Creative Embedded Devices: Class 2

A Pi to the Face

Today:

- Intro to Pi, Hardware and Software Overview
- Pi Vs. Phone
- Talk through HW1 (and associated tasks (2))
- Demo *possible* application software: Processing
- Activity: Form Teams and Strategize HW1

Today:

Tuesdays in Leeds (here)

Logistics for this space: three different types of chairs, three different dollies

Thursdays in CCAM 110

Always bring your laptop, Pis, and other hardware

What is the Raspberry Pi?

A tiny Linux Computer

Moderate specs (more later)

40-pins GPIO (more later)

What does the Raspberry Pi do well?

Tiny (embeddable)

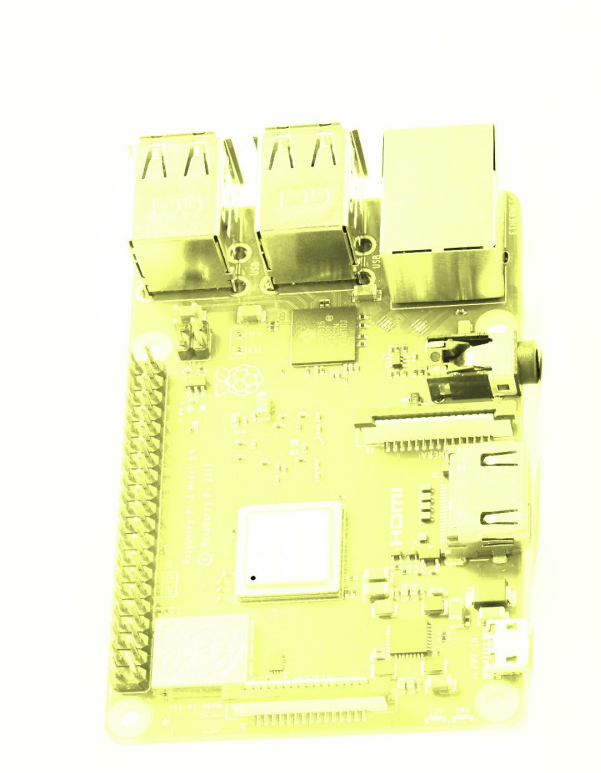
Runs "headless"

Raspbian image = ready for many use cases

Battery powered

Robust hardware (good design)

RetroArcade + 2 SNES controller clones = fun for years



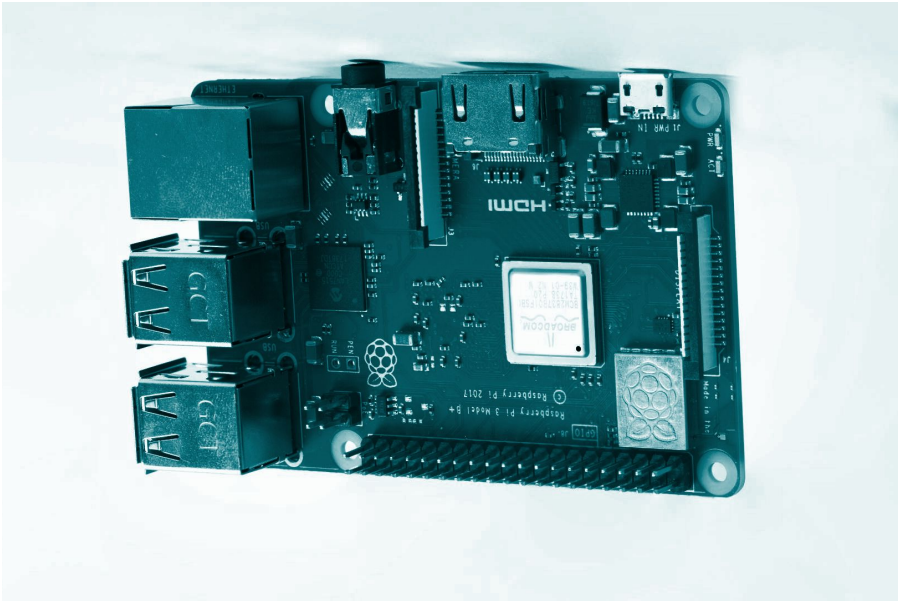
What does the Raspberry Pi not do well?

General purpose computing (slow)

Prone to corruption (software)

No ADCs: requires additional hardware for analog sensor input

High power draw



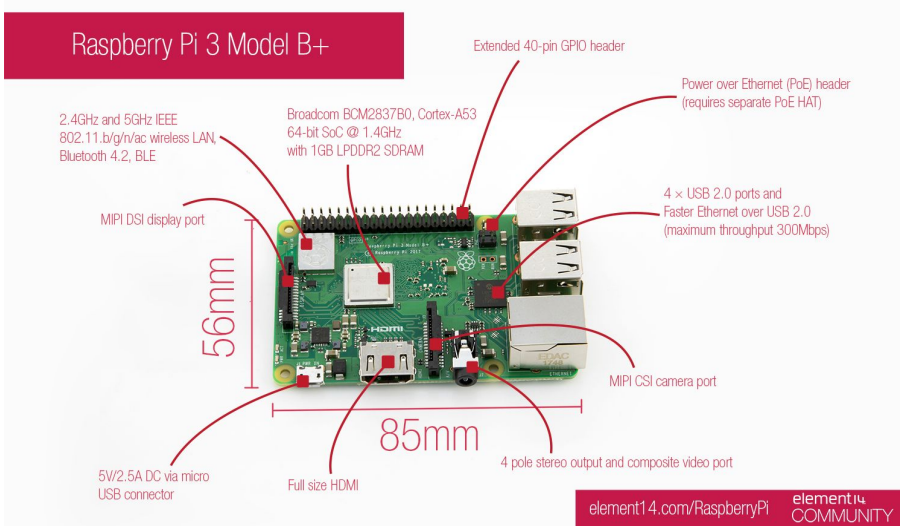
There are other microcomputers out there… (and other Pi versions)

Technical Overview

Specs

https://www.raspberrypi.org/magpi/raspberry-pi-3bplus-specs-benchmarks/

SoC: Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
GPU: Broadcom Videocore-IV
RAM: 1GB LPDDR2 SDRAM
Networking: Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi
Bluetooth: Bluetooth 4.2, Bluetooth Low Energy (BLE)
Storage: Micro-SD
GPIO: 40-pin GPIO header, populated
Ports: HDMI, 3.5mm analogue audio-video jack, 4x USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
Dimensions: 82mm x 56mm x 19.5mm, 50g



A quick note on data speeds

WLAN and BT:	**Tx bandwidth (Mb/s)**	**Rx bandwidth (Mb/s)**
Raspberry Pi 3B	35.7	35.6
Raspberry Pi 3B+ (2.4GHz)	46.7	46.3
Raspberry Pi 3B+ (5GHz)	102	102

Gigabit ethernet limited by USB 2.0 connection to application processor.

	Tx bandwidth (Mb/s)	Rx bandwidth (Mb/s)
Raspberry Pi 3B+	315	315

More comparisons in MagPi pdf

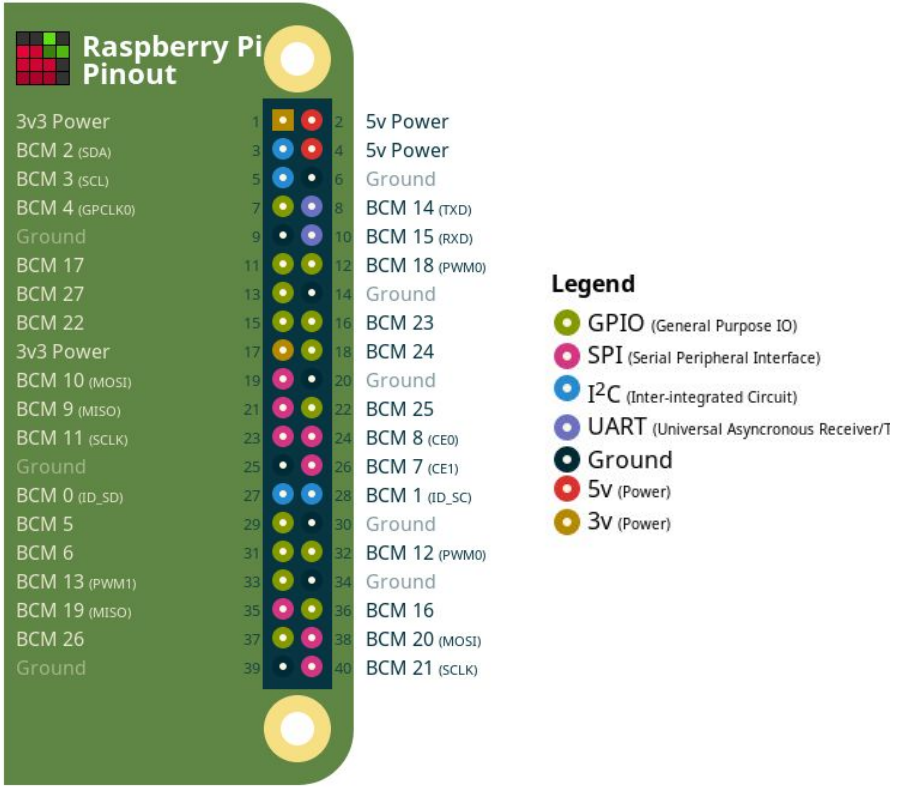
GPIO Pinout

https://pinout.xyz/#

The above has an amazing interactive pinout description with configurations of additional hardware as well.

https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md

GPIO: General Purpose I/O pins are those that can be programmed to multiple input/output configurations. The pins are 3.3v hi and 0v low. These are *most often* used with "alternative functions" -- communications hardware, GPIO expansion hardware, etc. Pinouts for these advanced configurations can be found at pinout.xyz linked to above.



Power

https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md

The amount of power required of the RPi varies based on application. In other words, it depends on the type and number of connected circuits/peripherals.

Low: the pi can be powered from a laptop USB cable if no additional peripherals are connected. (3.3v, 700mA)

High: Use of motor/actuators with the Pi invariably require additional hardware and a separate power source.

Changing Low-level Hardware Settings

The Raspberry Pi has no equivalent to a BIOS where low-level machine options can be set.

From: https://elinux.org/RPi_config.txt

"As the Raspberry Pi doesn't have a conventional BIOS, the various system configuration parameters that would normally be kept and set using the BIOS are now stored in a text file named "config.txt". The Raspberry Pi config.txt file is read by the GPU before the ARM core is initialized."

Thus the config.txt file is where you go to set low-level options for RPi operation. See here for instructions: https://elinux.org/R-Pi_configuration_file

Boot-time(ish) Execution

One basic functionality of embedded systems is their ability to operate autonomously and self-correct if operations are disrupted.

There are a number of ways to fire up processes and execute scripts when the RPi is powered on. The method you choose will have to do with 1) ease of use, 2) where in the boot sequence you need your programs/scripts to execute and 3) the nature of the programs/scripts themselves.

- Crontab
- rc.local
- .bashrc
- Systemd
- Init.d

Disaster Planning

Your Raspbian OS *will* become corrupted and you will have to re-flash it, losing all contents of the card. This is a certainty.

- Use git to store all software, resources and configurations

- Consider writing a script to automate the install/setup process and include it in your repo

- Alternatively, keep a backup of your working system (dd)

Think ahead when it comes to embedded development (god help you…) and/or troubleshooting

- Make sure you always have access (network, eth cable, peripherals)

Case Studies: Pi vs Smartphone vs Combo

Break into groups of 4 and list pros and cons of each device, what challenges will you need to overcome

#1 : A musical “hot potato” that is tossed around the room and sonfies the gyroscope

#2 : A twitch-plays-pinball machine

#3 : An installation in a forest that allows users to remotely generate tree-falling sounds

#4 : A device that waters your plants based on the weather outside, or on demand

#5 : The OMIPOD

Activity: HW 1 Strategy Session

Task 1: Due Sep 10
Task 2: Due Sep 17


YALE CENTER FOR ENGINEERING
INNOVATION AND DESIGN


EXIT
WINCHESTER HALL
EXIT


BECTON ENGINEERING AND APPLIED SCIENCE CENTER
EXIT

Task 1:

Schedule Sessions in Becton Cafe

Strategize implementation

Make one github repo and invite all members, as well as

@santolucito and @scacinto

Teams

Team 1: Alexi, Grace, Sabrina, Xavier

Team 2: Daniel, Lukas, Felicia, Sam

Team 3: Carlos, Yuki, Joe, Julia

Team 4: Bryce, Cynthia, Sarim, Tantan

Team 5: Jordan, Justin, Varsha, Itai