EEL 4914 Senior Design

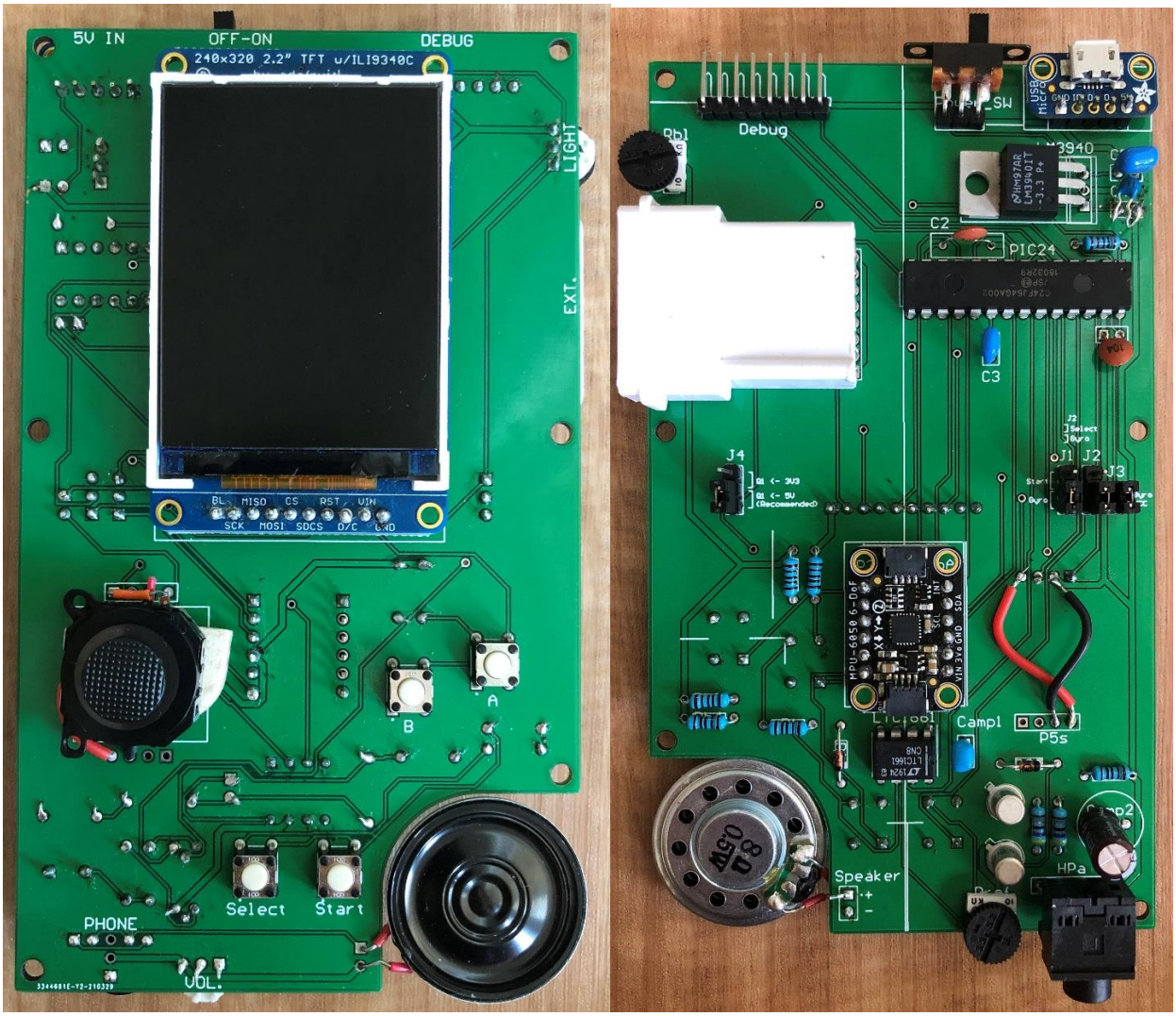
Final Design Report

Handheld Game Console

Group Name: Eric Di Gioia

Student Name: Eric Di Gioia [[redacted]@ufl.edu]

Consulting Engineer: [redacted]


5V IN
OFF-ON
DEBUG
240x320 2.2" TFT w/ILI9340C
LIGHT
EXT.
BL MISO CS RST VIN
SCK MOSI SDCS D/C GND
A
B
Select
Start
PHONE
VOL
Debug
PIC24
C2
C3
J4
J1 J2 J3
MPU-6050 6-DoF
Camp1
P5s
Speaker
HPa


SELECT
START
B
A

Table of Contents

Project Summary (Abstract)

The application domain of a handheld video game console should require no explanation. The product is designed to entertain, so the 'solution' this engineering project is designed to solve would be boredom. In addition, video games in general are another medium as are books, movies, and television, so the purpose of the software is decided by its author. Video games like all other media can not only entertain but also teach, invoke thought, and so on. This handheld game console that whose design is explored in this document would be most akin to earlier handheld devices from companies like Nintendo (Game Boy, 1989), Sega (Game Gear, 1990), and Atari (Lynx, 1990). While these particular handhelds mentioned are dated in their technology, the underlying designs and technical concepts for handheld consoles were established here and remain true. Newer generations of handhelds with some notable exceptions are implementations of the same core design with higher spec hardware (ex. 1989 Game Boy vs 1998 Game Boy Color. The Color featured virtually the same design but with a color screen, more console RAM, and slightly faster processor).

This project consists of designing and building a simple portable game console that features a color screen, mono sound output and sound engine, and of course, controls on the console itself. In addition, the console features a micro SD card slot that can be used in the future to load new software as opposed to being limited to the built-in program (akin to the differences between the Nintendo Game & Watch and Game Boy families of handheld consoles). There is also a controller port to allow for the use of an external controller to replace the on-board controls and a gyroscope/accelerometer.

Project Features/Objectives

The design objectives are to develop this console and have a fun and functional end product as a proof of design. The features implemented in this portable game console are listed below.

- Color graphics display (TFT LCD with adjustable backlight)
- On-board digital and analog controls (digital buttons and analog stick)
- Gyroscope/accelerometer for added element of control
- Ext controller port for optional external gamepad
- BIOS menu with system configuration options
- Partial Multilingual support (external software can detect system language)
- Built-in diagnostic/system test software
    - System Sound Test
    - System Controls Test
    - Gamepad Controls Test
    - Screen Writing / Graphics Test
    - Gyroscope / Accelerometer Test
- Mono sound + 3.5mm audio out for headphone use (switching, with volume control)
- Functional SD card slot for future feature expansions / implementations (loading games)

Competitive Products

It is quite well known that the video game industry is very fierce in both the home and portable markets. Current generation game consoles are products of decades of evolution of design concepts and innovation. Of course, given that the portable game console outlined in this report was designed and implemented by an individual university student with rather tight time constraints, this console is not to be compared with consoles of the recent generations but rather to their the earlier predecessors of the 1980s.

This console does, however, feature optional tilt controls, a mechanic that had yet to really be experimented with in portable consoles until the later 1990s constrained to some special cases (for example, game cartridges with built-in accelerometers rather than them being built into the console itself). This console, including both a gyroscope and accelerometer, has much potential in new gimmicks and interesting control mechanics.

Figure 1. A Game Boy Color cartridge featuring a built-in accelerometer for tilt controls (©2000 Nintendo)

Analog thumb sticks are also a feature of this console which did not become mainstream in portable consoles until decades later (for example, with the Sony PSP in 2004/2005 and later with the Nintendo 3DS in 2011). The analog thumb stick in this design

allows for it to be used as both an analog directional stick and an emulation of a digital directional pad, depending on the code.

Finally, the addition of an external controller port has never been mainstream in portable game consoles but has been seen implemented in some pre-release developer kits for ease-of-use in debugging a console whose form factor has not yet been finalized.

Figure 2. A Game Boy Advance development kit with external SFC controller (©pre-2001 Nintendo)

Concept/Technology Selection

The first thing to note regarding the technical aspect of the console is that it does not support hardware-accelerated 3D graphics. This could be implemented in a future hardware revision given a powerful enough microprocessor and/or additional PPU are chosen. At the time of writing the project proposal, it was yet to be determined whether or not an additional picture processing unit will be needed alongside the main microprocessor to drive the graphical display, but I ended up going with just a PIC24 MCU with no additional PPU. Such, some of the attached earlier hardware revisions (Rev 2.1), depict a PPU between the LCD and the CPU. All other revisions depict the MCU driving the LCD by itself. The additional of another processor for this particular application would probably be overkill when a simple external oscillator can be used instead to speed up the SPI data writes to refresh the LCD. An external oscillator is not used in the final design, but would certainly be a welcome addition to any future hardware revisions to allow for less lag in moving sprites on the screen.

The LCD chosen is a small TFT LCD that shared an SPI but with the micro SD card controller located on the same breakout board. This particular display was chosen for both its resolution and the fact that having both the LCD and SD on the same SPI bus allow for the other available SPI bus on the PIC24 to be used with the DAC for sound output. There is also a brightness wheel to adjust the LCD backlighting.

Both digital and analog controls were chosen to be implemented and different control schemes can be used by different software. Some software will only require digital buttons and 4-direction movement but other software may be able to take hefty advantage of the directional analog stick. The software running can choose to read the analog stick as a 4 or 8-way directional pad or as an analog input with varying degrees in each direction. One thing to note about the analog stick is that the use of an analog stick rather than 4 separate directional buttons (a traditional d-pad) uses two fewer MCU pins.

The console also features an accelerometer/gyroscope which includes a built-in temperature sensor. The feature I see to be the most useful for games would be the accelerometer which allows for tilt controls. Tilt controls in a mobile environment can prove to be quite fun if well-executed. At the time of designing the PCB, it was unknown whether or not the accel/gyro would be used in the final design. Because of this, there are three jumpers on the back of the console which are used to enable or disable the gyro. When the gyro in use, however, this disables the functionality of the start and select buttons, and they will therefore always be read in as not pressed.

An external controller port allows for the use of an external controller (optional). The controller port is a 15-pin DIN connector the same as those used on the Nintendo Family Computer/Famicom (the Japanese/Domestic version of the Nintendo Entertainment System/NES). While the controller is connected, the on-board controls are automatically disabled. The controller read code allows for the ability to hotswap controllers and the console detects whether or not an external controller is connected. The optional controller allows for more ways to play, depending on end user preference.

The SD card format was chosen for the cartridges as it is simple, easy to implement, and non-proprietary, which is OK for a non-commercial product such as this one. The design of a proprietary cartridge and connector would be beyond the scope of this project. Given the readable AND writeable nature of the SD card, the system SRAM was removed in revision 2.0. Game data could potentially be stored on the SD card itself. The SD-loadable software was the only design goal that has not been fully realized in the final revision. While the SD card slot/controller is fully connected to the MCU via an SPI bus, there is no software implementation of flashing the MCU program memory at runtime. This feature could be added in a future firmware update without the need of a hardware revision.

The console features a debug port for updating the firmware/flashing the MCU via an MPLAB Snap in-circuit debugger.

The BIOS menu offers system diagnostic tools, language options, and the option to boot SD card software. More menu options may be added in later firmware revisions.

The exp/multiplayer port was removed in hardware revision 2.0 to reduce unnecessary digital complexity and to remain within the scope of the project.

The design includes a sound circuit consisting of an SPI-connected DAC IC whose output feeds into a class AB amplifier. The output of the amplifier goes into a switching 3.5mm headphone jack and a small speaker. When the headphones are connected, the console's internal speaker is disabled. The volume can be adjusted with the volume wheel located next to the headphone jack.

Finally, micro USB power was chosen as it is one of the most readily available cables that supply sufficient power (5V that is stepped down to 3.3V).

Figure 3. The console with external controller shown

Project Architecture

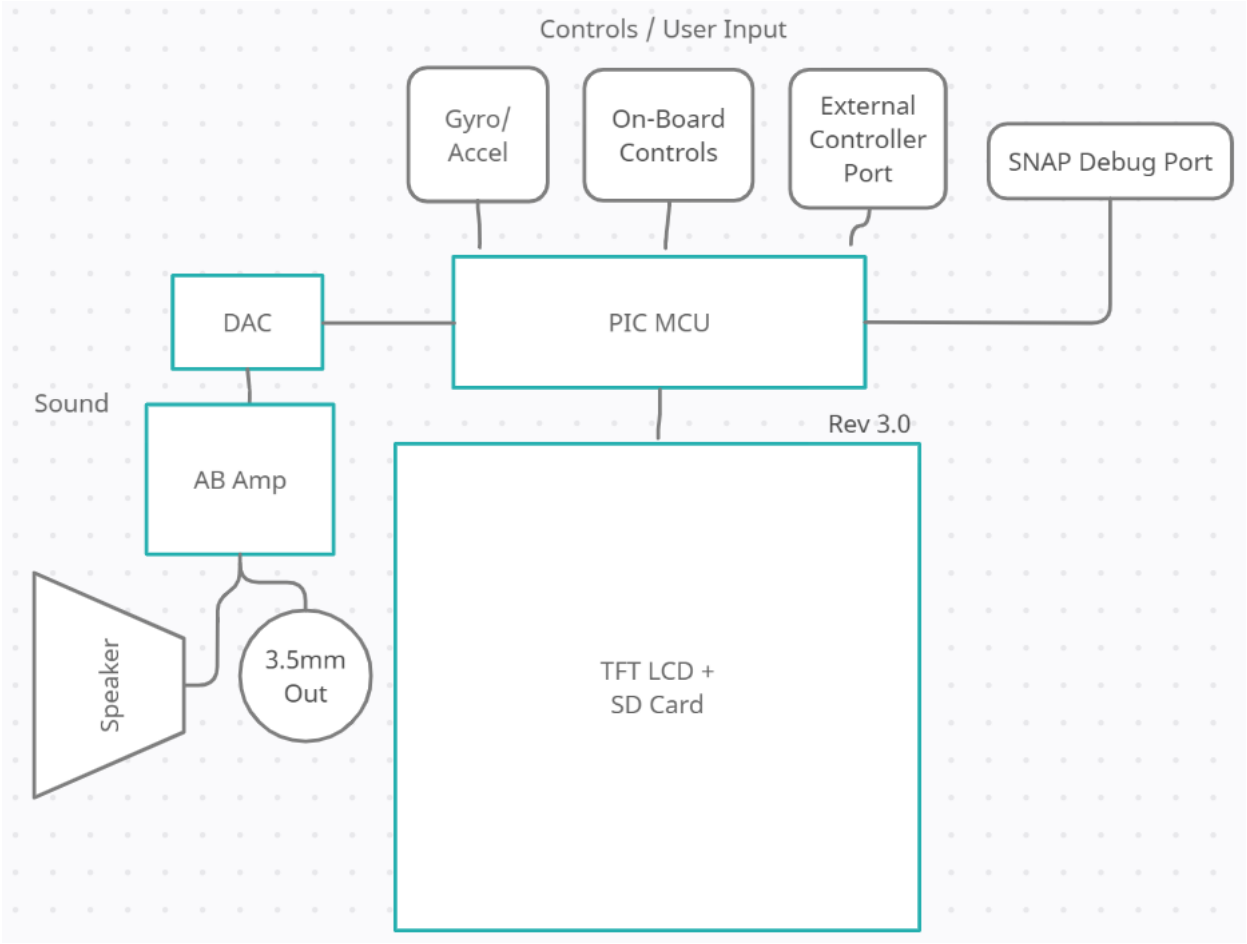


Figure 4. Final (Production) Revision of the design architecture

The PIC24FJ64GA002 is the center of the console to which all peripherals are connected. As shown above, the IL9340 LCD and the SD card slot are both on the same breakout board. These two peripherals share a single SPI bus and have separate chip select (CS) signals. This allows for the LTC1661 DAC to also be connected to the second (of 2 available) SPI modules of the PIC MCU. The output of this DAC is controlled by a potentiometer which, to the end user, is a volume wheel. The DAC feeds into a class AB amp and then into a 3.5mm headphone jack and

speaker. When headphones are connected, the console's internal speaker is disabled and sound is played through the connected headphones instead.

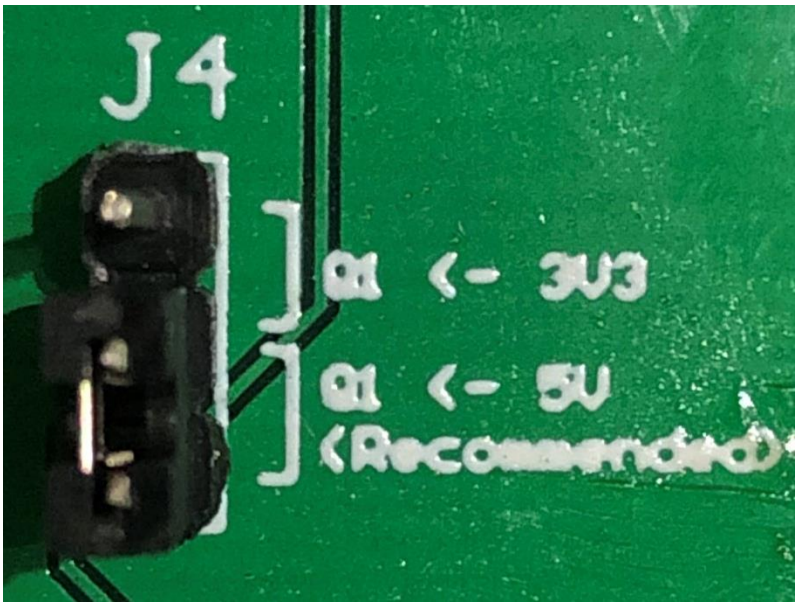


Figure 5. Jumper for the AB amp's voltage input. If 5V is used, it will come directly from the USB input rather than tax the voltage regulator.

The digital buttons are each connected to their own MCU pin and the analog stick is essentially two potentiometers which are connected to two analog input MCU pins to be converted to a digital reading using the MCU's internal ADC. The external controller port only has three connections to the MCU: latch, data, and clock. The button presses from an external gamepad are read into the MCU serially via the data line. Which controls to be reading are a matter handled within the software.

The gyro/accelerometer used is the MPU-6050. This is connected to the MCU via I2C protocol. There is an additional special interrupt pin that is connected to the MCU from the MPU but remains unused. The MPU's connection and functionality is determined by some jumpers on the back of the console which determine whether or not the MCU is supplied power and whether or not the I2C lines are connected to the MCU. The MPU has been enabled for the final design.

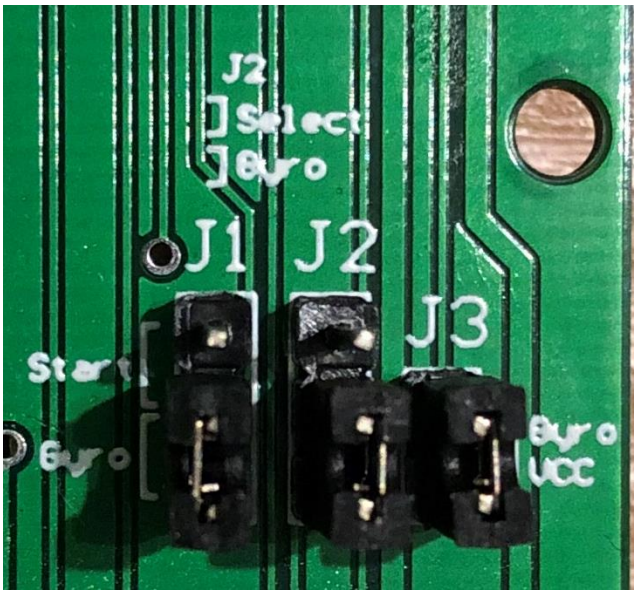


Figure 6. MPU-related jumpers

The power input (not pictured in the above diagram) is a generic micro USB. 5V comes into the console via the USB port and it is then stepped down to 3.3V which is supplied to the entire console. Power supply to the rest of the console is controlled via the power switch.

There is also a debug port for the MPLAB Snap connected to the usual 3 MCU pins.

Hardware/Software Selection

The MCU I ended up going with was the PIC24FJ64GA002, which I believed to have an adequate enough speed for my application while not being overkill. I believed the processor to feature just enough IO for my needs and it was also through-hole, meaning it would be easier for me to solder myself in my home.

Consequently, the IDE used for writing all the code was the MPLAB X IDE, designed for use with the PIC microcontrollers. The code is written completely in C89 standard and the compiler used in this project was the XC-16 compiler also from Microchip.

Design Procedure

In the beginning, I first tried to think of how a game console's architecture is traditionally laid out. I planned all the peripherals and other hardware before choosing an appropriate MCU that is capable of handling all the IO and speed requirements.

I then started the process of pin planning alongside interfacing with some of the more basic peripherals such as the external controller and the SPI DAC in software, after which is began to interface with more complex peripherals like the LCD and accelerometer.

Interfacing with the graphics LCD proved to be possibly the toughest part of the entire design effort, as it involved porting a C++ graphics library from the Arduito to a C89 version for the PIC24. This was also the longest part of the design process and I spent much time reading the IL9340 manual and researching how to read and write to/from generic graphical displays, as I had never worked with one previous to this project. Additionally, I added to the graphics library many functions and C structures to deal with graphics specific to my needs.

Once I was able to communicate with all the peripherals, I started taking the interface code I had written so far and organize it into various functions and modeled the whole software design with an ASM. From there, I simply began adding and editing different states to the design. The firmware ASM has not changed since conception, although the SD card is always detected as not present since it has not been implemented.

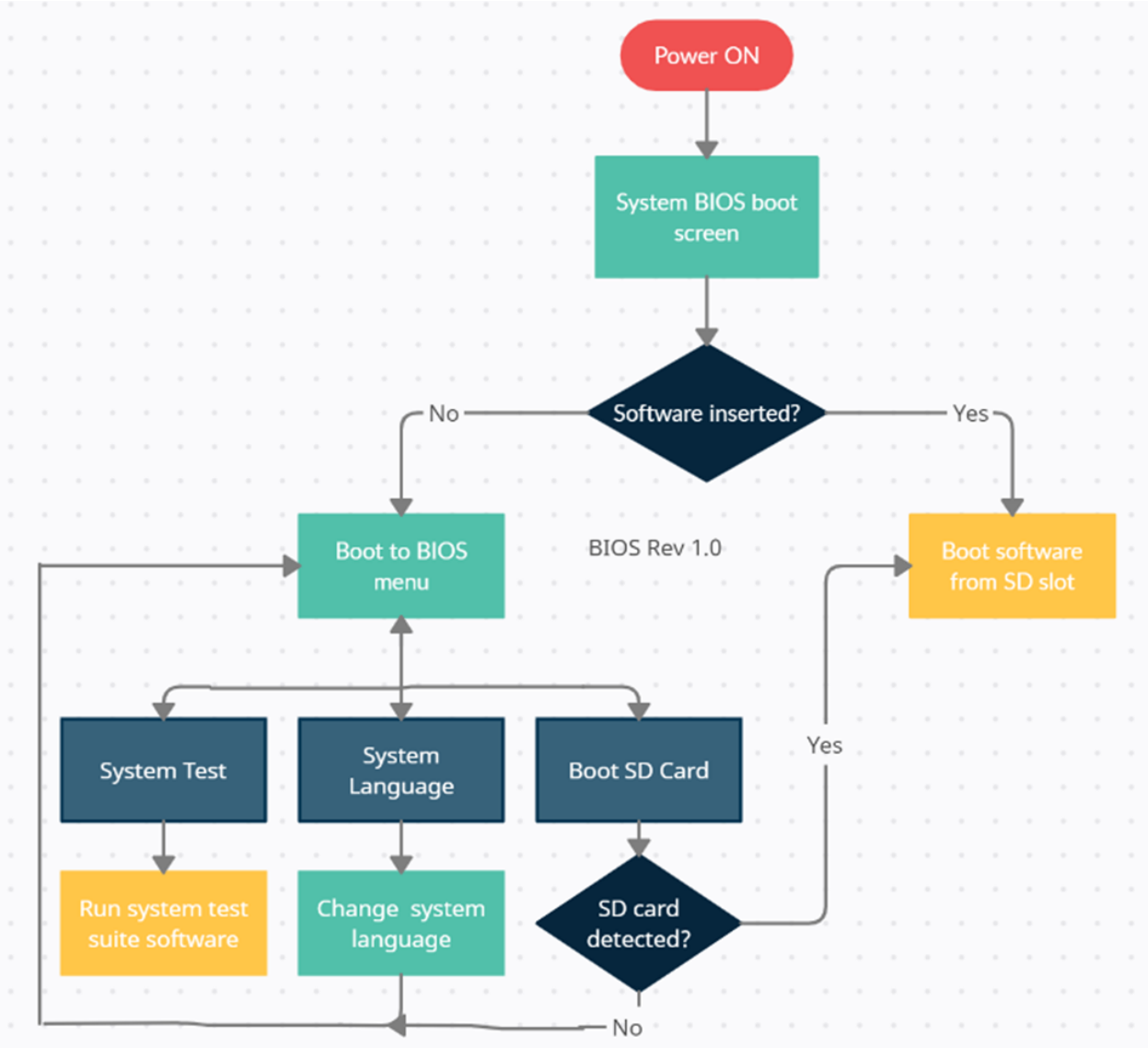


Figure 7. Firmware architecture

Flowcharts and Diagrams

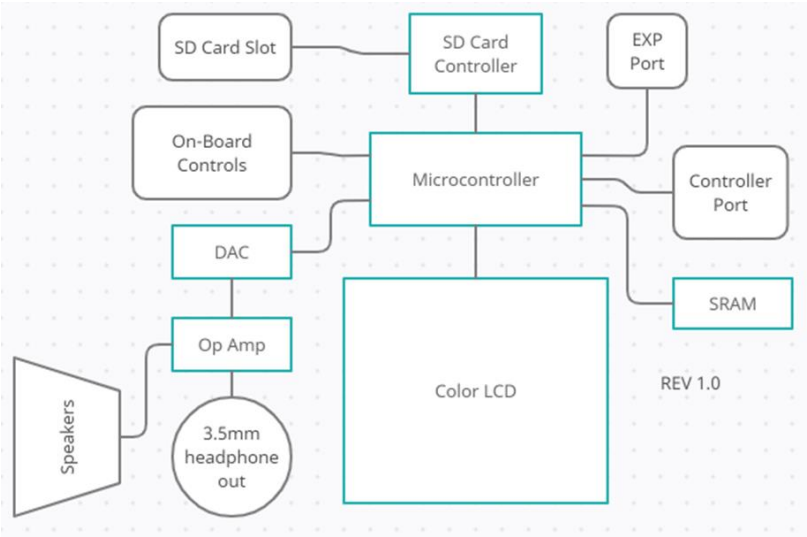


Figure 8. Hardware Block Diagram Rev1.0 (old, included here to show progression)

(Rev 2.0: exp port removed, SRAM removed, analog sound circuit updated, tilt sensor added)

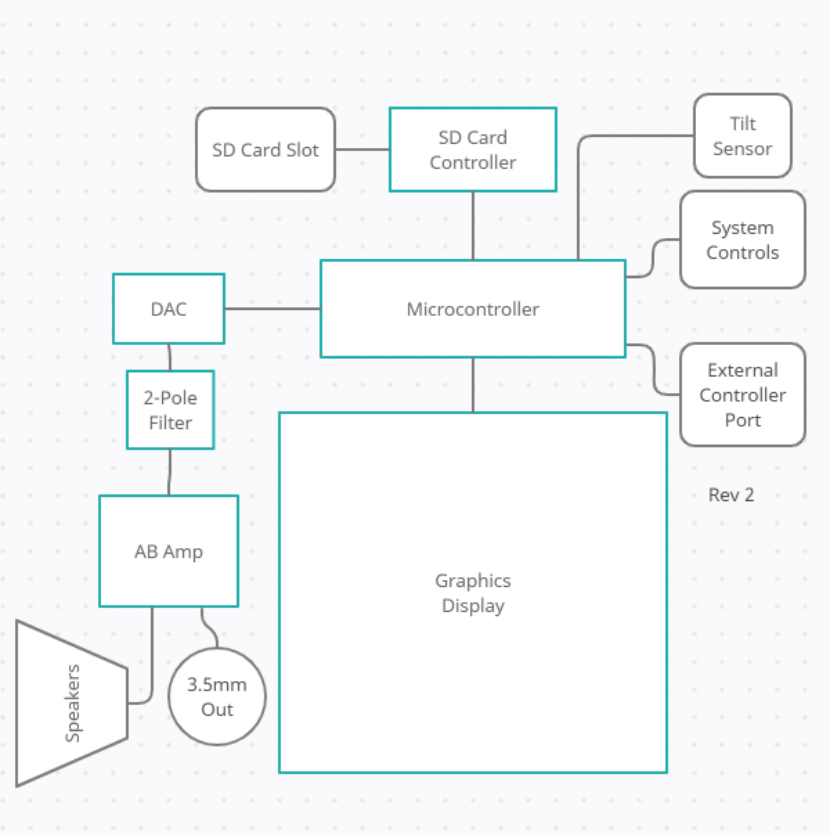


Figure 9. Hardware Block Diagram Rev2.0 (no PPU)

(Rev 2.1: PPU added to drive graphical display)

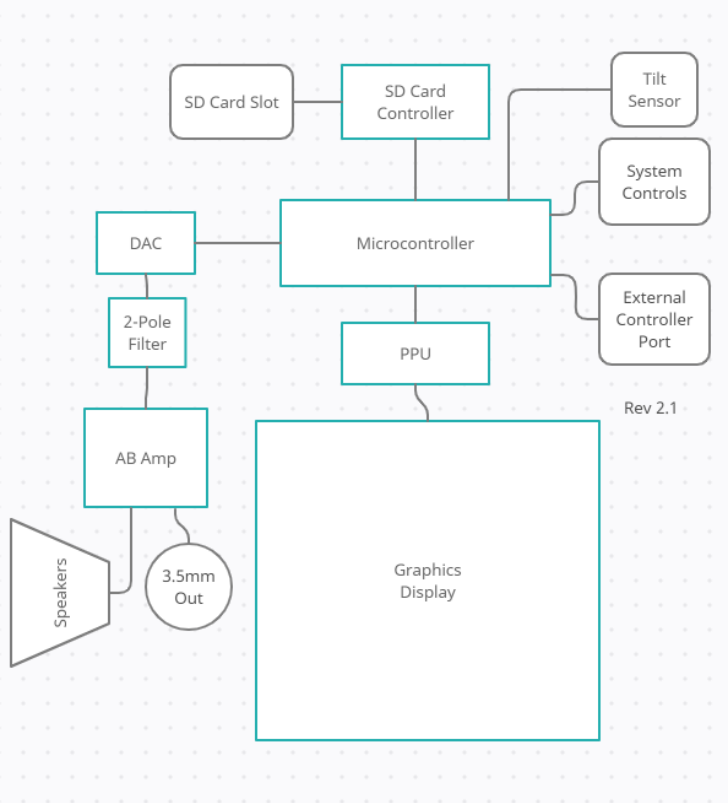


Figure 10. Hardware Block Diagram Rev2.1 (PPU)

(Rev 3.0: Removed PPU and sound filter, integrated SD and LCD)

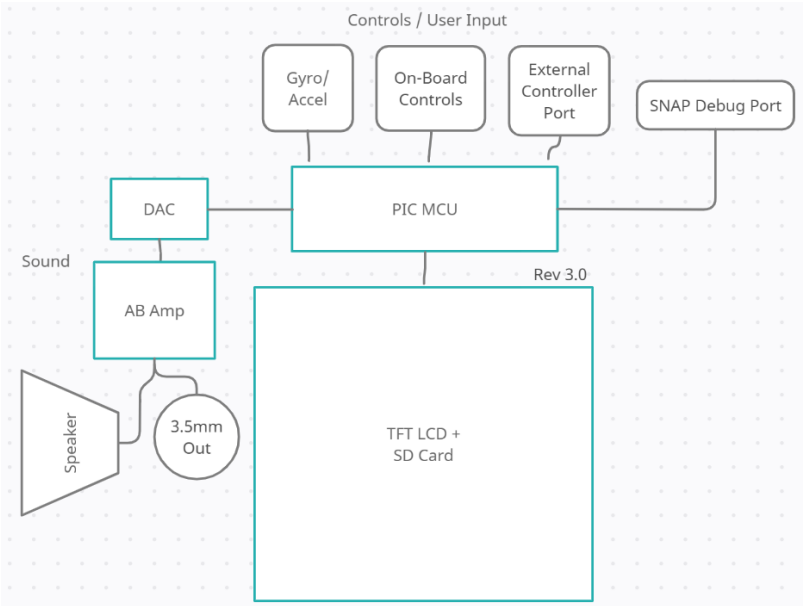


Figure 11. Hardware Block Diagram Rev3.0 (Final/Production)

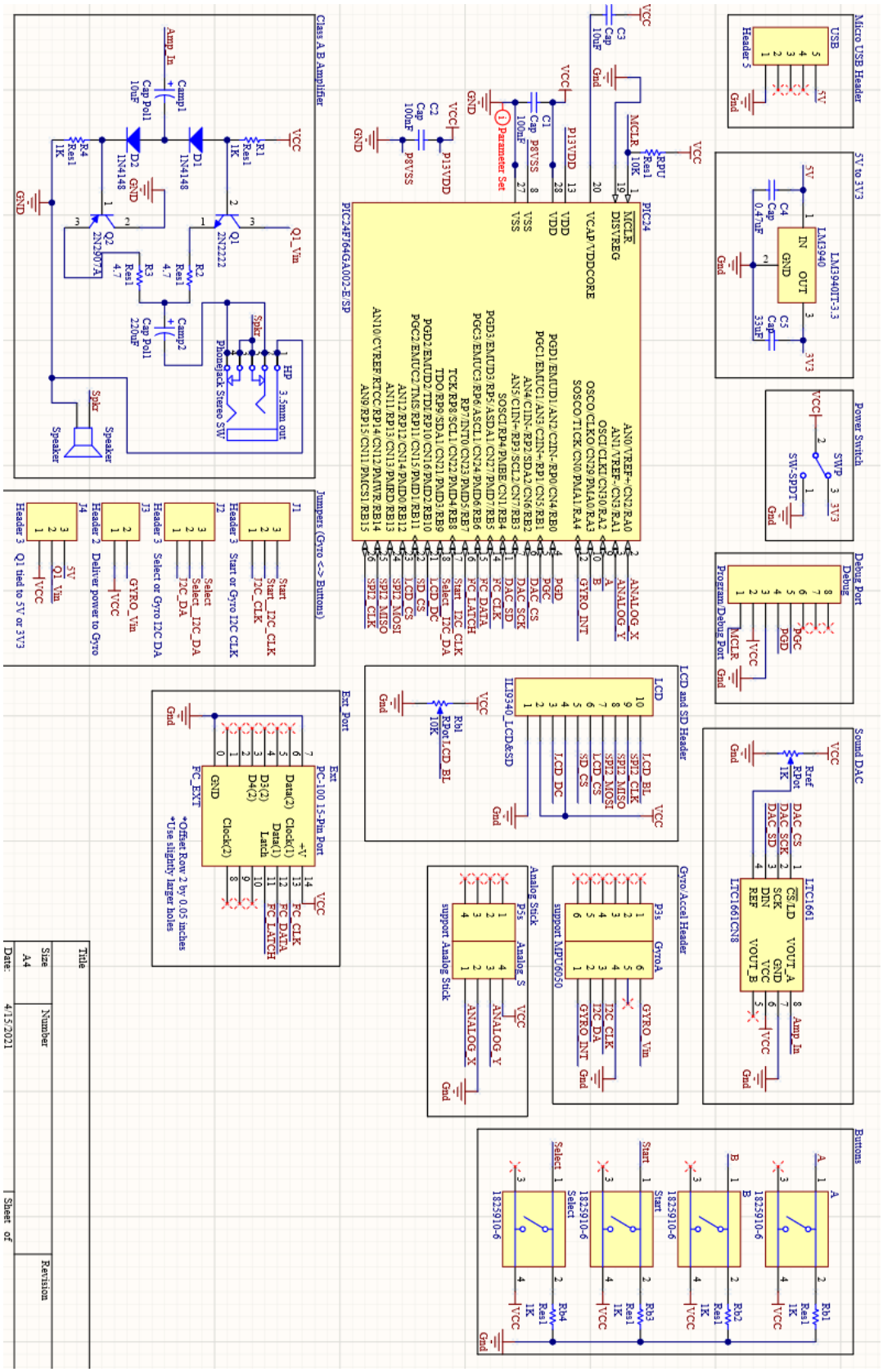


Figure 12. Hardware schematic

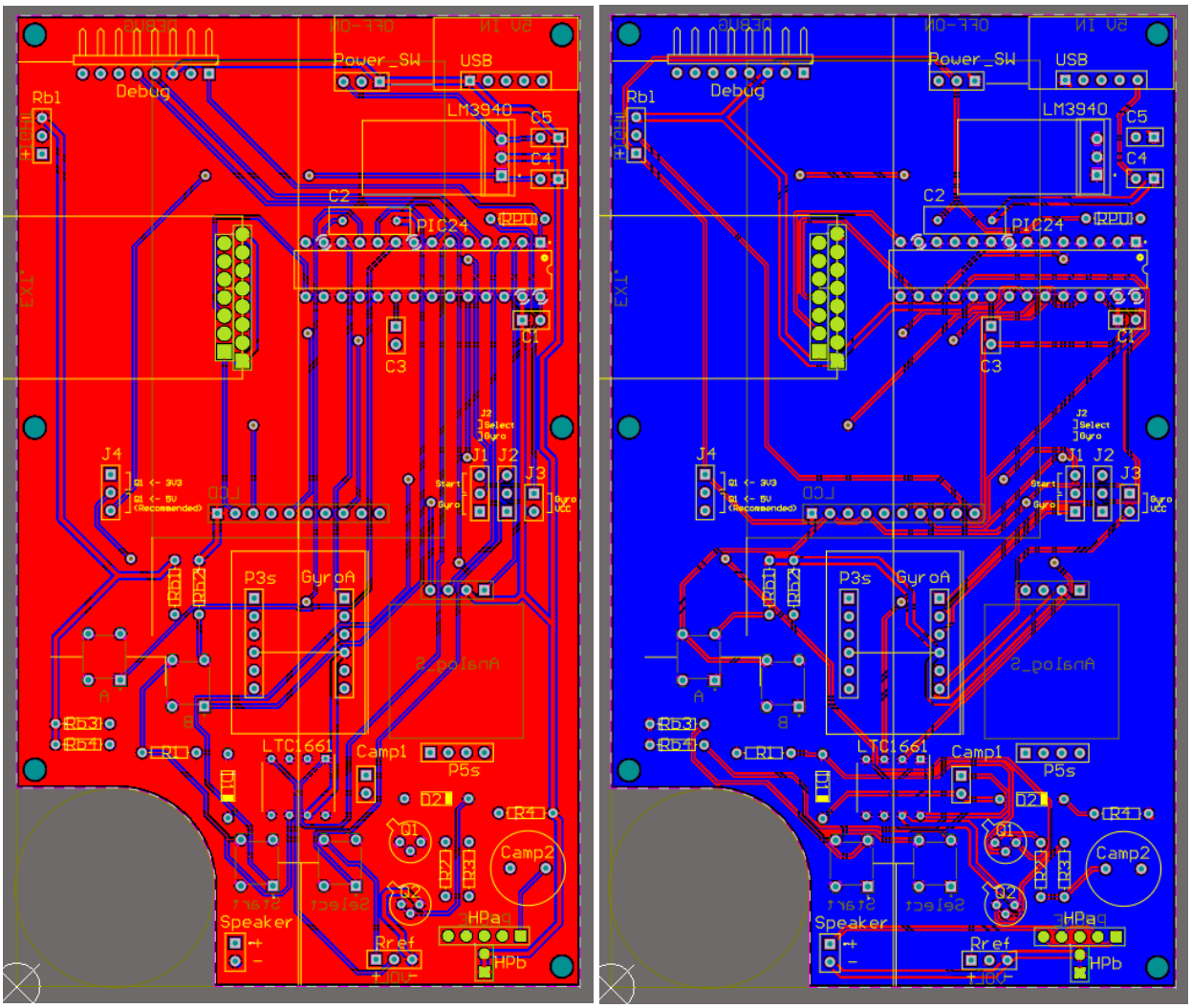


Figure 13. PCB schematic top and bottom layers

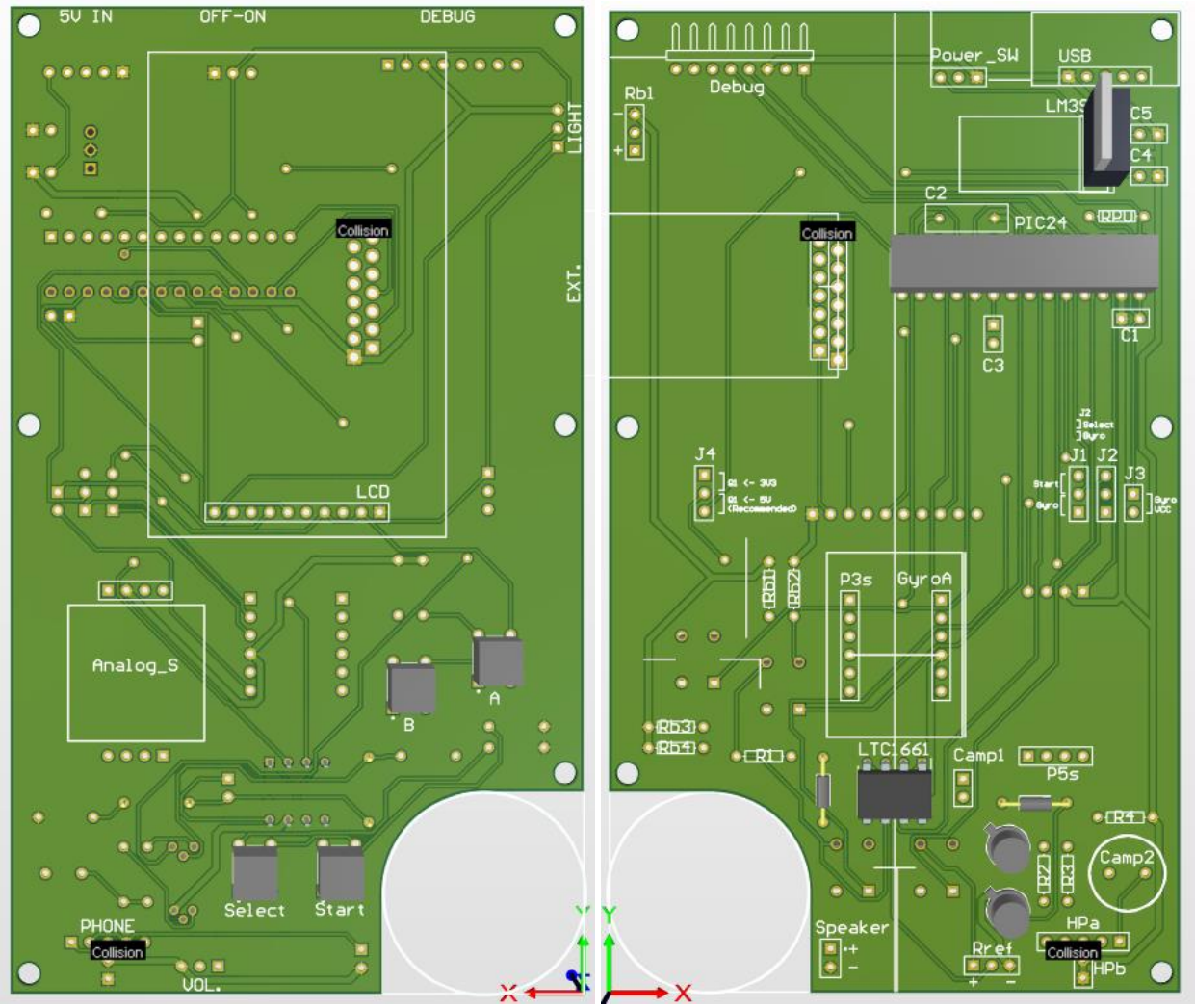


Figure 14. PCB diagram front (left) and back (right)

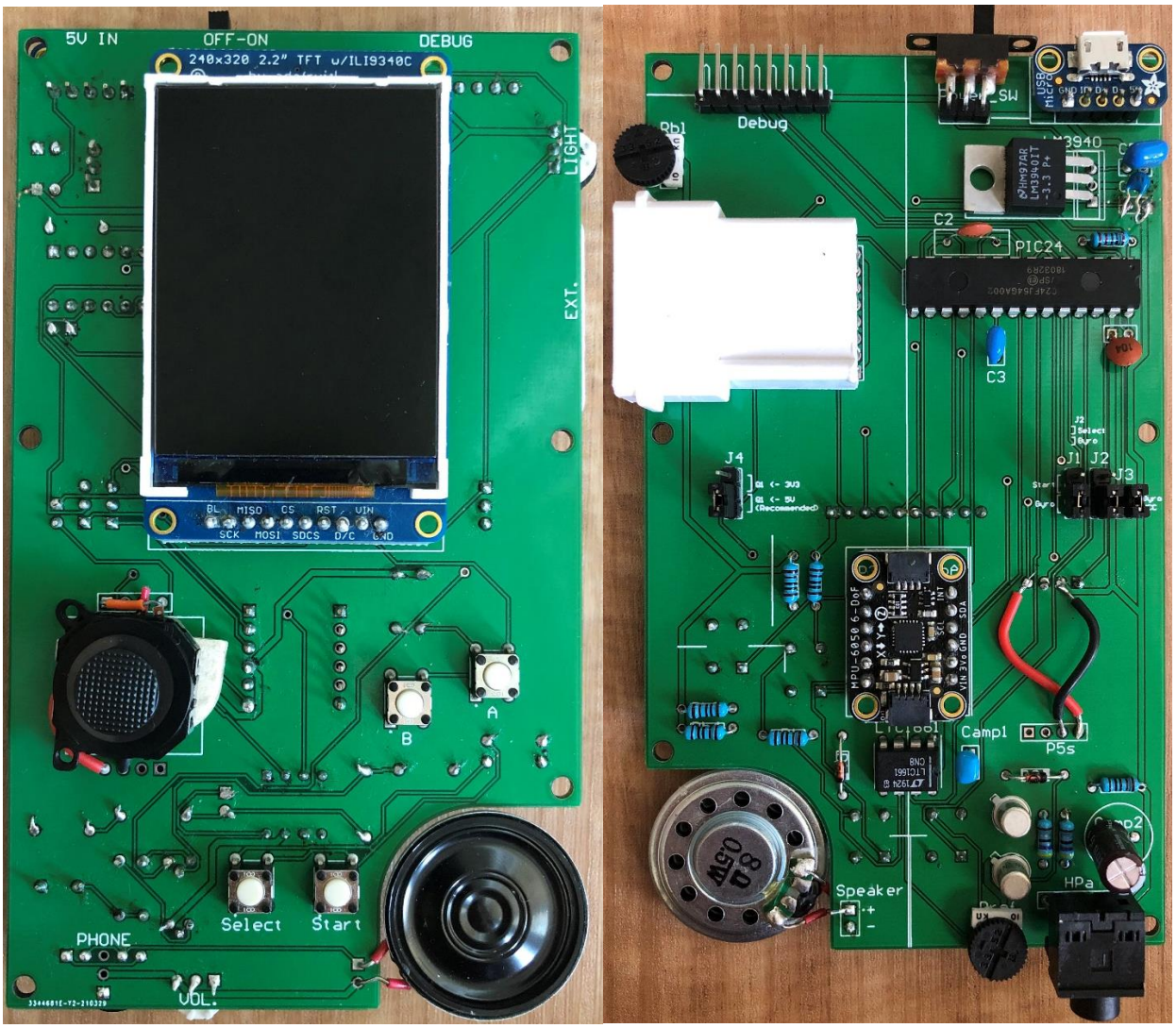


Figure 15. Final product front and back

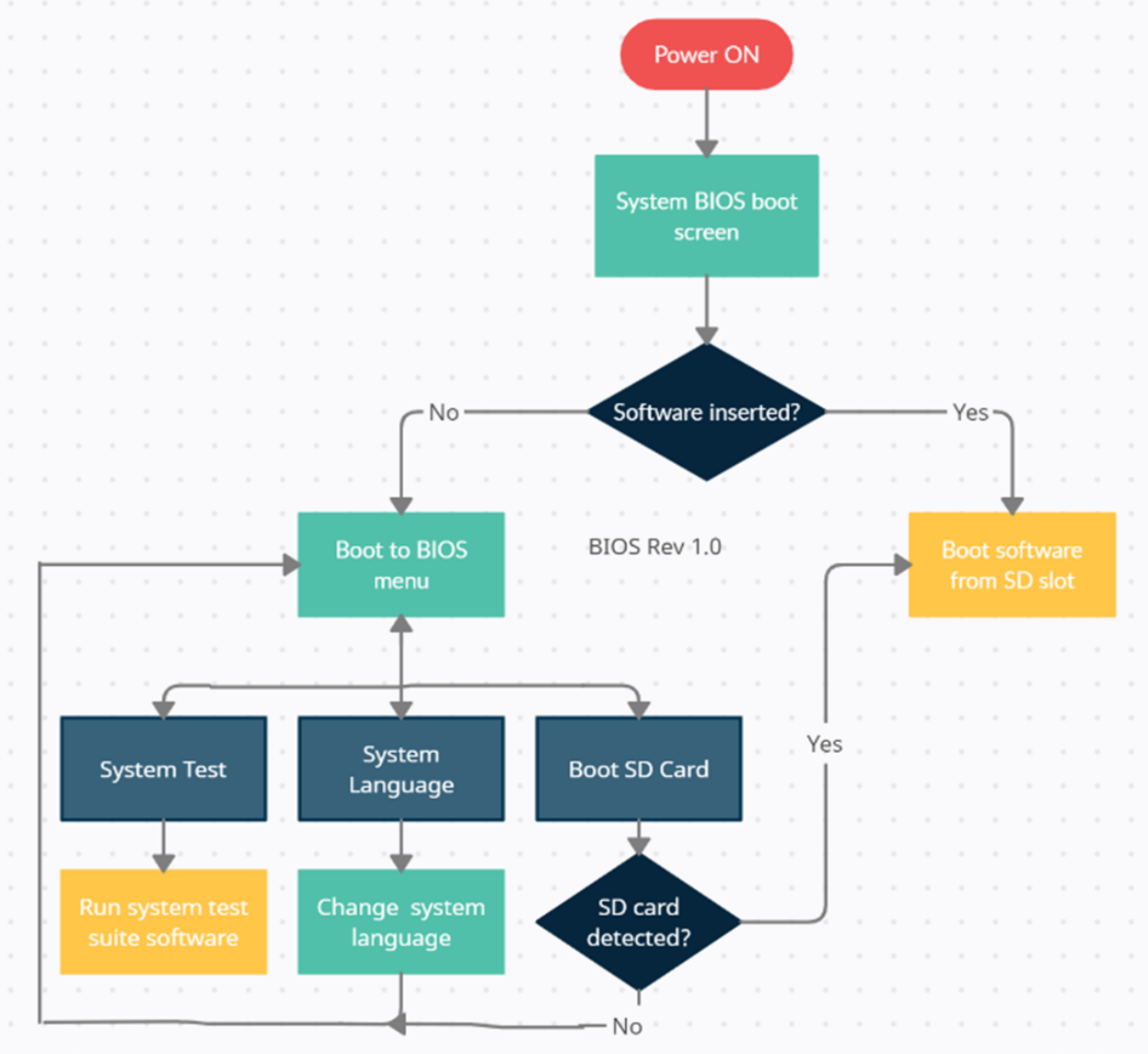


Figure 16. System BIOS Software Diagram (from boot)

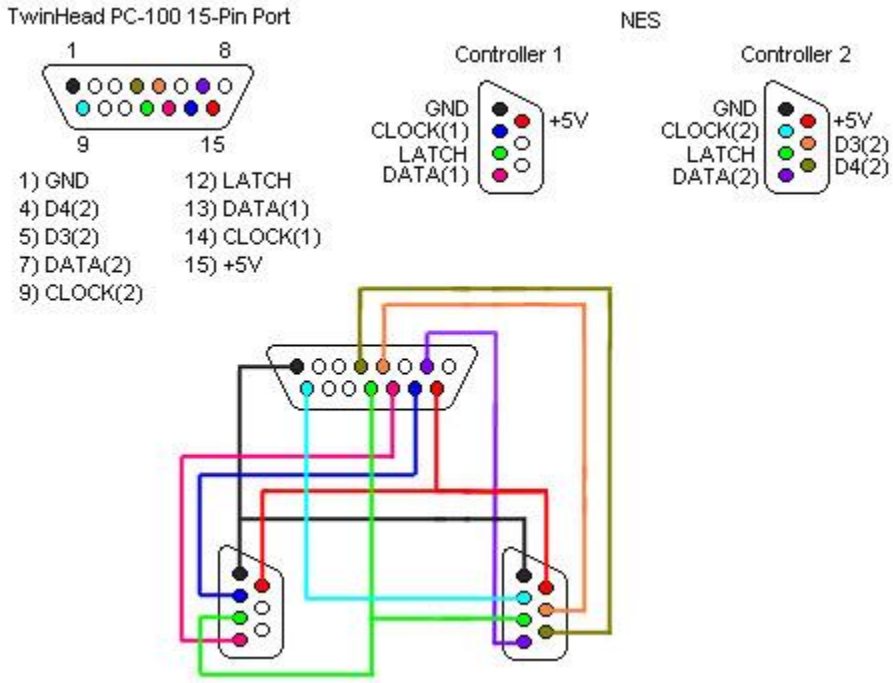


Figure 17. External controller port pinout

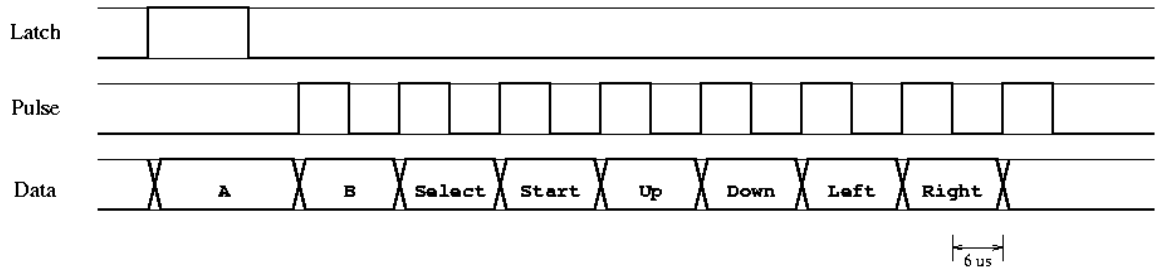


Figure 18. Timing diagram of external controller reading

28-Pin SPDIP, SSOP, SOIC

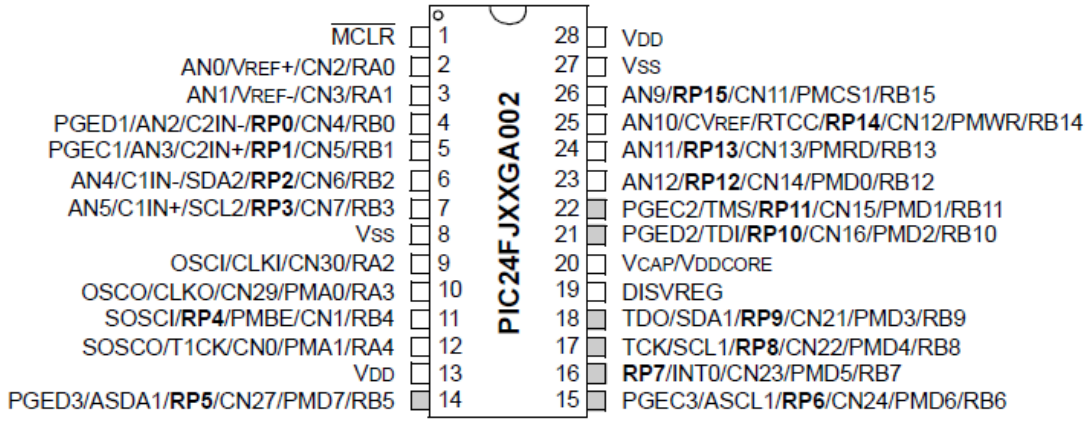


Figure 19. MCU pinout diagram

Obstacles Overcome

The first (and most daunting) obstacle I faced in my design was interfacing with the graphical LDC. I had never before used a graphics display and the only LCD experience I had was with the 1602 non-graphics display. That being said, learning how to draw graphics from zero was quite the undertaking. I first found the IL9340 reference manual (which is 233 pages long) and began to see how the codes were written to the SPI display. The main issue was that there was no graphics library that I could find for use on the PIC microprocessors for these particular displays. After searching for many days, I found someone else who was wanting to do something similar on the Microchip forums and he or she mentioned how they were in the process of rewriting the Arduino C++ library. Unfortunately, he had not provided his complete code, so I decided it was up to me to port the Arduino C++ graphics library provided by Adafruit (who made the LCD breakout board I am using). The C++ code used classes and other functionality exclusive to C++, so I at first thought it might be best to rewrite the classes using structures, but later found that classes were not necessary for my use anyway and began porting the various methods for writing pixels, characters, and other basic geometry. After that, I added many of my own original functions such as structures to act as classes in handling graphics. I gave the structures various attributes/variables such as x and y position, a pointer to the sprite (graphical data), whether or not it is to be drawn, the size, etc.. This was the most time-consuming part of the project, though I gained a lot of knowledge.

Another obstacle I encountered was the issue of available IO on the microprocessor itself. It seems that I had gotten and MCU with just barely enough pins, but if I wanted to implement the accelerometer, I was going to have to find a way to free up a few pins. First, using an analog stick rather than a 4-button digital d-pad allowed me to save 2 pins on the MCU since each of the 2 potentiometer analog readings can be one of 2 directions. However, this was still not enough to allow the use of the accel/gyro without any sacrifices. In the end, I decided to disable the select and start buttons if the gyro/accel is in use. This is accomplished with some jumpers on the back of the console as well as a bit in the code that can be set to

either enable the gyro or disable it. This bit affects how the on-board controls are read. If the bit USE_GYRO is set, during the OBC (on-board control) read code, the select and start buttons will automatically be read in as 0 and the accel/gyro readings are read via I2C protocol. I figured that this would be the best way to have both in the final product, though not at the same time. It allowed to future-proof any design changes that might favor one over another.

The final obstacle I faced was after PCB production when I was populating the board. I had accidentally broken an analog thumbstick so I had to use a replacement thumbstick that I thankfully had. The replacement analog stick fit perfectly in the allotted space on the board, but the issue was that the pinout was slightly different (for example, the ground pin was where the X pot pin needed to be). I was able to resolve the issue by crossing some wires on the back of the board, as can be seen in the PCB photos. As long as the correct signals are going to the correct places, there are no issues.

Additionally, the implementation of the system tests, while initially designed for the end user, ended up helping myself a lot as well in the process of interfacing with various controls schemes, sprite logic, sound composition, gyro/accel reading, etc.

Some more things I would like to change in a future hardware revision would be adding an external oscillator to increase the LCD write speed to reduce lag when large sprites are being moved on the screen, and also the addition of a shift register for all the digital on-board buttons to reduce the number of pins used on the MCU. I would also remove the jumpers and hardwire those connections the way I ended up setting the jumpers in the final product. Finally, I would change the screen out for a similar spec one of a different aspect ratio so that the screen is not so vertical. There is room on the console to add a wider screen, and it would look much more natural this way.

User Manual

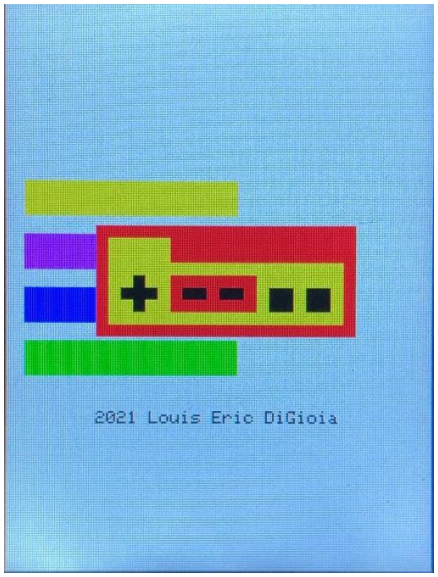


Figure 20. Console Splash Screen

Upon startup, the console will display the splash screen. It will then check whether or not an SD card is inserted.

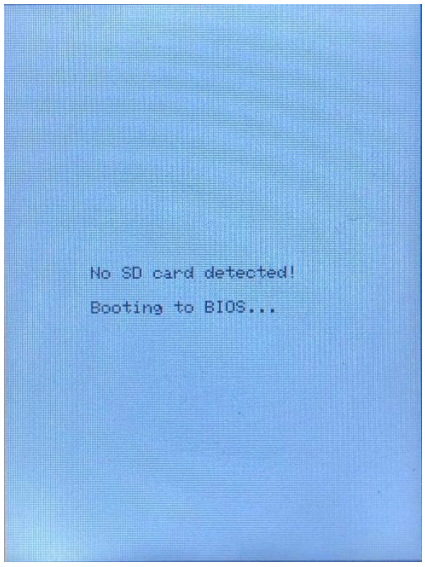


Figure 21. SD card boot screen (not detected)

If the SD card is detected, the console will automatically load the software. If no SD card is detected, it will automatically boot into the BIOS main menu.

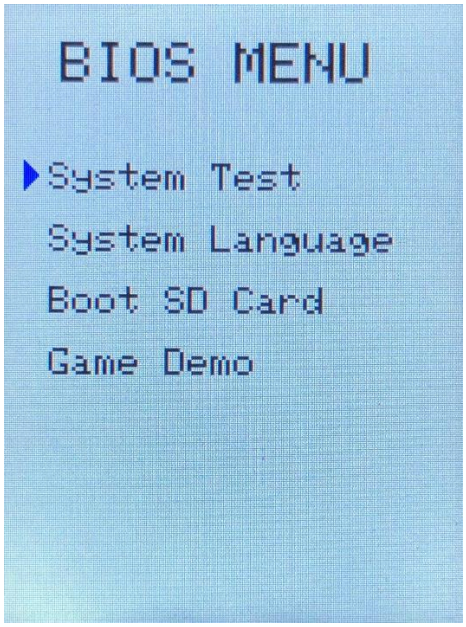


Figure 22. BIOS main menu

The BIOS main menu displays many options.

Figure 23. System language menu

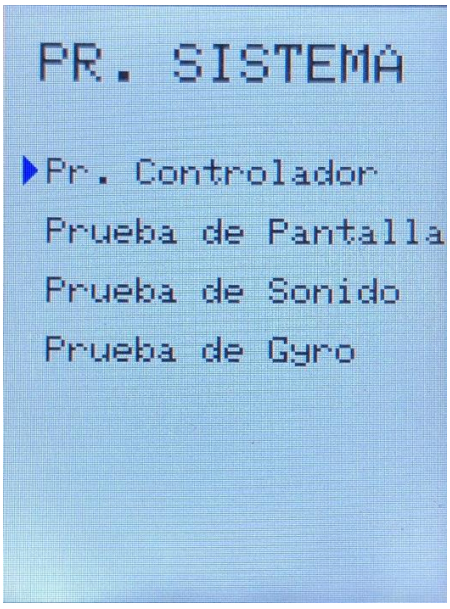


Figure 24. System test menu in Spanish

The language menu lets the user change the system language which can also be detected by software.

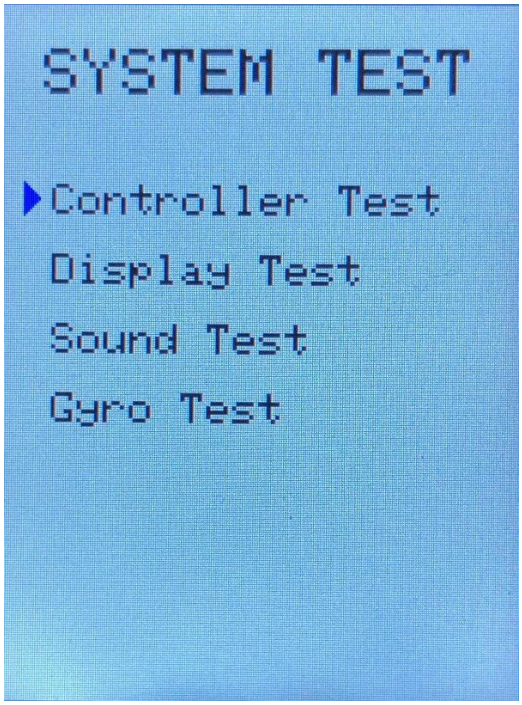


Figure 25. System test menu

The system test menu provides access to various diagnostic tools.

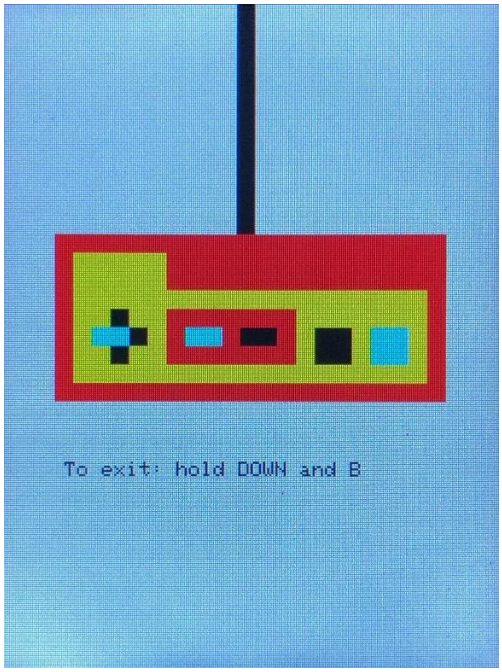


Figure 26. Controller test screen (with Left, Select, and A pressed)

On the controller test screen, the user can test their various input methods. Both an external gamepad and the on-board controls may be tested here.

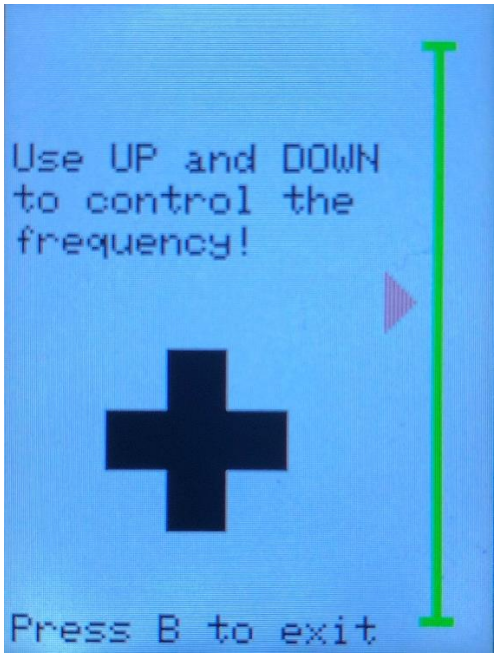


Figure 27. Sound test screen

On the sound test screen, a note is constantly output and the user can use the up and down buttons to change the frequency.

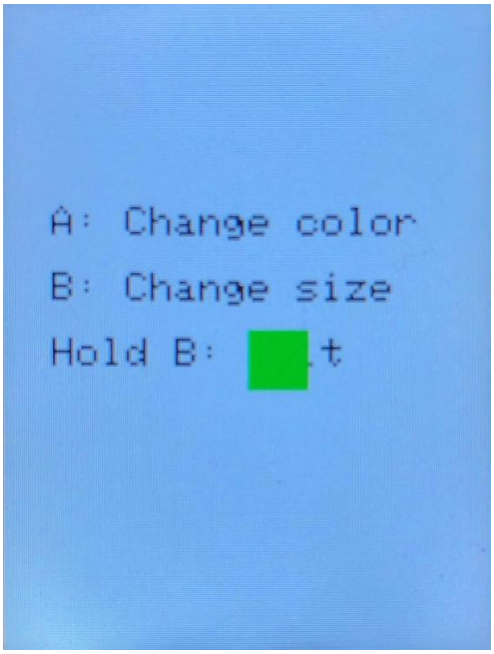


Figure 28. Screen test screen (size 2 green square)

In the screen test, the user can move around a square while changing its color and size with the A and B buttons. The square can move to any pixel of the screen.

Figure 29. The gyro/accel test screen

The gyro/accel test screen shows real-time readings from the accelerometer, gyroscope, and temperature sensor. It will also notify the user as to whether the gyro/accel is enabled or disabled in firmware.

Figure 30. Note: the MPU-6050 is disabled when using an external gamepad.

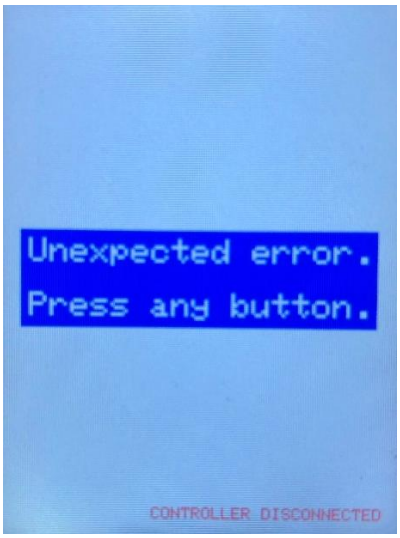


Figure 31. Error screen

If any error occurs during operation, the software can set an error flag and the console brings up the error screen. Pressing any button will reboot the console back to the splash screen.

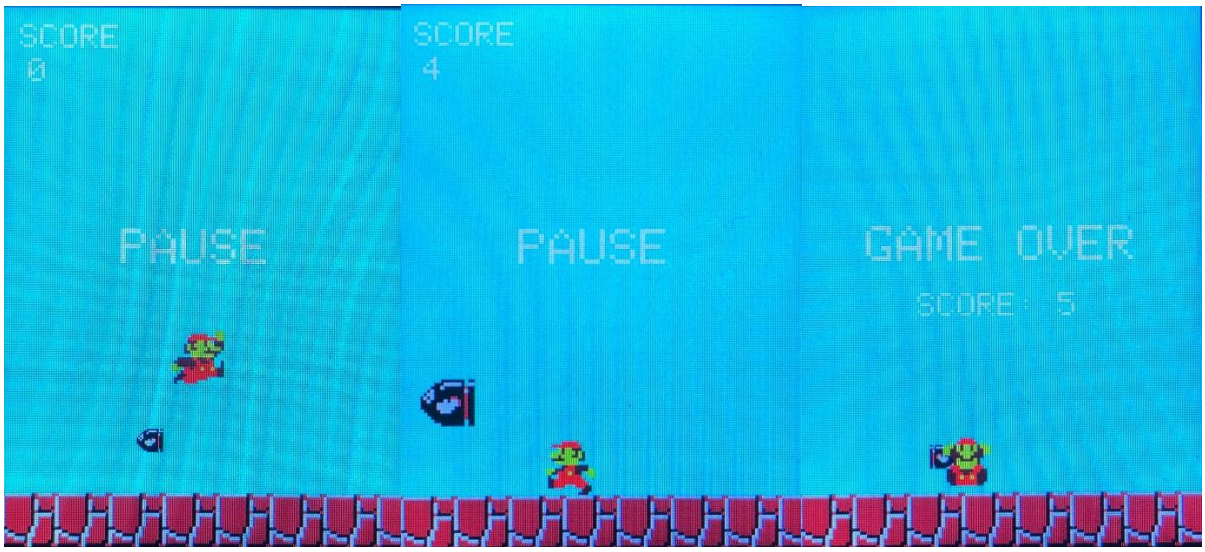


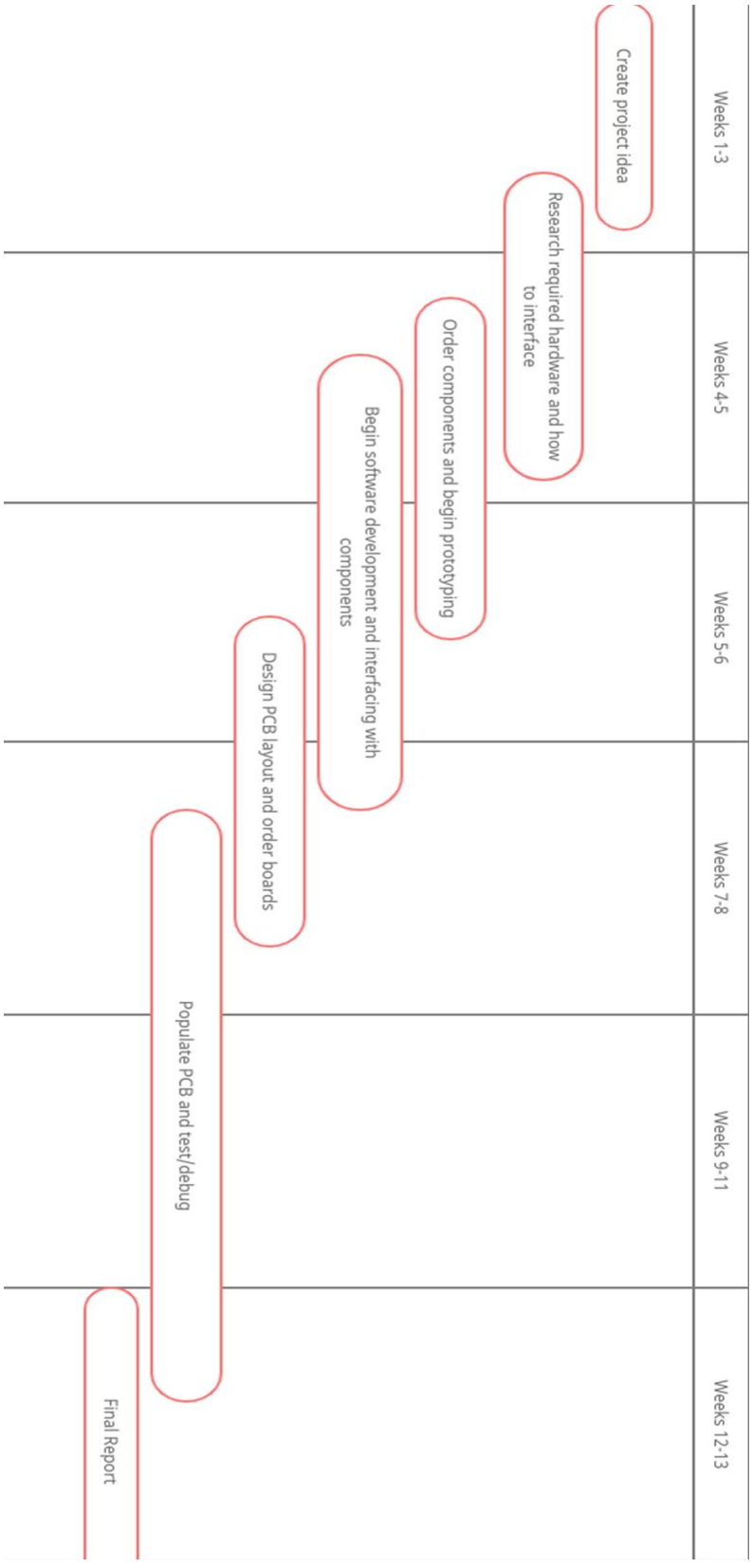
Figure 32. Mario game demo

The Mario-inspired game demo built into the console will load a screen like those above. Mario can walk, run with the B button, and jump with the A button. Start is pause. A Bullet Bill enemy will randomly spawn coming from either the left or right edge of the screen. The enemy will be a random size (small or medium) and have a random speed, and spawn at a random height (within the play area). Once the enemy reaches the opposite side of the screen, it will despawn and the player is awarded 1 point for each enemy cleared. A new enemy will respawn after a set time interval. If the player is hit, it's game over, after which any button press will restart the game.

Bill of Materials

PIC24FJ64GA002-E/SP MCU	x1	$3.70
MPU-6050 accel/gyro	x1	$6.95
LTC1661 DAC	x1	$4.12
Adafruit IL9340 + SD combo	x1	$24.95
PSP analog thumbstick	x1	$3.50
PC-100 15-pin ext. port	x1	$5.00
Pushbuttons	x4	$4.00
3.55mm HP jack	x1	$0.95
Mini metal speaker	x1	$1.95
Micro USB breakout board	x1	$1.50
SPDT toggle switch	x1	$0.60
1k ohm resister	x6	$1.80
10k ohm resister	x1	$0.30
4.7 ohm resister	x2	$0.60
LM3940 regulator	x1	$1.40
1N4148 diode	x2	$0.30
100nF capacitor	x2	$1.30
10uF capacitor	x2	$1.33
220uF capacitor	x1	$0.39
33uF capacitor	x1	$1.27
0.47uF capacitor	x1	$0.44
2n2222 transistor	x1	$2.48
2N2907A transistor	x1	$2.48
10k ohm thumbwheel pot	x2	$4.30
TOTAL		$75.61

Project Gantt Chart



Collaboration with Consulting Partner

Given the nature of the entirely online semester, meeting in person with my consulting partner was simply not an option. For this reason, the extent of our collaborative effort was mostly asking each other about some parts/materials recommendations or experiences, or asking about course-related deadlines and formats. For example, we had discussed the allowable use of breakout boards in our designs as well as the general feasibility of certain types of displays and external storage early on in the semester. We both understood that it is important to be able to work together if any problems may arise on either side, but luckily, neither of us experienced too much turbulence in our projects to warrant much extra help. However, we were always assured that the other person would be available and willing to aid.