

SENIOR DESIGN

---

# COST-EFFICIENT ANALYSIS & GENERATION OF POLYPHONIC MUSIC

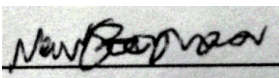
---

April 9, 2014

Matt Bagnara  
Purdue University  
ECET Department  
West Lafayette, Ind.  
[mbagnara@purdue.edu](mailto:mbagnara@purdue.edu)

AVR synthesizer · xmega · PS/2 · polyphonic music

Signed:

A small, rectangular image showing a handwritten signature in black ink on a light-colored background. The signature appears to be "Matt Bagnara" written in a cursive, slightly slanted style.

This report and project is submitted willingly as my own works. This document is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

## Abstract

The problem of an easy-to-use music-making device has been one that countless designers have tackled with a different approach. The balance between niche usage, low cost, and usability has been difficult to balance. Many designs that exist are for specific usages and the typical unification of these devices may seem redundant. Also, the marriage between such a device and hobbyist development may be too focused. The objective of this project is to design and construct a polyphonic digital synthesizer. Unification of the device into an educational kit would satisfy the curiosity of the hobbyist. A successful synthesizer will require a well-constructed input device and a microcontroller or processing platform. The synthesizer shall incorporate pulse, triangle, and noise tracks, as well as a means of mixing/synthesis. envelopes are anticipated, as they would allow much more musicality in dynamics. financially, the construction of the synthesizer is to remain within a set budget. optimally, the synth will be able to play music in real-time by the user and read data from a memory a modest key entry device shall be constructed for the user to interact with the hardware directly. the synthesizer should also include the ability to configure & display the interaction to the user in some kind of gui. the intent of this project is to construct a synthesizer that could be used by hobbyist and beginners alike. this would require an interface with enough usability for both amateurs and hobbyists.

The process of creating a synthesizer of this nature requires looking at the problem from the viewpoint of the users who would be using it. The device was created using low-cost materials, but also looked advanced into emerging technologies such as the XMEGA architecture. The development was done in such a way to facilitate free and fair-use of all development tools, and facilitated the knowledge of skills that would be practical and useful. This motivated the design of the software to all be accomplished using a POSIX environment. Also, the choice to use a PS/2 keyboard married legacy hardware, and new. The keyboard was chosen as an item that many users may have lying around could just plug-in without purchasing expensive hardware. This design caters to students and hobbyists alike.

The results of the design process display a functioning synthesizer prototype that is able to play real-time music, and also play recorded data. The project functions as a whole, only needing finalizations on the user-interface to conjoin the to modes of operation. Polishing is still needed, and the testing thus far reflects that. Nonetheless, this design is still a prototype with future capabilities down the pipeline.

# Contents

<b>1. Introduction</b>	<b>5</b>
1.1. Needs . . . . .	5
1.2. Objectives . . . . .	5
<b>2. System Overview</b>	<b>6</b>
2.1. System Requirements . . . . .	6
2.1.1. Marketing Requirements . . . . .	6
2.1.2. Requirement Specification . . . . .	7
2.2. Concept Generation & Evaluation . . . . .	8
2.2.1. Concept Generation Tree . . . . .	8
2.2.2. Concept Blocks . . . . .	8
2.3. System Decomposition . . . . .	9
2.4. Developmental Plan . . . . .	11
<b>3. Subsystem Technical Descriptions</b>	<b>13</b>
3.1. Synthesis subsystem . . . . .	13
3.2. LCD display subsystem . . . . .	13
3.3. PS/2 Keyboard subsystem . . . . .	14
3.4. Processor subsystem . . . . .	14
3.5. SD card subsystem . . . . .	14
<b>4. Testing Methods, Results and Evaluation</b>	<b>16</b>
4.1. Subsystem Test Plans . . . . .	16
4.2. Integration Plans . . . . .	16
4.3. Integration Testing . . . . .	18
4.4. Final Test Plans . . . . .	22
<b>5. Conclusions &amp; Recommendations</b>	<b>26</b>
<b>6. References</b>	<b>27</b>
<b>Appendices</b>	<b>28</b>
<b>A. User instructions</b>	<b>28</b>
<b>B. Project schedule</b>	<b>29</b>
<b>C. Budget</b>	<b>31</b>
<b>D. Bill of Materials</b>	<b>32</b>
<b>E. Overall Hardware Schematic</b>	<b>34</b>
<b>F. Large Software Diagrams</b>	<b>36</b>
<b>G. Large Software Listings</b>	<b>38</b>
<b>H. Calibration Instructions</b>	<b>40</b>

<b>I. Datasheets</b>	<b>41</b>
<b>J. Subsystem Unit Tests</b>	<b>42</b>
<b>K. Testing Documentation</b>	<b>45</b>
<b>L. Licenses</b>	<b>49</b>

# 1. Introduction

## 1.1. Needs

Electronic synthesizers are internationally found as means of communicating creative musical content. Chiptune is an application of electronic synthesizers that emulate the classic 8-bit or early electronic music using the limited hardware of the time. While the ability to interpret sound data directly to a synthesizer is not new, the investigation of such a project would be an enjoyable learning experience, and perhaps create a new approach to an existing product. The device could be directed towards construction as a kit; as such the intended users could be hobbyists and students who wish to construct a device they can use, while learning from it. This covers the major initial decision to license the project as it has been. The design and customization can be tailored to a specific usage. Also, the usage of limited hardware is anticipated to cost less.

A typical synthesizer is capable of real-time synthesis & playback. Songs are likely stored in a removable media which can be swapped for recording. The sound emitted from a typical polyphonic electronic synthesizer is in the form of arbitrary waveforms, including a noise track for percussion.

Synthesizer kits are also available to be built and operate. The user may prefer to purchase a commercial available option, However, a custom-made synthesizer will offer all the ability of hardware chiptune creation with the functions of a modern synthesizer at a modest price preferable under USD\$200.

## 1.2. Objectives

The objective of this project is to design and construct a polyphonic digital synthesizer. A Successful synthesizer will require a well-constructed input device and a microcontroller or processing platform. The synthesizer shall incorporate pulse, triangle, and noise tracks, as well as a means of mixing/synthesis. Envelopes are anticipated, as they would allow much more musicality in dynamics. Financially, the construction of the synthesizer is to remain within a set budget. Optimally, the synth will be able to play music in real-time by the user and read data from a memory A modest key entry device shall be constructed for the user to interact with the hardware directly. The synthesizer should also include the ability to configure & display the interaction to the user in some kind of GUI. The intent of this project is to construct a synthesizer that could be used by hobbyist and beginners alike. This would require an interface with enough usability for both amateurs and hobbyists.

## **2. System Overview**

The main idea is to create a synthesizer that can interface the user through a keyboard or other key entry device. The input shall allow the instrument to be played in real-time. The user interface or display can be navigated to select different sounds or waveforms(instruments/voices) in this mode. The secondary mode allows the playback of external music data which can be connected to a workstation computer to play data. The user display would be utilized in this mode to select the music data or file to be played. The output audio would be a consumer audio interface which can be later amplified or connected to a speaker.

### **2.1. System Requirements**

#### **2.1.1. Marketing Requirements**

1. The cost shall stay within a low level budget in a non-niche market.
2. The synthesizer shall contain an intuitive interface for amateurs and hobbyists.
3. The synthesizer should provide playback of recorded digital data.
4. The synthesizer should provide real-time playtime of data.
5. The synthesizer should synthesize digital data as polyphonic music.

### 2.1.2. Requirement Specification

Table 1: Requirements

Marketing Requirement	Engineering Requirement	Justification
1.	The budget for the synthesizer will be USD\$200, but the total cost shall not exceed USD\$225.	The budget constraint will provide a modest price.
2.	The synthesizer shall offer a visual user interface in the form of a two-line character LCD with a lit screen.	To provide an easy, intuitive, powerful control interface.
2.	The synthesizer will not contain more than three sublevels of menus.	Simple, easy to use interfaces that are organized will be logical to follow.
3.	The synthesizer will provide interpretation of a digital format for playback. Dat speed from from storage should be at least 8 Mbit per Second.	Playback of recorded data will require an abstraction of music into a data representation.
4.	The synthesizer will offer a method for external storage following an SD (Secure Digital) SPI interface.	External storage may easily be interfaced by a PC system for generating digital data for playback.
5.	The synthesizer will offer interface that will facilitate playing as an instrument with a response time less than 5ms.	Realtime playback will require a method the user can play the instrument.
6.	The total output frequency response should be between 30 and 12KHz.	Audio playback shall play within the average response of the human ear.
7.	The synthesizer will generate no less than three voices, including one distinct square, sine, and noise voice to be played simultaneously.	Music composed of individual waveforms can be mixed to represent a wide variety of polyphonic music.

## 2.2. Concept Generation & Evaluation

### 2.2.1. Concept Generation Tree

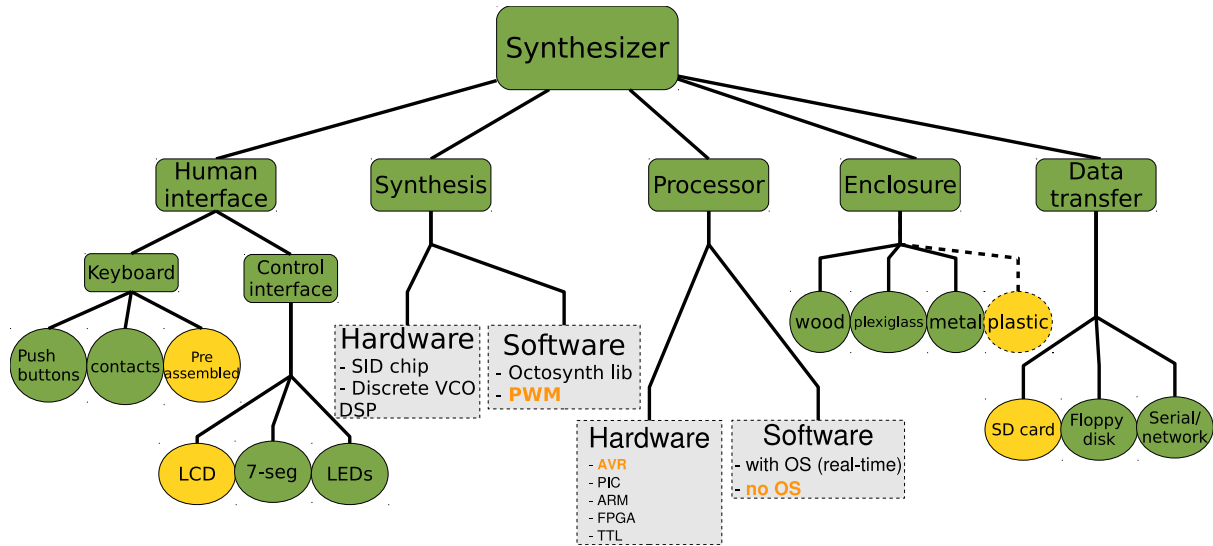


Figure 1: Concept Generation Tree displaying selected items in Orange

### 2.2.2. Concept Blocks

1. Main Processor The main processor will handle user input from the keyboard and the user interface. It will drive the user display, and handle the data-transfer abstraction from the user. It will also interpret the music data, whether it be MIDI, etc., and generate data for use in synthesis.
2. Keyboard The keyboard provides the basic user interaction for real-time music creation. The keyboard may be a prefabricated accessory, or may become its own block.
3. User Interface The user interface allows the user to select which song to play on the external media.
4. Synthesis The synthesis software block is where the waveform generation occurs. Music data generated by the processor is formed into arbitrary waveforms and amplified using an envelope function
5. Enclosure The enclosure will provide adequate housing for its use as a synthesizer. Looks and aesthetics will cater to usability for amateurs and hobbyists and could feature a retro design.
6. Data Transfer The data-transfer will provide support for external media. This will facilitate the transfer of external music data for use in playback. The data can be



fabricated on a workstation computer and played back on the synthesizer through this transfer interface.

## 2.3. System Decomposition

Table 2: Synthesizer Subsystem

<i>Module</i>	Synthesizer
<i>Inputs</i>	External DC power: 5V dc External Media for data storage playback: SD card
<i>Outputs</i>	Polyphonic music playback through consumer audio specification. TRS connector
<i>Functionality</i>	Provide polyphonic synthesis of keyed music by the user and playback of recorded media

Table 3: Processor Platform Subsystem

<i>Module</i>	Processor Platform
<i>Inputs</i>	Serial or MIDI interface from key: PS/2 data interface to tertiary storage: SPI SD card Logic signal from keyboard: 5V dc Power: 3.3V dc
<i>Outputs</i>	Logic signal for LCD display: 3.3V dc parallel(8-bit) Digital buffers to synthesis block: 3 voices, 256 bits
<i>Functionality</i>	To drive user interface of the synthesizer and manage data acquisition into the synthesis of polyphonic PWM waveforms.
<i>Rationale</i>	Provide method for interchangeable Processor interface for future expandability.

Table 4: SD Card Subsystem

<i>Module</i>	SD Card
<i>Inputs</i>	Power: 5V dc PS/2: psuedo I2C interface
<i>Outputs</i>	PS/2: psuedo I2c interface
<i>Functionality</i>	
<i>Rationale</i>	

Table 5: PS/2 Keyboard Subsystem

<i>Module</i>	PS/2 Keyboard
<i>Inputs</i>	Human interface through individual keys Power:5V dc PS/2: psuedo I2C interface
<i>Outputs</i>	PS/2: psuedo I2c interface
<i>Functionality</i>	Provide synthesis of output digital logic voltages for music playback.
<i>Rationale</i>	Provide method for interchangeable user-input.

Table 6: Synthesis Subsystem

<i>Module</i>	Synthesis
<i>Inputs</i>	Digital buffers to synthesis: 3 voices, 256 bits. 5V dc
<i>Outputs</i>	Audio signal for consumer electronics: 0.894Vpp
<i>Functionality</i>	Provide synthesis of output digital logic voltages and mixing for music playback.
<i>Rationale</i>	Choosing to create a discrete voltage controlled oscillator would allow a customized solution. It would allow multiple voices to generate their own waveforms in parallel, creating a form of modularity. It could be easily controlled using digital potentiometers. Mixing could easily be done at the end. The usage of envelopes would be facilitated through use of VCA (voltage controlled amplifiers)

Table 7: LCD Subsystem

<i>Module</i>	User interface(LCD)
<i>Inputs</i>	Power: 5V dc Logic from processor: 3.3V dc parallel 8 bits
<i>Outputs</i>	Human interface
<i>Functionality</i>	Provides the user with an interface for navigation of features.
<i>Rationale</i>	An LCD would allow the creation of a useful interface. Can create intuitive interfaces through use of graphical menus and display of information.

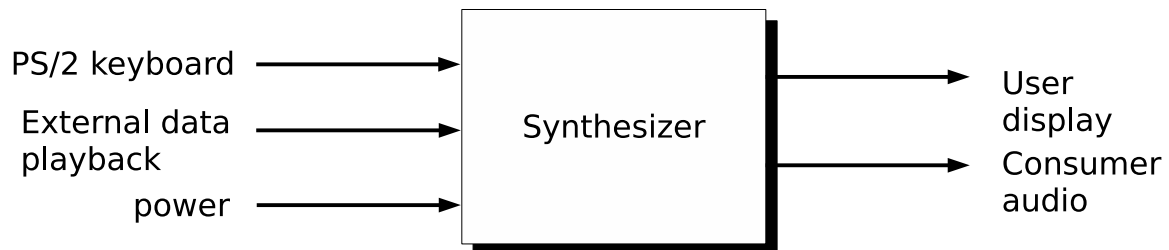


Figure 2: Level 0 - Synthesizer

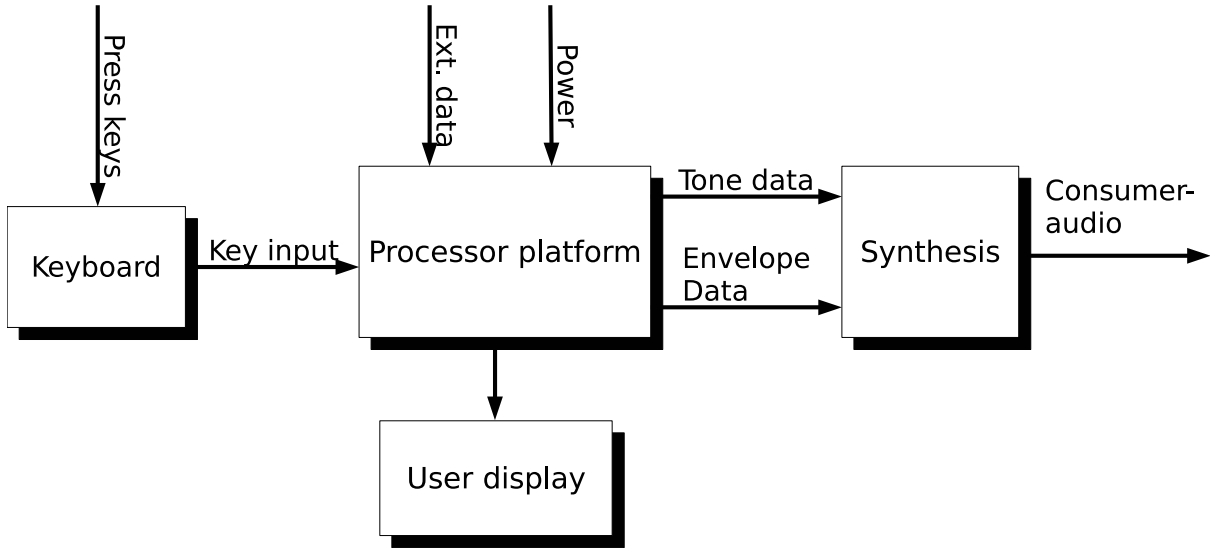


Figure 3: Level 1 - Synthesizer

## 2.4. Developmental Plan

The development plan for the synthesizer will divide between the hardware components and the software development. About 25% of the project will involve hardware. The following table will record this breakdown of the subsystems.

Table 8: Itemized subsystem development

Subsystem	Dev Ratio	Dependencies	Software	Hardware
PS/2 keyboard	0.10	Processor	PS/2 driver	Connector:proto
LCD display	0.10	Processor	LCD driver	protoboard
External SD	0.20	Processor	SD library available	
Synthesis	0.25	Processor	Software synthesis	Hardware mixer
Processor	0.45	N-A	Software operating system	Dev board

The development ratio gives an idea of what amount of time will be spent on each subsystem.

The order in which development shall carry out will tentatively follow this pattern:

### 1. Processor

Processor development shall start first and begin with the hardware board being

created or chosen. Currently, the choices are to use an AVR developmental board or an AVR programmer using a custom created proto-board or perf-board.

The software component will also begin and initial development can begin concurrently until the build environment is established. Current software choices are leaning toward the usage of a Real Time Operating System (RTOS) The processor software will be a large component of this development as the abstractions for runtime are formed in this stage.

## 2. **Synthesis**

Synthesis development will be tightly integrated with the processor platform. The software development can begin first, following usage of the processor as its resource. The wavetable and sampling routines are to be developed at this point. Following creation of the software synthesis component, the hardware DAC may be chosen for sound output.

## 3. **External SD**

Perhaps one of the more complex peripherals, the SD storage device shall be incorporated into the runtime of the processor. The current anticipated solution is to include an SPI[3] SD library and work with that. The hardware component of the SD interface shall be incorporated into the development board being used.

## 4. **PS/2 Keyboard**

The PS/2 keyboard requires no assembly because it shall be obtained as a finished component. The software-level driver shall be written into the runtime of the processor, and will be interfaced via serial peripheral on the processor platform[1]. The hardware port will be placed on the development board for testing and usage.

## 5. **LCD Display**

The LCD display requires no assembly because it shall be obtained as a finished component. The software-level driver shall be written into the runtime of the processor, and will be interfaced via the software specification of the LCD display, most likely being SPI. The hardware port will be placed on the development board for testing and usage.

## 6. **Mechanical Plan**

Mechanical chassis focuses on a small profile enclosure. The material of choice for the enclosure will most likely be wood or 3D printed to keep costs low. Figures 9 & 10 display a rendering using *Valve Hammer Editor* of what the chassis may look like. It features a mounting for the LCD display and main power switch on the front. The rear will provide fixtures for the DB-9, power, and TRS connectors. The right side will feature a slit to clear the turn potentiometer as it protrudes from the enclosure.

### 3. Subsystem Technical Descriptions

Much of the testing being done will feature the debug menu which will be incorporated into the development board. The debug menu will feature self-tests documented in [Testing Documentation](#) to use to verify functionality of the peripherals. The tests will run using an external serial interface on the processor and can be accessed using a serial terminal.

#### 3.1. Synthesis subsystem

The Synthesis subsystem consists of the software & hardware which creates the output waveform from the digital data being created by the processor. This subsystem is composed of software routines that create a audio buffer to write out to the DAC as well as the routines for mixing the separate buffers. This subsystem is vital to the tonal generation of music. First of all, to verify the proper operation of the synthesis portion, the waveshape, frequencies, mixing, and other parameters need to be validated.

Synthesis is being done in both hardware and software, so both components are required. To ensure proper generation of arbitrary waveforms, the output of the synthesizer shall be connected to a 50 ohm terminated oscilloscope. The visual oscilloscope tube can be used to capture the correct waveshape for each voice. Next, the frequency range of the synthesizer will be tested using the built-in sweep function in the debug menu. The oscilloscope can, again, be used to capture the frequency vs. time data to verify this part. The voltage levels of the waveforms can also be checked at this point. The final task to verify would be the mixing capability of the synthesizer by generating polyphonic tones in the form of two, and three note chords. This may be verified by viewing a frequency-domain plot on the oscilloscope.

#### 3.2. LCD display subsystem

The LCD unit consists of the LCD display, as well as the display driver being written on the software. The LCD subsystem will act as the user's interface to the synthesizer when it is being run in real-time for playback or for playback of premade audio. The purpose of this display is to provide the user with an interface.

The LCD will operate using a user-created driver and the output shall be connected using a parallel interface. First, the verification of the display of characters will simply be done by cycling through printable ASCII characters. This will be done using the functionality of the debug menu. Also, the processing of control characters and clearing the screen will be demonstrated using this same debug screen.

### 3.3. PS/2 Keyboard subsystem

The keyboard unit consists of the PS/2 display, as well as the PS/2 driver running in software. This subsystem is used to provide key entry to the synthesizer block. It is vital to the synthesizer as it provides the only keyed input from the user. It will be used as the musical keyboard, as well as the controls to the user-interface on the LCD.

### 3.4. Processor subsystem

The processor platform subsystem consists of the microprocessor platform being used, as well as the software layer being run on it. This subsystem is the heart of the synthesizer platform, as it interfaces with all peripherals, and all software will be written on.

#### Hardware

The hardware platform for this processor subsystem will be an ATXMega256D3. This processor will feature 256 kilobytes of programmable flash memory, and port-assignable timers. The selected clock will be a 16KHz crystal. Programming will be done via PDI using an AVRISP MKII. PWM timers output from the microcontroller mixed and filtered will be used to perform the waveform generation.

#### Software

The software of the processor platform will ideally provide hard or soft real-time support through use of a real-time operating system. The motivation of the operating system is to accomplish real-time tasks within stringent deadlines. [4]Because this system will be processing audio samples, a real-time operating system will be used to timely produce the music. As depicted in Figure ??, the software for the synthesizer will be divided into tasks which interface with a particular peripheral.

### 3.5. SD card subsystem

The SD unit consists of the SD card interface, as well as the software filesystem driver being used to interface via SPI. The SD subsystem provides the synthesizer a method to playback non-volatile memory or pre-recorded sequences. To verify the proper functionality of the SD device, the creation of file handles, and reading of blocks should be demonstrated.

The SD will operate using an available FAT\_FS [2] library, and the input shall be connected using an SPI interface. Because the synthesizer will only be reading music data, the demonstration will require the ability to read off the SD card. This subsystem is integrated into the processor platform, therefore, it requires the processor subsystem to be tested. The demonstration will require the creation of a file handle, and reading the blocks of data into memory of the processor. The debug program on the processor will read ASCII data off the SD card and output it via its external serial interface.

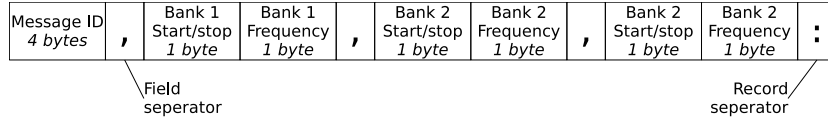


Figure 4: ASCII message structure

Figure 4 above depicts the preliminary data-layout chosen for the scope of each playback sequence. This data represents a sequence of messages that composes a tune being read from the SD in ASCII mode. As a result, filesizes will be much larger, but this is at the expense of creating an easily modifiable human-readable format.

Next, the ASCII messages are then evaluated and an associating **order** object is created to represent the item. A library was designed for the purpose of representing a linked-list datastructure. An executive decision was made at this point to represent the data using a linked-list. Although a linked-list does not allow random-access of data, they provide insertions and deletions to occur in constant time  $\Theta(1)$ [5]. This was evaluated due to the amount of messages that would be processed. The library used consists of the **onode** structure which points to the data containing the actual music.

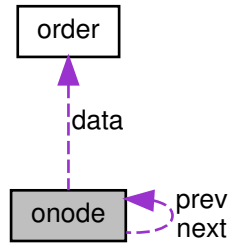


Figure 5: Collaboration Diagram for onode

Table 9: Struct **order** Data Fields

int	id
char	bank1_startstop
char	bank1_note
char	bank2_startstop
char	bank2_note
char	bank3_startstop
char	bank3_note

Figure 5 shown above is used as the encapsulation method for the **order** tone data. The data is represented above in 3.5 and this data is passed on to the synthesis block to process the individual messages that come in. Because these messages are passed asynchronously, they are not long-lasting in the life of the execution cycle. Transformation occurs fast, yet the structure used would allow future expandability for more complex tone messages.

## 4. Testing Methods, Results and Evaluation

### 4.1. Subsystem Test Plans

Testing for LCD unit functionality was successful as shown in Table 23. The testing was done on a stub development board ATMEGA16. Functionalities demonstrated were the ability to display two-line character buffers to the screen, presenting custom glyphs as characters, contrast adjust, and clearing the screen. The requirement for this test maps to requirement #2 to provide an intuitive user interface. The LCD testing results will demonstrate this ability in conjunction with the keyboard to develop a menu system. See Appendix [Testing Documentation](#) for test setup.

Testing for the PS/2 was successful using the ATMEGA16 as shown in Table 24. Functionalities demonstrated were the ability to read scancodes from the keyboard on key press and key release events. Because the intent is to use the alphanumeric keyboard for key entry, only level1 scancodes will need to be interpreted. This test maps to the requirement to provide an intuitive user interface, as well as a means to input data for playback. See Appendix [Testing Documentation](#) for test setup.

Testing for the SD storage was successful as shown in Table 25. The SD was damaged during testing and yielded inconclusive evidence. With recent acquisition of the SD card breakout board, testing can continue. The requirement for this test will map to requirement #3, and #4. Although inconclusive at this point, the testing will continue. See Appendix [Testing Documentation](#) for test setup.

Testing of the synthesis unit is complete in conjunction with the processor platform testing as depicted in Table 26. Because of the tight integration between these two subsystems, development was done on the same platform. Functionality to demonstrate is the ability to generate compound waveforms from arbitrary waveform buffers. The simplest of these would be to output an arbitrary waveform scaled to frequency  $f$ . This will be used to generate a chromatic frequency sweep. Noise, generation will follow. Superfixing two waveforms will create two simultaneous tones at 440Hz and 460Hz. Three tones will be superfixed next to create deeper chord structure at 440, 460, and 520Hz. Finally, square and triangle waveforms are tested. This testing sequence will demonstrate the requirement of #7 and #6. See Appendix [Testing Documentation](#) for test setup.

Testing of a processor platform demonstrates overall functionality of the microprocessor. Testing was be done in a stateless mode of operation to ensure proper functionality of the unit. Metrics observed in Table 27 were poweron functionality, and PDI advertisement.

### 4.2. Integration Plans

Integration of the subsystem blocks will follow an order which allows the system to be joined logically. The listing below illustrates the 5 phases of integration.

1. Processor platform



2. LCD display
3. SD storage
4. Synthesis
5. PS/2 keyboard

With the synthesizer, the initial block will be the processor platform. This will represent the base for which all subsequent tests will be dependent on. This subsystem is vital to all subsequent tests.

Second, the LCD display will be added with its hardware and software functionality. This integration will be vital to provide the user with an interface that can be used for testing further on. The initial LCD test will be the same test used in the LCD subsystem testing because it is essentially the same.

Third, the SD storage will be integrated with its FAT library component, as well as the hardware interface. The testing for this stage will include reading ASCII data from the card and writing it to the LCD display. This test will be part of the debug system as it is being built for integration.

Next, the synthesis subsystem can be integrated with its software components. The synthesis subsystem will make usage of the incrementing debug system being built up. The tests of this integration will involve simple waveform generation, more sophisticated chord synthesis and generation, and noise track synthesis.

Last, the PS/2 keyboard will be integrated into the system. The PS/2 keyboard testing can be used in conjunction with the LCD to display an intuitive menu that can be navigated using the keyboard. Testing for this integration will facilitate the creation of a whole debug-menu system that can be navigated and displayed on the LCD. The final system integration step will test some of the Engineering requirements as a whole to ensure proper functionality throughout. These tests will address the requirements listed in Table 2.1.2 . This will include tests of reading data from the SD card and playing back. Also, the system will need to allow real-time playback when set to that state.

### 4.3. Integration Testing

Table 10: Processor integration test

Test Writer		Matt Bagnara		
Test Case Name		Integ-processor	Type	integration
Description	Includes the introduction of processor platform onto the testing board		Date	11/22/13
Setup	Connect processor power and setup as in Appendix J		Time	
Test				
Name	Action	Expected output	Pass	Comments
Poweron test	Connect +5V, +3.3V, GND on CONN6	LED on board should light green	✓	
processor presence	See <a href="#">Testing Documentation</a>	Should respond with device signature 0x1e9844	✓	

Testing for the processor integration stage went successful. The stage of testing began with the testing board layout. The testing board layout is explained in Appendix J - Test Board. The processor is the first stage to be integrated onto the testing unit. This was done by simply inserting the processor assembly into the testing board via DIS1 and connecting the programmer to CONN1. The processor integration provides the base for which all subsequent integration steps will follow. For this reason, the processor integration needed to be checked for integrity while interfacing the testing board. The demonstrations for this integration step simply required power to be supplied correctly to the processor assembly, as well as detecting the presence of the processor by the PDI interface.

Table 11: LCD integration test

Test Writer		Matt Bagnara		
Test Case Name		Integ-LCD	Type	integration
Description	Includes the introduction of the LCD display library into the codebase along with connection		Date	11/23/13
Setup	Connect processor power and setup as in Appendix J		Time	
Test				
Name	Action	Expected output	Pass	Comments
Poweron test	Power board as before and plug LCD ribbon to CONN2	LCD should light with a blue glow	✓	$VSS = 4.96V, VCC = 3.28V$
Contrast adjust	Adjust the potentiometer near CONN2	Contrast should adjust for visibility per lighting condition	✓	full range visible
Display of text	Execute displayText routine <code>main.hex</code>	Test text should display on LCD display	✓	

Second stage of integration includes the physical LCD display along with software used by the LCD to interface. The reason for choosing the LCD to integrate second was to facilitate a means of standard output to be used for further integration tests. The LCD needs to be tested using its software library to ensure proper display and visibility of data. The first line of testing is to ensure proper voltage to the logic supply of the LCD as well as to the lighting unit of the LCD. Next, the contrast adjusting functionality is tested being integrated to the testing board to ensure proper visibility adjustment of the display. The library is put to the test being integrated into the development environment of the processor. Testing display of buffered text to the LCD was successful at this stage.

Table 12: SD card integration test

<b>Test Writer</b>		Matt Bagnara		
<b>Test Case Name</b>		Integ-SD	<b>Type</b>	integration
<b>Description</b>	Includes the introduction of the SD card library into the codebase along with connection		<b>Date</b>	11/23/13
<b>Setup</b>	Connect processor power and setup as in Appendix J		<b>Time</b>	
<b>Test</b>				
<b>Name</b>	<b>Action</b>	<b>Expected output</b>	<b>Pass</b>	<b>Comments</b>
SD create file	Power board as before and plug SD ribbon to CONN3	File FILE should be create on root of SD filesystem	✓	
SD open file	Power board as before	File should open and display a <b>File Created!</b> message on the LCD	✓	
SD read file	Power board as before	File should be opened and first 16 characters should display on LCD	✓	

The SD card interface is the next stage to be integrated, following the plan of iteration. The SD interface consists of primarily a software library titled FAT\_FS and a SPI SD interface connectivity board. The integration of the SD library was vital to ensure proper functionality and correct any conflicts that may have been present on the databus. The software was consolidated at this point to make the software development blocks independent and able to be separated for individual unit tests. This proved to be successful because the sequence of filesystem tests integrated nicely into the testing board. The final testing demonstrated reading of ASCII data from a file on the SD and displaying it to the LCD. This was used to facilitate the reading of music data.

Table 13: PS/2 interface integration test

<b>Test Writer</b>		Matt Bagnara		
<b>Test Case Name</b>		Integ-PS/2	<b>Type</b>	integration
<b>Description</b>	Includes the introduction of the PS/2 keyboard library into the codebase along with connection		<b>Date</b>	11/26/13
<b>Setup</b>	Connect processor power and setup as in Appendix J		<b>Time</b>	
<b>Test</b>				
<b>Name</b>	<b>Action</b>	<b>Expected output</b>	<b>Pass</b>	<b>Comments</b>
PS/2 key press	Power board as before and plug PS/2-DB9 adaptor to CONN5	PS/2 scancode should display on LCD	✓	PS/2 requires level shifter IC for 3.3V to 5V I2C
PS/2 key release	Power board as before and plug PS/2-DB9 adaptor to CONN5	PS/2 scancode should display on LCD	✓	
Asynchronous playback by key pressed	Power board as before and plug PS/2-DB9 adaptor to CONN5	PS/2 scancode should display on LCD and simulated sound should display on LCD	✓	

The PS/2 keyboard interface was the next integration step. This step required the use of the PCA9306 level shifter IC to interface 5V with the 3.3V logic of the microcontroller. The integration implemented a routine in the main execution thread to fetch the contents of the PS/2 message buffer. This buffer contains the scancode, shift flags, and press/release flags.

Table 14: Synthesis integration test

<b>Test Writer</b>		Matt Bagnara		
<b>Test Case Name</b>		Integ-synthesis	<b>Type</b>	integration
<b>Description</b>	Includes the introduction of the synthesis wavegen library and link-list library into the codebase		<b>Date</b>	11/26/13
<b>Setup</b>	Connect processor power and setup as in Appendix J		<b>Time</b>	
<b>Test</b>				
<b>Name</b>	<b>Action</b>	<b>Expected output</b>	<b>Pass</b>	<b>Comments</b>
Read string from file on SD card and compile a message object	Power board as before	Successful message malloc message when successful on LCD	✓	
Add message object to linked list and fetch data from the data structure	Power board as before and plug PS/2-DB9 adaptor to CONN5	PS/2 scancode should display on LCD	✓	
Asynchronous playback by key pressed	Power board as before and plug PS/2-DB9 adaptor to CONN5	PS/2 scancode should display on LCD and simulated sound should display on LCD	✓	

Synthesis integration was final as it required use of the other subsystems to test and debug. Synthesis was tested by assigning frequency values to each key and testing the response of the key messages into tone production. The synthesis was also divided into three buffers to be tested using key configurations.

## 4.4. Final Test Plans

Final testing of the project will demonstrate adherence to the specifications outlined. The metric used will quantify the measurable requirement in the Requirements.

Table 15: Specification Requirement 1

<b>Test Writer</b>	Matt Bagnara		
<b>Test Case Name</b>	User Interface	<b>Type</b>	Final Requirements Test
<b>Description</b>	Verification of the intuitive user interface.	<b>Date</b>	03/31/2014
<b>Setup</b>	Connect processor power and setup as in Appendix J	<b>Time</b>	12:44
<b>Test</b>			
<b>Name &amp; Metric</b>	<b>Expected output</b>	<b>Pass</b>	<b>Comments</b>
Interface Presence; Menu Traversal	Menu should provide an option to playback a recorded audio file, or offer a real-time playback option. Further options to follow.	✓	
Interface response; Response Time	Delay time less than 5ms per input on PS/2	✓	Display update visibly instantaneous. No measurement necessary.
Interface Complexity; Number of submenus	For every root item in the main menu, there should be at most two sublevels. Navigate the menu tree to determine the hierarchy.	✓	Menu heirarchy is OK.

The specification testing above will illustrate proper and sufficient user interface traversal. The User interface is to be evaluated at this point. This will include menu layout and features. The response of the interface to navigate to various modes of operation is to be tested via response times. The complexity of the menu subsystem as a whole is also to be evaluated via enumeration of submenus. As of the time of this report, the menu-system needs to be polished towards being a final user interface. The menu system needs to be established as a whole, and implemented as a front-end to the system.

Table 16: Specification Requirement 2

<b>Test Writer</b>	Matt Bagnara		
<b>Test Case Name</b>	Synthesis	<b>Type</b>	Final Requirements Test
<b>Description</b>	Verification of music-producing synthesis functionality	<b>Date</b>	05/05/2014
<b>Setup</b>	Connect processor power and setup as in Appendix J	<b>Time</b>	11:17
<b>Test</b>			
<b>Name &amp; Metric</b>	<b>Expected output</b>	<b>Pass</b>	<b>Comments</b>
Frequency Response; Frequency sweep	Using a provided debug routine, the output of the synthesizer should provide a sweep that begins at 30Hz and proceeds up to 12kHz in increments of 20Hz. Tolerance levels of 500mHz	✓	Frequency response is accurate with respect to the response. The transfer error slightly increases at higher frequencies.
Waveform Generation; Wave type	Ability to generate distinct square, sine, and noise waveforms.	✓	
Waveform Generation; Wave Superposition	Ability to mix three distinct waveforms realtime.	✓	
Interface response; Response Time	Delay time less than 5ms per input on PS/2	✓	Note generation instantaneous. No timing necessary.

Specification test 2 will demonstrate the synthesis component of design. It focuses on frequency generation, and response. These variables can be measured as before during the subsystem tests. Nothing needs to change for them. The PS/2 response is implemented at this point, because the keyboard serves as the entry-point to asynchronous tone generation. This can be evaluated spatially via response time as well. This test is a lengthy process and so far, not all the frequencies have been verified. Response times can be measured using instrumentation.

Table 17: Specification Requirement 3

<b>Test Writer</b>	Matt Bagnara		
<b>Test Case Name</b>	Recorded Playback	<b>Type</b>	Final Requirements Test
<b>Description</b>	Verification of digital playback.	<b>Date</b>	00/00/00
<b>Setup</b>	Connect processor power and setup as in Appendix J	<b>Time</b>	00:00
<b>Test</b>			
<b>Name &amp; Metric</b>	<b>Expected output</b>	<b>Pass</b>	<b>Comments</b>
Detect SD card; Initialize time	SD card should be detected on boot of device. SD should initialize upon detection and be available to the operating system. Time between poweron and SD detection should take less than three seconds.	✓	
Playback; File Initialization Time	Files should be read off the SD card and loaded into the stack. The file should begin playing within one second of selecting the file.	✓	

Specification test 3 deals with the tertiary playback and sequencing of said data. The SD card serves as the filesystem platform for the loaded music data. As a result, the SD should be initialized and detected through the filesystem. This module can be tested as the subsystem test went, with the inclusion of the individual response times of loading a song from the SD while ignoring underlying file representations. Files should begin playing within a certain interval of being loaded from the device.



Table 18: Additional Tests

<b>Test Writer</b>	Matt Bagnara		
<b>Test Case Name</b>	Reliability and Stability	<b>Type</b>	Final Requirements Test
<b>Description</b>	Long-term usage and functional stability	<b>Date</b>	00/00/00
<b>Setup</b>	Connect processor power and setup as in Appendix J	<b>Time</b>	00:00
<b>Test</b>			
<b>Name &amp; Metric</b>	<b>Expected output</b>	<b>Pass</b>	<b>Comments</b>
Long Up-time; Usability	Device should sustain longterm usage and uptime. Memory leaking over time should not occur over long usage. Verifiable using memory debugger such as <i>Valgrind</i> .	✓	Device test 24hrs provided sufficient.
Exception Handling; File reads	Files read from SD should obey the specification for data layout in Figure 4. Exceptions in data layout and corner cases should be handled properly and terminate gracefully.	✓	Longterm debug testing and trials required.
PS/2 Keyboard; Trial	Different PS/2 keyboards should obey the specific protocol. Test ensures validity of many types and brands of PS/2 keyboard. Trial will be done using PS/2 keyboards available.	✓	Trial used IBM and eMachines keyboard.

Additional tests to be performed will deal with usability and longterm usage. The uptime usage is scalable and will be much more difficult to measure, as memory will need to be inspected to verify any leaks. Tools can be utilized in software such as *Valgrind* but may be increasingly difficult to use without a JTAG debugger. Memory integrity will have to be measured by practice observation. Exception handling will have to be observed in the same way using structured test routines, along with arbitrary usage metrics.

## 5. Conclusions & Recommendations

So far at this stage in development, the project is in its final physical form. The PCBs are mounted to the enclosure. A few minor issues were observed with the final form of the project with the following handlings. First, the wiring of the data and power pin on CONN5 were incorrectly swapped and this issue was mitigated through an external dongle to reverse the pins. Next, the volume potentiometer POT1SW was improperly selected, resulting in clipping waveforms at high volumes. As a result, the potentiometer will be sunken to the enclosure and be designated as a tuning adjuster because hardware volume adjustment was never an endorsed requirement specification.

At this current stage, the project is slightly behind schedule. The desire was to have all final development complete to facilitate adequate system testing for the demo. As of now, the testing cannot begin until the final user-interface is completed and finalized. This will take a little time, and in due time to be completed on schedule (see [Project Session 2 Schedule](#)). It will only require more time input on my end in the next week.

Looking ahead, the next steps would include the final testing and development of the software user interface. This would include the final user environment menu system on the LCD along with the keyboard assignments and user-instruction manual. The development for the user interface of the design will facilitate all keys being used for tone generation and specific rows of keys for each buffer(sine/square/noise).

## 6. References

- [1] Avinash. Ps2 keyboard interface with avr mcu, June 2012. Available as <http://extremeelectronics.co.in/avr-tutorials/ps2-keyboard-interface-with-avr-mcu/>.
- [2] ChaN. Fatfs module application notes, 2013. Available as <http://elm-chan.org/fsw/ff/en/appnote.html>.
- [3] F. Foust. Secure digital card interface for the msp430, 2004.
- [4] M Naghibzadeh. *Operating System: Concepts and Techniques*. iUniverse, Inc., 2005.
- [5] R. Sedgewick and K. Wayne. *Algorithms*. Pearson Education, 2011.

# Appendices

## A. User instructions

N/A

## B. Project schedule

Figure 6: Project Session 1 Schedule

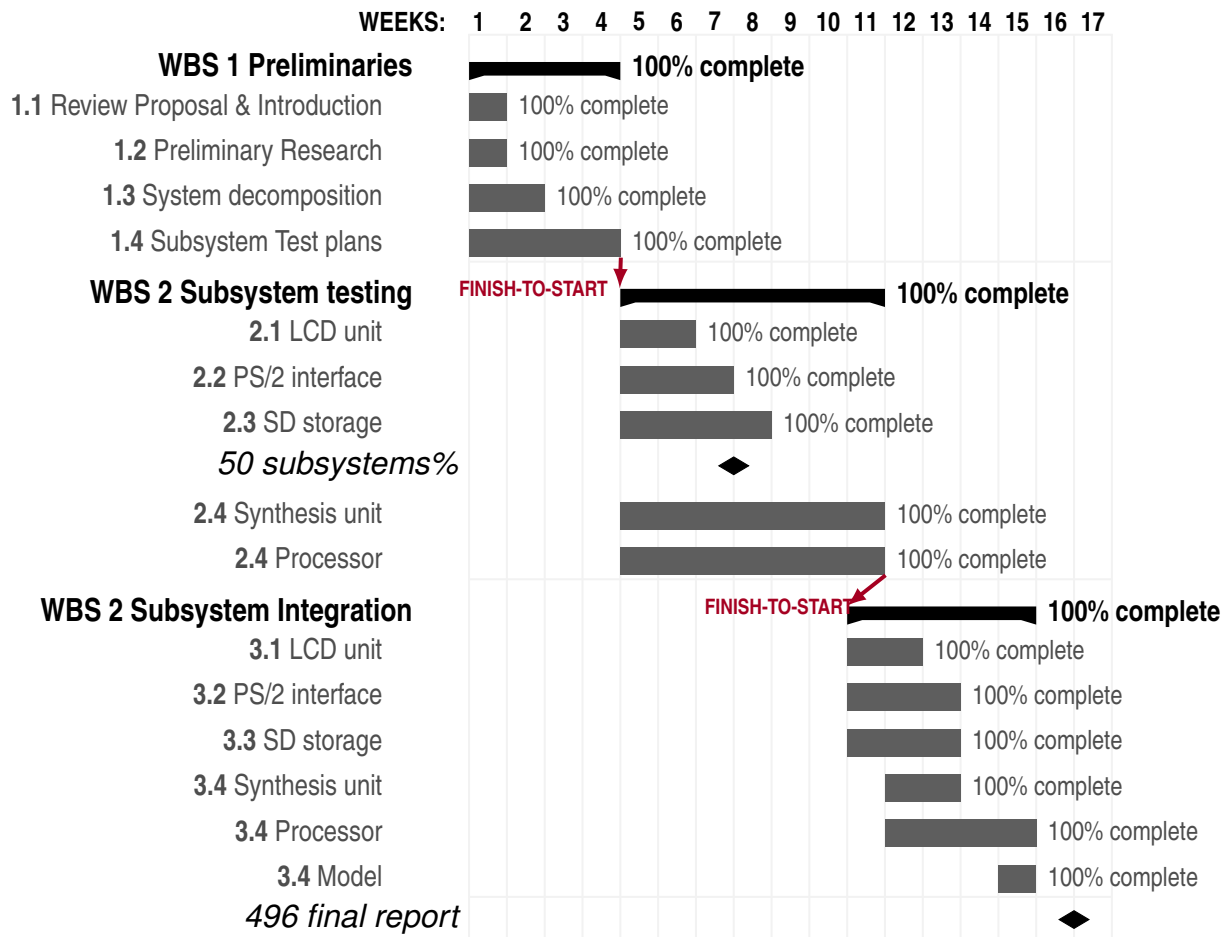
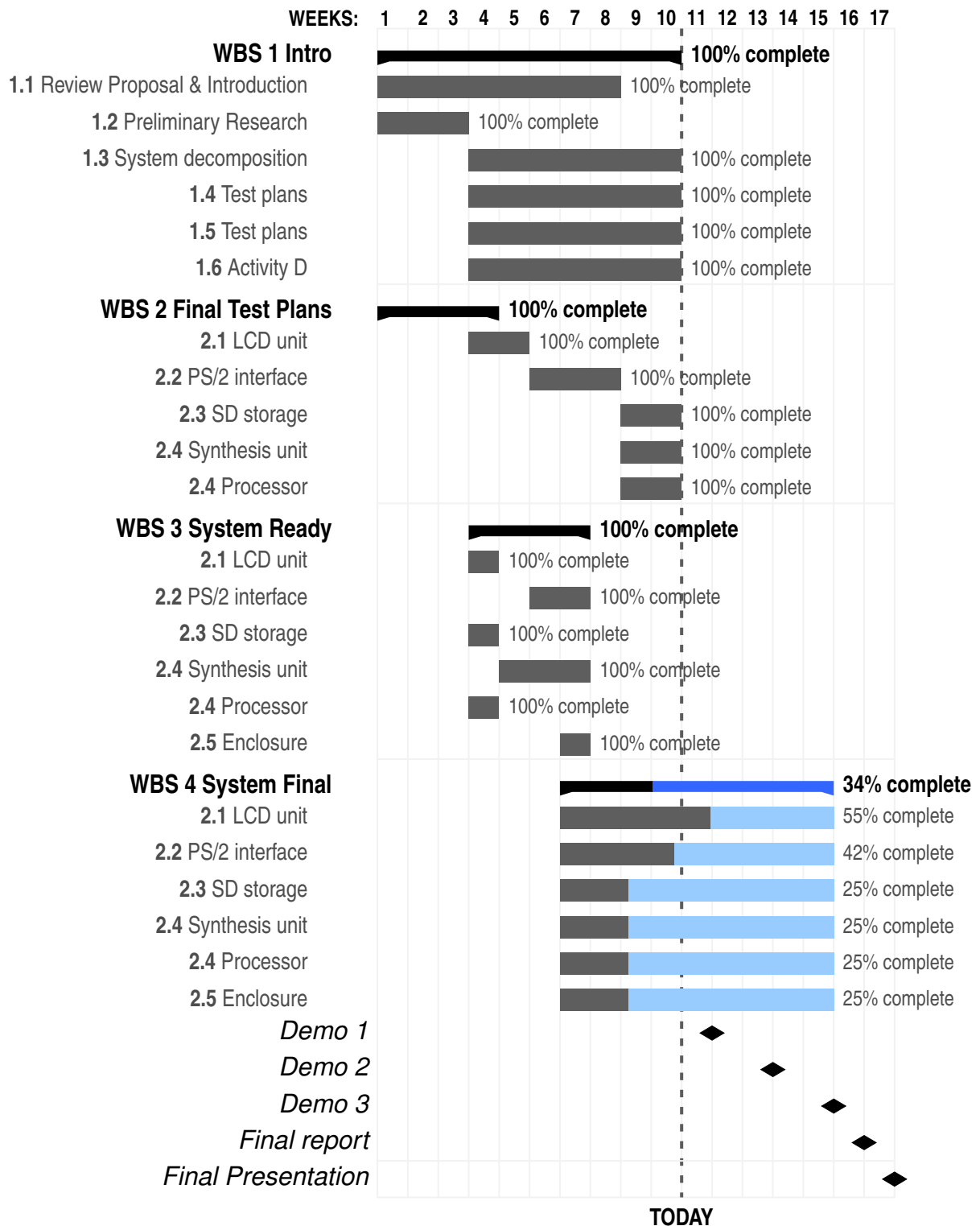


Figure 7: Project Session 2 Schedule



## C. Budget

Table 19: Projected Development Budget

Component	Projected cost(USD)	Actual cost(USD)	Delta
AVRISP programmer	\$15.00	\$28.00	\$43.21
SD/MMC socket	\$1.95	\$1.99	
LCD display	\$0.00	\$0.00 <sup>1</sup>	
Perfboard	\$5.00	\$6.75	
Enclosure materials	\$25.00	—	
ATxmega256D3	\$20.00	\$0.00 <sup>2</sup>	
PS/2 keyboard	\$0.00	\$0.00	
Misc. components \$10.00	\$20.00		
SD card	\$20.00	\$15.00	\$5.00

Table 20: Labor Development Costs

Component	Projected Hours	Total Hours
Review Proposal & Introduction	4	1.25
Preliminary Research	10	7.5
Documentation and Reports	12	20
Development Synthesis Block	5	8
Development LCD Block	5	4
Development PS/2	8	6
Development SD interface	8	6
Development Processor	20	10
Perfboard Testing	5	10
PCB layout & design	24	30
PCB construction	2	3
Physical Model	5	5
Testing Subsystems	10	5
Testing Integration	5	3
Testing System	10	0
	133	118.75

## D. Bill of Materials

Table 21: Development Overhead

Component	Count	Cost
QFP100(TQFPLQFP) DIP100 PCB converter adapter 0.5mm 0.8mm	2	\$2.99
2.54mm 1p-1p Pin 40pcs Dupont Wire Cable Line Connector 20c	1	\$1.81
SD card breakout reader module for AVR/ARM/PLC read and write	1	\$1.99
16MHZ crystal	1	\$2.00
Flux pen	1	\$5.95
3.9 inch x 6.3 inch 1 hole island eurocard	1	\$6.75
AVRISPMKII programmer	1	\$28.00
ATXMEGA256D3 processors	2	\$0 <sup>3</sup>
		\$98.56



Table 22: Bill of Materials

Component	Device	Cost
DIS1	QFP100(TQFPLQFP) DIP100 PCB converter adapter 0.5mm 0.8mm	\$1.25
SD CARD	SD card breakout reader module for AVR/ARM/PLC read and write	\$1.99
Q1	16MHZ crystal	\$2.00
DIS1	ATMEGA256D3	\$20.00
PCB	OshPark PCB manufacture	\$28.55
IC1	LM317 TS package	\$1.50
LED1	APT2012SURCK Red LED 0805	\$0.17
LED2	APT2012CGCK Green LED 0805	\$0.17
C1	0.1 $\mu$ EU2012	\$0.17
C2	0.1 $\mu$ EU2012	\$0.17
C11	electrolytic capacitor <sup>4</sup>	\$0.10
C12	0.05 $\mu$ EU1206 CAP	\$0.10
R2	1k $\Omega$ EU2012 RES	\$0.08
R4	1k $\Omega$ EU2012 RES	\$0.08
R5	1k $\Omega$ EU2012 RES	\$0.08
R7	1k $\Omega$ EU2012 RES	\$0.08
R11	1k $\Omega$ EU2012 RES	\$0.08
R12	1k $\Omega$ EU2012 RES	\$0.08
R13	1k $\Omega$ EU2012 RES	\$0.08
R14	1k $\Omega$ EU2012 RES	\$0.08
R15	10 $\Omega$ EU2012 RES	\$0.08
POT1	RV121SF-30-009J-A10K-00K 12mm Audio 10K W/SW	\$1.75
C4	16p EU1005	\$0.25
C5	16p EU1005	\$0.25
C6	100n EU2012	\$0.17
C7	100n EU2012	\$0.17
C8	100n EU2012	\$0.17
C9	100n EU2012	\$0.17
C10	100n EU2012	\$0.17
R6	200k $\Omega$ EU2012 RES	\$0.08
R1	240 $\Omega$ EU2012 RES	\$0.08
C13	250 $\mu$ PANASONIC-E ELEC_CAP	\$1.12
R8	330 $\Omega$ EU2012 RES	\$0.08
R9	330 $\Omega$ EU2012 RES	\$0.08
R10	330 $\Omega$ EU2012 RES	\$0.08
R3	330 $\Omega$ EU2012 RES	\$0.08
P1	10k $\Omega$ POT square	\$1.00
IC2	LM386M-1 S008	\$1.82
U2	PCA9306 SSOP08	\$0.94
CONN6	STEREOJACK STX3100	\$1.84
ENC	Plastic enclosure 1591CBK	\$6.95
		<hr/>
		\$72.12

## E. Overall Hardware Schematic

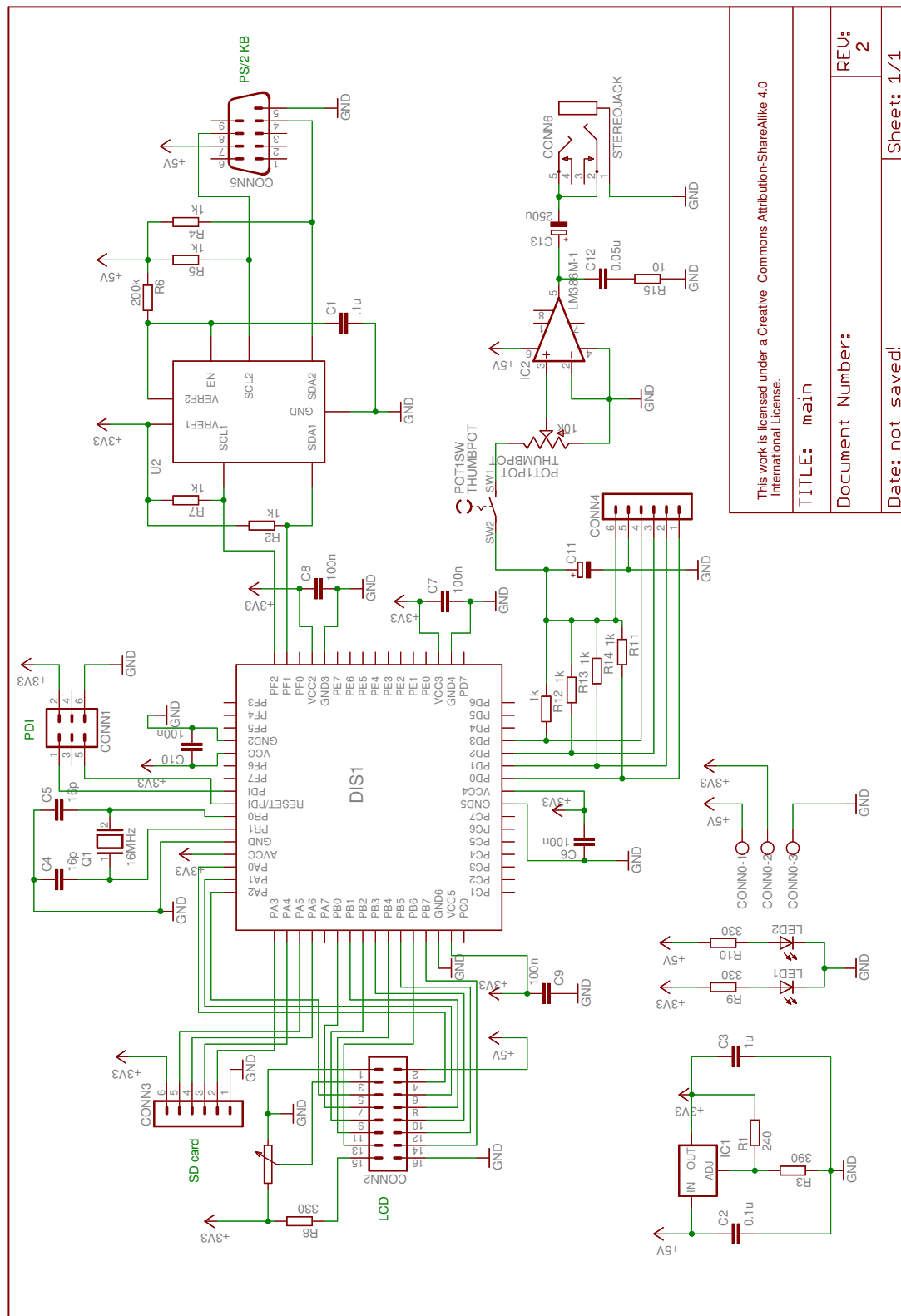


Figure 8: Full Schematic

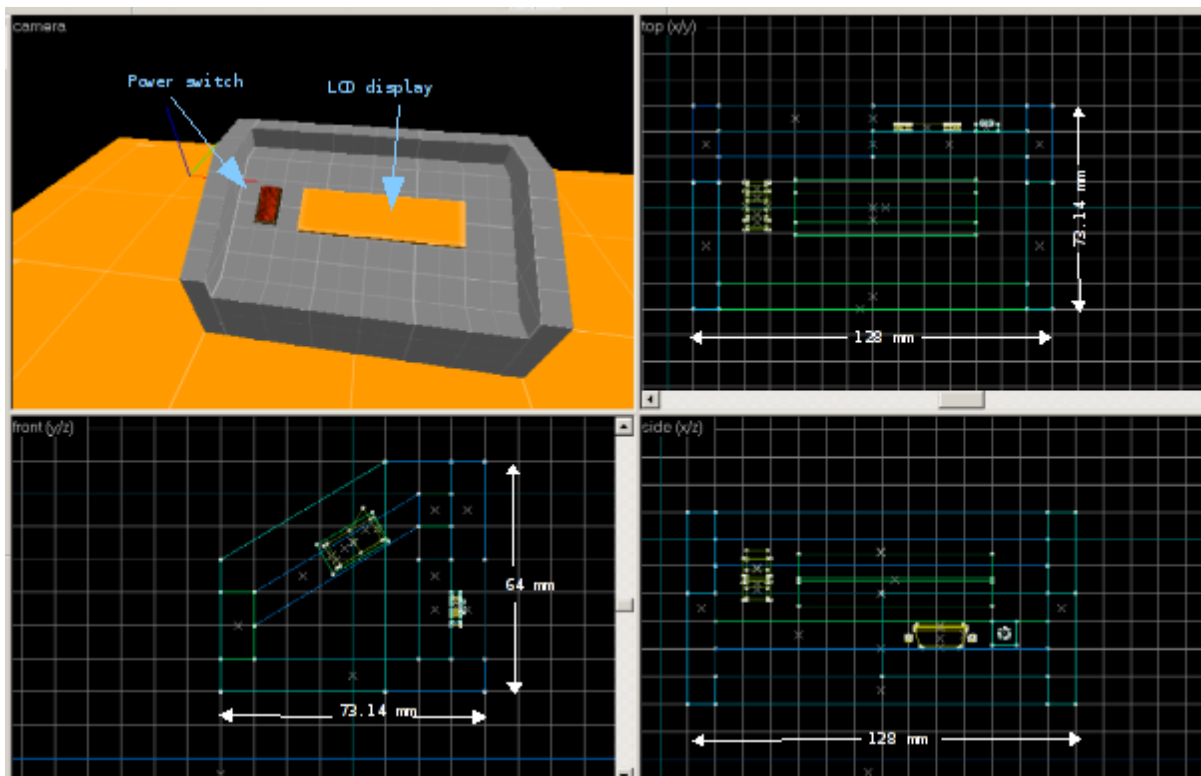


Figure 9: Mechanical schematic of enclosure

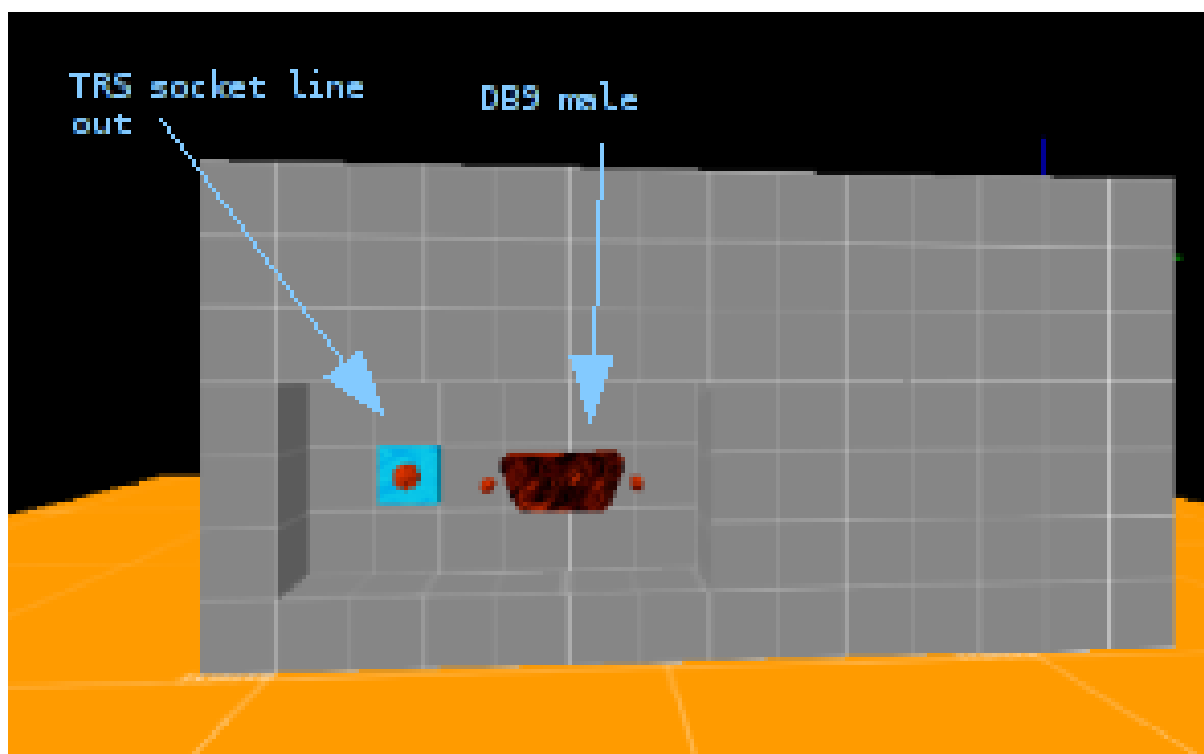


Figure 10: Rear 3D view of enclosure

## F. Large Software Diagrams

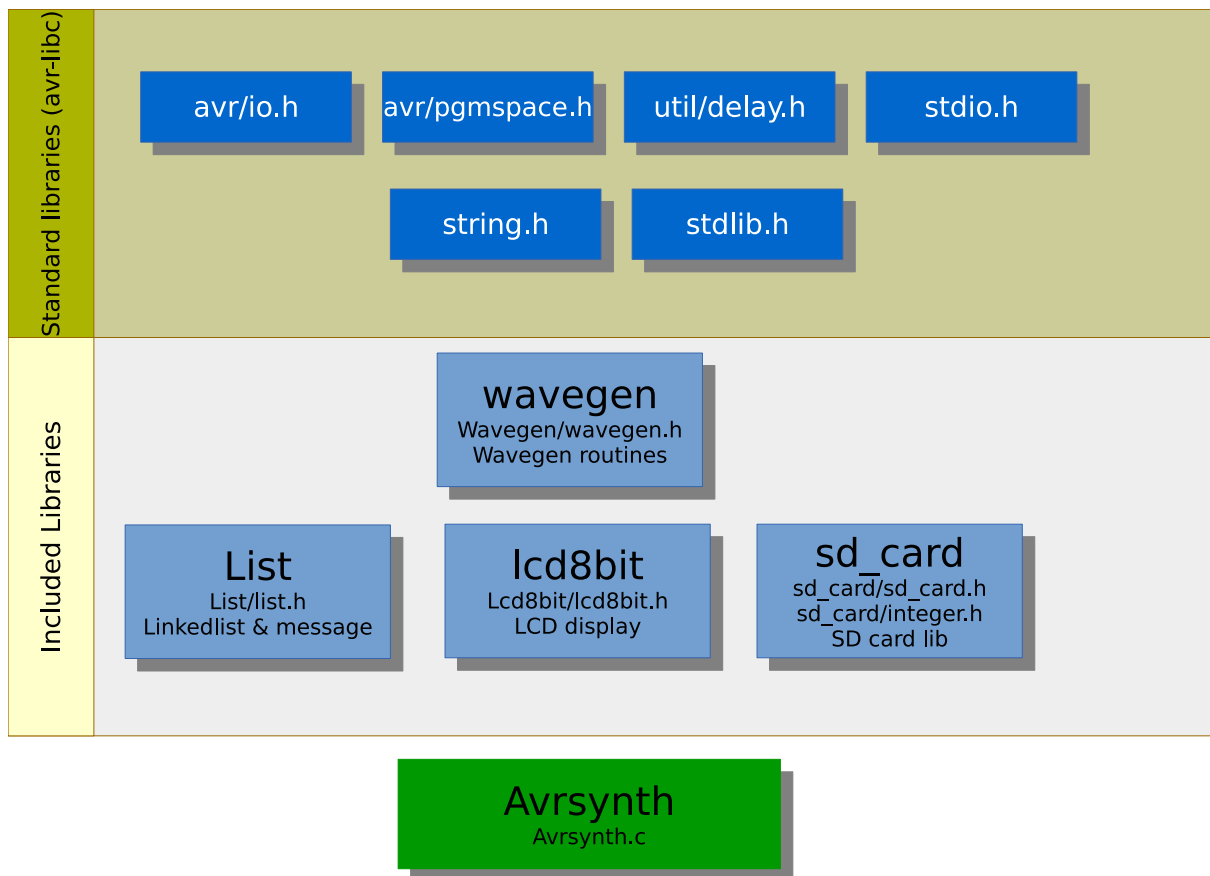


Figure 11: High Level Software Topology

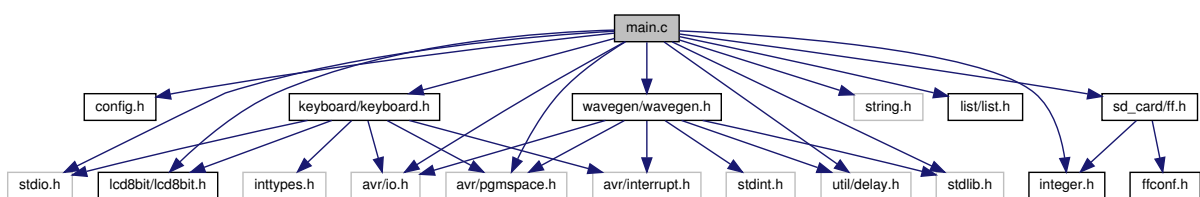


Figure 12: Include dependency graph for main.c

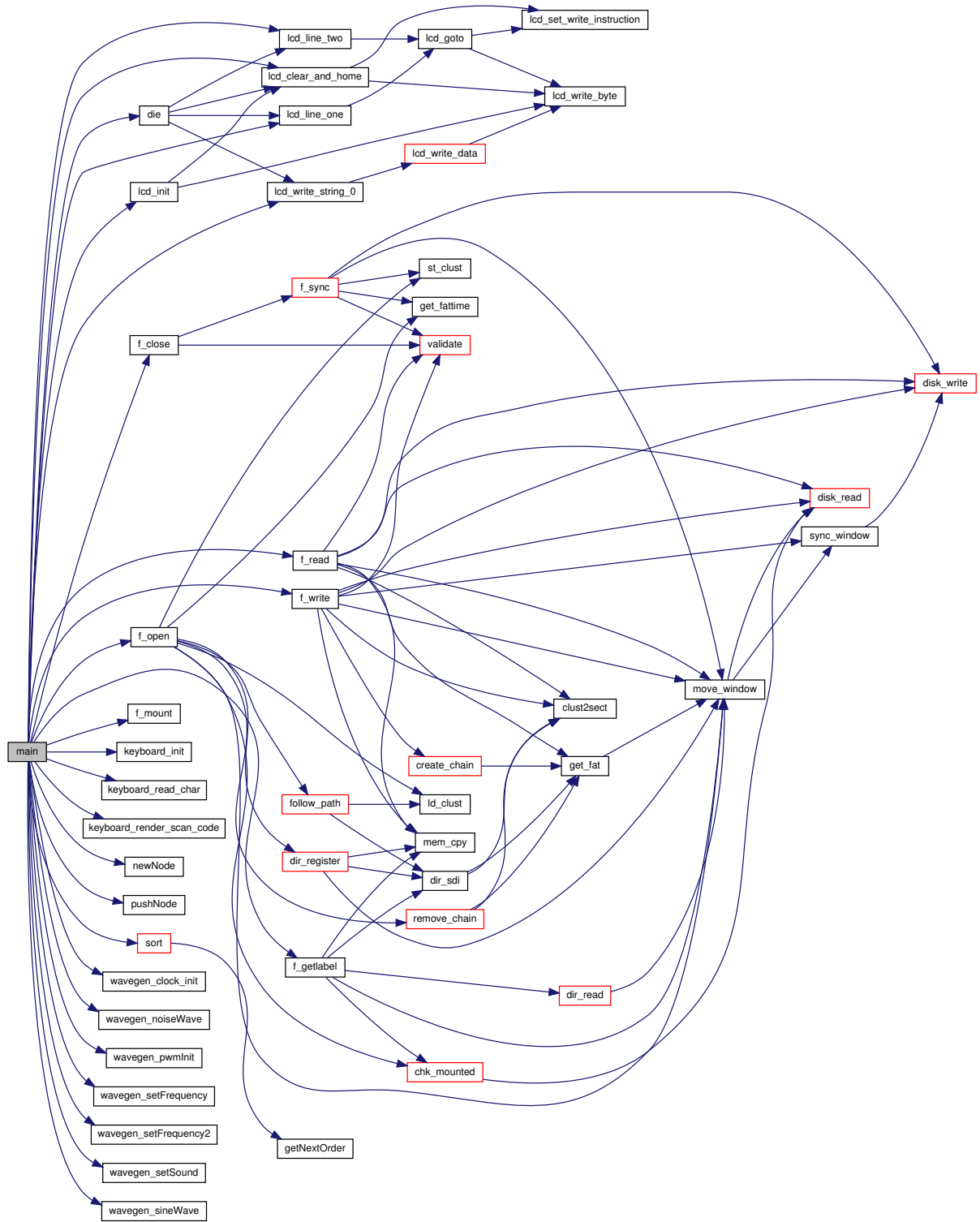


Figure 13: Callgraph for int main()

## G. Large Software Listings

Listing 1: Avrsynth project makefile

```
DEVICE = atxmega256d3
AVRDUDE_DEVICE = x256d3

PROG = -c avrispmkII -P usb

CFLAGS=-g -Wall -mcall-prologues -mmcu=$(DEVICE) -Os
CC=avr-gcc
OBJ2HEX=avr-objcopy
LDFLAGS=-Wl,-gc-sections -Wl,-relax

AVRDUDE=avrdude
TARGET=main
OBJECT_FILES=main.o
SRC_FILES=$(OBJECT_FILES:%.o=%.c)

all: $(TARGET).hex

clean:
    rm -f *.o *.hex *.obj *.hex

%.hex: %.obj
    $(OBJ2HEX) -R .eeprom -O ihex $< $@

%.obj: $(OBJECT_FILES)
    $(CC) $(CFLAGS) $(OBJECT_FILES) $(LDFLAGS) -o $@

program: $(TARGET).hex
    $(AVRDUDE) -p $(AVRDUDE_DEVICE) $(PROG) -U flash:w:$(
        TARGET).hex
```

Listing 2: Snippet from avrsynth.c performing frequency generation from list structure

```
sineWave();
clock_init();
pwmInit();
PORTD.DIRSET=0xFF;
struct order* gett = (getOrderNode(head,1)->data);
setFrequency2(getOrderFrequency(gett));

gett = (getOrderNode(head,2)->data);
setFrequency(getOrderFrequency(gett));
unsigned long count=0;
unsigned int count_ms=0;

//10 sec of music
while(1)
{
    count++;
    if((count == 320000))
    {
        count=0;
        count_ms++;
        //freq step
```

```
|| }  
|| //noiseWave();  
|| //wait here  
|| }
```

## H. Calibration Instructions

N/A



## I. Datasheets

N/A

## J. Subsystem Unit Tests

Table 23: LCD unit test

Test Writer		Matt Bagnara		
Test Case Name		LCD unit test	Type	subsystem
Description	Verify correct communication of ASCII text to display screen	Date	09/25/13	
Setup	Connect LCD tester using pinout displayed in Appendix J	Time	22:12	
Test				
Name	Action	Expected output	Pass	Comments
Poweron test	Apply power to microcontroller	Display lights up?	✓	
ASCII test	No action needed	Will display an ASCII message on LCD	✓	
Control character test	No action needed	Will display custom characters on the LCD	✓	
Contrast adjust	Adjust rheostat on tester board	Contrast should adjust on screen	✓	

Table 24: PS/2 interface unit test

Test Writer		Matt Bagnara		
Test Case Name		PS/2 Keyboard interface	Type	subsystem
Description	Test user input via PS/2 keyboard input		Date	09/20/13
Setup	Connect processor power and setup as in Appendix J		Time	14:22
Test				
Name	Action	Expected output	Pass	Comments
PS/2 poweron test	Power microcontroller	Keyboard LEDs should light up initially and turn off	✓	
Key press event	Press any alphanumeric key on keyboard	LEDs should light indicated scancode. See PS/2 scancode list.	✓	
Key release event	Release any alphanumeric key on the keyboard	LEDs should light indicated scancode. See PS/2 scancode list.	✓	
Contrast adjust	Adjust rheostat on tester board	Contrast should adjust on screen	✓	

Table 25: SD storage unit test

Test Writer		Matt Bagnara		
Test Case Name		SD storage interface	Type	subsystem
Description	Test reading of data from SD interface		Date	10/09/13
Setup	Connect processor power and setup as in Appendix J		Time	17:32
Test				
Name	Action	Expected output	Pass	Comments
SD library initialize	Power microcontroller	LED on PORTC should light	✓	
SD library read block	Power microcontroller	Test should read ASCII data off SD card and turn back LED on PORTC	✓	

Table 26: Synthesis unit test

Test Writer		Matt Bagnara		
Test Case Name		Synthesis unit test	Type	subsystem
Description	Includes conversion of digital wavetable data into a synthesized waveform		Date	10/09/13
Setup	Connect processor power and setup as in Appendix J		Time	17:32
Test				
Name	Action	Expected output	Pass	Comments
Frequency test	Run single tone example	440Hz	X	Still needs further calibration
Frequency sweep	Select Freq sweep in debug menu	sweep from 40 to 10000Hz in 10 seconds	X	current implementation
Noise generation	Run noise-gen routine	continous noise output	✓	
Two tones	Run two-tone routine	440Hz and 460Hz	✓	
Three tones	Run tri-tone routine	440Hz 460Hz 520Hz	✓	
Frequency test	Run single tone example	440Hz	X	Still needs further calibration
Squrare generation	Run square-gen routine	continuous squarewave at 440Hz 50% duty	✓	
Triangle generation	Run triangle-gen routine	continous triangle wave at 440Hz	X	In progress

Table 27: Processor unit test

Test Writer		Matt Bagnara		
Test Case Name		Processor platform test	Type	subsystem
Description	Includes demonstration of processor functionality as a hardware platform		Date	11/10/13
Setup	Connect processor power and setup as in Appendix J		Time	12:27
Test				
Name	Action	Expected output	Pass	Comments
Poweron test	Power microcontroller	1.7V to 3.6V on VCC and AVCC pin	✓	
PDI in-interface advertisement	See <a href="#">Testing Documentation</a>	Should respond with device signature 0x1e9844	✓	

## K. Testing Documentation

### 1. LCD Tester

The data bits are connected to PORTA. The control bits are connected to PORTD[7..5]. A common ground and +5V must be established between the microcontroller and the testing board.

The microcontroller will be loaded with the 'lcd8bit' hex file and will complete the tests automatically upon running. The microcontroller will display a short message on the first line of the display. Following, custom glyph characters will be displayed on the second line of the display. The core functionality of the display such as clearing and modifying will be done using these tests.

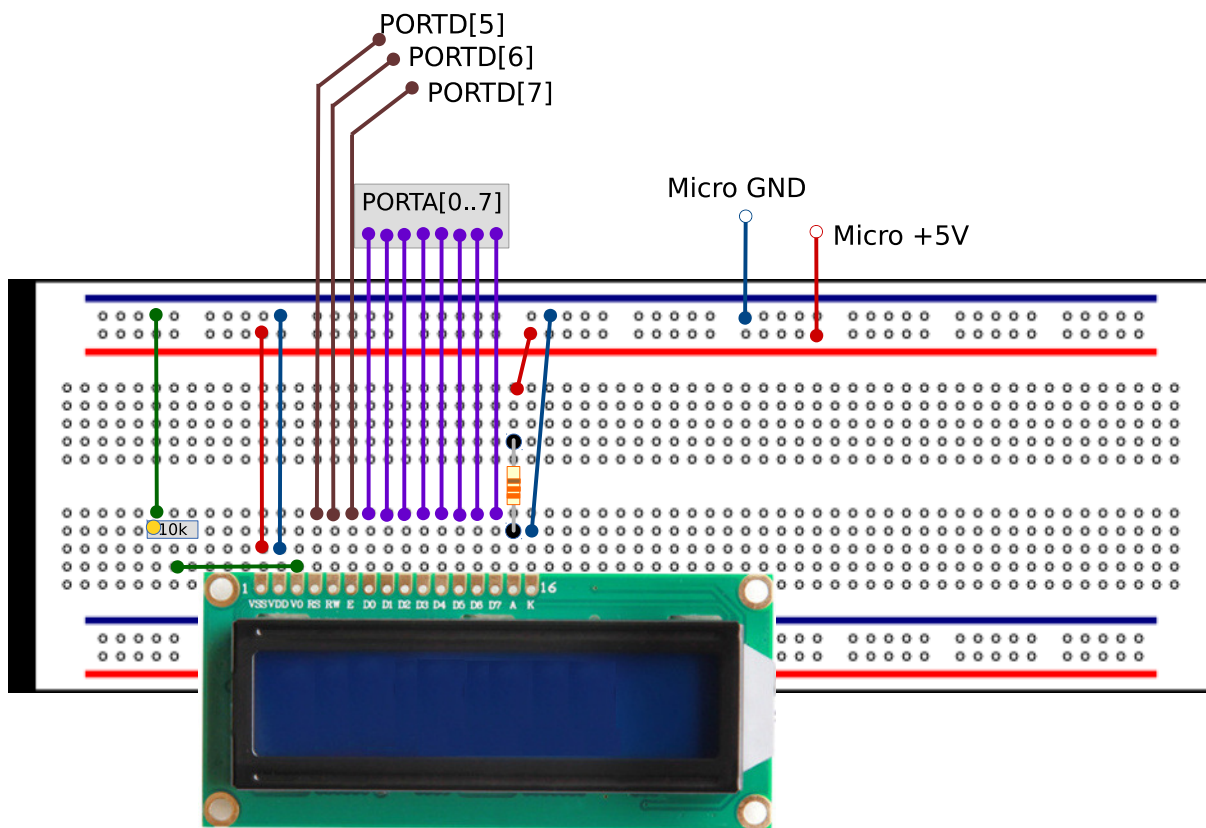


Figure 14: LCD tester layout

### 2. PS/2 Keyboard Tester

The PS/2 keyboard will be connected to the 6-pin mini DIN connector which is coupled to the microcontroller board. The mini DIN requires an adaptor to DB-9 to connect easily for testing. For the usage of the test, a DB-9 to mini-DIN adaptor dongle will be used. The dongle connector is wired as follows:

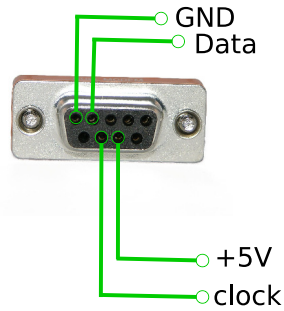


Figure 15: PS/2 Dongle pinout

The opposite side of the connector will be jumpered to the microcontroller board on PORTB; with clock being B.2 and data being B.1. A common ground and +5V must be established between the microcontroller and the testing board.

The PS/2 tester will display the regular scancode for the last key being pressed or released on the LEDs using PORTC. The press and release scancodes are displayed here.

### 3. SD Card Tester

The SD card breakout board will be connected using jumper connections to the microcontroller board. Because the SD breakout board includes a level converter, there is no voltage issue using +5V. The wiring is shown. PORTB is used for to connect to the breakout board. The SD is inserted into the slot when powered down. Only regular SD cards may be used with size up to 1GB.

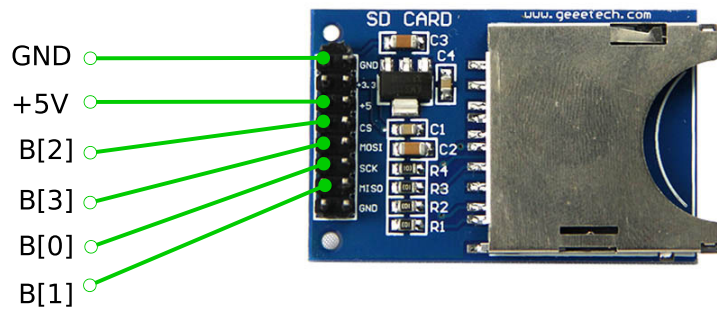


Figure 16: SD board pinout

The microcontroller is loaded with the 'sd\_test.hex' hex file and powered with the SD card in place.

### 4. Processor

The processor testing environment is established through a rudimentary PDI programming interface. The essential build environment requires this interface and a powered processor. With a fundamental environment created, it is possible to verify communication via the PDI programmer. This test will make use of the AVRISP mkII programmer.

The first test requires a verification of the processor device. This consists of a request by the development workstation and a simple reply advertisement by the processor. This can be simply executed by issuing `avrdude -p x256d3 -c avrismkII -P usb` in the terminal of the workstation. A valid response from the processor would be a device signature of `0x1e9844` when using the ATXMEGA256D3.

## 5. Integration Platform

The integration platform was designed for use throughout the subsystem integration phase of the project. Each individual hardware subsystem can be exclusively attached and detached from the platform. This allows comparative testing to be done using any integration regression.

Standard connections are as follows:

Table 28: Development board connections

Connection	Description
CONN1	PDI interface
CONN2	LCD
CONN3	SD interface
CONN4	Wave generation PWM
CONN5	PS/2 I2C interface
CONN6	Production power terminal
CONN7	Power terminal (+3.3V,GND,+5V)
DIS1	Xmega-compatible 64-pin processor assembly card

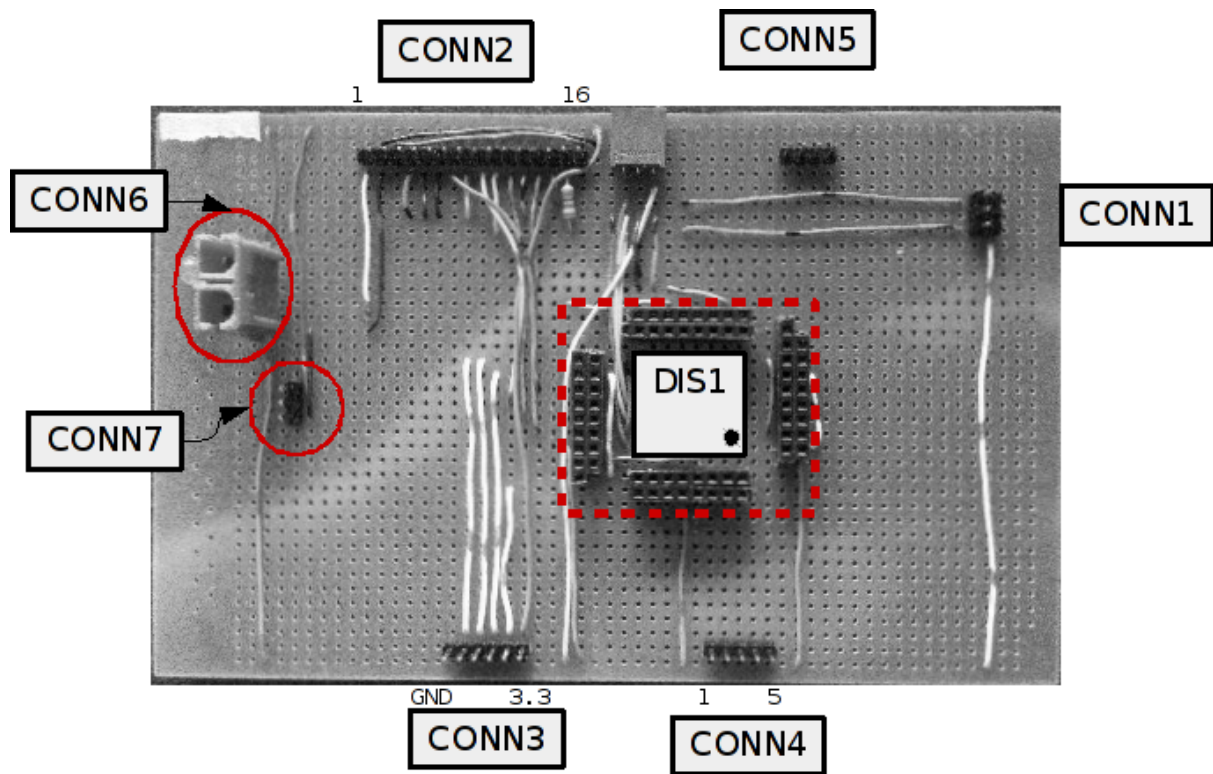


Figure 17: Development board



## L. Licenses

### 1. Hardware design:

The hardware for this project is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.

<http://creativecommons.org/licenses/by-sa/4.0/>



### 2. Software design:

The software for this project is licensed under the GNU General Public License Version 3 (GPLv3):

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.



### 3. Creative Content:

The creative content for this project includes the title, artwork or display of the project, graphics, illustrations, pictorial representations, etc. All are licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.

<http://creativecommons.org/licenses/by-sa/4.0/>



### 4. Documentation:

The documentation for this project includes development documentation, testing documentation, written or digital, and this report. All are licensed under the Creative Commons Attribution 4.0 International License.

<http://creativecommons.org/licenses/by/4.0/>

