All About Java™ Technology Based Robotics

Paul J. Perrone

Founder/CEO
Perrone Robotics
www.perronerobotics.com

Session TS-1519

Goal

Learn About Java™ Technology-Based Robotics

Describe and demonstrate robots of all shapes and sizes built 100% on Java platform!

Agenda

Why Robotics and Java Platform for Robotics?

Profiles, Shapes, and Sizes

Rat- and Cat-Sized Applications

Elephant-Sized Applications

Video

Benefits and Conclusions

Agenda

Why Robotics and Java Platform for Robotics?

Profiles, Shapes, and Sizes

Rat- and Cat-Sized Applications

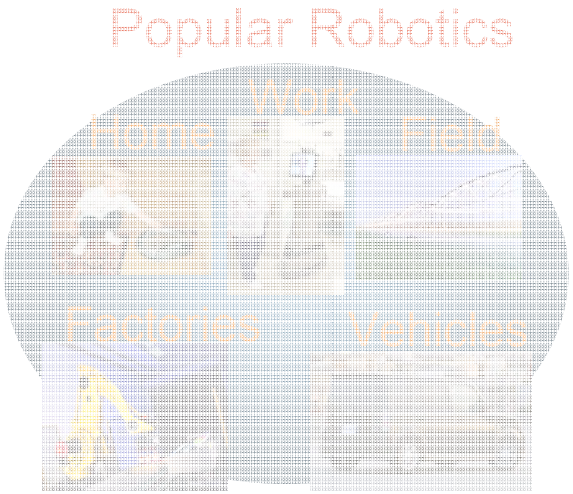
Elephant-Sized Applications

Video

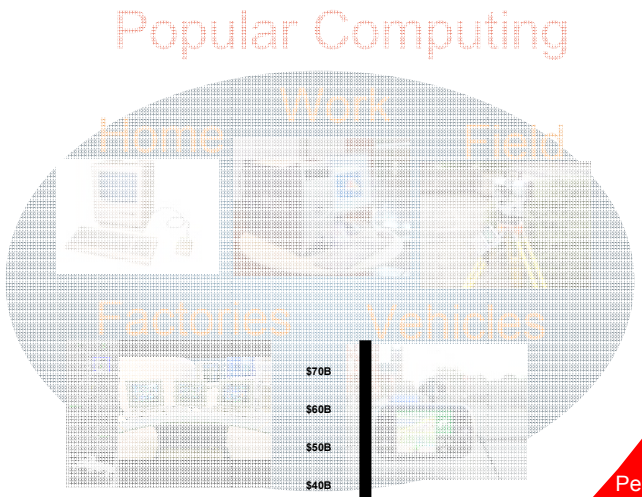
Benefits and Conclusions

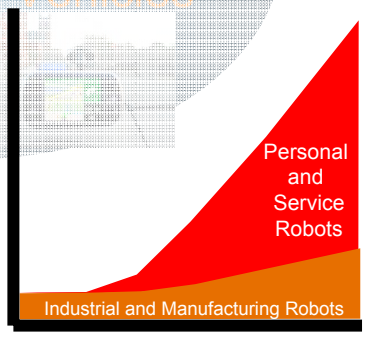
Why Focus on Robots Now?

Answer: Convergence = Reality



Robots Everywhere





1950s 1960s 1970s 1980s 1990s 21st Century

Why Java Platform for Robotics?

Answer: Simplicity, Support, and Portability

- Simplicity in programming
- Wealth of available software components/tools
- Wealth of available software expertise
- Hardware and operating system independence
- Scalable hardware support

Agenda

Why Robotics and Java Platform for Robotics?

Profiles, Shapes, and Sizes

Rat- and Cat-Sized Applications

Elephant-Sized Applications

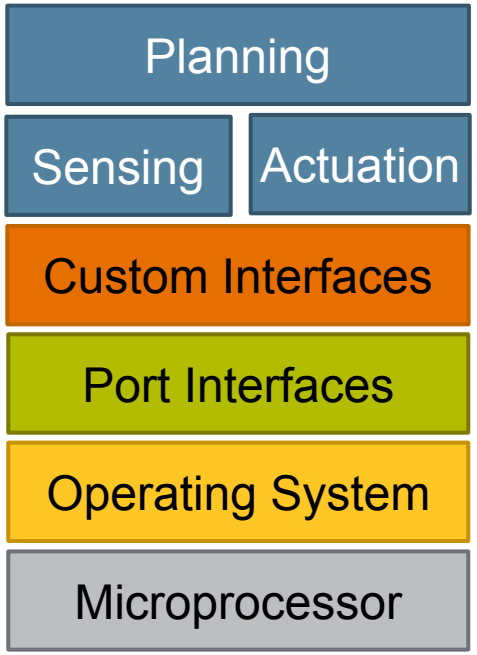
Video

Benefits and Conclusions

Java Platform, Standard Edition (Java SE) for Robotics

High-Level Robotics Processing

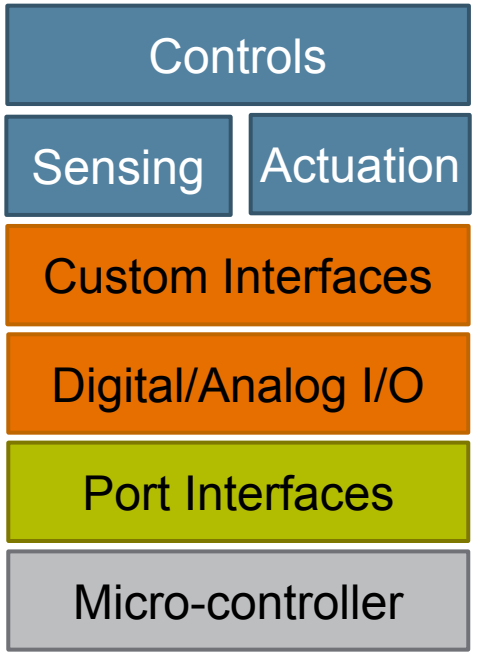
- Embedded PC, desktop, laptop, workstations…
- Linux, Windows, MAC, Solaris™ Operating System (Solaris OS)…
- Serial, parallel, USB, Ethernet…
- PCI and custom interfaces…
- High-level sensing
- Delegated actuation
- Planning/AI galore



Java Platform, Micro Edition (Java ME Platform) for Robotics

Low-Level Robotics Processing

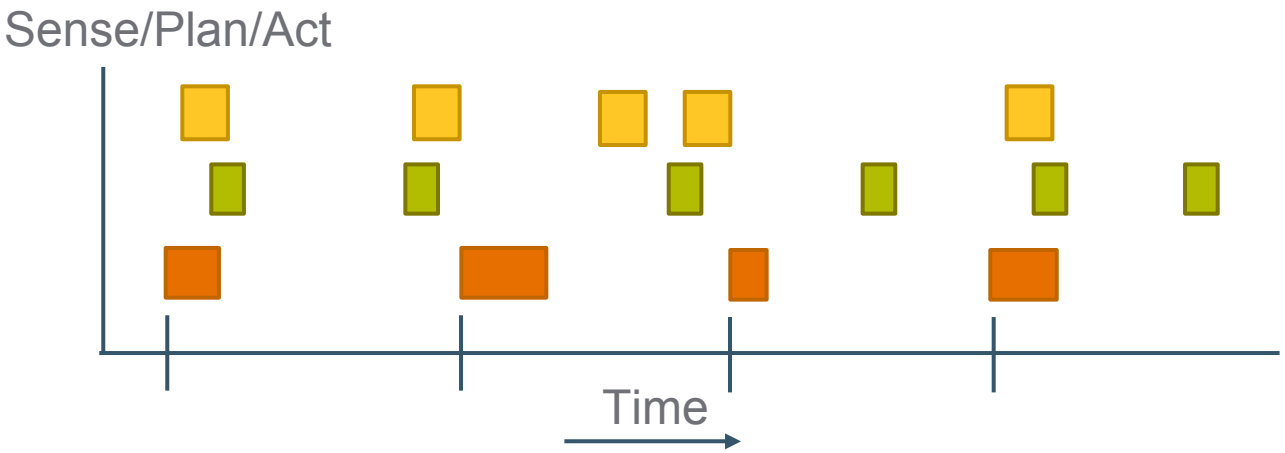
- Java platform on micro-controllers
- Serial, I2C, USB, Ethernet…
- Digital/Analog I/O
- Low-level sensing
- Direct actuation
- Basic controls

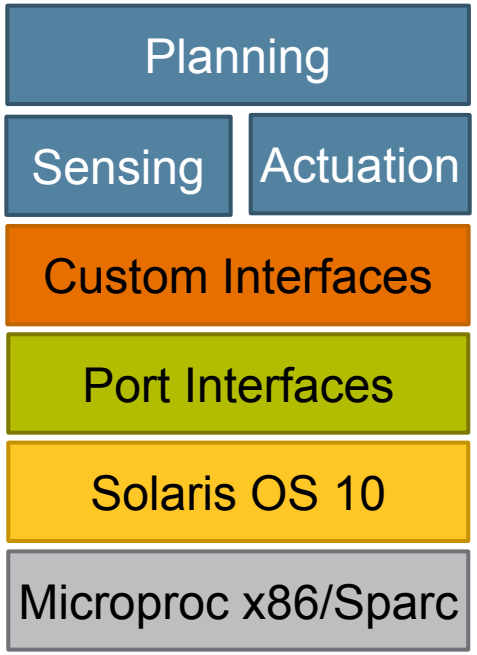


Sun Java Real-Time System (Java RTS) for Robotics

Deterministic Java Platform Processing

- Current SE-grade platform support
- But enabling deterministic and timely processing
  - Handle sensations
  - Execute plans
  - Handle actuations

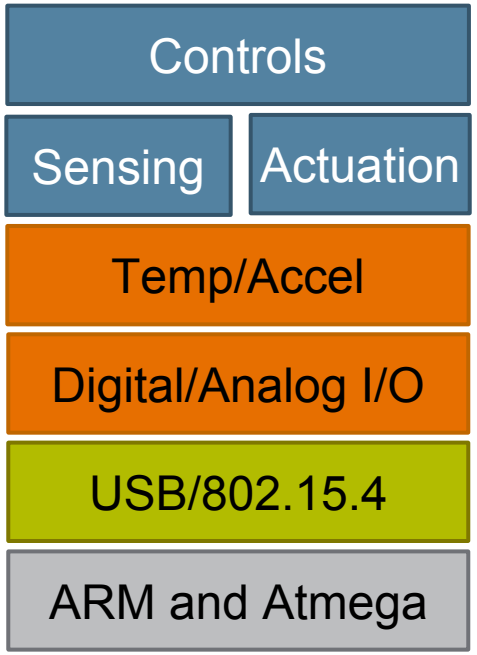




Project Sun SPOTs for Robotics

Robust/Wireless Java Technology-Based Micro-Control

- Simple Java platform controller commands
- Offloaded real-time actions
- Low-level I/O via Java platform

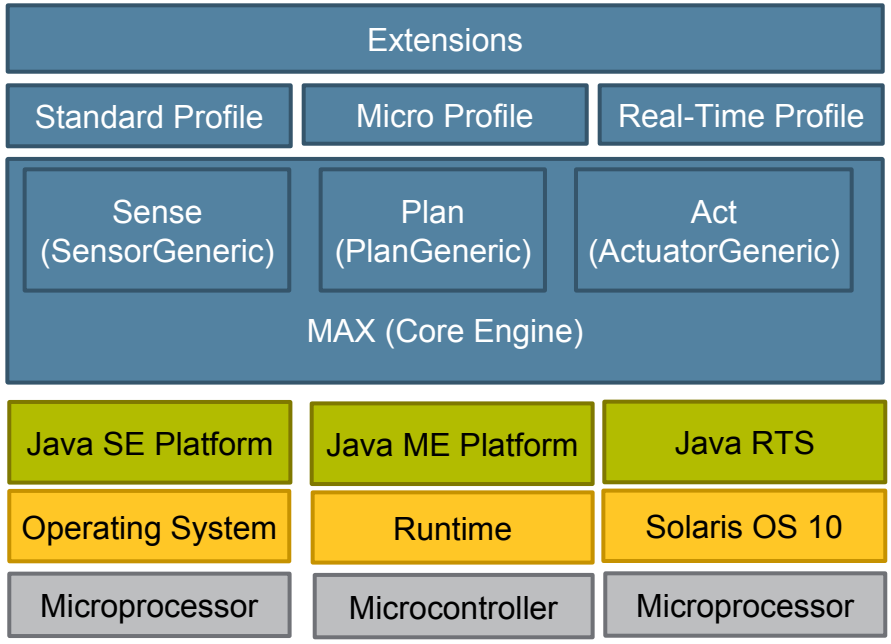


Sun SPOT = Sun Small Programmable Object Technology

MAX: A Java Technology-Based Robotics Engine

Familiar Java Platform Interface to Robots/Hardware

- Provides higher-level robotics API
- Common across platforms
- Scalable across profiles
- High-level extensions
- MAX™
  - Platform
  - Profiles
  - Extensions



MAX is a Trademark of Perrone Robotics. Patent pending.

General MAX Operation Example

```
import com.perronerobotics.plan.*;
import com.perronerobotics.sensor.*;
import com.perronerobotics.actuator.*;
import com.perronetech.system.*;
import com.perronetech.id.*;

public class MyPlan extends PlanGeneric{
  public void init(ID callerID){
    // Get components (threading, I/O, management, distribution, etc configured)
    sensor = (Sensor) super.getSensor(SENSOR_ID);
    actuator = (Actuator) super.getActuator(ACTUATOR_ID);
    actuatorPlan = actuator.getActuatorPlan();
    component = (Component) super.getComponent(COMPONENT_ID);
  }
  public void trigger(ID callerID){
    // Get some sensor state value
    SensorState state = sensor.getSensorState();
    double stateValue = state.getDouble(STATE_VALUE_ID);

    // Get a desired state value from some example component
    double desiredValue = component.getDouble(DESIRED_VALUE_ID);

    // Schedule the actuator to move to a position based on difference
    actuatorPlan.setPosition(desiredValue - stateValue);
  }
}
```

Source: Perrone Robotics, Inc. Copyrights reserved, MAX trademarked, and MAX technology patent pending.

Agenda

Why Robotics and Java Platform for Robotics?

Profiles, Shapes, and Sizes

Rat- and Cat-Sized Applications

Elephant-Sized Applications

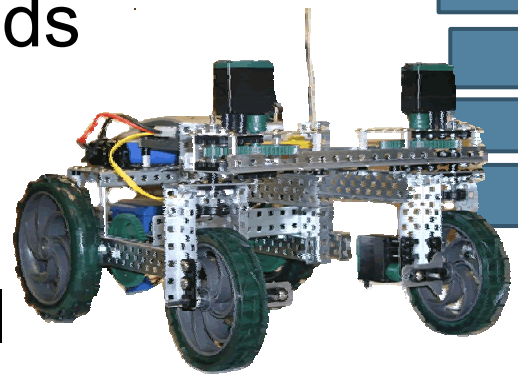
Video

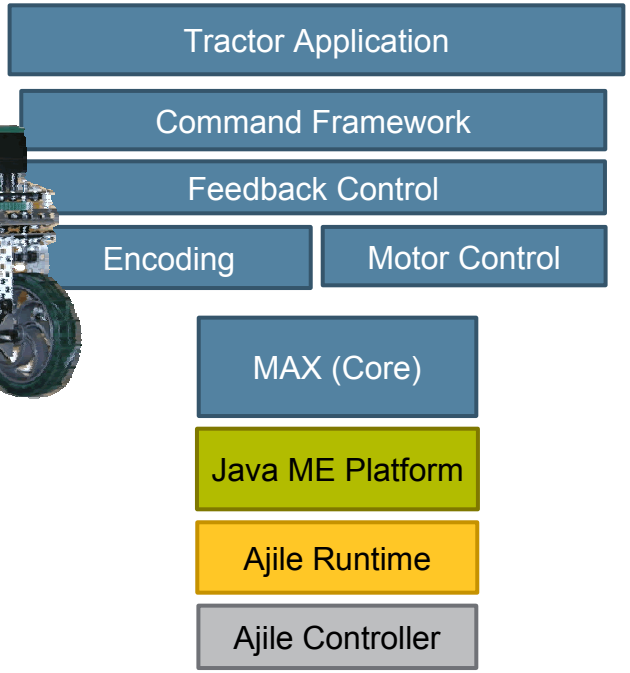
Benefits and Conclusions

Vex-Based Tractor Prototype

Java Platform on Vex Robotics Hobby Kit

- Customer desired proof of controls concept
- Six independently controlled motors
- Vet on COTS Vex Robot HW
- Vex R/C commands
- Vex Encoders
- Vex Motors
- Feedback Control
- Traxter Ap/Config atop
- Scalable to real system

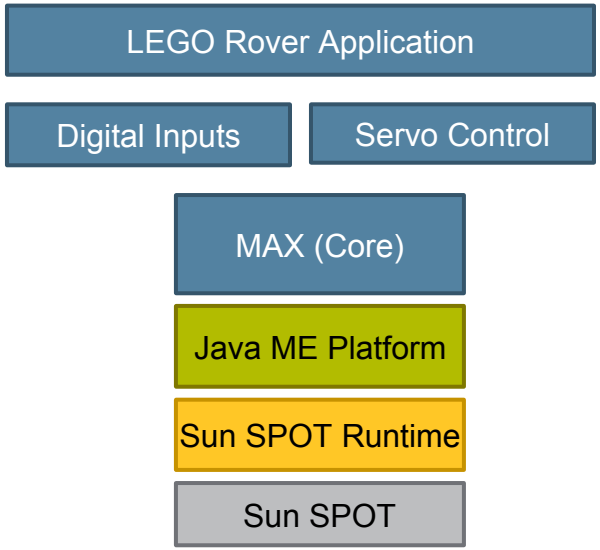




LEGO-Based Example

Java Platform on LEGO NXT Robotics Hobby Kit

- Tinkering with LEGOs
- LEGO Bump Sensors
- LEGO Servos
- Rover sample atop
- Readily evolveable



Code Sample: Actuation

```
public class BasicMotorPlan extends PlanGeneric {
    public void init(ID callerID){
        myMotor = (DCMotor) super.getActuator(MY_MOTOR_ID);
    }
   public void moveMotor(int movementPolarity, double movementAmount){
        if(positionResolution < Mathematics.abs(movementAmount)){
            currentPolarity = movementPolarity;
            myMotor.run(currentPolarity, duty, runTime);
        }
    }
    public void stop(){
        currentPolarity = AngularDirection.STOP_DIRECTION;
        myMotor.stop();
    }
}
```

Source: Perrone Robotics, Inc. Copyrights reserved, MAX trademarked, and MAX technology patent pending.

Code Sample: Sensing

```
public class BasicEncoderPlan extends PlanGeneric {
   public void init(ID callerID){
    mySensor = (FeedbackSensor) super.getSensor(MY_ENCODER_ID);
    motorPlan = (BasicMotorPlan) super.getComponent(MOTOR_PLAN_ID);
   }
   public void setPosition(double desiredPosition){
        // Limit desired position level
        if(desiredPosition > maxPosition){
            desiredPosition = maxPosition;
        }else if(desiredPosition < minPosition){
            desiredPosition = minPosition;
        }
        this.desiredPosition = desiredPosition;
    }
}
```

Source: Perrone Robotics, Inc. Copyrights reserved, MAX trademarked, and MAX technology patent pending.

Code Sample: Sensing (Cont.)

```
public void trigger(ID callerID){
        currentPosition = mySensor.getEncodedPulseValue();
        // Compute desired amount to move
        movementAmount = desiredPosition - currentPosition;
        // Compute direction for running motor
        if(movementAmount > 0){
            movementPolarity = AngularDirection.CLOCKWISE_DIRECTION;
        }else if(movementAmount < 0){
            movementPolarity =
                AngularDirection.COUNTER_CLOCKWISE_DIRECTION;
        }else{
            movementPolarity = AngularDirection.STOP_DIRECTION;
        }
        motorPlan.controlMotor(movementPolarity, movementAmount,
                               desiredPosition, currentPosition);
    }
}
```

Code Sample: Motor Controls

```
public void controlMotor(int movementPolarity, double movementAmount,
                         double desiredPosition, double currentPosition){
    if(movementPolarity != AngularDirection.STOP_DIRECTION){
      // If is not moving in desired direction already...
      if(currentPolarity != movementPolarity){
        // If haven't exceed max in applied direction...
        if(movementPolarity == AngularDirection.CLOCKWISE_DIRECTION){
          if(currentPosition <= desiredPosition){
            moveMotor(movementPolarity, movementAmount);
          }
        }else{ // == AngularDirection.COUNTER_CLOCKWISE_DIRECTION
          if(currentPosition >= desiredPosition){
            moveMotor(movementPolarity, movementAmount);
          }
        }
      }
    }
}
```

Code Sample: Motor Controls (Cont.)

```
 }else{ // Is running in correct direction already...
   if(positionResolution >= Mathematics.abs(movementAmount)){
                    stop();
   }else if(currentPolarity == AngularDirection.CLOCKWISE_DIRECTION){
                    if(currentPosition >= desiredPosition){
                        stop();
                    }
   }else if(currentPolarity == AngularDirection.COUNTER_CLOCKWISE_DIRECTION){
                    if(currentPosition <= desiredPosition){
                        stop();
                    } } }
}else{ // Dead on target current = desired
       // If not stopped...
       if(currentPolarity != AngularDirection.STOP_DIRECTION){
           stop();
       } } }
```

Source: Perrone Robotics, Inc. Copyrights reserved, MAX trademarked, and MAX technology patent pending.

Code Sample: Feedback Controls

```
public class BasicFeedbackControlPlan extends PlanGeneric{
  public void init(ID callerID){
    control = (FeedbackControl) super.getComponent(FEEDBACK_COMPONENT_ID);
    // Motor and sensor automatically configured/registered
  }
  public void setPosition(double desiredPosition){
    this.desiredPosition = desiredPosition;
  }
  public void trigger(ID callerID){
    // Set desired steering value
    control.setDesiredPosition(desiredPosition);
  }
}
```

Agenda

Why Robotics and Java Platform for Robotics?

Profiles, Shapes, and Sizes

Rat- and Cat-Sized Applications

Elephant-Sized Applications

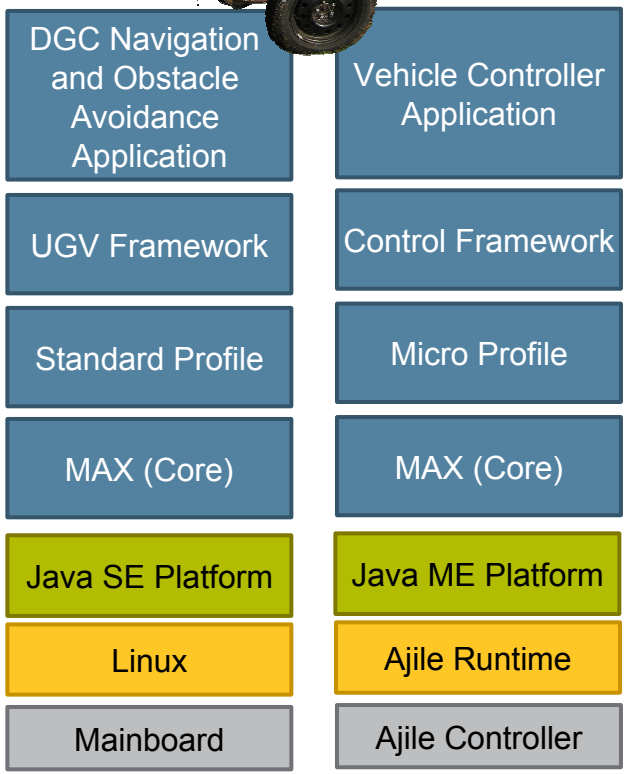
Video

Benefits and Conclusions

Tommy: DARPA Grand Challenge AGV

Autonomous Desert Racing Robot

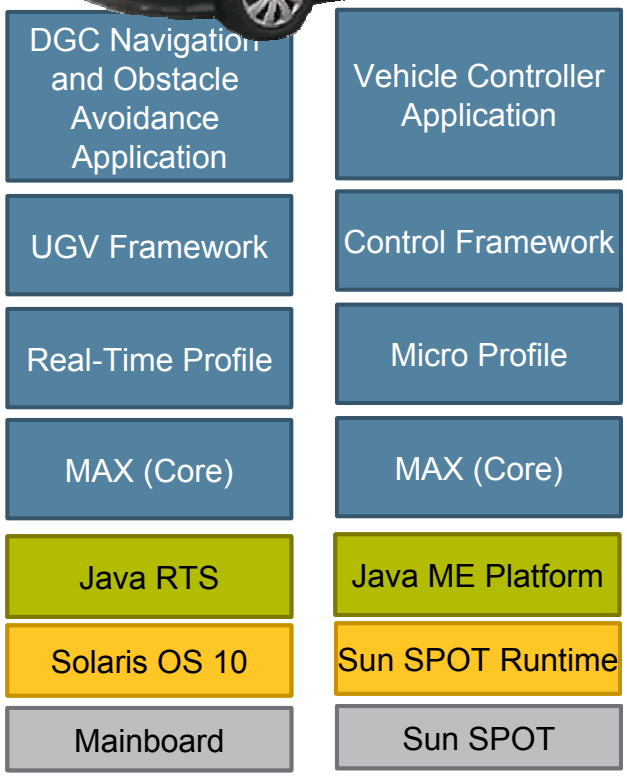
- Single processor card
  - GPS, Inertial Navigation System
  - Lasers/RADAR for detection
  - Brains: commands controllers
- Two micro-controllers
  - Steering/shifting
  - Throttle/braking
  - Low-level sensing
- 10 man-months software development time



Tommy Jr: DARPA Urban Challenge AGV

Autonomous City Racing Robot

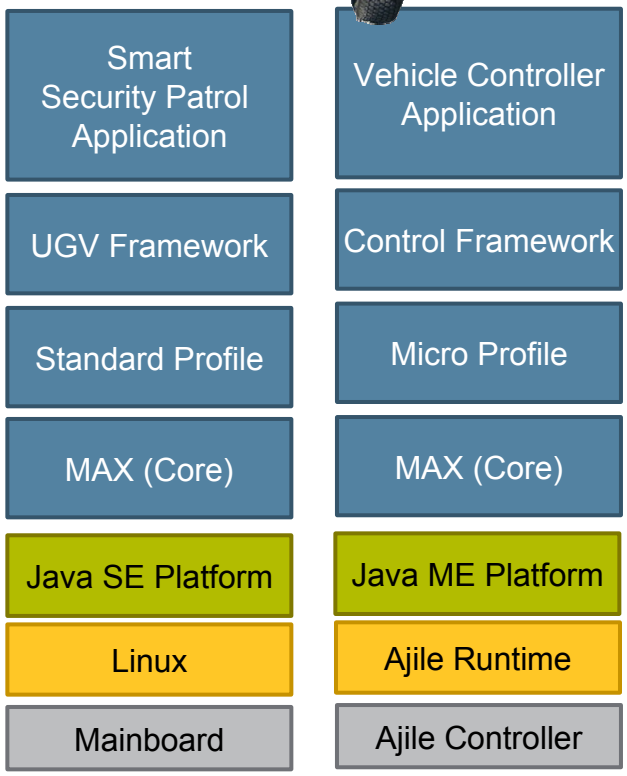
- Leverage reusable platform
- Drop in more navigation rules
- Drop in more detection rules
- Drop in stereo sensor driver
- Java real-time system
- Project Sun SPOTs



KartBot: Low-Cost UGV

A Low-Cost UGV Approach

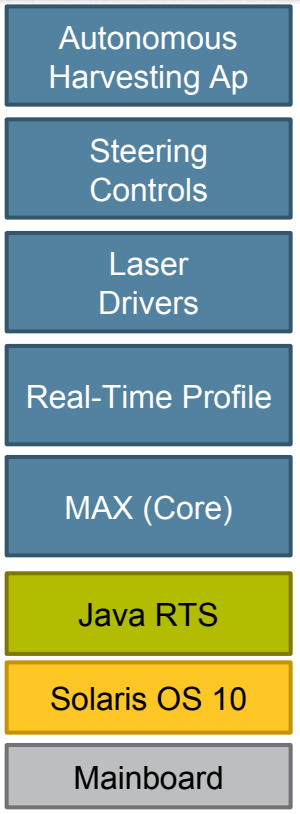
- Modified go kart
  - Bigger engine
  - More gas
  - 35 mph
  - 400 pound payload capacity
  - 8-hour mission before refuel
- After-market UGV drop-in
  - Drop-in actuation
  - Drop-in electronics
  - Drop-in sensors
- Tommy's DNA
  - Same software
  - Tweaked parameters



Auto-Steering Harveste

Autonomous Farming

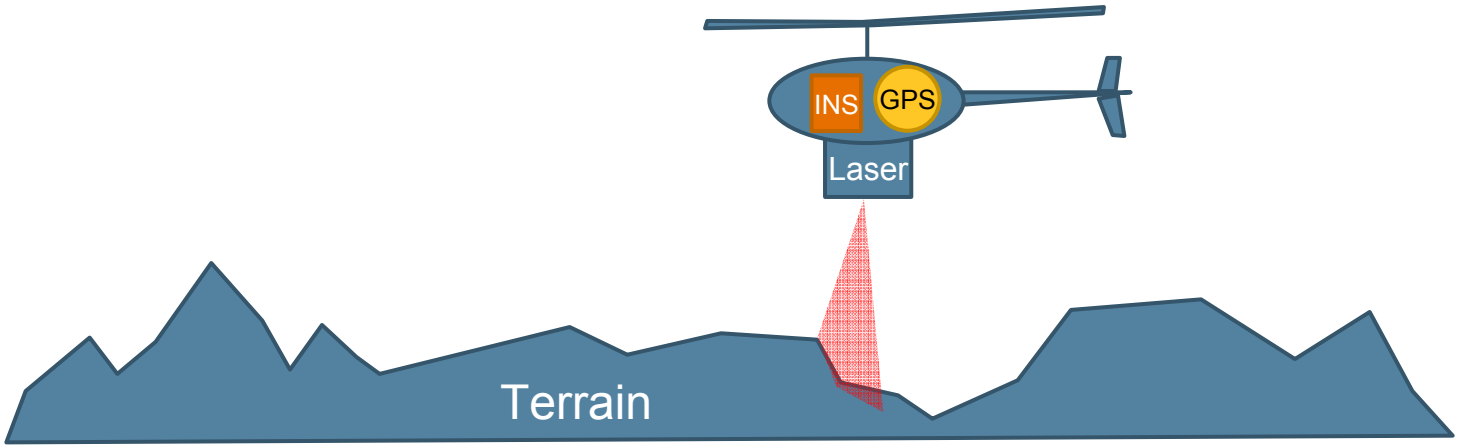
- Harvesting
  - Harvester throttles trees for nuts
  - Driver slowed by controls
- Autonomous harvesting
  - Sense trees and form paths
  - Closed loop control of steering
  - Real-time steering controls



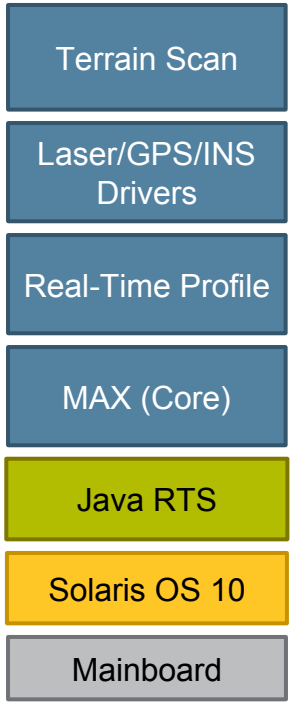
UAV Terrain Scanning

Real-Time Data Collection

- Laser scans of terrain
- Fused real time with GPS/INS
- Data processed to create point grid
- Data acquisition timing is critical



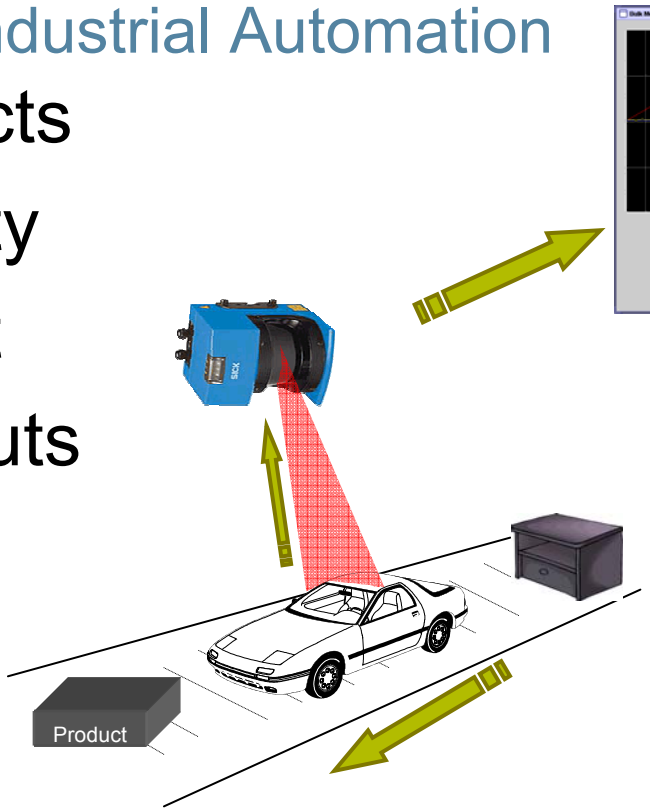
INS = Inertial Navigation System

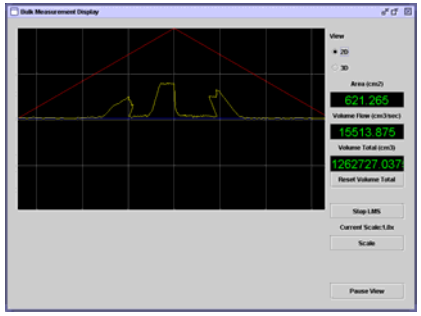


Industrial Automation Example

Distributed Sensing for Industrial Automation

- Laser scans products
- Area/volume/density
- Width/length/height
- Digital/analog outputs
- Encoder inputs





Product Measurement

Laser Drivers

Standard Profile

MAX (Core)

Java SE Platform

Operating System

Embedded PC

Code Sample: Vehicle Navigation

```
public class MyNavigationPlan extends PlanGeneric{
  public void init(ID callerID){
    gpsSensor = (ReferenceSensor) super.getSensor(GPS_SENSOR_ID);
    insSensor = (ReferenceSensor) super.getSensor(INS_SENSOR_ID);
    steeringMotor = (Actuator) super.getActuator(STEERING_MOTOR_ID);
    steeringPlan = steeringMotor.getActuatorPlan();
    route = (Route) super.getComponent(ROUTE_ID);
  }
  public void trigger(ID callerID){
    Position currentPosition = gpsSensor.getPosition();
    Position desiredPosition = route.getDesiredPosition();
    Orientation direction
          = desiredPosition.calculateDirection(currentPosition);
    Orientation currentOrientation = orientationSensor.getOrientation();
    Orientation directionDiff
          = currentOrientation.calculateDifference(direction);
    steeringPlan.setPosition(directionDiff.getHeading().getValue());
  }
}
```

Agenda

Why Robotics and Java Platform for Robotics?

Profiles, Shapes, and Sizes

Rat- and Cat-Sized Applications

Elephant-Sized Applications

Video

Benefits and Conclusions

Video

Java Technology-Based Robots on the Run!

Agenda

Why Robotics and Java Platform for Robotics?

Profiles, Shapes, and Sizes

Rat- and Cat-Sized Applications

Elephant-Sized Applications

Video

Benefits and Conclusions

Benefits

- Why Java platform for Robotics?
  - Simplicity
  - Available tools
  - Available experts
  - Hardware independence
  - OS independence
  - Scalable profiles
    - Java ME platform—Java platform for micro/embedded controls
    - Java SE platform—Java platform for standard processing power
    - Java RTS—Java platform for real-time determinism

Conclusions

- What we've seen
  - Rat- to cat- to elephant-sized applications
  - From experimental to industrial applications
  - Wheeled, winged, fixed, or mobile
  - Hardware must vary
  - Profiles must vary
  - Real time on the micro-controller
  - Real time on the main processor

For More Information

Perrone Robotics and MAX

www.perronerobotics.com

Contact

Paul Perrone

703-728-0115

pperrone@perronerobotics.com

Team Jefferson **Robotic Racing Team**

www.teamjefferson.com

Sponsors

Perrone Robotics

University of Virginia Engineering

Sun microsystems

Java

TDF

45Fix

SICK Sensor Intelligence.

BDMetrics™

TOMMY Jr.

Q&A

Paul J. Perrone

Founder/CEO
Perrone Robotics
www.perronerobotics.com

All About Java™ Technology Based Robotics

Paul J. Perrone

Founder/CEO
Perrone Robotics
www.perronerobotics.com

Session TS-1519