
JavaOne™

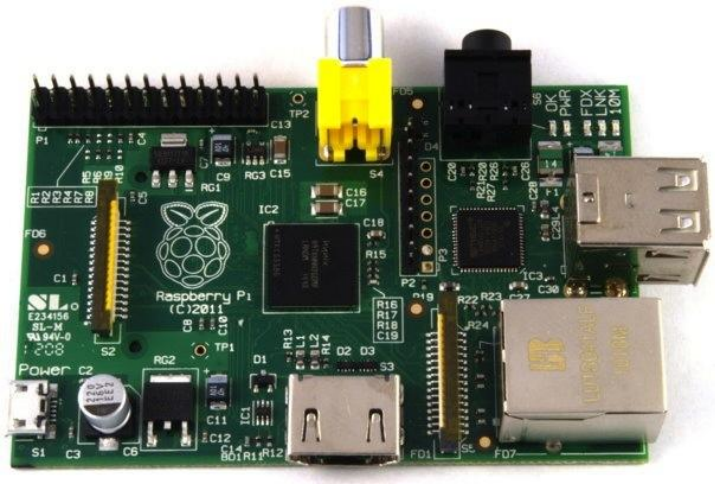

JavaOne™
Do You Like Coffee with Your Dessert?
Java and Raspberry Pi
Simon Ritter
Technology Evangelist
Twitter: @speakjava
MAKE THE FUTURE JAVA
ORACLE

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

- The Raspberry Pi
- ARM Processors
- Java on ARM and the Raspberry Pi
- Using Java on the Raspberry Pi
- Demos

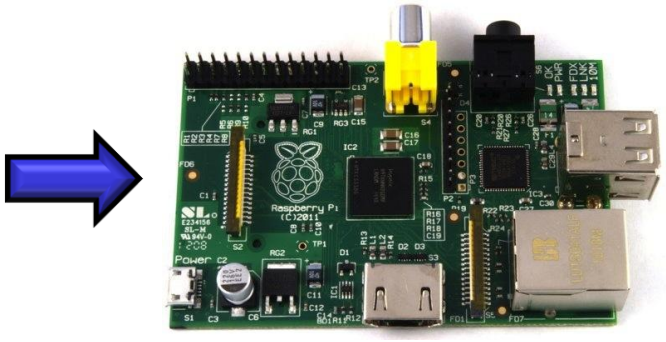
The Raspberry Pi



Raspberry Pi

History and Goals

- Project started in 2006
  - Goal was to devise a computer to inspire children
  - Inspiration from the BBC Micro project from 1981



Raspberry Pi

History and Goals

- 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
    - Originally in China, now by Sony in the UK
  - Current estimate is that around 1 million boards sold so far

Raspberry Pi

Specification

- CPU: ARM 11 core running at 700MHz
  - Broadcom SoC package
  - Can now be overclocked to 1GHz (without breaking the warranty!)
- Memory: ~~256Mb~~ 512Mb
- I/O:
  - HDMI and composite video
  - Audio out (3.5mm plug)
  - 2 x USB ports (Model B only)
  - Ethernet (Model B only)
  - Header pins for GPIO, UART, SPI and I2C

ARM Architecture

A Brief (But Interesting) History Lesson

- Acorn BBC Micro (6502 based)
  - Not powerful enough for Acorn's plans for a business computer
- Berkeley RISC Project
  - UNIX kernel only used 30% of instruction set of Motorola 68000
  - More registers, less instructions (Register windows)
  - One chip architecture to come from this was… SPARC
- Acorn RISC Machine (ARM)
  - 32-bit data, 26-bit address space, 27 registers
  - First machine was Acorn Archimedes
- Spin off from Acorn, Advanced RISC Machines

ARM Features

- 32-bit RISC Architecture
  - ARM accounts for 75% of embedded 32-bit CPUs today
  - 8 billion chips sold last year, more than 30 billion in total
    - zero manufactured by ARM
- Abstract architecture and microprocessor core designs
  - Raspberry Pi uses an ARM11 with the ARMv6 instruction set
- Low power consumption
  - Good for mobile devices
  - Raspberry Pi can be powered from 700mA 5V only PSU
  - Raspberry Pi does not require heatsink or fan

Current ARM Technology

- ARMv6
  - ARM 11, ARM Cortex-M
- ARMv7
  - ARM Cortex-A, ARM Cortex-M, ARM Cortex-R
- ARMv8 (Announced)
  - Will support 64-bit data and addressing
  - 32-bit instructions, 30 registers

Java On The ARM and Raspberry Pi

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
  - Oracle JVM just released (Early Access)

JDK8 Early Access

Official Java For The Raspberry Pi

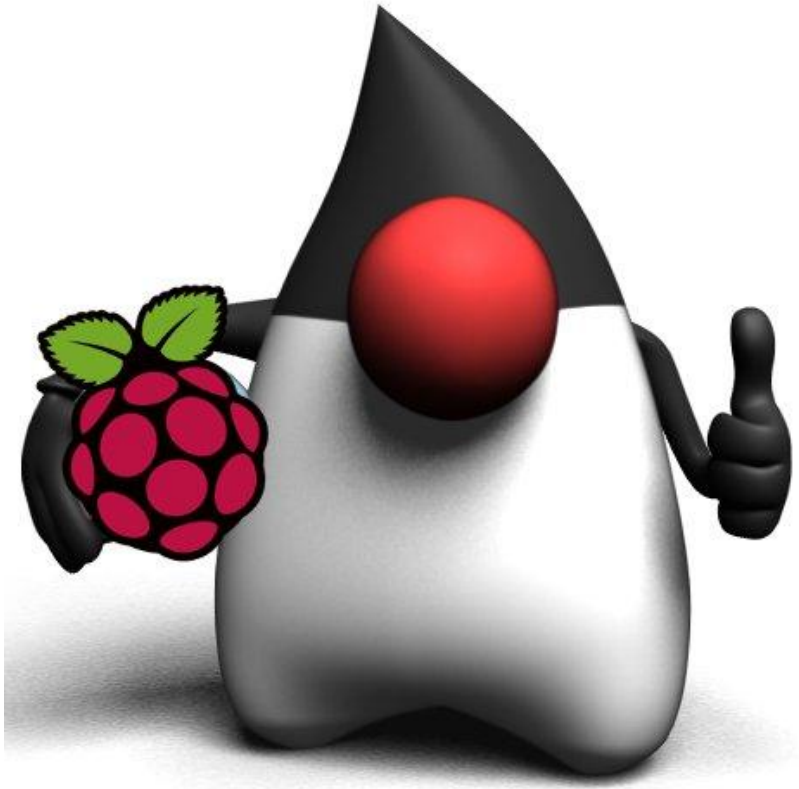
- Released at end of 2012
- Hard floating point support compiled in
- Includes JavaFX libraries
- Tested specifically for the Raspberry Pi
- Java SE 7 to follow at a later date

Beyond RISC

Performance Improvements

- DSP Enhancements
- Jazelle
- Thumb / Thumb2 / ThumbEE (Reduced instruction set RISC)
  - Better code density
- Floating Point (VFP)
- NEON
- Security Enhancements (TrustZone)
  - 2 Virtual processors (mode switch)
  - Hardware based access control

Using Java on the Raspberry Pi



Making A Noise With Java

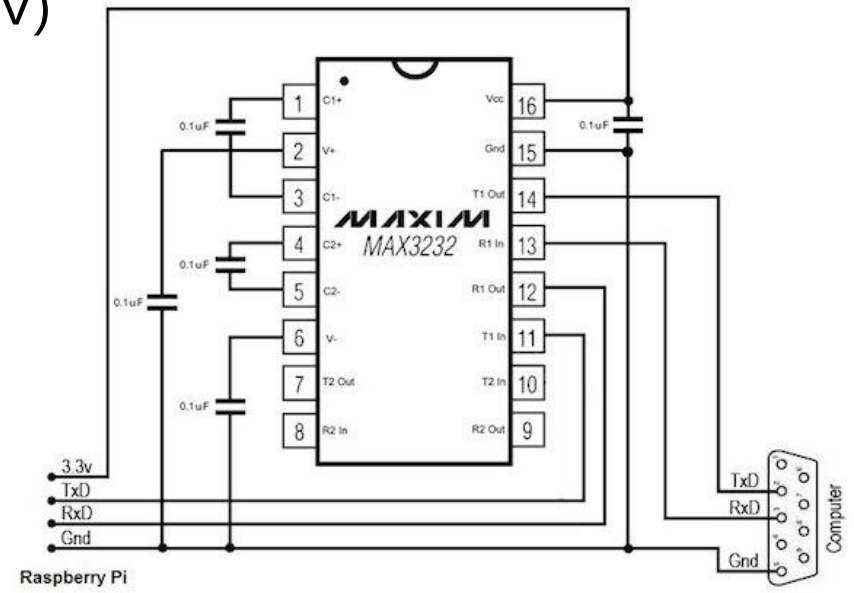
- Sound drivers now included in distros
- Java Sound API
  - Remember to add audio to user's groups
  - **`modprobe snd_pcm_oss`** (creates **`/dev/dsp`**, needed by MIDI interface)
  - Some bits work, others not so much
    - Playing (the right format) WAV file works
    - MIDI seems unbelievably slow
- FreeTTS text-to-speech
  - Some bits working (limited domain voice)

JavaFX on the Raspberry Pi

- Work involves optimal implementation of Prism graphics engine
  - Take advantage of Videocore 4 GPU
- Configure with **`-Djavafx.platform`**
  - ~~**`x11`** (X11, software rendering)~~ No X11 support
  - **`eglfb`** (OpenGL rendering to framebuffer) (JDK8 EA)
- Some things not supported
  - Media
  - WebEngine, WebView

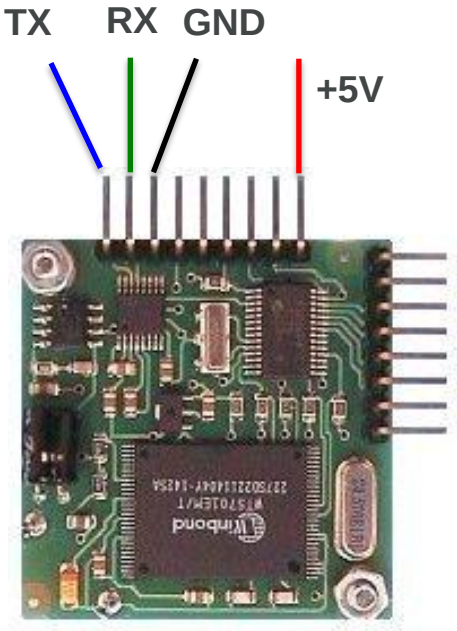
Using The Serial Port

- UART provides TTL level signals (3.3V)
- RS-232 uses 12V signals
- Use MAX3232 chip to convert
- Use this for access to serial console
- Serial port connectivity
  - **`/dev/ttyAMA0`**



The Talking Raspberry Pi

- SP03 text to speech board
  - Connected to UART via RS-232
- Disable console output
  - Edit **`/etc/inittab`**
    - Comment out **`getty /dev/ttyAMA0`**
  - Edit **`/boot/cmdline.txt`**
    - remove console and kgdboc references
- 38,400 baud, 8 Bit, 2 stop bits, no parity
- Simple byte sequence to send text to the board



USB Peripherals

Universal Serial Bus (But not as simple as serial)

- Easy devices are ones that appear as simple serial devices
  - `/dev/ttyUSB0`
- More complex devices need native code and libusb
  - `apt-get install libusb-1.0-0-dev`

Java and Serial Port/USB Serial Device

JavaComm API

- Install RXTX package
  - `apt-get install librxtx-java`
- How to solve the **`/dev/ttyS*`** only problem
  - `System.setProperty("gnu.io.rxtx.SerialPorts", "/dev/ttyUSB0");`
  - `System.setProperty("gnu.io.rxtx.SerialPorts", "/dev/ttyAMA0");`

The OWI Robot Arm

Cheap and cheerful

- Comes with USB interface
  - Windows only driver
  - Recognized as USB device by Linux
- 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

Robot Arm Control

JNI Code

- Native C functions
  - Initialisation 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 can be fiddly

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);
```

Gamepad Controller

Manual dexterity

- Linux supports most of these out of the box
- Drivers create entries in `/dev/input`
- Java API through Jinput
  - Mature technology (not been touched since 2003)
  - Recompile code on RasPi
  - Needed to tweak build script for incomplete classpath
    - EVIOCGUSAGE disappeared
    - Rename `libjinput-linux.so` to `libjinput-linux64.so`
  - Devices do not have general read/write access
    - Possible (but frustratingly difficult) to use udev.rules to fix this

Gamepad Controller

Code

- Wrote library on top of JInput
  - JInput too generic, needed code to be more specific to gamepad

```
GamePadController gpc = new GamePadController();
gpc.addButtonListener(GamePadController.BUTTON_1, this);
gpc.addJoystickListener(GamePadController.JOYSTICK_LEFT, this);
new Thread(gpc).start();
```

Gamepad Controller

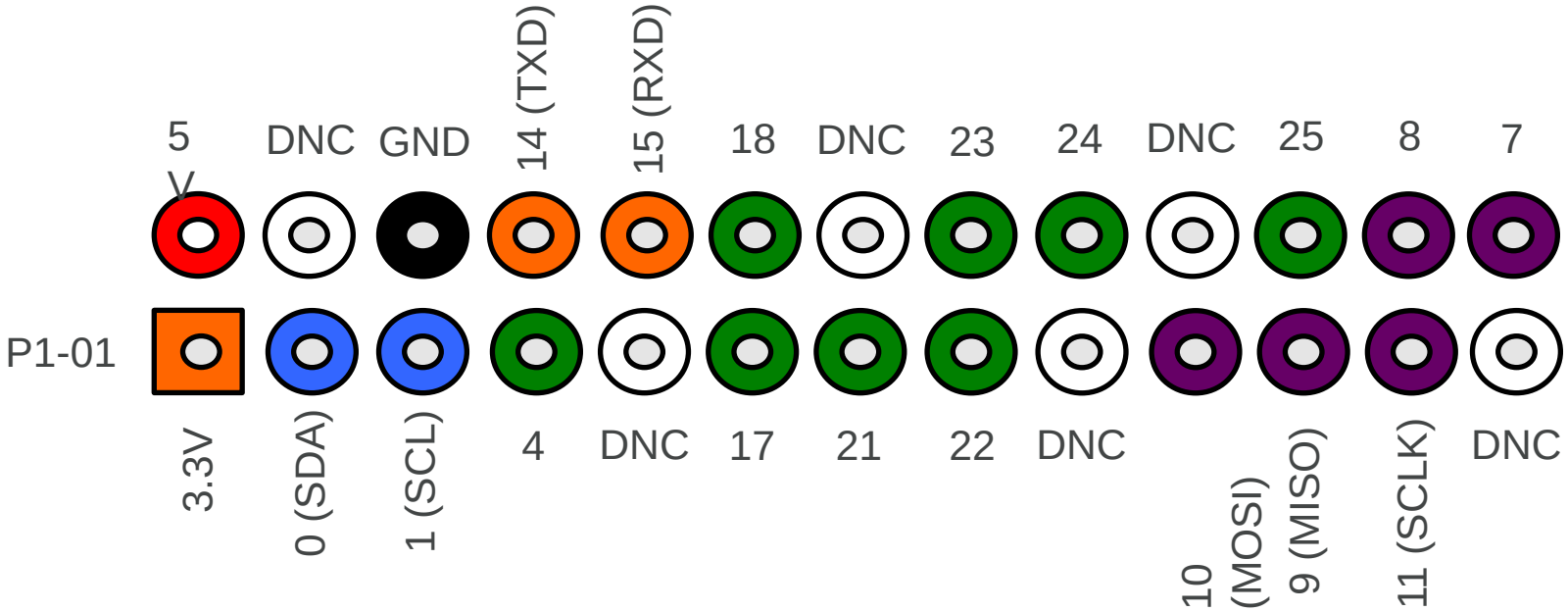
Code

```
public void buttonAction(ButtonEvent be) {
  if (be.getId() == GamePadController.BUTTON_1)
    robotArm.setGripperLight(true);
  ...
}

public void joystickAction(JoystickEvent jse) {
  if (jse.getId() == GamePadController.JOYSTICK_LEFT) {
    if ((position & JoystickEvent.POSITION_LEFT) != 0)
      robotArm.moveElbow(ArmController.UP);
  }
  ...
}
```

Using The GPIO Lines

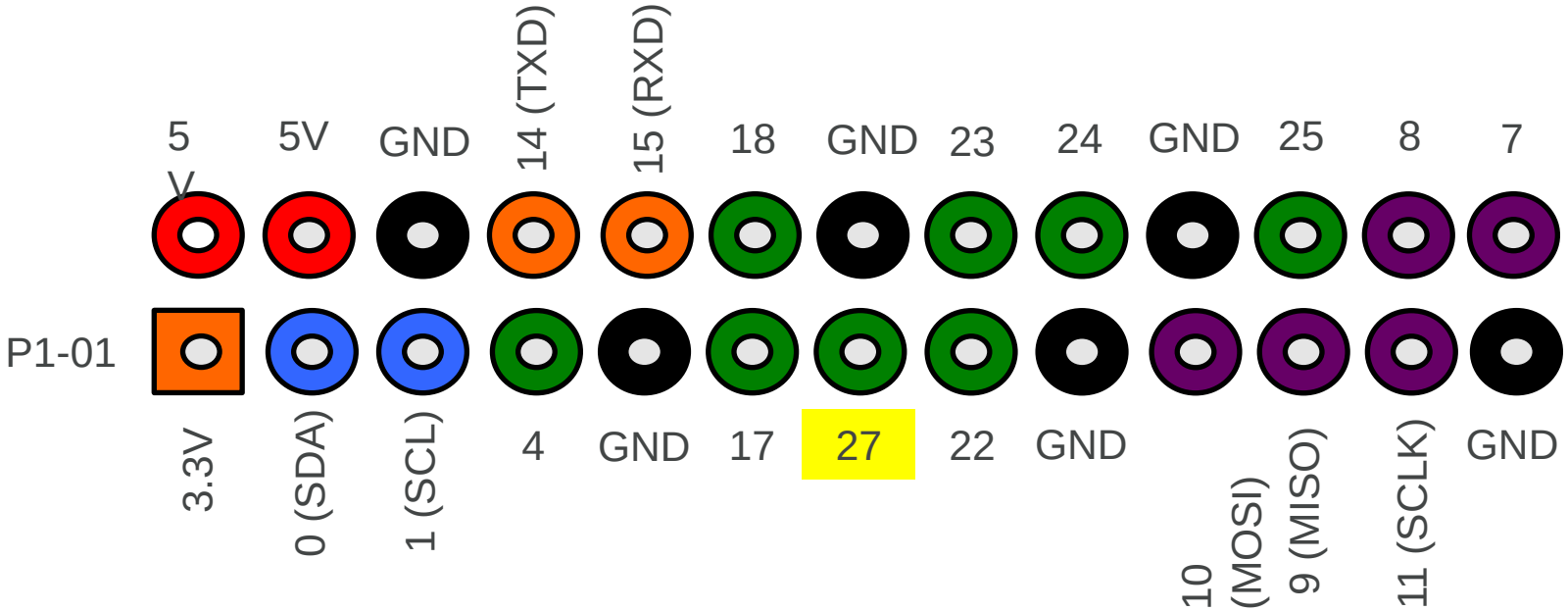
P1 Connector Layout (REV 1)



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

Using the GPIO Lines

P1 Connector Layout (Rev 2)



Using the GPIO Lines

Magic Incantations

```
#define BCM2708_PERI_BASE 0x20000000
#define GPIO_BASE (BCM2708_PERI_BASE + 0x200000)
#define BLOCK_SIZE (1024 * 4)
#define PAGE_SIZE (1024 * 4)

/* MMAP */
mem_fd = open("/dev/mem", O_RDWR | O_SYNC);
gpio_mem = malloc(BLOCK_SIZE + (PAGE_SIZE - 1));

gpio_map = (unsigned char *)mmap(
  (caddr_t)gpio_mem, BLOCK_SIZE, PROT_READ | PROT_WRITE,
  MAP_SHARED | MAP_FIXED, mem_fd, GPIO_BASE);

gpio = (volatile unsigned *)gpio_map;
```

Using the GPIO Lines

More Magic Incantations

```
/* Pin input */
*(gpio + (pin / 10)) &= ~(7 << ((pin % 10) * 3));

/* Pin output */
*(gpio + (pin / 10)) &= ~(7 << ((pin % 10) * 3));
*(gpio + (pin / 10)) |= (1 << ((pin % 10) * 3));

/* Pin high */
*(gpio + 7) = 1 << pin;

/* Pin low */
*(gpio + 10) = 1 << pin;
```

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);
```

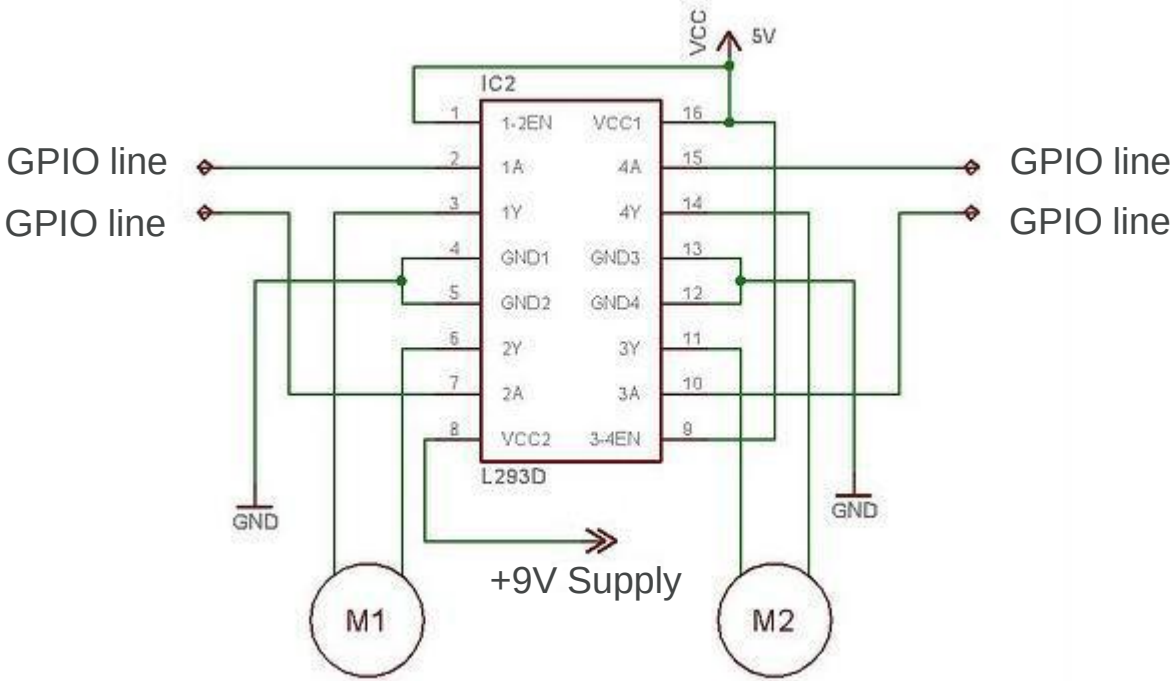
Pi4J

Simpler IO

- Bridge between native code and Java
  - GPIO
  - I2C
  - SPI
- GPIO code is simple and easy to use
- Still needs root access
- Still in development
  - Current version 0.0.4

GPIO Example: LEGO Motors

Using L293D Dual H-Bridge



How to Use SPI and I2C

Even more complex peripherals

- Drivers still experimental
  - Check Chris Boot's blog (www.bootc.net)
- Devices for SPI
  - `/dev/spidev-0.0` and `/dev/spidev-0.1`
- Devices for I2C
  - Run `i2c-dev`
  - `/dev/i2c-0`
- Not yet tried these with Java (Screen and JavaFX project next)

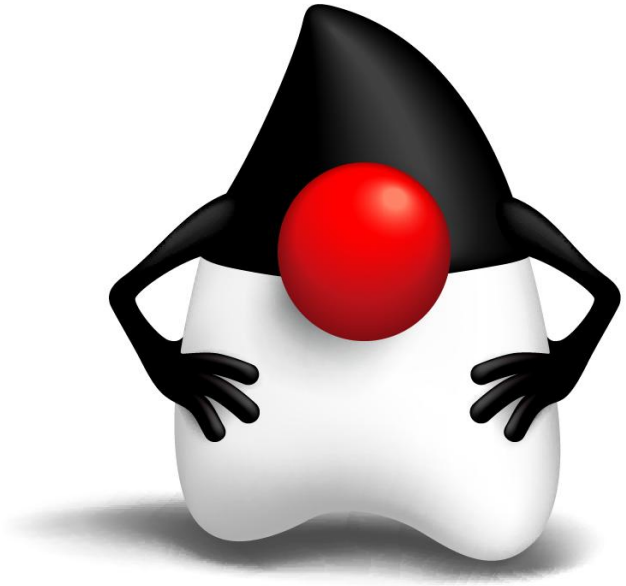
Conclusions

- 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!

Further Information

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

Demos



MAKE THE FUTURE JAVA

ORACLE


JavaOne™