Super Engineering Bros

Super GameBro

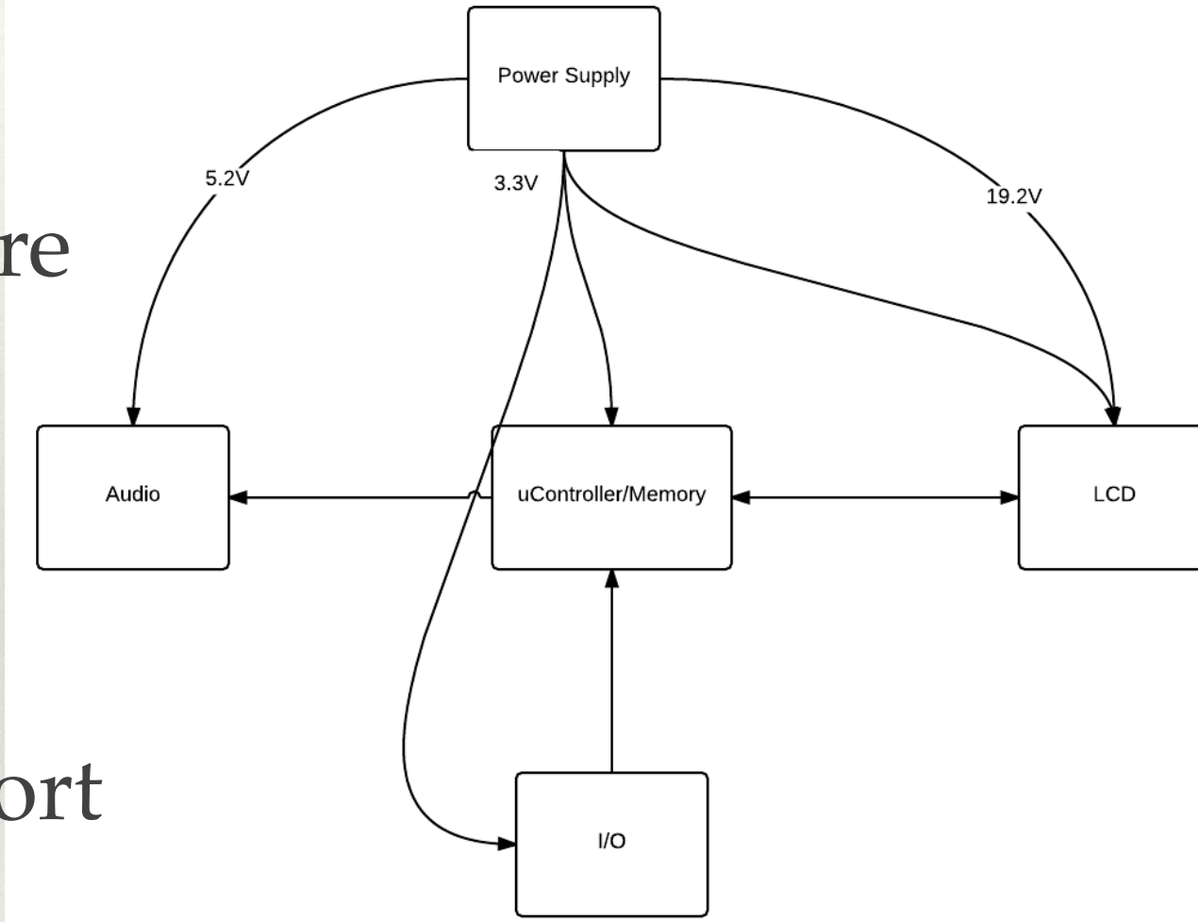
by Riley Duffy and Brandon Pollack

Project Goals

- Create An Open Source Battery Powered Handheld Gaming Console
- Inspiration and Benchmark Drawn from Gameboy Advance
- No OS, i.e. bare-metal. This means no multiprocessing built in.

Hardware Design Overview

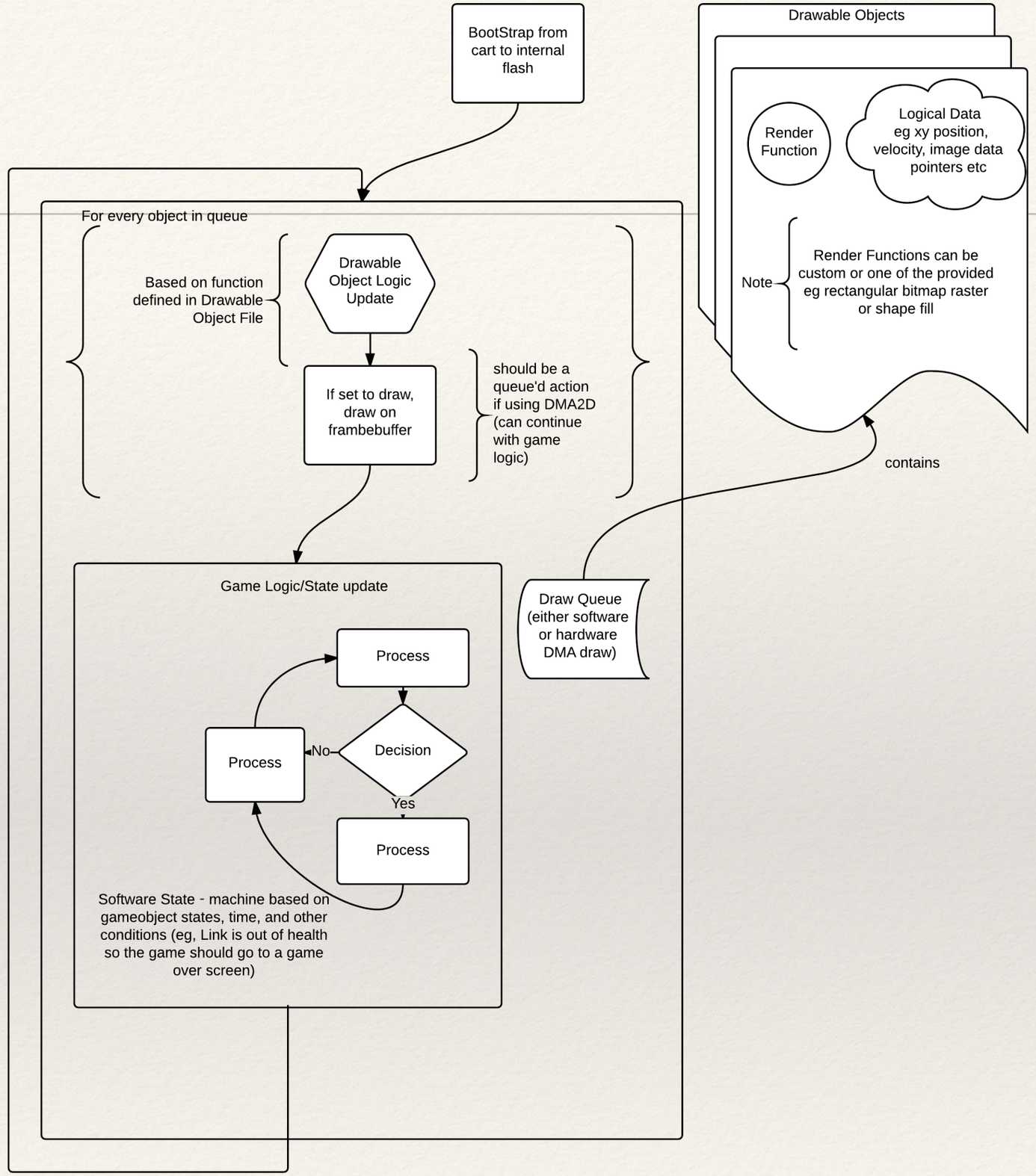
- STM32F4 ARM Cortex M4 core
- 320x240 resolution screen
- Button and Touch control
- Stereo Audio with MP3 support



Software Design Overview

- Comprehensive HAL interface that is faster than standard issue
- Double Buffer page swapping
- Data Structures for Game State and drawable objects
- Rendering queue
- Designed with versatility, ease of use, and generic use in mind

```
import java.sql.*;
import java.awt.*;

/**
 *
 * @author jeff
 */
public class Main {

    public static String AppName = "SQL Mail";
    public static String AppVersion = " 0.0.1 ";
    public static String AppAuthor = "Jeffrey Cobb";
    public static String AppDate = "August 8th, 2007";
    public static String AppPath = System.getProperty("user.dir");
    public static String AppDriver = "smallsql.database.SSDriver";
    public static String AppDBHeader = "jdbc:smallsql:";
    public static String AppDBPath = AppPath + "/sqlmail";
    public static String AppPreferences = AppPath + "/sqlmail_prefs";
    /** Creates a new instance of Main */
    public Main() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception {
        // TODO code application logic here

        boolean bDBConnect = false;
        int result = 0;
        frmMain SQLMailForm = new frmMain();
        System.out.println("\r\n" + AppName + "\r\nVersion" + AppVersion + "\r\nAuthor: " + AppAuthor + "
-- " + AppDate + "\r\n");

        Toolkit tk = Toolkit.getDefaultToolkit();
        Dimension screen = tk.getScreenSize();
        System.out.println(screen.getWidth() + " --- " + screen.getHeight());
```


BootStrap from cart to internal flash
Drawable Objects
Render Function
Logical Data eg xy position, velocity, image data pointers etc
Note
Render Functions can be custom or one of the provided eg rectangular bitmap raster or shape fill
contains
For every object in queue
Based on function defined in Drawable Object File
Drawable Object Logic Update
If set to draw, draw on frambebuffer
should be a queue'd action if using DMA2D (can continue with game logic)
Draw Queue (either software or hardware DMA draw)
Game Logic/State update
Process
Decision
No
Process
Yes
Process
Software State - machine based on gameobject states, time, and other conditions (eg, Link is out of health so the game should go to a game over screen)

CPU Design Choice

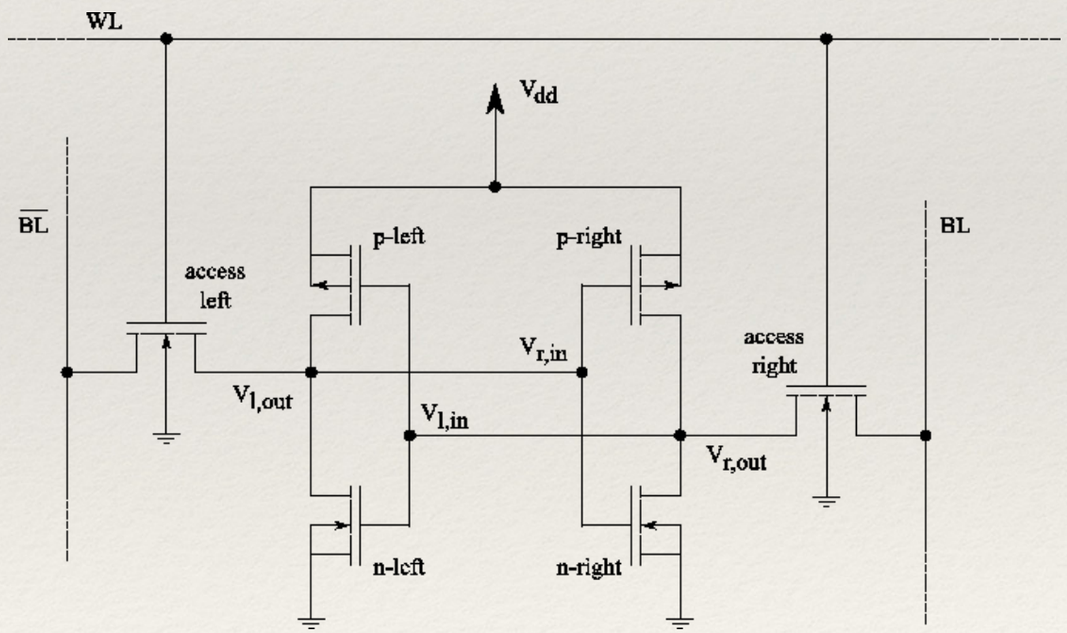
- Many options were considered
  - PIC32/MIPS core — Brandon is very familiar with MIPS architecture
  - PIC24 —build in LCD controller, simple μController architecture
  - Atmel ARM cortex A — very powerful

LCD Design Choice

- Originally desired resolution equivalent to PSP
  - Too much memory
- Went with 320x240
  - Higher resolution than GBA
  - Half the memory required

Memory Choices

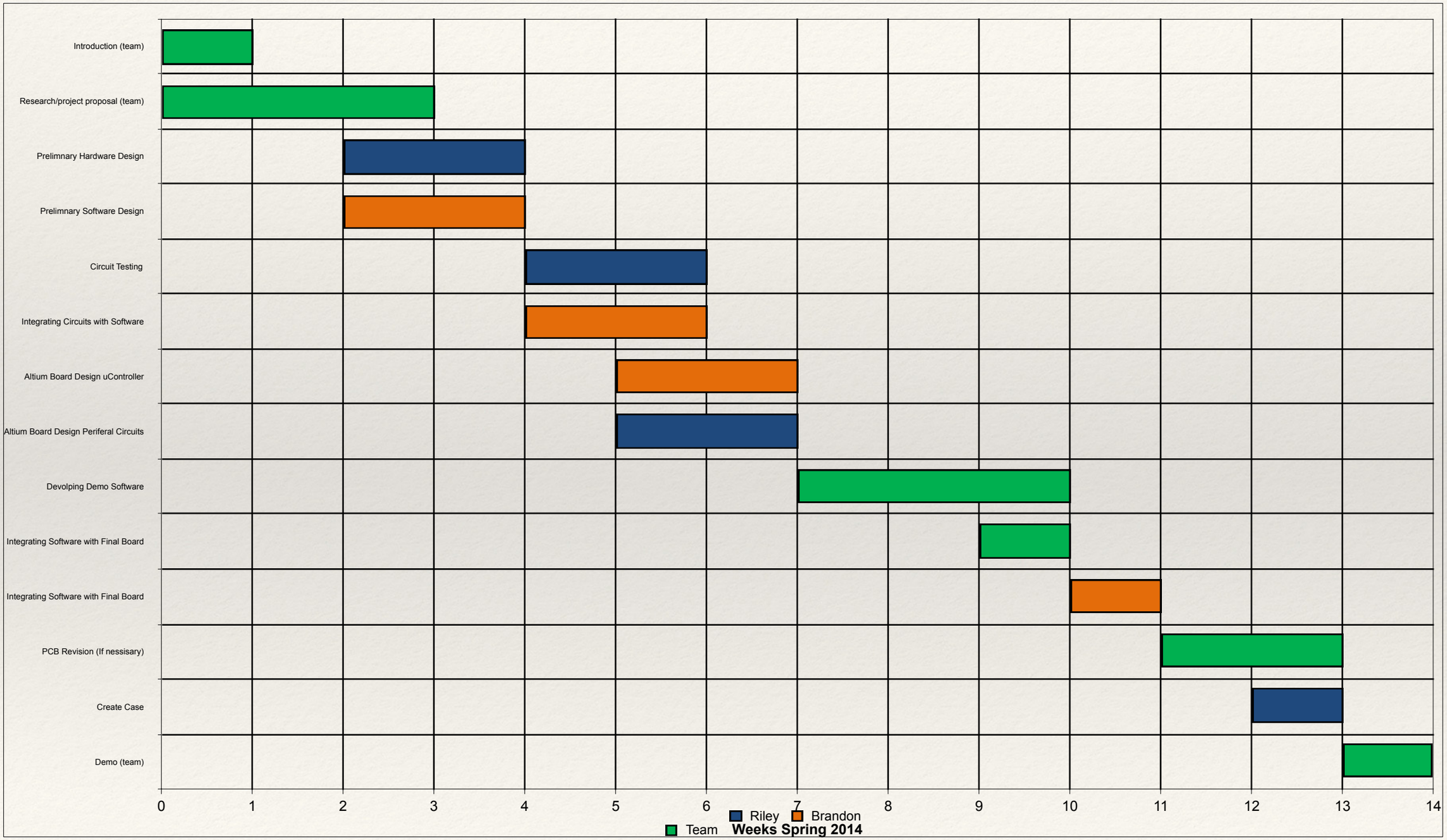
- SRAM is necessary due to
  - speed, ease of use, energy savings
- Space Needed for
  - Background Scrolling
  - Page Swapping
  - Extra Design Space



NOR FLASH

- Random Access
- Faster Reads
- Easily Writable
  - Runtime
  - Program Time
- Modern

Gantt Chart


Introduction (team)
Research/project proposal (team)
Prelimnary Hardware Design
Prelimnary Software Design
Circuit Testing
Integrating Circuits with Software
Altium Board Design uController
Altium Board Design Periferal Circuits
Devolping Demo Software
Integrating Software with Final Board
Integrating Software with Final Board
PCB Revision (If nessisary)
Create Case
Demo (team)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Riley
Brandon
Team
Weeks Spring 2014

Questions?