Sensors, Actuators, and Platforms
Motivation

- Embedded systems **operate** in, **interact** with, and **react** to an analog, real-time world
  - Interfacing with this world is not easy or monolithic

- **Sensors**: provide measurements of the outside world
- **Actuators**: provide an output, or a means to modify the physical world
- **Platforms**: everything in between – the framework that allows a system to analyze sensors, process their data, and drive actuators
Platforms

- Infrastructure between sensors and actuators
  - Especially important to embedded systems

- Increased emphasis on embedded software structure and functionality
  - Why?
Embedded Platform Requirements

- Operate within hardware/resource constraints
- Real-time requirements
- Safety/Reliability
- Upgradability
- Limited manual interfacing
- I/O interface
Microcontrollers

- Low-power, low-capacity System-on-Chip (SoC)
  - Processor core
  - Memory (Flash ROM + RAM)
  - General-purpose I/O (GPIO)
- Intended for *sequential control* rather than general computation
- Interrupts for I/O handling, preemption
- Programmable interval timer
- Tools to implement MoCs
- “Flash” the controller to upgrade
- Software-driven GPIO + Conversion
Microcontrollers – Development Workflow

- Software development
  - Tools for formal models, MoCs…
  - …or just plain C!

- Libraries:
  - I/O
  - ADC, DAC
  - Basic memory management
  - Interrupts, timing, and constraints

- Compile for MCU
  - “Flash” the device – load the binary
  - Execute!
Microcontrollers: Arduino

- Atmel AVR MC (ARM on the higher-end models)
- I²C individually addressable serial bus for communicating with custom modules – “shields”
- GPIO
- Microcontroller pins exposed with established interface for direct connection

- Merge traditional MCU with a well-established library system and IDE
  - Simplify addressing & control
  - Program in C or C++
  - setup() – initialization
  - loop() – main loop
  - bootloader

```c
#define LED_PIN 13

void setup () {
  pinMode(LED_PIN, OUTPUT); // Enable pin 13 for digital output
}

void loop () {
  digitalWrite(LED_PIN, HIGH); // Turn on the LED
  delay(1000); // Wait one second (1000 milliseconds)
  digitalWrite(LED_PIN, LOW); // Turn off the LED
  delay(1000); // Wait one second
}
```
General-Purpose Embedded Processing - ARM

- Designed around embedded use
  - RISC for reduced size, power, cost

- Specialized DSP instructions for I/O

- Multiple CPU modes allow for interrupts, RTOS task scheduling and device monitoring

- Conditional execution for nearly every CPU instruction

- SIMD extensions for streaming data use
Direct ARM Deployment – ARM SDT

- ARM Software Development Toolkit (SDT)
  - Development tools for both CL and Windows, incl.:
    - C, C++, Thumb compilers, assemblers & linkers
    - Project management software
    - Utilities & debuggers
    - ARMulator – ARM core emulator

- Workflow very similar to that of MCUs, but more tools and libraries to work with:
  - Code optimization
  - Object-oriented programming
  - Memory management
  - ARM big.LITTLE
  - GPGPU processing

- Discussion: What is the ease of deployment compared to MCUs?
Indirect ARM Deployment - OS

- OS takes care of system management – threads, memory & caching, I/O handling
- Dozens of popular ARM operating systems
  - Linux (including RTLinux)
  - VxWorks (RTOS)
  - TinyOS (unofficial)
  - iOS
  - Android
  - Windows RT
  - …and more!
Android – OS-based ARM Deployment

![Diagram of Android system architecture]

- **Applications**
  - Home
  - Contacts
  - Phone
  - Browser
  - ...

- **Application Framework**
  - Activity Manager
  - Window Manager
  - Content Providers
  - View System
  - Package Manager
  - Telephony Manager
  - Resource Manager
  - Location Manager
  - Notification Manager

- **Libraries**
  - Surface Manager
  - Media Framework
  - SQLite
  - OpenGL ES
  - FreeType
  - WebKit
  - SGL
  - SSL
  - libc

- **Android Runtime**
  - Core Libraries
    - Dalvik Virtual Machine

- **Linux Kernel**
  - Display Driver
  - Camera Driver
  - Flash Memory Driver
  - Binder (IPC) Driver
  - Keypad Driver
  - WiFi Driver
  - Audio Drivers
  - Power Management
Kernel Layer

- At heart, Android is based on Linux (Version 2.6 and higher)
- Provides OS services:
  - Memory Management
  - Process/thread handling
  - Network stack
  - Drivers – system, hardware devices, sensors, power management
- Abstracts hardware away from software stack
- Provides interface to Android Libraries

- **Kernel-level development is performed here**
  - Code is written in C, and entire OS is either compiled from source, or *cross-compiled* for the device and installed as a module
  - Identical to Linux kernel development
Android OS Layer

- **System Libraries** – abstracts kernel interface for Application Framework APIs

- **Kernel-level interfaces** are abstracted and extended with developer libraries

- **Common Libraries**:
  - **SystemC**: C library, optimized for embedded
  - **Media**: playback and decoding of common formats
  - **Surface Manager**: display subsystem, composes multiple implementations of 2D, 3D graphics
  - **LibWebCore**: web browser engine
  - **SGL, 3DLib**: 2D, 3D graphics, respectively
  - **SQLite**: compact DB engine – within Android, used for app data storage

- **Library development is performed here**
  - Code is written in C/C++, and OS is either compiled from source
  - Platform designers introduce special libs, hooks to custom sensors, etc.
Android Runtime

- Sits alongside system libraries – *unique* to Android and Java-based operating systems

- Libraries for compilation, core Java libraries

- Dalvik Virtual Machine – Android-specific JVM
  - Interface b/w compiled Java apps and underlying Linux kernel
  - Enables full-fledged Java features (threading, memory management)
Application Framework

- The heart of the Android front-end
  - Sits on top of JVM and system libraries, provides functionality for both user (Angry Birds, etc.) and core applications (Phone, Browser, etc.)

- Contains system resources and interfaces needed for applications, extensible with new, custom libraries
  - Enables reusability of reliable modules – the power of Android
  - Views for display, Content Providers for data access, Resource Manager for system resources, Notification and Activity Managers

- Application development is performed on top of this
  - Code is written in Java, using libraries found in the App Framework
Individual applications are composed of *Application Components*

- **Activities**: each UI screen + interaction elements (applications can have multiple activities)
- **Services**: background, long-running processes, with no UI components
- **Content Providers**: application data storage interface, for file system, SQLite, etc.
- **Broadcast Receivers**: non-UI listener for system events and broadcast messages
Android Manifest file

- AndroidManifest.xml
  - Ties the application together, declaring components and their associated attributes
  - Declares overall application specifications

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest ...
  <application android:icon="@drawable/app_icon.png" ...
    <activity android:name="com.example.project.ExampleActivity"
      android:label="@string/example_label" ...
    </activity>
    ...
  </application>
</manifest>
```
Activities

- Declare needed libraries in an activity manifest
  - Auto-generated libraries for maintaining activity life cycle (resumed, paused, stopped)
  - Maintain saved states for paused and stopped, and recover when resumed.
Services

- **Bound** to other components via manifest, which can start/stop a service
- Handle background work, data retrieval from sensors and other sources
- Store data in local or persistent storage, or pass directly to bound component via broadcast receivers
Content Providers

- Provides access to storage, either local (file system, SQLite DB) or remote (HTTP, FTP, etc.)
- Content component is identified with URI
- Register needed content with components via manifests and application permissions
- Built-in Examples: Contacts, Calendar

- Use the same content provider among applications for sharing
- Built-in providers:
  - SQLite database (with ACID provision and queryability)
  - File system (create, modify, delete)
  - Message passing
Intents

- Android component messaging system
  - Starts/stops *activities, services, broadcast receivers*
  - Message body determines target, action, and (optionally) any data

```xml
<activity android:name="NotesList" android:label="@string/title_notes_list">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <action android:name="android.intent.action.EDIT" />
        <action android:name="android.intent.action.PICK" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.GET_CONTENT" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
    </intent-filter>
</activity>
```
ARM Deployment – Raspberry Pi

- Low-tier ARM SoC (700 MHz)
  - Flash-based persistent storage
  - Onboard 256/512MB RAM
  - GPIO for peripheral connectivity
  - I²C open-pin bus
  - Linux
  - USB-connected Ethernet + USB ports
  - HDMI Out
- Development:
  - Python, primary
  - C, C++, Ruby, supported
Sensors

- **Goal**: create a mapping of the physical world

- **Basic Types**:
  - **Analog**: generate a voltage or current difference that must be measured and processed
    - e.g. ambient light
  - **Digital**: sensors directly generate a digital value
    - e.g. GPS
Sensor Issues

- Multi-property sensitivity (e.g. temperature)
- Processing overhead
- Drift
- Noise
- Power
- Accuracy
- Latency
Smartphone Built-in Sensors

- Proximity sensor
- GPS, A-GPS
- Ambient light
- Gyroscope
- Environment – air temperature, pressure, humidity
- Capacitive/resistive touch
- Camera
- RFID/NFC
Onboard MEMS Accelerometer

- Microelectromechanical device
  - Motion triggers displacement of mass, creates a change in capacitance
  - Sensor measures the change in capacitance
  - Multiple sensors in different orientations determine different axes
External Sensor Examples

- **Electrochemical**
  - Active electrode exposed to gas (or liquid), reference electrode insulated
  - Change in current between the two is proportional to fractional volume \( \rightarrow \) translates to concentration

- **Biosensor**
  - Receptor (ligand) binds to target molecules, causing electrical or chemical change

- **MOS Sensor**
  - Clean air causes adsorption of donor electrons, preventing current flow
  - Presence of other gases reduces surface density of oxygen, increasing current flow
Sensor-Facing Platforms

- **Quantization**: digital representation of an analog value
  - Quantization error
  - Data format
  - Resolution
  - A/D conversion overhead
- **Sampling**: convert analog $f(t)$ into time-series sequence
  - Discretization
  - Sampling interval
  - Nyquist rate/Aliasing
  - Power/accuracy tradeoff
- **Noise**
MCU Development Workflow

IDE

C file

Compiler/Assembler

Hex file

Loader

Binary file

Microcontroller

Flash memory

Program Execution

CPU, I/O, etc.