Make Your Own PiDoorbell!

Handout for PyCon, Montreal, April 2014

Rupa Dachere and Akkana Peck, co-presenters, for CodeChix

Parts List:

- Raspberry Pi (you provide) and power supply (preferably 1 amp or better)
- SD card loaded with Raspbian and other required software
- HC-SR04 ultrasonic rangefinder
- Solderless breadboard
- Jumper wires: 4 male-female, 2 male-male
- Resistors: 1 360 Ω (or optionally 2), 1 470 Ω
- LED
- USB to TTL Serial Cable for Rpi, OR ethernet cable (for RPi-laptop communication)

Optional:

- Camera for RPi: either USB webcam or Pi Camera
- USB wi-fi dongle
- cell phone (for text alerts)

Software

Installed on the Raspberry Pi: Raspbian (or comparable distro), RPi.GPIO, git, fswebcam; PiDoorbell repository, installed via: git clone https://github.com/codechix/PiDoorbell

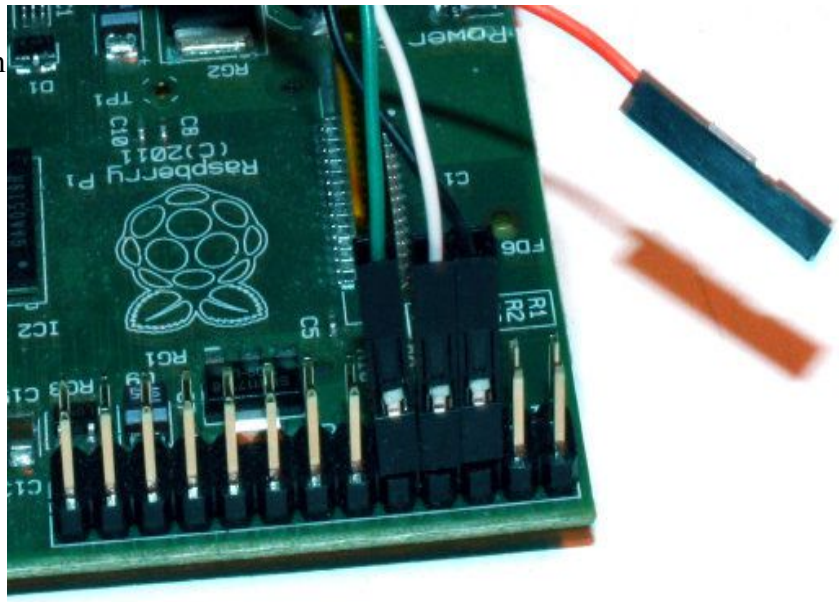
On your laptop: some means of communicating with the RPi. Either a program like screen plus drivers for a USB-serial device (most Linux systems and some Mac systems have this already), or a way of using IP masquerading over an ethernet cable.

First step: conect to the Raspberry Pi

Without a monitor and USB keyboard, there are two ways of talking to a Raspberry Pi: over a serial cable, or over ethernet.

Using a serial cable is more reliable, lets you see boot messages and is easiest on Linux.
Using ethernet takes more setup, but lets you give your Pi ethernet access through your laptop, and is easier on some Macs.

Connect with Serial Cable



Wire up the serial cable as shown.
The black wire goes on the third pin from the corner; the white wire goes next to that, and then the green.

DO NOT USE THE RED WIRE!
If you do, the Pi will try to get power from that wire (over the USB) which conflicts with getting its power from its micro-USB power plug. Just leave that wire hanging free.

Once the serial cable is connected to the Pi, plug the USB end into your laptop. Then run this command:

```
screen /dev/ttyUSB0 115200
```

You can use any other program you prefer for talking to a serial port: minicom, etc. The device ID may be something other than /dev/ttyUSB0, depending on your OS.

Once you have a connection established, you should be able to plug in the Raspberry Pi's power supply and see boot messages.

Connect with Ethernet Cable

Connect the two ends of the ethernet cable to the Pi and your laptop.

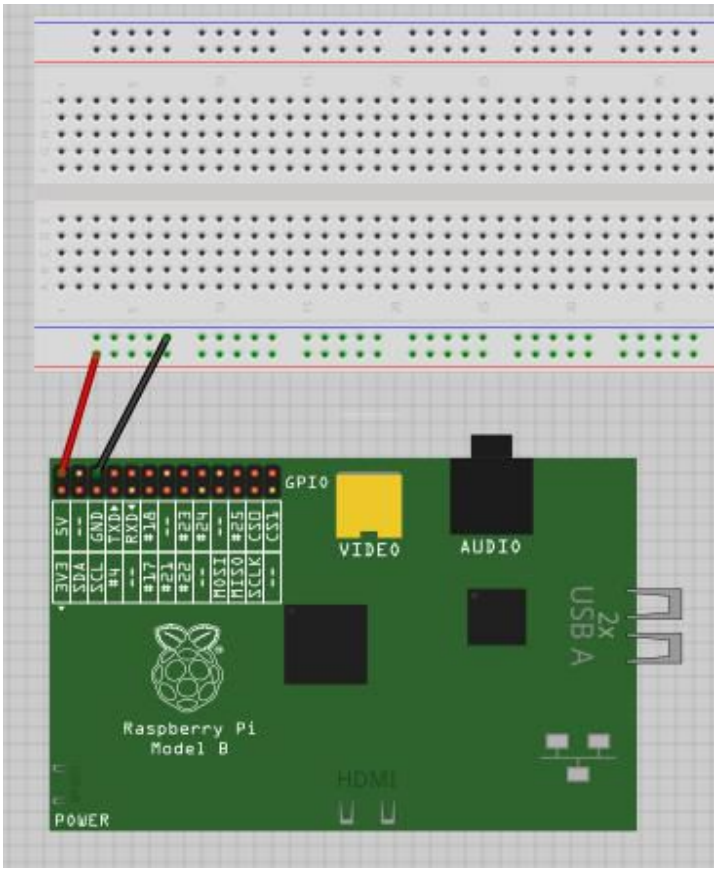
[insert instructions here]

Now see if you can log in to your RPi. Try a command like:

```
ssh pi@192.168.0.yournumber
```

(each Raspberry Pi in our tutorial has its own static IP for ethernet).
Of course, you can use an ssh client like putty if you prefer.

Using GPIO: Blinking an LED



1. Wire Power and Ground

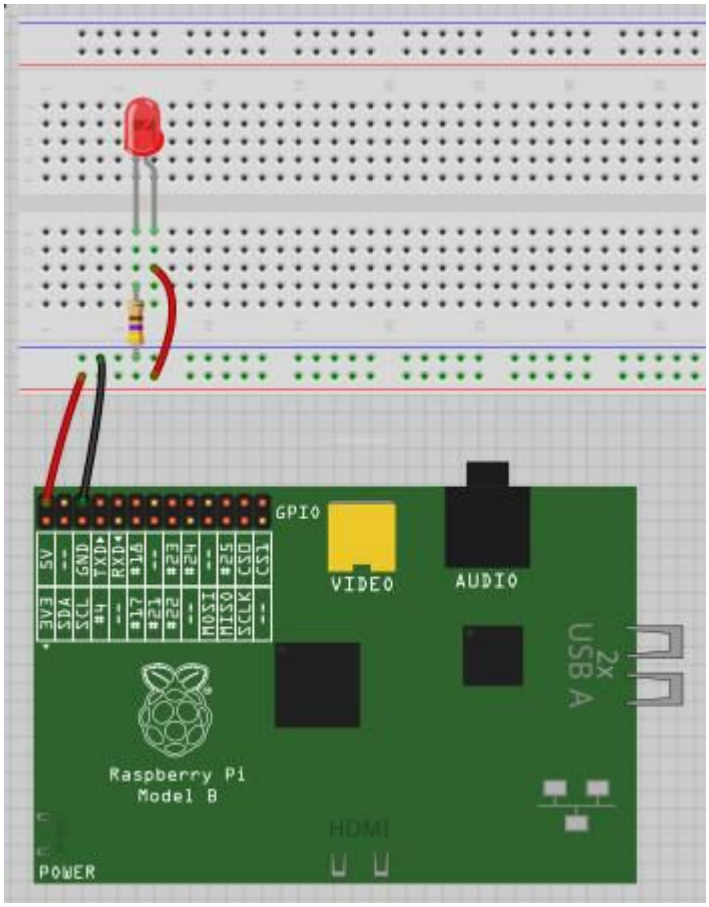
You'll be using the Raspberry Pi's GPIO header. The first step is to wire up power and ground to your breadboard.

On a solderless breadboard, the strips along the edges of the board – labeled with red and blue – are generally used for power (red) and ground (blue).

On the Raspberry Pi's GPIO header, the pin in the corner gives you 5 V power – the power supply the RP is getting from its power input.

The third pin from the corner, in the outer row, is ground. (There are several other pins that can provide ground, if you need more than one. But since we're using a breadboard that's not a problem.)

Use male-female jumpers the female end of each jumper fits over the Pi's GPIO pin, and the male end plugs into the breadboard. If possible, use red for power and black for ground.



2. Wire up the LED

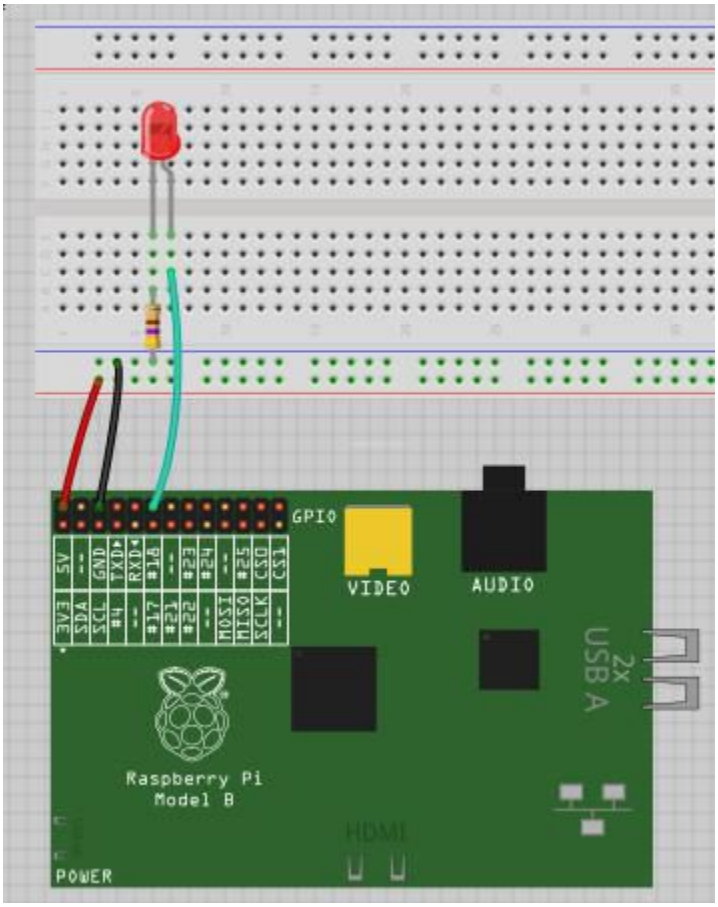
An LED has one leg longer than the other. The longer leg should be connected to power, the shorter one to ground. But you need a small value resistor in the circuit so you don't burn out the LED.

Use your 360 Ω resistor for this. It's the one with red and orange bands.

In the middle part of the breadboard (between the power/ground strips), each line of 5 is wired together. So with this wired up, the current flows from the Pi to the red power strip, to the long pin of the LED, through the LED, through the resistor, out to the ground (blue) power strip of the breadboard, and out to the Pi's ground pin.

If your Pi is plugged in and running, the LED should light up as soon as you finish connecting the wires.

3. Connect the LED to a pin you can control



It's not much fun to have an LED always on. So instead of connecting the long pin to the breadboard's power rail, connect it to pin 18 of the Pi's GPIO header.

The pin numbering on the GPIO header doesn't make any sense, so you should always check a reference to figure out which pin is which. And worse, there are two different incompatible ways of numbering the pins. We'll be using the "BCM" numbering system. BCM pin 18 is the 6th pin from the corner on the outer row.

The LED should be off now. Try running the led.py script. You'll have to run it as root: only root can access the Raspberry Pi's GPIO pins.

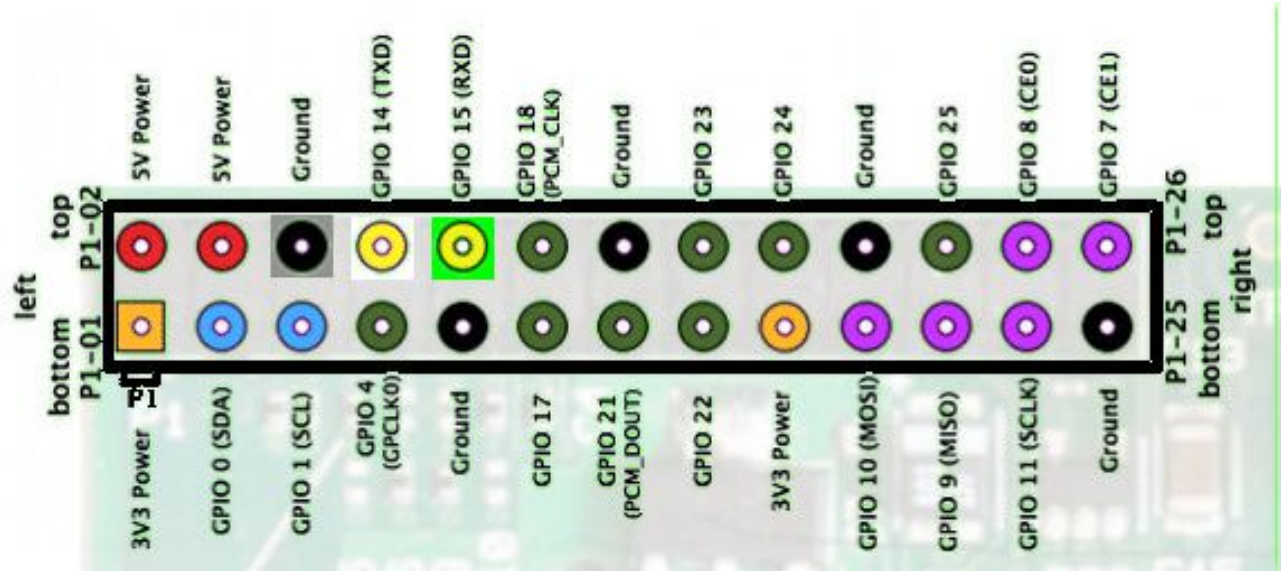
```
sudo python led.py
```

Your LED should start blinking.

You can edit *led.py* and adjust the sleep times.

GPIO Pin Diagram

Here's a diagram showing all the pins on the Raspberry Pi's GPIO header.
Warning: some pins may vary between different Pi versions.



Hooking up the sonar rangefinder

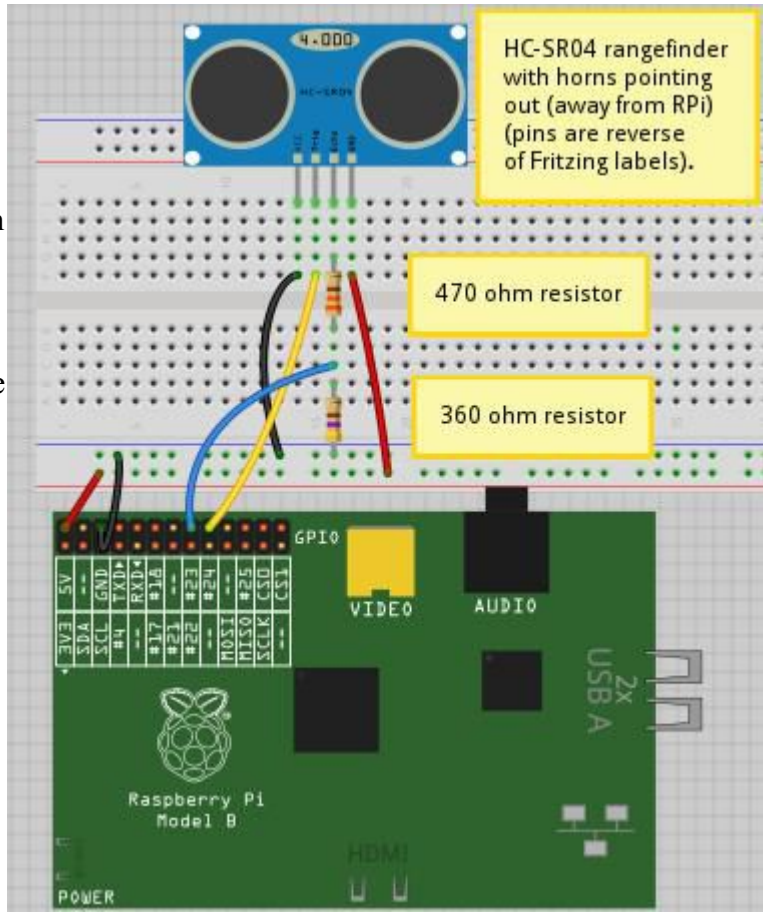
The HC-SR04 distance sensor emits a sound pulse, then measures the time it takes for the sound to return. Same principle as bats, dolphins or submarines.

The HC-SR04 has 4 pins. If you point the sonar horns away from you, the pins are, from left to right: Gnd (ground), Echo, Trigger, VCC (5 V power).

We'll connect power and ground, and wire Trigger to BCM pin 23, Echo to pin 24. Those are the 5th and 6th pins from the audio connector end of the GPIO header on the outer row.

There's one snag: the HC-SR04 runs on 5V, while the Raspberry Pi's GPIO pins can only handle 3V (even though it has a 5V power pin). We can compensate for that by using two resistors as a "voltage divider" to convert 5V to a level the Pi can handle.

Remember, your 360 Ω resistor is the one with red and orange bands. **Be careful not to get the resistors backwards. Too much voltage on a GPIO pin can damage your Pi!**



Controlling the Rangefinder

To control the HC-SR04, we need to turn the Trigger line high briefly, then turn it low again. The Echo line goes low. When the sound pulse returns, the Echo line goes high. So we just need to time how long it takes to go from low to high.

The HC_SR04.py script does just that. When you have everything wired up, run it as root to see if everything's working:

```
sudo python HC_SR04.py
```

If everything's okay, you'll see it print repeated lines measuring the distance in front of it. Try putting your hand or some other object in front of it and see if the distance changes.

If you don't see repeated printouts, double-check your wiring. It's amazingly easy to find yourself one pin off, or with the Trigger and Echo pins reversed.

Once all the rangefinders are working, we can move on to the real PiDoorbell logic, in *pidoorbell-recognizer-gpio.py*.

PiDoorbell Recognizer

pidoorbell-recognizer-gpio.py is the script that puts everything together.

It needs to be run as root (because it needs access to GPIO). It can take several arguments: `pidoorbell-recognizer-gpio.py -h` will list them all.

For initial testing, run it like this:

```
sudo python pidoorbell-recognizer-gpio.py -i -local
```

-i specifies interactive mode, so you can see verbose messages about what it's doing. -local keeps it from using any web services while you're still getting everything hooked up.

In interactive mode, you should see a steady stream of messages as it reads distances from the rangefinder. Try putting your hand or another object in front of the sensor to trigger it.

By default, the script considers anything from 0 to 30 inches to be within its target range, and an object has to remain there for at least 5 seconds before it reacts. These values are set near the beginning of the script and are easy to change.

Taking photos or video of your visitors

The recognizer can take a photo if you have a camera on your Raspberry Pi. That can be either a USB webcam that works with *fswebcam* and *ffmpeg*, or the Raspberry Pi camera module attached to your Pi.

Ffmpeg and fswebcam are optional software, not installed by default, so they need to be installed:

```
apt-get.install fswebcam ffmpeg v4l-utils
```

The Pi camera module doesn't use the normal Linux video framework, so it needs its own programs: *raspistill* and *raspivid* (which should already be installed as part of package libraspberrypi-bin), or, ideally, a Python module called *picamera* which isn't installed by default:

```
apt-get install python-picamera
```

By default, the recognizer will try to take a still photo using fswebcam if it finds a USB camera, or using the pi camera module if one is installed. With a USB camera, you can shoot a 10-second video instead (video isn't supported yet for the pi camera) by passing `-pic_mode 1` as an argument to *pidoorbell-recognizer-gpio.py*.

Adding Dropbox

(insert material here)

AddingTwilio

(insert material here)

Appendix: References and Thanks

This handout, all the code, slides, and other materials for this workshop are available on GitHub:

https://github.com/codechix/PiDoorbell

This tutorial is run by members of Bay Area CodeChix, a nonprofit organization for promoting women software developers: http://CodeChix.org

A big thank-you to our volunteer T/As, without which we could not have done this tutorial: Deepa Karnad Dhurka, Lyz Krumbach, Serpil Bayraktar and Stuart.

And thanks to the Python Foundation, which provided a grant for the hardware kits as well as lots of help with logistics, and having this great conference in the first place!