Interfacing the Mac to the Real World

Boisy G. Pitre

About Me

- Senior Software Engineer in the Mac Products Group at Nuance Communications

- Owner of Tee-Boy, developer of *WeatherSnoop*

- Author of *Developer to Developer* monthly column in MacTech Magazine


Presentation Goals

- Understand the idea of "Real-World Data"
- Explore Interfaces to Obtain the Data
- Look at Example Code
- Hardware Demonstrations

Real World Data

- Data representing quantifiable and measurable properties
  - Velocity
  - Temperature
  - Pressure
  - Flow Rates

- Sampled by specialized sensors

Applications

- Industrial & Commercial
- Automotive
- Medical
- Home Automation
- Environmental

Why the Mac?

- Reliability
  - BSD Unix Foundation
  - A Safe Environment
- A Philosophy of Simplicity and "Just Works"
- New Interface Technologies such as Multi-Touch and Speech can bring data to "life"

Interfacing Options

- USB (IOKit Framework)
- Serial Port (BSD I/O)
- Network (NSURLConnection)

Interfacing via USB

IOHID Manager

- Apple's API for User level access to USB
- Available in OS X 10.5 and later
- Entitlement: com.apple.security.usb (currently broken!)

Accessing HID Devices

- RunLoop based
- Register Match/Removal Callbacks based upon USB vendor & product ID
- When device inserted, register for input callback

Setting Up

```
#import <IOKit/hid/IOHIDManager.h>

uint16_t vendorID = 0x1DFD;
uint16_t productID = 0x0002;

gHIDManager = IOHIDManagerCreate(kCFAllocatorDefault, kIOHIDOptionsTypeNone);

NSDictionary *matchDict = [NSDictionary dictionaryWithObjectsAndKeys:
                                [NSNumber numberWithInt:vendorID], @kIOHIDVendorIDKey,
                                [NSNumber numberWithInt:productID], @kIOHIDProductIDKey,
                                nil];

IOHIDManagerSetDeviceMatching(gHIDManager, (CFDictionaryRef)matchDict);

// Callbacks for device plugin/removal
IOHIDManagerRegisterDeviceMatchingCallback(gHIDManager, Handle_DeviceMatchingCallback, self);
IOHIDManagerRegisterDeviceRemovalCallback(gHIDManager, Handle_DeviceRemovalCallback, self);

// Schedule with the run loop
IOHIDManagerScheduleWithRunLoop(gHIDManager, CFRunLoopGetCurrent(), kCFRunLoopCommonModes);
```

At Insertion

```
long reportSize = 0;

gHidDeviceRef = inIOHIDDeviceRef;

(void)IOHIDDevice_GetLongProperty(gHidDeviceRef, CFSTR(kIOHIDMaxInputReportSizeKey),
      &reportSize);

if (reportSize)
{
    report = calloc(1, reportSize);
    if (report)
    {
        IOHIDDeviceRegisterInputReportCallback(inIOHIDDeviceRef,
                                               report,
                                               reportSize,
                                               Handle_IOHIDDeviceInputReportCallback,
                                               self);
    }
}
```

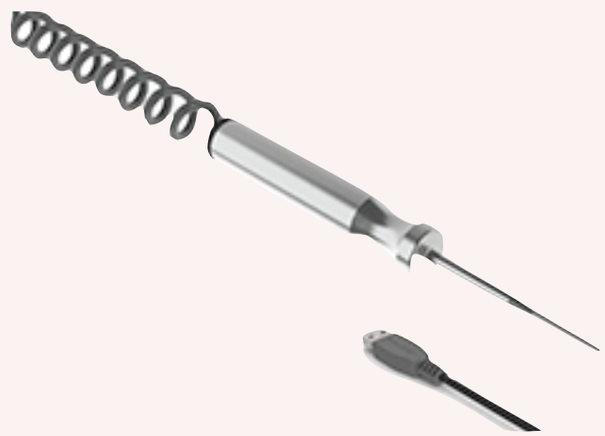
Input Report Callback

```
static void Handle_IOHIDDeviceInputReportCallback(void *          inContext,
                                                  IOReturn        inResult,
                                                  void *          inSender,
                                                  IOHIDReportType inType,
                                                  uint32_t        inReportID,
                                                  uint8_t *       inReport,
                                                  CFIndex         inReportLength)
{
    NSData *incoming = [NSData dataWithBytes:inReport length:inReportLength];

    . . .
}
```

USB Demo:

Thermistor



MACTECH CONFERENCE 2011

TEE-BOY

Interfacing via Serial Port

Accessing Serial Devices

- Utilizes USB to RS-232 convertor
- Requires installation of 3rd-party drivers (Prolific & FTDI)
- Least User-Friendly Experience
- Being usurped by USB


Serial Details

- Find device via IOKit function calls
- Open path to `/dev/tty.XXX` **via** `open()`
- Consider baud rate, word size, parity
  - Settable with `ioctl()`
- Use `read()`/`write()` for data access

Serial Port Demo:

Rad OS X

Summary

- Real World Interfacing is about collecting data that is measurable and useful
- The Mac is an ideal real-world platform
- Interfacing is achievable through APIs

More Information

- Apple Technical Note TN2187
  "New HID Manager APIs for Mac OS X version 10.5"

- QTI Thermistor
  http://www.thermistor.com/

- GM-10 Radiation Detector
  http://www.blackcatsystems.com/

Boisy G. Pitre
boisy@tee-boy.com