mbed Application Shield Peripherals

128x32 Graphics LCD, SPI Interface	**Newhaven C12832A1Z** MOSI: D11 (PA7) nRESET: D12 (PA6) SCK: D13 (PA5) A0: D7 (PA8) nCS: D10 (PB6)
3-Axis Accelerometer, I2C Interface	**Freescale MMA7660** SCL: SCL (PB8) SDA: SDA (PB9) Address: 0x98
Temperature Sensor, I2C Interface	**LM75B** SCL: SCL (PB8) SDA: SDA (PB9) Address: 0x90
5-Way Navigation Switch, GPIO Interface	**ALPS SKRHADE010** Up: A2 (PA4) Down: A3 (PB0) Left: A4 (PC1) Right: A5 (PC0) Select: D4 (PB5)
Potentiometers, ADC Interface	**Iskra PNZ10ZA, 10K** Pot 1 (left): A0 (PA0) Pot 2 (right): A1 (PA1)
RGB LED, GPIO/PWM Interface	**Cree CLV1A-FKB** Red: D5 (PB4) Green: D9 (PC7) Blue: D8 (PA9)
Speaker, GPIO/PWM Interface	**Multicomp MCSMT-8030B-3717** Speaker (+): D6 (PB10)
Xbee Socket, UART Interface	RXD: D1 (PA2) TXD: D0 (PA3) nReset: D3 (PB3) Status: D2 (PA10)

Hardware and Software Development Processes

When developing new hardware, there are certain milestones that require you to freeze the design – printed circuit boards (PCBs) must be fabricated, components must be ordered, and the components must be assembled onto the PCB (resulting in a printed circuit assembly, or PCA). Once the PCB is released for fabrication ("Goes out to fab") it is difficult to make changes to the circuitry. It is relatively easy to change the value of components at that point, providing the new components will fit onto the PCB ("Have the same footprint as the old components"). Once components are assembled onto the PCB, it is more difficult to make changes, but not impossible. Components can be swapped out ("R&R – remove and replace") and modifications to the copper traces on the PCB can sometimes be made ("Cuts and jumps").

Because hardware is not very easily modified, hardware engineers are used to fairly rigorous design reviews leading up to the PCB going out to fab. There is a rule of thumb that says the cost of making a change goes up by 10X at each stage of the development process. That is, it is relatively easy to change the schematic design, somewhat more difficult to change the PCB design (due to physical constraints that may drive other aspects of the design, such as the enclosure), more expensive to scrap a batch of PCBs, even more expensive to scrap a batch of PCAs, and very expensive to recall finished products from the field and replace them. There are other intangible costs that escalate along the way. Up until the product is released, changes are internal to the company developing the product, but if they delay the product release, market share may be lost to competitors. Once the product is released, flaws in the product reduce consumer confidence and may lead to negative reviews and a bad reputation.

Hardware's characteristic of being hard to change has resulted in rigorous development processes and documentation:

- Development plan
- User requirements and features
- Product specifications (functionality, shape, size, weight, cost, power, user interface)
- Risk analysis
- Design verification test plan
- Design reviews
- Design for manufacturability (DFM) review
- Design verification testing
- Usability testing
- Product validation testing
- Certification testing – safety, environmental, radiated emissions, susceptibility
- Highly accelerated life testing (HALT)
- Packaging design
- Documentation control
- Formal release to production
- Manufacturing testing
- Engineering change orders (ECOs)

For an extreme example of the need for rigor in hardware development, consider the oldest form of engineering, civil engineering. It is really hard to make changes to a bridge or a building after it has been built and mistakes can cause fatalities and serious injuries. That's why civil engineering has developed standards for all manner of design and construction and safety.

Unlike hardware, software is very easy to change – it has the word "soft" in it and hardware has the word "hard" in it! Software is "malleable" or "ethereal" – it can be changed with a few keystrokes and no one will be the wiser. It is also fun – you can jump right in and start writing code and the world is your sandbox or playground.

However, an undisciplined approach to software development has its costs:

- Difficult to quantify progress
- Difficult to manage
- Difficult to plan and schedule
- The result may be difficult to maintain ("Job security" – not if the product fails)
- The result may not be what customers want
- The result may not be easy to update in the field
- The result may not be reliable ("Buggy")

It is a common misconception that applying discipline to the software development process negatively affects one's creativity. The reality is that everyone has to work under constraints – time, cost, resources, office space, computer power, monitor size, number of monitors, multiple projects, personal activities, family obligations, laws, ethics, reputation, intellect, education, training, and so on. It can be argued that applying discipline to software development, which adds constraints, requires more creativity than the free-for-all approach.

Software's "softness" is both a blessing and a curse. It is fun! You can sit down and start coding and get a result very quickly – it's almost magic, creating something from nothing! In the end, however, the result must be robust, it must meet the customers' needs, it must be completed on a schedule, it must fit within the constraints of the processor, and it must be structured and commented well to accommodate future changes.

A good software development process should allow some experimentation within a disciplined and managed structure. It is important to try various approaches and "push the envelope" on functionality and performance – that can lead to unforeseen breakthroughs and product improvements. To compare software development to artistic painting, an artist can choose from a variety of paints and watercolors and inks and charcoals and canvasses and papers and needs to experiment with those to understand their natures and what can be done with them. At the end of the day, however, an artist needs to plan what they wish to render in order to get what they set out to create.

Likewise, hardware and software developers need to plan for the end result and then use a combination of creativity and discipline to achieve the desired results under a set of constraints. One of the most important documents that will lead to success is the specification.

Final Project Specification Development

We will develop a specification in class for the Final Project. Capture the requirements below and circle the features required for the Minimum Viable Product (MVP). Discuss how to "shotgun" code to get started on the project.