Working with a Raspberry Pi

The RPi is based on Linux. You can communicate to the world using GPIO pins as well as Ethernet, USB, Audio, and Video. The RPi stands out from a regular computer because of its GPIO pins which can be controlled a variety of ways. This talk will discuss basic RPi uses and how to use the Java PI4j Library to work with hardware devices.
What is a Raspberry Pi?
Not this kind of Raspberry Pi!
This is a Raspberry Pi!
What can it do?
Retro Gaming with Raspberry Pi

https://learn.adafruit.com/retro-gaming-with-raspberry-pi/overview
https://imgur.com/gallery/o5vjL
PiPhone - A Raspberry Pi based Smartphone

http://www.davidhunt.ie/piphone-a-raspberry-pi-based-smartphone/
USS Constitution Museum Cannon Force Exhibit
http://josephpcohen.com/w/uss-constitution-museum-cannon-force-exhibit/
How do you make things with them?
It's just a regular computer! ...But it's a bit different

- For the OS it runs Raspbian Linux instead of Debian Linux
- It runs an ARM processor instead of a x86 or x64
  - Raspberry Pi 2 runs a 900 MHz quad-core ARM Cortex A7 with 1GB RAM!
  - Special package repository that has ARM compatible packages
  - Comes with gcc so you can compile anything you want to run!
  - Runs Java and Python
- Uses ~100mA to ~600mA from a Micro-USB cable.
  - 10000mAh battery = 16~100 hours!
- GPIO Pins! (General-purpose input/output)
Outline

1. What is a GPIO?
2. GPIO command line interface
   a. SysFS
   b. wiringPi
3. Pi4J Java Interface
   a. GPIOReadExample.java
   b. GPIOWriteExample.java
   c. WalkTurtleDemo.java
   d. WalkTurtleGame.java (You finish the code!)
GPIOs are huge in industry!

Programmable Logic Controllers (PLCs) use GPIOs to power power plants, airplanes, ships, water filtration plants, bottling plants, and almost everything you have seen on How it's Made!
SCADA (supervisory control and data acquisition)

These systems are full of PLCs which recently have become a target of war. It's important to understand them in order to build secure systems!
GPIO Pins!

This pins allow you to read and write 3.3 volt values with the world.

When you write a value of 1 or 0 the pin will then have a 0 or 3.3 volt potential.

Before you can read a value you must configure the pin to be a pullup or pulldown input. A pullup input will have a default potential of 3.3 volts (value 1) and will have the value 0 once the pin is grounded. A pulldown is the opposite.
Some pins on the Raspberry Pi header allow access to other inputs such as the SPI and I2C busses as well as a UART.

**UART**: A serial controller that allows buffered and timed serial communication.

**SPI/I2C**: Busses similar to USB that connect to LCD Panels, LED arrays, Analog to Digital converters (A2D), etc.
Each GPIO is a digital input from an analog signal. When the signal is around 3.3 volts the device will read in a 1.

A initial value of 0 volts (LOW) (HIGH) A button press can bring the voltage on a wire up to cause a momentary rise.
GPIO command line interface
GPIO SysFS Interface

Part of the Linux kernel!

- Filesystem abstraction to GPIOs
- Not just for Raspberry Pi
- Works on desktop Linux
  - where are the pins?!
- Debug projects in the field

https://www.kernel.org/doc/Documentation/gpio/sysfs.txt

```
pi@raspberrypi /sys/class/gpio $ ls
export gpio2 gpio3 gpiochip0 unexport
pi@raspberrypi /sys/class/gpio $
pi@raspberrypi /sys/class/gpio $ unexport
pi@raspberrypi /sys/class/gpio $ find .
./unexport
./gpio2
./gpio3
./export
./gpiochip0
```
Connect LED
Plug in + to GPIO2
Plug in - to Ground
We must be root to work with GPIO pins

This folder gpio is a fake folder that provides access to the gpio driver

If we pipe into export it creates a new folder with new fake files. If there are errors, unexport and try again.

direction will specify "in" or "out" communication

What changes the state: "none", "rising", "falling"

Setting value to 1 or 0 will set the pins voltage to 0 or 3.3 volts.

---

```bash
$ sudo su
#

# ls /sys/class/gpio
export gpiochip0 unexport

# echo 2 > /sys/class/gpio/export

# ls /sys/class/gpio
export gpiochip0 unexport

# ls /sys/class/gpio/gpio2
direction edge uevent value ...
```

---

Echo and cat to read pin state
Set the direction of GPIO2 to "out"

Set the value of GPIO2 to HIGH

Reset the pin and change direction to in

read the current value at GPIO2. Connect GPIO2 to ground or 3.3 volts to change the value.

# echo out > /sys/class/gpio/gpio2/direction
# echo 1 > /sys/class/gpio/gpio2/value

# echo 2 > /sys/class/gpio/unexport
# echo 2 > /sys/class/gpio/export
# echo in > /sys/class/gpio/gpio2/direction

# cat /sys/class/gpio/gpio2/value
0
WiringPi

Author: Gordon Henderson
Licensed under the GNU LGPLv3

C library, GPIO utility, Easy access to:
- Read/write GPIO pin values
- Read/write gertboard a2d converters
- Debug i2c bus devices
- and more!

http://wiringpi.com/
http://git.drogon.net/wiringPi

<table>
<thead>
<tr>
<th>BCM</th>
<th>wPi</th>
<th>Name</th>
<th>Mode</th>
<th>V</th>
<th>Phys</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>SDA.1</td>
<td>OUT</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>SCL.1</td>
<td>OUT</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>GPIO. 7</td>
<td>OUT</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>GPIO. 0</td>
<td>IN</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>GPIO. 2</td>
<td>IN</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>GPIO. 3</td>
<td>IN</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>MOSI</td>
<td>IN</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>MISO</td>
<td>OUT</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>SCLK</td>
<td>IN</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>SDA.0</td>
<td>IN</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>GPIO.21</td>
<td>IN</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>GPIO.22</td>
<td>IN</td>
<td>1</td>
<td>31</td>
</tr>
<tr>
<td>13</td>
<td>23</td>
<td>GPIO.23</td>
<td>IN</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>19</td>
<td>24</td>
<td>GPIO.24</td>
<td>IN</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>26</td>
<td>25</td>
<td>GPIO.25</td>
<td>IN</td>
<td>0</td>
<td>37</td>
</tr>
</tbody>
</table>
Install WiringPi for gpio utility

```bash
# git clone git://git.drogon.net/wiringPi
# ./build
wiringPi Build script
=====================  
WiringPi Library
[UnInstall]
[Compile] wiringPi.c
[Compile] wiringSerial.c
[Compile] wiringShift.c
...
All Done.
# gpio
Usage: gpio -v
     gpio -h
     gpio <read/write/aread/awritewb/pwm/clock/mode> ...
     gpio readall/reset
...```
# gpio mode 8 in
# gpio read 8
0
# gpio mode 8 out
# gpio write 8 1
# gpio write 8 0

Read and write pins easily!

Find your pin layout: http://pi4j.com/pins/model-2b-rev1.html
Echo and cat to read pin state

Use a while loop to repeat the reading forever

# while [ true ]; do gpio read 8; done
0
0
0
0
0
0
0
0
1
1
...

Press Ctrl-C to stop the loop
The Pi4J Project

Connecting Java to the Raspberry Pi

Author: Robert Savage
Licensed under the GNU LGPLv3

Included as a jar, able to build closed source project around them!

http://pi4j.com/
https://github.com/pi4j
Get the code and run it!

```
$ git clone https://github.com/ieee8023/RaspberryPi-ExampleGPIO
Cloning into 'RaspberryPi-ExampleGPIO'...
remote: Counting objects: 33, done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 33 (delta 2), reused 28 (delta 1), pack-reused 0
Unpacking objects: 100% (33/33), done.
Checking connectivity... done.

$ cd RaspberryPi-ExampleGPIO/

$ sh compile.sh

$ sh run.sh WalkTurtleDemo

...
How the scripts make our code work

The classpath is set using backticks which execute the script getclasspath.sh

All java files are found using find which recursively searches the folder src

```
$ cat compile.sh
mkdir -p classes
javac -cp `sh getclasspath.sh` -d classes `find src -type f -name "*.java"`
```

```
$ cat getclasspath.sh
echo `find lib-pi4j | tr '\n' ':'`
```

```
$ cat runWalkTurtleDemo.sh
sudo java -Xmx128m -cp `sh getclasspath.sh`:classes WalkTurtleDemo
```

sudo is needed in run.sh in order for Pi4J access GPIO pins

This script runs the WalkTurtleDemo class.
Attach switches to GPIO8 and GPIO9 for wiringPi

<table>
<thead>
<tr>
<th>GPIO#</th>
<th>NAME</th>
<th>GPIO#</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>GPIO 8</td>
<td>15</td>
<td>GPIO 15</td>
</tr>
<tr>
<td>9</td>
<td>GPIO 9</td>
<td>16</td>
<td>GPIO 16</td>
</tr>
<tr>
<td>7</td>
<td>GPIO 7</td>
<td>11</td>
<td>GPIO 11</td>
</tr>
<tr>
<td>0</td>
<td>GPIO 0</td>
<td>4</td>
<td>GPIO 4</td>
</tr>
<tr>
<td>2</td>
<td>GPIO 2</td>
<td>14</td>
<td>Ground</td>
</tr>
<tr>
<td>3</td>
<td>GPIO 3</td>
<td>5</td>
<td>GPIO 5</td>
</tr>
<tr>
<td>12</td>
<td>GPIO 12</td>
<td>25</td>
<td>SDA0 (I2C ID EEPROM)</td>
</tr>
<tr>
<td>13</td>
<td>GPIO 13</td>
<td>27</td>
<td>SCL0 (I2C ID EEPROM)</td>
</tr>
<tr>
<td>14</td>
<td>GPIO 14</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>GPIO 7</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Connect Switches
Ground<->GPIO8
Ground<->GPIO9
GPIOReadExample.java

```java
final GpioController gpio = GpioFactory.getInstance();
final GpioPinDigitalInput trigger = gpio.provisionDigitalInputPin(RaspiPin.GPIO_08, PinPullResistance.PULL_UP);
final GpioPinDigitalInput input = gpio.provisionDigitalInputPin(RaspiPin.GPIO_09, PinPullResistance.PULL_UP);
...
```

A singleton GPIO Controller is required to create input objects

Without pull-up and pull-down resistors the GPIO may float between values or 0 or 1.

To deal with the Raspberry Pi has built in resistors that you can configure with code!

Pi4j addresses GPIOs from 0 to 29
```java
final GpioController gpio = GpioFactory.getInstance();
final GpioPinDigitalInput trigger = gpio.provisionDigitalInputPin(RaspiPin.GPIO_08, PinPullResistance.PULL_UP);
final GpioPinDigitalInput input = gpio.provisionDigitalInputPin(RaspiPin.GPIO_09, PinPullResistance.PULL_UP);
...
```

**Pull-down:** Input starts as 0 and 3.3 volts is needed to become 1

**Pull-up:** Input starts as 1 and connecting ground results in a 0
final GpioPinDigitalInput trigger = ...  
final GpioPinDigitalInput input = ... 
trigger.setDebounce(100); 
trigger.addListener(new GpioPinListenerDigital(){  
public void handleGpioPinDigitalStateChangeEvent(...){   
  System.out.println(trigger.getPin() + " triggered!");   
  PinState state = input.getState();   
  if (state == PinState.HIGH)   
    System.out.println(input.getPin() + " is high");  
  else   
    System.out.println(input.getPin() + " is low");  
}   
}); 
while (true){ Thread.sleep(500);} 

We set a debounce time of 100ms to avoid false changes

We had a listener in typical Java fashion

Inside the listener we can access the trigger and input variables

PinState is an enum with values HIGH and LOW

Continue running and wait for the pin to change state
Attach LED to GPIO7

Connect LED
  + <-> GPIO7
  - <-> Ground
GPIOWriteExample.java

Configure GPIO7 as output

final GpioPinDigitalOutput output =
    gpio.provisionDigitalOutputPin(RaspiPin.GPIO_07);
ScheduledExecutorService exec =
    Executors.newSingleThreadScheduledExecutor();

exec.scheduleAtFixedRate(new Runnable() {
    public void run() {
        if (output.getState() != PinState.LOW)
            output.setState(PinState.LOW);
        else
            output.setState(PinState.HIGH);
    }
}, 0, 100, TimeUnit.MILLISECONDS);

Use scheduled executor to execute Runnable object at an interval

Check and set to opposite
double x0 = 0.5, y0 = 0.5, a0 = 0.0;
final Turtle turtle = new Turtle(x0, y0, a0);

ScheduledExecutorService exec = ...
exec.scheduleAtFixedRate(new Runnable() {
    double step = 0.002;
    public void run() {
        turtle.goForward(step += 0.02);
        turtle.turnLeft(90);
    }
}, 0, 1, TimeUnit.SECONDS);

The turtle can go forward and turn left. Every second the turtle turns 90 degrees and steps forward more and more.
A scheduled task checks the GPIOs and updates the game

Read GPIO pins and set new rotation and speed here

The turtle turns left when the GPIO is 1 otherwise continues

Speed increases as long as GPIO is 1 otherwise we stop moving

```java
ScheduledExecutorService exec = ...
exec.scheduleAtFixedRate(new Runnable() {
    double rot = 0, spd = 0;
    public void run() {
        if (true /*check if GPIO is 1*/) {
            rot = (rot + turn) % 360;
        } else {
            rot = 0;
        }
        if (true /*check if GPIO is 1*/) {
            spd += step;
        } else {
            spd = 0;
        }
        turtle.goForward(spd);
        turtle.turnLeft(rot);
    }
}, 0, 500, TimeUnit.MILLISECONDS);
```
Ideas to improve WalkTurtleGame.java

1. Use a listener to speed up feedback
2. Instead of stopping, just reduce speed
3. Change to left and right control
4. Make a goal space that gives you points
5. Paint a car that drives around
6. Add an a2d converter as accelerator
**Basement Flood Alarm:**

- Take a water level sensor and instrument it to add intelligent monitoring and notification capability

**HVAC Alarm:**

- Take a HVAC moisture sensor and extend it to add intelligent monitoring and notification capability

**Mail Notification:**

- Instrument a mailbox to get notified when mail arrives.

**Driveway Alarm:**

- Add a sensor to driveway to get notified when someone approaches the house.
Sprinkler System
- Remotely control, configure and schedule the system.
- Skip watering schedules if raining or if rain is forecasted

Garage Door Opener:
- Remote control and monitoring of garage door
- Auto-close if left open

HVAC System
- Interface with HVAC thermostat to remotely monitor and control the HVAC system.

Security System
- Remote control and monitoring of the system
- Activate other devices based on the state of the system
1.1 Normally Closed Contact in Series Connection

Control Purpose:
- Detecting the standing bottles on the conveyor and pushing the fallen bottles out

Devices:

<table>
<thead>
<tr>
<th>Device</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td>X0 = ON when the detected input signal from the bottle-bottom is sheltered.</td>
</tr>
<tr>
<td>X1</td>
<td>X1 = ON when the detected input signal from the bottle-neck is sheltered.</td>
</tr>
<tr>
<td>Y0</td>
<td>Pneumatic pushing pole</td>
</tr>
</tbody>
</table>

http://www.slideshare.net/geterrdone/plc-projects-application-examples
2.1 Product Mass Packaging

Control Purpose:
- Once the photoelectric sensor detects 10 products, the robotic arm will begin to pack up. When the action is completed, the robotic arm and the counter will be reset.

Devices:

<table>
<thead>
<tr>
<th>Device</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td>Photoelectric sensor for counting products. X0 = ON when products are detected.</td>
</tr>
<tr>
<td>X1</td>
<td>Robotic arm action completed sensor. X1 = ON when packing is completed.</td>
</tr>
<tr>
<td>C0</td>
<td>Counter: 16-bit counting up (general purpose)</td>
</tr>
<tr>
<td>Y0</td>
<td>Robotic arm for packing</td>
</tr>
</tbody>
</table>
This talk was organized and created by Joseph Paul Cohen

Raspberry Pi Giveaway sponsored by BATEC

Email: joseph@josephpcohen.com
Website: http://josephpcohen.com

National Science Foundation Graduate Fellow
Ph.D Candidate - Computer Science
University of Massachusetts Boston