Raspberry Pi

Connecting Electronics

Trystan Jones
@crennsmind
www.crennsmind.com

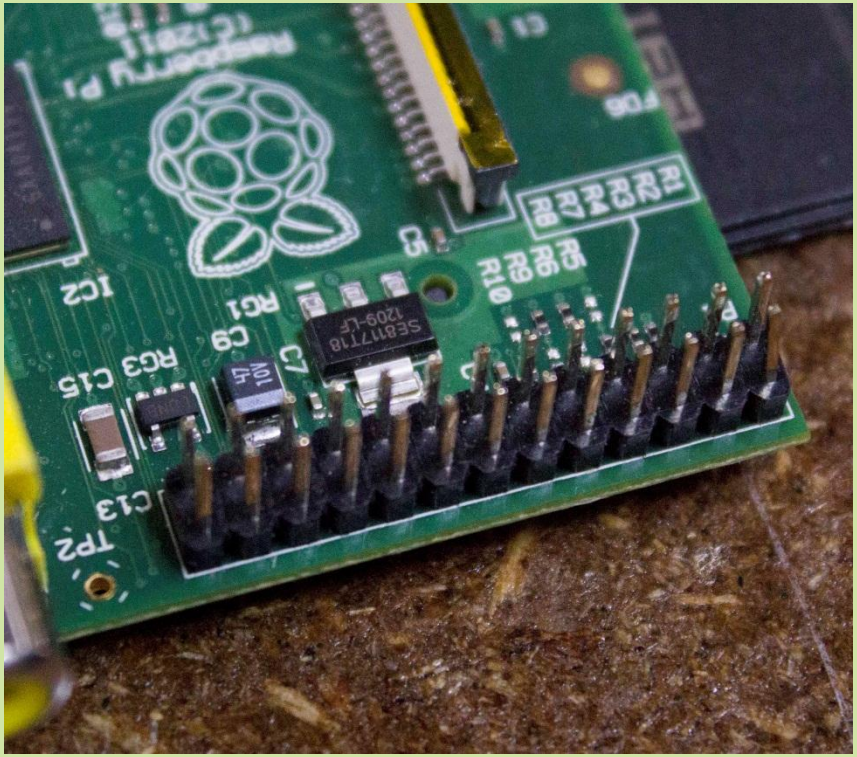
What's being covered?

- A Brief Warning
- Meet the GPIO (General Purpose Input/Output) Port
- Options for interacting with the GPIO
- Using a LED with the GPIO
- Using a port expander to drive LEDs instead
- Using a microcontroller to drive LEDs instead

A Brief Warning

- It is possible to damage your Raspberry Pi:
  - The pins of the Raspberry Pi are 3.3V tolerant, anything higher will damage the pin and potentially cause the RPi to cease working completely
    - Use level convertors if needed or voltage dividers
  - Drawing too much current from the pins can also cause damage to the pins and potentially cause the RPi to cease working
    - Use transistors or external electronics to drive higher current loads
  - Drawing too much current from the voltage regulators may make the RPi unstable and unexpectedly reset
    - Limit the amount of current drawn from the RPi, use external power supplies

Meet the GPIO Port!



- 17 GPIO
- 2 3.3V pins
- 2 5V pins
- 5 GND pins
- Many special functions available on the GPIO
  - SPI
  - I^2C
  - UART
  - PWM

Information from:
http://elinux.org/Rpi_Low-level_peripherals

Options for interacting with the GPIO

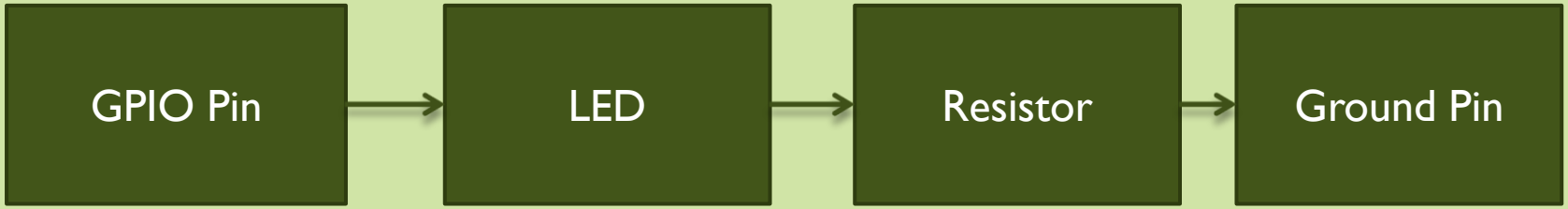
- Hardware
  - Floppy Drive Cables
  - Breakout boards
    - Gertboard
    - PiFace
    - PiPlate
    - PiCrust
    - Cobbler
  - Prototyping Wire
  - Breadboards

Options for interacting with the GPIO

- Software
  - WebIDE
  - Python
  - C/C++ Libraries
  - Scratch
  - Bash Scripts
  - Command Line

Using an LED with the GPIO

- Nice simple example, using the GPIO to light an LED
- Using Python, just some prototyping wire and a breadboard
- Code base used from http://elinux.org/Rpi_Low-level_peripherals



Using an LED with the GPIO

```
import time
import RPi.GPIO as GPIO

# Setting library to using BCM GPIO 00..nn numbers GPIO.setmode(GPIO.BCM)
GPIO.setmode(GPIO.BCM)

# Setting the GPIO to an output
GPIO.setup(18, GPIO.OUT)

# Turning the GPIO On and Off 4 times
for x in range(0, 4):
        GPIO.output(18, True)
        time.sleep(1)
        GPIO.output(18, False)
        time.sleep(1)

# Return the GPIO to being an input
GPIO.setup(18, GPIO.IN)
```

Using a port expander to drive LEDs

- Using an example utilising Adafruit's WebIDE
- The instructions are available here: http://learn.adafruit.com/mcp230xx-gpio-expander-on-the-raspberry-pi/
- Installation instructions for the WebIDE can be found here: http://learn.adafruit.com/webide/installation

Using a microcontroller to drive LEDs

- Bit more complex as we need to communicate with the microcontroller, we'll use SPI
- Need to make a way of getting telling the Raspberry Pi to turn the LED on or OFF, we'll send a byte to the Microcontroller
- Using Mike's C library which supports SPI, available here: http://www.open.com.au/mikem/bcm2835/

Any questions?

I don’t bite, I swear!