Critical Design Review (CDR)

Project Description

This project will design and implement a Smarthome lighting and ceiling fan control system. Controls will allow light fixtures to vary their brightness in a dimmable fashion, and ceiling fans driven by single-phase AC induction motors will be speed controllable.

Control messages will be sent wirelessly. The wireless communication will be using the Zigbee Mesh protocol. Xbee Series 2 modules will be used, in API mode and with built-in encryption capability enabled.

Wall “switch” controls will be small LCD touchscreens, communicating with the items they control via wireless command messaging. The LCD will allow the user to select a room, and then which target item in that room to view current status of, or to change it's state. (ie. The selected item in the selected room will get brighter, dimmer, slower, faster, etc.)

The target items being controlled will use a microcontroller as the Zigbee command receiver. Dimmable light fixtures or speed-controllable motor-driven target items will use a common Triac-based circuit to control the target as desired by the user.

A central computer server will log events, control “vacation mode” away events, and act as a gateway for automatic and manual controls sent from distant control devices carried by the home residents. This server will also act as the Zigbee wireless network “coordinator” node, in charge of the Zigbee mesh network. This server will also have an LCD touchscreen panel for

Android smartphones will offer both automatic and manual controls sent via the Linux server, and also a way to remotely view the server's log data in text form. Current status of the home targets will be visible in a GUI which mimics that seen on the LCD touchscreen wall “switches”. The mobile devices will primarily use a GPS based geofence to indicate whether the user is home or away. An alternative control method will be with the use of NFC tags at the front door of the home, so the user can manually swipe his or her mobile device across a tag to manually indicate that they are leaving or returning home. These two control capabilities allow the server to automatically enter and leave an automated, quasi-random “vacation mode”, where it will turn lights on and off at reasonable times to emulate the residents being home for added security from burgulars.

This project will also be designed to allow future expansion, with possibility to add new kinds of target items to control and new types of controls. The intention is to complete a project comparable to popular X-10 systems, but which will send control commands wirelessly instead of signaling over power line. Some example

ideas for future additions are garage doors, electronic entry door locks, thermostat HVAC environment control, or safety shutoff for electrics ovens or lab soldering stations, in case you forget to turn them off when you leave home, but these examples are not included in this semester's project effort.

While the ideal would be for the wall “switch” controls to fit into a standard Decora ™ wallplate and housing gang area, and target controls would fit into common ceiling fan wiring canopies or other light fixture wiring boxes, this project during the semester will be made using prototyping computing and interface boards, which will not cleanly fit such common switch housing, and so will not implement a cosmetic ideal. To cosmetically fit into the popular housings would require custom PCB design and assembly to compact the various components used, as well as alternate choice of LCD touch panel of proper size, which may not be supported by the intended Arduino code libraries available, and require software porting to make use of. These additional tasks are beyond the scope of this semester's effort to create a working prototype.

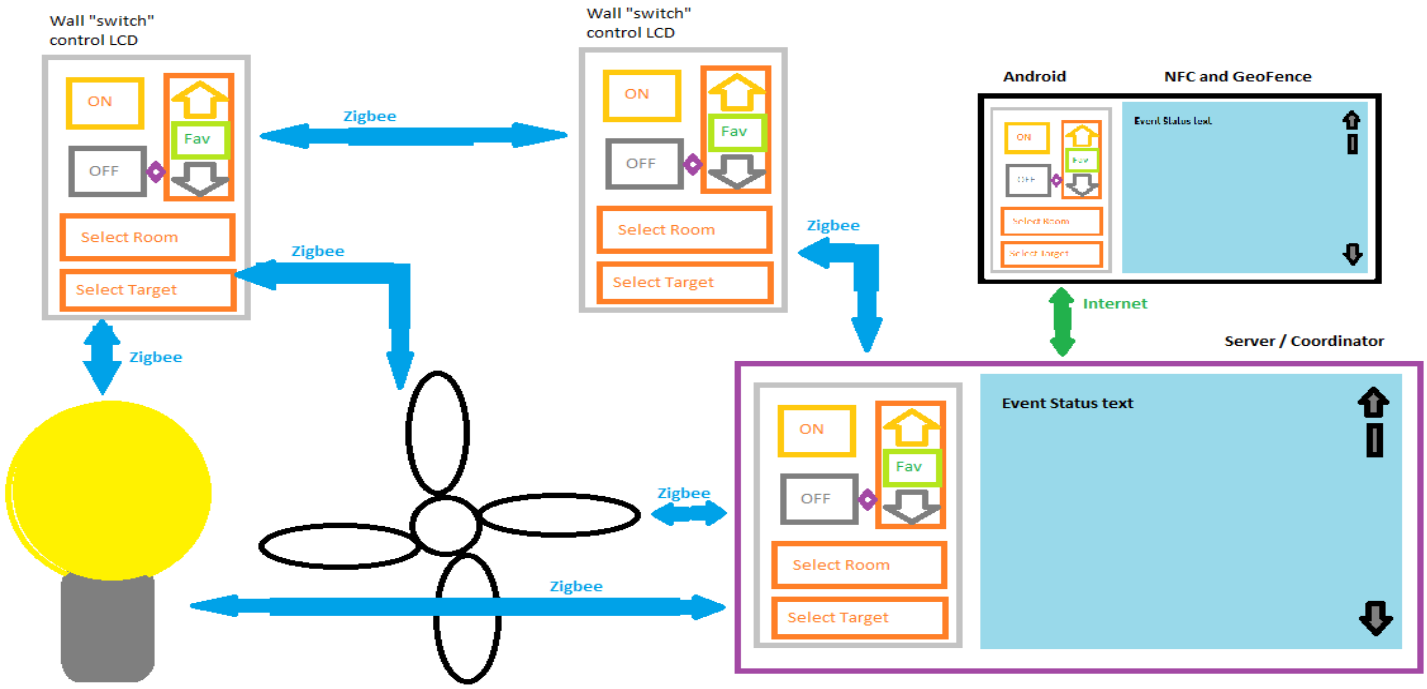
This project evolved from an initial concept fo a wireless controlled light switch, and complexity was added to bring that up to an acceptably complex project for this course.

This project is an individual effort, and does not depend on other projects, nor do other projects depend on this.

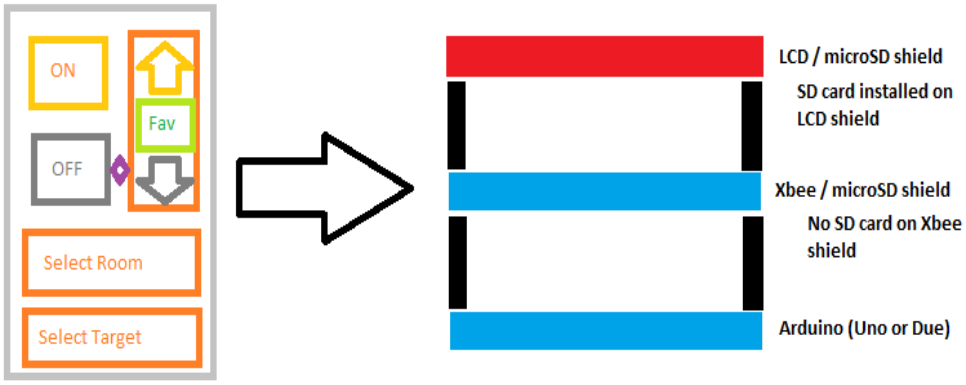
Functional Description

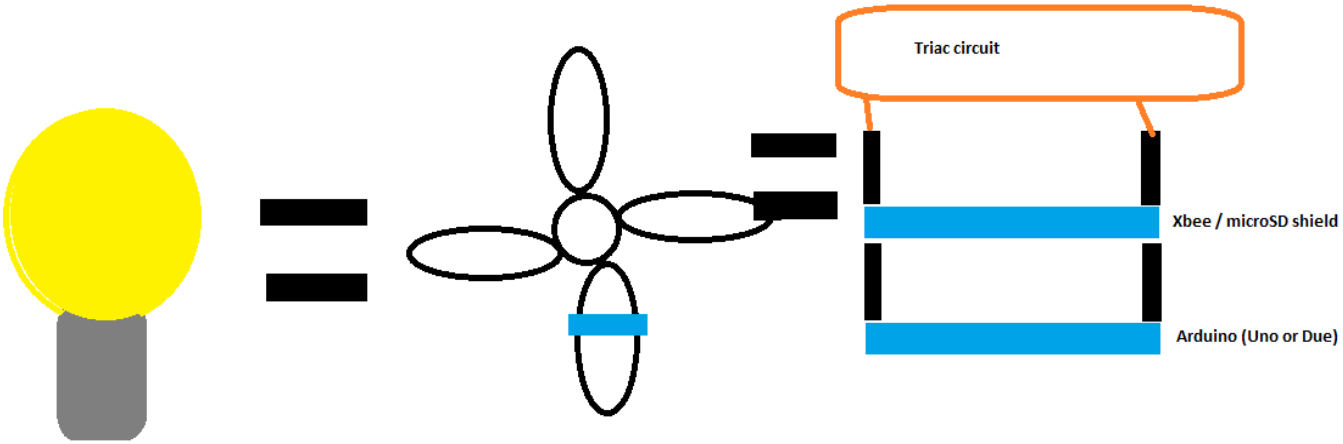
TODO

Block diagrams



The Linux Server and the Android app will display the same control panel as described above for the wall controls, and additionally TODO





Arduino pinouts

Wireless/SD	LCD/SD	Due	Uno R3	pin ID		pin ID	Uno R3	Due	LCD/SD	Wireless/SD
						17				
						16				
						15				
						14				
				NC		13			SD_SCK	SD
				IOREF		12			SD_DO	SD
				Reset		11			SD_DI	SD
	3V3			3V3		10			SD_SS (CS?)	
	5V			5V		9			LCD_D1	
				GND		8			LCD_D0	
	GND			GND						
				Vin		7			LCD_D7	
						6			LCD_D6	
	LCD_RD			A0		5			LCD_D5	
	LCD_WR			A1		4			LCD_D4	SD_CS
	LCD_RS			A2		3			LCD_D3	
	LCD_CS			A3		2			LCD_D2	
	LCD_RST			A4		1				TX
				A5		0				RX
			x	A6						
			x	A7		14	x			
						15	x			
			x	A8		16	x			
			x	A9		17	x			
			x	A10		18	x			
			x	A11		19	x			
			x	DAC0		20	x			
			x	DAC1		21	x			
			x	CANRX						
			x	CANTX						

Material/Resource Requirements

There are no dependencies from other projects. This project does not provide any dependencies to other projects. It is a completely independent effort.

This project is an individual effort. So there is no particular assignment of certain tasks between multiple team members, all project tasks are assigned to this single individual, Bill Toner.

There are three major components of the smarthome system. There are also some general household items, such as light fixures, ceiling fan units, wiring boxes and power cabling which will be required.

Both the controls and the targets in the Arduino class will require an Arduino microcontroller board and the Wireless/SD shield, which holds an Xbee module and possibly a microSD card. The controls will also require an LCD touchscreen/SD shield, and the targets will also require the triac based AC line control circuitry. The target end will also require a couple lights and a ceiling fan to control. An Xbee module will be required for each Arduino and target node. A 2.4inch LCD touchscreen+SD module was chosen so that these prototypes are close to a standard Lenovo Decora™ wall-plate slot. A commercial product would need to refine into a more exact LCD package.

I have not yet purchased components for the triac AC line control circuits, as I will need more time to evaluate available app notes and examples to determine appropriate part numbers to use in this design. I will do this during the weeks leading up to my scheduled time to work on this part of the system, leaving a week or so for shipment and delivery. If delivery is for some reason delayed a little, then I will begin assembly of the Fan and light fixtures with some wiring and electronics enclosures in preparation of the triac circuits coming, and then begin work on the Android app until the AC control components are received.

I already have a couple Arduino Uno boards, have ordered one each of a few items for fast delivery, and ordered additional copies of some items at lower cost with slower delivery. I have now received Wireless+SD shield and LCD touch/SD shield. I also have ordered a few Arduino Due boards, and already have one in possession. Additional Unos and Dues are in shipment from Ebay sellers located in China. I have already received some items from Chinese sellers, and anticipate the remainder of my orders arriving by the end of the first week in October, 2015.

The Linux server will be built using an AMD Gizmo embedded computing board, an 7 inch LCD touchscreen, a USB WiFi dongle, an Msata storage card, and some custom connectivity to the Xbee module. I already have the Gizmo kit. I had purchased a Gizmo kit a few months ago in order to get an AMD jtag probe for unrelated personal interest. I will likely need to do some breadboarding

For the Android class, I will be using the smartphones that My wife and I already own and use. I will also use a home Wifi network router that I already own as the backbone of my simulated internet between the Android devices and the Linux server.

I have acquired a couple simple, inexpensive light fixtures, wall electrical boxes, and an inexpensive ceiling fan unit, to act as my targets to be controlled by this system. If this fan unit, the least expensive one I could find amongst my local hardware stores, proves unsuitable, I do have another ceiling fan unit of higher quality pending a small home improvement project, which could be used as a backup in this project. I don't expect this to be necessary as there are several online examples and app notes in general agreement for how to use triacs to control ceiling fan motors. While a few discussions indicate that a capacitor and/or frequency based speed control circuit may be more ideal, I have not found any app notes or examples for this approach, but I will keep it in mind as an alternative should I have significant problems with the triac design. I will initially focus on the triac concept due to apparent popularity.

I plan to assemble the triac circuit I will use in an Ltspice simulation. Then, ocne satisfied with function and performance, I will use a suitable oscilloscope to observe the AC waveform applied to the initial test load, by a prototype hardware

setup with an inexpensive incandescent light bulb. This will start out as an Arduino Uno being hardcoded for speed and/or being controlled by a PC laptop. I also have a commercial wired style light dim and fan speed control pending a home improvement project, which I can observe with a scope to learn behavior if I have problems with triac and need some ideas.

Development Plan

The source code, circuit diagrams, documentation, and any other digital data related to this project will be committed and tracked using a Design Management (or Configuration Management in some vocabularies) repository using the GitHub service, located at the link below. This is set as a private repository, and I'll add professor and class members who may be interested, on request.

https://github.com/amigabill/jhu_EN525.743

I plan to work on the various system components in the following order:

1. Arduino
    - SD card storage (contains some node configuration data)
    - Zigbee communication (With PC laptop as coordinator at this point)
    - Target control of AC line triacs (with at least some visible output indication, such as a small LED, in place of final triac circuit)
    - Touchscreen input events, using some pre-planned location for test “buttons”
    - LCD GUI implementation
    - Anticipating the possibility that the available libraries I plan to use may not all fit together into an Arduino Uno memory space, I plan to use Due boards for the LCD touch control nodes. Uno's are likely to be suitable for the target nodes, as those do not require LCD graphics related libraries or touch drivers, and will have a smaller memory footprint.
2. Linux Server (AMD Gizmo embedded project board)
    - Zigbee coordinator
    - Communicate with targets
    - Communicate with wall touchscreens
    - Implement event logging
    - GUI using WxWidgets library
    - Networking with internet, using a home WiFi router to emulate the internet. (Overlaps with Android App communicating with Linux server)
3. Android smartphone app
    - Communicate with Linux Server
    - GUI indication of targets current states
    - GUI manual control of targets

- NFC tag manual location indications (easier to demo than Geofence)
- GPS Geofence locations (somewhat harder to demo than NFC)

The anticipated schedule is shown below:

Milestone Number(s)							1	2			3	4				5
week of semester (Fall 2015)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
date	20150831	20150907	20150914	20150921	20150928	20151005	20151012	20151019	20151026	20151102	20151109	20151116	20151123	20151130	20151207	20151214
First class - initial ideas																
Second class - review ideas and commit																
PDR																
CDR (Critical Design Review)																
Software tools install																
Arduino SD card data access																
Arduino Zigbee communication																
Arduino LCD touch sensing																
Arduino LCD graphics																
Server Zigbee																
Server event logging																
Server vacation mode events																
Server LCD panel																
Server LCD touch																
Arduino triac interface for lights/fans																
Android connection to server/status/logs																
Android manual events to server																
Android NFC events to server																
Android geofence events to server																
Demo																
Travel for Work																

There are 5 major milestones.

1. Arduino Uno target controller
2. Arduino Due wallplate LCD touchscreen controller
3. Linux Server/Coordinator
4. AC line load controller circuit, which is controlled by #1 above.
5. Android Smartphoen Application

It's important to start with the Arduino targets and controls, as without them there is nothing for a Linux server or Android app to do. The next major priority is the Linux based server, to coordinate the other Zigbee nodes, observe and control target events, etc. This server along with the Arduino nodes then make up the bulk of this project's core functionality. Once the server is complete, then I will implement the triac circuitry to control the AC line to a real target light or fan motor in place of the earlier placeholder status LED. At this point, within the home is a fully functioning system with “real” targets working. Finally, I will move on to implementing the Android app. Should scheduling slip, the Android app is not as critical as the Arduino nodes or Linux server roles in this project, and so if anything should not get completed, this app is the most reasonable choice to be worked on last.

Some modifications may be required to the shields to improve signal quality on Arduino pins used on both modules, such as D4 which is SD_CS on the Xbee/SD shield, and an LCD data pin on the LCD shield. Cutting the D4 and other SD slot traces on the Xbee/SD shield, and not using SD on the Xbee/SD shield can help. The LCD shield has another microSD slot on it, using D10 for SD_CS instead of D4.

There are some risks in this schedule.

- I have not worked with Arduino or AVR microcontrollers before. During my previous experience with Atmel Sam-4 ARM devices, I worked some with SD protocol, in both C and Verilog, which would be beneficial to this project. I also worked with other peripherals such as Usart and Duart, parallel ports, interrupts, and I also had experience programming the Rabbit3000 and Xilinx Microblaze in prior courses at Johns Hopkins University, to show that I can move to different library APIs/ABIs in an acceptable timeframe for comparably complex projects.
- I have not worked with wxWidgets for a GUI application in Linux before. There are Rapid Application Development tools available, which I will make use of to assist in building the general code framework and creating the GUI interface. There is a thriving online support community for wxWidgets, and I have been looking at the API a little while pondering an unrelated project to port the wxWidgets framework to an operatign system it is not currently available for, so I'm not going into that with a completely clean slate.
- I have a small amount of experience with internet networking, such as the communication between Android app and Linux server. This experience is from the JHU course EN.525.742 System on Chip FPGA Design Lab, which I took a couple years ago. There I learned about ethernet message packets using Xilinx Microblaze microcontroller and a minimal library to hit the hardware, and the use of WireShark for debugging packets coming out of the FPGA board. I will be sending simpler messages in this project.
- How to demo geofence within range of WiFi router in the classroom? I will see if I can bribe my wife into being a remote participant during demo night, using her Android device over cell data network communicating with the Linux server via the WiFi hotspot in my own Android device, which will act as my in-classroom “home internet router” for the Linux server.
- I have not programmed in Java before. I plan to use a Rapid Application Development tool to assist with general code framework and creating the GUI interface. From there, I will have to be able to take a general pseudocode of my indended application and quickly learn the Java syntax to accomplish that design. Online Java examples abound, and there are several online support communities available when I have a question.
- During week 5, after the PDR presentation, my supervisor at work has asked me to attend a week-long training seminar in the month of November, along with everyone else in the group under him. It will be difficult to keep up with my normal work, the training, evening group

meetings and other activities, and also continue a good pace on this project. I have requested permission to take a few days vacation to focus on this project to compensate, and also described the situation regarding this course and scheduling, and await a reply to this request. I rarely take time off work, I now have a second person assisting in my role, and my overlords have a history of being very supportive of the time off requests that I have made in the past, so I do not anticipate that this training event will have a significant impact on this project schedule.

References

The programming environments, operating systems, and support libraries that I plan to use are all available at no cost. There are no expensive licenses to buy or to expire at an inconvenient time.

Programming IDEs, SDKs, and Frameworks:

- Arduino - https://www.arduino.cc/en/Main/Software
- Android Studio/SDK for Java - https://developer.android.com/sdk/index.html
- Android NDK for C/C++ - https://developer.android.com/ndk/index.html
- Linux for AMD Gizmo board - http://www.timesys.com/register/gizmo
- wxWidgets (for Linux GUI in C++) - http://wxwidgets.org/

I plan to make use of available libraries as much as possible for various components of the software.

Arduino Libraries:

- Arduino SD library - https://www.arduino.cc/en/Reference/SD
- Arduino TFT Library for LCD panel driver, or relevant equivalent for the LCD controller used - https://www.arduino.cc/en/Reference/TFTLibrary
- Yamsam LCD https://github.com/yamsam/TFTLCD_ST7781

Arduino Shield documentation:

- Xbee+microSD - https://www.arduino.cc/en/Main/ArduinoWirelessShield
- LCD touch+microSD
    - http://www.smokeandwires.co.nz/blog/a-2-4-tft-touchscreen-shield-for-arduino/ (for same LCD controller info and library)
    - https://github.com/yamsam/TFTLCD_ST7781
    - SmokeandWires LCD http://www.smokeandwires.co.nz/blog/a-2-4-tft-touchscreen-shield-for-arduino/
    - Adafruit Graphics Core library - https://learn.adafruit.com/adafruit-gfx-graphics-library/overview
-

Digi Xbee

- http://ftp1.digi.com/support/documentation/90000976_W.pdf
- http://knowledge.digi.com/articles/Knowledge_Base_Article/What-is-API-Application-Programming-Interface-Mode-and-how-does-it-work
- https://github.com/andrewrapp/xbee-arduino
- http://ftp1.digi.com/support/utilities/digi_apiframes2.htm
- http://arduino.stackexchange.com/questions/1500/how-to-make-xbee-module-interrupt-wake-arduino

Ltspice

- http://electronicdesign.com/power/model-diacs-and-triacs-ac-line-control
- http://electronics.stackexchange.com/questions/87546/ltspice-mac97a8-triac-model-vgs-error

There are several online app notes and example designs which will be relevant to this project:

- Atmel AVR datasheet (Arduino Uno) - http://www.atmel.com/Images/doc8161.pdf
- Atmel SAM3 Cortex-M3 datasheet (Arduino Due) - http://www.atmel.com/Images/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf
- Triac AC line control:
- **Microcontrollers: From Assembly Language to C Using the PIC24 Family** http://www.freescale.com/files/microcontrollers/doc/ref_manual/DRM039.pdf
- http://cache.freescale.com/files/microcontrollers/doc/app_note/AN3471.pdf
- http://www.st.com/web/en/resource/technical/document/application_note/CD00003820.pdf
- http://ww1.microchip.com/downloads/en/AppNotes/91094A.pdf
- http://www.mouser.com/catalog/specsheets/stevalill004v1.pdf
- https://books.google.com/books?id=ndALAAAAQBAJ&pg=PA683&lpg=PA683&dq=120V+AC+triac+3.3v&source=bl&ots=I3s7rlZYNT&sig=KcAZ0yObYB2d9QJC0kRAYhIQLkU&hl=en&sa=X&ved=0CCYQ6AEwAWoVChMItZzzj_yPyAIVgyaUCh0C8Az7#v=onepage&q=120V%20AC%20triac%203.3v&f=false

LCD touchscreen for Linux Server

- http://www.sainsmart.com/7-inch-tft-lcd-monitor-for-raspberry-pi-touch-screen-driver-board-hdmi-vga-2av.html

1. Project description
    a. High level description of project
    b. Brief description of how your project fits into a larger system, if applicable
    c. Capabilities
    d. Limitations
2. Functional description
    a. Functional block diagrams, and or
    b. State machines, and or
    c. Architectural diagrams
3. Interface descriptions
    a. Internal (within your project)
    b. External (to elements you don't control, such as public internet or partner project)
4. Material and resource requirements
    a. Everything you need to complete the project
        i. Development tools
            1. Any special build, or installation, instructions
        ii. Test equipment
        iii. Knowledge base
        iv. consumables
    b. Where its coming from?
    c. When is it available?
    d. Items expected of you by other students (team projects)
5. Development Plan and Schedule
    a. Approach
    b. Order of development
    c. Milestones & schedule
    d. For team projects
        i. Clear delineation of responsibilities & dependencies
        ii. Clear description of interfaces
        iii. Integration and test plan
    e. Risk
        i. Identify known risk areas
        ii. Present mitigation plan
        iii. Does schedule reflect risk areas?
6. References
    a. Reuse of existing material encouraged, as in real life.
    b. Cite all existing work that will be used in your project
7. Assembly instructions (final project document only)
    a. Instructions to set up development environment
    b. Describe basic steps to assemble from scratch (i.e. bare metal rebuild)
8. Performance and measurement data (final project document only)
    a. Provide screen shots, test data, measurement data as necessary to show functionality of each subsection, and of the entire system.

After the PDR, this document will evolve into your CDR document and then later to the document which satisfies the documentation requirements for this course. It is a living document which will evolve and grow as your project progresses. Both CDR and final document should address, at least, the items outlined above. The evolutionary cycle is outlined below.

- The PDR presentation (MS PowePoint Brief) is a high level description of your project.
- The PDR charts will become the CDR design document (MS Word format), which is a more detailed version that includes any changes you might have made since the PDR. ***The CDR doc should contain narrative for each section.*** You will submit this document, but will not present it to the class.
- After CDR submittal, the remainder of the course will be dedicated to development. The development activity will be documented for inclusion in the project document. The CDR document will be updated to reflect changes and expanded to include assembly and performance data.
- The final project document should provide enough detail that any of your peers could modify/improve/incorporate your design without interacting with you.