DEVOXX™

Let's Get Physical:

I/O Programming with Java on the Raspberry Pi using Pi4J

Robert Savage
The Pi4J Project

Project: http://pi4j.com
Blog: http://savagehomeautomation.com

Agenda

- Pi4J Overview
- Pi4J Introductory Concepts
- Smart Devices & IoT
- DIY Smart Devices
- MQTT

What is Pi4J

Pi4J is an open-source project providing a library for Java programmers to interact with the low-level I/O capabilities on the Raspberry Pi platform.

- Open Source Project
- Low Level I/O Library
- Object Oriented API
- Event Based
- Java & C (JNI + Native)

www.pi4j.com

Low Level I/O Interfaces

Digital Interfaces

- GPIO — *General Purpose Input Output*
- PWM — *Pulse Width Modulation*

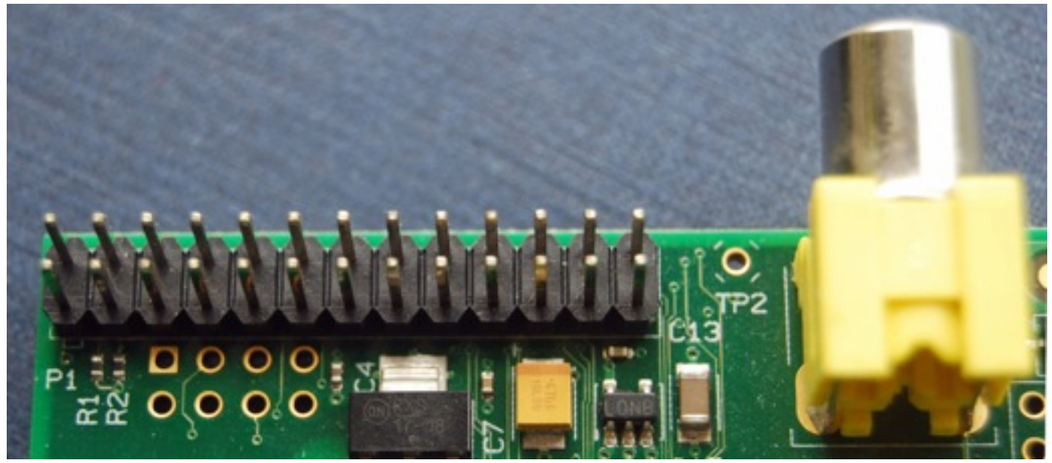
Data Interfaces

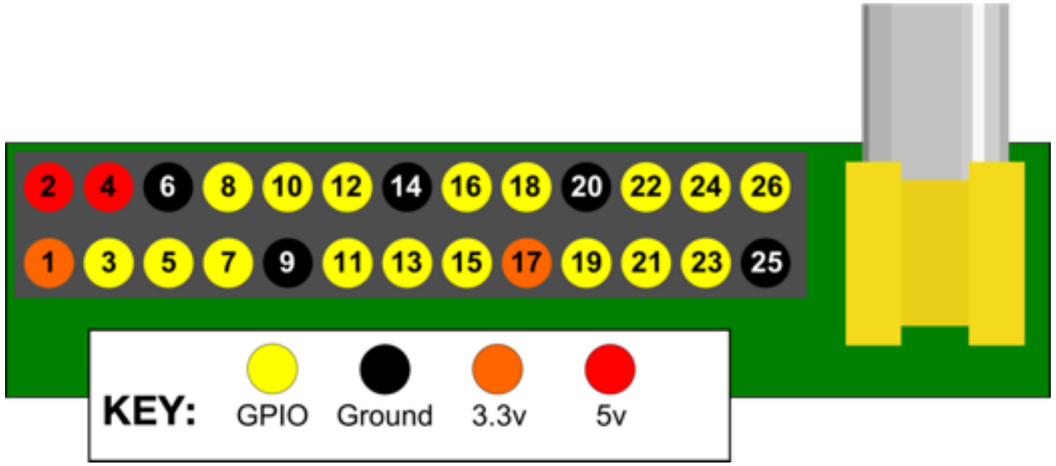
- UART, SERIAL — *Universal Asynchronous Receiver/Transmitter*
- SPI — *Serial Peripheral Interface*
- I^2C — *Inter-Integrated Circuit*

Analog Interfaces *

GPIO

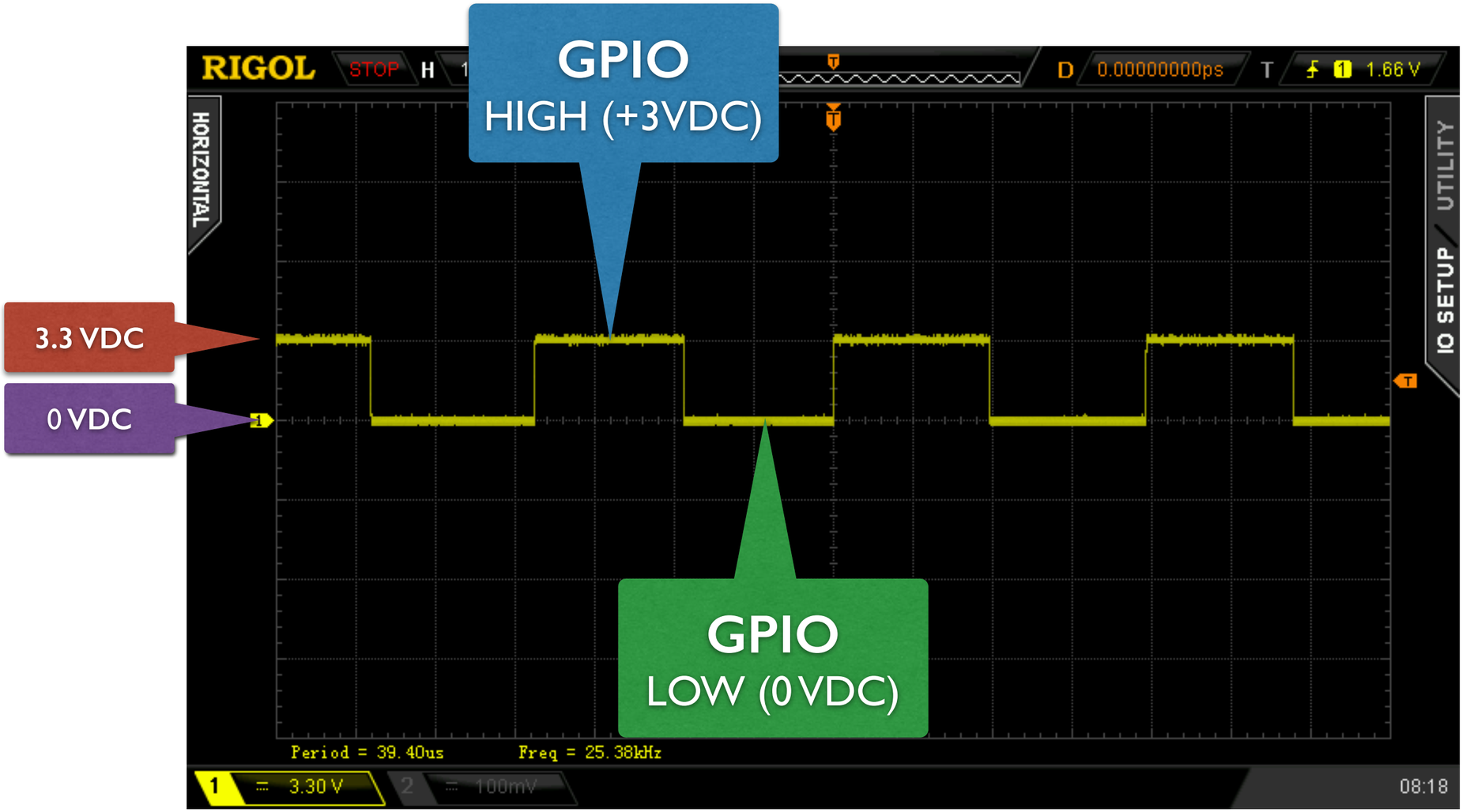
- Input or Output
- Digital States
  - HIGH ~ 3.3 VDC
  - LOW ~ 0 VDC
- Models
  - A & B = ~21 GPIO
  - B+ = 28 GPIO
  - Compute Module = 46 GPIO





(images from raspberrypi.org)

GPIO Digital States



Pi4J GPIO Pin Addressing

Raspberry Pi P1 Header

PIN #	NAME			NAME	PIN #
	3.3 VDC Power	1	2	5.0 VDC Power	
8	SDA0 (I2C)	3	4	DNC	
9	SCL0 (I2C)	5	6	0V (Ground)	
7	GPIO 7	7	8	TxD	15
	DNC	9	10	RxD	16
0	GPIO 0	11	12	GPIO1	1
2	GPIO2	13	14	DNC	
3	GPIO3	15	16		
	DNC	17	18		
12	MOSI	19	20		
13	MISO	21	22		
14	SCLK	23	24	CE0	10
	DNC	25	26	CE1	11

http://www.pi4j.com

Raspberry Pi P5 Header

PIN #	NAME			NAME	PIN #
	3.3 VDC Power	2	1	5.0 VDC Power	
18	GPIO18	4	3	GPIO17	17
20	GPIO20	6	5	GPIO19	19
	0V (Ground)	8	7	0V (Ground)	

http://www.pi4j.com

Raspberry Pi J8 Header (Model B+)

GPIO#	NAME			NAME	GPIO#
	3.3 VDC Power	1	2	5.0 VDC Power	
8	GPIO 8 SDA1 (I2C)	3	4	5.0 VDC Power	
9	GPIO 9 SCL1 (I2C)	5	6	Ground	
7	GPIO 7 GPCLK0	7	8	GPIO 15 TxD (RS232)	15
	Ground	9	10	GPIO 16 RxD (RS232)	16
0	GPIO 0	11	12	GPIO 1 PCM_CLK/PWM0	1
2	GPIO 2	13	14	Ground	
	MISO (SPI)				
14	GPIO 14 SCLK (SPI)	23	24	GPIO 10 CE0 (SPI)	10
	Ground	25	26	GPIO 11 CE1 (SPI)	11
	SDA0 (I2C ID EEPROM)	27	28	SCL0 (I2C ID EEPROM)	
21	GPIO 21 GPCLK1	29	30	Ground	
22	GPIO 22 GPCLK2	31	32	GPIO 26 PWM0	26
23	GPIO 23 PWM1	33	34	Ground	
24	GPIO 24 PCM_FS/PWM1	35	36	GPIO 27	27
25	GPIO 25	37	38	GPIO 28 PCM_DIN	28
	Ground	39	40	GPIO 29 PCM_DOUT	29

http://www.pi4j.com

RPi Compute Mod Dev Board - J5 Header

GPIO#	NAME			NAME
0	GPIO 0	1	2	Ground
1	GPIO 1	3	4	5.0 VDC Power
2	GPIO 2 (I²C) SDA1 [ALT0]	5	6	Ground
3	GPIO 3 (I²C) SCL1 [ALT0]	7	8	3.3 VDC Power
4	GPIO 4 GPCLK0 [ALT0]	9	10	Ground
5	GPIO 5	11	12	1.8 VDC Power
6	GPIO 6	13	14	Ground
7	GPIO 7 SPI0_CE1_N [ALT0]	15	16	VG0 Power
8	GPIO 8 SPI0_CE0_N [ALT0]	17	18	Ground
9	GPIO 9 SPI0_MISO [ALT0]	19	20	3.3 VDC Power
10	GPIO 10 SPI0_MOSI [ALT0]	21	22	Ground
16	GPIO 16	33	34	Ground
17	GPIO 17	35	36	1.8 VDC Power
18	GPIO 18 PWM0 [ALT5]	37	38	Ground
19	GPIO 19 PWM1 [ALT5]	39	40	VG0 Power
20	GPIO 20	41	42	Ground
21	GPIO 21	43	44	3.3 VDC Power
22	GPIO 22	45	46	Ground
23	GPIO 23	47	48	1.8 VDC Power
24	GPIO 24	49	50	Ground
25	GPIO 25	51	52	VG0 Power
26	GPIO 26	53	54	Ground
27	GPIO 27	55	56	5.0 VDC Power
	RUN	57	58	Ground
	GPIO47_CTL_1V8	59	60	Ground

http://www.pi4j.com

RPi Compute Mod Dev Board - J6 Header

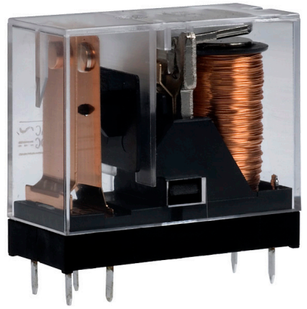
GPIO#	NAME			NAME
28	GPIO 28	1	2	Ground
29	GPIO 29	3	4	5.0 VDC Power
30	GPIO 30	5	6	Ground
31	GPIO 31	7	8	3.3 VDC Power
32	GPIO 32	9	10	Ground
33	GPIO 33	11	12	1.8 VDC Power
34	GPIO 34	13	14	Ground
35	GPIO 35	15	16	VG1 Power
36	GPIO 36	17	18	Ground
37	GPIO 37	19	20	3.3 VDC Power
38	GPIO 38	21	22	Ground
		23	24	1.8 VDC Power
		25	26	Ground
		27	28	VG1 Power
		29	30	Ground
		31	32	3.3 VDC Power
44	GPIO 44	33	34	Ground
45	GPIO 45 PWM1 [ALT0]	35	36	1.8 VDC Power
	CD1_SDA	37	38	Ground
	CD1_SCL	39	40	VG1 Power
	CAM1_IO1	41	42	Ground
	CAM1_IO0	43	44	3.3 VDC Power
	CD0_SDA	45	46	Ground
	CD0_SCL	47	48	1.8 VDC Power
	CAM0_IO1	49	50	Ground
	CAM0_IO0	51	52	VG1 Power
	VDD_CORE	53	54	Ground
	USB_OTGID	55	56	5.0 VDC Power
	Ground	57	58	Ground
	TV_DAC	59	60	Ground

http://www.pi4j.com

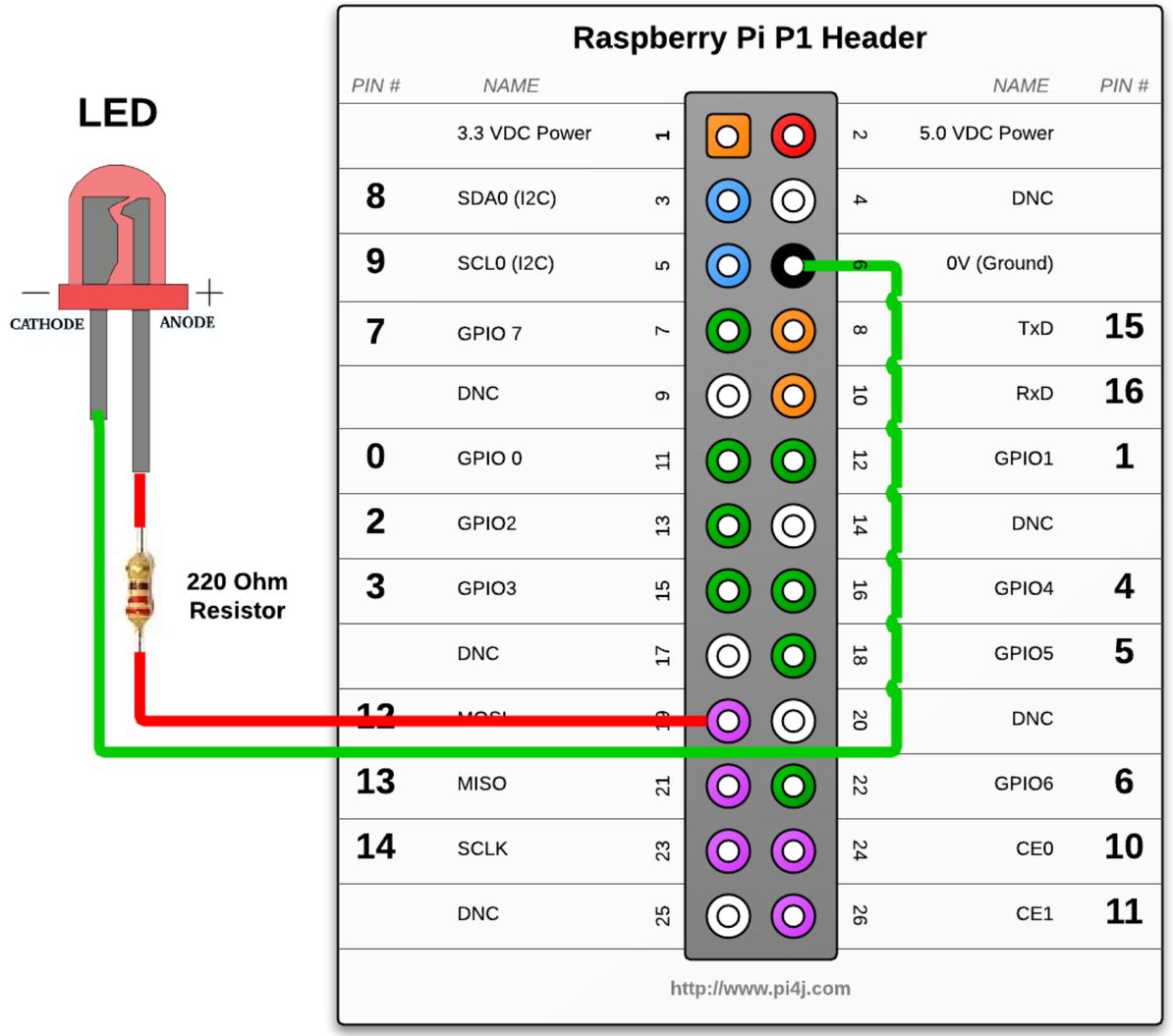
Visit pi4.com for a detailed pin addressing diagram!

GPIO Outputs

GPIO Outputs can be used to control things

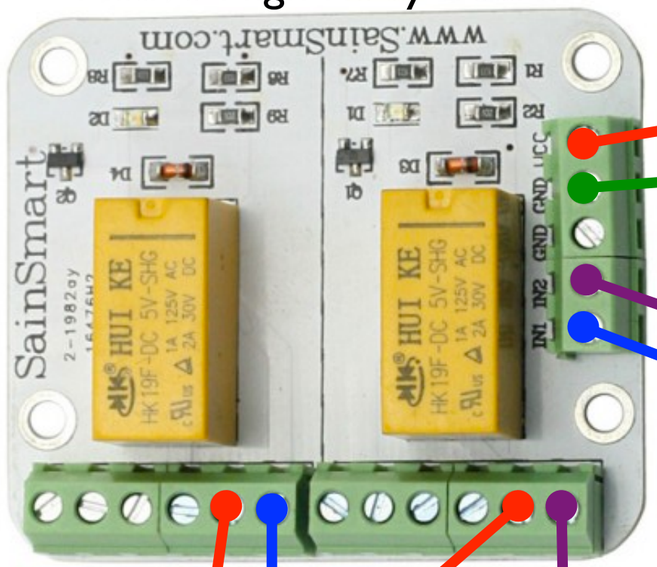


GPIO Output Circuit



GPIO Output Circuit

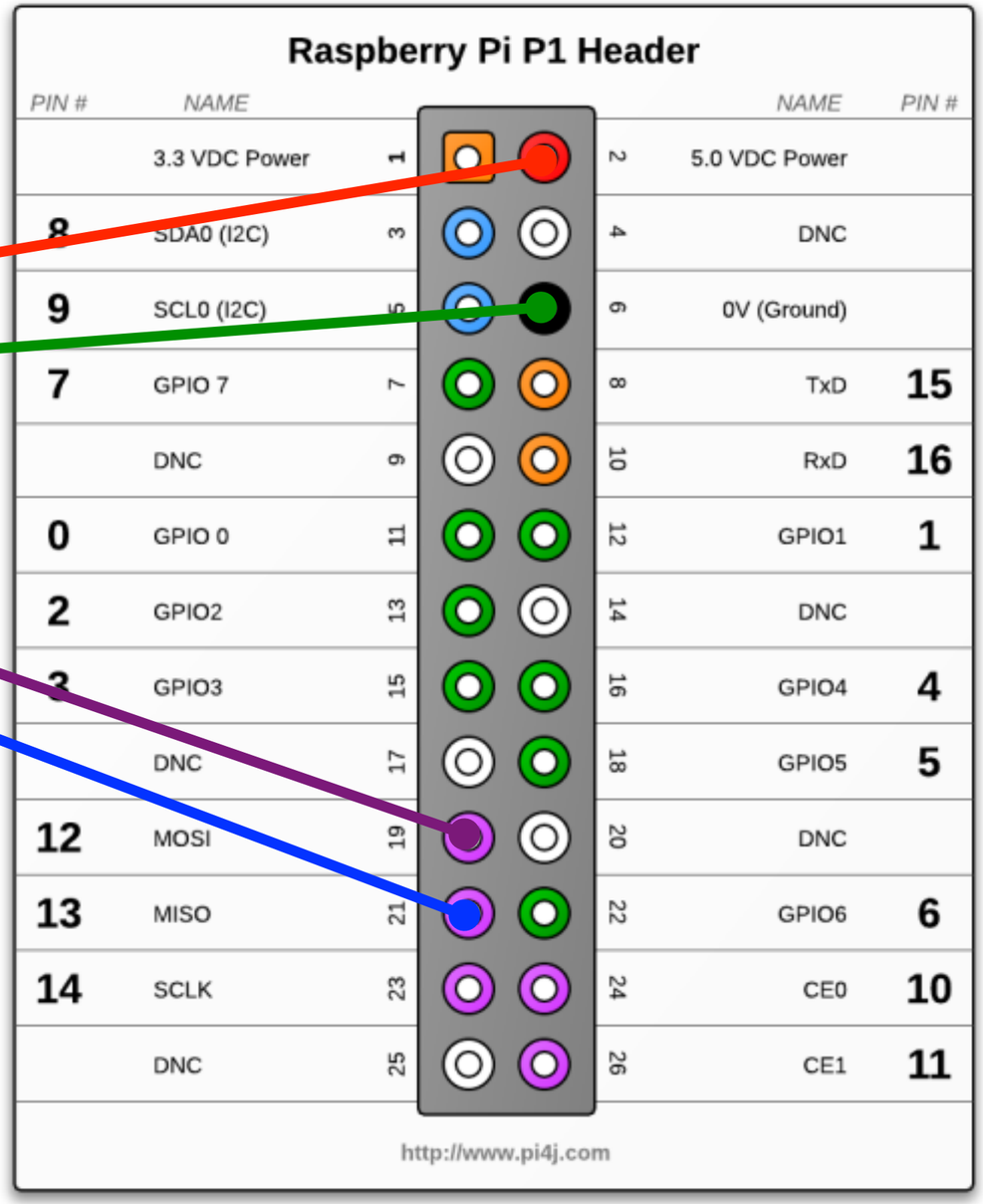
Active-High Relay Board



12 VDC
Power Source

12 VDC
Strobe
Light

12 VDC
Illuminated
Switch



GPIO Output Example

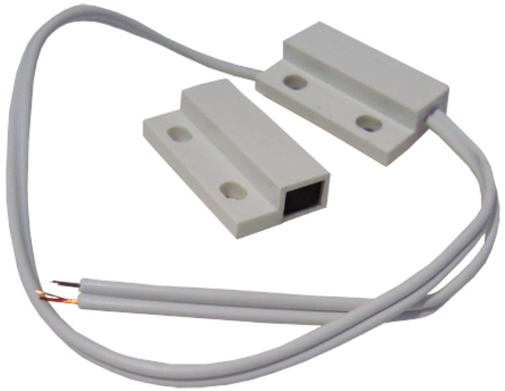
```
// create GPIO controller
final GpioController gpio = GpioFactory.getInstance();

// create a GPIO output
final GpioPinDigitalOutput output =
                gpio.provisionDigitalOutputPin(
                                RaspiPin.GPIO_12, PinState.LOW );

// control GPIO output pin
output.high();
output.low();
output.toggle();        // invert current state
output.pulse(1000);     // set state for a limited duration
```

DEVOXX™

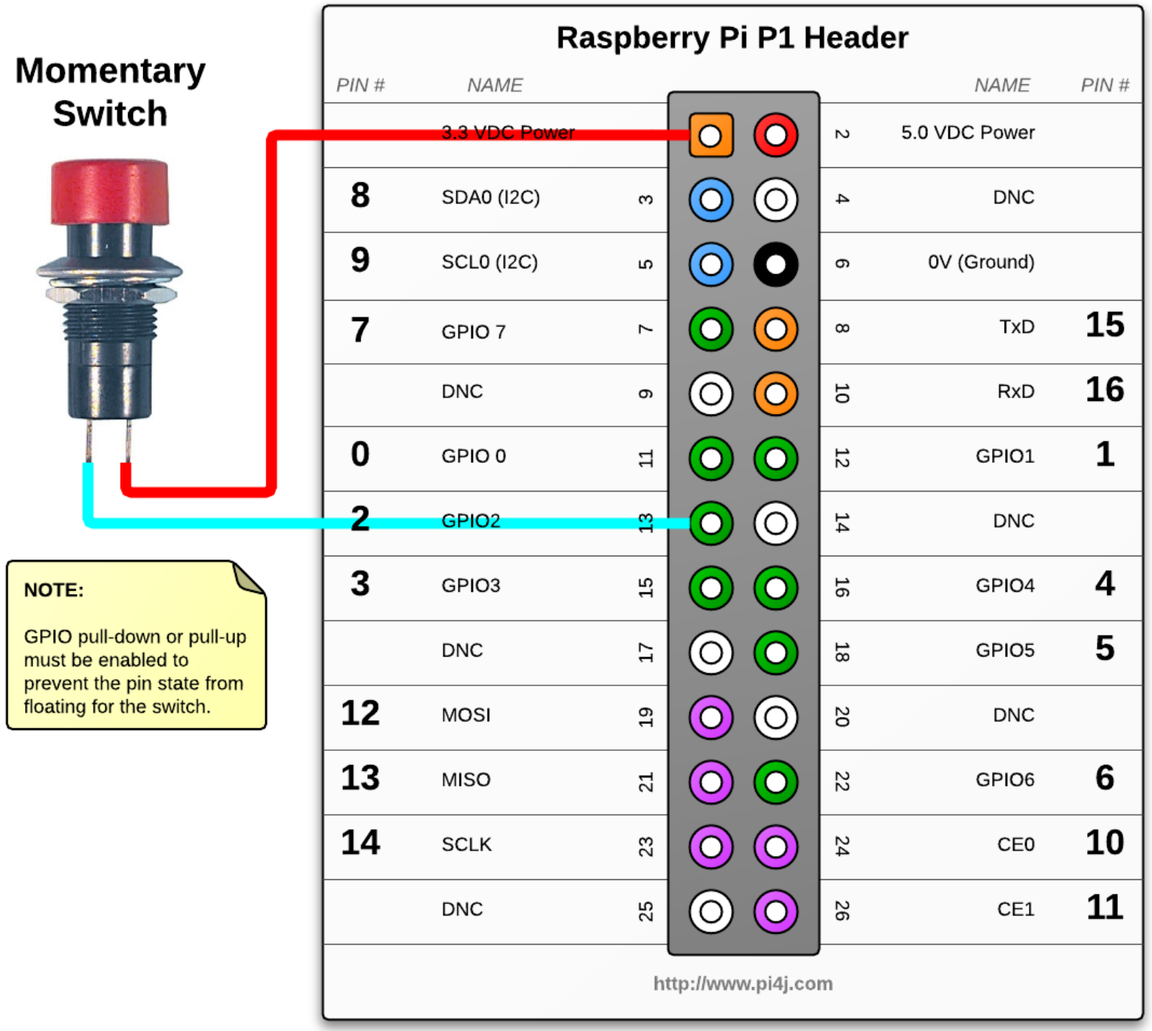
Demo

GPIO Inputs

GPIO Inputs can be used to “sense” things



GPIO Input Circuit



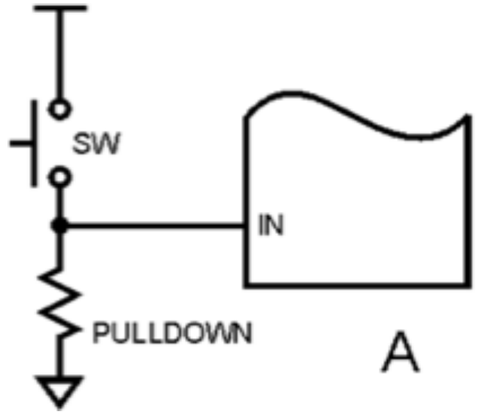
GPIO Input Reference

- GPIO inputs require a "reference" voltage.
- Without a reference, a GPIO pin can "float"
- The Raspberry Pi includes internal PULL-UP and PULL-DOWN resistor settings that can be configured via Pi4J.

GPIO Input Reference

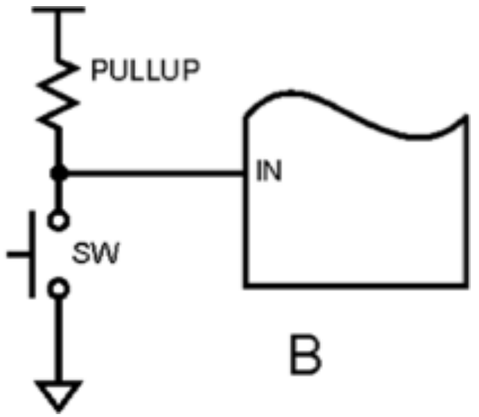
PULL-DOWN

Resistance provides a reference (bias) to GROUND (0 VDC). If your circuit expects to provide +3.3VDC to signal the GPIO pin HIGH, then you need a PULL-DOWN reference.



PULL-UP

Resistance provides a reference (bias) to +3.3 VDC. If your circuit expects to provide GROUND to signal the GPIO pin LOW, then you need a PULL-UP reference.

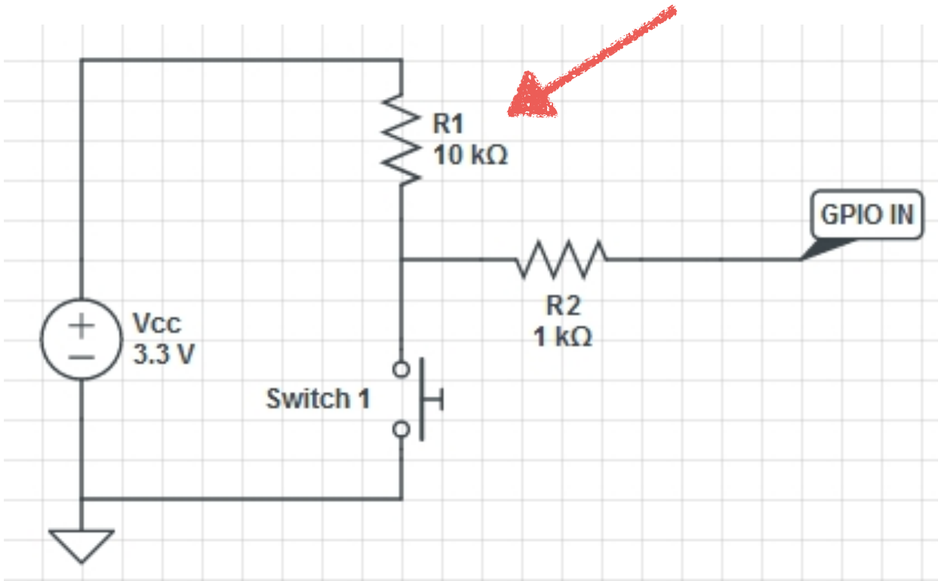


GPIO Input Circuit

Alternatively, you can build the PULL-UP or PULL-DOWN reference in the hardware circuit.

The circuit below demonstrates a PULL-UP resistor at R1.

This circuit signals the GPIO input pin to LOW when the switch closes the circuit and a path to GROUND is complete.



GPIO Input Example

```
// create GPIO controller
final GpioController gpio = GpioFactory.getInstance();

// create a GPIO output
final GpioPinDigitalInput input =
                gpio.provisionDigitalInputPin(
                                RaspiPin.GPIO_02,
                                PinPullResistance.PULL_DOWN);
```

GPIO Input Listener

Java 8 Lambda

```
// create event listener for GPIO input pin
input.addListener((GpioPinListenerDigital)
        (GpioPinDigitalStateChangeEvent event) -> {

    // set output state to match the input state
    output.setState(event.getState());

});
```

DEVOXX™

Demo

Pi4J Component API

- The component APIs provides an abstraction layer from the hardware I/O layer.
- This allows hardware design/circuitry to change with *less* impact to your implementation code.
- For example, a RELAY could be controlled from GPIO, RS232, SPI, or I2C. You program defines the RELAY impl up front based on the hardware interface, but the rest of your program logic works against the RELAY component

Pi4J Component API

- Keypad
- Light / LED
- Dimmable Light
- LCD
- Power Controller
- Relay
- Momentary Switch
- Toggle Switch
- Analog Sensor
- Distance Sensor
- Motion Sensor
- Temperature Sensor

GPIO Components Example

```
// create LED component
final Light light = new GpioLightComponent(output);

// usage example
light.on(); (or) light.off();



// create momentary switch component
final MomentarySwitch ms = new GpioMomentarySwitchComponent(
                  input,
                  PinState.LOW,    // “OFF” pin state
                  PinState.HIGH); // “ON” pin state
```

DEVOXX™

Demo

Smart Devices / Internet of Things

"A smart device is an electronic device, generally connected to other devices or networks via different protocols such as Bluetooth, NFC, WiFi, 3G, etc., that can operate to some extent interactively and autonomously. It is widely believed that these types of devices will outnumber any other forms of smart computing and communication in a very short time, in part, acting as a useful enabler for the Internet of Things."

(reference: http://en.wikipedia.org/wiki/Smart_device)

Smart Devices / Internet of Things

- Network/Internet Accessible (*Connected*)
- Autonomous Behavior
- Interactive Capability
  - Notifications
  - Control

Create A “Smart” Home

- Recently moved into a new home.
- I want to automate the subsystems in the home and make it a “smart” home.
- I want to remotely control and monitor my home.

Off The Shelf Smart Devices

- Lots of emerging IoT Devices
- Proprietary Hardware, Software/Apps & Protocols
- Limited Interconnectivity or Interoperability


D.I.Y. Smart Devices

- Using Raspberry Pi + Java + Pi4J
- Augment existing "dumb" devices and sensors to create new interactive capability and autonomous intelligence.
- With simple GPIO and a little circuitry you can create all kinds of "Smart Devices".

Basement Flood Alarm:

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

Water Level Sensor

RaspberryPi Java Pi4J

Notifications

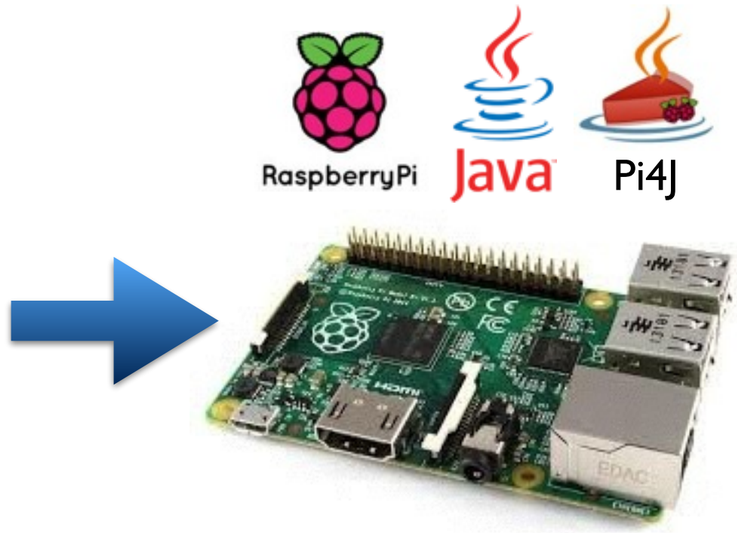
Control other devices

Remotely Monitor

HVAC Alarm:

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

HVAC Condensate Leak Detector
(Moisture Detector)



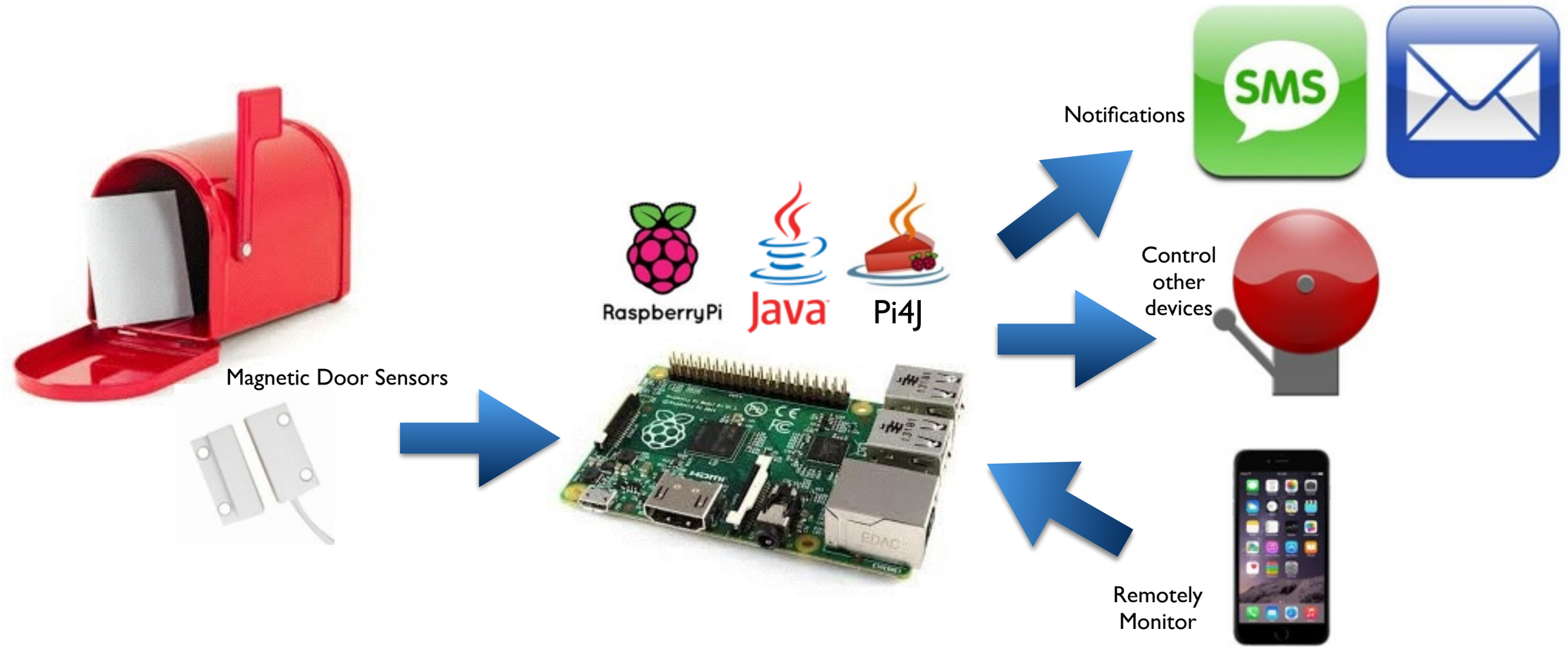
Notifications

Control other devices

Remotely Monitor

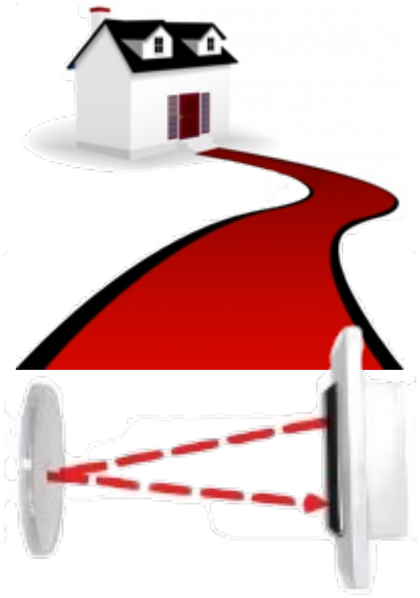
Mail Notification:

- Instrument a mailbox to get notified when mail arrives.

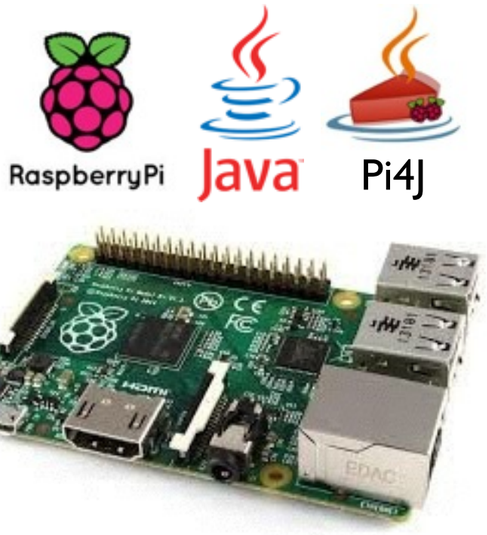


Driveway Alarm:

- Add a sensor to driveway to get notified when someone approaches the house.



Infrared Beam Sensors

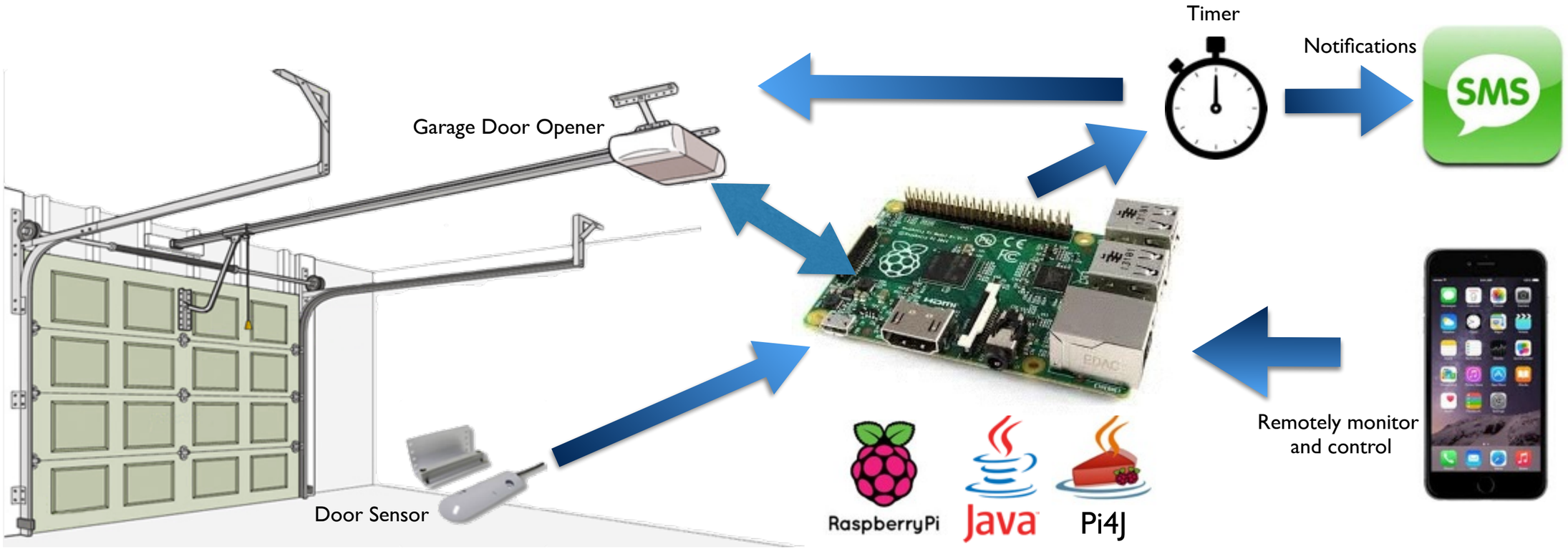


Notifications

Control other devices

Garage Door Opener:

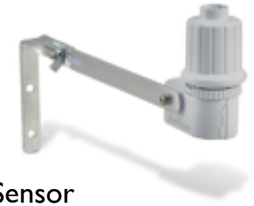
- Remote control and monitoring of garage door
- Auto-close if left open



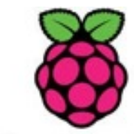
Sprinkler System

- Remotely control, configure and schedule the system.
- Skip watering schedules if raining or if rain is forecasted

Sprinkler Controller



Rain Sensor



RaspberryPi

Java

Pi4J

Schedule

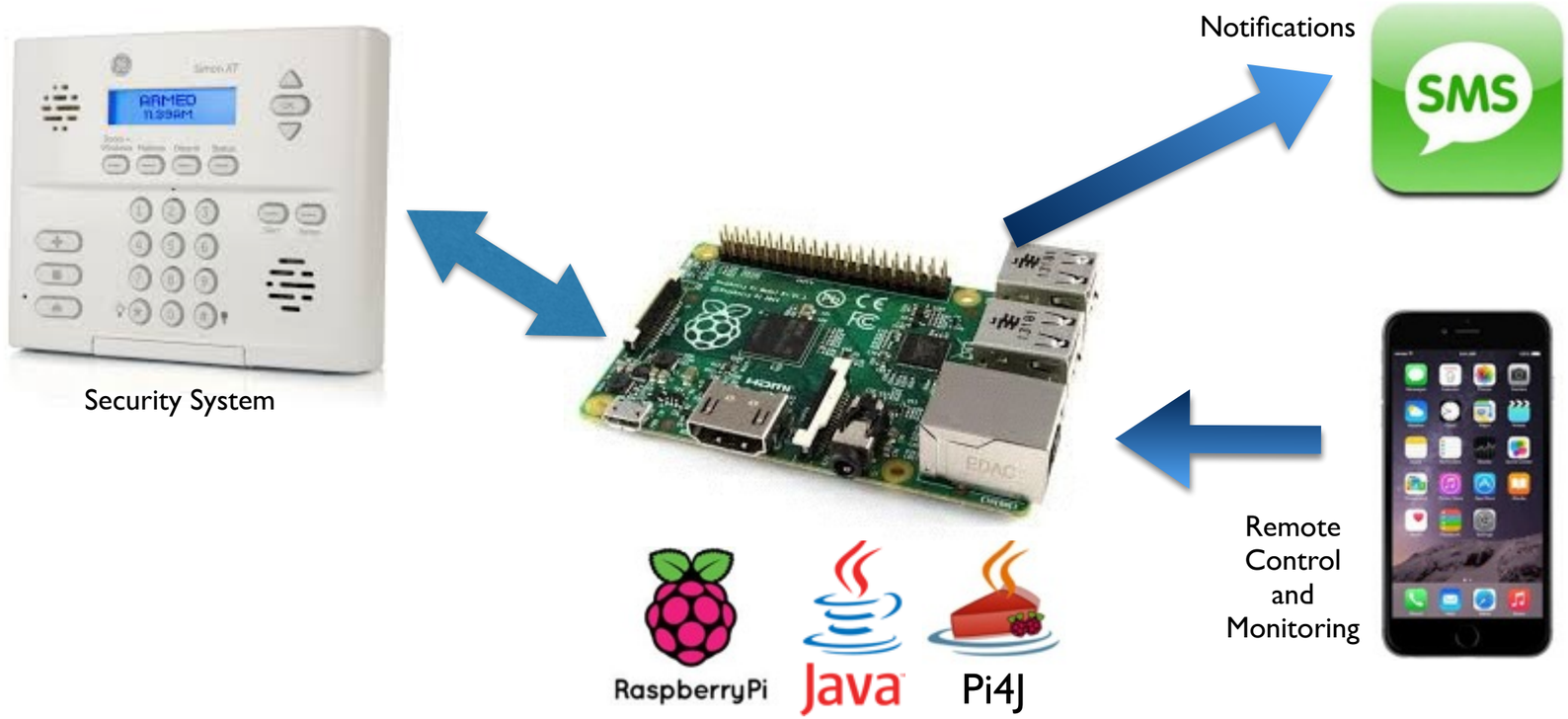
Notifications

Weather Forecasts

Remote Control
and Monitoring

Security System

- Remote control and monitoring of the system
- Activate other devices based on the state of the system



HVAC System

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

Thermostat

Notifications

Remote Control and Monitoring

RaspberryPi Java Pi4J

Let's Build a Proof of Concept Demo

- Driveway Alerts
- Flood Alerts
- Mailbox Notifications
- HVAC System Alerts

DEVOXX™

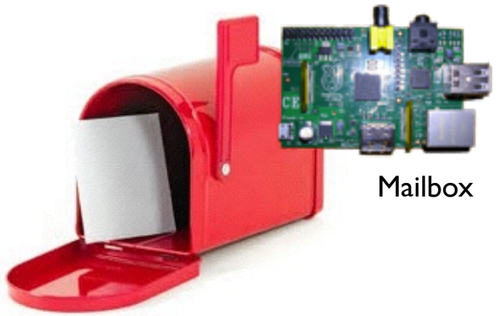
Demo

What About Connectivity?

- How do we aggregate monitored data?
- How do we share information between sensors and devices to make intelligent automation decisions?
- How do we access data and control these “smart” devices remotely?

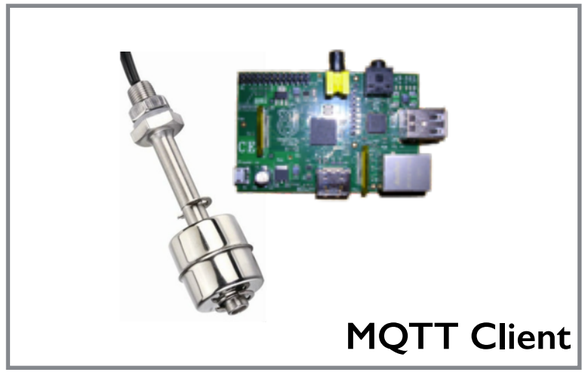

Basement

Attic

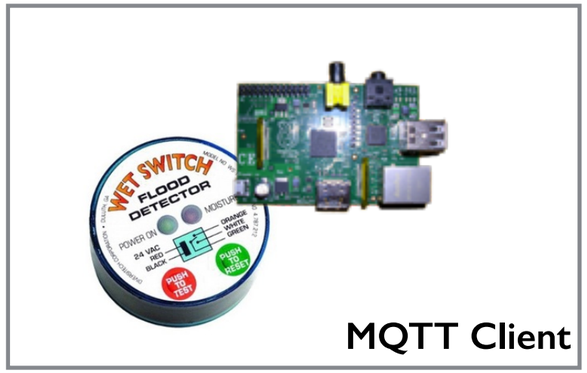

Mailbox


Garage

MQTT



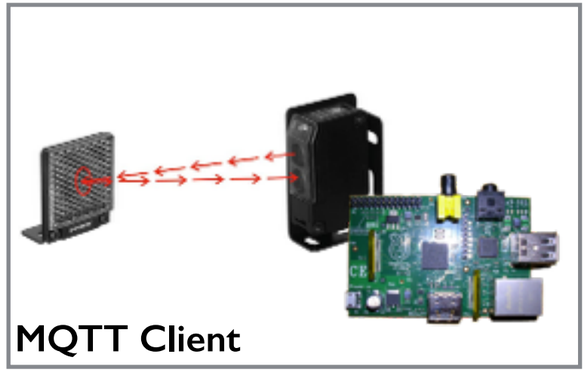
Basement



Attic

MQTT
BROKER

Mailbox

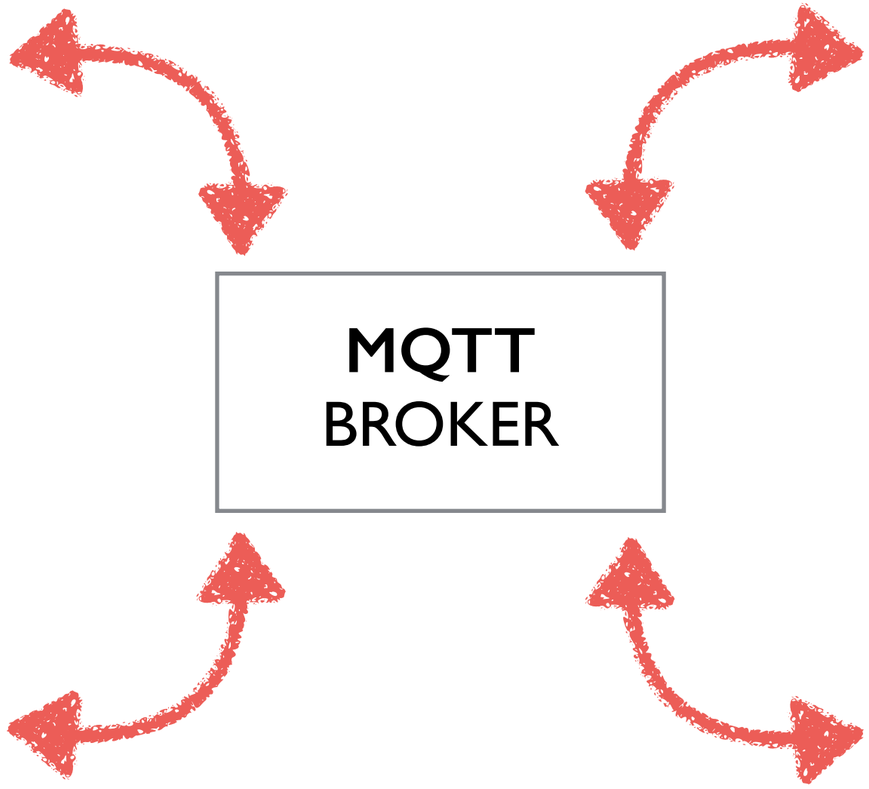


Garage

MQTT

Desktop

Tablet



Web Server

Mobile

MQTT Overview

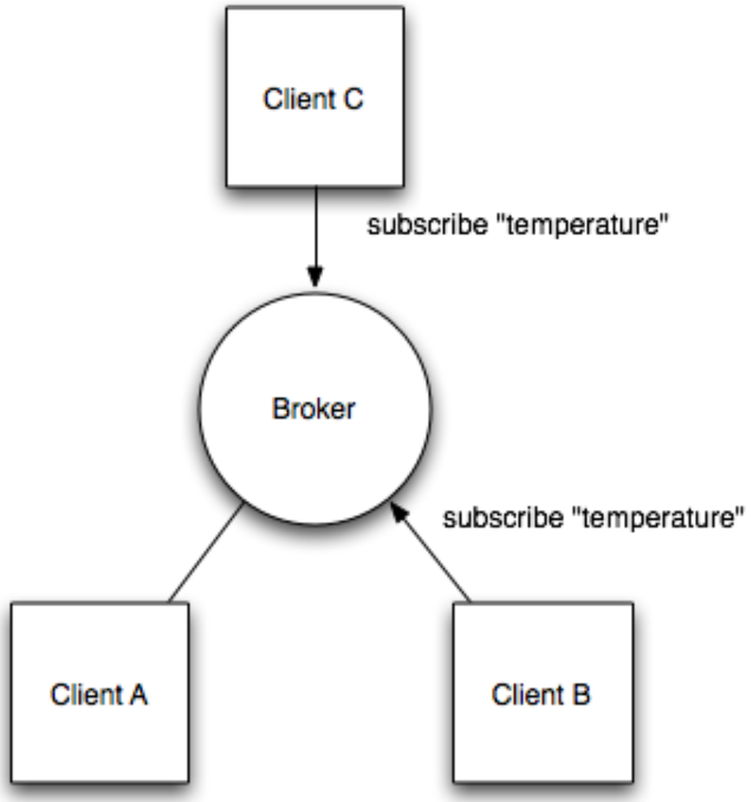
- MQTT == “Message Queuing Telemetry Transport”
- M2M connectivity protocol
- Lightweight Communications
- Small Code Footprint
- Open Standards Based
- Asynchronous Messaging
- Emerging protocol competing to be an IoT “standard” protocol

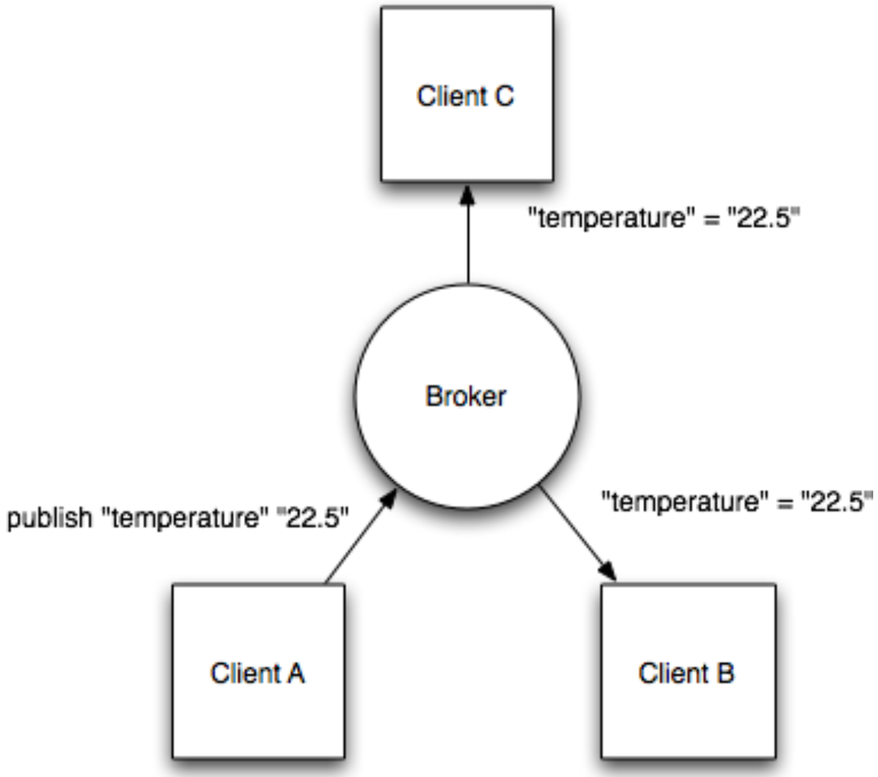
MQTT Features

- Publish / Subscribe Messaging Transport (*topics*)
- Last Will & Testament
- QOS (Quality of Service)
- Message Persistence
- Supports Binary Data (Payloads)
- TCP Based
- Security (SSL/TLS)

MQTT Topics

- Publishers, Subscribers & Broker





(images from eclipse.org)

MQTT Software

- Mosquito Broker (Server)
  http://www.mosquitto.org/

- Eclipse Paho (Client)
  http://www.eclipse.org/paho/

- MQTT.Fx
  http://mqttfx.jfx4ee.org/
  by Jens Deters (@Jerady)

DEVOXX™

Demo

2015 Goals

- Start creating DIY “smart” devices using Raspberry Pi, Pi4J, and Java.
- Create a blog article for each project and publish the wiring diagrams, source code, and materials listing.
- Implement these DIY smart devices throughout my home and create a “smart” home.
- Aggregate the information from these connected smart devices to start creating an intelligent and automated home

DEVOXX™

Questions

Savage Home Automation Blog

savagehomeautomation.com

@savageautomate

The Pi4J Project

pi4j.com

@pi4j

Sides & source code available now at http://www.savagehomeautomation.com/devoxx