Course of

Robot Programming with ROS 2

Day 1

3. Node Programming I

ikerlan

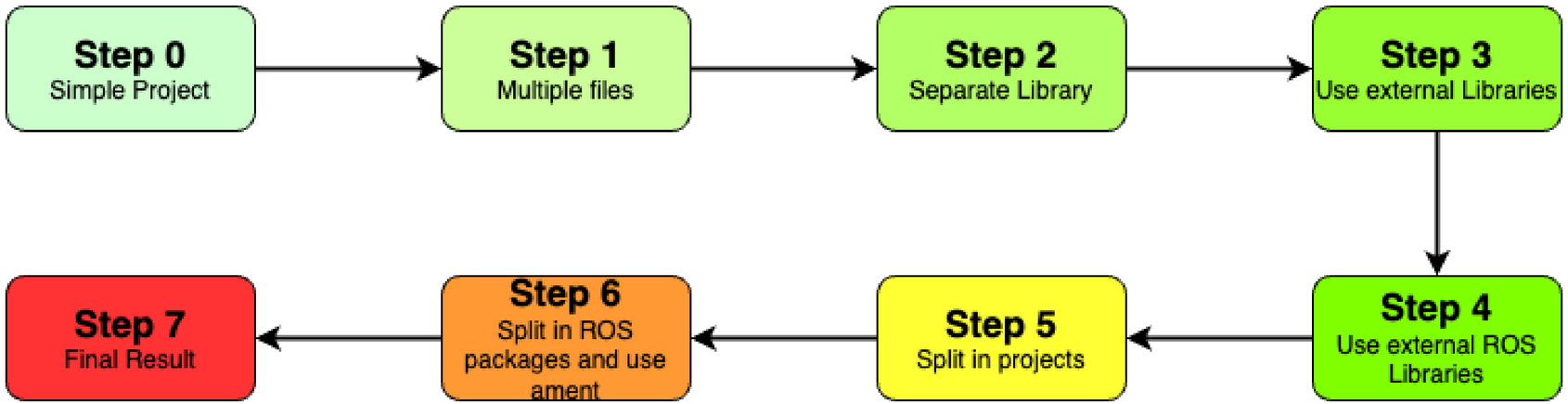
Universidad Rey Juan Carlos

Intelligent Robotics Lab

CMAKE

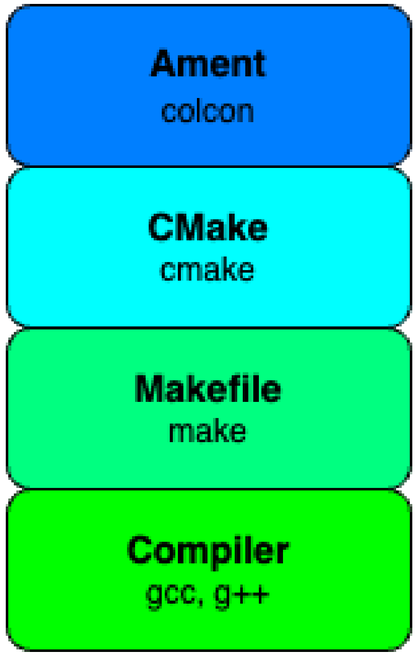
- Works on top of Makefile
- Make easier to manage projects
- Standard in Open Source development (and beyond)

- Let's walk into the examples



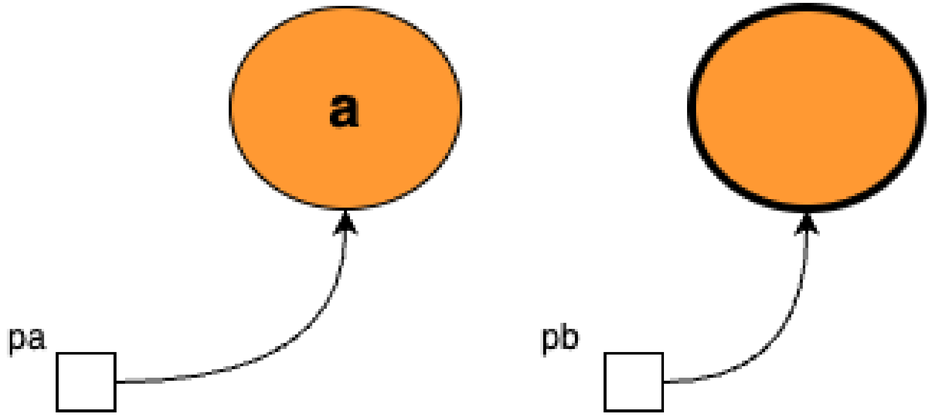
CMAKE

- Works on top of Makefile
- Make easier to manage projects
- Standard in Open Source development (and beyond)



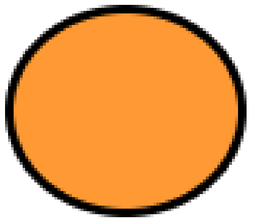
Smart Pointers

```
{
  Rectangle a;
  Rectangle * pa = &a;

  Rectangle * pb = new Reactangle();

}
```



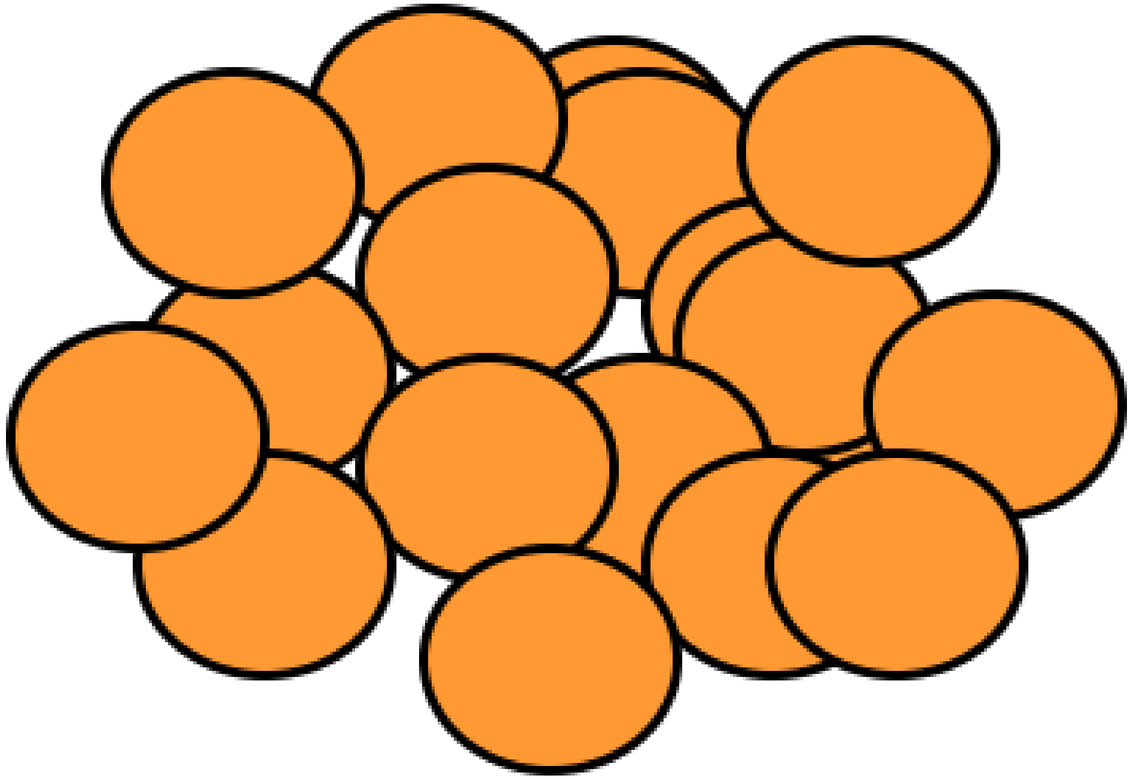
Smart Pointers

```
{
  Rectangle a;
  Rectangle * pa = &a;

  Rectangle * pb = new Reactangle();

}
```



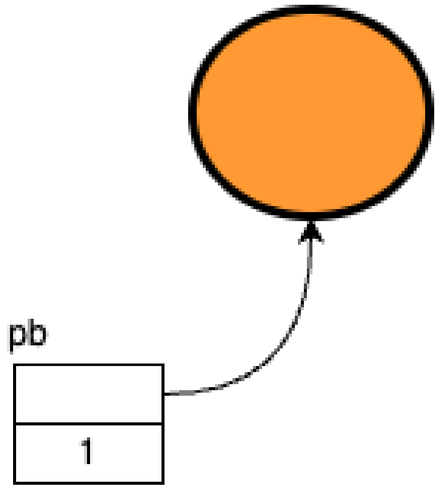
Smart Pointers

```
for (int i = 0; i < 100; i++) {
  Rectangle a;
  Rectangle * pa = &a;

  Rectangle * pb = new Reactangle();

}
```



Smart Pointers

```
{
    std::shared_ptr<Rectangle> pb = std::shared_ptr<Rectangle>(new Reactangle());

}
```



Smart Pointers

```
{
    std::shared_ptr<Rectangle> pb = std::shared_ptr<Rectangle>(new Reactangle());

}
```

Smart Pointers

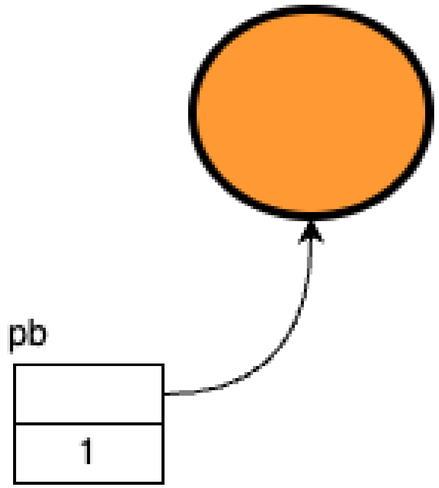
```
{
    std::shared_ptr<Rectangle> pb = std::shared_ptr<Rectangle>(new Reactangle());

}
```

```
{
    auto pb = std::shared_ptr<Rectangle>(new Reactangle());
}
```

```
{
    auto pb = std::make_shared<Rectangle>();
}
```

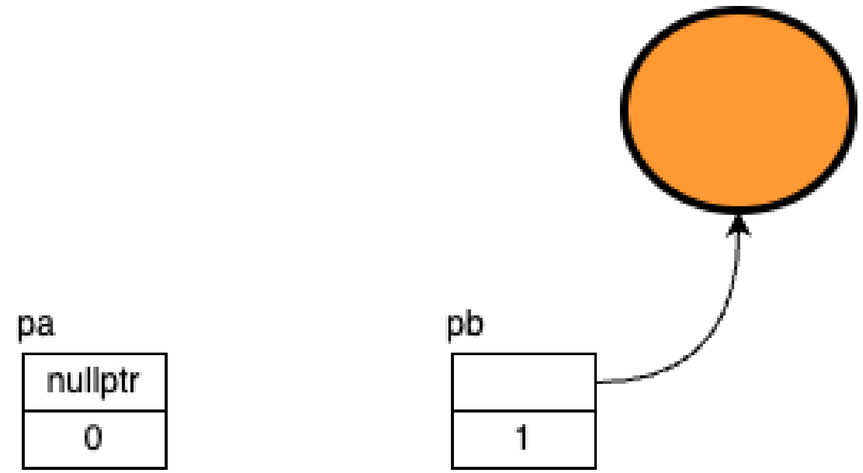
Smart Pointers

```
{
    std::shared_ptr<Rectangle> pb = std::shared_ptr<Rectangle>(new Reactangle());

}
```



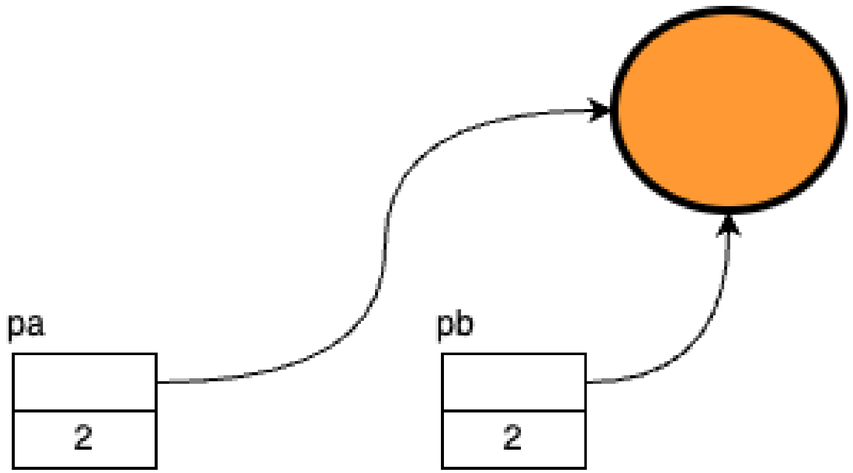
Smart Pointers

```
{
  std::shared_ptr<Rectangle> pa;

  {
    auto pb = std::make_shared<Rectangle>();

    pa = pb;

  }

}
```



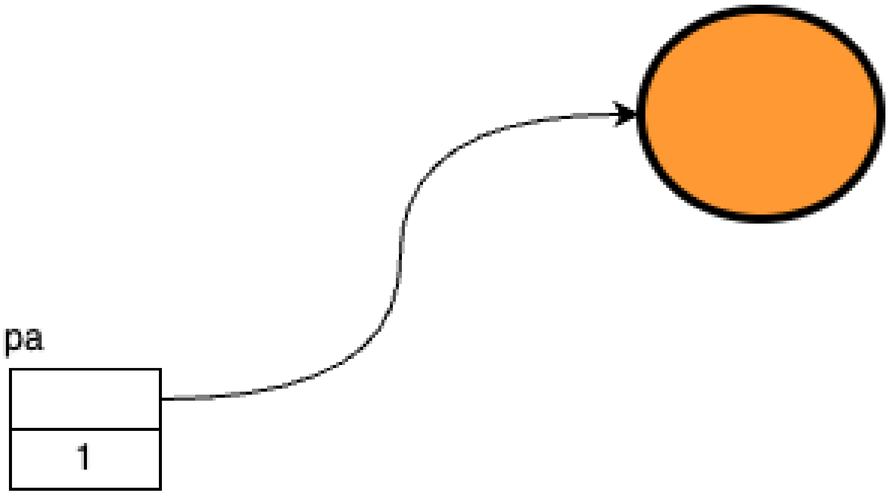
Smart Pointers

```
{
  std::shared_ptr<Rectangle> pa;

  {
    auto pb = std::make_shared<Rectangle>();

    pa = pb;

  }

}
```



Smart Pointers

```
{
  std::shared_ptr<Rectangle> pa;

  {
    auto pb = std::make_shared<Rectangle>();

    pa = pb;

  }

}
```

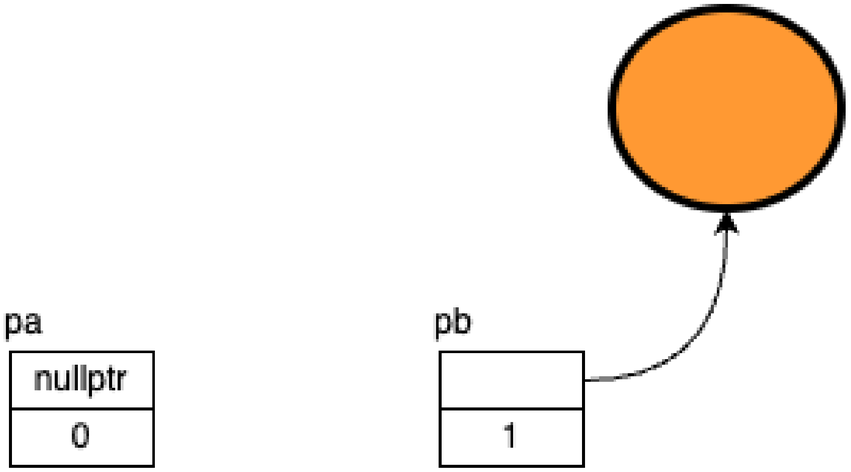


Smart Pointers

```
{
  std::shared_ptr<Rectangle> pa;

  {
    auto pb = std::make_shared<Rectangle>();

    pa = pb;

  }

}
```

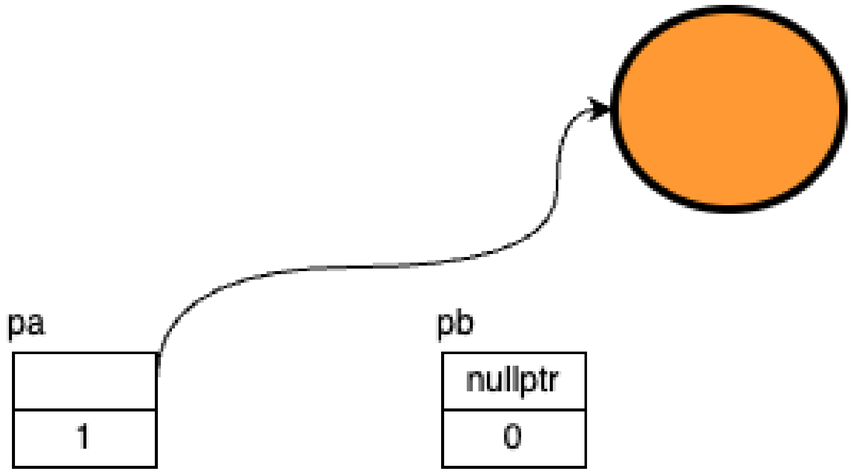
Smart Pointers

```
{
  std::unique_ptr<Rectangle> pa;

  {
    auto pb = std::make_unique<Rectangle>();

    pa = std::move(pb);

  }

}
```



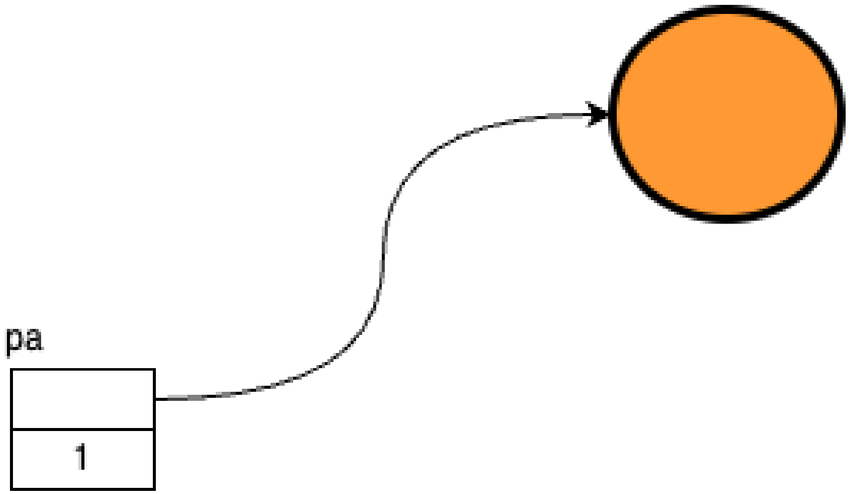
Smart Pointers

```
{
  std::unique_ptr<Rectangle> pa;

  {
    auto pb = std::make_unique<Rectangle>();

    pa = std::move(pb);

  }

}
```



Smart Pointers

```
{
  std::unique_ptr<Rectangle> pa;

  {
    auto pb = std::make_unique<Rectangle>();

    pa = std::move(pb);

  }

}
```

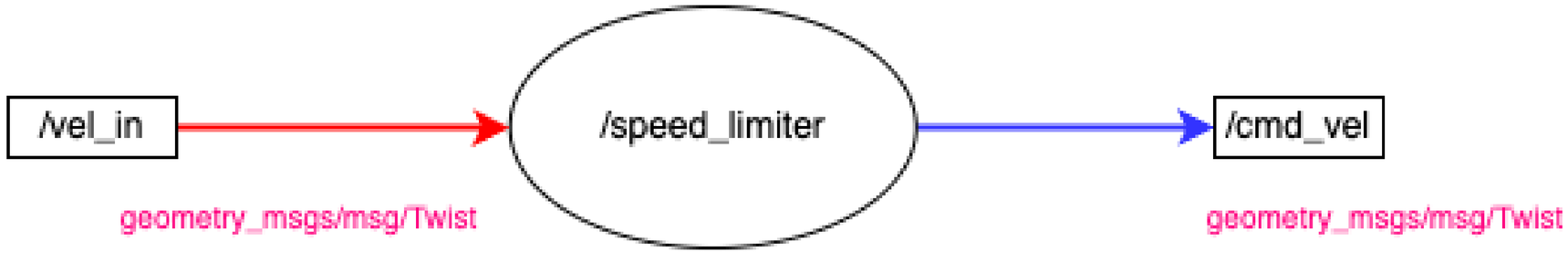


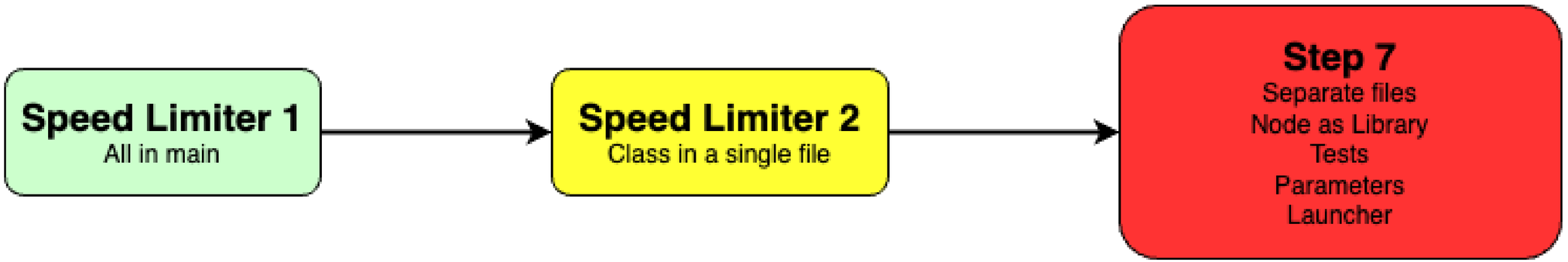
Smart Pointers

```
{
  std::unique_ptr<Rectangle> pa;

  {
    auto pb = std::make_unique<Rectangle>();

    pa = std::move(pb);

  }

}
```

Node Programming

Speed limiter

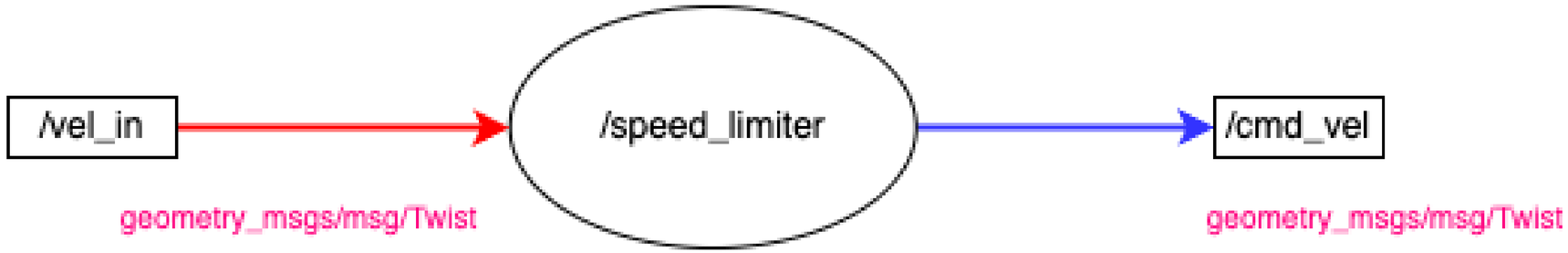


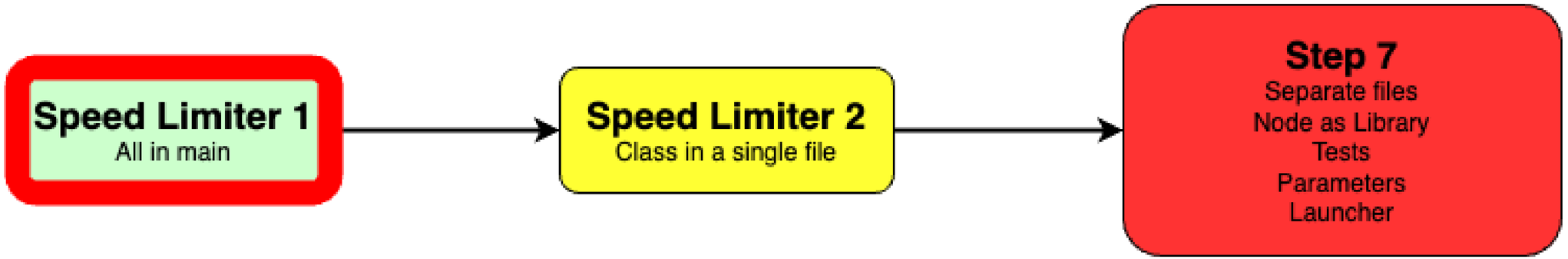


Intelligent Robotics Lab

Node Programming

Speed limiter

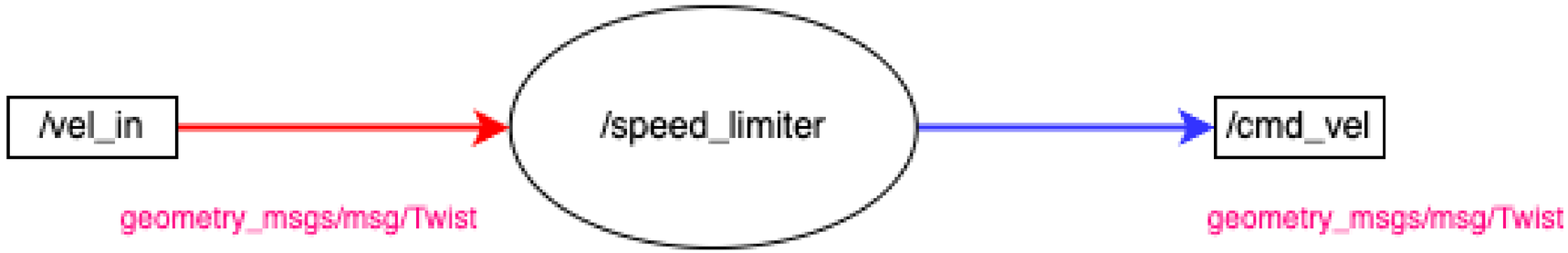


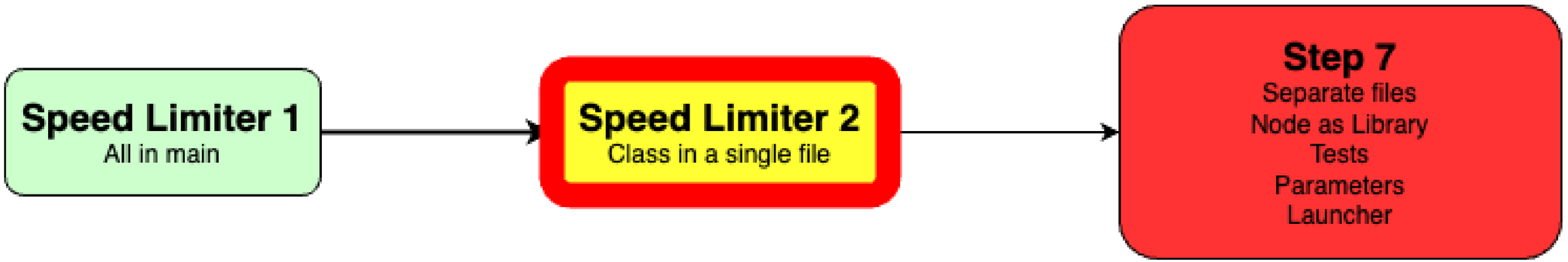


Intelligent Robotics Lab

Node Programming

Speed limiter

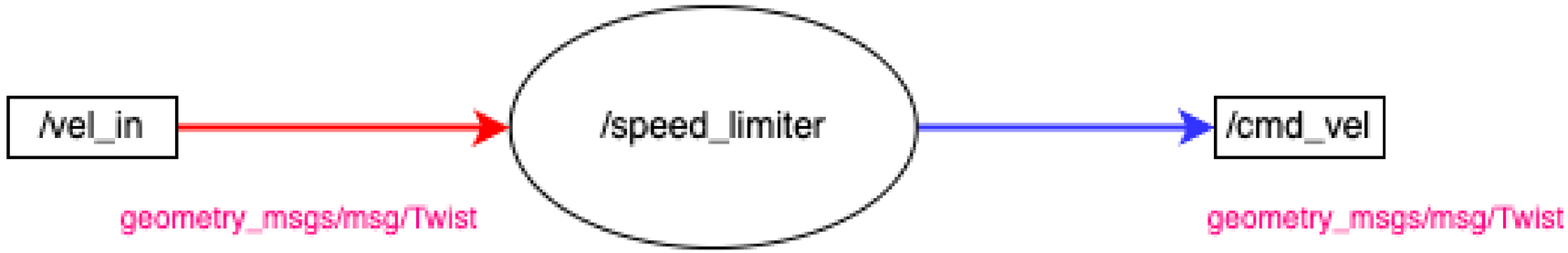


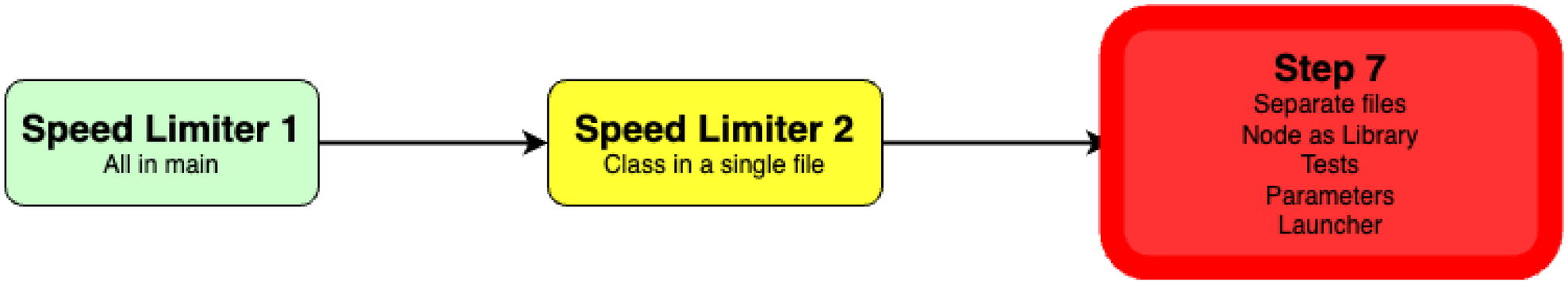


Intelligent Robotics Lab

Node Programming

Speed limiter





Intelligent Robotics Lab

Further topics

About QoS

Default	Reliable	Volatile	Keep Last
Services	Reliable	Volatile	Normal Queue
Sensor	Best Effort	Volatile	Small Queue
DParameters	Reliable	Volatile	Large Queue

```
publisher = node->create_publisher<std_msgs::msg::String>(
  "chatter", rclcpp::QoS(100).transient_local().best_effort());
```

```
publisher_ = create_publisher<sensor_msgs::msg::LaserScan>(
  "scan", rclcpp::SensorDataQoS().reliable());
```

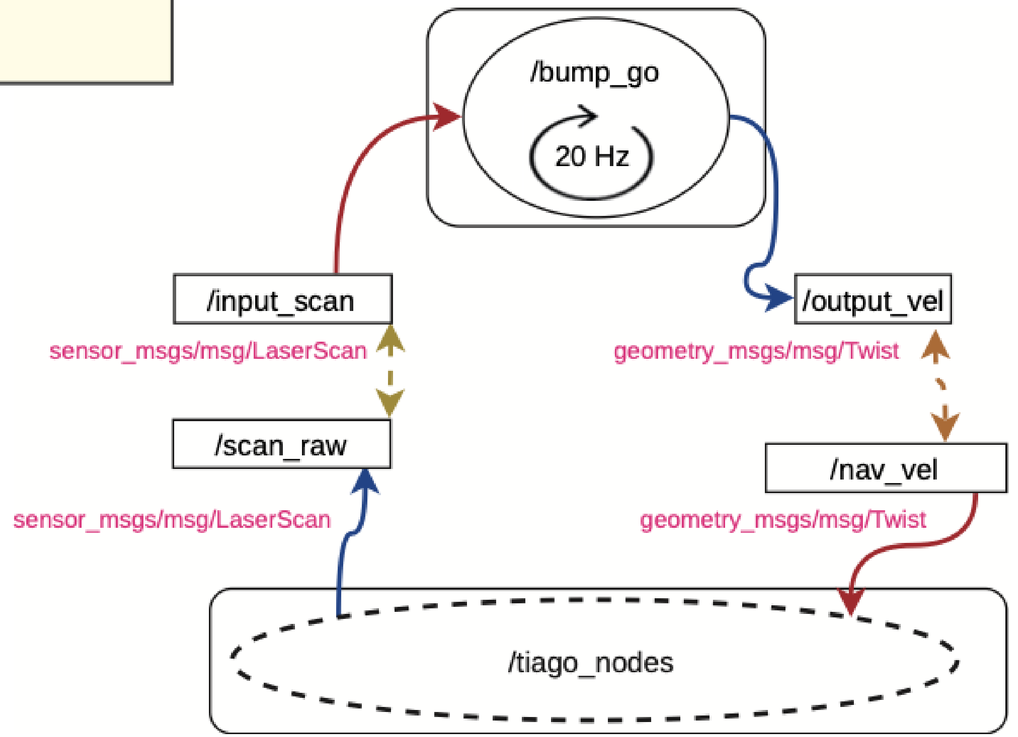
Compatibility of QoS **durability** profiles		**Subscriber**	
		Volatile	**Transient Local**
Publisher	**Volatile**	Volatile	**No Connection**
	Transient Local	Volatile	Transient Local

Compatibility of QoS **reliability** profiles		**Subscriber**	
		Best Effort	**Reliable**
Publisher	**Best Effort**	Best Effort	**No Connection**
	Reliable	Best Effort	Reliable

Further topics

About topic names and remaps

```
$ ros2 run br2_fsm_bumpgo_cpp bumpgo --ros-args -r output_vel:=/nav_vel -r
input_scan:=/scan_raw -p use_sim_time:=true
```



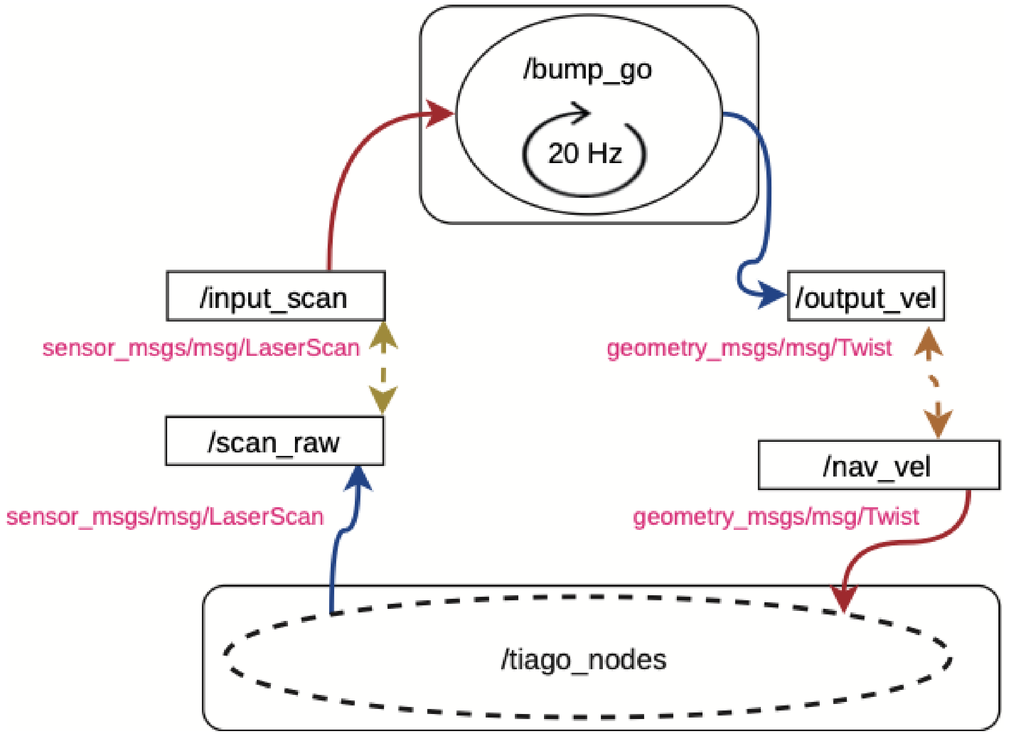
Intelligent Robotics Lab

Further topics

About topic names and remaps

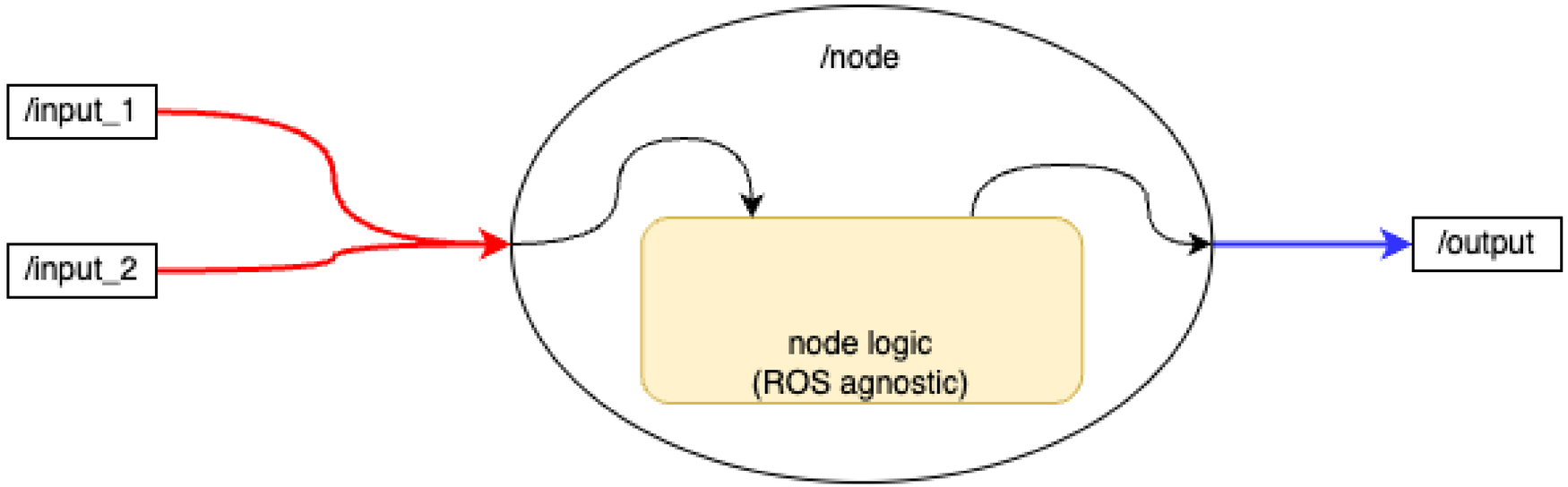
launch/bump_and_go.launch.py

```
bumpgo_cmd = Node(package='br2_fsm_bumpgo_cpp',
    executable='bumpgo',
    output='screen',
    parameters=[{
      'use_sim_time': True
    }],
    remappings=[
      ('input_scan', '/scan_raw'),
      ('output_vel', '/nav_vel')
    ])
```



Further topics

About logic separation



Further topics

About publication

Check your subscribers!!!

Don’t waste CPU time if nobody is interested in your result

```
void
ObstacleMonitorNode::control_cycle()
{
  if (marker_pub_->getNumSubscribers() == 0) {
    return;
  }

  geometry_msgs::msg::TransformStamped robot2obstacle;

  try {
    robot2obstacle = tf_buffer_.lookupTransform(
      "base_footprint", "detected_obstacle", tf2::TimePointZero);
```