About Me
A Little about me

- WPI Computer Science Professor
- Research interests:
  - mobile computing especially mobile health, computer graphics
- Started working in mobile computing, wireless in grad school
- CS + ECE background (Hardware + software)
- Current active research: Mobile health apps
  - E.g: AlcoGait app to detect how drunk Smartphone owner is
    - [https://www.youtube.com/watch?v=pwZaoKmfq8c](https://www.youtube.com/watch?v=pwZaoKmfq8c)
Administrivia
Administrivia: Schedule

- **Week 1-4:** I will introduce class, concepts, Android (Students: Android programming, assigned projects)
  - **Goal:** Students acquire basic Android programming skills to do excellent project
  - Focus on programming mobile & ubicomp components
- **Week 4:** Students will present final project proposal
- **Week 5-7:** Students work on final project
- **Week 7:** Students present + submit final projects
- Quizzes (5) throughout
Requirements to get a Grade

- **Grading policy:**
  - Assigned Projects 40%, Final project: 35%, Quizzes: 25%

- **Final project phases:** (See class website for deadlines)
  1. Pick partners, form project groups
  2. Submit 1-slide of proposed idea (problem + envisioned solution)
  3. Present project proposal
     + plus submit proposal (intro + related work + methodology/design + proposed project plan)
  4. Build app, evaluate, experiment, analyze results
  5. Present results + submit final paper (in week 7)

- **New final project aspects this offering:**
  - Larger teams (5 or 6 members)
  - Points for degree of difficulty of project
Course Texts

- **Android Texts:**
  - *Head First Android Dev, (2nd ed)*, Dawn and David Griffiths, O'Reilly, 2017

- Will also use official Google Android documentation
- Learn from research papers: Why not text?
Course Assistants

TA: Chai Nimkar

SA: Rachel Plante
Class in 2 Halves

- 2 Halves: About 50 mins each half
- Break of about 10 mins
- Talk to me at the end NOT during break
  - I need break too
Poll Question

- How many students:
  1. **Own** recent Android phones (running Android 4.4, 5, 6, 7 or 8?)
  2. **Can borrow** Android phones for projects (e.g. from friend/spouse)?
  3. **Do not own and cannot borrow** Android phones for projects?
Mobile Devices
Mobile Devices

- Smart phones (Blackberry, iPhone, Android, etc)
- Tablets (iPad, etc)
- Laptops
- Smartwatches
SmartPhone Hardware

- **Smart = Communication + Computing + Sensors**
  - **Communication**: Talk, text, Internet access, chat
  - **Computing**: Java apps, JVM, apps
    - Powerful processors: Quad core CPUs, GPUs
  - **Sensors**: Camera, video, location, temperature, heart rate sensor, etc

- Google Pixel XL phone: Quad core 1.6 GHz Snapdragon CPU, Adreno 530 GPU, 4GB RAM
  - A PC in your pocket!!
  - Multi-core CPU, GPU
  - Runs OpenGL ES, OpenCL and now Deep learning (Tensorflow)
Smartphone Sensors

- Typical smartphone sensors today
  - accelerometer, compass, GPS, microphone, camera, proximity
- Can sense physical world, inputs to intelligent sensing apps
  - E.g. Automatically turn off smartphone ringer when user walks into a class
Growth of Smartphone Sensors

- Every generation of smartphone has more and more sensors!!

Future sensors?
- Complex activity sensor,
- Pollution sensor,
- etc
Wireless Networks
Wireless Network Types

- **Wi-Fi (802.11)**: (e.g. Starbucks Wi-Fi)
- **Cellular networks**: (e.g. Sprint network)
- **Bluetooth**: (e.g. car headset)
- **Near Field Communications (NFC)**
  - e.g. Mobile pay: swipe phone at dunkin donut
## Wireless Networks Comparison

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Speed</th>
<th>Range</th>
<th>Power</th>
<th>Common Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLAN</td>
<td>600 Mbps</td>
<td>45 m – 90 m</td>
<td>100 mW</td>
<td>Internet.</td>
</tr>
<tr>
<td>LTE (4G)</td>
<td>5-12 Mbps</td>
<td>35km</td>
<td>120 – 300 mW</td>
<td>Mobile Internet</td>
</tr>
<tr>
<td>3G</td>
<td>2 Mbps</td>
<td>35km</td>
<td>3 mW</td>
<td>Mobile Internet</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>1 – 3 Mbps</td>
<td>100 m</td>
<td>1 W</td>
<td>Headsets, audio streaming.</td>
</tr>
<tr>
<td>Bluetooth LE</td>
<td>1 Mbps</td>
<td>100+ m</td>
<td>.01–.5 W</td>
<td>Wearables, fitness.</td>
</tr>
<tr>
<td>NFC</td>
<td>400 kbps</td>
<td>20 cm</td>
<td>200 mW</td>
<td>Mobile Payments</td>
</tr>
</tbody>
</table>

Table credit: Nirjoin, UNC

Different speed, range, power, uses, etc
Mobile Computing
**mobile**

*adjective*

/ˈməʊbəl, ˈmōˌbəl/

1. able to move or be moved freely or easily.  
"he has a major weight problem and is not very mobile"  
*synonyms*: able to move (around), **moving**, walking; **motile**; **ambulant**
Mobile Computing

- Human computes while moving
  - Continuous network connectivity,
  - Points of connection (e.g. cell towers, WiFi access point) might change
- **Note:** Human initiates all activity, (e.g launches apps)
- Wireless Network is *passive*
- **Example:** Using *foursquare.com* on Smartphone
Related Concept: Location-Awareness

- Mobile computing = computing while location changes
- **Location-aware**: Location must be one of app/program’s inputs
- Different user location = different output (e.g. maps)
- **E.g.** User in California gets different map from user in Boston
Location-Aware Example

- Location-aware app must have different behavior/output for different locations
- Example: Mobile yelp
  - **Example search:** Find Indian restaurant
  - App checks user’s location
  - Indian restaurants **close to user’s location** are returned
Example of Truly Mobile App: Word Lens

- Translates signs in foreign Language
- Location-dependent because location of sign, language? varies
Some Mobile apps are not Location-Aware

- If output does not change as location changes, not location-aware
- Apps run on mobile phone just for convenience
- Examples:
  - Distinction can be fuzzy. E.g. Banking app may display nearest locations

Mobile banking app  
Diet recording app
Which of these apps are Location-Aware?

a. Yahoo mail mobile

b. Uber app
Mobile Device Issue: Energy Efficiency

- Most resources increasing exponentially except battery energy (ref. Starner, IEEE Pervasive Computing, Dec 2003)

- Some energy saving strategies:
  - **Energy harvesting**: Energy from vibrations, charging mats, moving humans
  - **Scale content**: Reduce image, video resolutions to save energy
  - **Auto-dimming**: Dim screen whenever user not using it. E.g. talking on phone
  - **Better user interface**: Estimate and inform user how long each task will take
    - E.g: At current battery level, you can either type your paper for 45 mins, watch video for 20 mins, etc

![Diagram showing improvements in laptop technology from 1990 to 2001](image)
Ubiquitous Computing
ubiquitous
/yooˈbikweɪtəs/

adjective

present, appearing, or found everywhere.
"his ubiquitous influence was felt by all the family"
synonyms: omnipresent, ever-present, everywhere, all over the place, pervasive,
Ubiquitous Computing

• Collection of specialized assistants to assist human in tasks (reminders, personal assistant, staying healthy, school, etc)
• App figures out user’s current state, intent, assists them
• How? array of active elements, sensors, software, Artificial intelligence
• Extends mobile computing and distributed systems (more later)
• Note: System/app initiates activities, has intelligence
• Example: Google Assistant, feed informs user of
  • Driving time to work, home
  • News articles user will like
  • Weather
  • Favorite sports team scores, etc
• Also supports 2-way conversations
User Context

- Imagine a genie/personal assistant who wants to give you all the “right information” at the right time
  - Without asking you any questions
- Examples:
  - Detect traffic ahead, suggest alternate route
  - Bored user, suggest exciting video, etc
- Genie/personal assistant needs to passively detect user’s:
  - Current situation (Context)
  - Intention/plan
Ubicomp Senses User’s Context

- Context?
  - **Human**: motion, mood, identity, gesture
  - **Environment**: temperature, sound, humidity, location
  - **Computing Resources**: Hard disk space, memory, bandwidth
  - **Ubicomp example**:
    - **Assistant senses**: Temperature outside is 10F (environment sensing) + Human plans to go work (schedule)
    - **Ubicomp assistant advises**: Dress warm!

- Sensed **environment + Human + Computer resources = Context**

- **Context-Aware** applications adapt their behavior to context
Sensing the Human

- Environmental sensing is relatively straight-forward
  - Use specialized sensors for temperature, humidity, pressure, etc
- Human sensing is a little harder (ranked easy to hard)
  - **When:** time (Easiest)
  - **Where:** location
  - **Who:** Identification
  - **How:** (Mood) happy, sad, bored (gesture recognition)
  - **What:** eating, cooking (meta task)
  - **Why:** reason for actions (extremely hard!)
- Human sensing (gesture, mood, etc) easiest using cameras
- Research in ubiquitous computing integrates
  - location sensing, user identification, emotion sensing, gesture recognition, activity sensing, user intent

5 W’s + 1 H
Sensor

- **Example:** E.g. door senses only human motion, opens
- **Sensor:** device that can sense physical world, programmable, multi-functional for various tasks (movement, temperature, humidity, pressure, etc)
- Device that can take inputs from physical word
  - Also includes camera, microphone, etc
- Ubicomp uses data from sensors in phone, wearables (e.g. clothes), appliances, etc.

(courtesy of MANTIS project, U. of Colorado)  
RFID tags  
Tiny Mote Sensor, UC Berkeley
Ubiquitous Computing: Wearables
Ubiquitous Computing: Wearable sensors for Health

remote patient monitoring

UbiComp: Wearables, BlueTooth Devices

Body Worn
Activity Trackers

Bluetooth
Wellness
Devices

External sources of data for smartphone
Definitions: Portable, mobile & ubiquitous computing
Distributed Computing

- Computer system is physically distributed
- User can access system/network from various points.
- E.g. Unix cluster, WWW
- Huge 70’s revolution

**Distributed computing example:**
- WPI students have a CCC account
- Log into CCC machines,
- Web surfing from different terminals on campus (library, dorm room, zoolab, etc).

**Finer points:** network is fixed, Human moves
Portable (Nomadic) Computing

**Basic idea:**
- Network is fixed
- Device moves and changes point of attachment
- No computing while moving

**Portable (nomadic) computing example:**
- Mary owns a laptop
- Plugs into her home network,
  - **At home:** surfs web while watching TV.
- Every morning, brings laptop to school, plug into WPI network, boot up!
- **No computing while traveling to school**
Mobile Computing Example

- Continuous computing/network access while moving, automatic reconnection

**Mobile computing example:**
- John has SPRINT PCS phone with web access, voice, SMS messaging.
- He runs apps like facebook and foursquare, continuously connected while walking around Boston

**Finer points:**
- John and mobile users move
- Network deals with changing node location, disconnection/reconnection to different cell towers
Ubiquitous Computing Example

- **Ubiquitous computing**: John is leaving home to go and meet his friends. While passing the fridge, the fridge sends a message to his shoe that milk is almost finished. When John is passing grocery store, shoe sends message to glasses which displays “BUY milk” message. John buys milk, goes home.

- **Core idea**: ubiquitous computing assistants **actively** help John
SmartPhone Sensing
Smartphone Sensing

- Smartphone used to sense human, environment
  - **Example:** Human activity sensing (e.g. walking, driving, climbing stairs, sitting, lying down)
  - **Example 2:** Waze crowdsourced traffic
Sensor Processing

- **Machine learning** commonly used to process sensor data
  - Action to be inferred is hand-labelled to generate training data
  - Actual data is mined for combinations of sensor readings corresponding to action
- Example: Smartphone detects user’s activity (e.g. walking, running, sitting,) by classifying accelerometer sensor data
What Can We Detect/Infer using These Sensors

**Smartphone Sensing!!**

Smartphone Sensor data → Machine Learning

- Eating/Drinking
- Social interactions
- Cardiac health
- Sleep Quality
- Stress, Mood
- Activity
- Mobility patterns
- Conversations

Image Credit: Deepak Ganesan, UMass
Internet of Things (IoT)
IoT: Networked Smart Things (Devices)

- Smart things: Can be accessed, controlled over the network, learns users patterns

Nest Smart thermostat
- Learns owners manual settings
- Turns down heat when not around

Smart Fridge
- See groceries in fridge from anywhere
Other Ubicomp Systems

- **Smart Homes:** Continuously monitors elders who live in smart home, automatically dials 911 if elder ill, fall
  - Falls kill many old people who live alone

- **Smart buildings:** Senses presence of people, ambient temperature, people flow, dynamically adjusts heating/cooling
  - Can save over 40% of energy bill

- **Smart Cities:** Real time data from Sensors embedded in street used to direct drivers to empty parking spots
  - About 30% of traffic jam caused by people hunting for parking
Introduction to Android
What is Android?

- Android is world’s leading mobile operating system
  - Open source (https://source.android.com/setup/)

- Google:
  - Owns Android, maintains it, extends it
  - Distributes Android OS, developer tools, free to use
  - Runs Android app market
SmartPhone OS

- Over 80% of all phones sold are smartphones
- Android share 86% worldwide

Source: Statista
Android Growth

- Over 2 billion Android users, March 2017 (ref: the verge)
- 2.8 million apps on the Android app market (ref: statista.com)
  - Games, organizers, banking, entertainment, etc
Android is Multi-Platform

- Google Glass (being redone)
- In-car console
- Smartwatch
- Tablet
- Smartphone
- Television

Android runs on all these devices

This Class: Focuses Mostly on Smartphones!
Android for Mobile Computing and Ubicomp

- Android for Mobile programmable modules
  - Audio/video playback, taking pictures, database, location detection, maps

- Android for Ubicomp programmable modules
  - Sensors (temperature, humidity, light, etc), proximity
  - Face detection, activity recognition, place detection, speech recognition, speech-to-text, gesture detection, place type understanding, etc
  - Machine learning, deep learning
Android Versions

- Class will use Android 7 ("Nougat")
- Officially released December 5, 2016
- Latest version is Android 8 (Oreo), released August 2017
- Below is Android version distribution as at January 8, 2018

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.3-2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>0.4%</td>
</tr>
<tr>
<td>4.0.3-4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>0.5%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>1.9%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>2.9%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>0.8%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>12.8%</td>
</tr>
<tr>
<td>5.0</td>
<td>Lollipop</td>
<td>21</td>
<td>5.7%</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>22</td>
<td>19.4%</td>
</tr>
<tr>
<td>6.0</td>
<td>Marshmallow</td>
<td>23</td>
<td>28.6%</td>
</tr>
<tr>
<td>7.0</td>
<td>Nougat</td>
<td>24</td>
<td>21.1%</td>
</tr>
<tr>
<td>7.1</td>
<td></td>
<td>25</td>
<td>5.2%</td>
</tr>
<tr>
<td>8.0</td>
<td>Oreo</td>
<td>26</td>
<td>0.5%</td>
</tr>
<tr>
<td>8.1</td>
<td></td>
<td>27</td>
<td>0.2%</td>
</tr>
</tbody>
</table>

Android Developer Environment
New Android Environment: Android Studio

- Old Android dev environment used **Eclipse + plugins**
- Google developed its own IDE called **Android Studio**
- Integrated development environment, cleaner interface, specifically for Android Development (e.g. drag and drop app design)
- In December 2014, Google announced it will stop supporting Eclipse IDE
Where to Run Android App

- Android app can run on:
  - Real phone (or device)
  - Emulator (software version of phone)
Running Android App on Real Phone

- Need USB cord to copy app from development PC to phone
Emulator Pros and Cons (Vs Real Phone)

- **Pros:**
  - Conveniently test app on basic hardware by clicking in software
  - Easy to test app on various emulated devices (phones, tablets, TVs, etc), various screen sizes

- **Cons:**
  - Limited support, access to hardware, communications, sensors
  - E.g. GPS, camera, video recording, making/receiving phone calls, Bluetooth devices, USB devices, battery level, sensors, etc
  - Slower than real phone
New Support for Sensors

- Can now emulate some sensors (e.g. location, accelerometer), but still limited
Android Software Framework
Android Functionality as Apps

- Android functionality: collection of mini-applications (apps)
- Even dialer, keyboard, etc
Android Software Framework

- **OS:** Linux kernel, drivers
- **Apps:** programmed & UI in Java
- **Libraries:** OpenGL ES (graphics), SQLite (database), etc
Each Android app runs in its own security sandbox (VM, minimizes complete system crashes)

Android OS multi-user Linux system

Each app is a different user (assigned unique Linux ID)

Access control: only process with the app’s user ID can access its files

Ref: Introduction to Android Programming, Annuzzi, Darcey & Conder
References

- Android App Development for Beginners videos by Bucky Roberts (thenewboston)
- Ask A Dev, Android Wear: What Developers Need to Know, https://www.youtube.com/watch?v=zTS2NZpLyQg
- Busy Coder’s guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014