Hardware

FUNDAMENTALS

Getting started with Do It Yourself circuits and Raspberry Pi

Hardware?

- Three main types of hardware (in this course)
  - Commercial, off-the-shelf devices/appliances
  - Do It Yourself solutions
  - The central gateway
    - On which runs the environment intelligence
    - Centralized approach for the sake of simplicity

Focus

- In this tutorial we focus on
  - The central gateway
  - Do It Yourself solutions
- And the rest?
  - Will be treated in detail throughout the course

Goals

- Knowledge of the reference platform
- Getting started to develop ad-hoc solutions when needed

The Gateway

- Hosts the environment intelligence
- Must have enough computational power
- Should easily interface existing automation networks
- Should easily interface appliances and smart devices (e.g., TVs, Monitors, etc.)
- Should exploit Internet connectivity
  - When available
- Should support integration of ad-hoc solution

Candidates

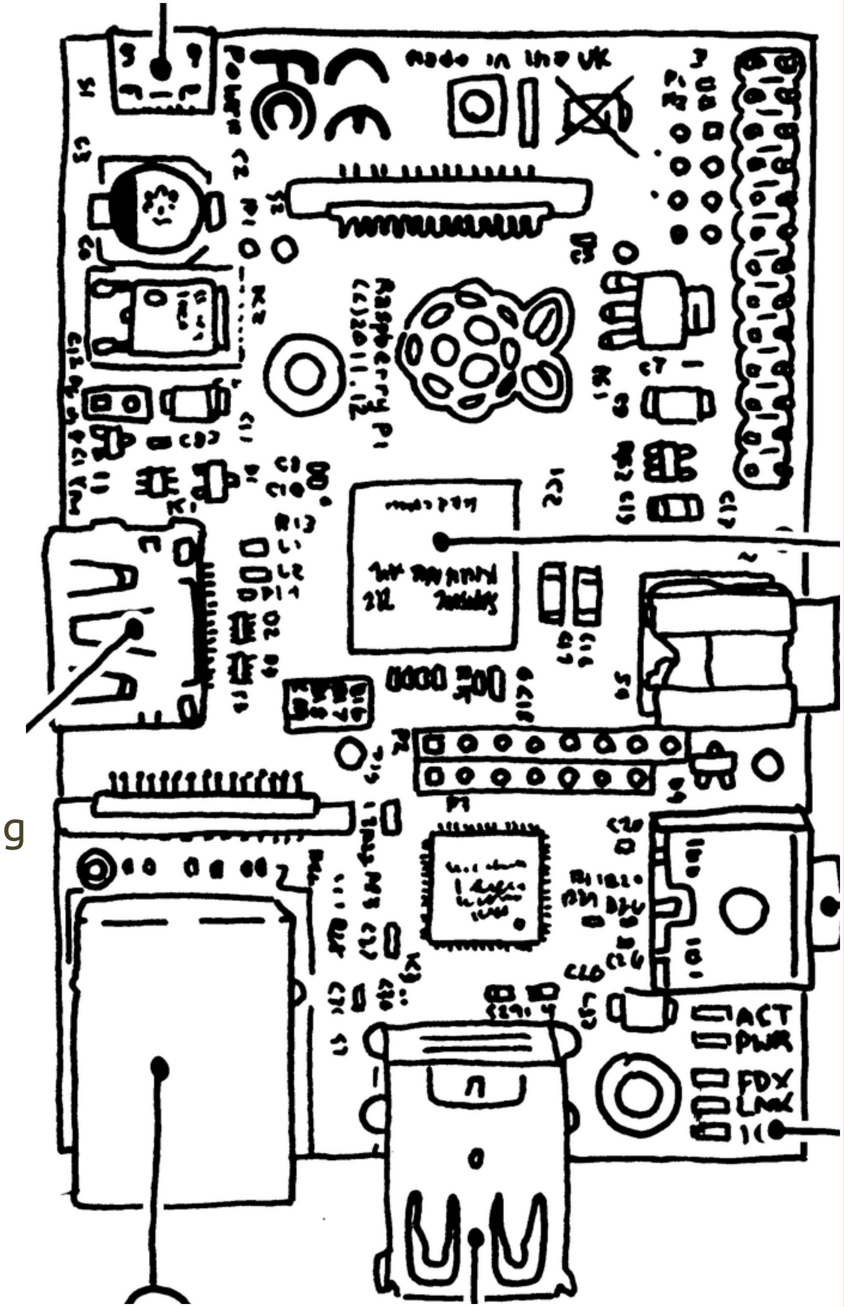
- Beagle black
  - Medium cost
  - Ready for DIY
  - Less easy to interface commercial devices
  - High computational power
  - Good connectivity
- Raspberry Pi
  - Low cost
  - Ready fo DIY
  - Easy to interface commercial devices
  - Good computational power
  - Good connectivity
- Arduino
  - Cheap
  - Ready for DIY
  - Difficult to interface commercial devices
  - Low computational power
  - Low connectivity

For this course the Raspberry Pi represents an optimal trade-off

Raspberry Pi

A SHORT INTRODUCTION

A short introduction on Raspberry Pi including hardware, software and DIY capabilities



Components

- Processor
  - The same you would have found in the iPhone 3G and the Kindle 2
  - ARM11, 700MHz, 32bit (quad-core on Pi-2)
  - 512MByte of RAM (1GByte on Pi2)
- SD
  - everything is stored on an SD Card
- USB
  - 2 USB 2.0 ports (4 on Pi-2)
  - Up to 500mA
  - Not advisable to use for high power loads
    - Phone cell chargers
    - Portable HDD
- Ethernet
  - Standard 10/100 (Model B only)
  - WiFi connectivity via a USB dongle

Components (Cont'd)

- HDMI
  - 14 different resolutions
  - Composite out available (NTSC / PAL)
  - Can be converted to other formats
- Status LEDs
  - Visual feedback on the Pi status
- Analog Audio output
  - Designed to drive high-impedence loads (e.g., active speakers)
- Power input
  - Micro USB connector
  - Typical rating 5V, 1200mA
  - Could work from a PC USB (but exceeds USB max current...)

Components (cont'd)

- General Purpose Input and Output (GPIO)
  - To read buttons and switches
  - To control actuators
  - Etc.
- Display Serial Interface (DSI)
  - To communicate with a LCD or OLED display
- Camera Serial Interface (CSI)
  - To directly connect a camera module

Ratings (Pi model B)

- Power supply
  - 5V
  - At least 700mA
- SD card
  - 8GByte
  - Class 6 or higher for reasonable performances
    - We will use class 10 cards
- GPIO rating
  - Max sink current: 16mA
  - Max source current: tunable from 2 to 16mA
    - Lower is better
  - Total drawable current on 3.3V supply
    - 40mA
  - Total drawable current on 5V supply
    - Around 500mA
    - Can be increased by increasing the power supply ratings

Operating system

- Raspberry Pi runs Linux
- Several supported distributions
  - Raspbian
    - The “officially recommended” one
  - Occidentalis
    - Developed by Adafruit to support electronic development
  - Arch Linux
  - PiDora
    - A fedora port for Raspberry Pi
  - Raspbmc
    - XBMC based distribution
    - To use the Pi as a media center
  - OpenElec
    - Similar to Raspbmc, based on Syslinux

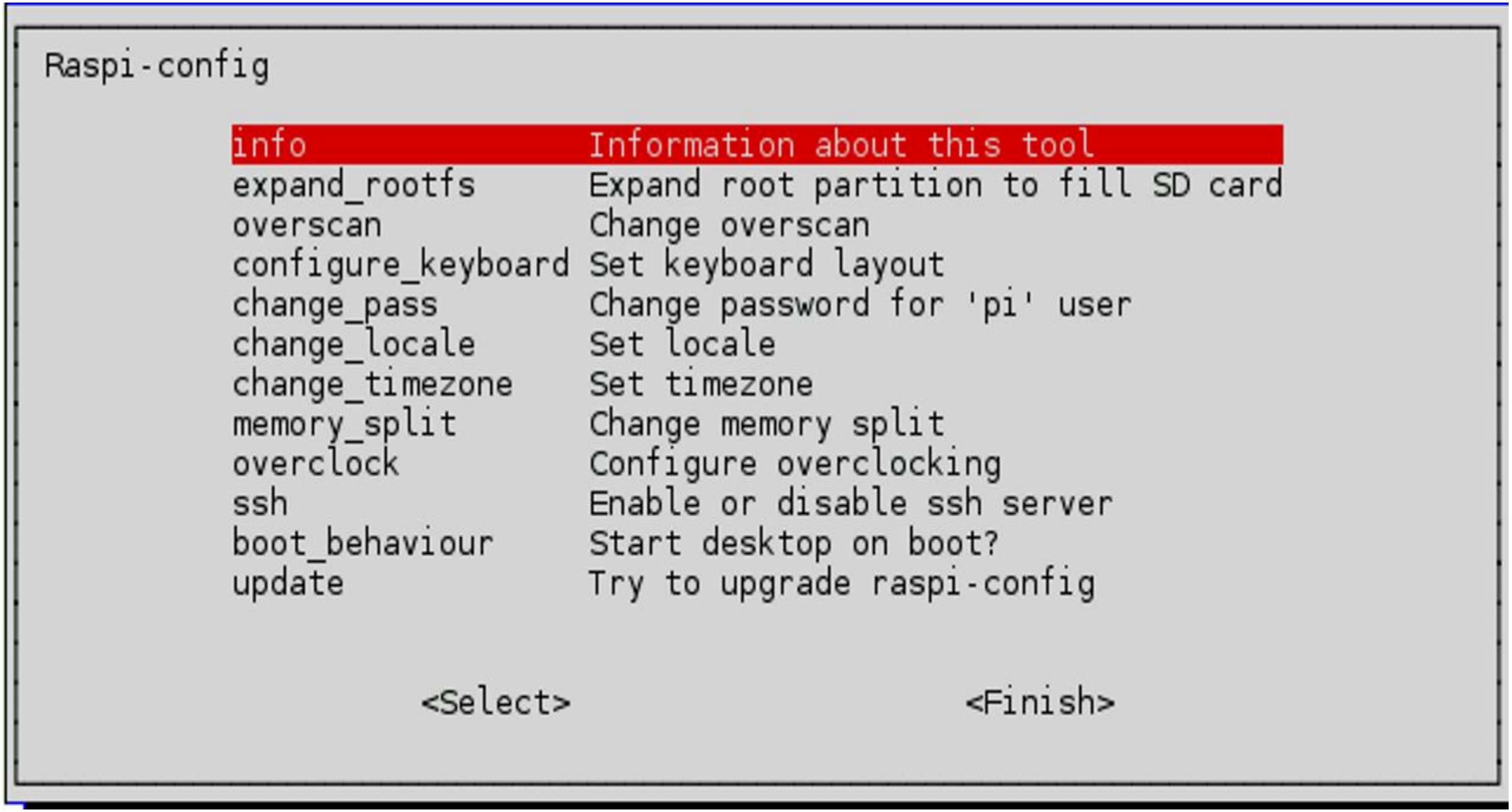
Installation

- We adopt the «official» Raspbian distribution
- Raspbian is provided as raw image
  - Bit-for-bit representation of how the data shall be written on disk
  - Cannot be simply copied to the SD card
  - A disk imaging utility must be used
    - dd (Linux/Mac)
    - Win32DiskImager (Windows)

First boot

- Plug the SD card into the socket.
- Plug in a USB keyboard and mouse.
- Plug the HDMI output into your TV or monitor.
- Plug in the power supply.

Initial configuration



First Checks

- Login
  - User: pi
  - Password: raspberry (must possibly be changed)
- Board revision
  - cat /proc/cpuinfo
- Python version
  - python - -version
- Python RPi tools
  - easy_install Rpi.GPIO
    - must be connected to the Internet

Do It Yourself

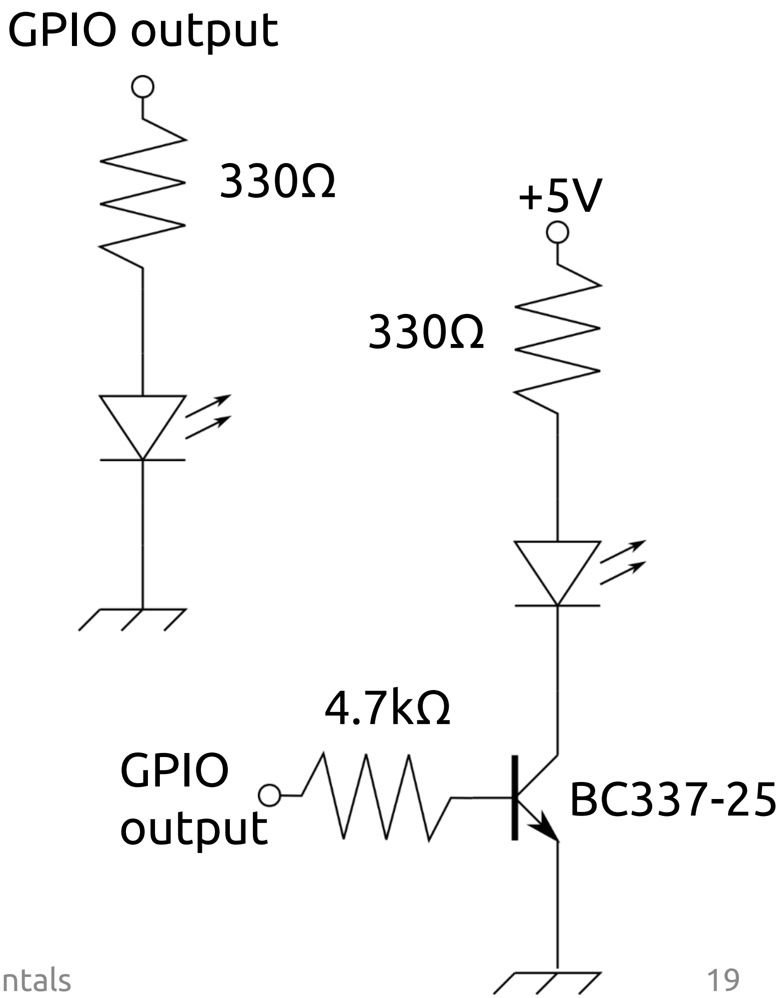
DEMO PROJECTS

Project 1: LED control

- Goal
  - Light-up a red LED using one GPIO port
  - Control the LED switching from Python
- Required Components
  - The Raspberry Pi
  - A red LED
  - A NPN transistor (BC337-25 in our example)
  - A couple of resistors

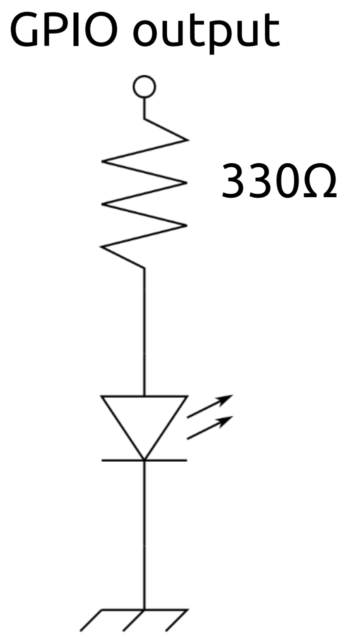
LED control - schematics

- Can be directly driven by the GPIO output,
  - safer to use as control for a power switch (a transistor)
- Care must be taken to not exceed the maximum ratings
  - 40mA on all GPIO outputs
  - 8mA on a single GPIO (can be tuned)
  - Better if lower than 1mA



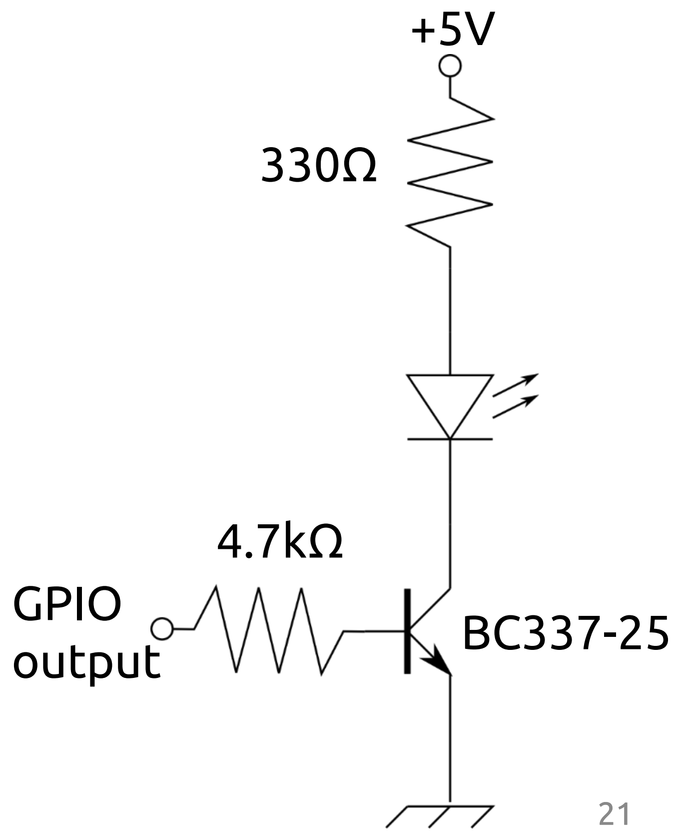
LED control – design

- Direct control
  - The LED causes a voltage drop from 1.2V to around 2V
  - I_{GPIO} = 3.3V-2V/330Ω = 1.3V/330Ω = 3.94mA
  - I_{GPIO} = 3.3V-1.2V/330Ω = 2.1V/330Ω = 6,34mA



LED control – design (cont'd)

- Power switch
  - The LED causes a voltage drop from 1.2V to around 2V
  - I_{GPIO} = 5V-2V/330Ω = 3V/330Ω = 9mA
- Transitor in saturation
  - h_{fe}min = 160
  - I_B=3.3V-0.6V/4.7kΩ= 574 µA



LED control - Python

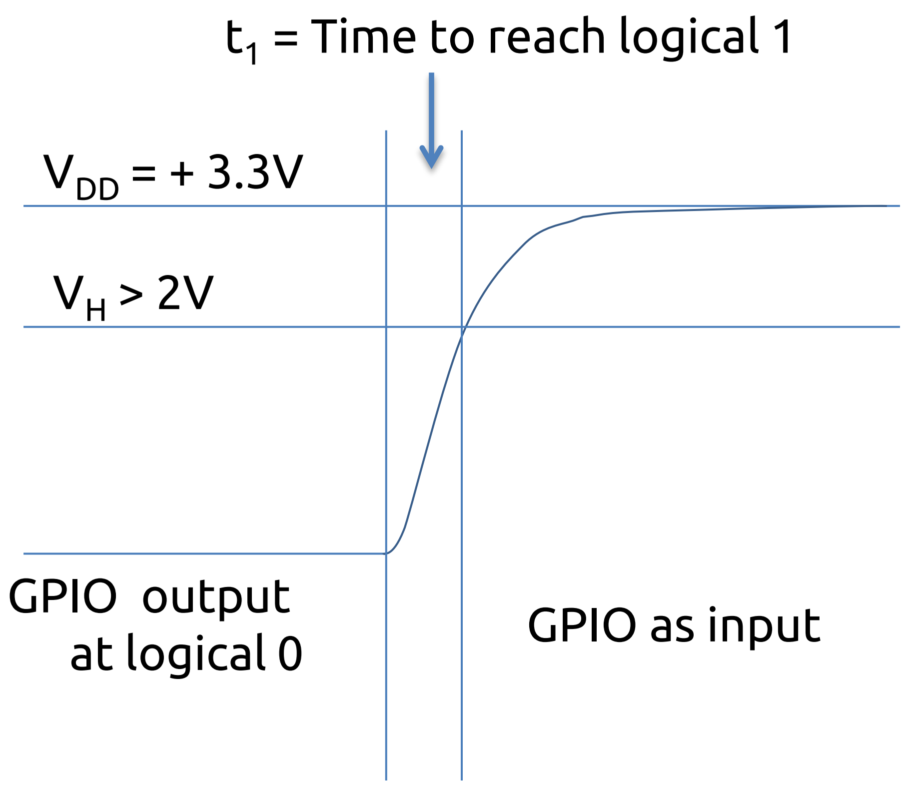
```
import RPi.GPIO as GPIO
import time

#set-up pin numbering
GPIO.setmode(GPIO.BOARD)
#set-up pin function
GPIO.setup(15,GPIO.OUT)

#iterate 10 times
for i in range(10):
        GPIO.output(15,1) #set the output at 1, LED on
        time.sleep(1)   # keep it for 1 second
        GPIO.output(15,0)   # set the output at 0, LED off
        time.sleep(1) #keep it for 1 second
```

Turn alternatively on and off the LED for 10 times

Project 2: light sensor

- Goal
  - Design a cheap light sensor
  - Read the «light level» by using Python
- Components
  - The Raspberry Pi
  - A photo-resistor
  - 2 fixed resistors
  - A capacitor
- Constraints
  - No analog input available

Light sensor – basic principles



t1 depends on the circuit time constant (RC)

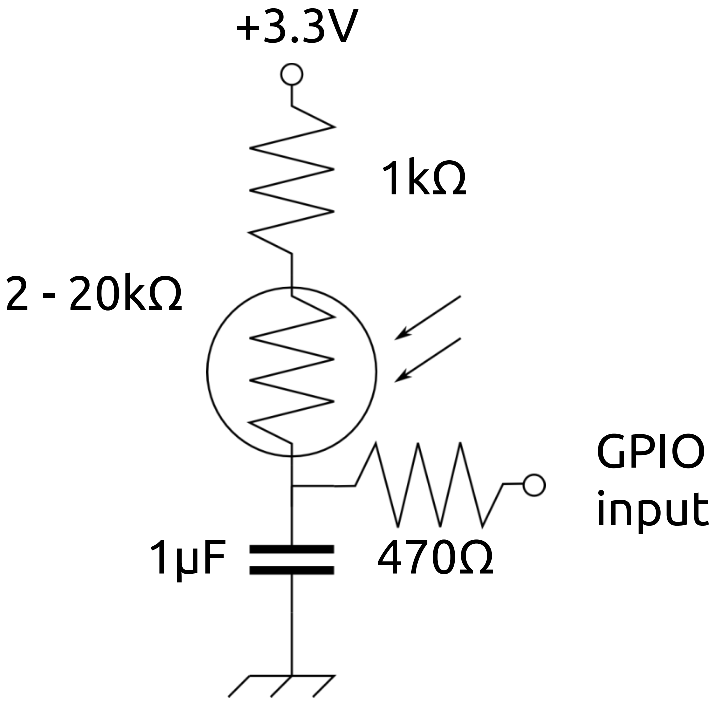
By varying RC t_1 increases or decreases

If the R value depends on the amount of incident light, then t_1 depends on the light intensity

Photoresistors: reduce their actual resistance when illuminated

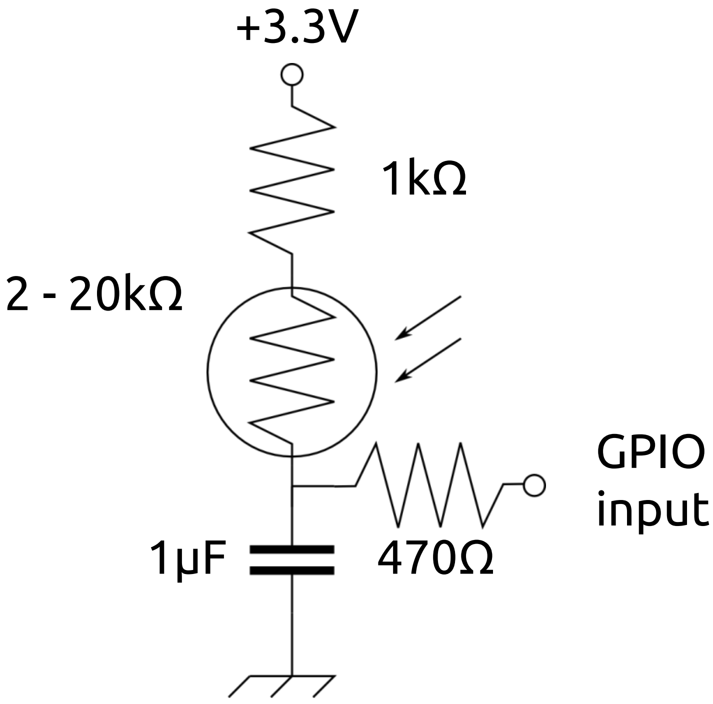
Light sensor - schematics

- Logical 0 if
  - $V_{GPIO}<0.7$ V
- Logical 1 if
  - $V_{GPIO}>2$ V
- Time to reach Logical 1 is roughly given by RC (time to reach 63% of the final voltage)
- 3.3*0.63 = 2,079 V



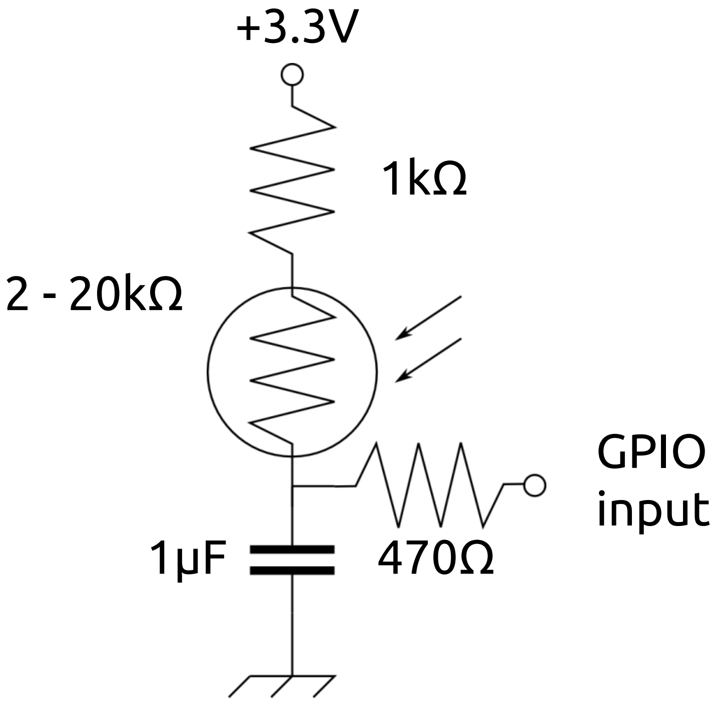
Light sensor - design

- Time ranges
  - RC_{min} = 2kΩ * 1µF =2ms
  - RC_{max} = 21kΩ * 1µF =21ms
- May be tuned by tuning the fixed resistor



Light sensor - algorithm

- Algorithm:
  - Set GPIO as output
  - Write 0
  - Set GPIO as input
  - Count time to get 1 in input



Light-sensing example

```
import RPi.GPIO as GPIO, time, os
DEBUG = 1
GPIO.setmode(GPIO.BOARD)

def RCtime (RCpin):
        reading = 0
        GPIO.setup(RCpin, GPIO.OUT)
        GPIO.output(RCpin, GPIO.LOW)
        time.sleep(0.2)
        GPIO.setup(RCpin, GPIO.IN)
        # This takes about 1 millisecond per loop cycle
        start = time.time()
        while (GPIO.input(RCpin) == GPIO.LOW):
                reading += 1
        print (time.time()-start)*1000.0, "ms"
        return reading

GPIO.setup(15,GPIO.OUT)
while True:
        rc = RCtime(13)
        #print rc
        if rc < 1000:
                GPIO.output(15,GPIO.LOW)
        else:
                GPIO.output(15,GPIO.HIGH)
```

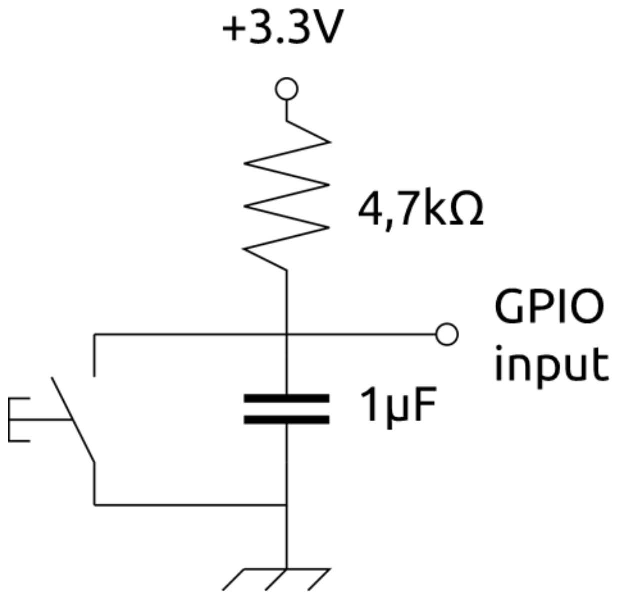
Turn on the LED if the lighting level drops over a given threshold

Project 3: push button

- Goal
  - Light-up a red LED using one GPIO port when a button is pressed
  - Detect button pressing from Python
- Required Components
  - The Raspberry Pi
  - A red LED
  - A NPN transistor (BC337-25 in our example)
  - A couple of resistors
  - A capacitor
  - A push-button

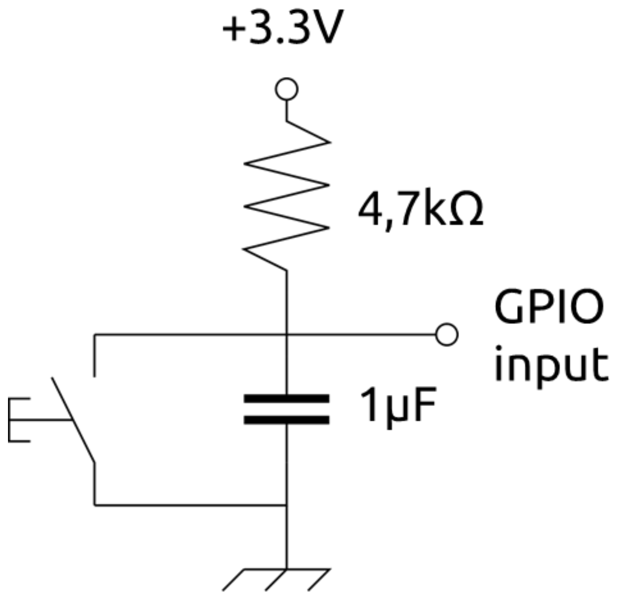
Push-button – basic principles

- De-bounce needed
  - To avoid capturing button bounces
  - Based on the RC circuit
  - TTL 3V has V_{on} at 2V
  - In first instance we can assume that the time required to reach such level is equal to RC (time to reach 63% of V_{DD})



Push-button – design

- RC
  - $4.7k\Omega * 1\mu F = 4.7ms$
- The button can bounce for up to 4.7ms
- Sufficient in typical applications



Button-sensing example

```
import RPi.GPIO as GPIO, time, os

DEBUG = 1

#set-up pin numbering
GPIO.setmode(GPIO.BOARD)

def button_to_led (RCpin):
        # set-up pins
        GPIO.setup(RCpin, GPIO.IN)
        GPIO.setup(15,GPIO.OUT)
        # This takes about 1 millisecond per loop cycle
        while (True):
                if(GPIO.input(RCpin) == False):
                        GPIO.output(15,GPIO.HIGH)
                else:
                        GPIO.output(15,GPIO.LOW)
        return

button_to_led(11)
```

Turn on the LED when the button is pressed (by design the GPIO input will be at 0)

Questions?

01PRD Ambient Intelligence: Technology and Design

Dario Bonino
bonino@ismb.it