Java

JavaOne℠

LincVolt Car: Driving Toward 100 MPG

Paul J. Perrone

Software – LincVolt

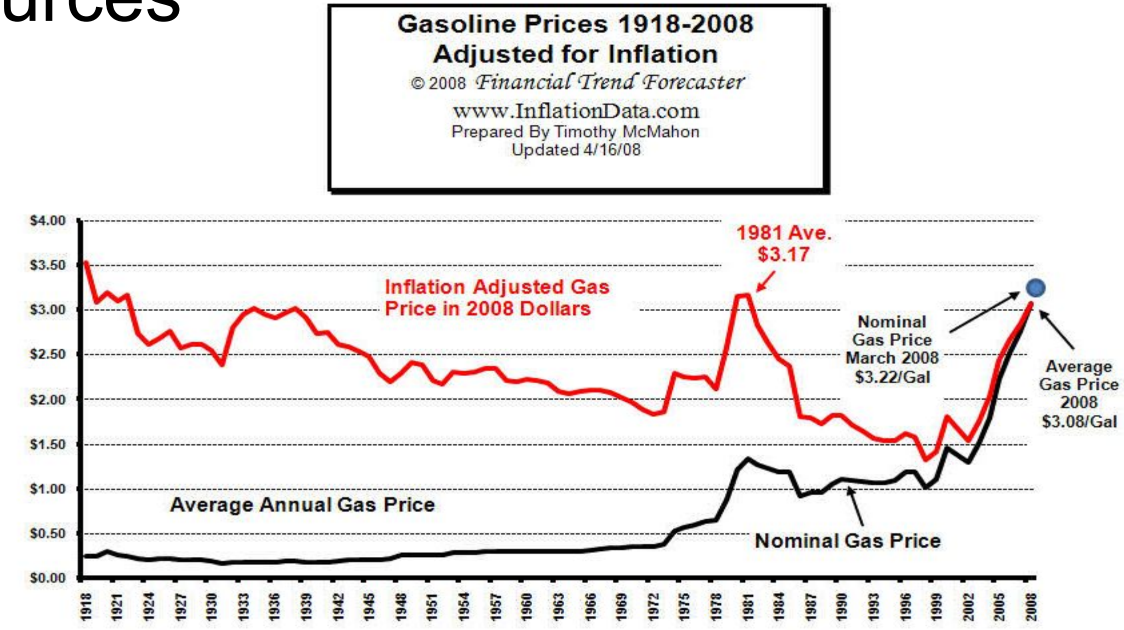
Founder – Perrone Robotics

LINCVOLT

Perrone Robotics

The Problem at Hand

- Rising Gas Costs
- Clean Energy Demands
- Limited Resources



Gasoline Prices 1918-2008
Adjusted for Inflation
© 2008 *Financial Trend Forecaster*
www.InflationData.com
Prepared By Timothy McMahon
Updated 4/16/08

Note: Prices are Average Annual prices not Peak Prices
so peaks are smoothed out considerably

Source of Data: US Energy Information Administration
CPI-U Inflation index- www.bls.gov

LincVolt’s Birth

Neil, Jonathon, and a Lincoln

The Automotive X-Prize

- Competition to build super efficient vehicles
  - > 100 MPGe
- $10M to three class winners
- Qualification/final events in 2010
- http://www.progressiveautoxprize.org

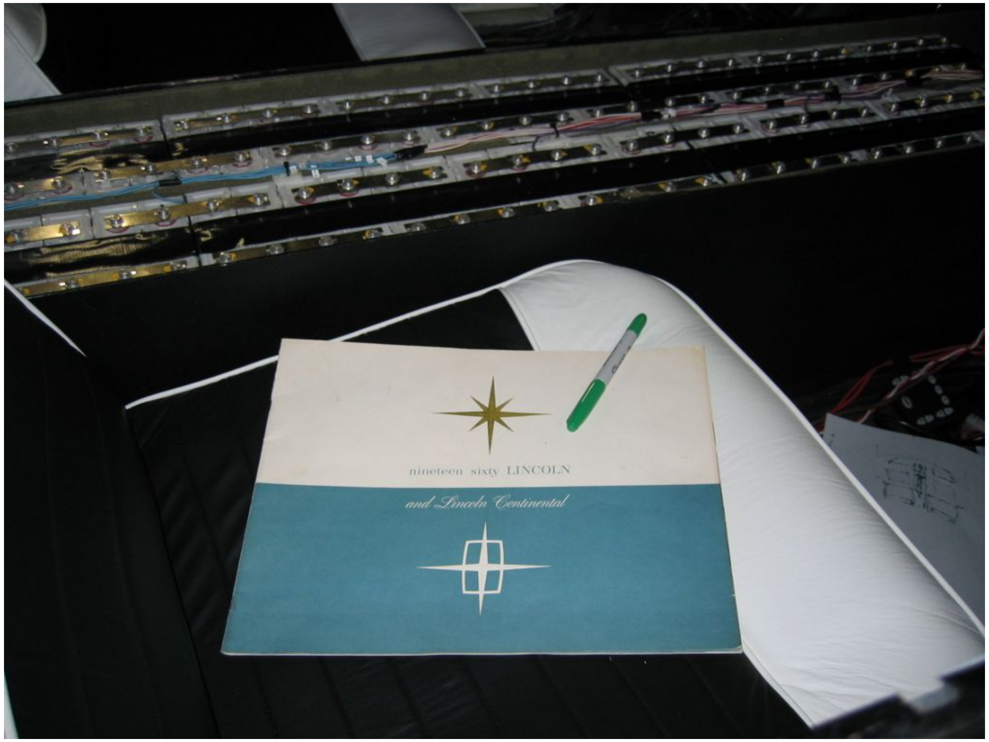
LincVolt’s Parts

1959 Lincoln Continental

- 19 feet long
- 2.5 tons

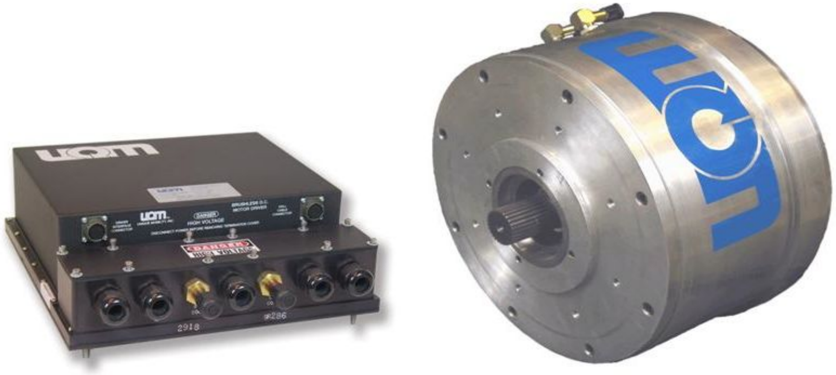
Batteries

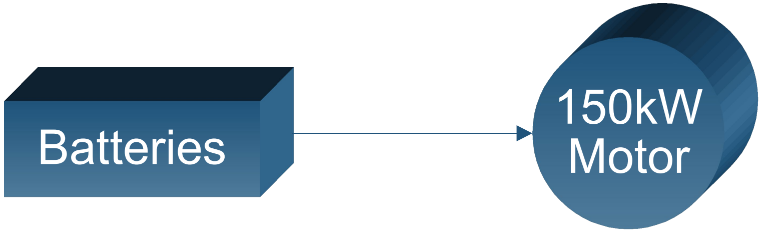
- 100+, 3.2V Lithium-Ion batteries
  - Power main drive train
- 12V cells
  - Power equipment



Electric Motor

- Existing Lincoln Engine Removed
- Drop in of Electric Motor to Drive Shaft
  - 320 V
  - 150 kW
  - Water cooled

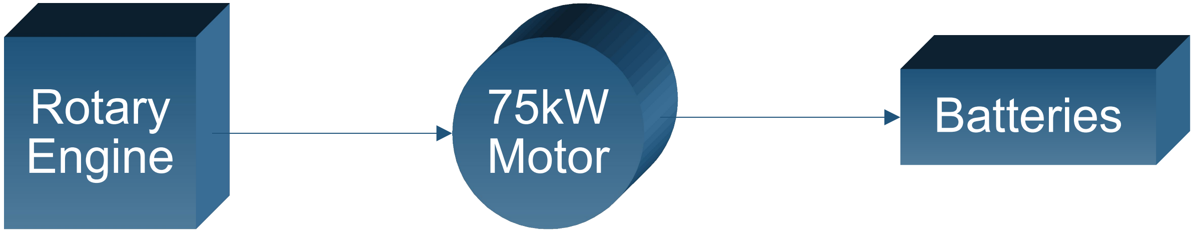



Batteries

150kW Motor

Generator

- Rotary Engine
- Drives 75kW/320V Electric Motor
- Produces charge to batteries



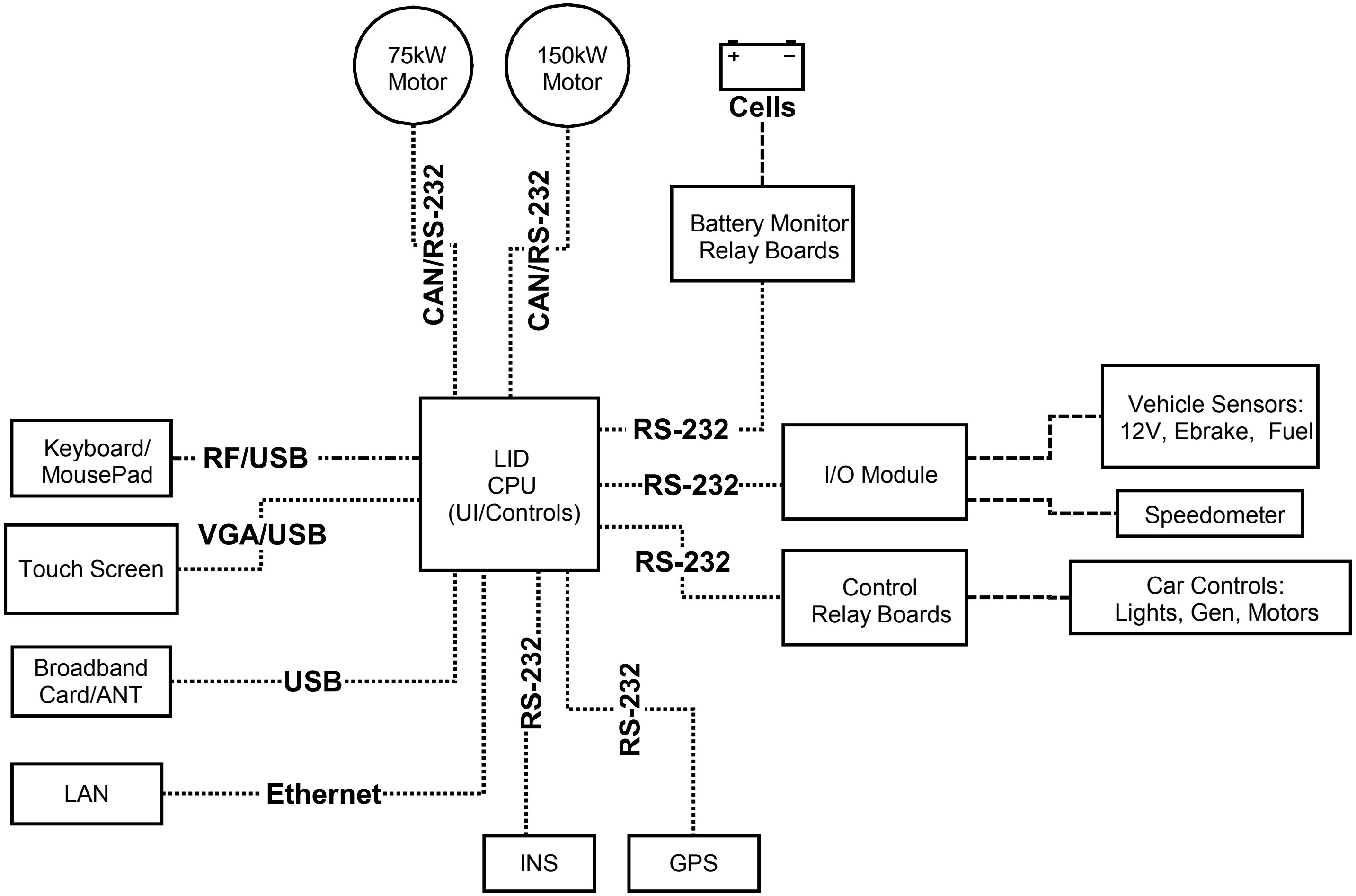
Fuels

- Modular/adaptable approach
- v2.0:
  - CNG-based
- v2.5:
  - Gas
  - Bio-Diesel
  - Water Gas
- v3.0:
  - Stay tuned.

The “LID”

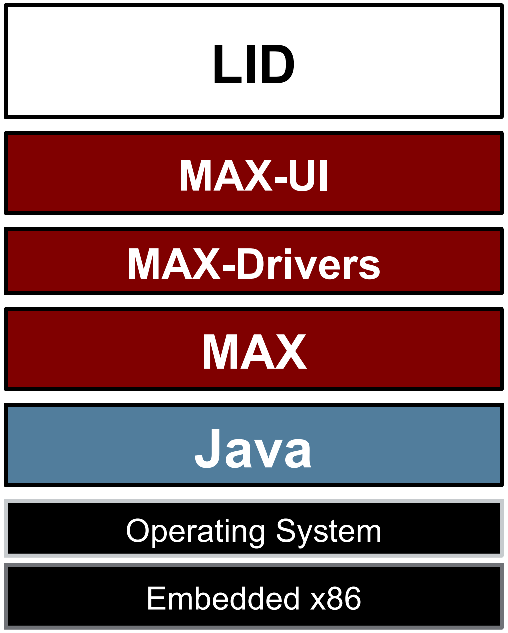
- LincVolt Intelligent Dashboard
    - Sense vehicle information
    - Push to Web
    - Provide user interface
    - Control automotive functions manually
    - Control automotive functions automatically

LID Hardware



LID Software Layers

- Java, Java, Java
  - Java SE
  - Java RTS when partition controls
- MAX Robotics Platform
  - Standard and Real-Time profiles
  - General purpose integration platform
  - Many available drivers and libraries
- MAX Drivers
  - GPS, INS, Analog, Digital
- MAX-UI
  - Rapid UI creation framework
- The LID
  - LincVolt-specifics and intelligence



LID

MAX-UI

MAX-Drivers

MAX

Java

Operating System

Embedded x86

LincVolt Sensing

Automotive State Sensing Example

- LincVolt management code
  - Calculating fuel consumed and MPGGE
  - Publish to UI and Web
- Fuel Sensing
  - MAX analog input driver
  - Analog current proportional to fuel level
  - Non-linear fuel maps = linear segments
- Fuel Calcs
  - Update when fuel tanks filled and depleted

Automotive State Sensing: MPGGE

```
// Update the gas sensor stats
gasGallons = gasTank.update();
// Get the current gallons consumed
gasConsumed = gasTank.getGallonsConsumed();

// Update the bio diesel sensor stats
biodGallons = bioTank.update();
// Get the current gallons consumed
bioConsumed = bioTank.getGallonsConsumed();

// Compute the mpgge
double mpgge = tripMiles / (gasConsumed * 1.0 + bioConsumed * 0.9);

// Publish to the UI and the Web
uiPub.updateMPGGE(mpgge);
webPub.updateMPGGE(mpgge);
```

Automotive State Sensing: Fuel Sensor

```
// Updated from analog input sensing
private void updateCurrent(double value){
    currentFilter.set(value);
}

public double getGallons(double current){
    Line fuelMap = getFuelMap(current);
    return fuelMap.computeY(current);
}

public double getGallonsRemaining(){
    double current = currentFilter.getAverage();
    return getGallons(current);
}
```

Automotive State Sensing: Fuel Calcs

```
public double update(){
      gallonsRemainingInTank = getGallonsRemaining();

      double gallonsDifference = prevGallonsRemainingInTank - gallonsRemainingInTank;

      if(filledTank(gallonsDifference)){ // If filled the tank
                  gallonsConsumedOnPriorTanks = gallonsConsumedTotal;
                  gallonsAtTankRefill = gallonsRemainingInTank;
      }else{ // Else have not filled tank, either no consumption or some consumption...
            double gallonsConsumedOnCurrentTank
                                    = gallonsAtTankRefill - gallonsRemainingInTank;
            if(gallonsConsumedOnCurrentTank >=0 ){
                  gallonsConsumedTotal
                                    = gallonsConsumedOnCurrentTank + gallonsConsumedOnPriorTanks;
            }

            prevGallonsRemainingInTank = gallonsRemainingInTank;
      }

      return gallonsRemainingInTank;
}
```

Motor State Sensing Example

- CAN-Serial Interface
- MAX Serial Driver
- MAX Peripherals and Messages
  - Sense CAN messages
  - Parse messages
  - Access message data

Motor State Sensing Example

```
// Concrete message class…
private void parseFeedback(String response){
        // T04EF01028807DA07D807D807D8568
        feedbackMessage.setMessage(response);
}

 public double getVoltage(){
        return super.getShortValue(7, 0.1, -3212.8);
 }

// Generic message class…
public synchronized void setMessage(String msg){
        synchronized(this){
            message = msg;
            bytesValid = false;
        }
 }


 public double getShortValue(int startIndex, double scaler, double offset){
        setBytes();

        short val = BinaryInputMessage.parseShort(startIndex, bytes, false);

        int valInt = (((int) val) & 0x0000FFFF);

        return (valInt + offset/scaler)*scaler;
    }


 private synchronized void setBytes(){
        if(!bytesValid){
            synchronized(this){
                bytes = Bytes.hexStringToBytes(message);
                bytesValid = true;
            }
        }
    }
```

Generator State Sensing Example

- Parse messages from motors (e.g. RPM, volts)
- Push to generator controls

```
double rpm = uqmgen.getFeedbackMessage().getSpeed();
generator.updateRPM(rpm);

double volts = uqmgen.getFeedbackMessage().getVoltage();
generator.updateVoltage(volts);
```

Power State Sensing Example

- Monitor 12V level through analog input
- Cycle through all 100+ cells via relay control
- Sense each cell via analog input
- MAX drivers for relay control and analog inputs

```
for(int i=0; !hasFailed() && i < numberBatts; ++i){
    // First make sure the battery isn't blacklisted. Ignore it if it is.
    if(!isBatteryBlackListed(i)){
            // If returns false, then relay is closed still...set a failure condition.
            if(!readNCDRelayBattery(i)){
                    setFailed(true);
            }
    }
}
```

Power State Sensing Example

```
pause(100);

// Turn on relay
turnOnRelay(index);

// Read from analog input (and pause beforehand)
float voltage = pollAnalogInput();

// Turn off all relays (for safety)...and pause first before turn off output
pause(100);
turnOffAllRelays();

// Pause for off amount of time
pause(millisOff);

// If the relay is indeed off...
if(isRelayOff(index)){
	// Set the voltage and check battery levels
	setBatteryLevel(index, voltage);

	// Return true (relay off/open)
	return true;
}else{
	setBatteryLevel(index, Battery.INVALID_VOLTS);

	// Return false (relay on/closed)
return false;
}
```

LincVolt Controls

Automotive Controls Example: Manual

- MAX-UI maps buttons to events
- Map events to concrete interlock checks.
- MAX digital relay controls configured for control points.

```
public synchronized void turnOnUQM150(boolean on){
     // If turning on UQM 150...
     if(on){
               // Check if butt box is interlocked...
               boolean isButtSafe = isButtSafe();

               // Check if DCP is on
               boolean isDCPSafe = isOn(DCPRIMARY);

               // Check if Water Pump on
               boolean isWaterPumpSafe = isOn(WATERPUMP);

               // Now...only turn on if all are true...
               if(isButtSafe && isDCPSafe && isWaterPumpSafe){
                              turnOn(UQM150);
               }else{
                              // Turn off UQM 150 to be safe..
                              turnOff(UQM150);
               }
     }else{ // Is turning off...is always safe to turn off
               turnOff(UQM150);
     }
}
```

Automotive Controls Example: Automatic

- MAX analog output for updating speedometer

```
public void updateSpeedometer(double speedMPH){
    // Analog output configured to map mph to voltage
    analogOutput.write(speedMPH);
}
```

- MAX digital outputs for controlling brake lights

```
public void update(double load){
	if(load < minRegenPower){
			turnLights(true);
	}else{
			turnLights(false);
	}
}

private void turnLights(boolean on){
	// If turning on and lights are off...
	if(on && digitalOutput.isOpen()){
			// Turn on.
			digitalOutput.write(true);
	// Else if are turning off and lights are on...
	}else if(!on && digitalOutput.isClosed()){
			// Turn off.
			digitalOutput.write(false);
	}
}
```

Generator Controls Example

- Automatic controls update every 250 mSecs

```
public void control(){
  // If voltage is less than some min voltage
     if(shouldTurnOn()){
     // Turn on the generator
        turnOn();
     }else if(shouldTurnOff()){ // Else if above max voltage
     // Turn off the generator
        turnOff(false, true);
     }

     // Update the motor control loop on a cyclic basis.
     updateMotorControl();
  }
```

Generator Controls Example

- Example check for on and activation of generator:

```
private boolean shouldTurnOn(){
      boolean turnOn
                    = (currentVoltage < minOnVoltage) && (currentVoltage > minValidVoltage);

      return isStartOverriden() || turnOn;
}


 private void turnOn(){
      if(!on){
                    on = true;
                    turnOn(DCSECONDARY);
                    turnOn(WATERGAS);
                    turnOn(UQM75);
                    turnOn(GENIGN);
                    turnOn(CHERRY);
                    turnOn(FUELPUMP);

                    controlTime.updateCurrentTime();

                    warmingUp = true;
      }
  }
```

Generator Controls Example

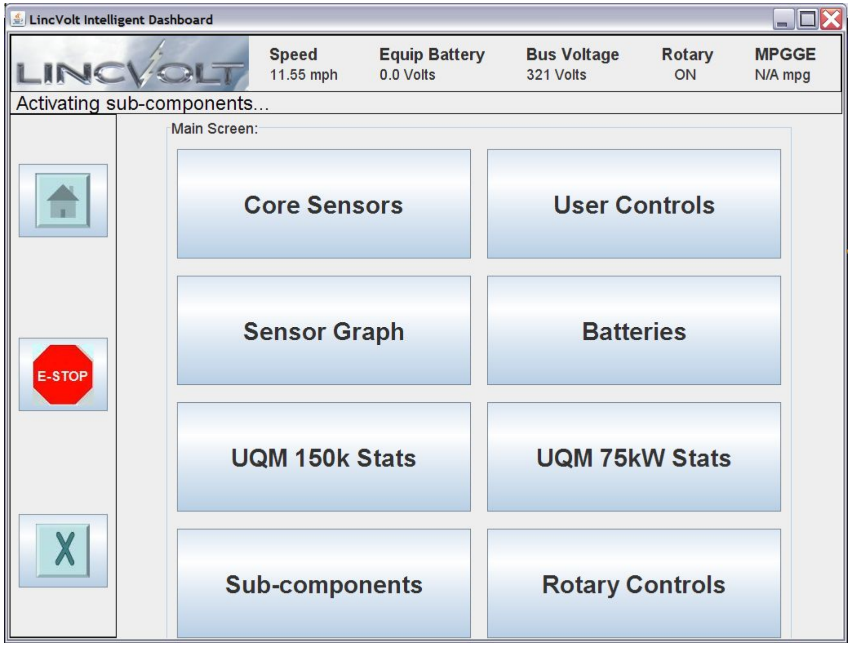
- Example control state check

```
 private void startUp(){
   // If generator started OR timed out...
   if(generatorStarted() || controlTime.isLaterBySeconds(rpmStartUpTimeout)){
      // Turn off starter relay
      turnOff(GENSTART);

      // Also turn off cherry bomb
      turnOff(CHERRY);

      startingUp = false;
      engineWarmUp = true;
      controlTime.updateCurrentTime();
   }
}
```

LincVolt User Interface & the Web

LID UI

- MAX-UI Based
  - Drop config files in directory to create widget display
  - Push events to widgets
  - Receive events from widgets



LincVolt Intelligent Dashboard

Speed	Equip Battery	Bus Voltage	Rotary	MPGGE
11.55 mph	0.0 Volts	321 Volts	ON	N/A mpg

Activating sub-components...

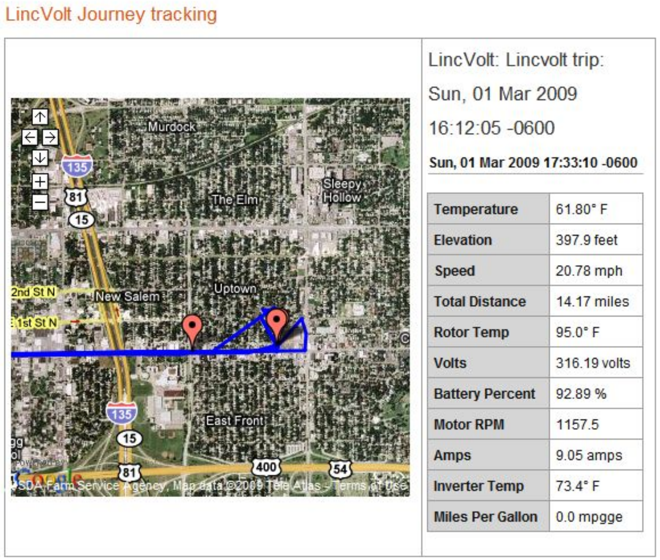
Main Screen:

Always Connected

- Broadband card connected to Internet
- Onboard LincVolt
  - Publish car stats
  - Browse Web
  - Use navigation software
  - Cameras mounted on/in car – streaming
- Mobile software development
  - Updates from PRI no matter where car is
- Global support
  - People watching/helping from everywhere

java.com Streaming

- LincVolt management code:
  - Each cycle, push data to Web publisher class
- Web publisher:
  - Caches car state data locally onboard
  - Periodically formats and sends data to RSS feed
- java.com Site:
  - Pulls data from RSS feed
  - Formats into user-friendly display
  - *Travel along with LincVolt!*



LincVolt Journey tracking

LincVolt: Lincvolt trip: Sun, 01 Mar 2009 16:12:05 -0600

Sun, 01 Mar 2009 17:33:10 -0600

Temperature	61.80° F
Elevation	397.9 feet
Speed	20.78 mph
Total Distance	14.17 miles
Rotor Temp	95.0° F
Volts	316.19 volts
Battery Percent	92.89 %
Motor RPM	1157.5
Amps	9.05 amps
Inverter Temp	73.4° F
Miles Per Gallon	0.0 mpgge

LincVolt Demo

LincVolt Video

Conclusions

The LincVolt Ride

- Demonstrating fuel efficiency with large cars now.
- High profile showcase of capabilities/ingenuity.
- Java onboard:
  - Pushing statistics.
  - Providing info/diagnostics to driver.
  - Intelligently controlling LincVolt.

LincVolt Info

- The LincVolt Web site/store:
  - www.lincvolt.com
  - lincvolt.shop.musictoday.com
- Sun Microsystems supported:
  - java.com/en/java_in_action/lincvolt.jsp
- Perrone Robotics & MAX:
  - www.perronerobotics.com

Thank You

Paul J. Perrone

paul@perronerobotics.com

703-728-0115