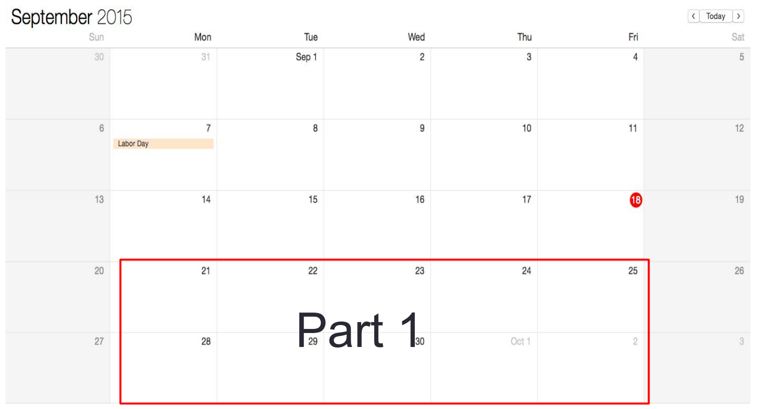
CS123 - Recap

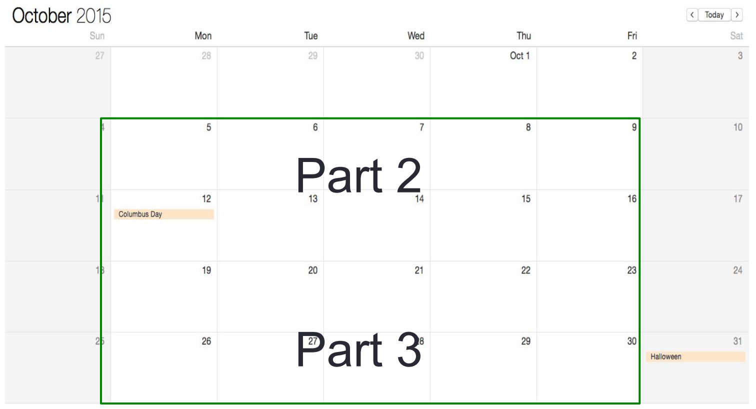
Programming Your Personal Robot

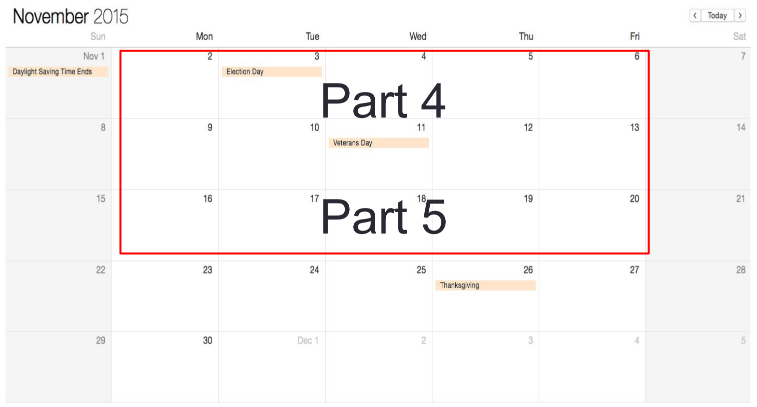
Kyong-Sok "KC" Chang, David Zhu

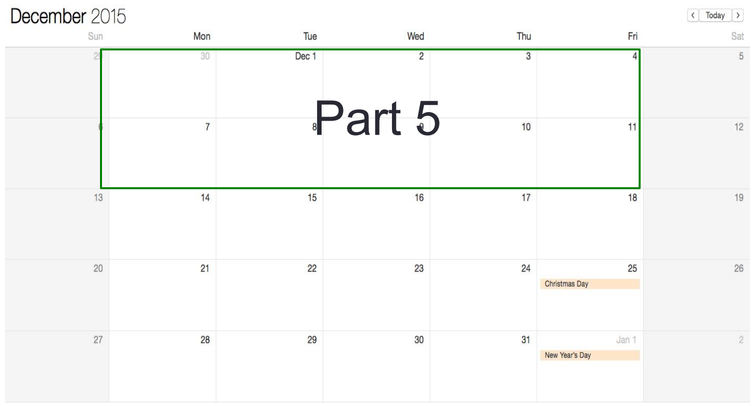
Fall 2015-16

Calendar









KC
Teaching

David
Teaching

Syllabus

- Part 1 - Communicating with robot (2 weeks)
  - BLE communication and robot API
- Part 2 - Event Driven Behavior (2 weeks)
  - Finite State Machine (Behavior Tree)
- Part 3 - Reasoning with Uncertainty (2 weeks)
  - Dealing with noisy data, uncertainty in sensing and control
- Part 4 - Extending the robot (1 weeks)
  - I/O extensions: digital, analog, servo, pwm, etc
- Part 5 – Putting it together (including UI/UX) (3 weeks)
  - Design and implement of final (group) project
  - Encourage you to go “above and beyond”

Logistics

- Getting new PSD Scanner
  - Update Hamster firmware
    - Over-The-Air Device Firmware Update (DFU)
    - nRF Toolbox App
  - Install the hardware
  - Sign-up sheet
- TA sessions (office hours): this week
  - Location: Gates B21 (Th: Huang basement)
  - Time: M:2~4pm, Tu:2~4pm, W:12:30-2:30pm, Th:2~4pm
- Lab reserved for CS123: this week
  - MTuW: 12~6pm @ Gates B21
- My office hours (KC)
  - Tues & Thurs: 1-2pm @ Gates B21(Tu), Huang Basement(Th)

Robotics Company: HW vs. SW

- Google
- Toyota
- Disney
- Softbank
- Honda
- Amazon
- Apple
- Foxconn
- Alibaba
- Tesla
- iRobot
- reThink
- SKT, LG, Samsung, Naver

- ABB
- Fanuc
- Yaskawa
- Adept
- Denso
- Kawasaki
- Kuka
- Mitsubishi
- Schunk
- Staubli
- Yamaha

Outline

- Logistics
- Future robots: Robotics Company - HW vs. SW
- Recap: Part 1~4 (more on Part 1 and 4)
- Part 5: Putting it together (Navigation)
    - Robot Programming
        - Modeling
        - Localization
        - Planning
        - Execution
        - UI / UX
- Assignment #4
- Final project

Lec#01: Introduction

- An introductory class for students who have no (or very limited) experience of programming robots(physical devices) but are interested in learning more.
- A “sampler” (overview): we will cover a range of fundamental topics of robot programming, but we will not be able to go into any specific topic in depth.
  - Communication, Behavior, Uncertainty, Extension, Team
- It’s a good preparatory class for “Experimental Robotics”
- A very hands-on class (learning by doing)
-
- There will be 4 individual projects and 1 team (final) project
  - Project #1 (Communication) : 2 weeks - 20%
  - Project #2 (Finite State Machine) : 2 weeks - 20%
  - Project #3 (Uncertainty) : 2 weeks - 20%
  - Project #4 (Robot Extension) : 1 week - 10%
  - Project #5 (Final, Group project) : 3 weeks - 30%
    - Design your (team) project (need to get approval)

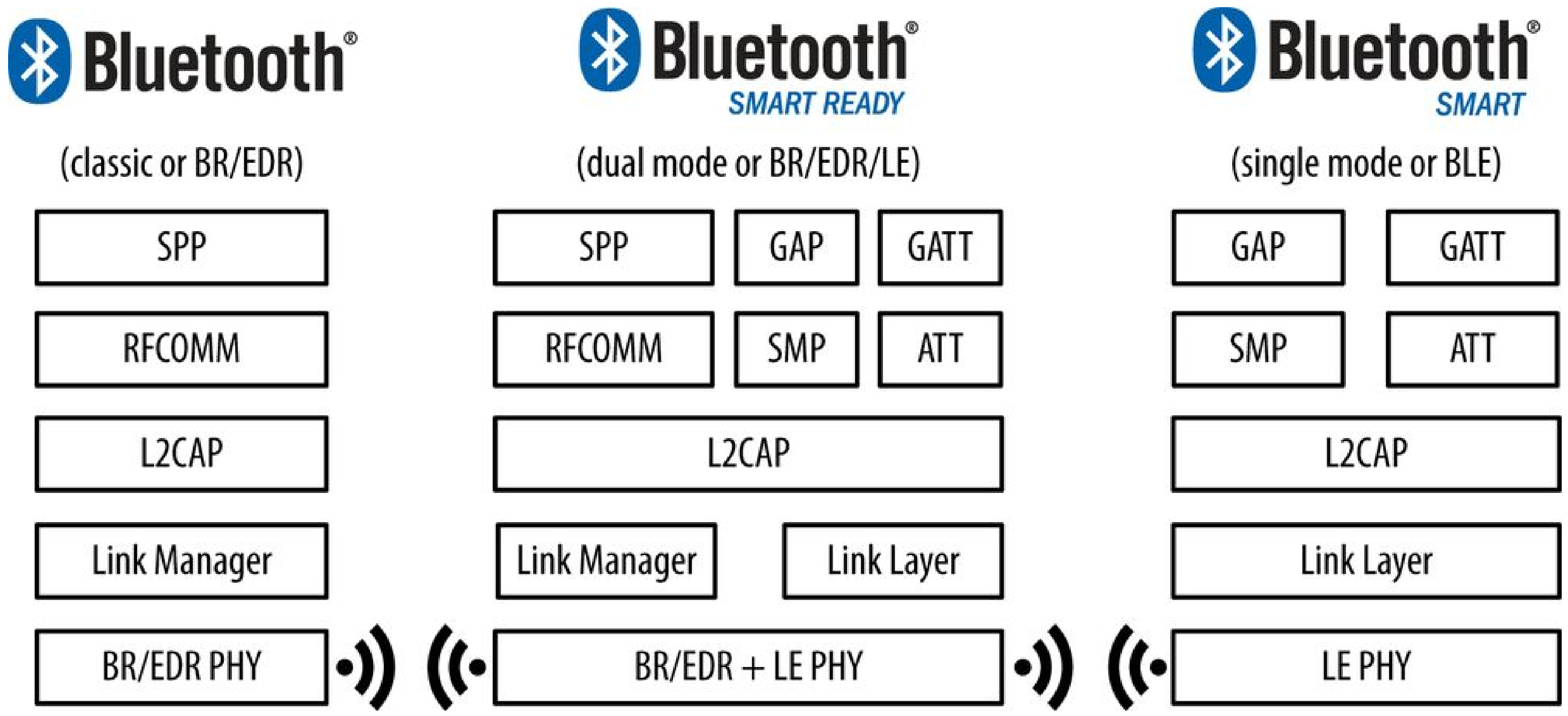
Objectives

- Expose to the challenges of robot programming
  - Gain a better understanding of the difficulty of programming in the real (physical) world
  - Appreciate the challenges of programming in the real worlds
- Learn basic concepts and techniques
  - Modeling the robot
  - Mapping between the Real (physical) world and Virtual world
  - Localization & Plan Execution
- "Opened" problems
  - No 100% guaranteed solution
  - You can always do better
- "Not well defined" problems
  - Further constraining and decompose the problem

Lec#02: Communication

- BLE: Hamster test
- DRC: communication
- Robot communication protocol
- IoT communication protocol
- Bluetooth History / versions
- BLE Specifications
- BLE Protocol
- GAP: Generic Access Profile
- GATT: Generic Attribute Profile
- Assignment#1-1

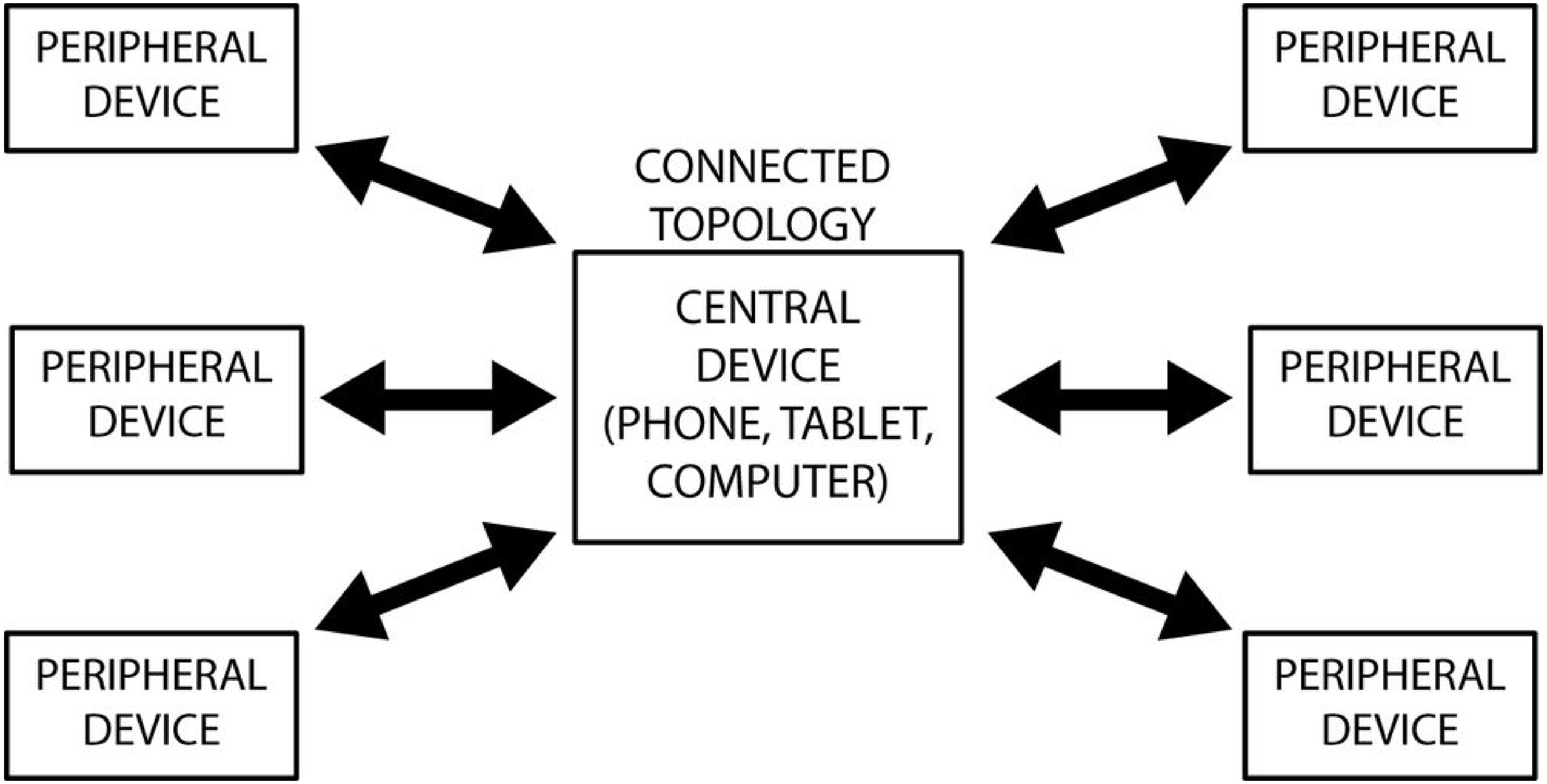
Bluetooth protocols



Configurations between Bluetooth versions and device types

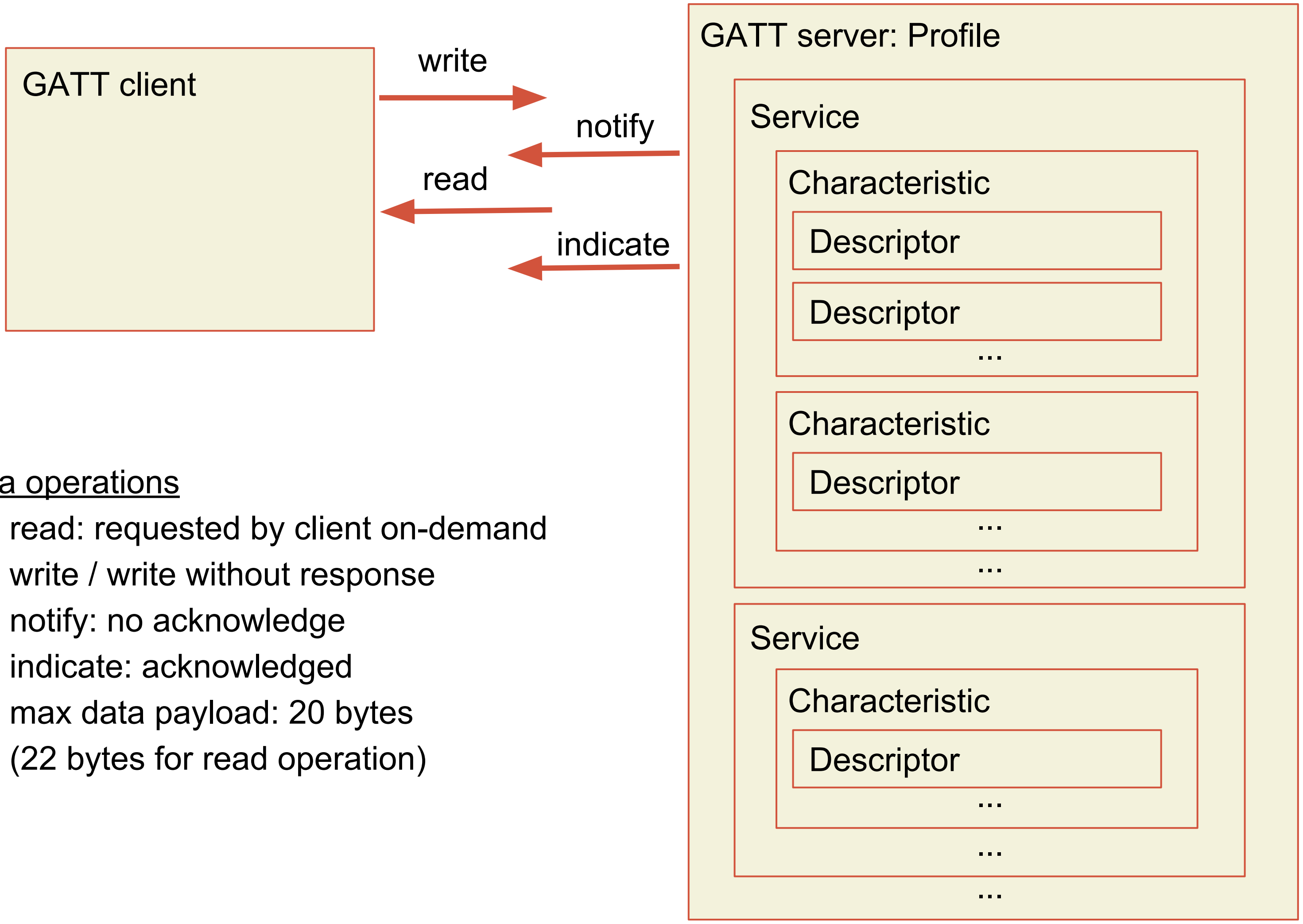
Ref. "Getting started with Bluetooth Low Energy" by Townsend, Davidson & Akiba, O'Reilly

Role: Central - Peripheral



Ref. “Getting started with Bluetooth Low Energy” by Townsend, Davidson & Akiba, O’Reilly

GATT: Data Transfer Methods



4 data operations

- read: requested by client on-demand
- write / write without response
- notify: no acknowledge
- indicate: acknowledged
- max data payload: 20 bytes
  (22 bytes for read operation)

Lec#03: BLE & API

- BLE: Review - Core Bluetooth
- Delegation vs Protocol
- Assignment#1-1
- API: Definition
- Interface vs Implementation
- Naming Convention
- Robot API: Accessor vs. Mutator
- Bytes and Bits
- Bitwise operators and masks
- Two’s complement
- Assignment#1-2

I/O mode: Hamster

Sensors Service Packet format definition

	Details	Value from Robot	User converted value
0	**Version / Topology**	0 ~ 255	0 ~ 255
1	**Network ID**	0 ~ 255	0 ~ 255
2	**Command / Security**	0 ~ 255	0 ~ 255
3	**Signal Strength**	-128 ~ 0	-128 ~ 0 dBm
4	**Left Proximity**	0 ~ 255	0 ~ 255
5	**Right Proximity**	0 ~ 255	0 ~ 255
6	**Left Floor**	0 ~ 255	0 ~ 255
7	**Right Floor**	0 ~ 255	0 ~ 255
8	**Acc X High**	-32768 ~ 32767	-32768 ~ 32767
9	**Acc X Low**		
10	**Acc Y High**	-32768 ~ 32767	-32768 ~ 32767
11	**Acc Y Low**		
12	**Acc Z High**	-32768 ~ 32767	-32768 ~ 32767
13	**Acc Z Low**		
14	**Flag**		
15	**Light High or Temperature**	0 ~ 65535 -128 ~ 127	0 ~ 65535 Lux -40 ~ 88 ºC
16	**Light Low or Battery**	0 ~ 255	0 ~ 100 %
17	**Input A**	0~255	0 ~ 255 (0 ~ 3.3 V)
18	**Input B**		
19	**Line Tracer State**	0 ~ 255	0 ~ 255

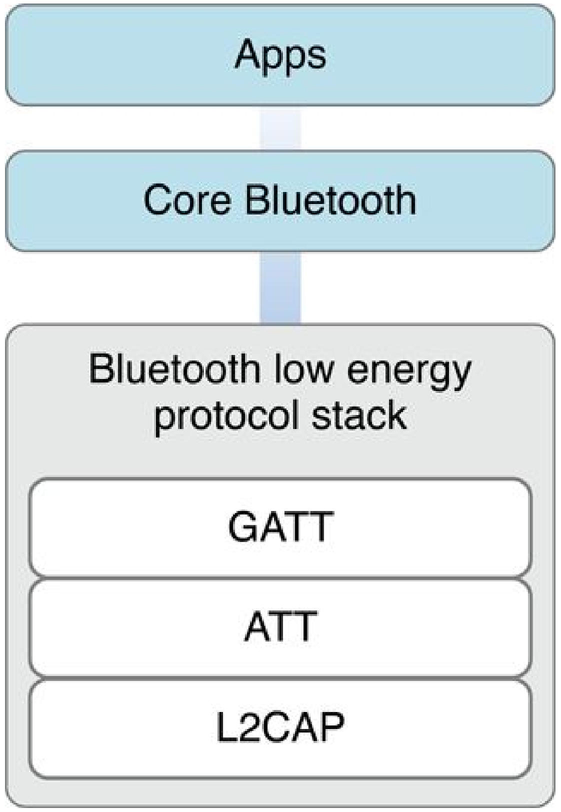
Effector Service Packet format Definition

	Data	Value to Robot	User input value
0	**Version / Topology**	0 ~ 255	0 ~ 255
1	**Network ID**	0 ~ 255	0 ~ 255
2	**Command / Security**	0 ~ 255	0 ~ 255
3	**Left Wheel**	-100 ~ +100 (+fwd, -bwd)	-100 ~ 100 %
4	**Right Wheel**		
5	**Left LED**	0 (off) ~ 7	0 (off) ~ 7
6	**Right LED**		
7	**Buzzer High**	0(off) 1 ~ 16777215	0(off) 1.00 Hz ~ 167.77215 KHz,
8	**Buzzer Middle**		
9	**Buzzer Low**		
10	**Musical Note**	1~88(piano key) 0(off)	1~88 0(off)
11	**Line Tracer Mode/Speed**	0x11 ~ 0x6A 0x0?(off)	0x11 ~ 0x6A 0x0?(off)
12	**Proximity IR Current**	0 ~ 7 (default 2)	0 ~ 7 (default 2)
13	**G-Range, Bandwidth**	0 ~ 3 (default 0), 0 ~ 8 (default 3)	0 ~ 3 (default 0), 0 ~ 8 (default 3)
14	**IO Mode(A, B)**	0 ~ 127	0 ~ 127
15	**Output A**	0 ~ 255	0 ~ 255
16	**Output B**		
17	**Wheel Balance**	-128 ~ 127	-128 ~ 127
18	**Input Pull**	0~16	
19			

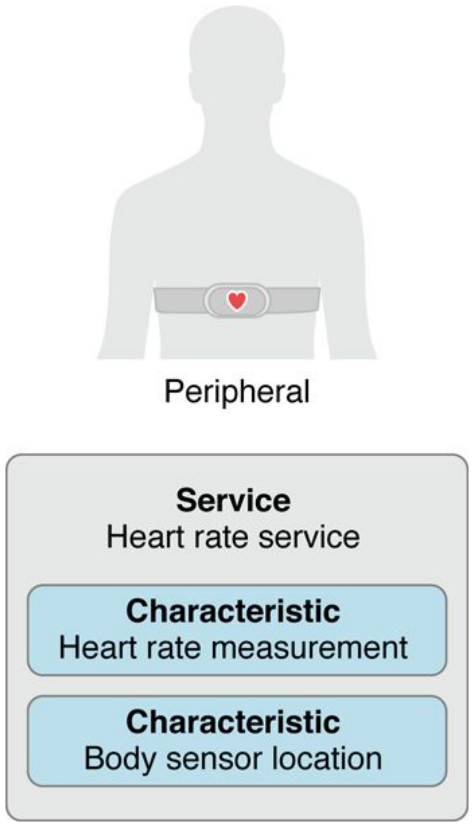
Ref. Kre8 Technology, Inc.

BLE: Core Bluetooth by Apple

Overview

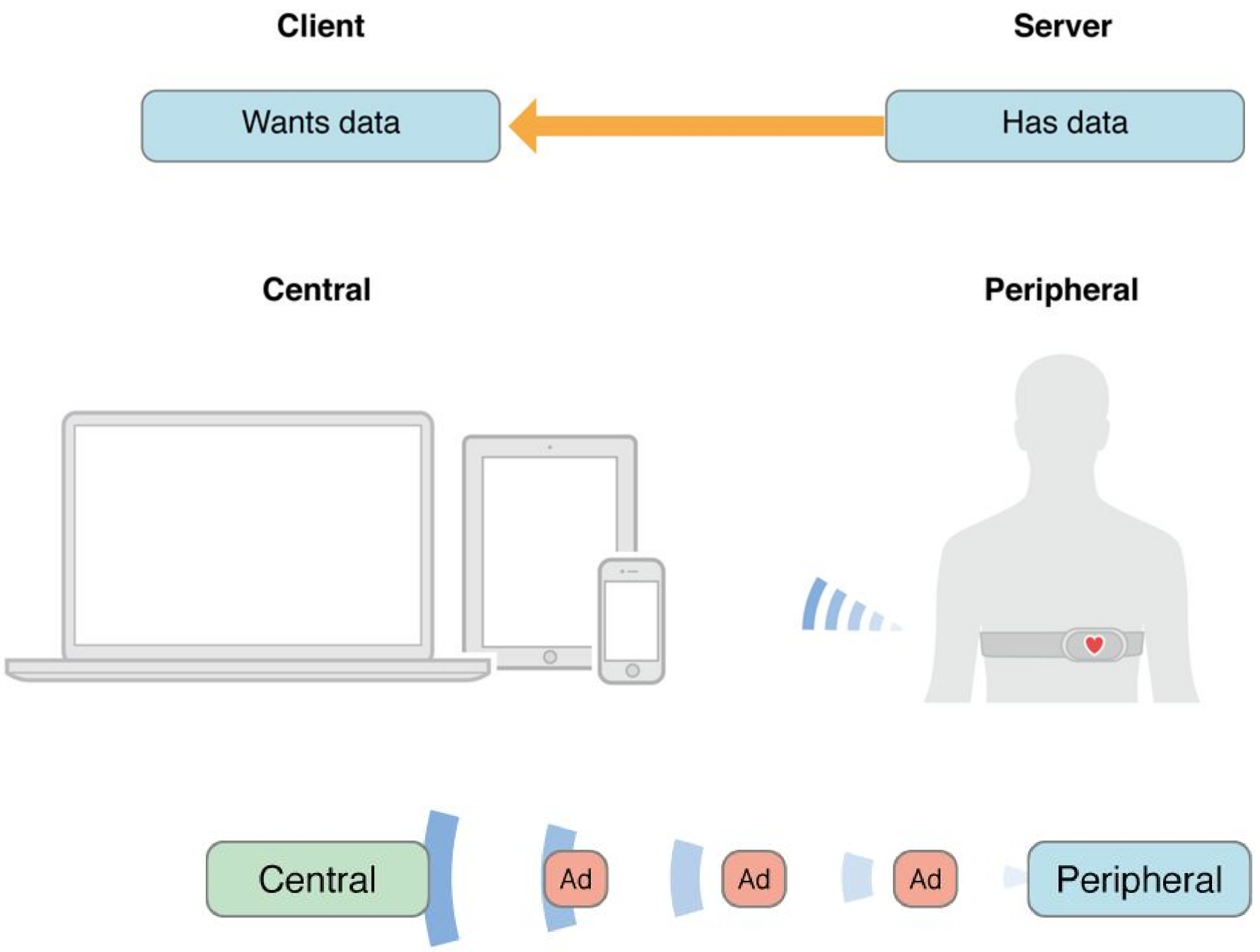


GATT Profile



Ref. Apple, Inc.

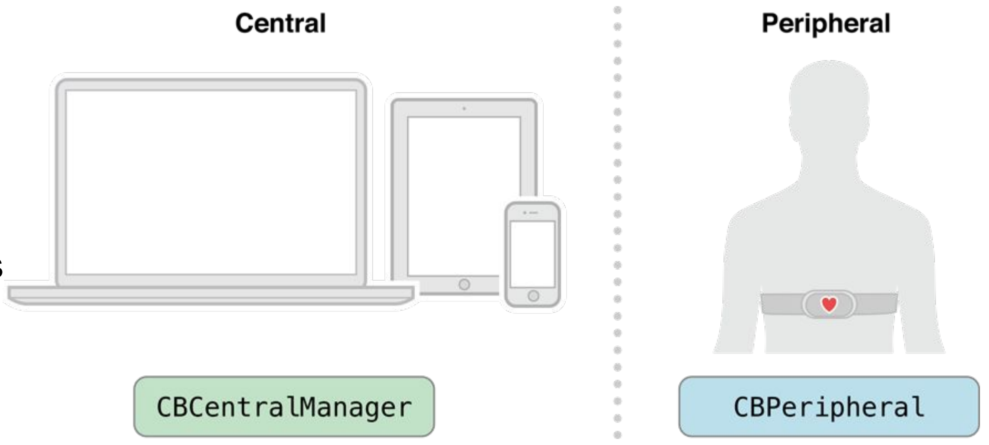
BLE: Roles



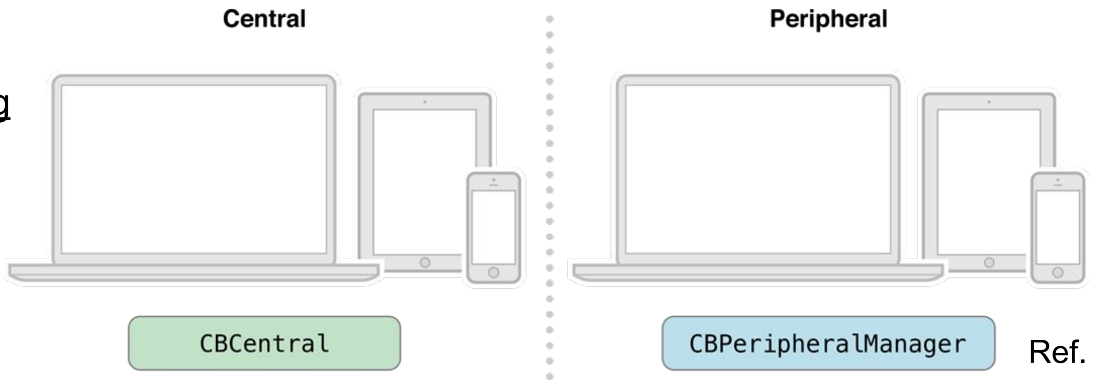
Ref. Apple, Inc.

Central vs Peripheral Programming

<u>Central(host) programming</u>
-Core Bluetooth objects on the central side
-Local centrals and remote peripherals



<u>Peripheral(device) programming</u>
-Core Bluetooth objects on the peripheral side
-Local peripherals and remote centrals

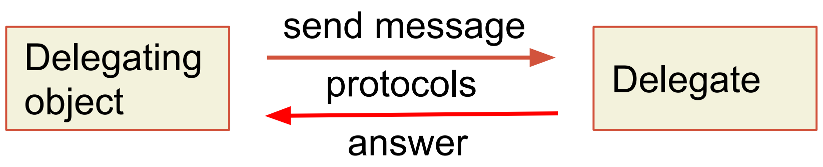


Ref. Apple, Inc.

Delegation vs Protocol

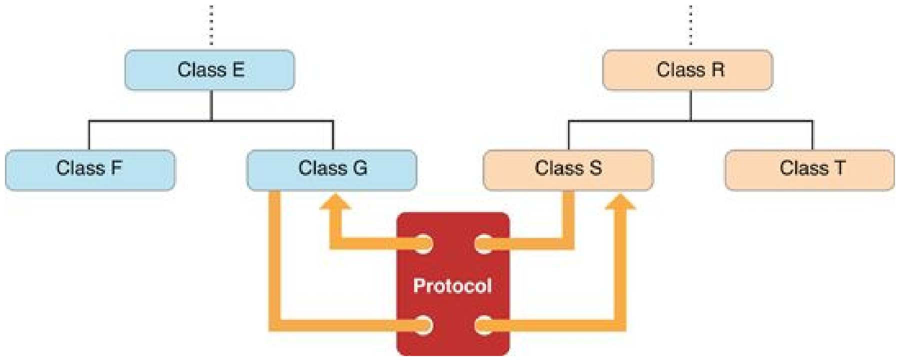
Delegation: Acting on Behalf of Another Object

In delegation, an object called the *delegate* acts on behalf of, and at the request of, another object. That other, delegating, object is typically a framework object. At some point in execution, it sends a message to its delegate; the message tells the delegate that some event is about to happen and asks for some response. The delegate (usually an instance of a custom class) implements the method invoked by the message and returns an appropriate value. Often that value is a Boolean value that tells the delegating object whether to proceed with an action. The delegating object has to keep track of the delegate and call upon it when needed by sending it a message.



Protocol: Enabling Communication Between Objects Not Related by Inheritance

A protocol is a declaration of a programmatic interface whose methods any class can implement. A protocol is thus, as is delegation, an alternative to subclassing and is often part of a framework's implementation of delegation.



Ref. Apple, Inc.

What is an API?

API: Application Programming Interface.

Application

software that we use

Programming

process of creating software

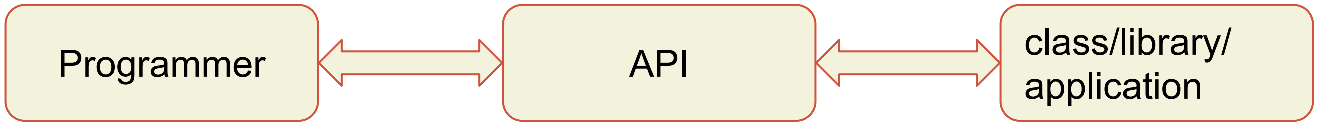
Interface

a common tool that enables two applications or programs to communicate with one another

API Call

When specific information from an API is needed, a program needs to call that API (make an API call).

API: a way for programmers to communicate with class/library/application.



Interface vs Implementation

- API describes
  - what a class/library/application does (interface of the class)
  - not how a class does it (implementation of the class)
  - Encapsulation: you have to implement API
- Interface
  - ex) CBCentralManager, CBCentralManagerDelegate
  - all we need to know to use a class
  - how we interact with a class
    - what methods to call
    - what they will return
- Implementation
  - ex) CBCentralManager: we don't care
  - ex) CBCentralManagerDelegate: we implement
  - using specific solutions for the given problems

Bytes and Bits

1 Byte = 8 bits (1 bit = binary number):

- smallest addressable unit of memory
- usually expressed in hex since 1 byte = 8 bits = 2^8 numbers
- a single character of text
- [0, 255] if positive value only
- [-128, 127] in Two's complement

ex) 0x00 = 0000 0000 = 0

0x0F = 0000 1111 = 2^0 + 2^1 + 2^2 + 2^3 = 2^4 - 1 = 15

0xFF = 1111 1111 = 2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7

= 2^8 - 1 = 255 or -1 in Two's complement

0xF0 = 1111 0000 = 2^8 - 2^4 = 240 or -16 in Two's complement

0x80 = 1000 0000 = 2^7 = 128 or -128 in Two's complement

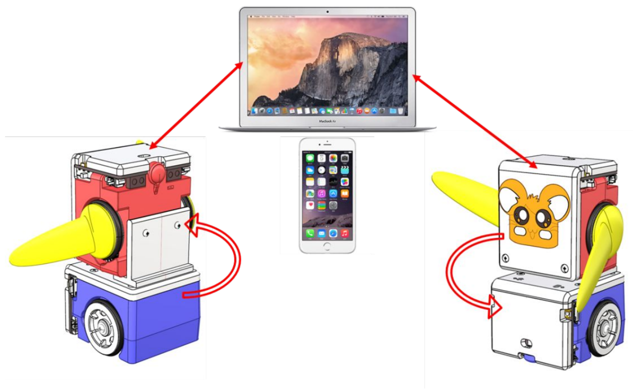
0x7F = 0111 1111 = 2^7 - 1 = 127

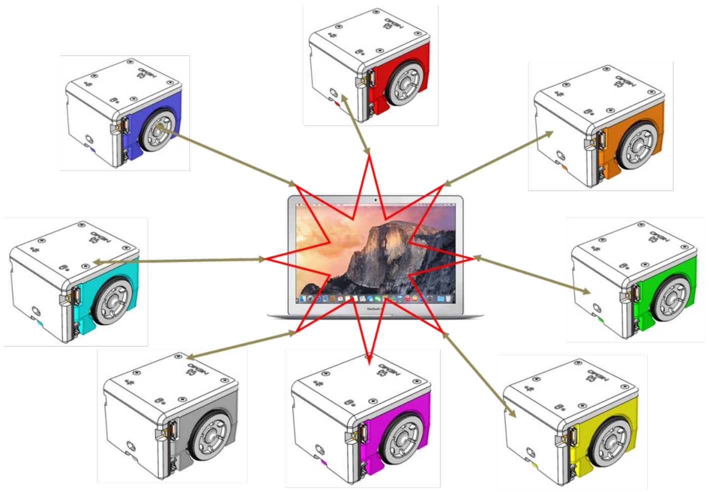
Lec#04: Structure & Mobile

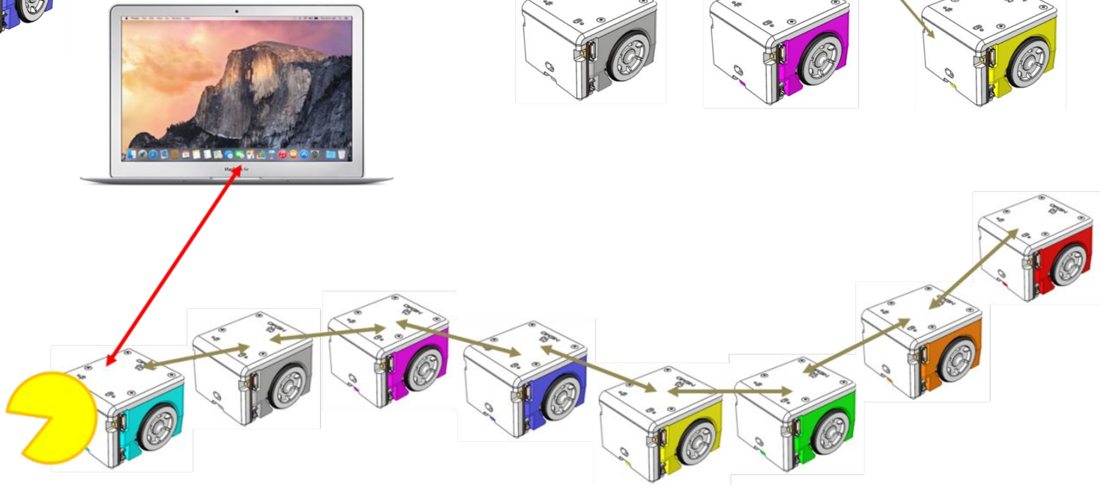
- Recap: Lecture #03
  - Bytes and Bits
  - Bitwise operators and masks
  - Two's complement
- BLE Mobile Device
  - Network Topology
  - Android
- Structure: Python Project
  - Module vs. Package
  - Importing
  - Utility functions

Hamster: Network Topology

Hamster: Star (8), Tree (3), Daisy chain







BLE Mobile Device: Android

Bluetooth Low Energy (Android API Guides)

Android 4.3 (API Level 18) introduces built-in platform support for Bluetooth Low Energy in the *central role*

1. **Roles and Responsibilities**
   a. Central vs. peripheral : (advertisement) : scanning vs advertising
   b. GATT client vs. GATT server : (data) : requesting vs providing
2. **BLE Permissions**
   a. Declare the Bluetooth permission(s) in your application manifest file (xml file)
   b. `<uses-permission android:name="android.permission.BLUETOOTH"/>`
   c. `<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>`
   d. `<uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>`
3. **Setting Up BLE**
   a. Get the BluetoothAdapter (BLE host hardware): CBCentralManager
   b. Check isEnabled(). Enable Bluetooth
4. **Finding BLE Devices**
   a. To find BLE devices, you use the startLeScan() method.
   b. This method takes a BluetoothAdapter.LeScanCallback as a parameter. You must implement this callback.
   c. **public abstract void onLeScan (BluetoothDevice device, int rssi, byte[] scanRecord)**
   d. **BluetoothDevice** (BLE device)
      i. connection
      ii. info

Ref. android.com by Google

BLE Mobile Device: Android

5. **Connecting to a GATT Server**
    a. To connect to a GATT server on a BLE device, you use the connectGatt() method.
    b. mBluetoothGatt = device.connectGatt(this, false, mGattCallback);
    c. **BluetoothGatt**:
        i. CBPeripheral
        ii. discoverServices(), writeCharacteristic(), setCharacteristicNotification(), etc
    d. BluetoothGattCallback:
        i. CBPeripheralDelegate
        ii. onServiceDiscovered(), onCharacteristicWrite(), onCharacteristicChanged() , etc
6. **Reading/writing BLE Attributes (receiving GATT notification)**
    a. **BluetoothGattService**
    b. **BluetoothGattCharacteristic**
    c. **BluetoothGattDescriptor**
7. **Receiving GATT Notifications**
    a. set a notification for a characteristic, using the setCharacteristicNotification()
    b. Once notifications are enabled for a characteristic, an onCharacteristicChanged() callback is triggered if the characteristic changes on the remote device:
8. **Closing the Client App**
    a. close() : release resources

Ref. android.com by Google

Structure: Python Project

- "Structure" means
  - making clean and effective code
  - whose logic and dependencies are clear
  - and files and folders are organized effectively in the filesystem
- Low-level layer
  - low-level manipulation of data
  - RobotDelegate
- Interface layer
  - interfacing with user actions
  - needs to import low-level file
  - RobotAPI
- Main application
  - needs to import interface file
  - hamster_class_api.py

Structure: Module vs Package

- Module
  - provides abstraction layer
  - namespace
  - a file
  - group of related classes
- Package
  - extension of module mechanism to a directory
  - namespace
  - a directory
  - group of related modules
  - __init__.py file
    - executed first when imported
    - gathers all package-wide definitions
    - empty file: normal practice if there is no

Structure: Importing

- Importing modules/packages
  - import
  - import … as
  - from … import
- Example
  - Very bad
    from math import *
    [...]
    x = sqrt(9) *# Is sqrt part of math? A built-in? Defined above?*
  - Better
    from math import sqrt
    [...]
    x = sqrt(9) *# sqrt may be part of math, if not redefined in between*
  - Best
    import math
    [...]
    x = math.sqrt(9) *# sqrt is visibly part of math's namespace*

Lec#05: Event Driven Behavior

- 2.1 Event Driven Programming
  - Programming Paradigms and Paradigm Shift
  - Event Driven Programming Concept
    - Tkinter – as a simple example
  - More on threads
  - Implementation of a simple event driven behavior for Hamster
- 2.2 Finite State Machine
  - Concept of FSM
  - Implementation details (a simple FSM for Hamster)
  - FSM driven by an event queue
- 2.3 Related Topics and Discussion
  - Concept of HFSM and BT

  (if time allows, not needed for projects)

Lec#06: Event Driven Behavior 2

- Threads
  - What are threads?
  - Why use threads?
  - Communication between threads?
- Queues
  - FIFO vs. Priority
  - Multi-thread safe
- Implementing an Event System using Threads and Queue
  - Dispatcher
  - Handlers
- Folder Structure (Behavior Package)
- Assignment#2-1

Lec#07: Finite State Machine

- Concept: Finite State Machine (FSM)
  - What are FSM's
  - Why / When to use FSM
- Implementation of Finite State Machines
  - FSM driven by an event queue
- Assignment#2-1

Lec#08: HFSM & BT

- HFSM: Hierarchical Finite State Machine
- BT: Behavior Tree

Topics For Part 3

3.1 The Robot Programming Problem

- What is “robot programming”
- Challenges
- Real World vs. “Virtual” World
    - Mapping and visualizing Hamster’s world
- A decomposition of the “mobile robot programming” problem

3.2 “Modeling” Hamster

- Hamster’s Motion and Sensors

3.3 Localization

- Where am I?
- Sub-goal navigation

3.4 Plan and Execution

- Motion Planning & Control with Uncertainty

Lec#09: Reasoning w/ Uncertainty

- Part 3-1: Challenges of Robot Programming
- What is robot programming
    - Modeling
    - Localization
    - Planning
    - Execution
    - Reactive is not enough: better knowledge of environment
- Physical world vs. virtual world
    - Modeling of Hamster: physical vs. virtual world
    - What does the robot see
    - How to make sense of what the robot see
- Graphic toolkit to help you visualize Hamster
- Homework Assignment Part #3-1

Lec#10: Localization

- Localization
  - Relative (Internal): dead reckoning
  - Absolute (External): distance sensors (Geometric feature detection), IR, Landmark
- Modeling Environment
  - Least Square (Fit): minimization
- Assignment#3-1 – demo and refine specification

Lec#11: Motion Planning

- Introduction to Robot Motion Planning
  - Configuration Space (C-Space) Approach
  - Basic Motion Planning Methods: Discretization
    - Visibility Graph, Voronoi Diagrams
    - Cell Decomposition: Exact, estimate
- Plan Execution (Control)
  - Virtual World (Perfect Control)
  - Real World (Uncertainty in control)
- Planning Under Uncertainty
  - Landmarks
  - Preimage backchaining
- Homework Assignment Part #3-2

Lec#12: Motion Planning & Control

- More on Motion Planning
  - Search (A*)
    - Uninformed (Blind): BFS, DFS
    - Informed (Heuristic): Evaluation function: Dijkstra’s, A*
  - Potential Field Method
- More on Control Under Uncertainty
  - Motion “Primitives”
  - Avoiding “Unexpected” Obstacles
- More on Assignment#3-2
  - student demo (Starbuck reward still good)

Lec#13: IO Extensions

- Future robots: Humanoids
- Recap Part 3: Reasoning with Uncertainty
- Part 4: I/O extensions
  - Electricity
  - Analog vs. Digital
  - ADC
  - PWM
  - Hamster I/O Mode

Analog to Digital Converter (ADC)

- ADC
  - converts an analog voltage on a pin to a digital number
  - converting from the analog world to the digital world
  - to use electronics to interface to the analog world
- Relating ADC Value to Voltage
  - The ADC reports a *ratiometric value*

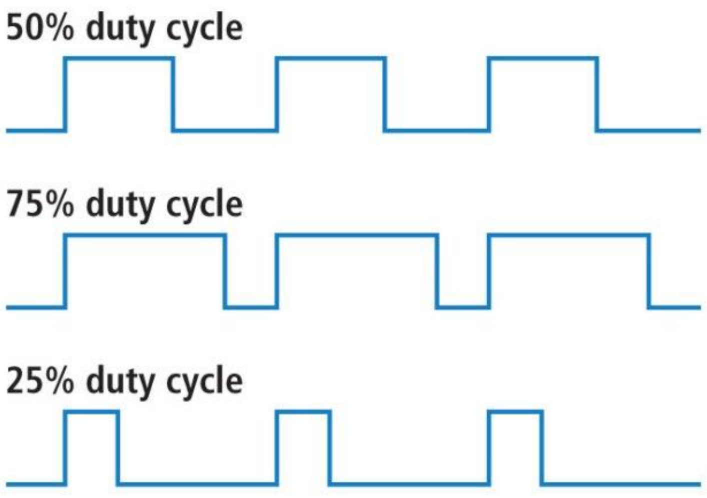
$$\frac{Resolution\ of\ the\ ADC}{System\ Voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

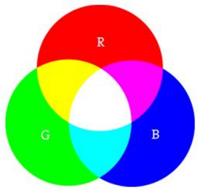
- Hamster
  - System Voltage = 3.7 V
  - Resolution of the ADC = 8 bit = 255 (= 0xFF)
  - Input (Analog): Voltage measured
  - Output (Digital): ADC Reading = Input * 255 / 3.7

Ref. learn.sparkfun.com

Pulse Width Modulation (PWM)

- Duty Cycle
  - on-time: when signal is high
  - duty cycle: amount of on-time
  - measured in % over a period
  - Ex) 5V
    - 50% duty cycle: 2.5V
- Examples
  - RGB LED
    - all equal duty cycle: white
  - Servo motors
    - frequency: 50 Hz waveform
    - duty cycle: 5~10%
    - 1.0 ms pulse: 0 deg
    - 1.5 ms pulse: 90 deg
    - 2.0 ms pulse: 180 deg





Ref. learn.sparkfun.com

Lec#14: IO Extensions 2

- Future robots: Disney a humanoid company?
- Recap Part 4-1: I/O extensions: ADC, PWM
- Part 4-2: I/O extensions 2
  - Better knowledge of environment: more sensors and effectors
  - Scanning
    - PSD Sensor, Servo motor
  - Filtering: Low-pass filter
    - Accelerometer, Signal strength
  - Feedback control
    - Line-tracing: floor sensors (grey scale: 0~100)
  - Modeling Environment
    - Least Squares (Fit): minimization

Low-pass filter: example

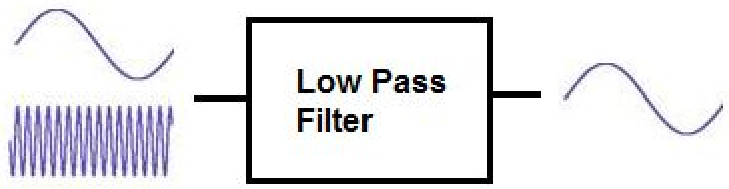
Simple infinite impulse response filter
-- First-order discrete-time realization
-- digital filter

```
// Return: RC low-pass filter output samples y
// Given: input samples x, time interval dt,
//          and time constant RC
 function lowpass(real[0..n] x, real dt, real RC)
   var real[0..n] y
   var real α := dt / (RC + dt)
   y[0] := x[0]
   for i from 1 to n
       y[i] := α * x[i] + (1-α) * y[i-1]
   return y
```

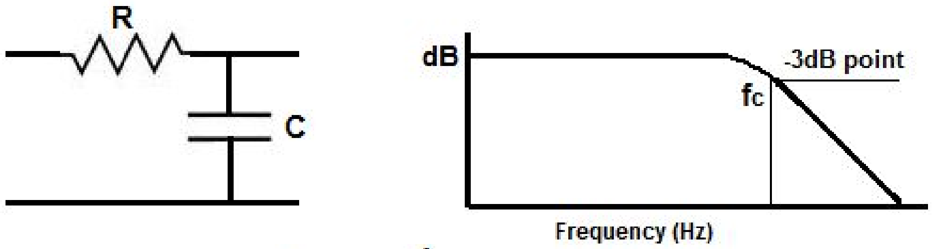
Note:

- α: *smoothing factor*
- False-positive vs. False-negative

Low Pass Filter Calculator



RC Low Pass Filter



$$f_c = \frac{1}{2\pi RC}$$

Ref. Wikipedia

Ref. www.learningaboutelectronics.com

Lec#15: Miscellaneous

- Part 5-1: Miscellaneous
    - Feedback control
        - Line-tracing: floor sensors (grey scale: 0~100)
    - Least Squares Method: line fitting from Scanning
    - Vision: OpenCV, Tracking, Demo by Kornel Niedziela (TA)
    -
- Part 5: Putting it together (Navigation)
    - Robot Programming
        - Modeling
        - Localization
        - Planning
        - Execution
        - UI / UX

Feedback control: Line-tracing

- Threshold Method
  - frequency
  - default_speed
  - more tweak required

```
def threshold(self, left, right):
    diff = 10
    speed_l = self._default_vel
    speed_r = self._default_vel
    if (self._line_loc == 0):
        if (left > self._threshold):
            speed_l = speed_l + diff
        else:
            speed_l = speed_l - diff
        if (right > self._threshold):
            speed_r = speed_r + diff
        else:
            speed_r = speed_r - diff
    return (speed_l, speed_r)
```

- Floor sensors: IR
  - Grey scale (0~100:white)
- Feedback Control
  - frequency
  - default_speed
  - p_gain: error multiplier
  - more robust
  -

```
def p_control(self, left, right):
        speed_l = self._default_vel
        speed_r = self._default_vel
        if (self._line_loc == 0):
            speed_l = self._default_vel
                        + self._kp * (left - right)
            speed_r = self._default_vel
                        - self._kp * (left - right)
        return (speed_l, speed_r)
```

- Demo

Line fitting: Least Squares Method

Step 1: Calculate the mean of the *x*-values and the mean of the *y*-values.

$$\overline{X} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \overline{Y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Step 2: The following formula gives the slope of the line of best fit:

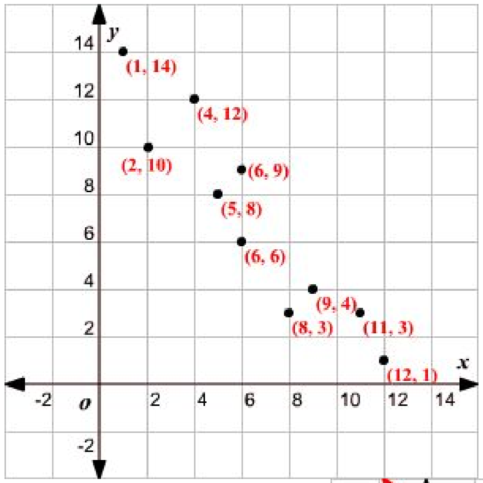
$$m = \frac{\sum_{i=1}^{n}\left(x_i - \overline{X}\right)\left(y_i - \overline{Y}\right)}{\sum_{i=1}^{n}\left(x_i - \overline{X}\right)^2}$$

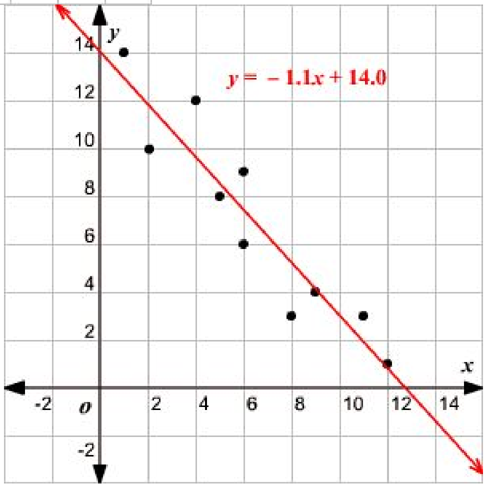
Step 3: Compute the *y*-intercept of the line by using the formula:

$$b = \overline{Y} - m\overline{X}$$

Step 4: Use the slope *m* and the *y*-intercept *b* to form the equation of the line.

$$y = mx + b$$





y = -1.1 x + 14

Ref. HotMath.com

Line fitting: Least Squares Method

```
import scipy.optimize as optimization
import numpy
….
def get_line(self, x, y):
        n = len(x)
        m, b = self.least_square_fit(x, y, 0, n)
        print m, b

def line_func(self, x, A, B):
        return A*x + B

def least_square_fit(self, x, y, i, f):
         xdata = numpy.array(x[i:f])
         ydata = numpy.array(y[i:f])
         popt, pcov = optimization.curve_fit( self.
   line_func, xdata, ydata)
          return popt
```

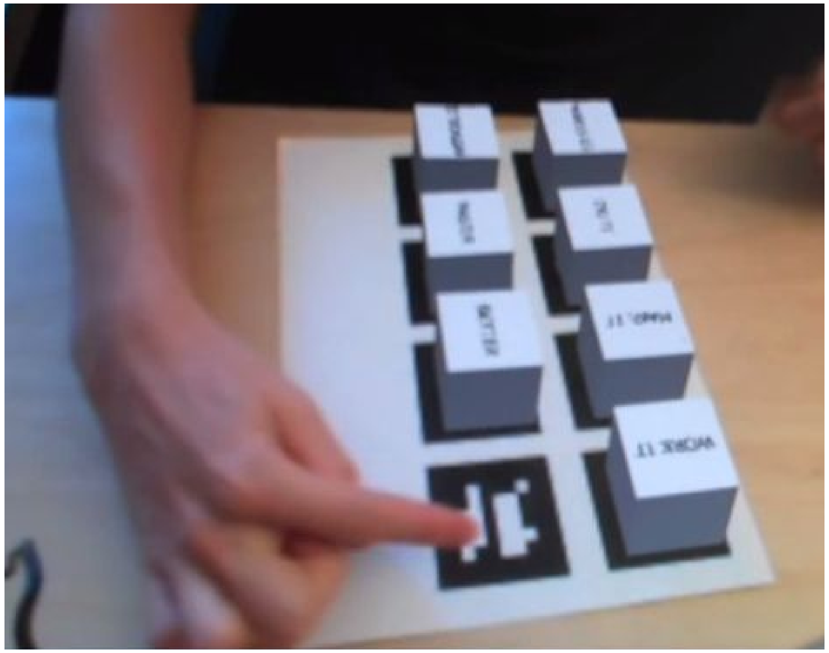
- Python Modules
  - http://www.scipy.org/
  - SciPy library:
    Fundamental library for scientific computing
  - NumPy package:
    Base N-dimensional array package
- Demo

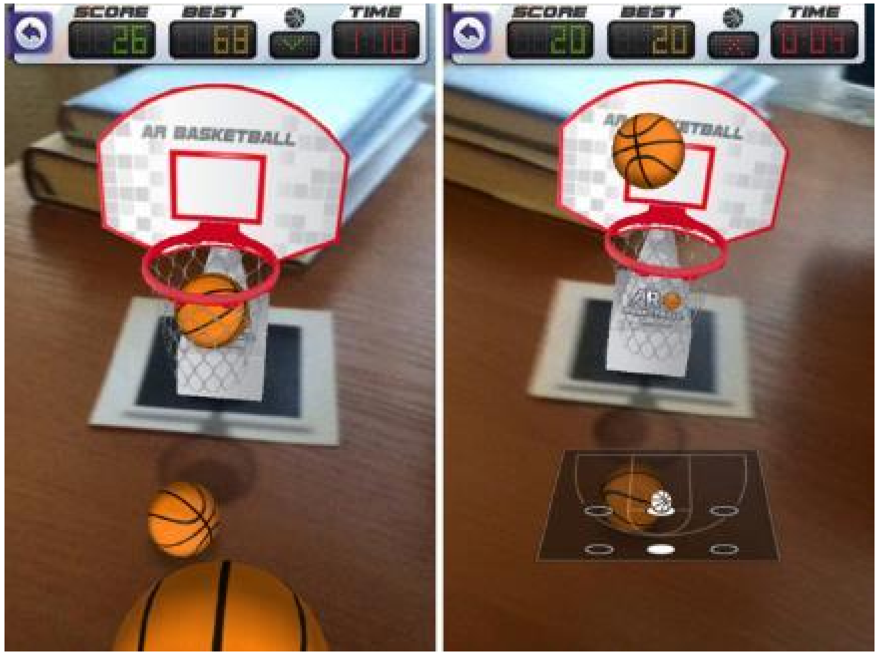
Vision: Tracking

- Prepared and presented by Kornel Niedziela (TA)
- OpenCV
- Tracking
- Demo

python-ar-markers

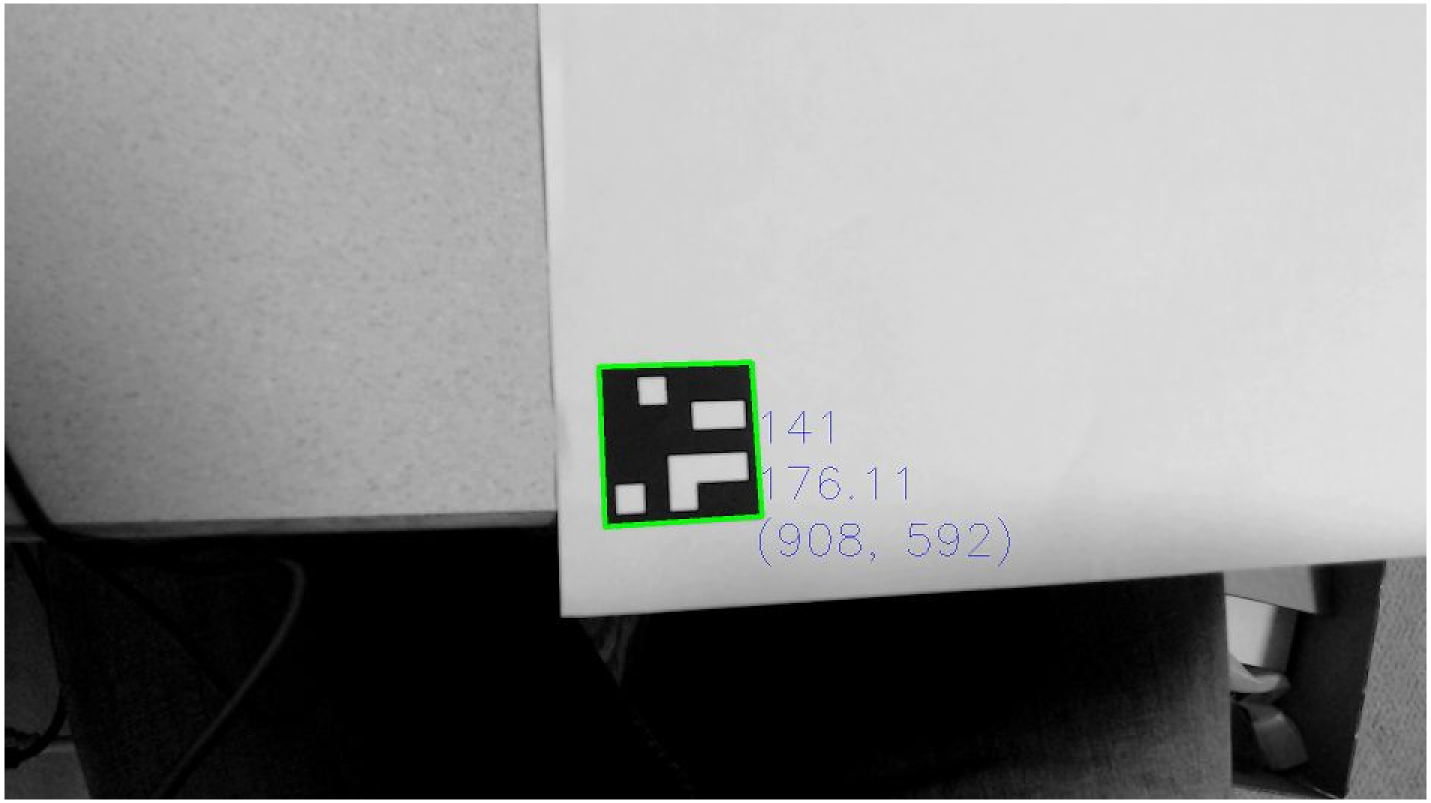
- 3rd party library to detect Augmented Reality markers
- Used for AR, but useful to track anything globally





python-ar-markers

- Fit polygon to contours (approxPoly2D)
  - Keep ones with 4 sides only
  - Calculate center of square for x/y, and angle of sides for rotation



Assignment#4

1. "Drop-off" problem: global localization with landmarks

   -- known map (known set of obstacles) + unknown position

   Solution) Make the scanning sensor work and model the sensor values.

   -- PSD IR Sensor + Servo motor

2. Collision detection problem

   -- false-positive vs. false-negative

   Solution) Apply low-pass filter and compare to the raw data.

   -- Accelerometer

3. UI/UX problem

   Solution) Model the control input and the sensor data graphically.

Assignment#4: Global Localization and Collision Detection

Final Project

- Robot programming
  - Mobile robot
  - Navigation
    - modeling
    - localization
    - planning
    - execution
    - UI/UX
  - Team of 2 people
    - Multiple robots

- CS 123 Final Project Proposal Guidelines

Reference and Reading

- None