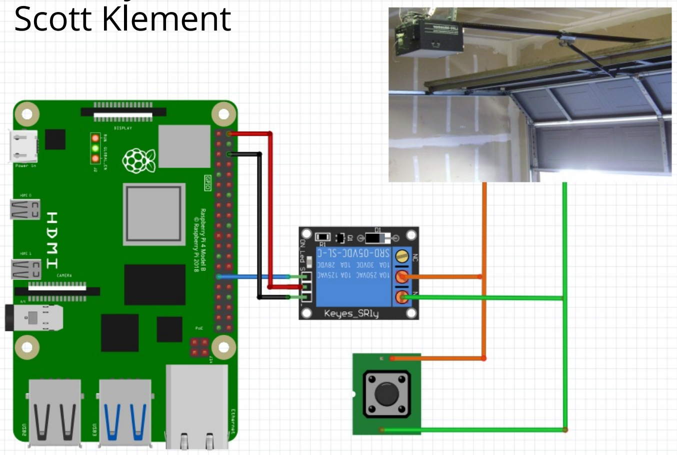
IoT: Open My Garage Door with RPG

by
Scott Klement



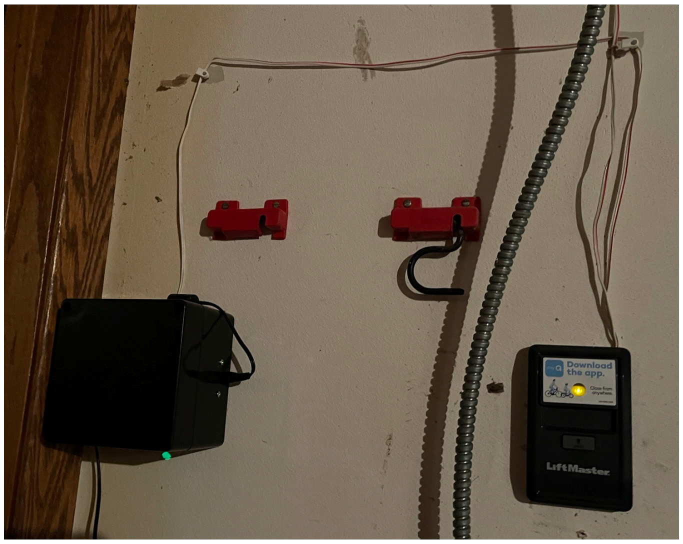
```
open   GarageDoor;

close  GarageDoor;

*inlr = *on;
```

Open My Garage

```
**free

dcl-f GARAGEDOOR disk(1)
                 handler('DOOROA')
                 usropn;

open GarageDoor;

close GarageDoor;

*inlr = *on;
```

This is an actual, working, RPG program.

My Homemade Opener



The box on the left contains two things:

- Raspberry Pi (ZeroW)
- A relay
- Wires that connect to the garage door switch

What?

Two Raspberry Pi computers (with my hand for comparison)

- Zero-W (approx. $10)
- 4B 4GB (approx. $60)

The Raspberry Pi is a small, inexpensive, energy efficient computer.

- Storage on microSD cards
- HDMI video/audio
- USB ports for keyboard/mouse
- Wired & Wifi Networking
- ARM-based CPU
- GPIO pins

Why Why Ras Pi?

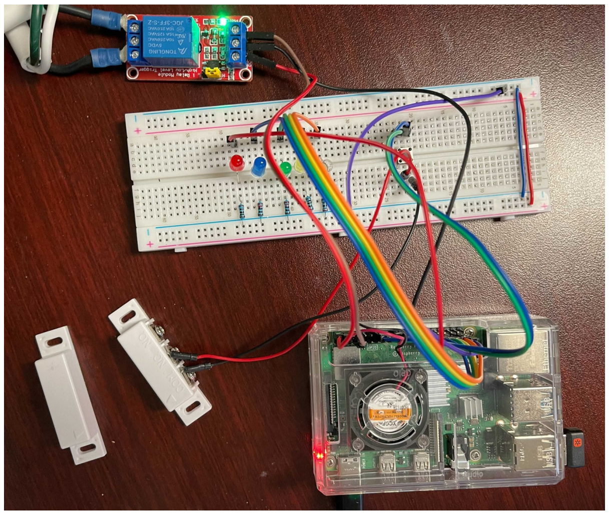
Small, efficient, and inexpensive.

- Stick it under a table
- Back of a monitor
- Inside/above/under a cabinet
- Inside a vehicle
- …really, anywhere…

Use it to control electronics, and interface with a bigger computer.

- Such as your PC
- Power Systems running IBM i

Introduction to Raspberry Pi



Scott will demonstrate (or show a video that demonstrates) the basics of the Raspberry Pi.

What Are GPIO Pins?

GPIO = General Purpose Input/Output

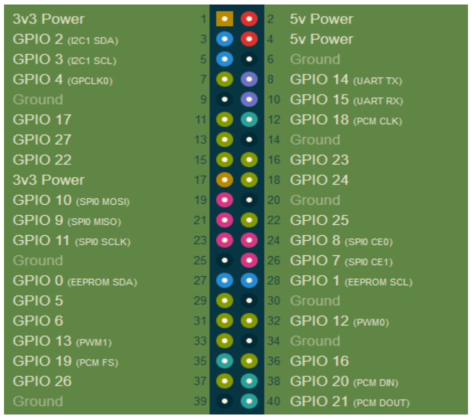
They can be opened for:

- Input -- check to see if there's voltage.
- Output -- Send voltage down the pin

Some terms:

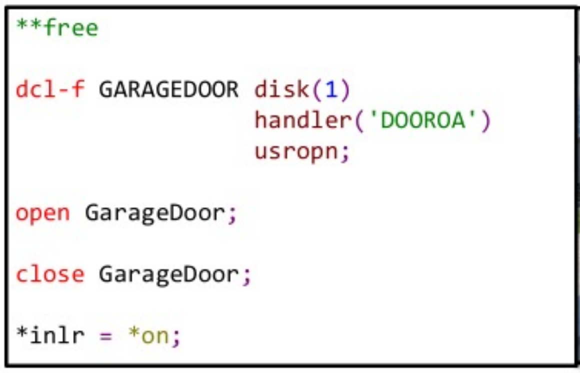
- HIGH = means voltage is present
- LOW = means voltage is off
- Rising Edge = The point at which voltage turned on.
- Falling Edge = The point at which the voltage went off.

Note: The physical pin on the board does not typically match the GPIO number. See diagram on right.



3v3 Power	1	2	5v Power
GPIO 2 (I2C1 SDA)	3	4	5v Power
GPIO 3 (I2C1 SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (UART TX)
Ground	9	10	GPIO 15 (UART RX)
GPIO 17	11	12	GPIO 18 (PCM CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3v3 Power	17	18	GPIO 24
GPIO 10 (SPI0 MOSI)	19	20	Ground
GPIO 9 (SPI0 MISO)	21	22	GPIO 25
GPIO 11 (SPI0 SCLK)	23	24	GPIO 8 (SPI0 CE0)
Ground	25	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM DIN)
Ground	39	40	GPIO 21 (PCM DOUT)

Back to the Garage!



```
**free

dcl-f GARAGEDOOR disk(1)
                 handler('DOOROA')
                 usropn;

open GarageDoor;

close GarageDoor;

*inlr = *on;
```

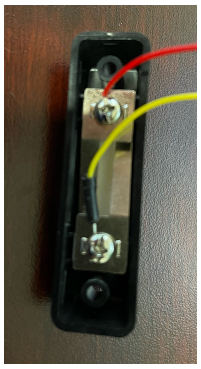
Let's go back to the garage door example, and I'll show you how it's coded!

Remember our goal: Make the RPG OPEN opcode open a garage door.

To open a garage door, we push a button…

What Does the Button Do?

The typical garage door button is very simple, it's just a momentary switch. It connects the two wires while you are pressing it.



Think about when you open your garage door:

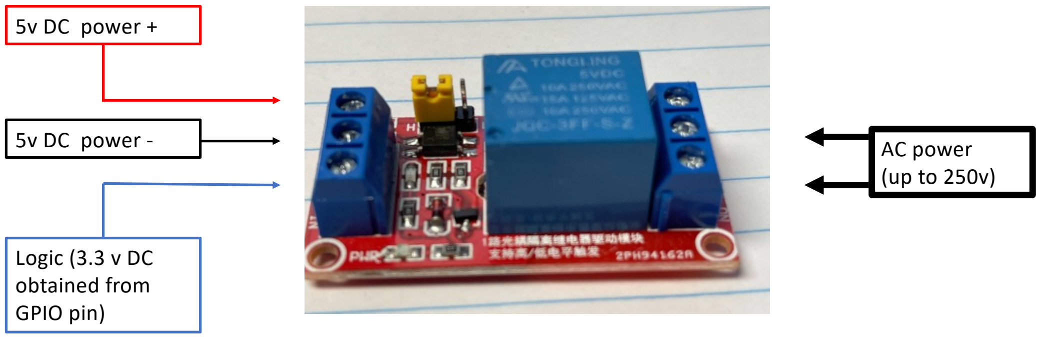
- You push the button
- Hold it for a second or so
- Then release it

Rising/Falling

Another way to say it:

- When you are holding the button, it is "high"
- When you are not, it is "low"
- When you first push it in, the electricity incrases to high. That's the "rising edge"
- When you release it (no matter how long you held it in) the electricity drops -- that's the "falling edge"
- The garage door opener will activate on the falling edge.

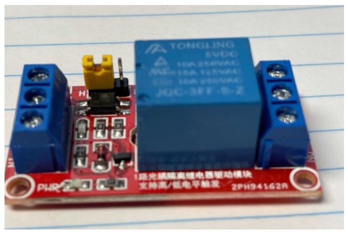
What is a Relay?



A relay is a switch that can be "flipped" electronically.

- Low volt power (from the Raspberry Pi) is connected on one side.
- Logic wire controls whether on or off.
- Opposite side is connected/disconnected via switch.
- Opposite side can be higher voltage (up to 250 v AC. Though, only 24 v AC was needed for my garage door.)

Creating the Falling Edge



We can therefore:

- Turn on a GPIO pin to signal the relay to connect the wires.
- Wait for a second or so
- Turn off the GPIO pin to signal the relay to turn off.
- This creates a "falling edge" -- so will activate the garage door opener!
- *As far as the opener knows, a person just pressed the button.*

Python Code to Turn Pin On/Off

This is Python code that runs on the Raspberry Pi. Python is a simple language to learn, so is a good place to start!

```
import time
import RPi.GPIO as GPIO

pin = 5
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin, GPIO.OUT)

GPIO.output(pin, GPIO.HIGH)
time.sleep(1)
GPIO.output(pin, GPIO.LOW)

GPIO.cleanup()
```

Python w/Flask (1 of 3)

But, of course, we want to activate this from an RPG program on an IBM i over a network. To do that, we can use "Flask" (a simple HTTP server) to make it into a REST API -- then call that API from RPG.

```
import time
import RPi.GPIO as GPIO
from flask import Flask, request

pin = 5
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin, GPIO.OUT)

app = Flask(__name__)
```

Python w/Flask (2 of 3)

http://192.168.0.1:5000/openDoor/myPassword?mode=on

```
@app.route('/openDoor/<password>')

def openDoor(password):
    if password == 'myPassword':
        mode = request.args.get('mode')
        try:
            if mode == 'on':
                GPIO.output(pin, GPIO.HIGH)
                time.sleep(1)
                GPIO.output(pin, GPIO.LOW)
            else:
                GPIO.output(pin, GPIO.HIGH)
                time.sleep(1)
                GPIO.output(pin, GPIO.LOW)
        finally:
            return 'success';
```

Python w/Flask (3 of 3)

```
if __name__ == '__main__':
    app.run(debug=False, host='0.0.0.0')

idle

GPIO.output(pin, GPIO.LOW)
GPIO.cleanup()
```

Open Access Handler (1 of 3)

```
dcl-f GARAGEDOOR disk(1) handler('DOOROA') usropn;
```

```
**free
ctl-opt dftactgrp(*no) actgrp('KLEMENT')

option(*srcstmt:*nodebugio:*noshowcpy)
        bnddir('HTTPAPI');

/copy QOAR/QRPGLESRC,QRNOPENACC
/copy HTTPAPI_H

dcl-pi *n;
  io likeds(QrnOpenAccess_t);
end-pi;

io.rpgStatus = 0;
```

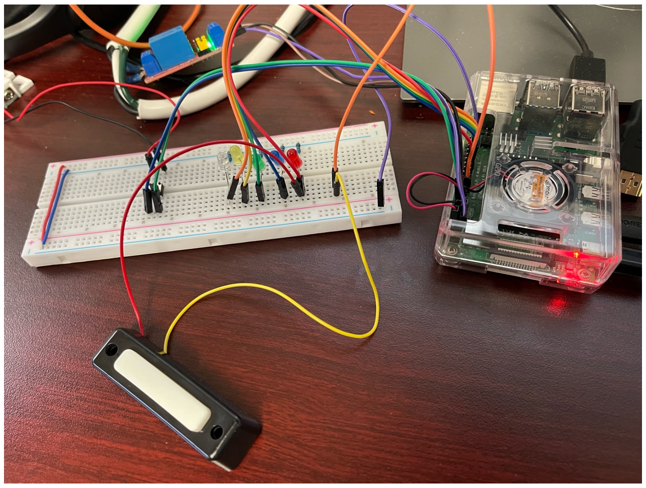
Open Access Handler (2 of 3)

```
select;
when io.rpgOperation = QrnOperation_OPEN;
  makeRestCall('on': io.rpgStatus);

when io.rpgOperation = QrnOperation_CLOSE;
  makeRestCall('off': io.rpgStatus);

other;
  io.rpgStatus = 1299; // 1299 = Other I/O
error detected
endsl;

return;
```

Open Access Handler (3 of 3)

```
dcl-proc makeRestCall;

  dcl-pi *n;
    mode   varchar(3) const options(*trim);
    status int(10);
  end-pi;

  monitor;
    http_string( 'GET'
               : 'http://scottraspi4b:5000/openDoor/+
                  myPassword?mode=' + mode);
  on-error;
    status = 1217;  // status 1217 = File not found.
  endmon;

end-proc;
```

Show Me!



Demonstration.

Will walk through the code so you can see it running.

Physical Computing

Physical Computing lets you write programs that interact with the physical world!

- Turn stuff on or off.
- Read sensors (temperature, pressure, "eyes"/infrared, cameras)
- Make conveyors move.
- Interface printers, scanners, scales, industrial terminals…
- The possibilities are endless!
- Motors/Servos
- Ultrasonic sensors
- Solenoid valves

Internet of Things (IoT)

When you do physical computing and make it available to the Internet (even if protected with passwords, encryption, VPNs, etc) it's called IoT -- or "internet of things".

Basically, my garage door is the "thing", and I've made it available to the Internet.

Now I can open/close my door from an RPG program, even though its running on an IBM i in Ohio.

That's not all, by the way -- I also wrote an app for my cell phone, and even routines for my Amazon Echo (Alexa) so I can open/close the door that way, too.

Imagine...

I demonstrated turning LEDs, a light on/off and opening/closing a garage door.

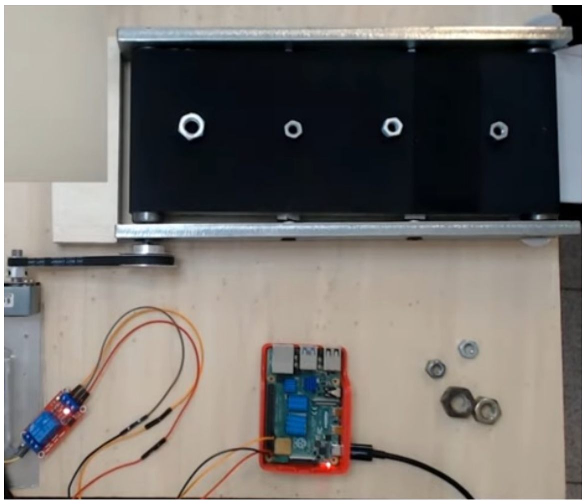
Think of all of the other things you could turn on or off!

And it's done by program logic, so could be done under any logic you can imagine.

Likewise, I demonstrated reading from a door sensor switch.

Think of all of the other sensors and devices you can read from.

Conveyor

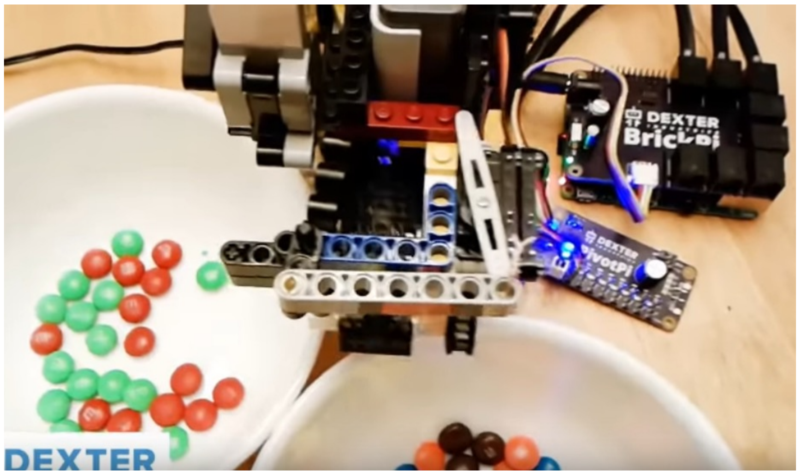


There are cameras available that work well with image detection software (such as OpenCV)

This project stops the conveyor belt when a nut over a given size is detected at the end of the conveyor. The worker could then remove the one that's too large.

With servos you could build a robotic arm (or buy a prebuilt one) that removes the nut. (Or any other type of item.) -- not shown.

Sorting M&Ms



Sorts M&Ms by color. Pretty cool, to play with -- but there are also industrial uses for it.

Two Main Languages on RaspPi

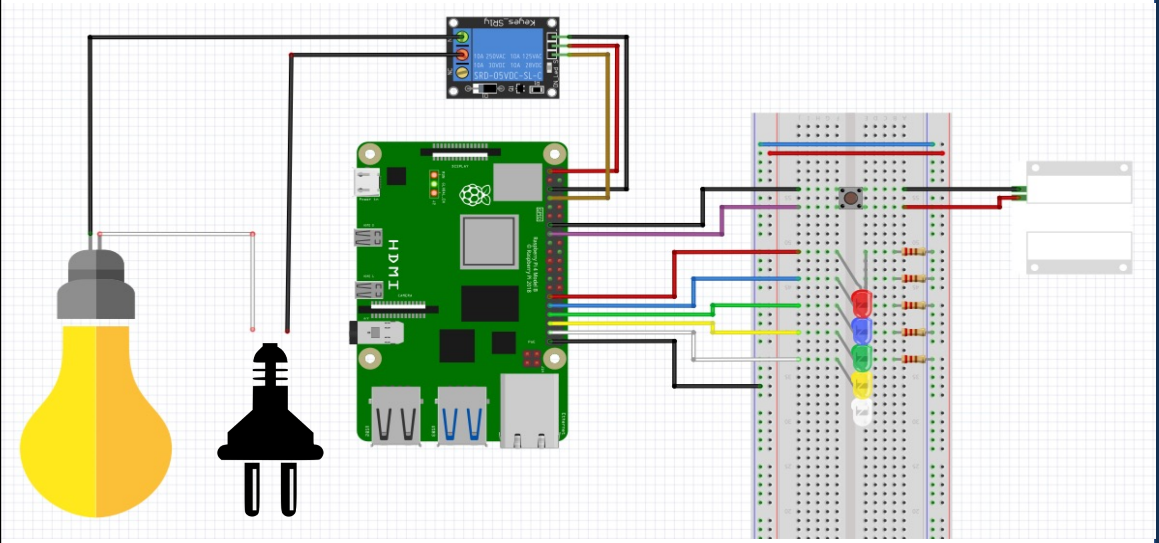
Python

Node.js

Two Main Languages on Raspberry Pi

Python	Node.js
Easy to learn	Extremely efficient for:
Extremely popular	• Web applications
Lots of examples of IoT projects	• REST APIs
Favored by electrical engineers	• Waiting for electronics
Lots of open source plugins	More complex to learn
	Also very popular
	Favored by software engineers
	Even more open source plugins

Diagram of the Intro Project



Intro Project Used Node.js

Interested in the code for the LED, Relay and Door Switch project I showed earlier to introduce Node.js?

The entire code is on GitHub, if you want to see it.

https://github.com/ScottKlement/rpg-raspi-demo

But will give a basic introduction to how it worked:

- Used a Node.js module called "onoff"
- Has functions for read/writing GPIO both synchronously and asynchronously.
- I will show you some quick examples

Node Writing GPIO

To turn one on, you simply write 1 to it. To turn it off, write 0.

```
var Gpio = require("onoff").Gpio;

var led = new Gpio(5, 'out');

// Turn on
led.writeSync(1);

// Turn off after 5 seconds
setTimeout(() => led.writeSync(0), 3000);
```

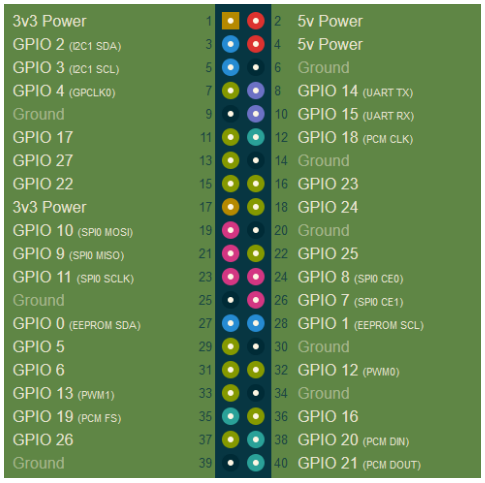
GPIO Toggle

Using "onoff" on Raspberry Pi, you can read from pin (even if it is in 'out' mode)

This makes it easy to toggle.

```
var Gpio = require("onoff").Gpio;

var relay = new Gpio(4, 'out');

function toggle() {
  var currentValue = relay.readSync();
  relay.writeSync(currentValue ^ 1);
}

toggle();
setTimeout(toggle, 3000);
```

Ease of Async Operations



3v3 Power	1	2	5v Power
GPIO 2 (I2C1 SDA)	3	4	5v Power
GPIO 3 (I2C1 SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (UART TX)
Ground	9	10	GPIO 15 (UART RX)
GPIO 17	11	12	GPIO 18 (PCM CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3v3 Power	17	18	GPIO 24
GPIO 10 (SPI0 MOSI)	19	20	Ground
GPIO 9 (SPI0 MISO)	21	22	GPIO 25
GPIO 11 (SPI0 SCLK)	23	24	GPIO 8 (SPI0 CE0)
Ground	25	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM DIN)
Ground	39	40	GPIO 21 (PCM DOUT)

Imagine you wanted to wait for a button to be pressed on GPIO 23.

You don't want to sit in a loop, constantly reading the pin -- this would use a lot of CPU.

Plus, you wouldn't be able to handle REST requests at the same time!

Async Button

onoff provides a watch event that can fire a function when the state of a button changes. It can be 'falling' (for the falling edge), 'rising' (for the rising edge) or 'both'.

There's also a debounceTimeout to avoid the situation where a button might open/close more than once rapidly.

```
var Gpio = require("onoff").Gpio;

var button = new Gpio(23, 'in', 'both',
                      {debounceTimeout: 10});

button.watch((err, value) => {
  console.log((value===1) ? 'up':'down');
});
```

Questions?

For this presentation as well as the sample code, visit my web site:

http://www.scottklement.com/presentations/