Low-Level Hardware Programming for Non-Electrical Engineers

Jeff Tranter
Integrated Computer Solutions, Inc.
Agenda

- About the Speaker
- Introduction
- Some History
- Safety
- Some Basics
- Hardware Interfaces
- Sensors and Other Devices
- Embedded Development Platforms
- Relevant Qt APIs
- Linux Drivers and APIs
- Tips
- Gotchas
- References
About The Speaker

Jeff Tranter <jtranter@ics.com>
Qt Consulting manager at Integrated Computer Solutions, Inc.
Based in Ottawa, Canada
Used Qt since 1.x
Originally educated as Electrical Engineer
Some History

• 1970s: Hard-coded logic
• 1980s: 8-bit microprocessors (assembler)
• Today: 64-bit, multicore, 3D, etc. (high-level languages)

• This presentation won't cover:
  • Programming languages other than C/C++
  • Much systems other than embedded Linux
  • Video, sound
  • Building embedded software: cross-compilation, debugging, etc.
A Few Words About Safety

- High voltage
- High current (e.g. batteries)
- High temperature (e.g. soldering)
- Eye protection (solder, clip leads)
- Chemicals
ESD

• Electrostatic discharge, i.e. static electricity
• Many devices can be damaged by high voltages from static
• Use static safe packaging, work mat, wrist strap, soldering iron
Some Basics

- Ohms Law: \( I = \frac{V}{R} \) (sometimes \( E \))
- Power \( P = V \times I \)
Measuring (e.g. with a multimeter)

- Voltage - in parallel (across)
- Current - in series (break the circuit)
- Resistance - out of circuit, powered off
**Common Electronic Components**

- **Passive components:**
  - resistor  unit: Ohm (kilohm, megohm)
  - capacitor unit: Farad (µF, nF, pF)
  - inductor  unit: Henry (µH, mH)

- **Active components:**
  - vacuum tube (valve)
  - diode/LED
  - transistor (many types)
  - ICs (many types)
Common Electronic Components

• Components identified by:
  • part identifier (e.g. 7400)
  • value (e.g. 1000 ohms)
  • power rating (e.g. 1 watt)
  • voltage rating (e.g. 10 VDC)
• Component values marked using colour codes or number conventions
# Common Metric Prefixes

<table>
<thead>
<tr>
<th>Name</th>
<th>Prefix</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pico</td>
<td>p</td>
<td>10⁻¹²</td>
</tr>
<tr>
<td>Nano</td>
<td>n</td>
<td>10⁻⁹</td>
</tr>
<tr>
<td>Micro</td>
<td>µ</td>
<td>10⁻⁶</td>
</tr>
<tr>
<td>Milli</td>
<td>m</td>
<td>10⁻³</td>
</tr>
<tr>
<td>Kilo</td>
<td>k</td>
<td>10³</td>
</tr>
<tr>
<td>Mega</td>
<td>M</td>
<td>10⁶</td>
</tr>
<tr>
<td>Giga</td>
<td>G</td>
<td>10⁹</td>
</tr>
<tr>
<td>Tera</td>
<td>T</td>
<td>10¹²</td>
</tr>
</tbody>
</table>
Digital versus Analog

• **Digital**: represent values/numbers using discrete voltages
• Modern computers generally use binary, two values, 1/0, true false
• Value represented as a voltage within a range, dependent on technology used
• e.g. standard TTL logic - 0 to 0.4V is false, 2.6 to 5.0V is true
Digital versus Analog

- **Analog**: can take any value within a continuous range
- Digital to Analog (D/A) and Analog to Digital (A/D) conversion processes can convert
- Conversion between the two is not perfect
- Key factors are sample rate (samples/sec) and sample size (bits)
- e.g. Audio CD: 16 bit sample size, 44100 bits per second sample rate
Hardware Interfaces - Processor Terminology

- **CPU**: Central Processing Unit. Hardware within a computer that carries out the instructions of a computer program.
- **Microprocessor**: Incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit.
- **Microcontroller**: Small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.
- **SOC**: System On a Chip; integrated circuit that integrates all components of a computer or other electronic system into a single chip.
Hardware Interfaces - Processor Terminology

• **SOM**: System on Module (SOM), a type of single-board computer (SBC).
• **SiP**: System In Package (SiP), also known as a Chip Stack MCM. A number of integrated circuits enclosed in a single module (package).
• **DSP**: Specialized microprocessor optimized for the needs of digital signal processing.
• **GPU**: Graphics Processing Unit, specialized CPU designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display.
Hardware Interfaces - Memory

- RAM, DRAM, Static RAM
- ROM, PROM, EPROM, EEPROM
- Flash memory: NAND, NOR
Hardware Interfaces - Simple I/O

- Inputs
- Outputs
- Bi-directional
- Tri-state (high-Z), pull-up, pull-down
- Open collector/open drain
- Analog
- Digital
- PWM
Hardware Interfaces - I²C

• I²C (Inter-Integrated Circuit), pronounced I-squared-C or I-two-C
• Multi-master, multi-slave, single-ended, serial computer bus invented by Philips Semiconductor
• Used for attaching low-speed peripherals to computer motherboards and embedded systems
• Uses two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors.
• Typical voltages used are 5V or 3V, although other voltages are permitted
• Will cover programming under Linux later
Hardware Interfaces - SMBus

- System Management Bus (SMBus or SMB)
- Simple single-ended two-wire bus for lightweight communication
- Commonly found on PC motherboards for communication with power management
- Derived from I²C
- Defined by Intel in 1995
Hardware Interfaces - SPI

• Serial Peripheral Interface or SPI bus
• Also known as SSI (Synchronous Serial Interface)
• Full duplex, synchronous, serial data link
• Four-wire serial bus
• Often used with sensors and SD cards
• Devices communicate in master/slave mode
• Multiple slave devices are allowed with individual slave select lines
Hardware Interfaces - GPIO

- General-Purpose Input/Output
- Generic pin that can be controlled by user at run time
- Typically can be programmed as input or output
- May support tri-state, pull-up pull-down, PWM, etc.
- Supported by e.g. Arduino, BeagleBone, Raspberry Pi
Hardware Interfaces - USB

- Ubiquitous
- Latest spec is 3.1
- Sometimes used (only) for power
- See later for some gotchas
Hardware Interfaces - IEEE-488/GP-IB/HP-IB

- Short-range digital communications bus
- Created in the late 1960s by Hewlett-Packard for use with automated test equipment
- Expensive connectors and cables
- Now mostly replaced by more recent standards such as USB, FireWire, Ethernet
Hardware Interfaces - MODBUS

- Serial communications protocol
- Originally developed by Modicon for use with programmable logic controllers (PLCs)
- Commonly used for connecting industrial electronic devices
- Used in supervisory control and data acquisition (SCADA) systems
- Enables communication among many (approx. 240) devices connected to same network
Serial/UART RS-232/RS-422/RS-485

- Asynchronous serial interfaces, send one bit at a time
- Need to agree on baud rate, data bits, start/stop bits, parity
- RS-232 uses voltage levels of +/- 3-15V
- RS-422 is differential signalling, longer distance
- RS-485 supports multi-point
- Some USB devices are serial devices (e.g. FTDI)
- On newer computers can use USB to serial converter
Hardware Interfaces - Parallel Ports

• As a generic term, means port with multiple data bits (as opposed to single bit serial)
• Typically data and handshaking lines as well
• In the past referred to a standard Centronics/IEEE-1284 PC printer port, now mostly obsolete
Hardware Interfaces - JTAG

• Joint Test Action Group
• Common name for IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture
• Initially intended for testing printed circuit boards using boundary scan (still widely used for this)
• Also used for IC debug ports
• Most embedded processors implement JTAG
• Supports operations like single stepping and breakpointing (in hardware)
Hardware Interfaces - 1-Wire

• Device communications bus system designed by Dallas Semiconductor (sometimes called Dallas 1 Wire)
• Provides low-speed data, signalling, and power over a single signal
• Master and slave devices
• Similar to I²C, but with lower data rates and longer range
• Typically used to communicate with small inexpensive devices such as digital thermometers and weather instruments
• Only two wires: data and ground. Device also powered by data line.
• Can be supported on Linux using GPIO and bit banging
• OWFS One Wire File System provides library and utilities for Linux and other platforms (owfs.org)
Hardware Interfaces - HD44780 LCD

- One of the most common dot matrix LCD display controllers
- Simple interface that can be connected to a general purpose microcontroller or microprocessor
- Many manufacturers make compatible displays
- Can display ASCII characters, Japanese Kana characters, and some symbols
- Low cost (under US$20)
- Typically 2 line by 16 or up to 80 characters
- 16 pin connector, 4 or 8 data bits
- Various drivers/libraries available for Linux if you don't want to code it all yourself
Hardware Interfaces - MIDI

- Musical Instrument Digital Interface
- Standard protocol, interface, and connector for electronic musical instruments
- Carries event messages that specify notation, pitch and velocity
- Also used for lighting
- Supports multiple devices
- A single MIDI link can carry up to sixteen channels of information
- Standardized in 1983
- Mostly used by professional musicians
Hardware Interfaces - PC Keyboard

• Original PC/XT/AT (5-pin DIN)
• PS/2 (6-pin mini-DIN)
• USB (USB type A)
• (first 2) protocols can be implemented by bit banging
Hardware Interfaces - PWM

• Pulse Width Modulation
• Can be used for D/A conversion
• Some devices use PWM for control
• Can be done in software with GPIO pins
• Some GPIO pins have direct hardware support for PWM
Hardware Interfaces - Stepper Motors

- Brushless DC electric motor that divides a full rotation into a number of equal steps
- Motor's position can then be commanded to move and hold at one of these steps without any feedback sensor
- Unipolar and bipolar types
- Typically need driver circuit for suitable voltage/current
- Read/write heads of hard and floppy disk drives typically use this
- Easy to control using Arduino
Hardware Interfaces - Servos

• Usually refers to hobby servo motors developed for radio control
• Small, low-cost, mass-produced actuators used for radio control and small-scale robotics
• Standard three-wire connection: two wires for a DC power supply and one for control
• Position controlled using a PWM signal
• Directly supported by Arduino (without additional hardware)
Hardware Interfaces - DSI/CSI

• Display Serial Interface
• Camera Serial Interface
• Specifications by the Mobile Industry Processor Interface (MIPI) Alliance
• DSI for LCD displays
• CSI for cameras
• Serial bus and a communication protocol between host and device
• Both are present on Raspberry Pi but currently no open source drivers
Sensors and Related Devices

- IR transmitters/receivers
- Sensors for physical values: temperature, light intensity, air pressure, humidity, pH, radiation, motion, proximity, radiation, sound, touch, etc.
- Accelerometers
- Output: light (LED), sound (speaker, piezo), motion (motor, stepper)
- GPS
- Commonly interface using analog, digital, I\(^2\)C etc.
- See, e.g. http://www.adafruit.com/categories/35
Displays

- LEDs: discrete, bargraph, matrix, 7-segment
- LCD: numeric, text, bitmapped graphics
- Video: VGA, composite, HDMI, etc.
Real-Time Considerations

- System is real-time if correctness of an operation depends on the time in which it is performed.
- Classified by the consequence of missing a deadline.
  - **Hard real-time**: missing a deadline is a total system failure.
  - **Soft real-time**: usefulness/quality of service degrades after missing deadline.
- Supported by an RTOS (Real-Time Operating System).
- Standard Linux is not an RTOS.
Approaches for Supporting Real-Time

- Set priority, scheduling policy (e.g. Linux/POSIX: setpriority, sched_setscheduler)
- Implement in kernel
- Real-time add-ons (e.g. for Linux)
- True RTOS (e.g. QNX)
- Offload to other hardware like microcontroller or PIC
Embedded Development Platforms

- Many to choose from
- Most vendors have evaluation boards
- Some popular ones:
  - Raspberry Pi
  - BeagleBoard/BeagleBone
  - Intel NUC, Edison
  - Arduino
Raspberry Pi

- Developed as low-cost platform for education
- Broadcom SOC (700 MHz ARM)
- Supports various OSes including Linux
- USB, SD card, Ethernet, audio out, composite and HDMI video
- Micro USB power
- Model A: US$25, 256MB RAM, 1 USB
- Model B: US$35, 512MB RAM, 2 USB
- Model B+: lower power (3W), 4 USB, microSD, more GPIO
- Compute Module: DIMM form factor, suitable for OEM, more GPIO
BeagleBoard/BeagleBone

- Open source SBC from TI and Digi-Key and Farnell/Element14
- OMAP3530 SOC (ARM)
- 600 MHZ to 1 GHz clock speed
- 128MB to 52MB RAM
- USB On-The-Go, DVI-D, PC audio, SDHC, JTAG, HDMI
- Accelerated 2D, 3D, OpenGL ES 2.0
- On-board and SD/MMC flash
- Cost $45 to $149
- Models: BeagleBoard, BeagleBoard-xM, BeagleBone, BeagleBone Black
- Run various operating systems including Linux and Android
- Add-on "capes"
Intel Offerings

**NUC:**
- Next Unit of Computing (NUC)
- Small form factor PC designed by Intel

**Galileo:**
- Arduino-compatible development boards based on x86
- Compatible with Arduino IDE and shields

**Edison:**
- Small computer for wearable devices

**MinnowBoard:**
- Low-cost Atom board
Texas Instruments Offerings

• e.g. Sitara ARM AM335X Starter Kit
Arduino

- Family of open source single board microcontrollers
- Most use 8-bit Atmel AVR processors
- No operating system per se
Highly popular due to "perfect storm" of:

- Low cost (clones under US$10)
- Ease and speed of programming (easy to use IDE, high-level language based on simplified C++)
- Many programming tutorials, examples, libraries
- Large user base
- Digital and analog inputs/outputs
- Many add-on "shields"
#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
}
Relevant Qt APIs

Several Qt modules fit category of low-level hardware:
- Serial Port
- Networking
- BlueTooth
- Location/Positioning API (GPS, Wi-Fi)
- Sensors (accelerometer, compass, etc.)
Linux Drivers and APIs

• Will cover I²C, SPI, GPIO
• Can use these from user space
• In some cases may want to write kernel code
• Kernel pros: access to kernel interfaces such as IRQ handlers or other layers of the driver stack
• Kernel cons: harder to write and debug, error can crash entire machine
Linux Drivers and APIs - I²C

- Kernel-level drivers make I²C interfaces look like standard Linux character devices.
- Devices are /dev/i2c-n where n is adaptor number starting from 0
- Also see /sys/class/i2c-adapter
- Linux i2c-tools package provides useful utilities like "i2cdetect".
- Can program using standard system calls open(), ioctl(), read(), write()
Linux Drivers and APIs - I²C

- Also higher level SMBus commands defined in `<linux/i2c-dev.h>`
- (SMBus is a subset of I²C, with a stricter protocol definition)

```c
__s32 i2c_smbus_write_quick(int file, __u8 value);
__s32 i2c_smbus_read_byte(int file);
__s32 i2c_smbus_write_byte(int file, __u8 value);
__s32 i2c_smbus_read_byte_data(int file, __u8 command);
__s32 i2c_smbus_write_byte_data(int file, __u8 command, __u8 value);
__s32 i2c_smbus_read_word_data(int file, __u8 command);
__s32 i2c_smbus_write_word_data(int file, __u8 command, __u16 value);
__s32 i2c_smbus_process_call(int file, __u8 command, __u16 value);
__s32 i2c_smbus_read_block_data(int file, __u8 command, __u8 *values);
__s32 i2c_smbus_write_block_data(int file, __u8 command, __u8 length, __u8 *values);
```
On some platforms, like Raspberry Pi, one may need to manually load the relevant kernel drivers, e.g. "sudo modprobe i2c-dev" and set permissions if non-root access is needed, e.g. "sudo chmod o+rw /dev/i2c*"

- Can put a script in /etc/rc.local to do this on boot up
- See https://www.kernel.org/doc/Documentation/i2c/dev-interface
Linux Drivers and APIs - SPI

- Appear as character devices.
- Creates character device nodes at /dev/spidevB.C where:
  - B is the SPI bus (master) number
  - C is the chip-select number of specific SPI slave
- SPI devices have a limited user space API, supporting basic half-duplex read() and write() access to SPI slave devices.
- Using ioctl() requests, full duplex transfers and device I/O configuration are also available
Linux Drivers and APIs - SPI

• `read()` for read only SPI transaction, with a single chip-select activation
• `write()` for write only SPI transaction, with a single chip-select activation
• Defined in `<linux/spi/spidev.h>
• See https://www.kernel.org/doc/Documentation/spi/spidev
Linux Drivers and APIs - GPIO

- Linux has unified driver for GPIO on different platforms
- Often GPIO pins can also be I²C, SPI, PWM, UART, etc. depending on how programmed
- Typically can control on a per pin basis: pin direction (input or output), read inputs, write to outputs, and maybe pullup, pulldown, open collector, and enable interrupts
- Typically need to run as root or change permissions on device files
- See https://www.kernel.org/doc/Documentation/gpio/
- Different ways to control
Linux Drivers and APIs - GPIO

• **Method 1: Kernel system calls**
  • `#include <linux/gpio.h>`
  • Old, deprecated integer-based interface
  • New, preferred descriptor-based interface
  • Examples:
    • `int gpio_get_value(unsigned int gpio);`
    • `void gpio_set_value(unsigned int gpio, int value);`
Linux Drivers and APIs - GPIO

**Method 2: Sysfs**

*Can be controlled via sysfs interface under /sys/class/gpio*

*Need to "export" pin that you want to use by writing pin number to /sys/class/gpio/export*

*Will see /sys/class/gpio/gpio\_N appear*

*Write to /sys/class/gpio/unexport to free when done*

*Set direction by writing "in" or "out" to /sys/class/gpio/gpio\_N/direction*

*Set value by writing "0" or "1" to /sys/class/gpio/gpio\_N/value*

*Read value from /sys/class/gpio/gpio\_N/value*

*Info about GPIO controllers in /sys/class/gpio/gpiochip_N/*
Linux Drivers and APIs - GPIO

• Raspberry Pi example (shell script):

```bash
#!/bin/sh
echo "4" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio4/direction
echo "1" > /sys/class/gpio/gpio4/value
cat /sys/class/gpio/gpio4/value
echo "4" > /sys/class/gpio/unexport
```
• Method 3: Memory Mapped
  • Works on devices where GPIO hardware is memory mapped e.g. Raspberry Pi
  • Hardware specific, but very fast
  • Steps:
    • Open /dev/mem
    • Call mmap() to get pointer to appropriate physical memory
    • Close /dev/vmem
    • Access memory as a volatile unsigned * (macros can make it easier)
  • Examples (with macros) exist for Raspberry Pi:
    http://elinux.org/RPi_Low-level_peripherals#C
Linux Drivers and APIs - GPIO

• **Method 4: Kernel Driver**
  • For maximum performance and flexibility, write customer kernel level code
  • As mentioned earlier, provides access to kernel interfaces such as IRQ handlers or other layers of the driver stack and gives you control over preemption.
  • Typically an order of magnitude harder than user space code to write and debug; errors can crash entire machine.
GPIO - Libraries

• Various libraries available
• WiringPi is a Linux Raspberry Pi library that is mostly compatible with Arduino: http://wiringpi.com
• Also supports serial, SPI, I²C
• e.g.

    pinMode(0, OUTPUT);  // aka BCM_GPIO pin 17
    digitalWrite(0, 1);  // On
    delay(500);          // mS
    digitalWrite(0, 0);  // Off
    delay(500);
Tips

- Tools:
  - Small wire cutters, pliers, strippers
  - Magnifier
  - Temperature controlled soldering station, solder
  - Desoldering tool (braid, pump)
  - Heat gun
Soldering

- Not hard, but requires practice
- Use proper iron and solder
- It is possible to hand solder the larger SMT parts
- Can even do reflow using toaster oven and controller
- Lots of good YouTube videos
- Recommend the "Soldering is Easy" comic book:
Test Equipment:

- DMM (very inexpensive)
- DC power supply, e.g. +/-5V, +/-12V, 0-30V
- Logic probe
- FTDI friend
- Bus Pirate
- Oscilloscope (analogue or digital, wide price range)
Tips

• Parts:
  • resistors
  • capacitors

• Miscellaneous:
  • hookup wire (solid for solderless breadboards)
  • clip leads (including small ones for IC pins)
  • DIP clips
  • good collection of cables and adaptors (USB, serial, header connectors)
  • proto boards (wireless breadboard)
  • SD/microSD cards, adaptors
Tips

Getting Started

• Buy a low-cost board like Raspberry Pi, BeagleBoard or Arduino and spend some time with it.
• Start with a flashing LED and then progress to more complex work.
Hardware Construction Methods

- Breadboards
- Protoboards
- Wirewrapping
- Veroboard
- Ugly style, Manhattan
- PCBs
- SMT versus through-hole
Gotchas

• Device power: 3.3V versus 5V (or less). Beware of cheap power supplies.
• Serial ports: RS-232 versus TTL (or other) levels. DTE/DCE, hardware handshaking, different connectors.
• USB: Different connectors. Some USB ports are for power only. Host versus device. Power capability (don't exceed). hubs.
• ESD/static
References - Websites

- EEVBlog (YouTube, eevblog.com)
- Hack A Day (hackaday.com)
- Wikipedia has articles on most buses, protocols, etc.
References

References - Books

• Make: Electronics
• ARRL Handbook for Radio Communications
• Hacking the XBox, Andrew "Bunny" Huang
• Make magazine
References - Suppliers

- AdaFruit (adafruit.com)
- Amazon (amazon.com)
- Digi-key (digikey.com)
- Farnell/Element 14 (farnell.com)
- Jameco (jameco.com)
- Maker Shed (makershed.com)
- Mouser (mouser.com)
- SparkFun (sparkfun.com)
• Thank you for attending
• Questions?