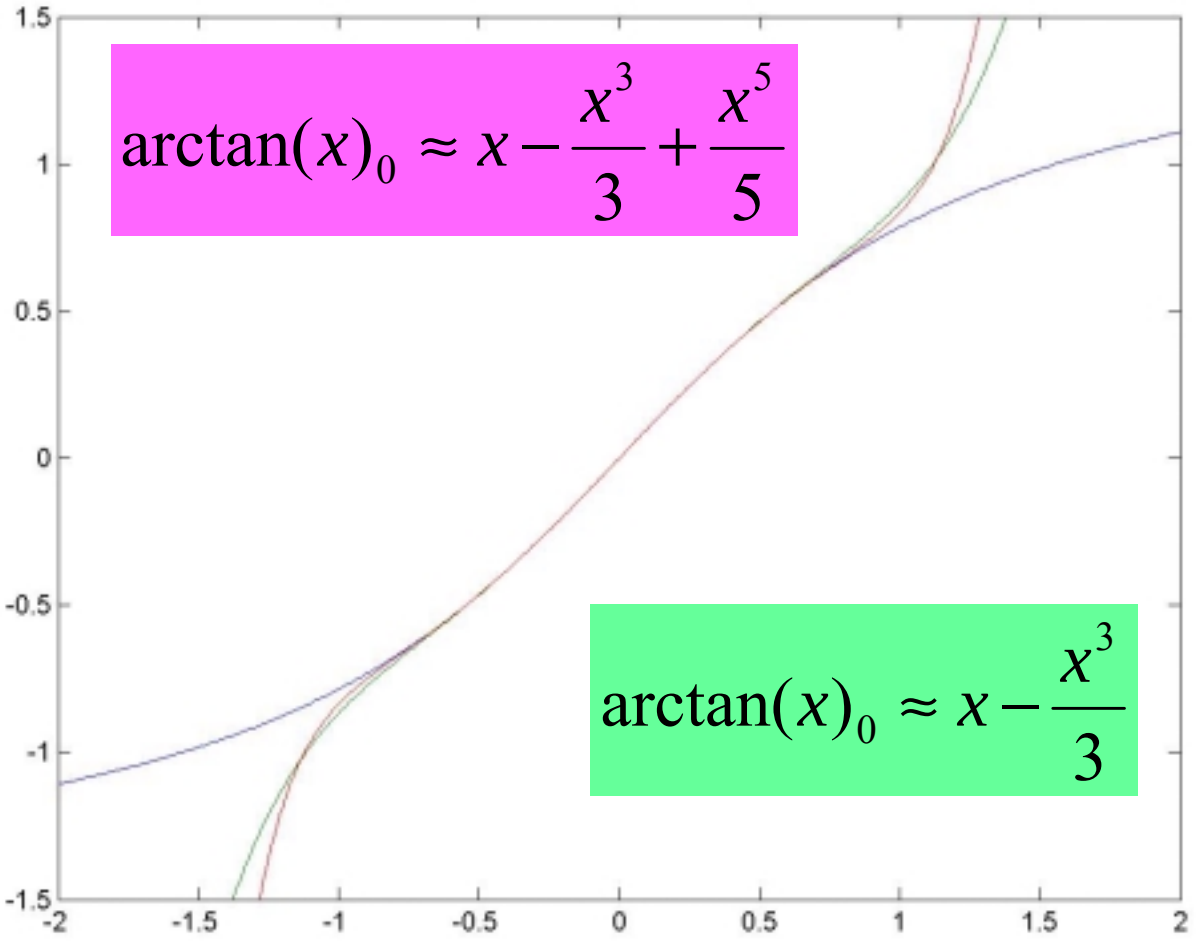
$$\sin(x)_0 \approx x - \frac{x^3}{6} + \frac{x^5}{120}$$



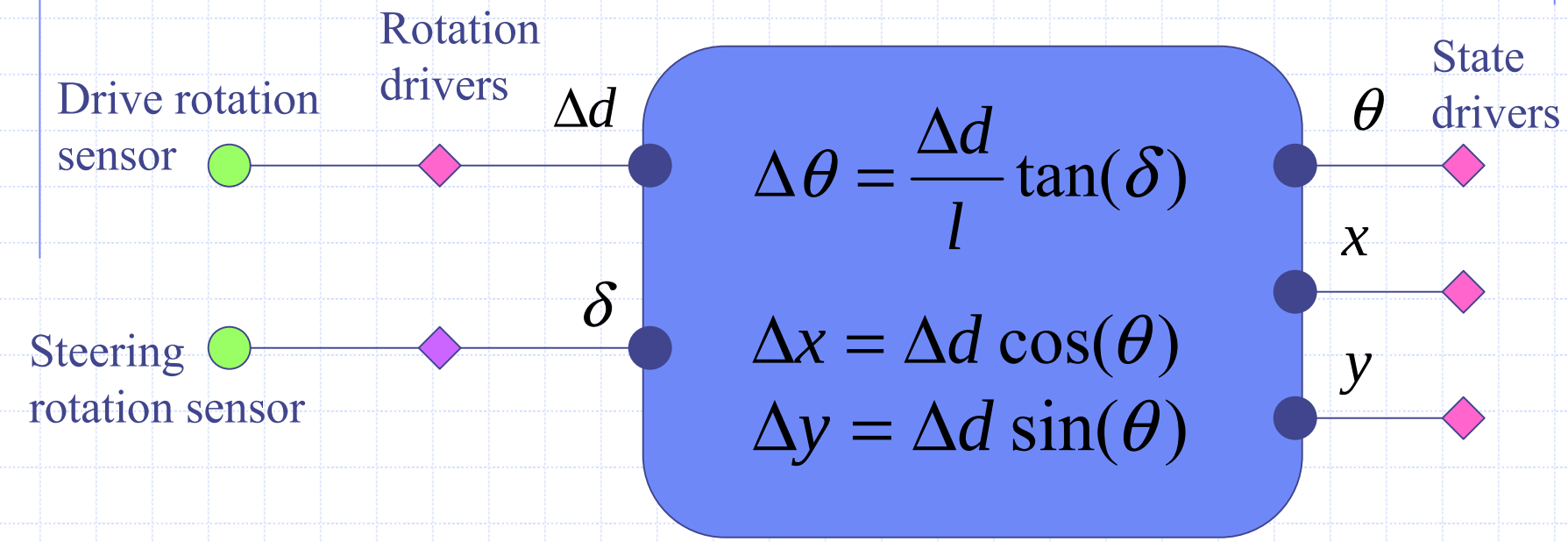
Taylor Series: tan



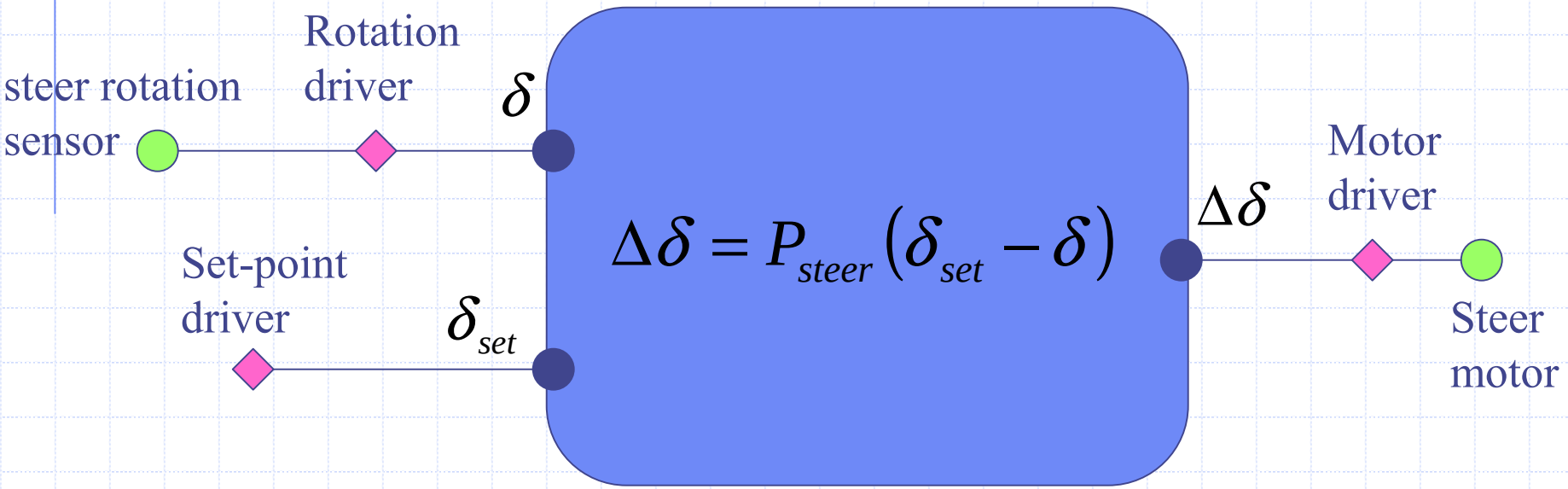
Taylor Series: arctan



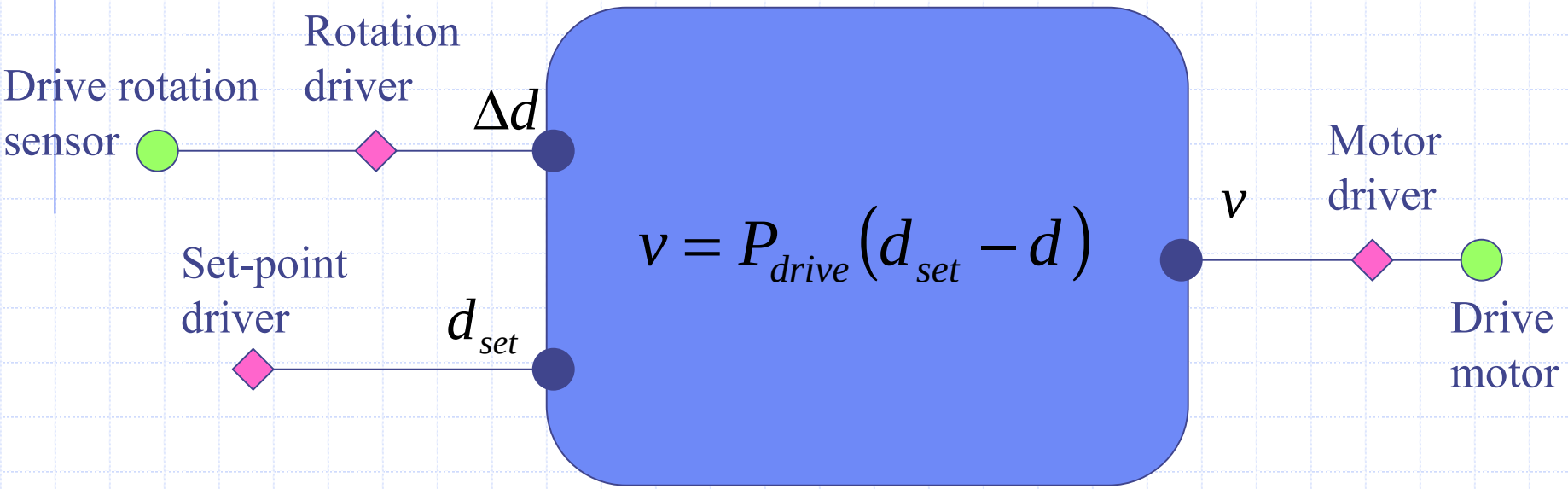
Giotto Task: State Estimator



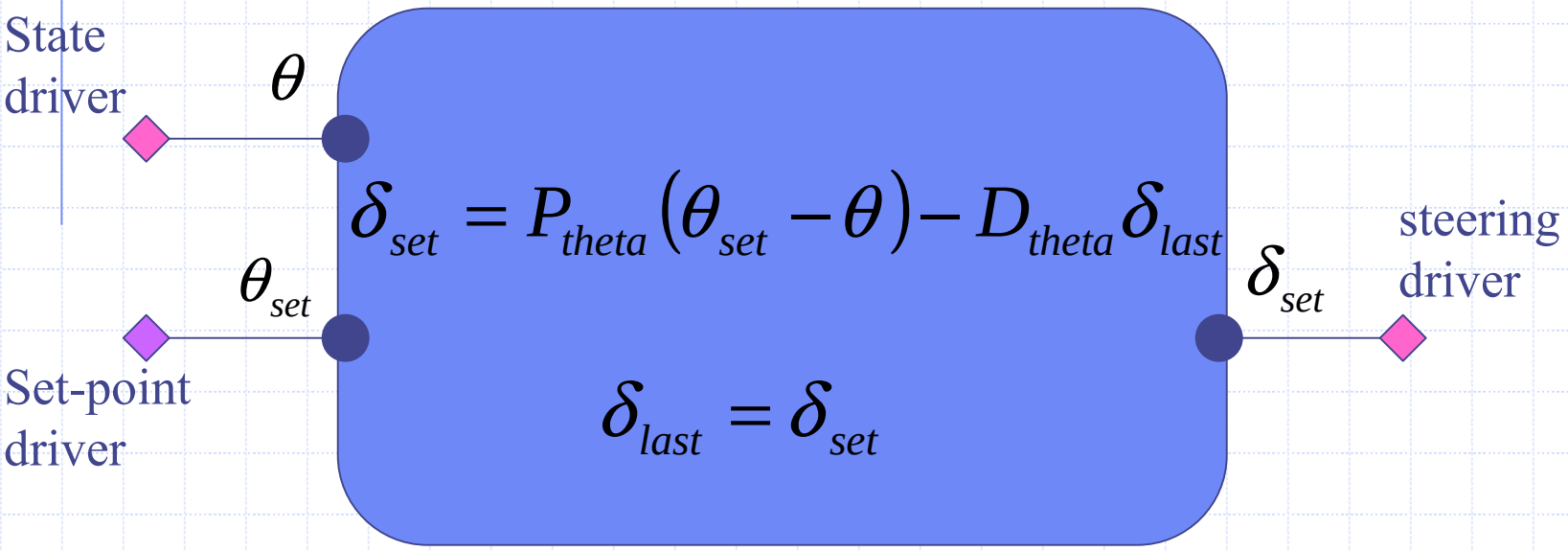
Giotto Task: Steering Controller



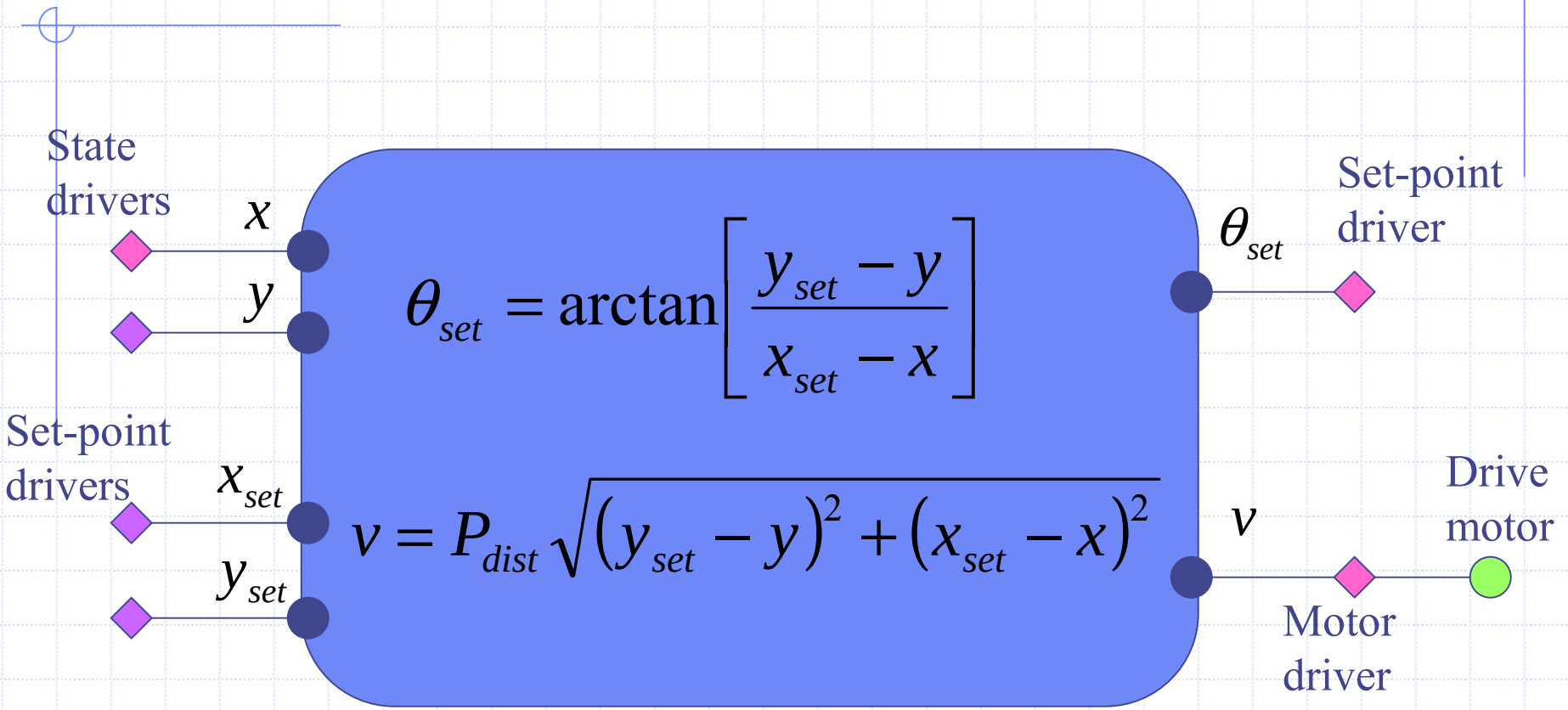
Giotto Task: Distance Controller



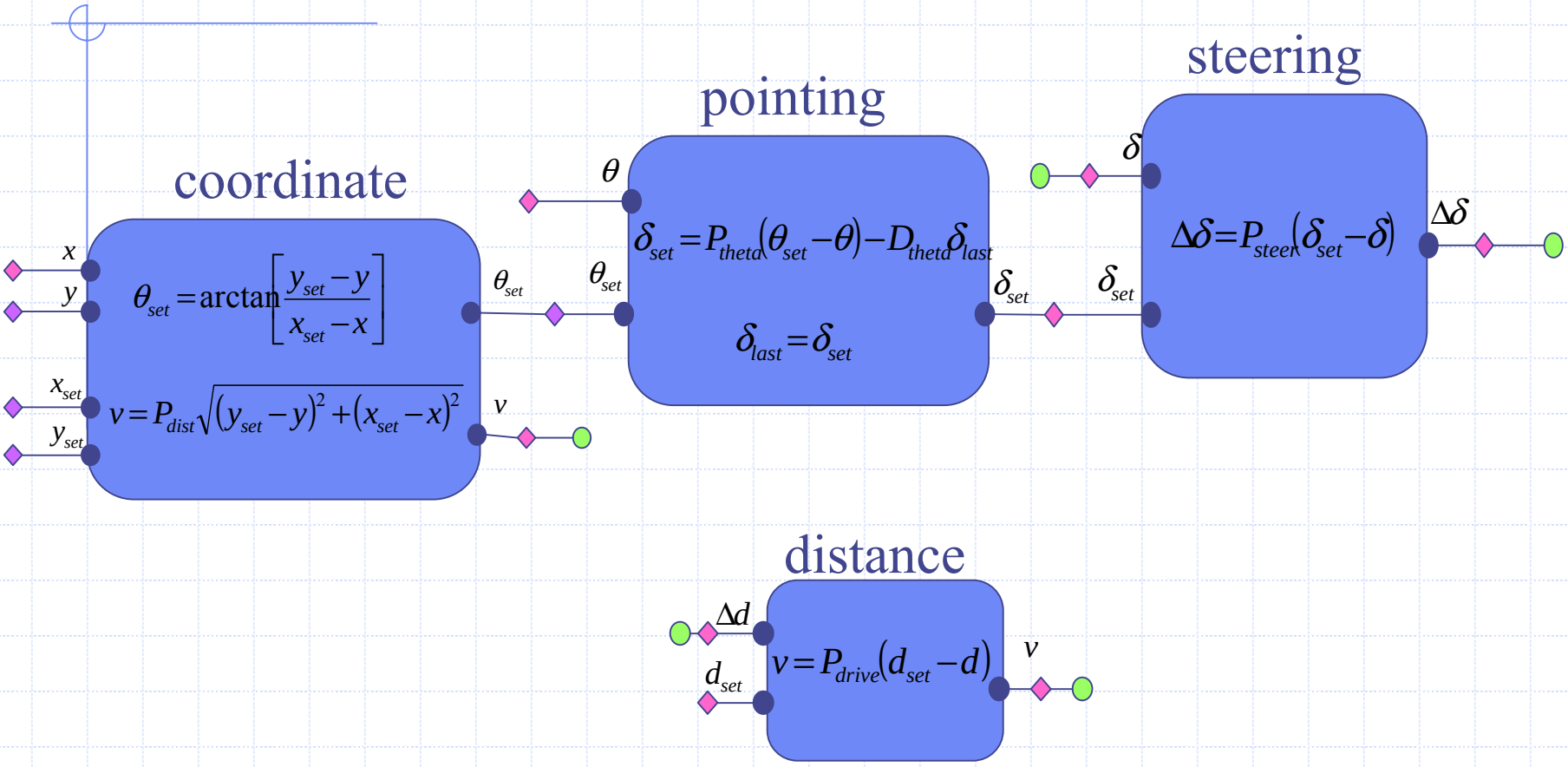
Giotto Task: Pointing Controller



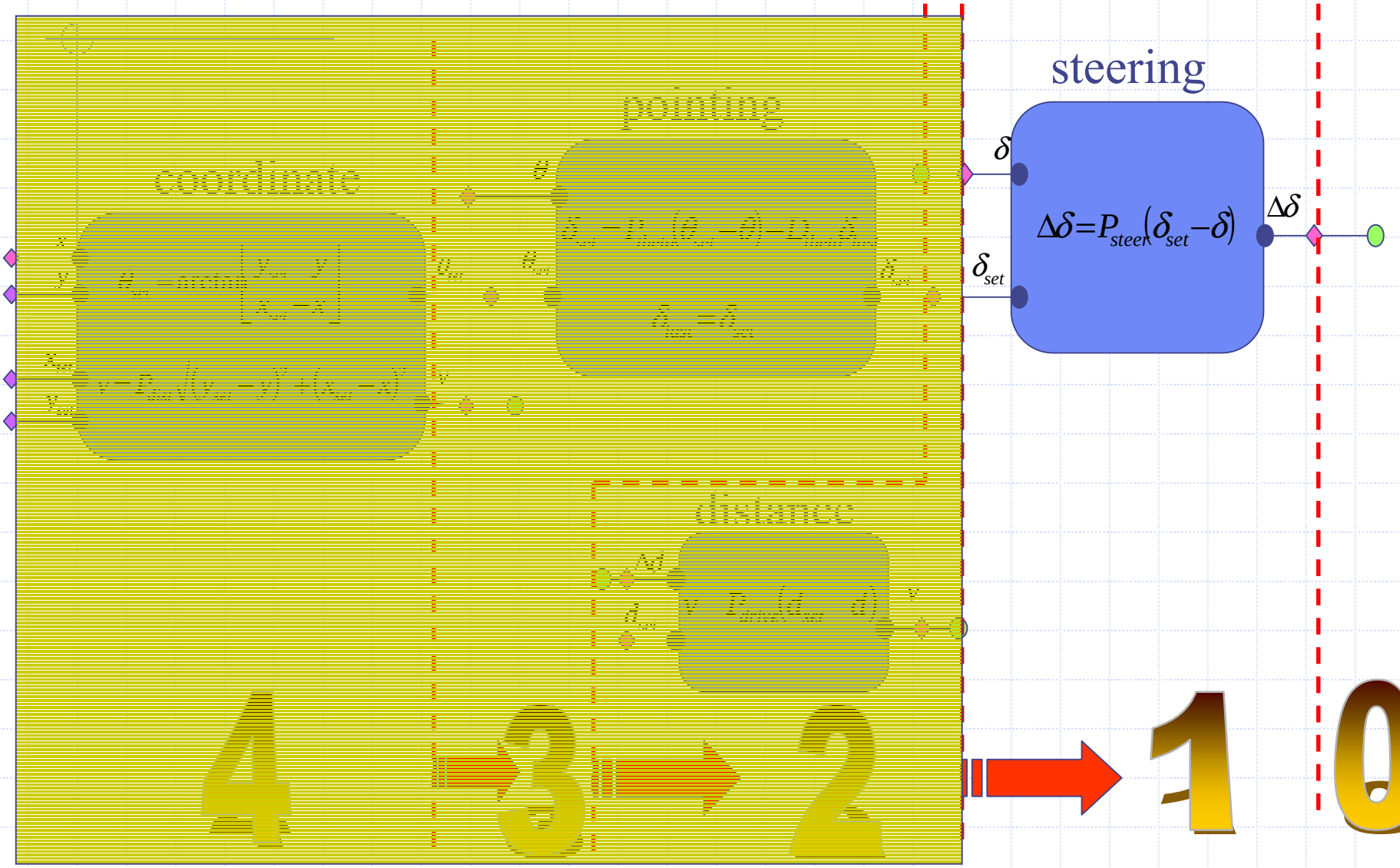
Giotto Task: Coordinate Controller



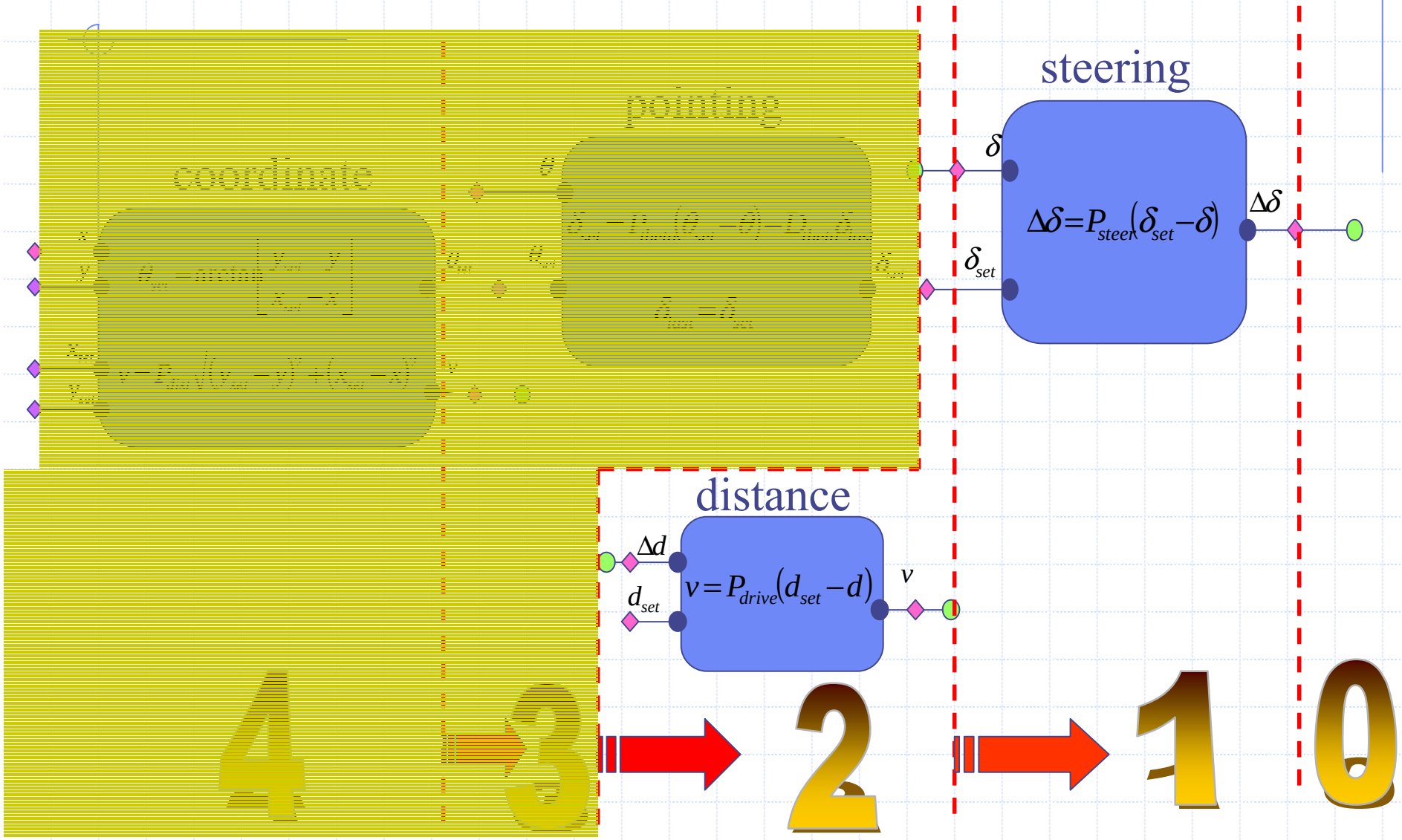
Giotto Control Overview



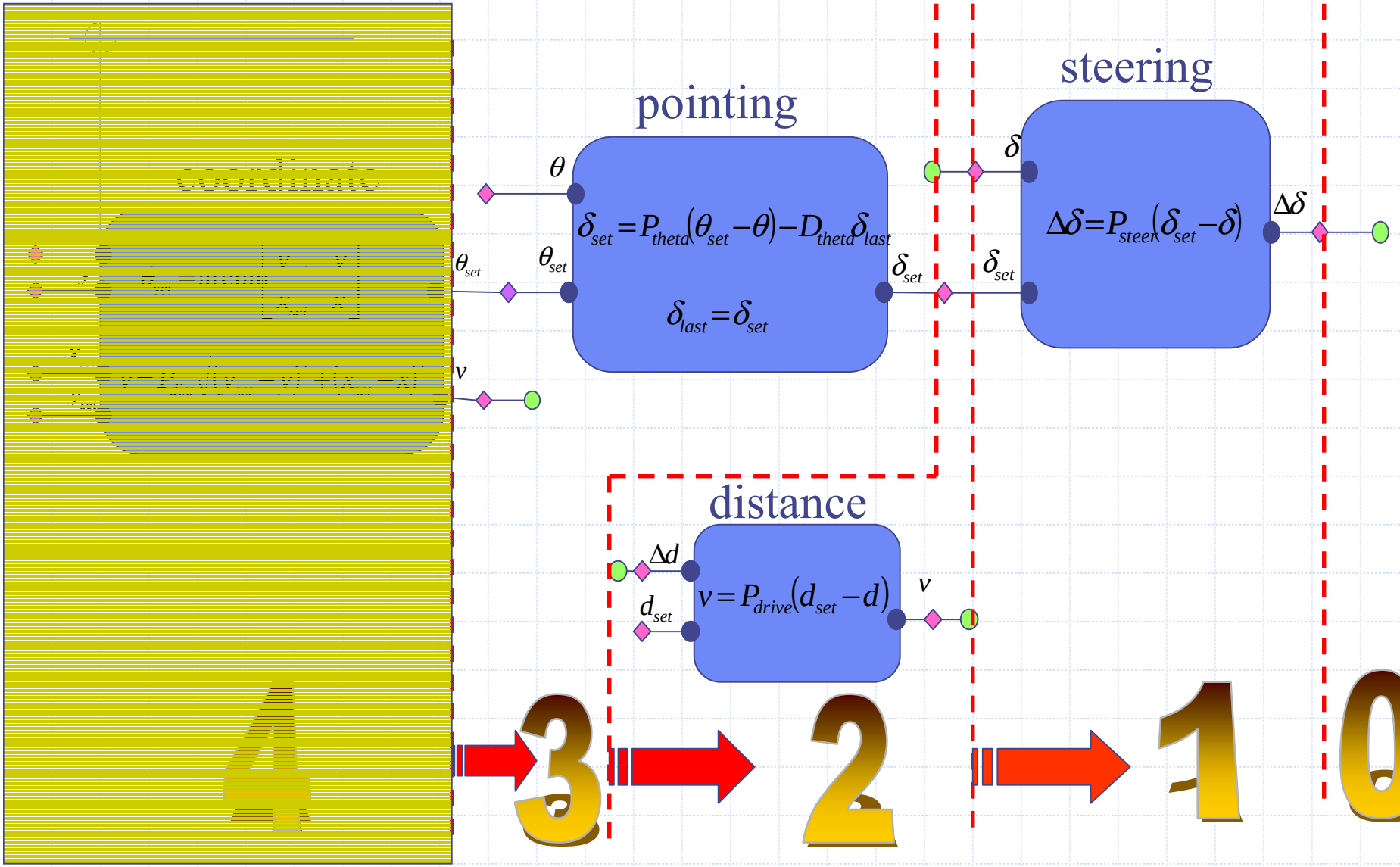
Giotto Control Overview



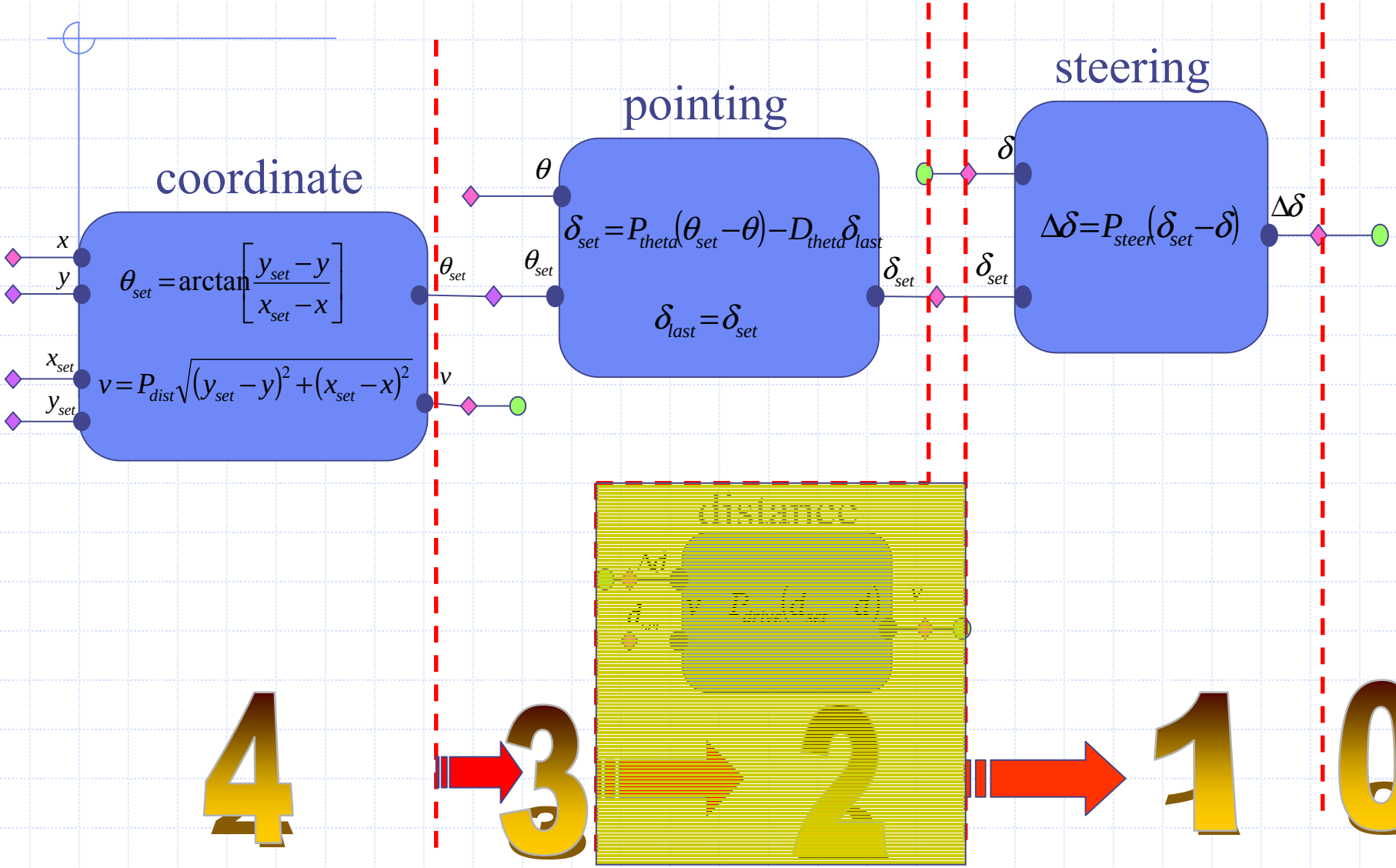
Giotto Control Overview



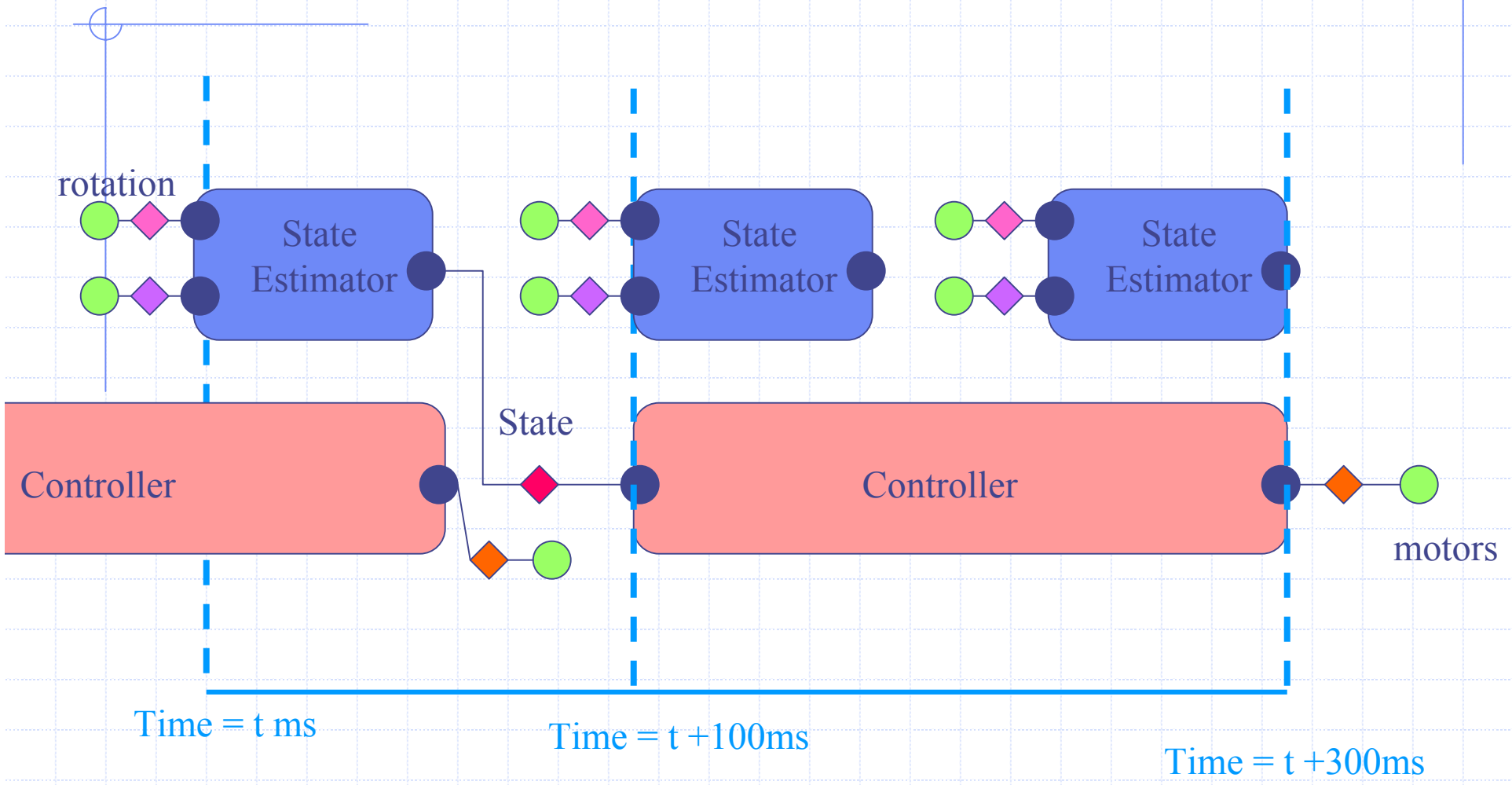
Giotto Control Overview



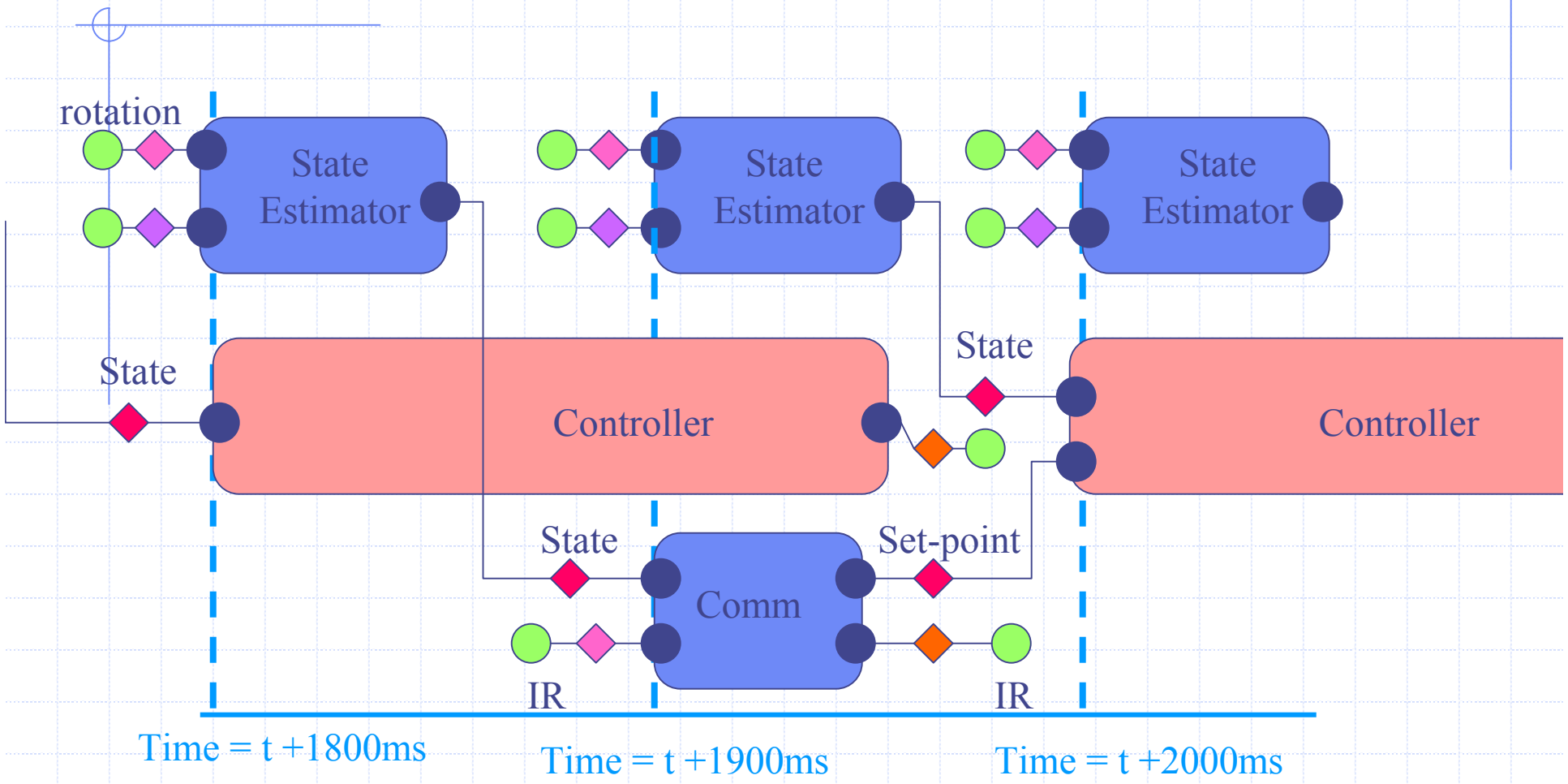
Giotto Control Overview



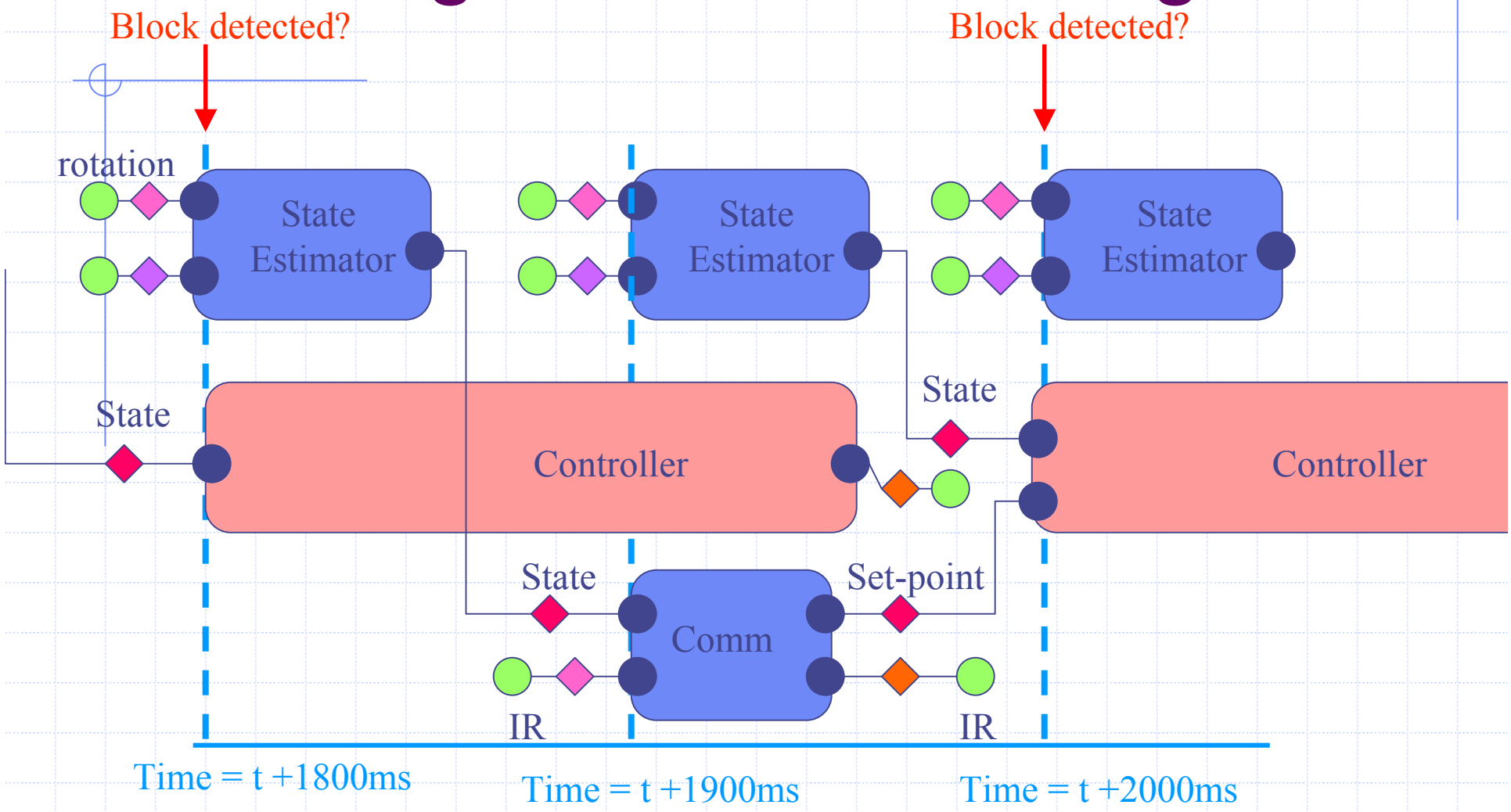
Giotto Program:Navigation Mode



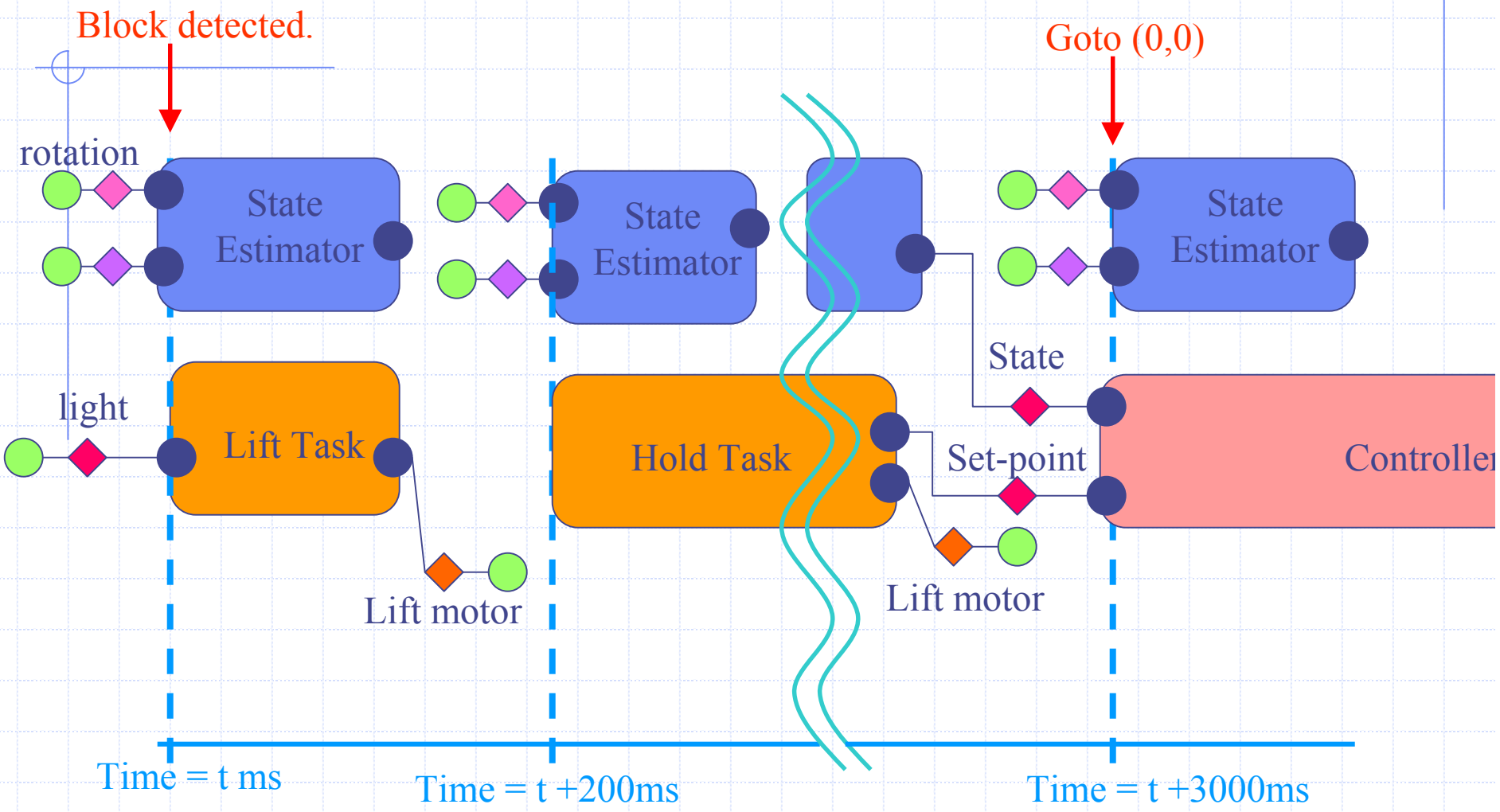
Giotto Program: Navigation Mode



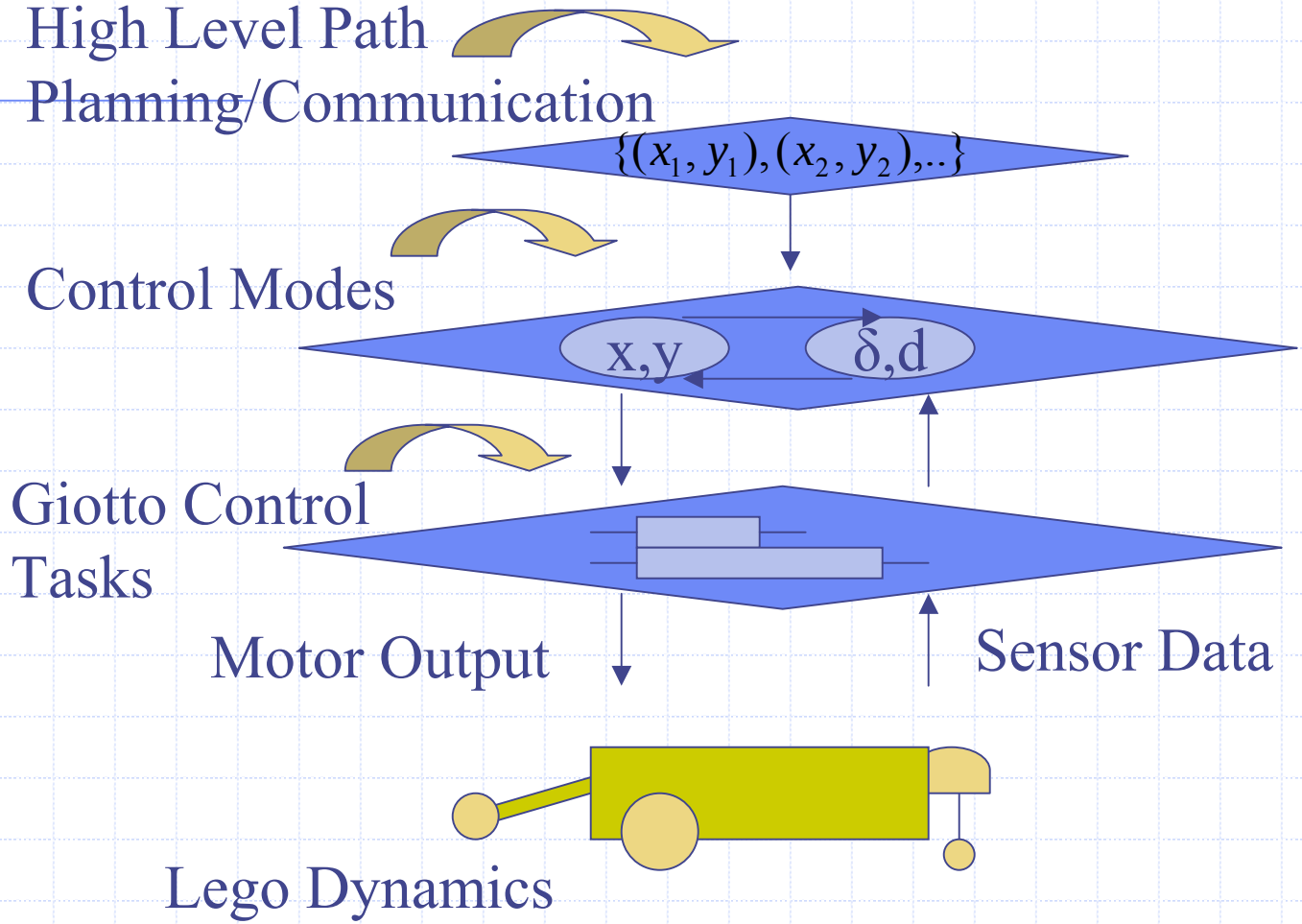
Giotto Program: Mode Change



Giotto Program: Pickup Mode

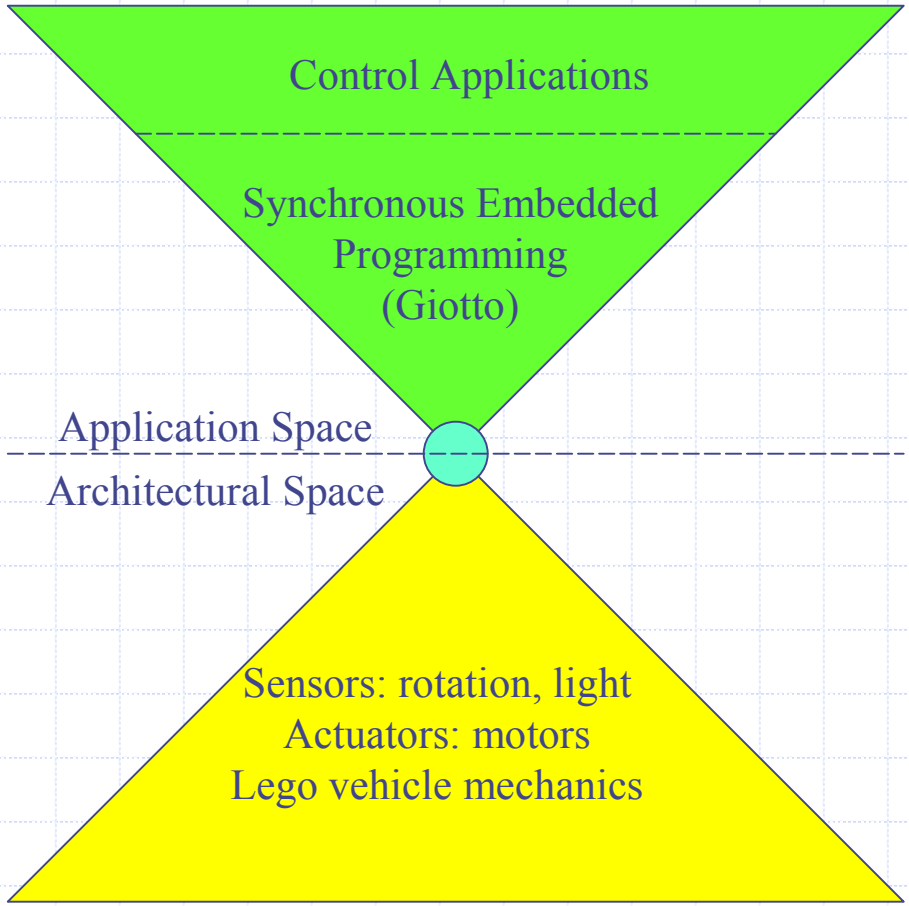


Nested Platforms



Platform Based Design with Giotto

- Goal
  - Abstract details of sensors, actuators, and vehicle hardware from control applications
- How?
  - Platform
  - Synchronous Embedded Programming Language (i.e. Giotto)



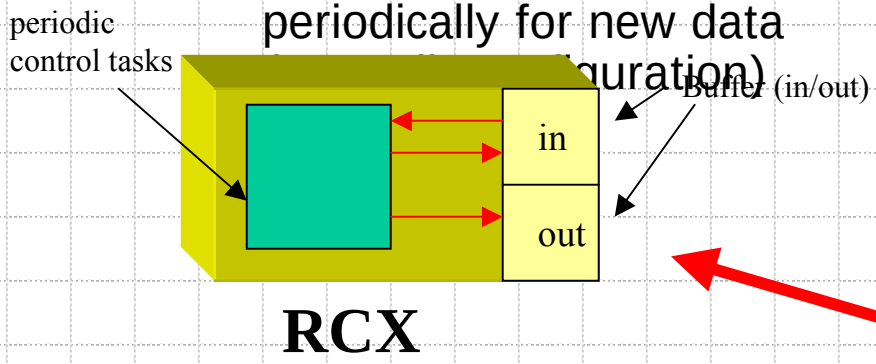
Network Communication

- Usages
  - Debugging
  - Real-time control
- Characteristics
  - Similar to unicast UDP messages
    - Uses logical layer to “address” messages to specific hosts and ports
    - no guarantee on packet delivery
    - packets received are uncorrupted

Network Communication

RCX Implementation

- C (using LegOS functions*)
- Sends periodic messages
- Received data is stored in a buffer
- Buffer is checked periodically for new data
  [illegible]guration)



PC Implementation

- Java (using JavaLNP**)
- Uses *LNPManager* class to handle LNP on the PC
  - Creates a thread to keep IR tower awake
  - Uses *LNPListener* interface to receive messages
- Sends aperiodic messages
- Immediately posts received data

**** Markus Noga's (www.noga.de/legOS/) LegOS***

***** Fabien Bergeret (fabien@bergeret.com) implementation of LNP using Java***

Conclusions

- Advantages
  - Low overhead as compared to a TCP implementation
  - User controls robot
  - Great for debugging!!
  - Can be used in LegOS Mindstorm games
- Possible problems with communications:
  - Time critical tasks
    - Buffering received data
    - Guaranteed packet arrival
- Improvements
  - Protocol implementation
    - Assuring packets arrive

Conclusions

- Used Developed Methodology
  - Platform-based design
    - Provides appropriate layers of abstraction
    - Eases Software Reuse
    - Eases Hardware Modifications
  - Time-based control
    - Verifiable real-time constraints
    - Eases controller modifications

Demo

- Control Modes via Communications Channel
- Auto Route with block return