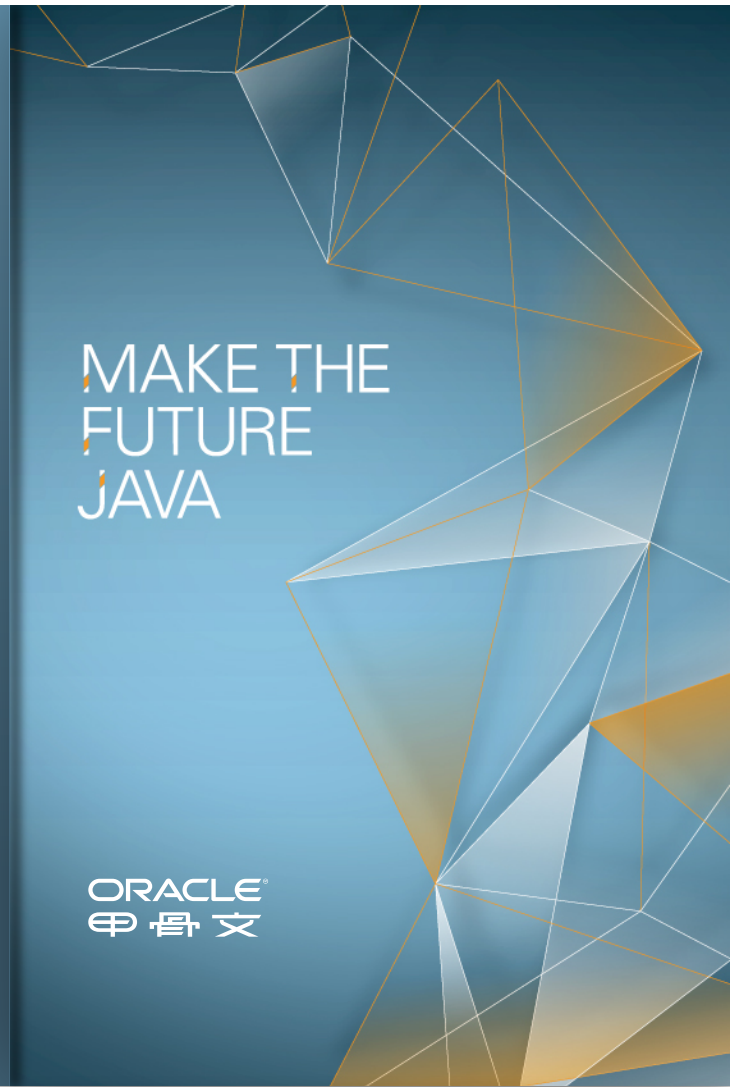

Java™

Beyond Beauty: JavaFX, Parallax, Touch, Gyroscopes and Much More

Angela Caicedo
Java Evangelist, Oracle



What Is this Talk About?

- Looking for a better UIs
- Current UI are so static, too flat, but the use of 3D will be too much.
  - Specially for Embedded
- UIs should react to user positioning, we have the tools
- JavaFX sounds great for more dynamic UIs
- How do we put all this together?

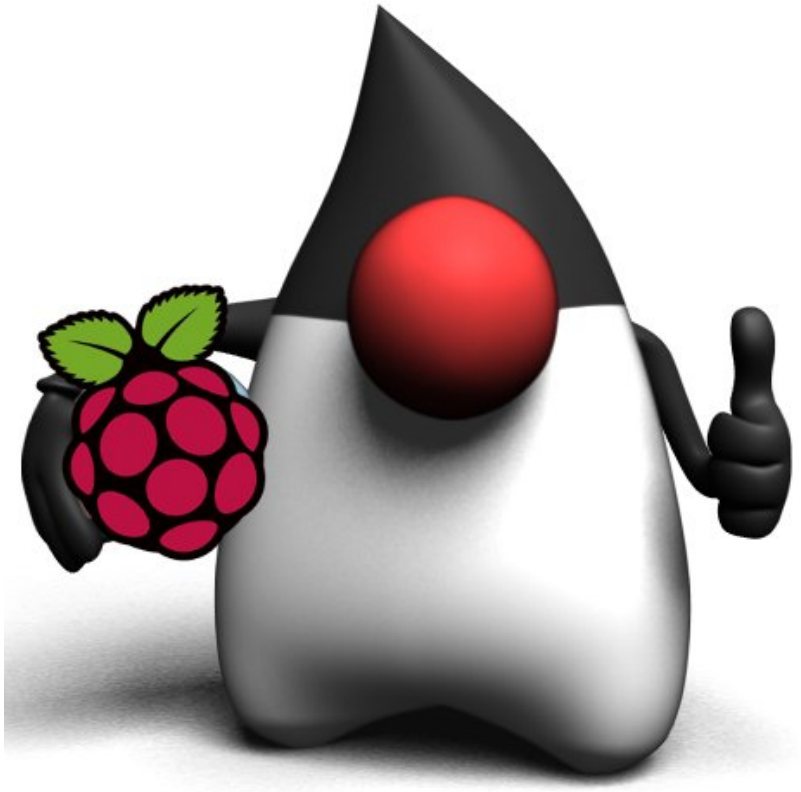
- This is what this talk is about!!!

Agenda

- The Hardware:
  - Raspberry Pi
  - The protocol: I²C
  - Enabling I²C on the Pi
  - Used Hardware, MPU-9150
  - Chalkboard Electronics LCD Screen
- The software
  - I²C Native and Java libraries
  - UI Principles used: Parallax
  - JavaFX for the implementation

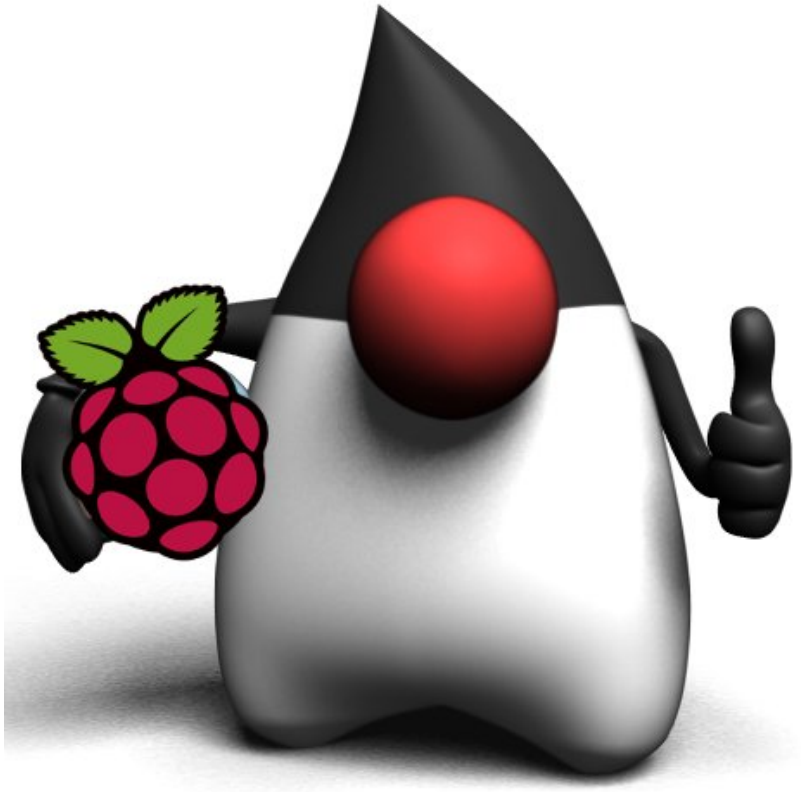
Java™ ORACLE®

The Hardware



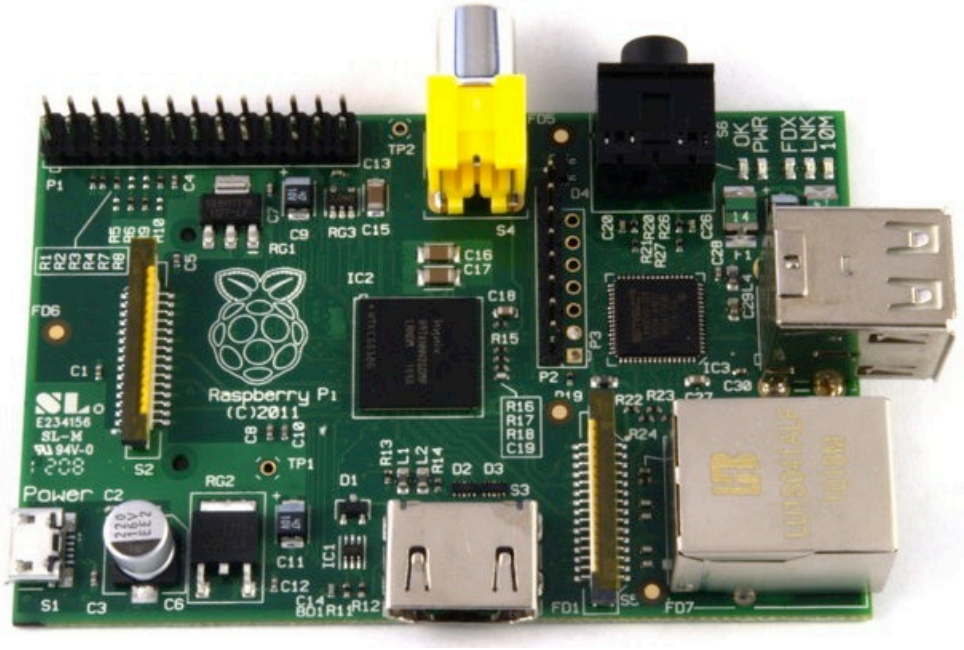
Java™ ORACLE®

The Raspberry Pi



Java™ ORACLE®

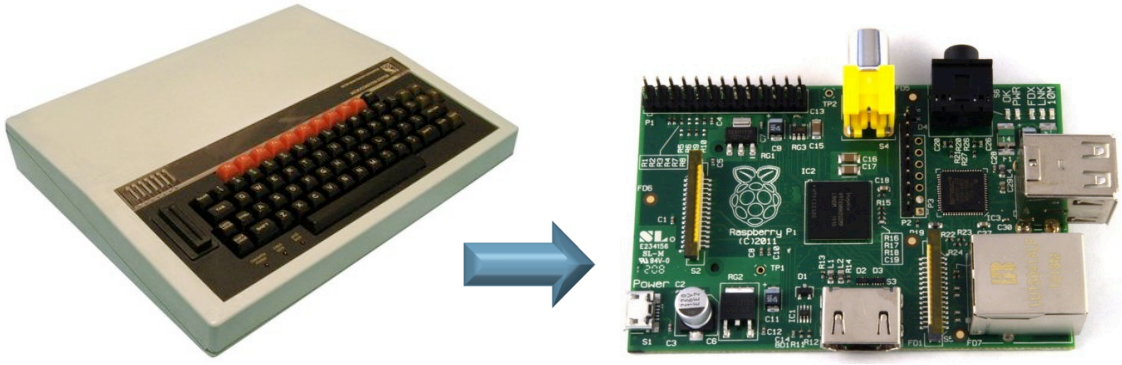
The Raspberry Pi



Java™ ORACLE®

Rapberry Pi

History and Goals



- Project started in 2006
  - Goal to devise a computer to inspire children
  - Inspiration from the BBC Micro project from 1981
- Officially launched on Febuary 29th 2012
  - First production run was 10,000 boards
  - Both RS and Farnell's servers were stalled on the day of launch
  - RS reported over 100,000 pre-orders in one day
  - Current production is about 4,000 boards per day

Java™ ORACLE®

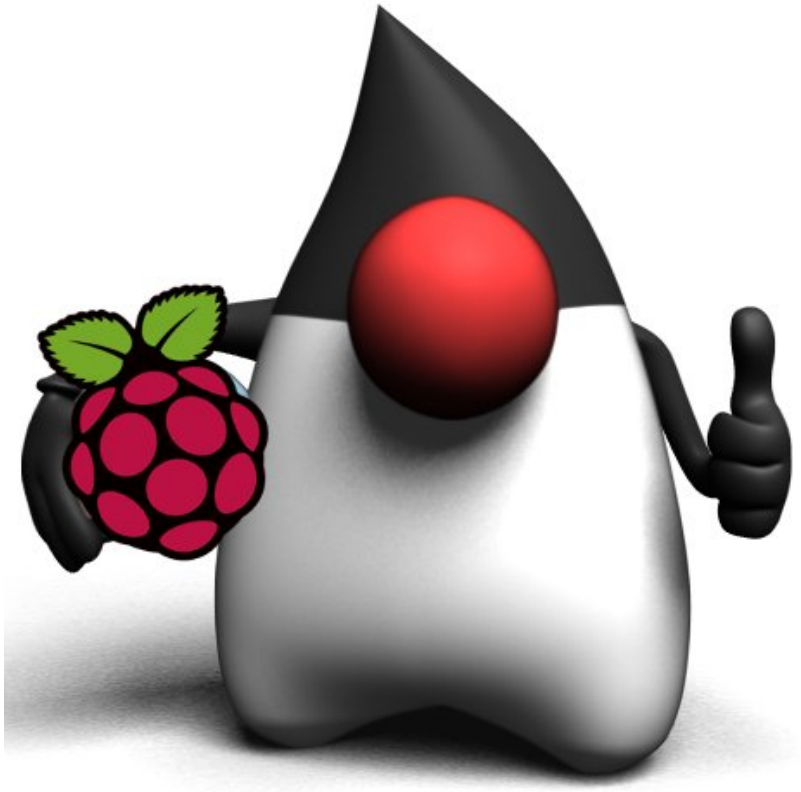
Raspberry Pi

Specification

- CPU: ARM 11 core running at 700MHz
  - Broadcom SoC package
  - Can now be overclocked to 1GHz (without breaking the warranty!)
- Memory: 512Mb
- I/O:
  - HDMI and composite video
  - 2 x USB ports (Model B only)
  - Ethernet (Model B only)
  - Header pins for GPIO, UART, SPI and I2C

Java™ ORACLE®

Java On The ARM and Raspberry Pi



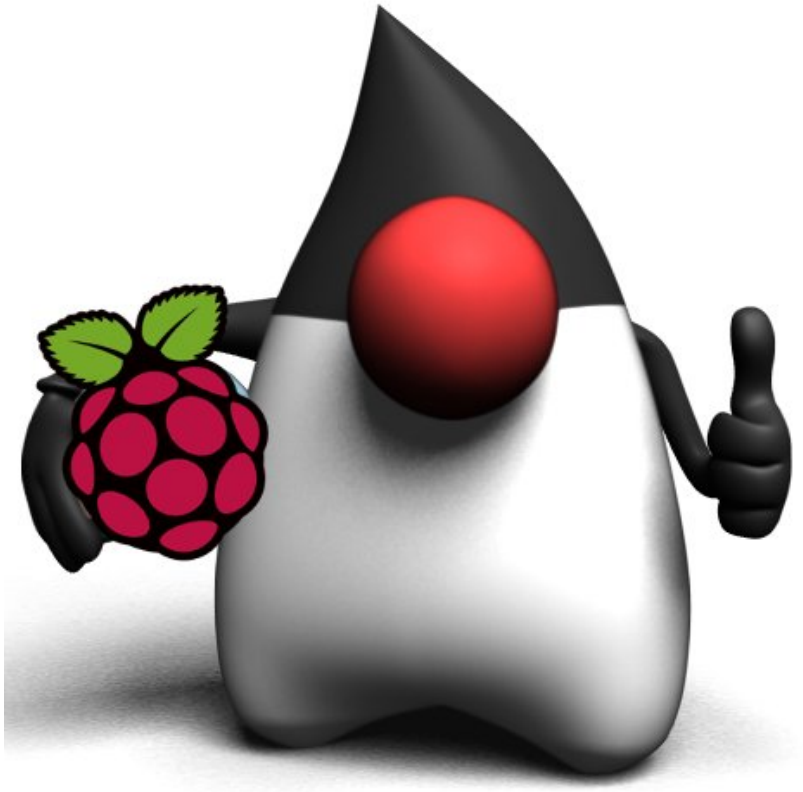
Java™ ORACLE®

Java Specifics For ARM

Floating Point Operations

- Despite being an ARMv6 processor it does include an FPU
  - FPU only became standard as of ARMv7
- FPU (Hard Float, or HF) is much faster than a software library
- Linux distros and Oracle JVM for ARM assume no HF on ARMv6
  - Need special build of both
  - Raspbian distro build now available (Based on Debian)
  - Oracle JVM released on Dec 2012!!! (Early access)

Extending the Raspberry Pi



Java™ ORACLE®

Using Java on the Raspberry Pi

Projects I've been work on:

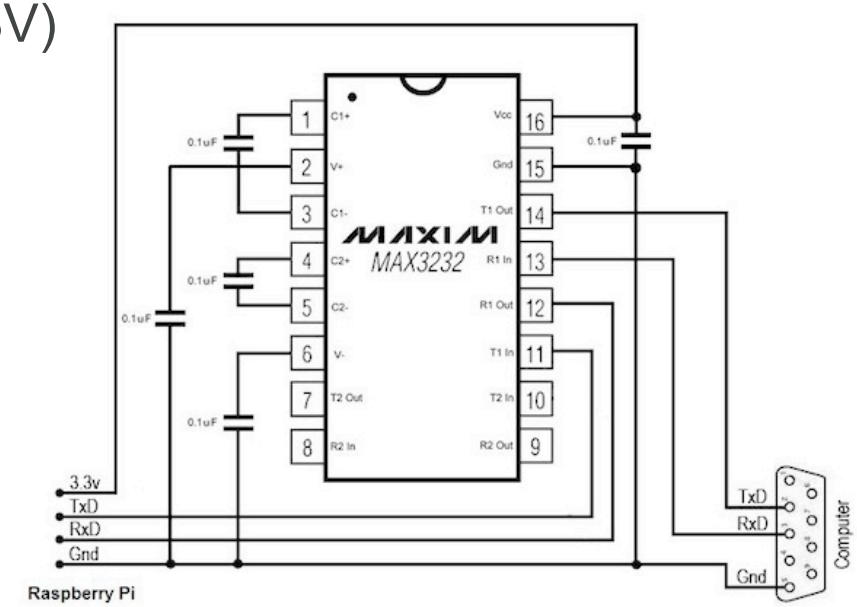
- Serial (TTL UART) DONE 
- USB DONE 
- GPIO DONE 
- I²C 

Lots of great ideas!!!

Java™ ORACLE®

Using The Serial Port

- UART provides TTL level signals (3.3V)
- RS-232 uses 12V signals
- Use MAX3232 chip to convert
- Use this for access to serial console



Java™ ORACLE®

USB & The OWI Robot Arm

Cheap and cheerful

- Comes with USB interface
  - Windows only driver
  - Recognized as USB device by Linux (`dev/bus/usb`)
- Use native code for control and JNI
- Simple control protocol
  - 3 bytes (1 = arm, 2 = base, 3 = light)
  - Combining movements requires some bit twiddling
  - Can only stop all motors, not individually

Java™ ORACLE®

Robot Arm Control

JNI Code

- Native C functions
  - Initialization of arm using libusb and appropriate device
  - Separate function for each control element
  - Compile to shared library
- Use JNI to generate header file appropriate to Java code usage
  - e.g. `native int arm_usb_init()`
  - Implement appropriate stub to call library
  - Compile to shared library
  - JNI is not easy to reuse

Robot Arm Control

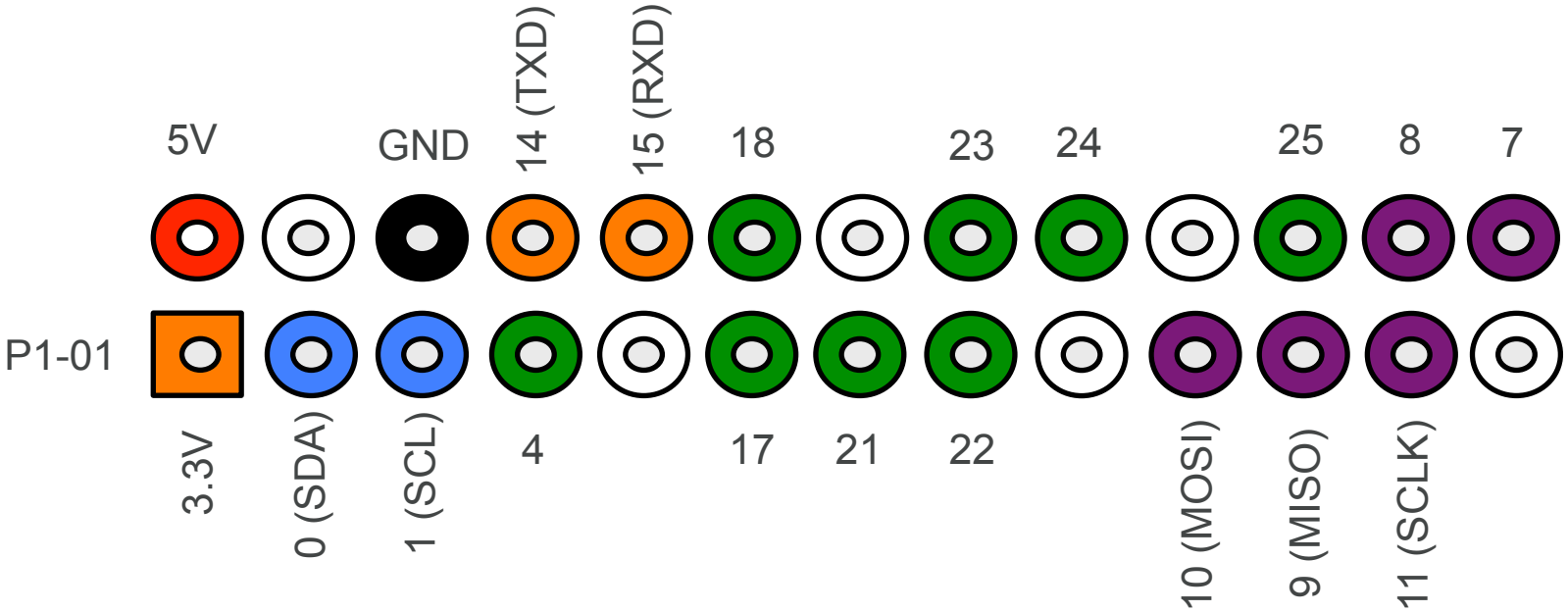
Java Code

- Java code is simple
  - Calibration required to determine time for specific movement

```
arm_gripper_move(OPEN);
uSleep(500);
arm_gripper_move(STOP);
uSleep(500);
arm_gripper_move(CLOSE);
uSleep(500);
arm_gripper_move(STOP);
```

Using The GPIO Lines

P1 Connector Layout

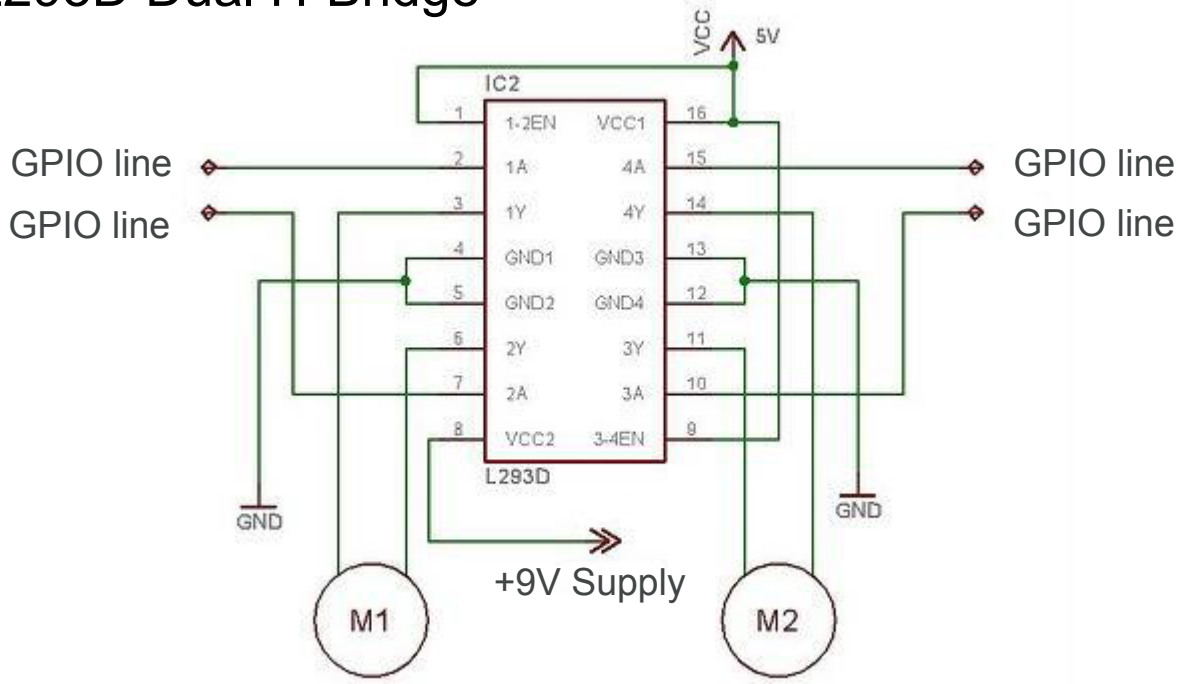


P2 Connector: Pin 1 = 3.3V Pin 7,8 = GND

Java™ ORACLE®

GPIO Example: LEGO Motors

Using L293D Dual H-Bridge



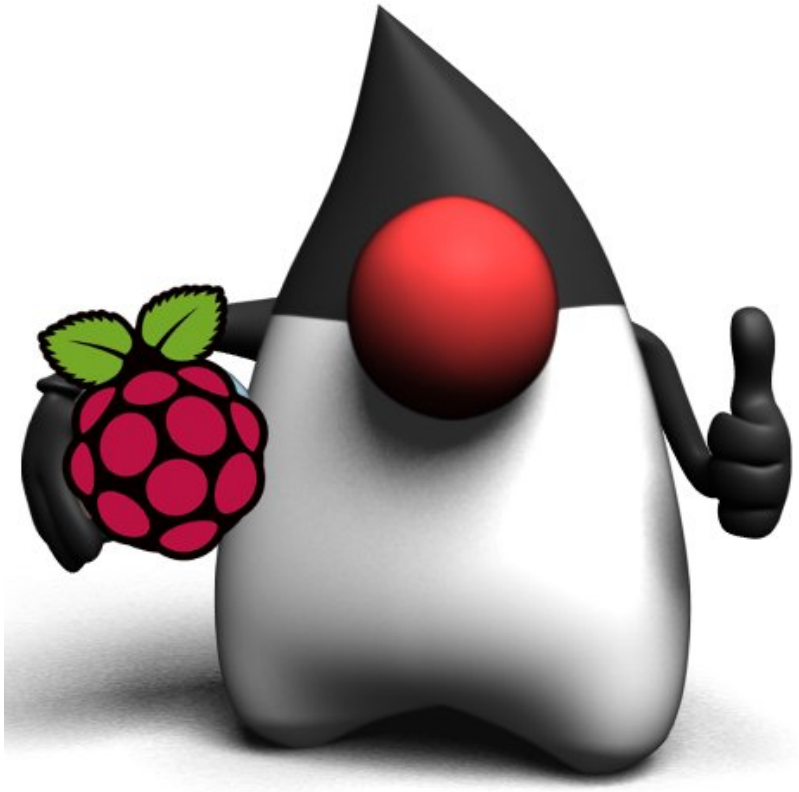
Hide The Magic Incantations With JNI

Simple Java Interface

- Access to `/dev/mem` needs root access
  - Could solve this by writing our own device driver

```
gpio_init();
gpio_pin_output(MOTOR_PIN_CLKWISE);
gpio_pin_output(MOTOR_PIN_ACLKWISE);

/* Turn clockwise */
gpio_pin_low(MOTOR_PIN_ACLKWISE);
gpio_pin_high(MOTOR_PIN_CLKWISE);
```

I²C in the Raspberry Pi



Java™ ORACLE®

I²C Overview

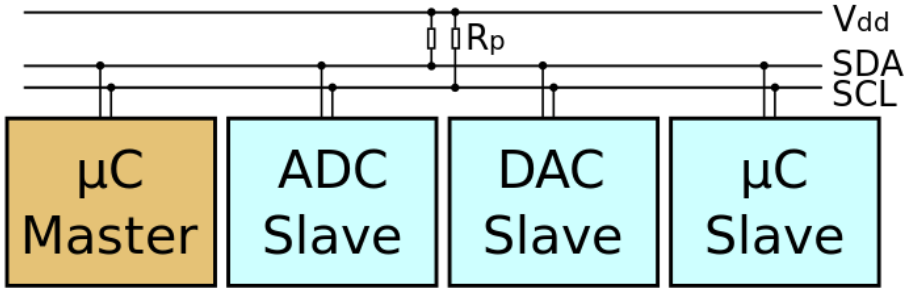
- Used for moving data simply and quickly from one device to another
- Serial Interface
- Synchronous:
  - The data is clocked along with a clock signal (SCL)
  - The clock signal controls when data is changed and when it should be read
  - Clock rate can vary, unlike asynchronous (RS-232 style) communications
- Bidirectional

I2C In Few Words…

- Inter-Integrated Circuit, ("two-wire interface")
- Multi-master serial single-ended computer bus
- Invented by Philips for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other electronic device.
- Uses two bidirectional open-drain lines
  - Serial Data Line (SDA) and
  - Serial Clock (SCL),
  - pulled up with resistors.
- Typical voltages used are +5 V or +3.3 V

I²C Design

- A bus with a clock (SCL) and data (SDA) lines
- 7-bit addressing.
- Roles:
  - Master: generates the clock and initiates communication with slaves
  - Slave: receives the clock and responds when addressed by the master
- Multi-Master bus
- Switch between master and slave allowed

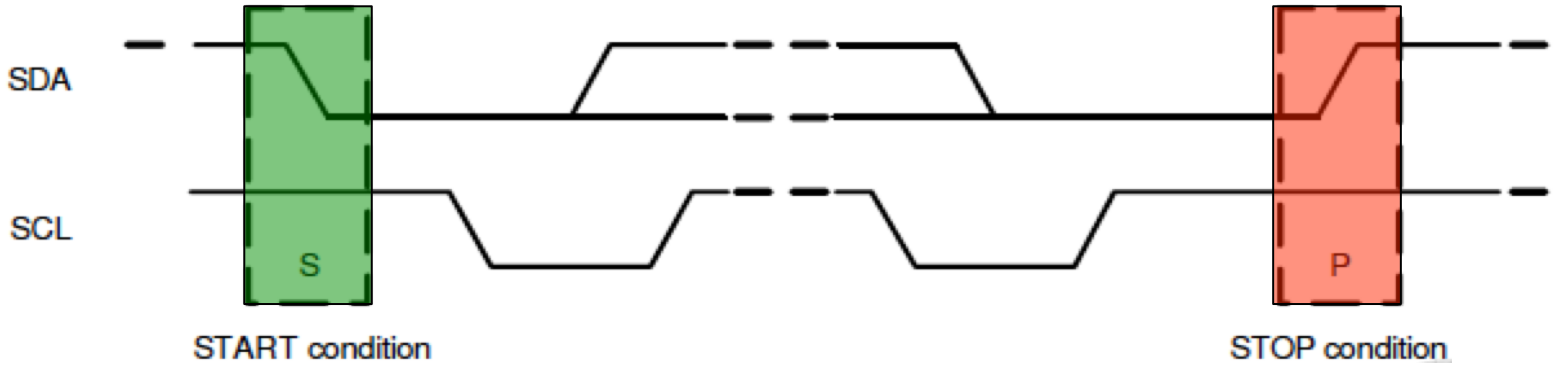


Java™ ORACLE®

I²C Communication Protocol

START (S) and STOP (P) Conditions

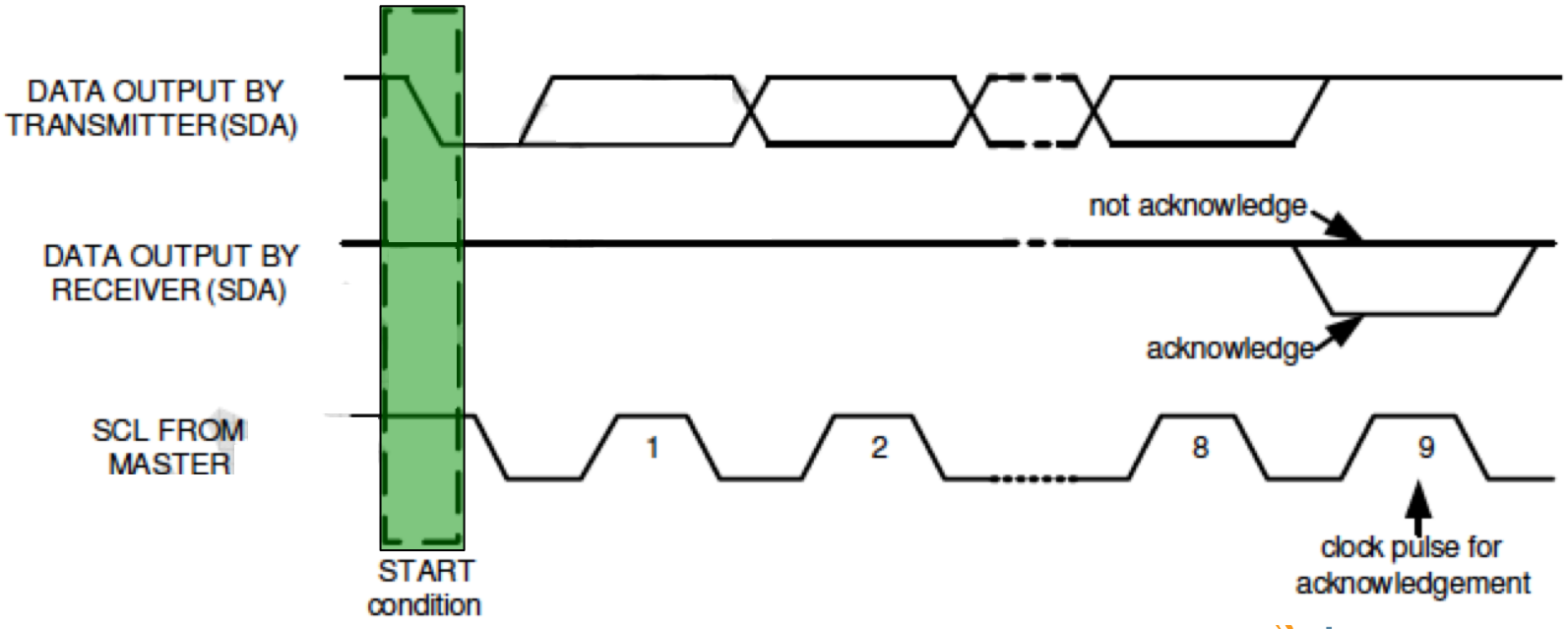
- Communication starts when master puts START condition (S) on the bus: HIGH-to-LOW transition of the SDA line while SCL line is HIGH
- Bus is busy until master puts a STOP condition (P) on the bus: LOW to HIGH transition on the SDA line while SCL is HIGH



I²C Communication Protocol

Data Format / Acknowledge

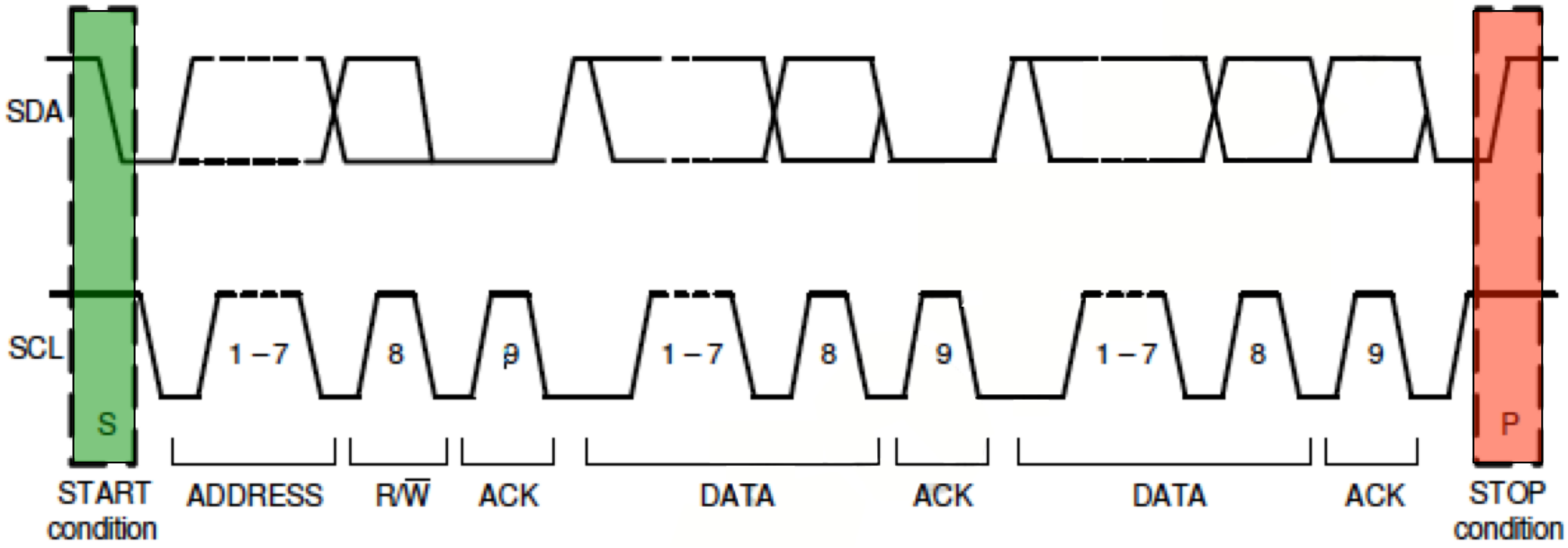
- I2C data bytes are 8-bits long
- Each byte transferred must be followed by acknowledge ACK signal



Java™ ORACLE®

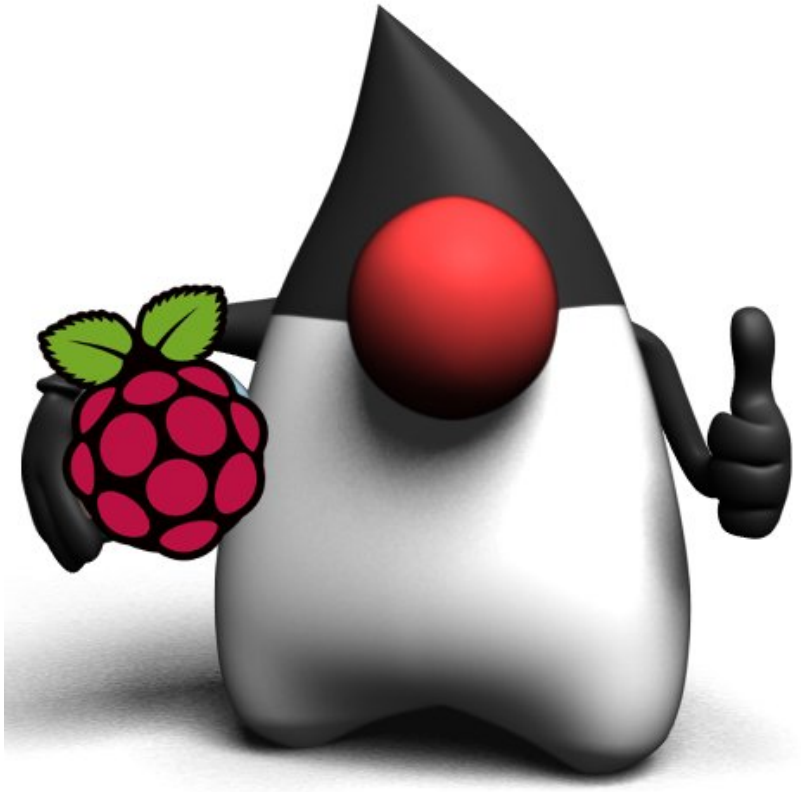
I²C Communication Protocol

Communications



Java™ ORACLE®

Enabling I²C on the Raspberry Pi



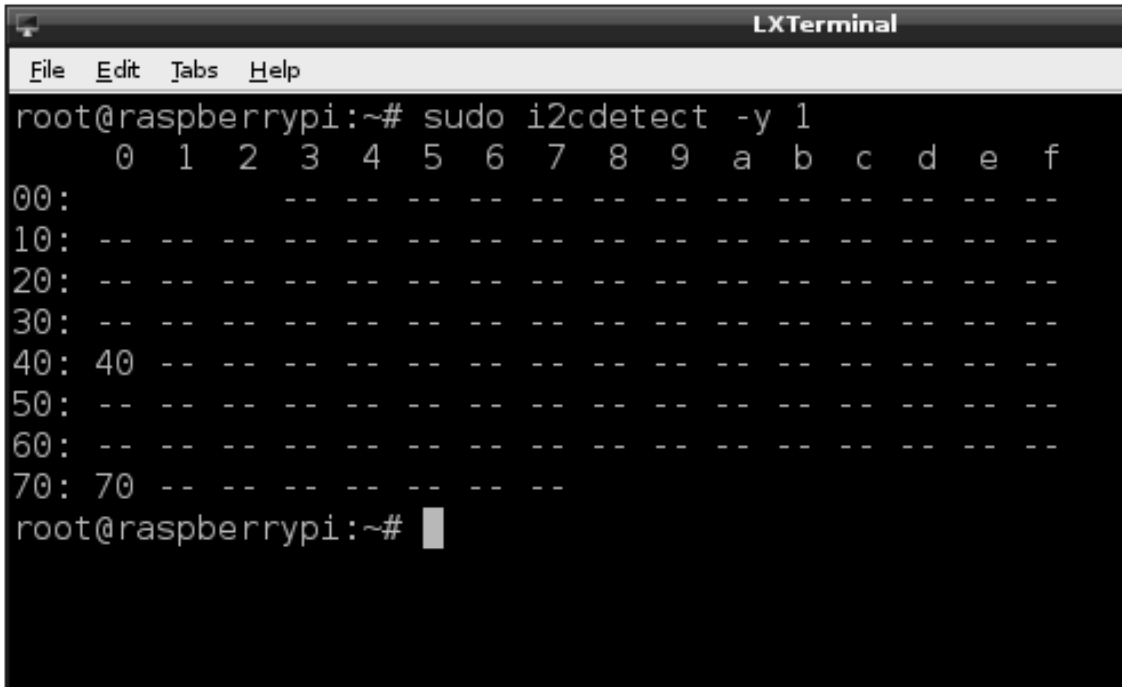
Java™ ORACLE®

I²C Not enabled!!!!

- /etc/modules
  - Add lines: i2c-bcm2708
    i2c-dev
- Install i2c-tools. Handy for detecting devices
  - sudo apt-get install python-smbus
  - sudo apt-get install i2c-tools
- /etc/modprobe.d/raspi-blacklist.conf comment out the lines
  - blacklist spi-bcm2708
  - blacklist i2c-bcm2708

I²C Not enabled!!!!

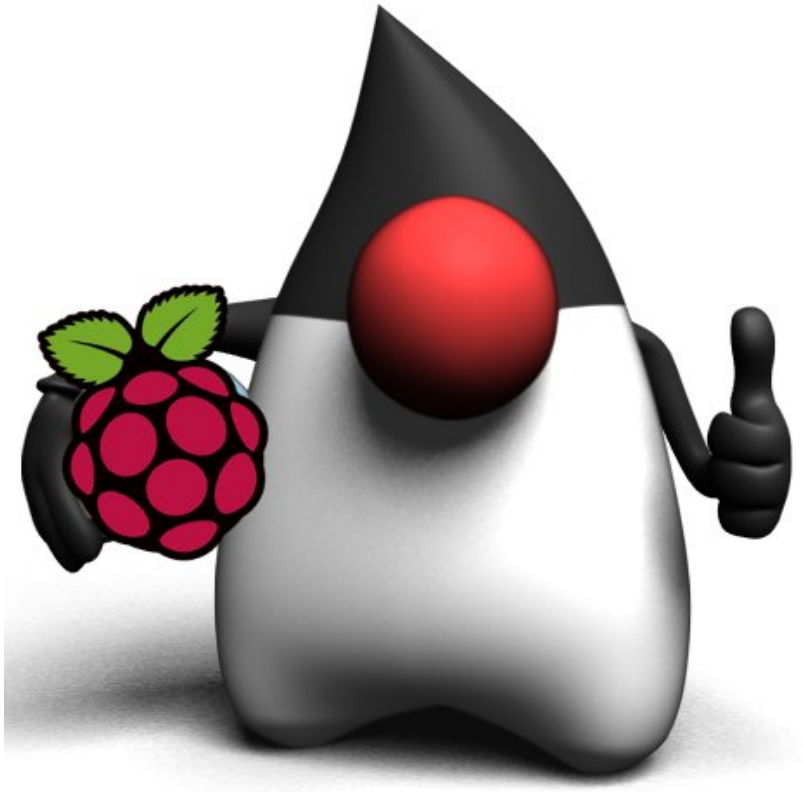
- Can you see your device?
  - sudo i2cdetect -y 1
  - Or
  - Sudo i2cdetect –y 0
    (for 256 Pi Model B)



I²C Verify…

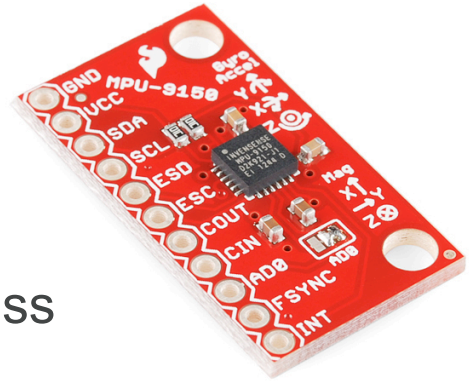
- You should have under /dev
  - spidev0.0
  - Spidev0.1
  - I2c-0
  - I2c-1
- If you don't see anything run
  - sudo modprobe i2c-dev

I²C Hardware



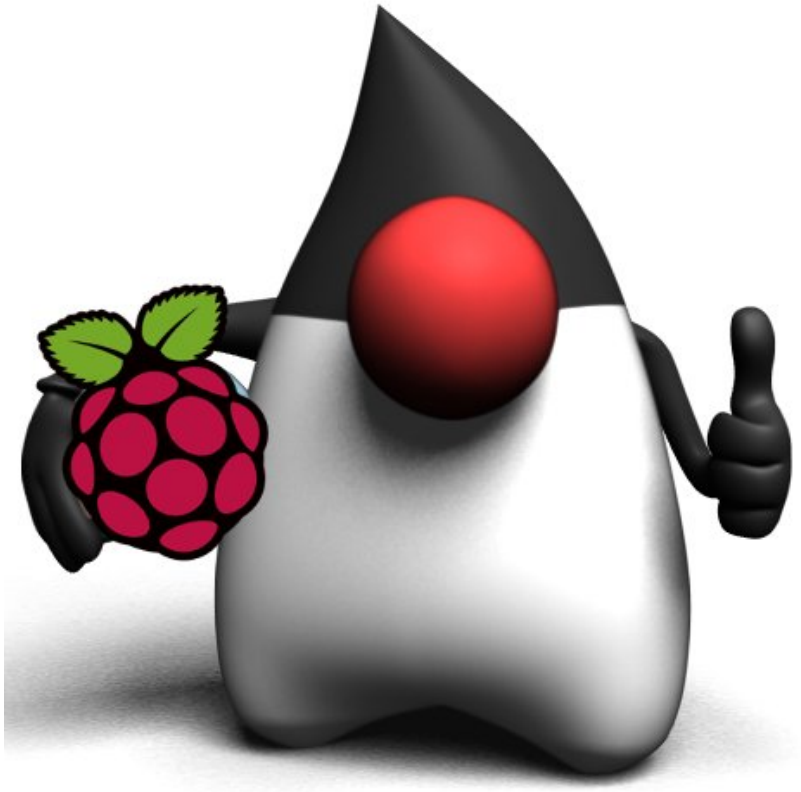
Java™ ORACLE®

MPU-9150 Breakout Board



- 9-axis MotionTracking device
- I²C serial host interface
- Solder jumper for switching LSB of the I2C address
  - Allow you to connect 2 of them
- Tri-Axis angular rate sensor (gyro)
- Tri-Axis accelerometer
- Tri-axis compass with a full scale range of ±1200μT
- VDD Supply voltage range of 2.4V–3.46V

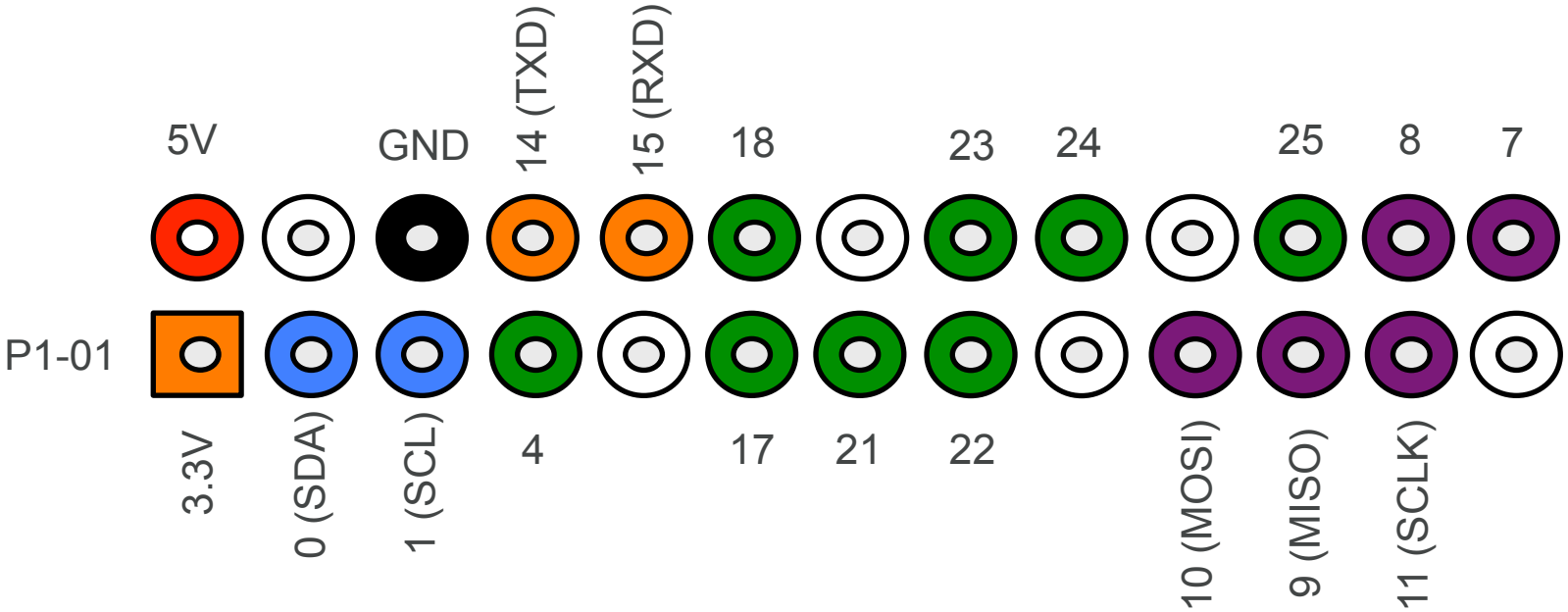
Connecting things



Java™ ORACLE®

Using I²C Lines

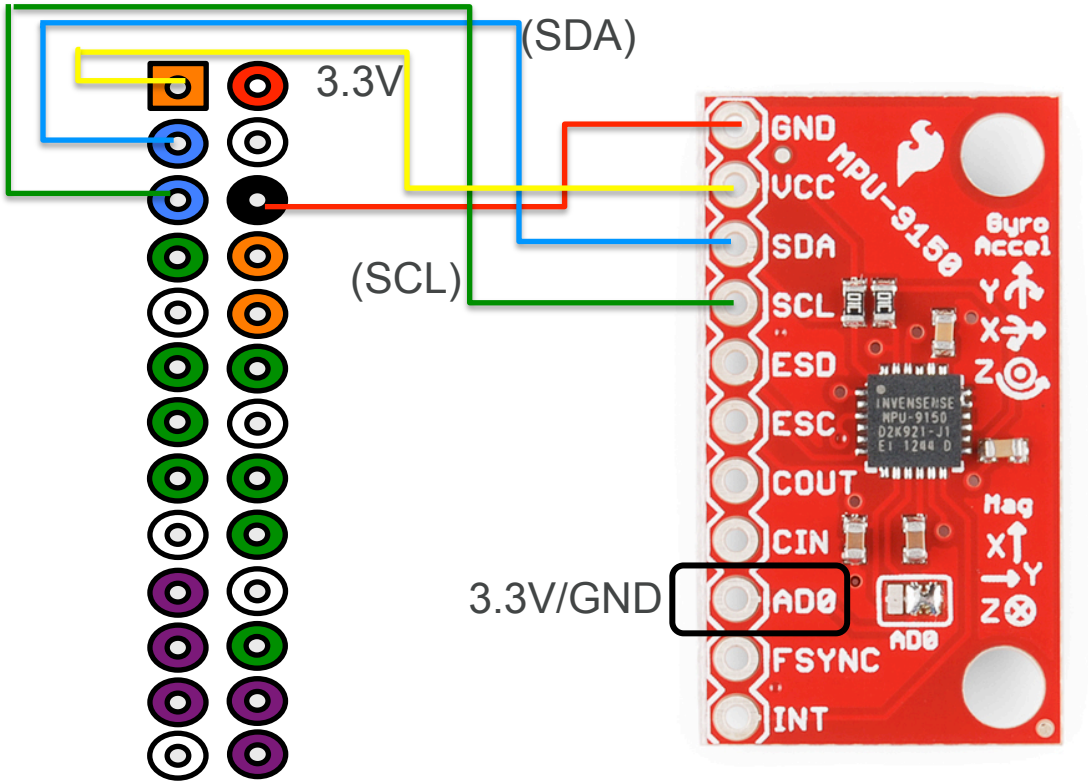
P1 Connector Layout



P2 Connector: Pin 1 = 3.3V Pin 7,8 = GND

Java™ ORACLE®

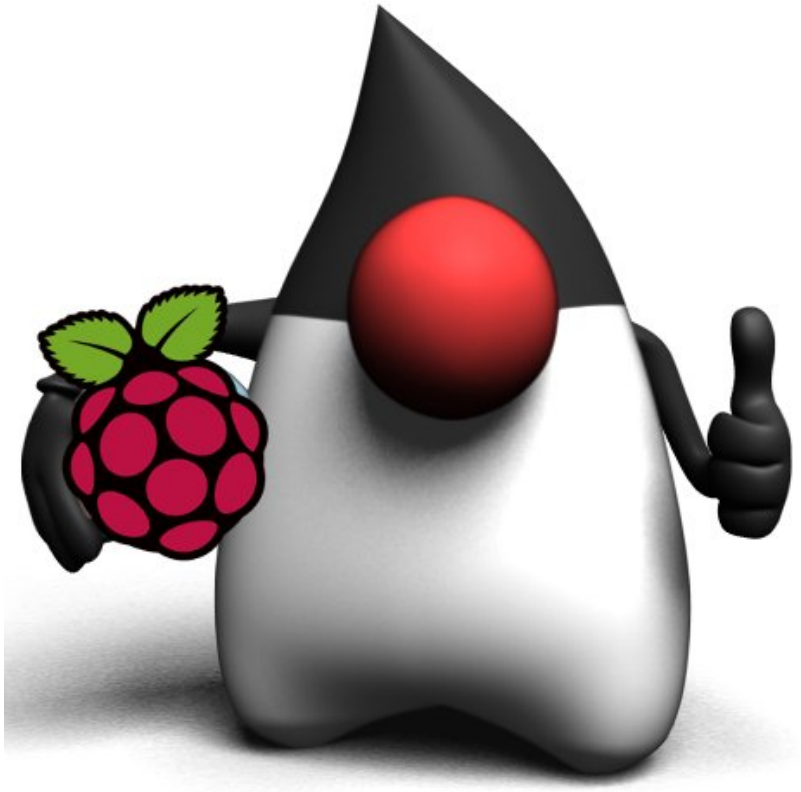
Pi Connector Layout



P2 Connector: Pin 1 = 3.3V Pin 7,8 = GND

Java™ ORACLE®

The Screen

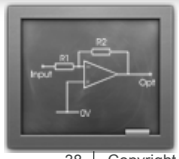


Java™ ORACLE®

Chalkboard Electronics Touchscreen

- 10" or 7" Form Factor
- Connects via HDMI/USB
- Tested with JavaFX 8
- 10% Exclusive Discount:

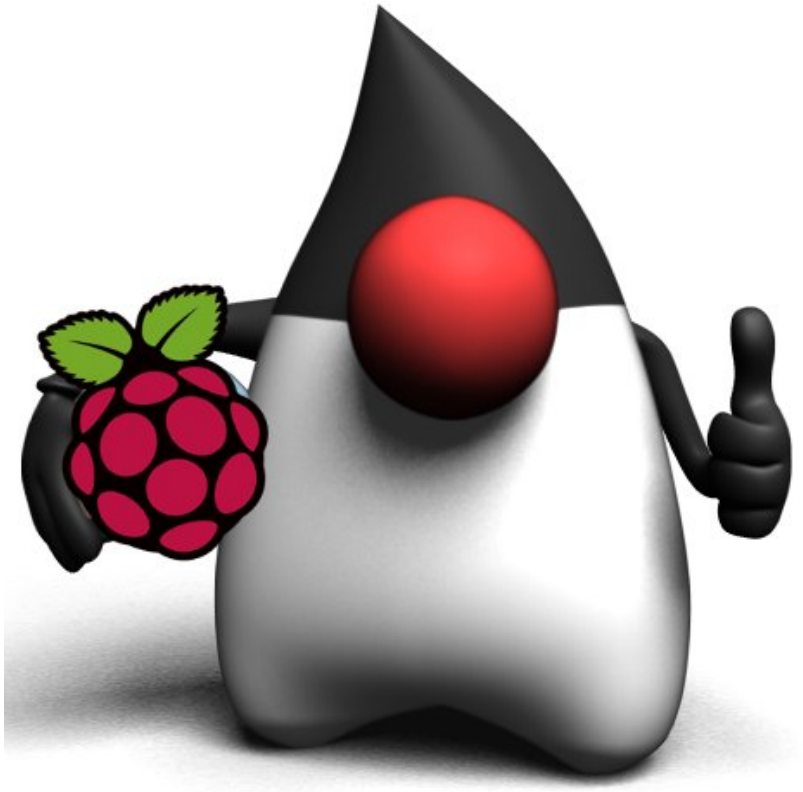
  G1F0U796Z083



Chalkboard Electronics

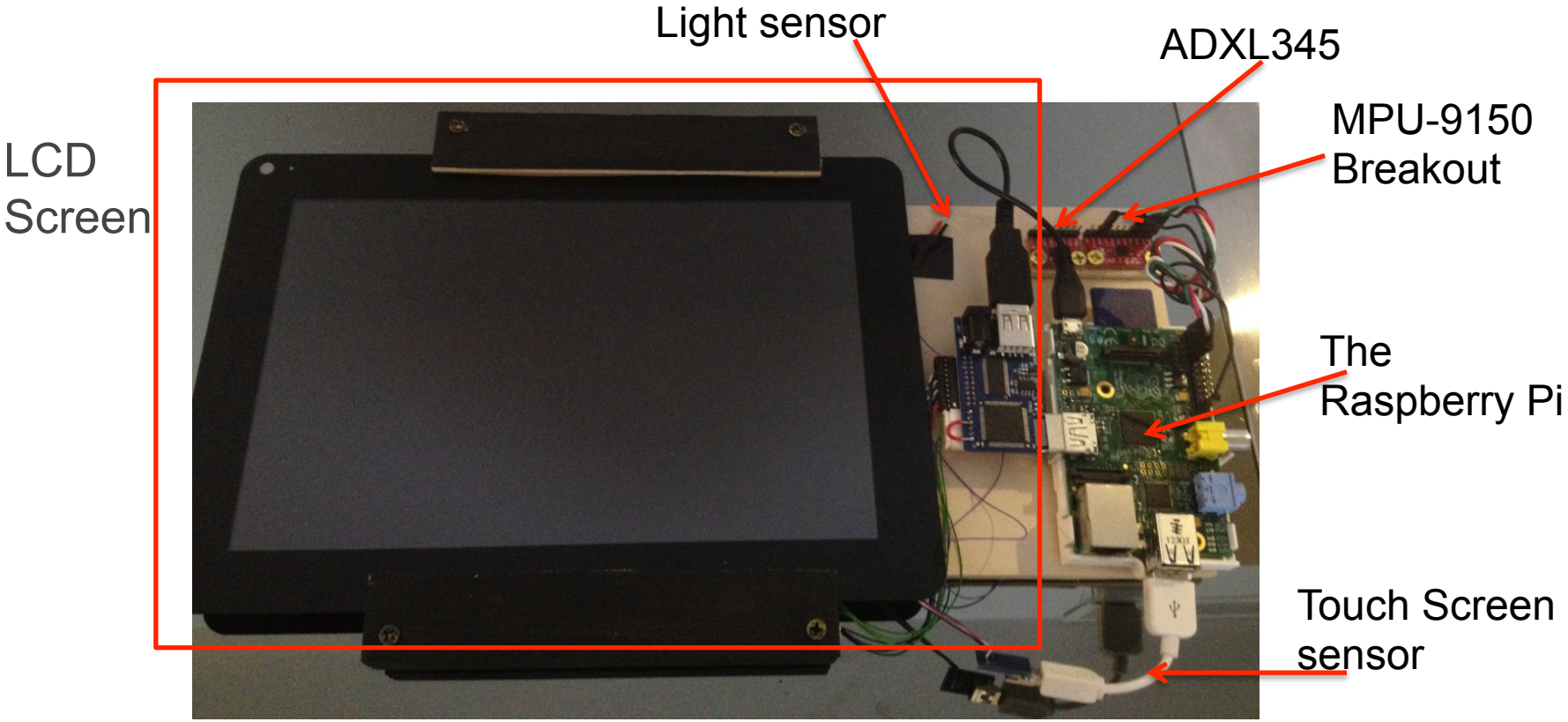
JavaOne ORACLE

The Wood Work…



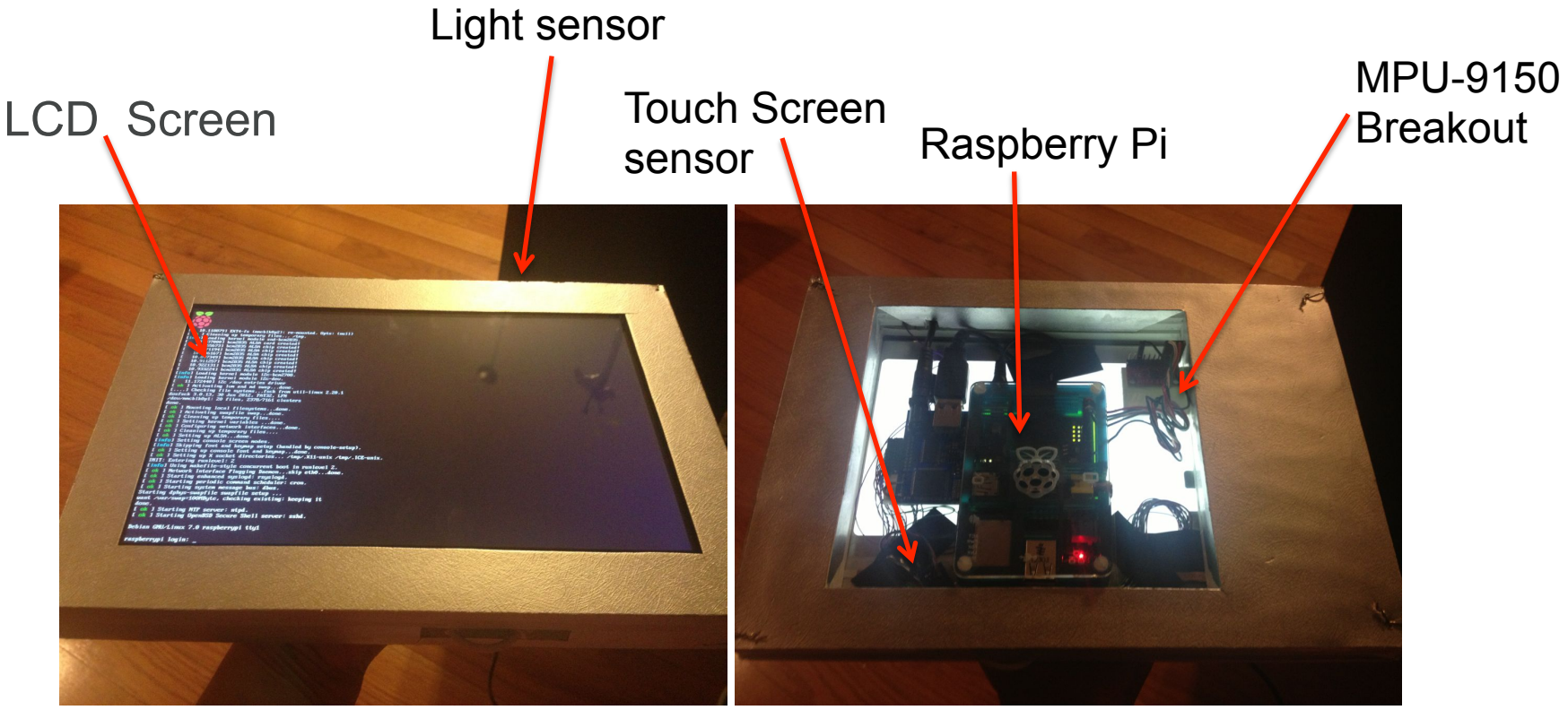
Java™ ORACLE®

Demo Setup (First Attempt)



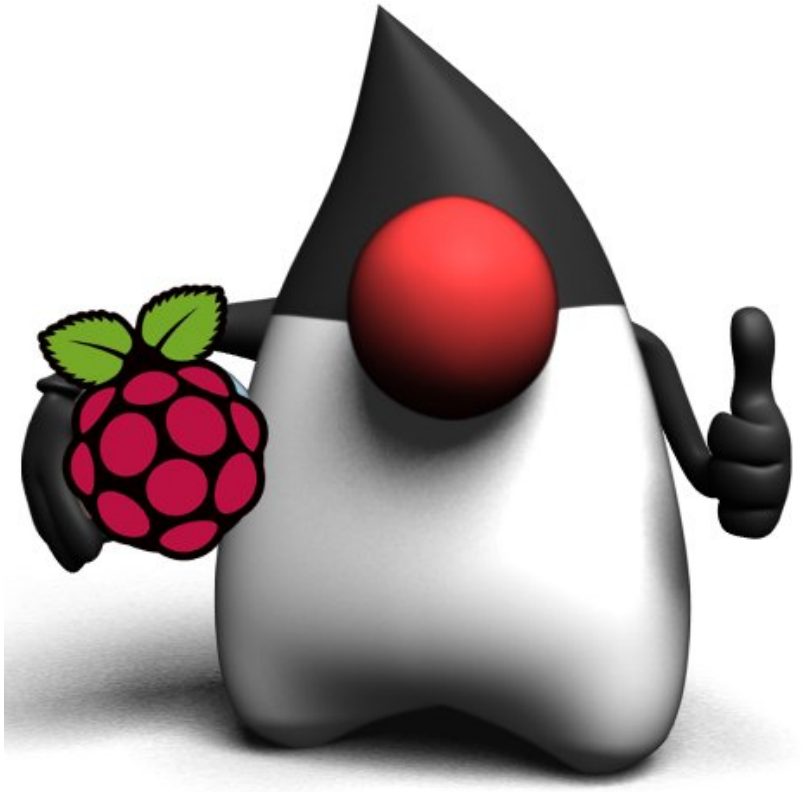
Java™ ORACLE®

Demo Setup (Second Attempt)



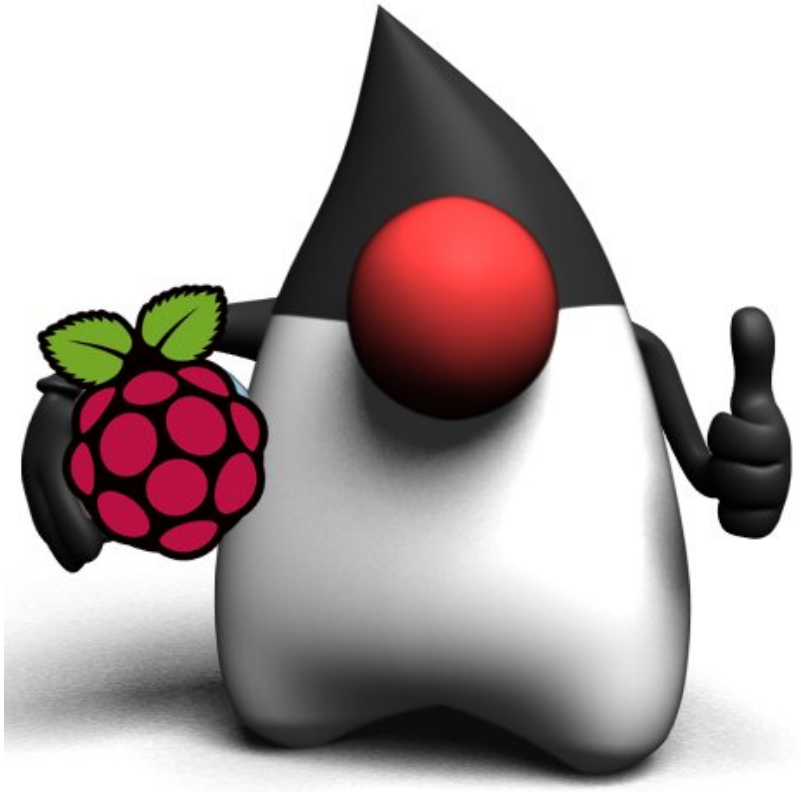
Java™ ORACLE®

The Software



Java™ ORACLE®

Java – I²C and the Raspberry Pi



Java™ ORACLE®

WiringPi Native Library

- Library written in C
- Inspired on Arduino's Wiring package
- Core of the input and output for the Raspberry Pi
- Provides access to:
  - GPIOs pin, I²C and SPI interface, UART lines
- Includes a command-line utility ***gpio:***
  - Allow to program and setup the GPIO pins
  - Read and write the pins and even use it to control them from shell scripts.

Pi4J

- Provide a bridge between the native libraries (WiringPi) and Java for full access to the Raspberry Pi.
- The Team:
  - Robert Savage
  - Chris Walzl
- Raspberry Pi Community Forums
- Pi4J Google Groups Discussion Forum
  - https://groups.google.com/forum/#!forum/pi4j
- @Pi4J on Twitter
- Pi4J Issue Tracking
  - https://github.com/Pi4J/pi4j/issues

Pi4J: Features (1/2)

- Wrapper classes for direct access to WiringPi Library from Java
- Export & unexport GPIO pins
- Configure GPIO pin direction
- Configure GPIO pin edge detection
- Control/write GPIO pin states
- Pulse GPIO pin state
- Read GPIO pin states
- Listen for GPIO pin state changes (interrupt-based; not polling)
- Automatically set GPIO states on program termination (GPIO shutdown)

Pi4J: Features (2/2)

- Triggers for automation based on pin state changes
- Send & receive data via RS232 serial communication
- I²C Communication
- SPI Communication
- Extensible GPIO Provider interface to add GPIO capacity via expansion boards
- Access system information and network information from the Raspberry Pi
- Wrapper classes for direct access to WiringPi Library from Java

Installing and Using Pi4J (1/2)

- Get the APIs

```
wget http://pi4j.googlecode.com/files/pi4j-0.0.5.deb
```

- Install

```
sudo dpkg -i pi4j-0.0.5.deb
```

  - Check the source files at:

```
/opt/pi4j/lib
/opt/pi4j/examples
```

- Include the Pi4J lib folder in the classpath:

```
javac -classpath .:classes:/opt/pi4j/lib/'*' ...
```

Installing and Using Pi4J (2/2)

- Include Pi4J lib folder in the classpath:

```
sudo java -classpath .:classes:/opt/pi4j/lib/'*' ...
```

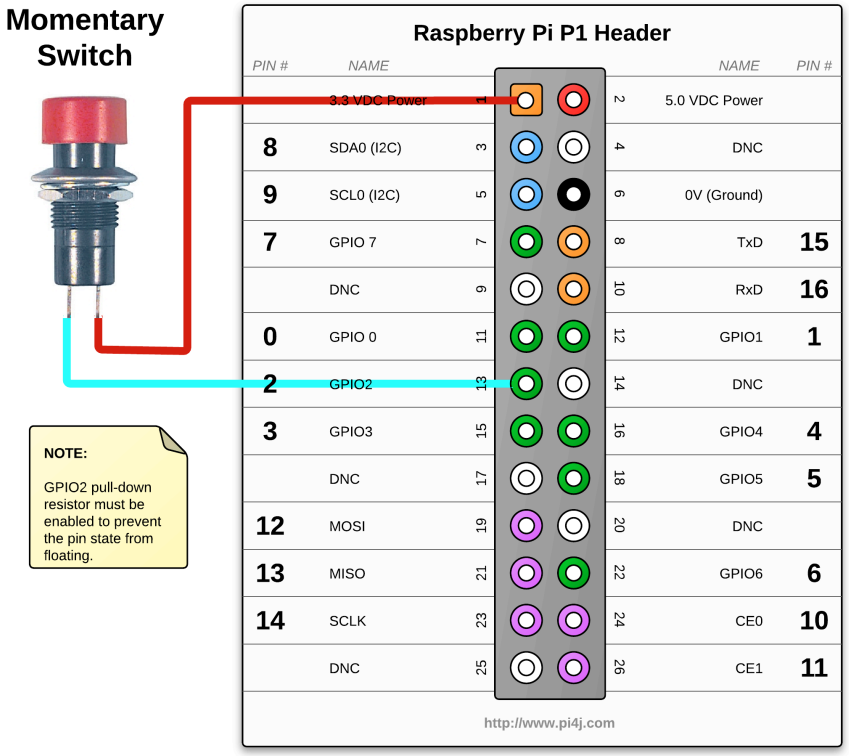
- Build all demos:

```
cd /opt/pi4j/examples
./build
```

- Uninstall Pi4J

```
sudo dpkg -r pi4j
```

GPIO State Listener Example using Pi4J



Java™ ORACLE®

GPIO State Listener Example using Pi4J

```
public class ListenGpioExample {
  public static void main(String args[]) throws InterruptedException {
    final GpioController gpio = GpioFactory.getInstance();
    final GpioPinDigitalInput myButton =
    gpio.provisionDigitalInputPin(RaspiPin.GPIO_02,
                                  PinPullResistance.PULL_DOWN);
    myButton.addListener(new GpioPinListenerDigital() {
      @Override
      public void handleGpioPinDigitalStateChangeEvent(
                           GpioPinDigitalStateChangeEvent event) {
        System.out.println("PIN STATE: " + event.getPin() + " = " +
                           event.getState());
      }
    });
    for (;;) {Thread.sleep(500);}
    // stop all GPIO activity/threads
    // gpio.shutdown();   <--- implement this method call if you wish to
    //terminate the Pi4J GPIO controller
    }
}
```

I2C with Pi4J (1/3)

```
public Sensors() {
  try {
    //get i2c bus
    bus = I2CFactory.getInstance(I2CBus.BUS_1);
    //get device itself
    device = bus.getDevice(0x68);
    //start sensing, using config registries 6B  and 6C
    device.write(0x6B, (byte) 0b00000000);
    device.write(0x6C, (byte) 0b00000000);
    //config gyro
    device.write(0x1B, (byte) 0b00011000);
    //config accel
     device.write(0x1C, (byte) 0b00000100);
    startReading();
  catch (IOException e) {System.out.println(e.getMessage()); }
}
```

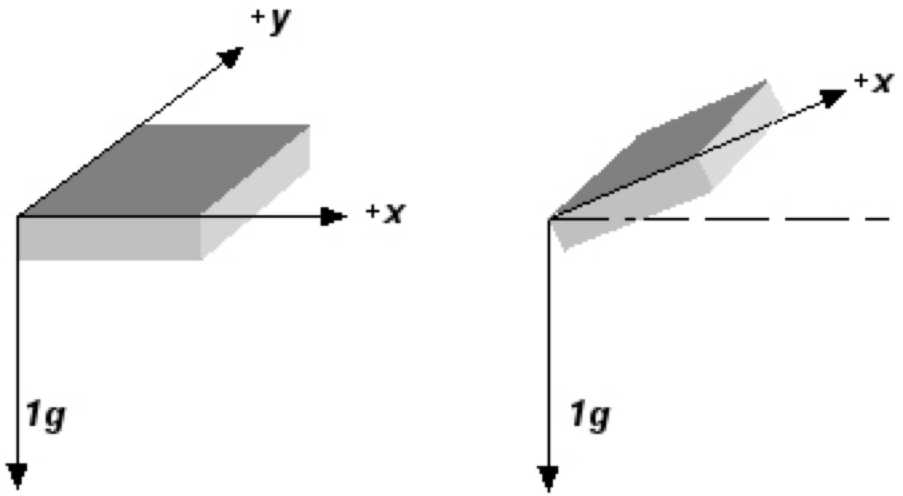
I²C with Pi4J (2/3)

```
private void readingSensors() throws IOException {
        bytes = new byte[6 + 2 + 6];
        DataInputStream gyroIn;
        ...
        while (true) {
            int r = device.read(0x3B, bytes, 0, bytes.length);
            if (r != 14) {   //14 registries to be read,
                             //6 for gyro, 6 for accel and 2 for temp
                System.out.println("Error reading,<" + bytes.length);
            }
            gyroIn = new DataInputStream(new ByteArrayInputStream(bytes));
            accelX = gyroIn.readShort();
            accelY = gyroIn.readShort();
            accelZ = gyroIn.readShort();
            temp = gyroIn.readShort();
            gyroX = gyroIn.readShort();
            gyroY = gyroIn.readShort();
            gyroZ = gyroIn.readShort();
            ...
```

I2C with Pi4J (3/3)

```
    tempX = (float) accelX / SENSITIVITY;
    accelX_G = (tempX > 1) ? 1 : ((tempX < -1) ? -1 : tempX);
    tempY = (float) accelY / SENSITIVITY;
    accelY_G = (tempY > 1) ? 1 : ((tempY < -1) ? -1 : tempY);
    tempZ = ((float) accelZ / SENSITIVITY) * (-1); //sensor upsidedown
    accelZ_G = (tempZ > 1) ? 1 : ((tempZ < -1) ? -1 : tempZ);
    myDevicePos.setAccelValues(accelX_G, accelY_G, accelZ_G);

    gyroXdeg = gyroX * (2000d / (double) Short.MAX_VALUE);
    gyroYdeg = gyroY * (2000d / (double) Short.MAX_VALUE);
    gyroZdeg = gyroZ * (2000d / (double) Short.MAX_VALUE);
    myDevicePos.setGyroValues(gyroXdeg, gyroYdeg, gyroZdeg);

    double tempC = ((double) temp / 340d) + 35d;
     analyzeData();
     try {
       Thread.sleep(100);
     } catch (InterruptedException ex) {
        Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE,null,ex);
}}}
```


Accelerometer Readings

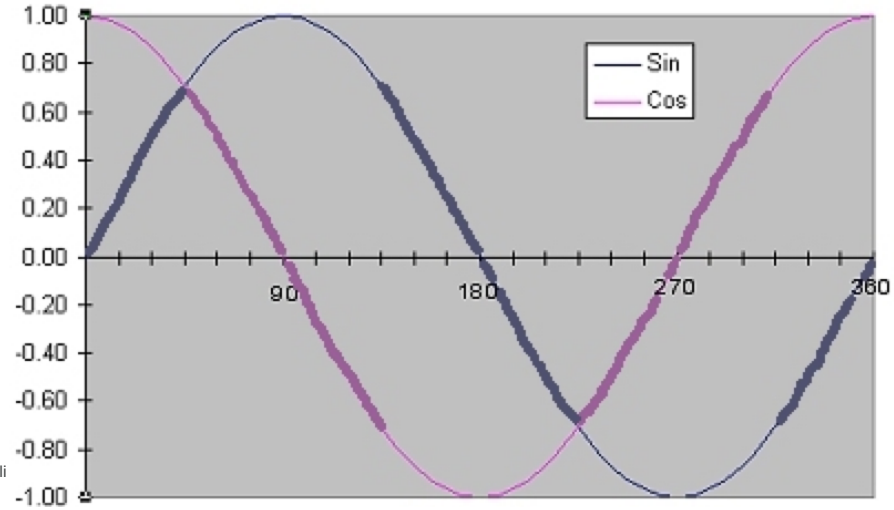


Tilt Angle using One Axis

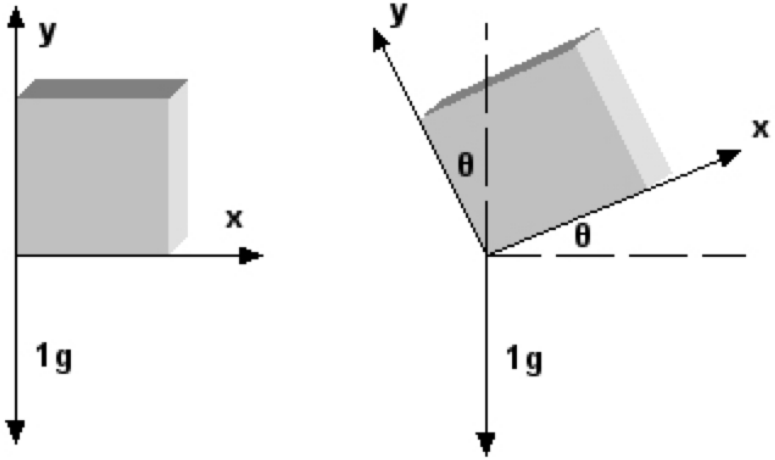
Sin-1 (a)

Measuring Tilt Angle with 2 Axis

- The component of gravity acting on the x axis is a sine function whilst that acting on the y axis is a cosine.
- When the sensitivity of the x axis starts dropping off after 45 degrees of tilt, the sensitivity of the y axis is increasing.
- Bold parts of each line show the area of most sensitivity.
- Combine x and y for better accuracy.



Measuring Tilt Angle with 2 and 3 Axis



$$\frac{x}{y} = \tan\theta$$

$$Ax = \arctan\left(\frac{X}{\sqrt{Y^2+Z^2}}\right)$$

$$Ay = \arctan\left(\frac{Y}{\sqrt{X^2+Z^2}}\right)$$

Accelerometer Readings

```
public void setAccelValues(float accelX, float accelY, float accelZ) {

    simpleAngles[0] = Math.toDegrees(Math.asin(accelX));
    simpleAngles[1] = Math.toDegrees(Math.asin(accelY));
    simpleAngles[2] = Math.toDegrees(Math.asin(accelZ));

    angleXProperty.setValue(Math.toDegrees(Math.atan(accelX /
                   Math.sqrt((accelY * accelY) + (accelZ * accelZ)))));
    angleYProperty.setValue(Math.toDegrees(Math.atan(accelY /
                   Math.sqrt((accelX * accelX) + (accelZ * accelZ)))));
}
```

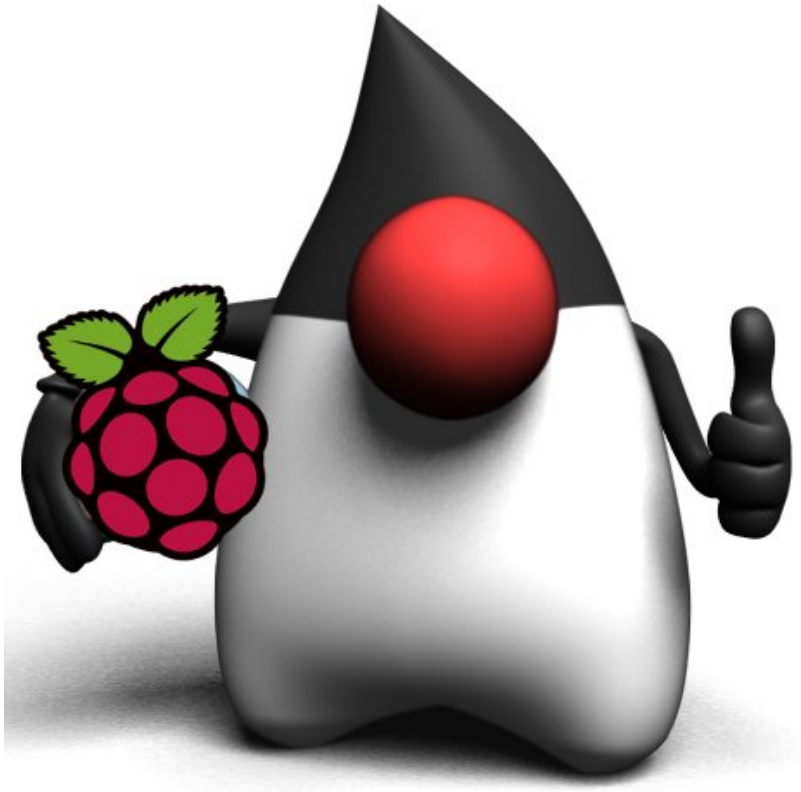
Gyroscope Readings (1/2)

```
public void setGyroValues(double gyroX, double gyroY, double gyroZ) {

   xRotation = (gyroX < -Sensors.rotationPeak) ?
               ((gyroX < -Sensors.rotationFastPeak) ?
                                ROLL_RIGHT_FAST : ROLL_RIGHT):
               ((gyroX > Sensors.rotationPeak) ?
                 ((gyroX > Sensors.rotationFastPeak) ?
                                ROLL_LEFT_FAST : ROLL_LEFT):
               NONE_DIR);

   yRotation = (gyroY > Sensors.rotationPeak) ?
               ((gyroY > Sensors.rotationFastPeak) ?
                                PITCH_BACKWARD_FAST : PITCH_BACKWARD):
               ((gyroY < -Sensors.rotationPeak) ?
                 ((gyroY < -Sensors.rotationFastPeak) ?
                            PITCH_FORWARD_FAST : PITCH_FORWARD) :
               NONE_DIR);
        ...
```


Gyroscope Readings (2/2)

```
    ...
    zRotation = (gyroZ > Sensors.rotationPeak) ?
                ((gyroZ > Sensors.rotationFastPeak) ?
                            ROTATING_FORWARD_FAST : ROTATING_FORWARD):
                ((gyroZ < -Sensors.rotationPeak) ?
                 ((gyroZ < -Sensors.rotationFastPeak) ?
                           ROTATING_BACKWARD_FAST : ROTATING_BACKWARD):
                NONE_DIR);
}
```

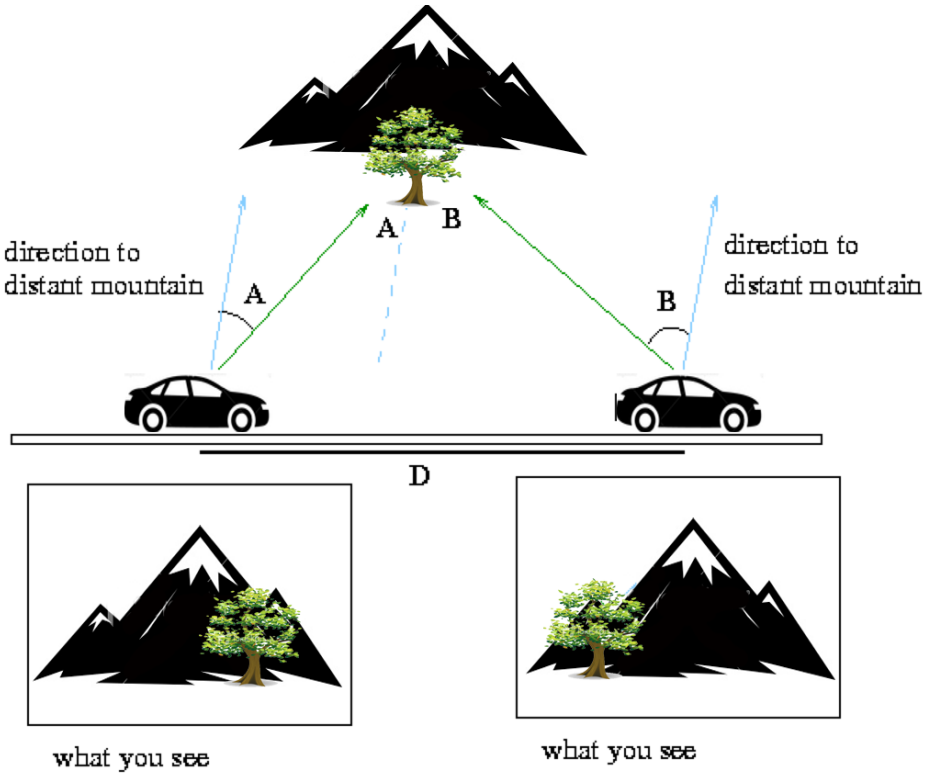
The User Interface: Parallax Effect



Java™ ORACLE®

The Parallax Principle

- Scrolling technique in computer graphics, wherein background images move by the camera slower than foreground images, creating an illusion of depth in a 2D video game and adding to the immersion.
- We see it everyday!

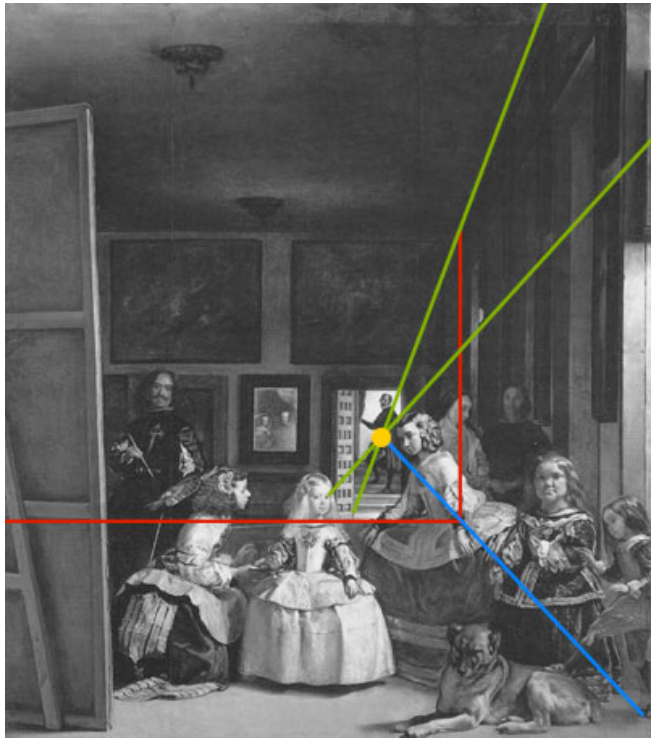


Java™ ORACLE®

Finding the Vanishing Point

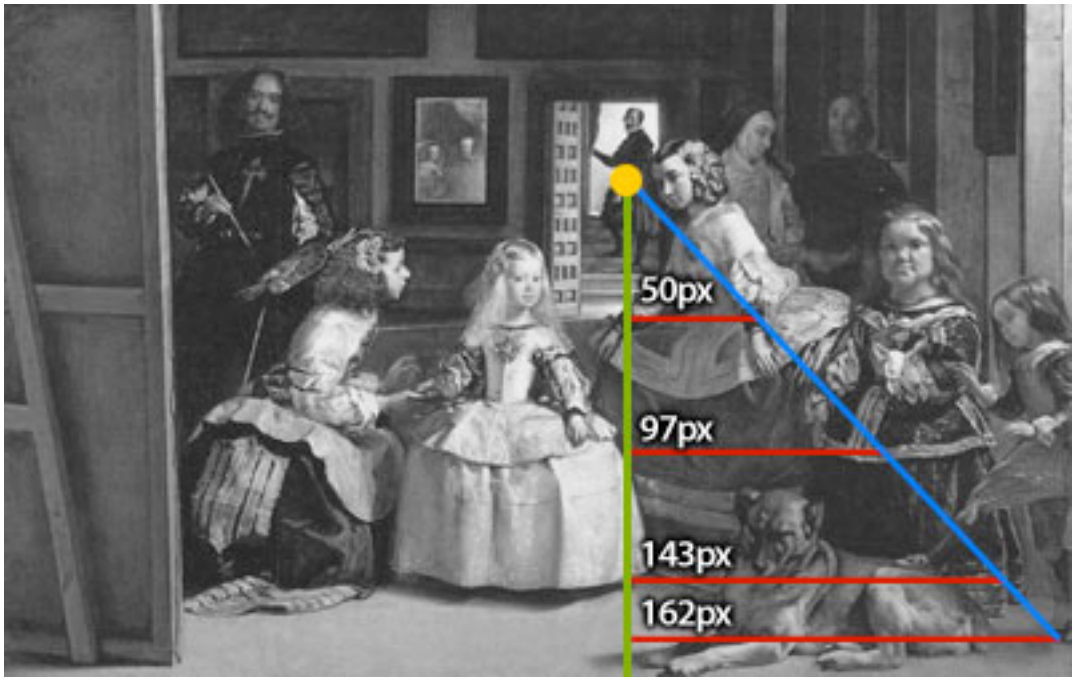
Roman Cortes

- Trace 2 lines following the 3d parallel lines (right wall)
- The crossing is the vanishing point.
- Trace two lines following the right and bottom corners of the back wall
- A line crossing the vanishing point and the intersection of the two red lines defines the corner of the ground with the right wall.



Java™ ORACLE®

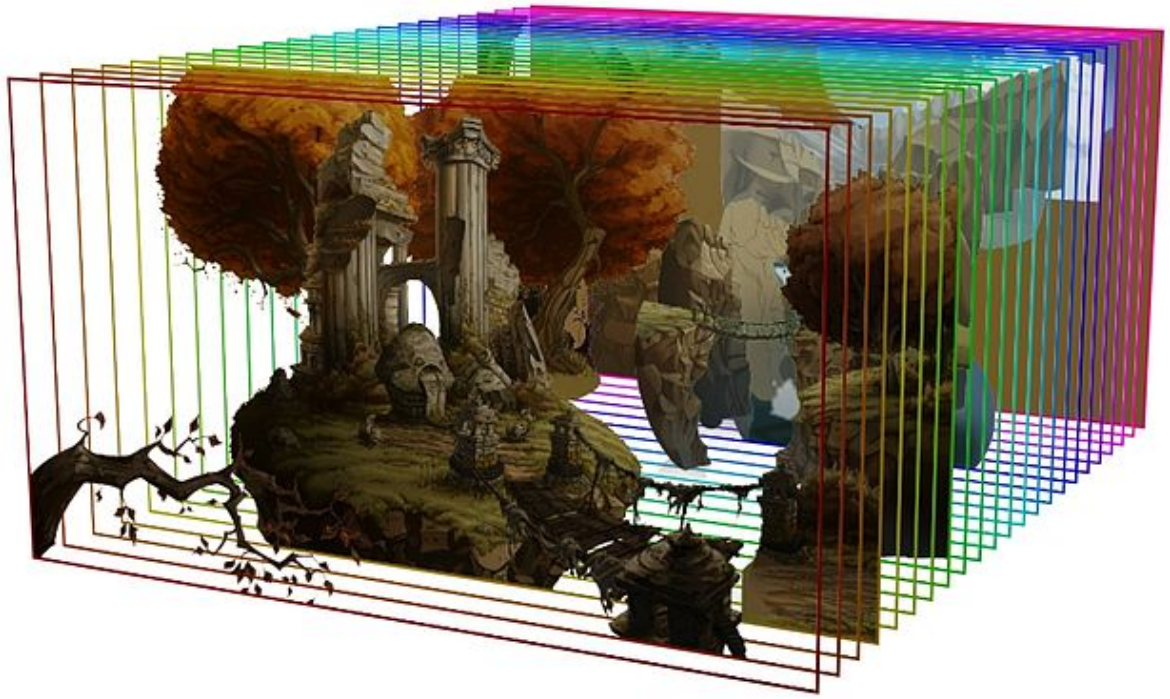
Layer's Movement Speed



Java™ ORACLE®

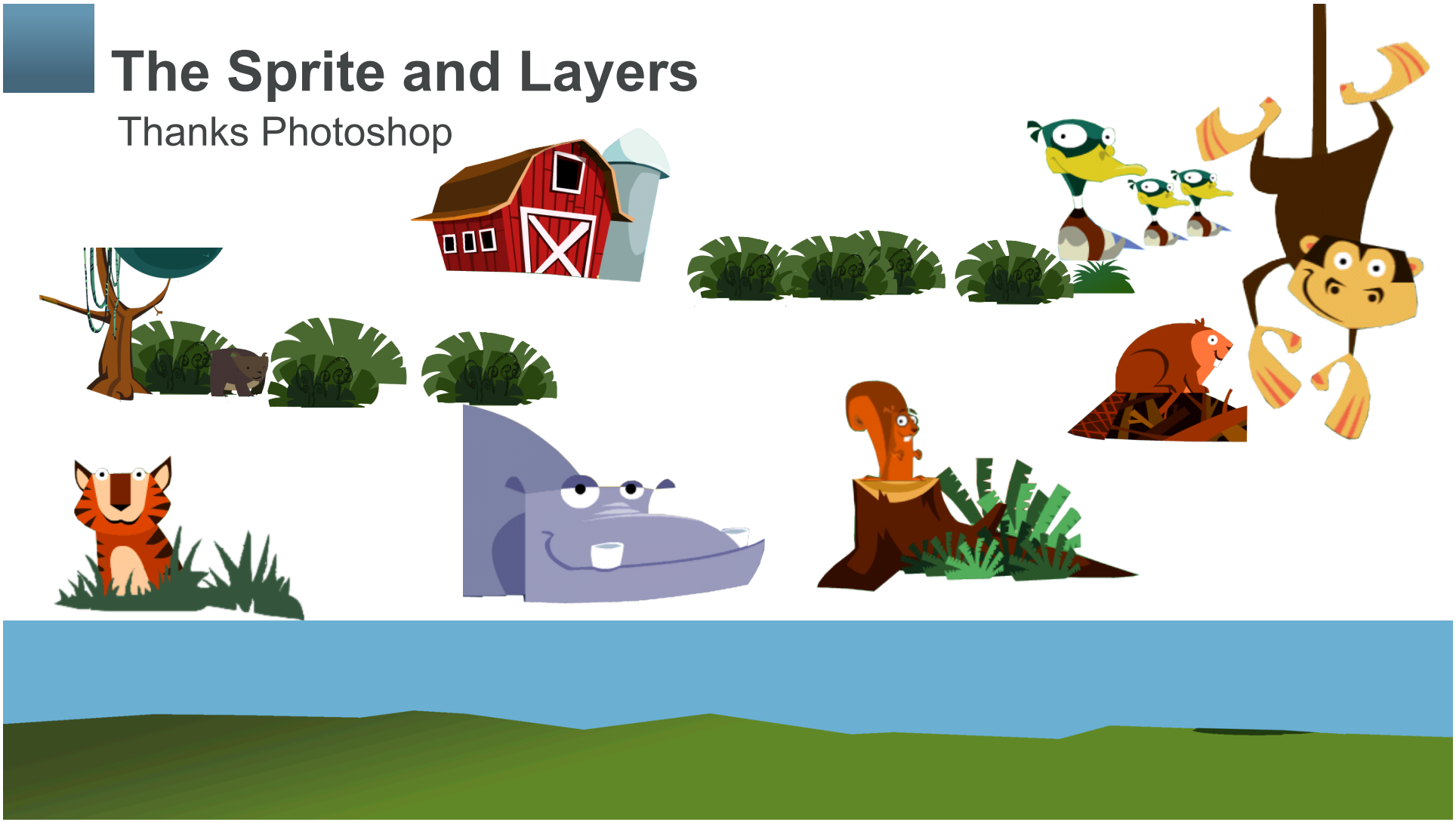
The more layers… the better!

Thanks Photoshop



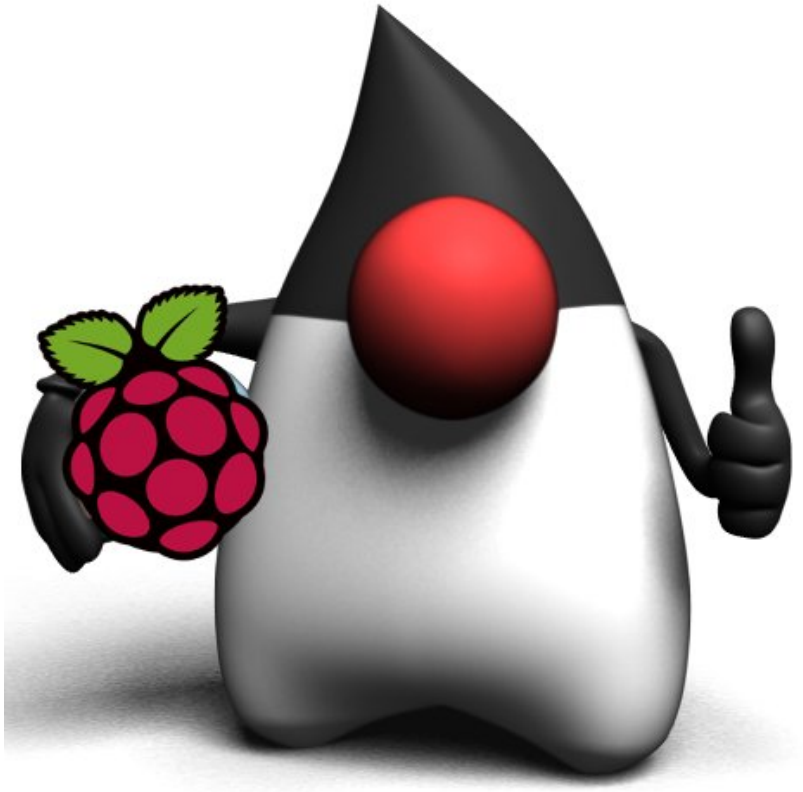
The Sprite and Layers

Thanks Photoshop



The Final Screen

JavaFX on the Raspberry Pi



Java™ ORACLE®

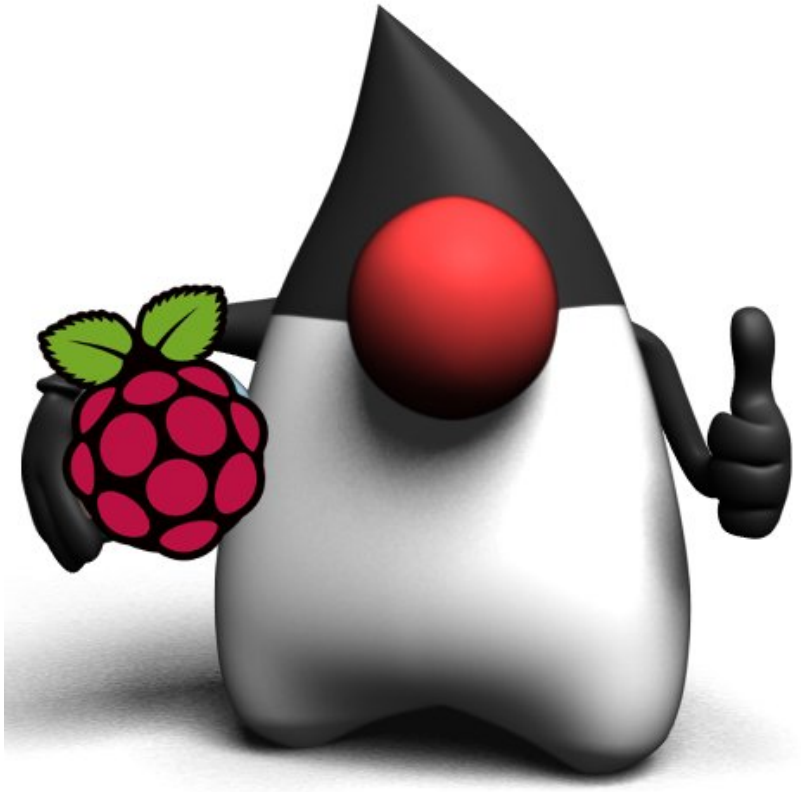
JavaFX and the Pi

- JavaFX is part of the JDK distribution
- Great for creating UI
- Great for doing animations
- Great binding capabilities
- Multi-threading support
- Mature and easy to use

PERFECT CHOICE!!!

Java™ ORACLE®

JavaFX Multi-Threading and Binding



Java™ ORACLE®

Tasks

- Implement the logic of work that needs to be done on a background thread
- Inherits from java.utils.concurrent.FutureTask class, which implements the Runnable interface

```
Task<Integer> task = new Task<Integer>() {
  @Override protected Integer call() throws Exception {
      int iterations;
      for (iterations = 0; iterations < 100000; iterations++) {
          if (isCancelled()) {break;}
          System.out.println("Iteration " + iterations);
      }
      return iterations;
  }
};
```

Binding

- Both high-level and low-level binding APIs
  - High-level APIs are clean and easy to use
  - Low-level APIs are faster and useful when speed matters
- Binding is Lazy
  - New values are only computed on demand
- Binding is made of:
  - Dependency tracking
  - Recomputing a value on read
  - Type specific bindings (no auto-boxing required)

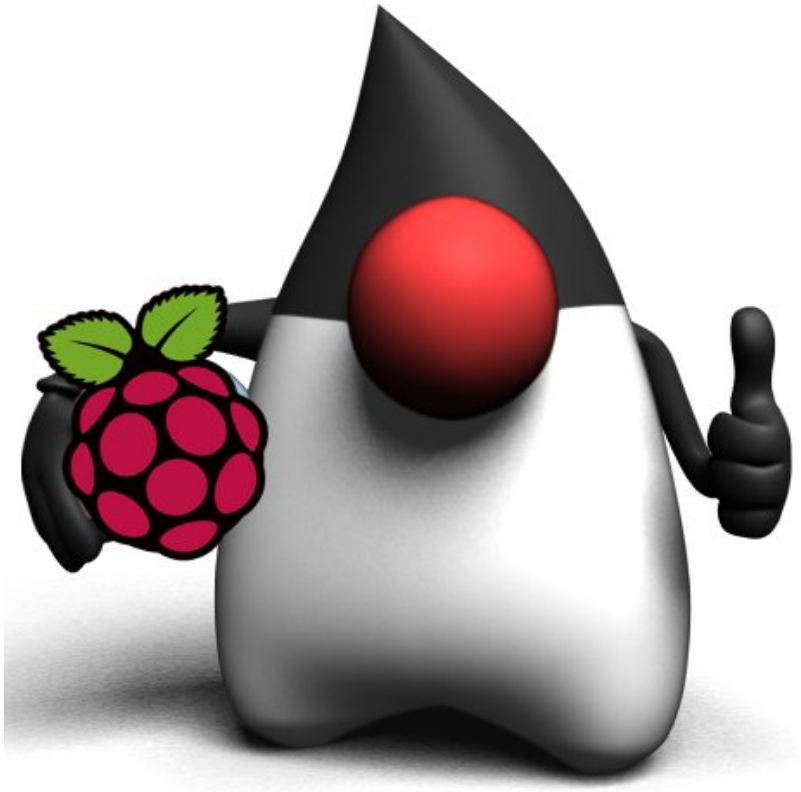
Property Binding

```
Rectangle r1 = new Rectangle();
Rectangle r2 = new Rectangle();

// Ensures that r2.x = r1.x
r2.xModel().bind(r1.xModel());
// Models can also be used to get/set
r1.xModel().set(10);
// Or do it the normal way
r1.setX(20);
```

Binding

```
clowdsPar.xCoord.bind(sensor.coordX);
farmPar.xCoord.bind(sensors.coordX);
backtreesPar.xCoord.bind(sensors.coordX);
…
clowdsPar.yCoord.bind(sensor.coordY);
farmPar.yCoord.bind(sensors.coordY);
backtreesPar.yCoord.bind(sensors.coordY);
…
```

Demo



Java™ ORACLE®

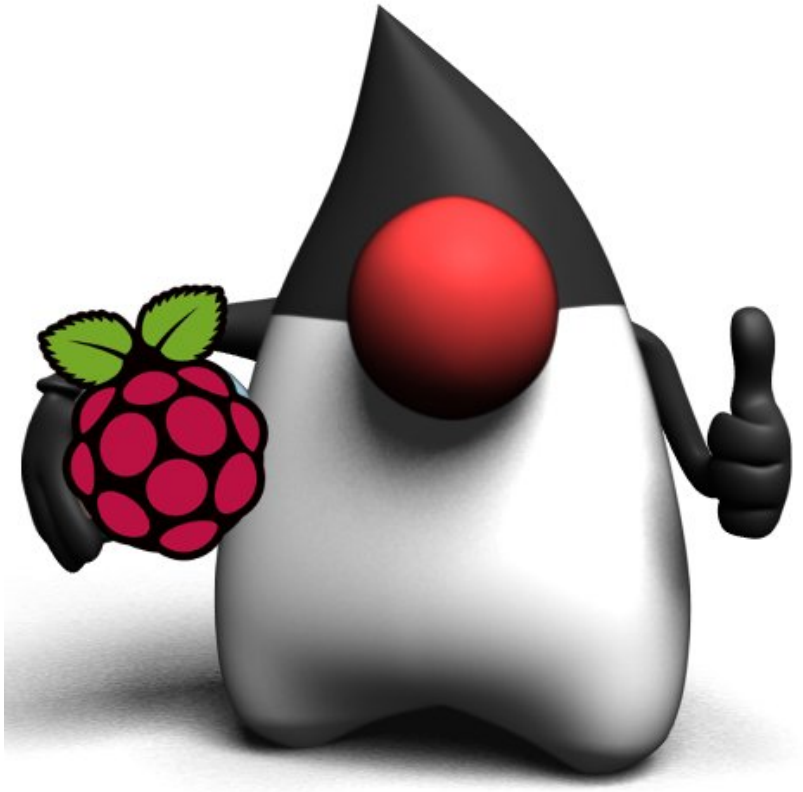
Conclusions

- Raspberry Pi is a very cool (and cheap) computer
  - Great for teaching
  - Great introduction to ARM
- Java works well and will get better
- Opportunities are limitless!
- HAVE FUN!

Further Information

- java.oracle.com
- www.oracle.com/technetwork/java/embedded
- Raspberry Pi User Guide – Eben Upton, Gareth Halfacree
- www.raspberrypi.org
- pi4j.com
- blogs.oracle.com/acaicedo

THANK YOU!



Java™ ORACLE®


MAKE THE
FUTURE
JAVA
Java
ORACLE


Java™