

Nuclear A2D Design

Group #9

Fall 2013

DESIGN PAPER

Kristen Berman

Cassandra Todd

Joseph Nichols

Michael Zellars

University of Central Florida



TABLE OF CONTENTS

1.0 Executive Summary	1
2.0 Project Description	2
2.1 Project Motivation and Goals	2
2.2 Objectives	2
2.3 Project Requirements and Specifications	3
3.0 Project Research Related to Definition	5
3.1 Nuclear Analog to Digital Conversion	5
3.2 ACTIVE Lab Power Plant Simulator Research	8
3.3 Analog Devices	11
3.3.1 Push Buttons	11
3.3.2 Gauges	12
3.3.3 LED's	14
3.3.4 Rotary Switches	18
3.3.4.1 3-D Printing Basics	19
3.4 Analog to Digital Signals	21
3.5 Device Communication	25
3.5.1 Synchronous Interfaces	25
3.5.1.1 SPI Communication Protocol	26
3.5.1.2 I2C Communication Protocol	28
3.5.2 Asynchronous Interfaces	30
3.5.2.1 UART Based Interfaces	32
3.5.2.1.1 Serial Data Standards	32
3.5.2.2 USB Protocol	33
3.5.2.3 Ethernet Protocol	33
3.6 Networking with UDP Data Transmission	34
3.7 Electrical Design	37
3.7.1 Microcontrollers	37
3.7.1.1 Freescale HCS12 Family	38
3.7.1.2 megaAVR Family of Microcontrollers	41
3.7.1.3 Microchip PIC32 Family	43
3.7.1.4 Microcontroller Decision	47
3.7.1.5 In-System Programming	47
3.7.2 Power Supply	49
3.7.2.1 Power Supply Technologies	50
3.7.2.1.1 Rectifiers and Filters	51
3.7.2.1.2 Chopper Circuit and Feedback	53
3.7.2.1.3 DC-DC Transformer Topologies	54
3.7.2.1.4 Output Rectifier and Filter	56
3.7.2.2 Power Supply Design Tool	56
3.7.3 Printed Circuit Board	57
3.7.3.1 Anatomy of PCB	57
3.7.3.2 Layers	57
3.7.3.2.1 Layer Selection	58
3.7.3.2.1.1 Via's	58
3.7.3.3 Design Overview	59

3.7.3.3.1 CAD Packages	59
3.7.3.3.2 Choosing a Manufacturer	59
3.7.3.3.2.1 PCB Specifications Example	60
4.0 Project Hardware and Software Design Details	61
4.1 Aesthetics	61
4.1.1 Hard Panel Housing Unit	61
4.1.2 Control Device Labels	62
4.2 Hard Panel Devices	63
4.2.1 Gauges	63
4.2.1.1 Stepping Motor	64
4.2.1.2 MCU and Shift Registers	66
4.2.1.3 Detailed Component Design	67
4.2.2 LEDs	69
4.2.2.1 LED Assignments and Description	69
4.2.2.2 LED Hardware Design	70
4.2.2.2.1 LED Block Diagram and Schematic	74
4.2.2.3 Panel Mounted Light Hardware Design	77
4.3 Microcontrollers	78
4.3.1 Using the ATmega Family	78
4.3.2 ATmega325 (Master MCU)	79
4.3.2.1 Master/Slave Configuration	79
4.3.2.2 Push Button/Rotary Switch Configuration	81
4.3.3 Power Supply Configuration	82
4.3.4 PC Communication	82
4.3.5 ATmega8 (Slave MCU #1)	84
4.3.5.1 Master/Slave Configuration	85
4.3.6 ATmega32 (Slave MCU #2)	85
4.3.6.1 Master/Slave Configuration	86
4.3.7 Programming AVR Microcontrollers	86
4.4 Communication Between Components and Devices	87
4.5 Printed Circuit Board	91
4.6 Power Supply	92
4.6.1 Power Breakdown	94
4.7 Software	95
4.7.1 Soft Panel Software	95
4.7.1.1 Application	95
4.7.1.2 Graphical User Interface	98
4.7.2 Power Plant Simulator Software	99
4.7.3 Hard Panel Software	100
4.7.3.1 Master MCU Operational Scenarios	101
4.7.3.2 LED MCU Operational Scenarios	104
4.7.3.3 Gauge MCU Operational Scenarios	104
4.7.3.4 Scenario Summary	104
5.0 Design Summary of Hardware and Software	104
6.0 Project Prototype Testing	109
6.1 Hardware Test Environment	109

6.1.1 ESD	109
6.1.2 Temperature	110
6.2 Hardware Specific Testing	110
6.2.1 Individual Components	110
6.2.1.1 Push Buttons	110
6.2.1.2 Gauges	110
6.2.1.3 LED's	110
6.2.1.4 Rotary Switches	111
6.2.1.5 Power Supply	111
6.2.2 Panel System	111
6.3 Software Test Environment	111
6.4 Software Specific Testing	113
6.4.1 Connectivity to Hard Panel	113
6.4.2 Connectivity to Soft Panel	114
7.0 Administrative Content	114
7.1 Milestone Discussion	114
7.2 Budget and Finance Discussion	116
7.3 Work Roles and Distribution	117
8.0 Operation of System	119
8.1 Prerequisites	119
8.2 Communication Throughout the System	120
8.3 Control Devices and What they Do	120
Appendices	
Appendix A: Copyright Permissions	121
Appendix B: Works Cited	126
Appendix C: Listing of Tables and Figures	127

1.0 Executive Summary

Ever since the age of nuclear power plants began, the industry has been centered in the analog world. Control rooms consist of many sensitive devices where a simple bump into a panel could set off a crisis. There has definitely been a need to update the technology of the control rooms however the industry is favored towards the old saying “if it ain’t broke, don’t fix it”. The nuclear industry establishes technology that works and is dependable, but is hesitant to mess with established technology.

The Nuclear Regulatory Committee (NRC) is currently headlining the transition towards the digital age in the American nuclear power plant control rooms. This digital transition would consist of multiple touch screen panels that would collectively act as all of the analog panels currently in use in the control rooms. These panels would have all of the abilities of the analog panels, but would establish an updated and user friendly environment.

The current concern in the industry is in the transition phase itself. As previously mentioned, the plants and their workers are very hesitant towards utilizing developing new technology. In order to implement this system a new training program would have to be established and so it is easy to see how there would be a higher result of errors during the initial transition to digital panels. Our goal is to help analyze the best way to make the user comfortable with this transition and to help the ACTIVE Lab (Applied Cognition and Training in Immersive Virtual Environments) research group with evaluating the risks involved. Our task is to make a physical panel which will connect with the soft panel application in order to accept instructions from the control room simulator. From now on we will refer to this physical device as the hard panel, and our software application as our soft panel. Their focus as a research group is to enhance training capabilities through academic advancement of technologies. The primary initiative of the research group for this specific project is to research the physiological and human-factor effects of this transition. Essentially they are looking to use both of our panels as a way to observe human interaction and determine common cause failures in this transition.

Our hard panel will emulate what an analog control panel looks like in an actual Nuclear plant. The ACTIVE lab will provide our team with the specifications they desire for this panel (i.e. how many gauges, how many valves etc.). In addition they will provide us with access to their digital implementation of a nuclear power plant control room for us to connect our panels to. This digital application is a simulator of sorts which will act as the testing environment for the groups research. In short, our hard panel, in coordination with our soft panel, will act as a visual testing device, at a novice level, to assist the ACTIVE Lab researchers with their work on testing and troubleshooting the Digital Control Room.

2.0 Project Description

This section will describe our motivations for taking on this project in addition to specific requirements we will need to meet in order to satisfy our customer, the ACTIVE research group.

2.1 Project Motivation and Goals

The primary motivation behind picking this particular project was our desire for a mentor in the process of working through our semesters of senior design. As a group we knew there were projects that we could push through by looking up things on the internet, which we definitely will still have to do, however as a team we decided to pursue the option of a guided project. Once this was decided, and we came into contact with the ACTIVE research group, we set our main goal to assisting our new mentors in any way possible.

The ACTIVE group had many projects that we could contribute to. Our initial role was to identify our interests and find a way to pursue them in a way that would be beneficial to their research. After guidance from our advisor as well as the research team we decided to assist them in their research on the transition of nuclear control rooms from analog to digital implementations. This specific project will allow all of our group members to interact with technologies that we wanted to gain experience with. Our partnership with this research group will allow our team to gain experience in custom building analog devices, 3-D printing, PCB design and more. The ACTIVE Lab is also the location of the Robotics club machine room, so our partnership will gain us access to multiple tools we will require in order to build our panel like a 3-D printer, saws, soldering stations etc.

After extensive collaboration between our team and the research group, a solid idea of the project came together. Our main task would be to create a physical control panel, which would be capable of emulating what an analog control room is like today, and will be capable of connecting to the digital implementation of a control room that the ACTIVE research group currently has, via our soft panel application. We have to make sure that both the soft and hard panels are easy to use because the testing performed by the ACTIVE Lab will be upon a novice user. Once our project is completed, our hard and soft panels will be used to assist the ACTIVE group in their research about the risks and rewards in this conversion from analog to digital devices.

2.2 Objectives

The primary objective of our team is to support the ACTIVE lab in their research on this nuclear topic in any way. More specifically we have come up with the objective of creating workable panels to emulate real life errors and responses seen in current control rooms in nuclear power plants. As a basic system overview of our objective, the hard panel will consist of various devices capable

of alerting the user to what is going on such as LED's, gauges, buttons and switches. In addition the soft panel consists of an application interface containing the same components, thus acting as the same station just in a digital environment. The hard panel transmits changes in state to the soft panel application via RS232. This soft panel communicates with the power plant simulator over UDP transmissions. The purpose of the simulator is send commands to any control panel that is listening to it via UDP multicasting, which our soft panel supports.

In terms of communication, we need to manage to establish a form of dual communication between both our hard and soft panels. This means that if something is changed digitally we want to have our measurement tools on our control panel to reflect this, i.e. change the value on a gauge. If we flip a switch on the control panel we want to see this action represented on the soft panel. In addition if an analog control, like a push button, is changed on the soft panel we will be reflecting this change via sync LED's which will be lit based on the system being in sync or not. The purpose of this is to make the testing environment easier on the research team so they can focus on the more important aspects of their job like determining the effects of this transition to a digital control room. It is imperative that we make our panel as user friendly as possible due to the testing audience being novice power plant operators.

Some additional objectives are that we want our panels to stay within budget, to have them be as useful as possible to the research team and to leave a lasting mark when we are finished with senior design and graduated. Finally we want our project to provide us with enough design experience, which we hope to achieve through 3-D printing, custom building analog parts, our circuit design and the complicated software connectivity.

2.3 Project Requirements and Specifications

Below we have listed the requirements and specifications for our project, organized by hardware requirements and software requirements, which we developed by collaborating with our contacts at the ACTIVE Lab.

Requirement Number	Description
1	Our hard panel shall consist of approximately 100 components
2	Our hard panel shall have a minimum of 4 switches, with an estimated value of 25 switches
3	Our hard panel shall have a minimum of 4 buttons, with an estimated value of 25 buttons
4	Our hard panel shall have a minimum of 4 gauges, with an estimated value of 25 gauges
5	Our hard panel shall have a light box with a minimum of 4 LED sectors with an estimated value of 25 LED sectors
6	Each LED sector shall have 1-4 LED's arranged in parallel depending on maximum brightness
7	Both switches and push buttons shall have status LED's which will light up depending on the state of the analog device
8	Hard panel dimensions shall not exceed 50cm W x 90cm L and 30cm D
10	Hard panel shall utilize power protection circuits to keep temperature low and noise maintained
11	Every device shall be labeled with a 7 character alphanumeric string
12	The hard panel shall be plug and play with an isolated 5 V DC power supply.

Table 1: Hardware Specifications and Requirements

Requirement Number	Description
1	Hard panel and soft panel shall be user friendly in order to appeal to the novice user
2	The soft panel shall listen to the simulator on a UDP port specified by a settings.xml file
3	The microcontrollers used in the hard panel shall be reprogrammable.
4	The microcontrollers used in the hard panel shall support in-system programming
5	All components shall reside in a local area network
6	All software components shall communicate with each other using one messaging paradigm
7	All data sent to the power plant simulator shall be sent from the soft panel with UDP transmissions
8	When necessary, the soft panel application shall provide the user with step by step instructions on how to get the panels in sync
9	The soft panel and hard panel shall only fully operate when they are in sync with each other

Table 2: Software Specifications and Requirements

3.0 Project Research Related to Definition

The following topics depict the research that our team completed in order to define our project as well as our design.

3.1 Nuclear Analog to Digital Conversion

The main purpose of our project is to work to support the ACTIVE Lab in their research on the conversion of nuclear control rooms from analog to digital. A control room in a nuclear power plant consists of multiple Instrumentation and Control Systems (I&C). These systems are considered the main form of communication in a plant as they monitor all aspects of the plant's health and help to respond with the adjustments or actions that are needed. It is essential that these systems run smoothly so the industry often trusts technology that is already very well established. For this reason, the nuclear industry has taken a long time to make this conversion. Digital systems have been proven to offer higher reliability, additional diagnostic capabilities and to improve plant performance.

In the United States, the last year in which construction was approved to start on a new reactor was 1978. In 2012, Southern Company was approved to build two new reactors in Georgia. This recent innovation in promoting new growth for the nuclear industry in the United States jumpstarted the initiative to convert the Instrumentation and Control Systems, both established and pending, to the digital age, something that many other countries have been implementing in their reactors.

As time goes on the maintenance on an analog system becomes more difficult because the technology is becoming obsolete (so the parts are much more difficult to obtain). Many of the components being used in nuclear power plants were designed over 30 years ago and many of these are approaching or exceeding the original life expectancy. The risk of continuing on this path to keep these analog components is that there will continue to be a decreasing availability of replacement parts in addition to decreasing supplier and maintenance support. This results in lower levels of reliability and an increase in operation and maintenance costs to establish a continuing level of acceptable performance.

Instrumentation and Control Systems are used for protection, control and monitoring different devices inside of a nuclear power plant. There are emergency systems, nuclear safety systems, and production and maintenance systems that all work together to keep a nuclear power plant running safely. Instrumentation and Control Systems affect every aspect of plant operation and are made using the following components:

- Sensors, items that interface with the physical processes within a plant. The sensors are responsible for continuously taking measurements of plant variables like temperature, pressure and flow.
- Control, safety and regulation systems. Control systems process measurement data to manage plant operation and optimize plant performance. The primary goal of the control systems is to make sure that the plant is in a safe operating envelope.
- Communication systems (data and information transfer)
- Human-system interfaces, currently consist of gauges, knobs, switches and push-buttons (representing the analog components). Digital components would consist of digital displays and touch screens.
- Surveillance and diagnostic systems
- Actuators, valves and motors that are operated by the control and safety systems to adjust physical processes within the plant. There are also status indicators for actuators which provide signals for automatic and manual control.

The human-system interface is where plant information is translated into required operator action. In the event of a digital conversion the human-system interface features like computerized procedures, touch-screen interfaces and large screen overview displays will all allow operators to control a power plant from their workstation without needing to move around the control room. The computerization of the human-system interface allows more efficient operations and maintenance due to the operators having a more comprehensive and detailed understanding of operating conditions. The estimated result of a computerized human-system interface would be improved safety, reduced operating costs (due to the avoidance of forced outages and unnecessary shut downs), increased efficiency and output.

The challenges of converting an analog control system to a digital system are the primary reason why the Nuclear Regulator Committee hasn't yet attempted to convert the power plants. The nuclear industry is conservative to its core, this is due to a desire to establish safety as the number one concern. The NRC and industry leaders have waited for extensive proof that a digital system could be used in a safety critical environment. In order to convert operators to the new digital system extensive training would have to occur in addition to revisions of operation and maintenance procedures. The Regulators have to adjust to the new and developing technology in order to ensure that the plant benefits safely from any advantages that are able to be established (greater reliability, lower cost or greater safety). There are two primary issues that the industry is worried about in this conversion: common cause failures and cyber security.

Common cause failures in statistics are issues that you see consistently upon analysis of data. There are already established common cause failures in the analog implementation, but the transition to a digital system invites in new common cause failures like increased human error, issues from software

dependency, and the complexity of features. With the new digital implementation, there is an increased complexity of the interaction between subsystems, and a greater probability of latent faults inside of the software. A common cause failure occurs when the system experiences a fault, a triggering event activates this fault, subsystems that are supposed to be independent become affected, safety is compromised and multiple systems must share the same fault. In a different light, a potential future common cause could be that many in this digital age are losing the psychological effect of actually changing something. There is a measurable feeling upon pressing a button or flipping a switch especially when it is important. This will definitely be something to look out for in the future.

The industry currently approaches common cause failures through the following methods; diversity, redundancy and independence. Diversity represents the establishment of multiple redundant systems or components with different attributes all for one particular function. Redundancy represents an implementation of alternative systems and components so that in the event of a failure any one could perform the required function. Independence prevents failures and common cause failures due to plant hazards. All of these factors which help to prevent common cause failures add to the complexity of a system design and encourages human error.

The second very serious issue for a digital implementation of a nuclear power plant system is the risk of a cyber threat. Nuclear computers and digital components are generally isolated from external threats, however there have been quite a few reported incidents overseas exploiting the ease of access some hackers have been able to achieve with nuclear power plants. One of the most televised or known events was the Stuxnet virus. The virus was discovered in 2010 and was targeting industrial control systems via Microsoft Windows. The primary target of the virus was nuclear facilities in Iran, but there have been incidents all over where hackers or viruses have infiltrated plants with the intent of causing a shut down. Computers and digital devices in a plant need to be protected in order to prevent unauthorized access that might be a part of an attack as well as to ensure safety and security. Potential threats are caused by business espionage, recreational hackers, cyber activists, nation states, terrorist organizations or technology theft. The variety of purpose shows the variety of effects and threats that the digital devices need to be protected from.

Digital systems need to be protected from 4 main categories;

- unauthorized access to information
- change of information, hardware or software
- shut down of systems
- unauthorized intrusion in data communication systems

The consistent advancements in technology and developing world of hacking means it's very difficult to protect against potential threats. The nuclear industry

approaches this by various technical tools like intrusion detection, virus scanners, encryption, and also administrative tools like security zones, security management systems, passwords and biometric identification. The nuclear industry will be obtaining support and guidance from other fields like the military, air-traffic control and national security in order to implement their new digital systems.

Digital technology has been booming and developing rapidly over the years and the nuclear industry is well aware of this. Their approach to the conversion is to implementing trusted and commercial off-the-shelf products into their digital systems in order to reduce the risk of new technologies. Updating digital devices and systems is much easier than updating analog systems. When the nuclear industry finds a trustworthy software update it's preferable because it's easy to install, as opposed to this monster of a conversion from analog to digital in every plant. Digital systems will be implemented in the new power plants but also many of the established power plants will be receiving technology overhauls. These overhauls will consist of new solutions in sensing technologies and digital control as well as advanced sensors, detectors, transmitters and data transmission lines. Our main focus as a team is interacting with the software representation of the digital conversion, which has been provided to the ACTIVE research group, and weeding out potential common cause failures for the conversion process. For this reason we won't be interacting as much with the developing digital components but we will be focusing on the risk of human error and weeding out potential bugs for the digital system.

3.2 ACTIVE Lab Power Plant Simulator Research

The focus of the Applied Cognition and Training in Immersive Virtual Environments (ACTIVE) group for their Nuclear Power Plant operations project is to analyze and understand the factors that influence operator performance, operator state (overloaded, under loaded, stressed etc.) and the type of errors that operators make. This research is applicable to complex machine systems run by highly trained operators so upon completion of this project, the research team will be able to apply their results to more than just Nuclear Power Plants. In addition this research will assist the Nuclear Regulator Committee (NRC) in determining the impact of technology upgrades, automation of tasks, and digital interfaces on human operators.

Here is some background information on the operation and operators of a nuclear power plant:

- Main Control Room consists of complex systems controlled by a human-system interface
- A minimum of three operators are required to manage and maintain a single nuclear reactor

- Two operators act as 'Reactor Operators' or RO's
- Third operator acts as the 'Senior Reactor Operator' or SRO
- Operators perform 4 categories of tasks
 - Monitoring and Detection
 - Monitoring: requires checking the plant to determine if it is functioning properly. This is done by checking parameters indicated on control panels
 - Detection: recognizing if the state of the plant has changed
 - Situation Assessment
 - This is performed by evaluating the current states of plant systems to determine if they are within required parameters
 - Response Planning
 - This requires deciding on a plan to diagnose and complete appropriate actions if/when an event occurs
 - Response Implementation
 - Performing actions required by response planning

In short it is easy to understand the tasks operators perform but the investigation of performance, errors and states in an experimental setting has been limited. Our control panel will assist in this investigation.

In order to assist in the research, our project will focus on a single panel of a nuclear power plant control room. This panel cannot be modeled after an existing panel due to confidentiality and security reasons so the research team has provided us with generalized information about control devices and the layout of a panel. Once our hard panel is built and our soft panel application has been implemented, the research team will introduce it to a non-operator population or a novice operator. This means that our panels need to be very simple and easy to understand to an outside participant. We must assume that the user of both our hardware and software has little to no knowledge of nuclear power plant operation.

An EOP is a procedure that an operator follows when certain symptoms are present in the plant. These procedures depict the type and specific order of actions that an operating crew must take. The research team had to select EOP's that would be best suited for a novice sample. In order to do this they had to follow this criteria:

- 1) Select an EOP that represents the typical task flow that is most common
- 2) Select an EOP that allows an investigation into all roles on the operator team
- 3) Select an EOP that requires participants to perform an equal ratio of task types being investigated
- 4) Select an EOP that incorporates the use of all major categories of instrumentation and controls

In order to accomplish the above criteria the research team had to simplify the control panel in focus. They did this by reducing the total value of instruments on the control panel in order to follow the theme of simplification for the novice user. They also removed any potentially complicated controls, leaving only switches, gauges, light boxes and buttons.

The next step was to reduce the difficulty of an EOP. This meant stepping down the quantity of steps an operator must perform. Typically an operator could perform tens of steps for an EOP, but for the novice user the research team reduced this down to a fraction of these steps with a defined stopping point. Many training sessions can last up to 3 hours for an EOP but it is also common to see 30-45 minute scenarios, especially in initial training. Due to this fact, the research team established their simplified scenarios to have a 30-45 minute timeline.

The final step of establishing their research plan was to select measures that allow them to understand performance, determine error types and understand the state of operators (stressed, overloaded, alert etc.) while working on a complex system. They established that performance can be measured in terms of response time, accuracy of actions and detection of changes. Errors can be categorized by slips, lapses, violations and mistakes. They also want to analyze the effect of workload on both errors and accuracy, including things like interruptions during an EOP. In order to evaluate and gather all of the data necessary to determine these items the research team is looking into the use of the following devices:

- Electrocardiography (ECG): measures cardiac activity. Heart rate and heart rate variability have been associated with mental workload
- Eye Tracking: measures ocular behavior. Can provide insight into task difficulty
- Transcranial Doppler (TCD) Sonography: monitors cerebral blood flow velocity. Studies show that a decrease in cerebral blood flow velocity parallels with decreased performance for sustained attention of highly demanding tasks
- Functional Near Infra-Red (fNIR): monitors hemodynamic changes in oxygenated hemoglobin and deoxygenated hemoglobin in the prefrontal cortex. Blood oxygenation increases are associated with increasing task difficulty
- Electroencephalography (EEG): measures neural activity. This device is sensitive to changes in mental workload.

Although we will not be involved in this portion of their research, we wanted to provide insight into the purpose of our project and what the future use of our panel will be.

3.3 Analog Devices

The purpose of the analog devices on the control panel in a nuclear power plant are to display either important readings (i.e. gauge values, LED alerts or warnings etc.) or to be a visual tool to toggle items (i.e. switches and push buttons). These items allow the power plant operators to keep control over the very sensitive equipment that make up their nuclear reactor. This section shows our research on these devices and in our opinion, how to best recreate them.

3.3.1 Push Buttons

Push buttons are simple switch mechanism used to control an aspect of a machine or process. Push buttons are used universally in things like computer keyboards, calculators, video games, and now are even impressively being implemented as a way to start a car. On the industrial side of things, push buttons can be used to signify something very dangerous or important, especially when installed as a red colored button. In addition push buttons in the industrial setting can be used exclusively as mechanical items with no electrical circuits for control. This can be done by mechanically linking two push buttons together so that if one is pressed the other button can be released.

The uses of push buttons are endless. For the nuclear industry, push buttons are used for a variety of tasks but most specifically are used for turning power either on or off for device. The long process of clean up for the Fukushima reactors in Japan have recently run into difficulty due to a worker pressing the wrong button, which cut off power to the cooling pumps, which were keeping the very sensitive fuel rods as stable as possible. This reflects how important the various types of buttons are to the nuclear industry.

In order to best emulate what these buttons are like in real life we had a couple of options. The problem is that the devices used in a control room are so diverse that many of our options are being simultaneously implemented. A push button could have a status LED above it to describe the state, it could be a standalone button or even be an illuminated push button. After looking through these options, and considering that the research team is primarily focused on our design of rotary switches, gauges and light boxes, we decided to settle in on the use of an illuminated push button. Our choice in product is shown below.



Figure 1: LED Push Buttons Reprinted with permission from Adafruit Industries

We picked these in order to bypass the need to set up status LED's for our push buttons. In addition these specific push buttons are latching, or in short will stay pushed until the button is pressed again. We will potentially obtain both latching and momentary buttons, which the distributor, Adafruit, is able to supply us with both for a decent cost. Finally we selected both green and red push buttons in order to be able to distinguish the importance of individual buttons on our hard panel. Red buttons usually signify items of more significant importance.

3.3.2 Gauges

The gauge is a classic measurement instrument that has always been an important part of the power plant industry. The clear view of various operating conditions that they provide ensures the plant is running normally and prevents problems from arising. A nuclear power plant control room is filled with gauges in order to monitor the complex system it is running.



Figure 2: Classic analog gauges with permission from GNU Free Documentation License

Unlike the circular gauges shown above, we will be designing rectangular gauges for our control board. This is the type of gauge that the ACTIVE lab is focusing on, primarily because digital gauge representation is simpler in rectangular form. It is easy to read and takes up less space than its circular counterpart. In analog terms, however, this rectangular analog design is not as simple to create and is not commonly used. It requires more space and extra measurements. Nevertheless, in order to support ACTIVE, this is a challenge we are willing to take on.

Gauges use a distance amplifying instrument, called an indicator, to accurately measure small distances and angles. Our gauges, like most others, will be using a dial indicator to amplify the correct measurement. This indicator must be required to specify very precise readings. This means that our gauges will have to provide plenty of room for indicators to rotate on.

Generally, analog gauges function with a variety of mechanical devices. Different gauges use different devices. For example, a temperature gauge uses a bimetallic strip, which reacts to heat by coiling and uncoiling, which moves the indicator to its proper position. A PSI gauge functions with a calibrated rod, spring, and piston. For our project, the types of gauges we are designing are not specified by requirements. Essentially, we are replicating a variety of different gauges that serve different purposes. This replication means that building true analog devices is outside the scope of our project. We may find a general method to simulate the actions of analog gauges. Because of our group's electrical and computer engineering disciplines, it is not required to build purely analog gauges with mechanical devices. Our gauge may be controlled digitally to replicate the image of an analog gauge. Our goal is to find one universal device create functional gauges.

For this case, a stepper motor is a device of high interest. This is a DC electric motor with a circular shape that divides a full rotation into a number of equal steps. The component that rotates is calling the motor's pointer shaft.



Figure 3: A bipolar stepper motor with permission from Sparkfun Electronics

With proper design, a stepper motor may be controlled programmatically to move the gauge's dial indicator around its available field of rotation. This would satisfy our goal of using one type of device for every gauge on our control board, no matter what its function is. With proper research, it is entirely possible to program a microcontroller to drive a multiple amount of motors in any way necessary. This is easily implementable and is the path we will pursue.

The rectangular shape required for the gauge design establishes a limitation. Stepper motors rotate in a circular direction, meaning the dial attached to it needs a circular space to rotate freely. The obvious solution would be to combine the two shapes together, as shown in the following figure.

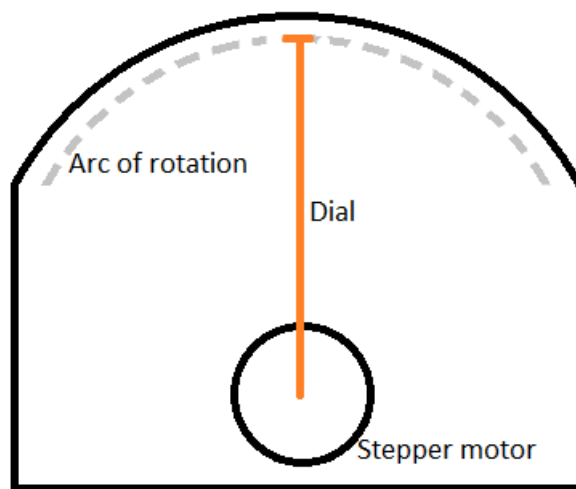


Figure 4: Tentative gauge design

It is clear that by using the shape shown in Figure 3, the orange dial will be able to rotate around to point to a wide range of measurements. This will be done with a series of steps performed by the motor, stepping from the zero degree to a yet to be determined maximum degree. Another helpful point to be made is that the length of the measurement display will be equal to the arc length of the semicircle. This fact will make measurement precision simple to accomplish.

Through the research described above, a basic gauge design has been determined. When specific parts have been selected, it will be possible make measurements to establish an exact design. This area will be covered within the hardware design of Section 4.

3.3.3 LED's

In order to accurately simulate an analog control panel there will be multiple predetermined programs for the user to go through for training. While a separate subsystem is in place to communicate between the input signals and the programs there must be a physical display to communicate to the user and

whether or not the nuclear reactor is fine or in fault based on their actions. To do this as well as accurately replicate the current “dated” systems, an 5 by 5 LED array must be created with the use of a microcontroller, timer, and communication with the master controller as detailed in section 3.7.1 Microcontrollers.

The use of a microcontroller is needed to create an individually addressable LED array. In order to avoid having to have an individual wire that runs from the logic to each LED, only the rows and columns of the array will be connected to the microcontroller. The microcontrollers under consideration for this subsystem are discussed further in the hardware design portion of the report, sections 4.0 Design and 3.7.1 Microcontrollers.

There are multiple ways to approach the designing of the individually addressable matrix. The first would be to create an array with LEDs and use shift registers and a driver to control which row and column needs to be addressed. A simple generic image of LEDs in an array is given below.

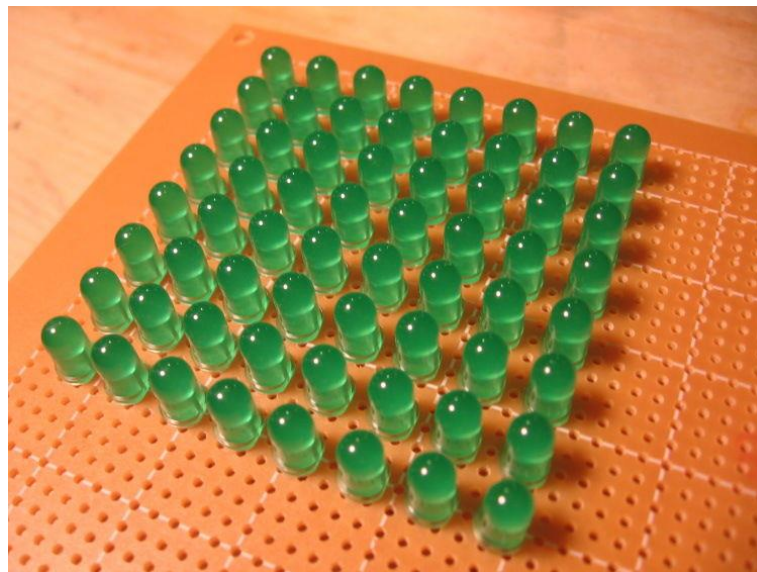


Figure 5: 8 by 8 LED Array

If this design were to be approached, the possibility of using RGB LEDs to meet the client’s future desire to test the error associated to different color recognition would need to be readdressed. An LED matrix designed in this manner would need to be individually soldered and with an 8 by 8 array with two leads each for a single color LED that’s 128 leads. For an RGB LED that has four leads that would be 256 leads that need to be soldered and accounted for in not only the hardware but also the programming design. The design would become significantly more complicated.

A way to avoid this is to use digital RGB LED strips such as the ones designed by Adafruit shown below and are commonly used in lighting displays.



Figure 6: RGB LED strips by Adafruit reprinted with permission from Adafruit Industries

These strips are desirable because not only are they RGB, but they have 7-bit pulse width modulation (PWM) precision which provides 21-bit color per pixel. Furthermore, the LEDs are already equipped with shift registers that are chained up and down the strip. This scheme allows for the user to be able to shorten or lengthen the strip as needed. There are cut marks already incorporated into the design making this an easily accomplishable task.

In addition, only two output pins are needed to send data to the LED strip. This will significantly lessen the amount of wires needed for the array construction and the microcontroller will only need two digital I/Os for the array. The LEDs do require a 5V DC power source and Adafruit is adamant about the power not exceeding this so a protection circuit will need to be designed and will be addressed in section 4.6 Power Supply. Our microcontroller is capable of running off of 5V DC as well so there should not be a conflict in power distribution.

The LED strip design is more desirable than the individual LED design. Even though it is the more expensive option, it will have a cleaner look and will allow for more flexibility when it comes time for designing the end aesthetics of the experiment. Since this is to be used after senior design for testing and simulation purposes this is a very important aspect of the project.

Since the LEDs come as a strip, they will need to be disassembled and reassembled into a matrix. This is accomplished by connecting the adjusted strips in a zigzag pattern. Even though the image below of the zigzag pattern uses different LED strips, the design concept is still the same.

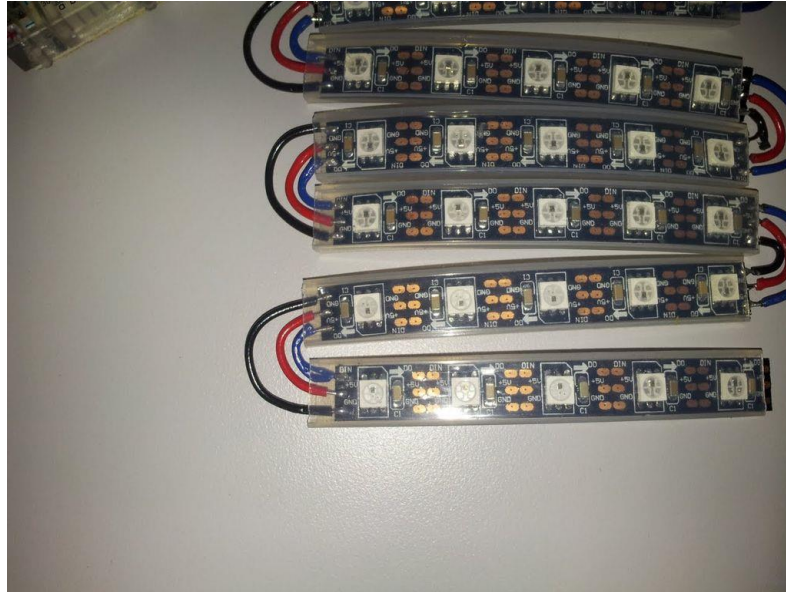


Figure 7: RGB LED Strips connected in zig-zag pattern reprinted with permission from Michelle Leonhart

Also, the LED strip design requires less hardware since the strips come pre-equipped with the proper shift registers and resistors. The matrix will only need to be directly connected to the microcontroller and the power protection circuit. Another nice feature is that adafruit has a large library available as open software for their LED strips. While there are other versions of the LED strip, they are no longer carried by Adafruit and would need to be purchased through a source such as ebay. One such part is the HL1606. While this part used to be cheaper, it is not equipped with PWM control and is only capable of being programmed on and off. Even though this would suit our purposes fine, the fact that it is not readily available makes it an undesirable part. However, the LPD8806 listed in the table below is a good substitute.

Part	Description	Supplier	Quantity	Price
LPD8806	PWM control	Adafruit	32	\$29.95

Table 3: Table describing potential LED RGB strip details

To create the LED matrix with the LED strips, while desirable because of its simplicity will not involve much design and therefore is not as beneficial to our group. The strategy which we believe will be the best path, more so than the first two options, would be to use a master microcontroller and several slave MCUs that control individual MOSFETs which will then either turn on or off an SMD RGB LED. A more detailed look into how the LED matrix will be designed can be found in section 4.2.2 LED Design.

In addition to the LEDs that will form the light box, LEDs are needed as status lights for the rotary switches to signify what position each switch is in. This will be simple to implement as they can be tied in series directly into each position of a

rotary switch. The switches have two positions, on or off, green or red respectively. An option for these status lights is to use the RGB SMD LEDs that will be used for the light boxes. Since, these are an added feature and not needed to enhance the electrical design of the project, it would make more sense to use pre-built panel mounted LEDs that will give the control panel a more professional look.

There are several options available for panel mounted LEDs. Dialight has several models of 16.5mm 5V forward voltage LEDs that are available in several colors. They are resistant to vibration and shock, are daylight viewable and are able to perform to upwards of 100,000 hours. However, the general price ranges for these lights are \$14.90 for one and we will require at the very minimum 50 status lights. This cancels them as an option as they are out of our budget.

A second option is to use the ¼" 5VDC panel mount LEDs by CML. These come with a built-in resistor and a luminosity of 10mcd which should be suitable for our needs. They are only \$1.25 a piece and come pre-equipped with lead wires which will allow for simple plug and play installation. While they do not have all the added features of the Dialight products, they fit better into our budget and are able to meet our minimum requirements.

Furthermore, two lights will be needed to signify to the user whether or not the hard and soft panels are in sync with one another. Two outputs on the master controller will need to be utilized and since the CML LEDs come equipped with built-in resistors they can be tied directly into these MCU outputs. A simple, red and green LED is all that is required of the hardware to implement this added feature.

3.3.4 Rotary Switches

In order to properly model a nuclear power plant control panel we need to implement rotary switches into our design. These switches will be used to supplement the training exercises that will be run using the digital implementation of the control panel.

The nuclear industry uses a variety of switches on their control panels. Some are multiple position or pole switches, like 3-pole, 4-pole or even 5-pole switches. In order to accommodate the novice user, the ACTIVE lab requested us to keep it simple with a 2-pole or 2-position rotary switch in order to control the state of various items. Their state will either be normally closed or normally open or essentially depicting whether something is on or off. We are currently estimating needing to use about 25 of these rotary switches on our control panel. An example of what they will look like will be similar to the controls on a stovetop or washing machine just smaller in size.

This switch we intend to use has a switch length of 70 mm and then contact block dimensions of 24x30x30 (mm). The size is relatively small but efficient for

saving space on our control panel. We intend to order these parts in bulk from Skycraft locally.

In the event we would like to employ more of a design difficulty to our project, or that we would have additional time we would like to pursue the idea of supplementing our rotary switches with a 3-D printing design. The idea would be to still order the 25 switches but to take them apart and use the contact block and basic switch mechanism. We would then utilize the ACTIVE lab 3-D printer and buy our own 3-D printing supplies. Our design would consist of a new handle for the switch, with a clear label for the on and off positions. This new handle would probably be a little longer with the benefits discussed below.

The benefit of a long handle is for an easier grip, but to purchase a long handle switch is usually more expensive. 3-D printing supplies are cheap so it would be easy for us to design our own. In addition our panel could benefit from having labels that depict what status the switch is currently in. In addition to selecting the switch above, our design has to incorporate LED's above each individual switch in order to display the state of the switch. The status of these LED's will also tell the application what status the switches are in. This will be discussed in more detail in the design section for LED's, 4.2.2.

Below is some research performed on 3-D printing to potentially support both our rotary switch and gauge designs.

3.3.4.1 3-D Printing Basics

Essentially 3-D printing allows users to create a physical model of any digital 3-D model. You can accomplish this by using either a scanned set of 3-D images or draw your object using computer-assisted design software (CAD). The digital model is saved in STL format and sent to the printer to be generated. A 3-D printer functions through layer-by-layer printing.

3-D printing layering is accomplished many different ways but primarily the following 3:

- SLS – Selective Laser Sintering
- FDM – Fused Deposition Modeling
- SLA – Stereolithography

SLS is an additive manufacturing layer technology. It utilizes a high power laser to fuse particles into a mass that has the desired 3-D shape. The laser is capable of selectively fusing powdered material by scanning cross-sections from the 3-D model of the part you are trying to build. After each of these cross-sections is scanned, the powder is lowered by a layer thickness and a new layer of material is added onto the top. This process is repeated until the entire model is built.

FDM is similar to SLS but uses extrusion instead of laser sintering. What this means is that the process slices the model for the build process. FDM also utilizes thermoplastics. Some noticeable differences between SLS and FDM are the texture (SLS has a sandy texture while FDM is smooth), FDM has inherent size restrictions but is also capable of yielding much more accurate models.

SLA is an additive manufacturing process which uses an ultraviolet laser to build parts one layer at a time. The laser traces a cross-section of the part pattern on to a surface of liquid resin. The laser light cures and solidifies the pattern that is traced on the resin fuses it to the layer below it. A new layer of material is added and the process is repeated, building up the 3-D model.

In order to generate the model, it is necessary to use one of the following CAD programs;

- Google Sketchup
- 3Dtin
- Tinkercad
- Autocad
- Pro Engineer
- Solidworks

This list was generated by collecting some of the basic CAD programs that are taught to engineering majors at UCF (usually mechanical and civil), as well as a list of what was found to be user friendly and simply programs specific to 3-D printing. One of our team members has background in Solidworks and a basic knowledge of CAD software and design. The most likely program our team will pursue using however will be Tinkercad. The software has a free trial but after that is relatively affordable. Its premise is having 3 simple tools, and after looking through the interface our member decided that would be the easiest route.

We would use the option of 3-D printing for a variety of items for our project. We believe utilizing 3-D printing for connectors or our handles for our rotary switches would be a great way to keep the control panel looking organized and clean. It would be relatively easy to model a new handle for the rotary switch which implements labels for on and off as well as a longer switch handle. In addition we are going to have multiple fiberglass casings to surround both our LED arrays and our gauges. We would like to create fittings or connectors so that these fiberglass parts are removable but secure. Similar to how a battery cover has notches to insert into its casing before it clicks in. If we find that we have sufficient time we would like to pursue this venture in order to supplement our design difficulty.

3.4 Analog to Digital Signals

There are many benefits in converting the analog signal from our power source to a digital one. Not only are digital signals less affected by noise but also a precise signal level is not required for transmitting information via digital signals. This will allow us to communicate information between subsystems with greater noise immunity. Furthermore, digital signals can handle multidirectional transmission and require the use of less bandwidth enabling the user to convey more information in less space. Since the majority of microcontrollers, including the ones we are considering using for this project, come equipped with A/D conversion capabilities it is more practical for us to take advantage of these pre-existing features.

These converters work by a pretty straightforward concept. An analog signal is received as the input and depending on its size a corresponding digital signal consisting of 0s and 1s is outputted. A basic block diagram of the system is given below.



Figure 8: Block Diagram for Analog to Digital Converters

How the A/D converter work is it takes an analog input voltage range and divides it into corresponding discrete values that will represent a digital output. Let us consider a system with an analog input in the range of $0 < V_A < 5V$; where $5V$ is the reference voltage, V_{ref} . The digital signal is a 4-bit word which means it will have 16 corresponding discrete values ($2^4=16$). In order to find the voltage input for each bit the following equation is used.

$$V_A = n\left(\frac{5}{16}\right)$$

It takes the maximum analog input voltage and divides it by the total number of bits. This value is then multiplied by a user pre-determined value n . This n value is there to represent the number of steps needed to represent each digital output. As one can see in the table below and its corresponding graph, there is a value that can be specified for each digital output required by the system. Even if the input value is $.5(5/16) < V_A < 1.5(5/16)$ the digital output will remain constant at $VD=0001$. So there is room for variation in the input signal value.

Step(n)	Analog Input Voltage(VA)	Digital Output Voltage(VD)
0	0	0000
1	0.3125	0001
2	0.625	0010
3	0.9375	0011
4	1.25	0100
5	1.5625	0101
6	1.875	0110
7	2.1875	0111
8	2.5	1000
9	2.8125	1001
10	3.125	1010
11	3.4375	1011
12	3.75	1100
13	4.0625	1101
14	4.375	1110
15	4.6875	1111

Table 4: A breakdown of an analog voltage output and the digital output that corresponds to it.

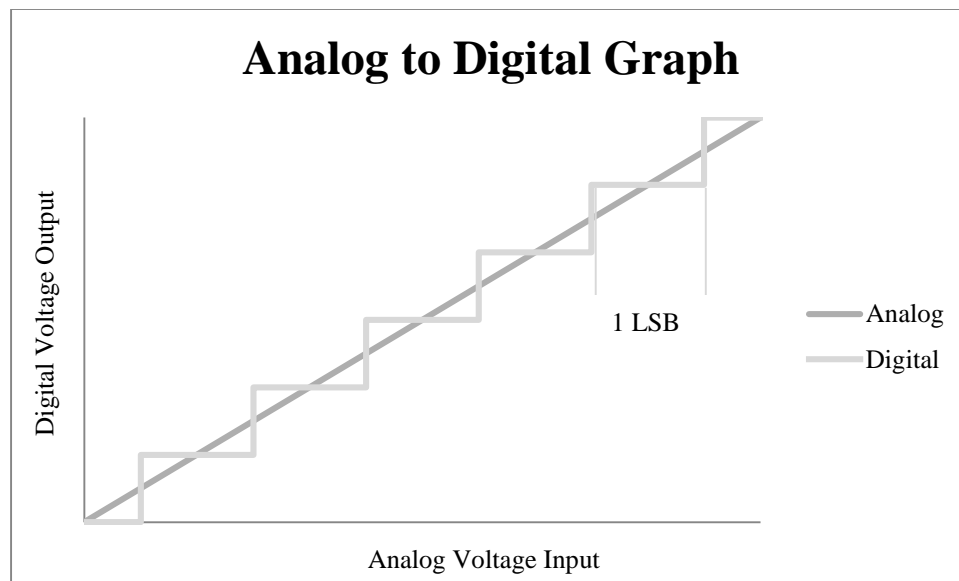
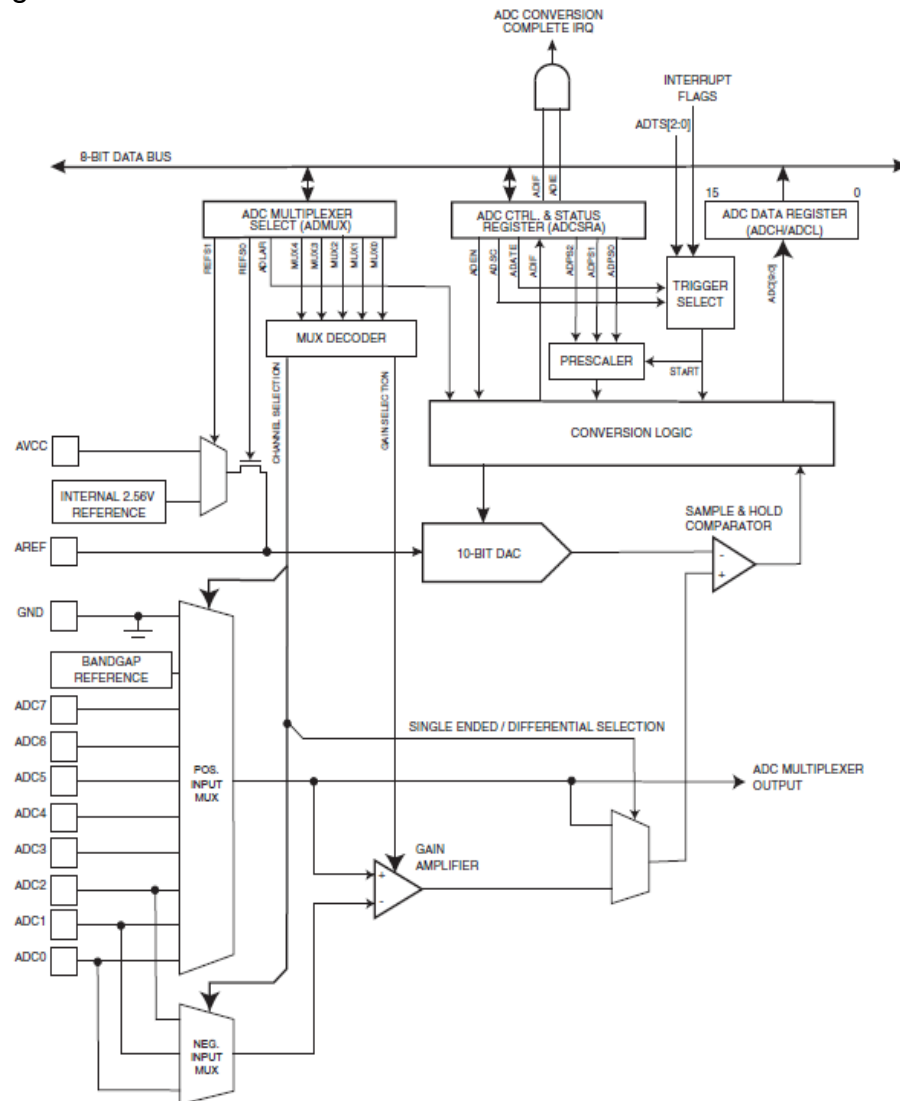


Figure 9: A graphical representation of an Analog to Digital Converter

There does exist an inherent quantization error in the analog to digital conversion. This error depends on the reference voltage range and does decrease as the number of bits of the digital signal increases. A physical representation of the quantization error is given in the graph below. It is always represented as $\pm 1/2\text{LSB}$ where LSB is the least significant bit. Referring to it in this manner is an attempt to normalize the parameters away. It is being

expressed in terms of a unit change in the digital value which represents the ideal analog voltage difference. It makes sense to do it in this manner because the unit change signifies a change in the least significant bit of the digital value.

One of the devices in consideration for use is the ATmega16. It comes equipped with an on-chip A/C converter. This ADC has a 10-bit resolution meaning it is capable of 1024 corresponding discrete values. This resolution decreases if one uses the programmable gain stage available though. As an added feature this device is capable of gains of 0dB (1x), 20dB (10x), and 46dB (200x). However if one uses this one can expect the bit-resolution to drop to 8 for gains of 1x or 10x, and 7 for 200x gain.



also comes equipped with an optional internal reference voltage of 2.56V. A nice added feature for better noise performance is that this internal Vref can be externally decoupled at pin AREF with a capacitor. This internal reference voltage can be used as Vref by writing to the REFSn bits in the ADMUX Register. There are two other options beside this though. The user can connect AREF pin directly to the AVCC pin or apply a voltage directly to the AREF pin. The reference voltage will then be taken as whatever is applied to AREF minus 1 LSB. While the minimum reference voltage is represented by GND.

The ADC receives its inputs to be converted from an 8-channel analog multiplexer. This multiplexer allots for 8 single-ended inputs that are obtained from pins of Port A. In addition to this the MCU supports 16 differential voltage input combinations. A conversion occurs by writing a logical 1 to the ADC start conversion bit (ADSC). This bit will only be cleared by the hardware once the conversion is complete. However, this is not the only way that a conversion can be done. There is a feature set up for auto triggering, which enables conversion when a positive edge occurs within the trigger signal. This source can be selected by selecting the ADC Auto Trigger Enable bit (ADTE). The user can set the trigger source with the ADC Trigger Select bits (ADTS).

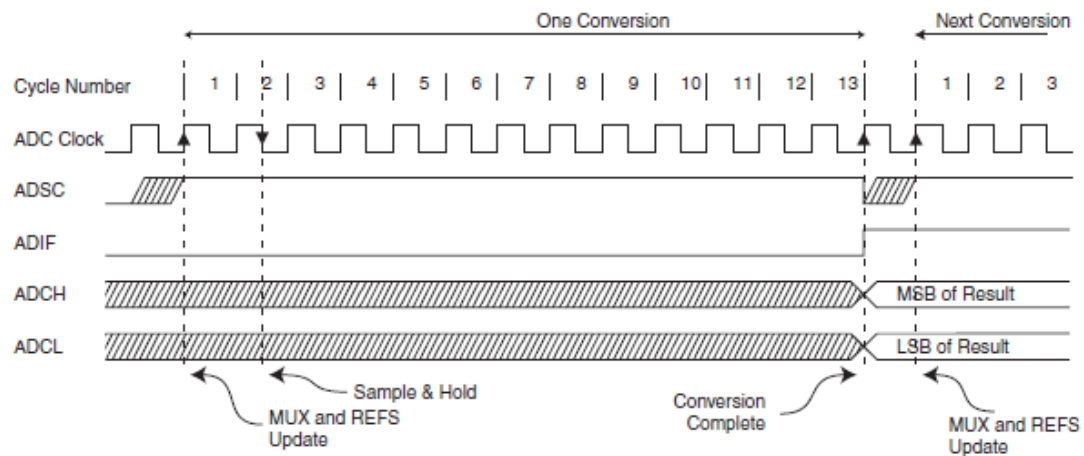


Figure 11: ADC Timing Diagram of a Single Conversion reprinted with permission from Atmel

The ADC is capable of conversions with a high speed time of 13-260µs. A diagram depicting the timing diagram is shown above. Where ADIF is the interrupt flag, ADCH and ADCL are the result registers and remember, ADSC is the start conversion bit. ADIF will be high after the conversion and the conversion results can be found in the ADCH and ADCL bits. The equation for the single ended conversion is:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

While if the differential channels are used the equation is changed to:

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

It is incredibly useful to know how the ADC works for the ATmega16 and will help in the later stages of programming and design.

3.5 Device Communication

When researching microcontroller communication interfaces you must first take into consideration all the external interfaces that the microcontroller will need to support. In today's market you can find a number of different communication interfaces integrated in microcontrollers to tackle almost any conceivable communication task. Before implementing our system of N microcontrollers we will first research the possibilities to let them communicate with one another. Ideally, a microcontroller will be placed inside the housing panel acting as "clients", while another (the "server") will be connected to the PC. From now on we will refer to the "server" microcontroller as the master and the "client" microcontrollers as the slaves. What we will be specifically looking into is how to connect our microcontroller to the server and what is the best solution for a distributed client system. This client system, which through data communication that takes into account max number of devices, conflict management system, and design feasibility.

There are a few things we must preface before we explore the types of communication protocols; one of them being to choose between serial or parallel communication.

- Serial: Sequentially transmits data one bit at a time via one signal line
- Parallel: Simultaneously transmits multiple-bit data via a number of signal lines

The decision has been made to use serial communication due to the fact that parallel has a problem in that it uses many signal lines and the timing can easily fluctuate during high-speed operation. The other factor to consider is whether to use synchronous or asynchronous devices. We will classify the types of interfaces available into these two basic categories.

3.5.1 Synchronous Interfaces

Synchronous interfaces were designed mainly to connect peripheral devices on the same circuit board and tend to be more suitable for bridging relatively short distances. More specifically synchronous interfaces are characterized by the presence of a dedicated transmit/receive clock signal. Information is transmitted from the transmitter to the receiver in the following manner:

- In sequence

- Bit after bit
- With fixed baud rate
- Clock signal is transmitted along with the bits

A “Master” device usually outputs a clock signal that is then received by all “Slave” devices to receive and transmit data in sync. This type of communication is faster in comparison to asynchronous communication since it is constantly transmitting the information with no stops.

The types of synchronous interfaces we will cover are the following:

- Serial Peripheral Interface (SPI)
- Inter Integrated Circuit (I²C)

3.5.1.1 SPI Communication Protocol

Coined by Motorola, The Serial Peripheral Interface is a synchronous data protocol that operates in full duplex mode. SPI is used by microcontrollers for communicating with one or more low speed peripheral devices using a minimal quantity of wires. Devices communicate in the manner of master/slave mode where there is typically one master device (the microcontroller) that controls the peripheral devices (the slaves). This interface consists of an 8-bit serial shift register and a programmable shift clock for the master device. Typically there are three lines common to all the devices and one line specific for every device:

- Master In Slave Out (MISO) – The slave line for sending data to the master.
- Master Out Slave In (MOSI) – The master line for sending data to the peripherals.
- Serial Clock (SCLK) – The clock pulses which synchronize data transmission generated by the master.
- Slave Select (SS) – The pin on each device that the master can use to enable and disable specific devices.

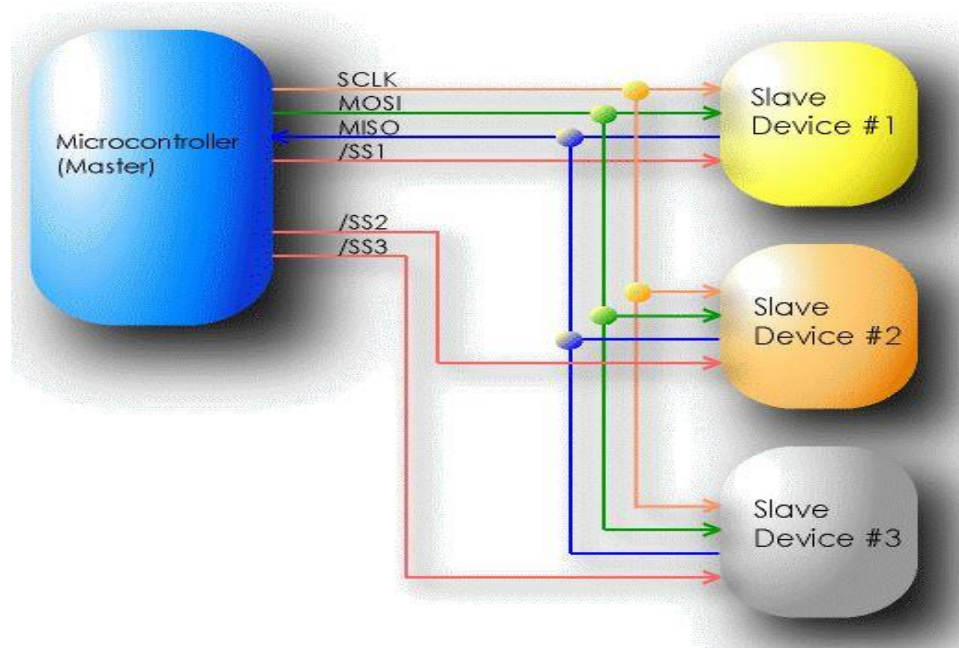


Figure 12: SPI Single Master Multiple Slaves Configuration Reprinted with Permission from Microcontroller Pros Corporation

Due to its full-duplex capability, SPI can receive and send data at the same time. This allows for faster read/write operations to a peripheral and in turn shorter time that the controller needs to be active. This is important because current consumption for microcontroller applications spends most of their time in power save modes and only short periods of time in normal operating mode.

Most SPI interfaces have two configuration bits:

- Clock Polarity (CPOL) – This determines whether the shift clock's idle state is low or high
- Clock Phase (CPHA) – This determines on which clock edges data is shifted in and out

Each bit has two states, which allows for four modes of transmission. These modes control whether the data is shifted in and out on the rising or falling edge of the data clock signal (CPHA) and whether the clock is idle when high or low (CPOL). For two SPI devices to communicate with each other they must be set to the same clock polarity and phase settings. Since SPI is edge sensitive and peripherals implement read and write operations via explicit commands over the bus, selecting the wrong operation is less likely and therefore immune to noise.

SPI is preferred when regarding low cost, high speed, and noise immunity. SPI's full duplex capability and fast data rates make this interface very efficient and simple for single master single slave applications. In practical applications, the requirement for dedicated slave select signals severely limits the number of slave

devices that can be connected to a microcontroller. Multi-master systems significantly increase complexity and are rarely used with this interface.

SPI	
Advantages	Disadvantages
<ul style="list-style-type: none"> Multiple slave devices can be accessed with few wires Low cost Easy to implement 	<ul style="list-style-type: none"> Data and clock lines can be shared but each device requires a separate slave select signal, limiting number of devices in limited I/O systems Short distance

Table 5: Advantages and Disadvantages of the SPI Protocol

3.5.1.2 I²C Communication Protocol

I²C is a half-duplex serial bus that uses two signal lines:

- Serial Data (SDA) – The data line
- Serial Clock (SCL) – The clock line that is used to synchronize all data transfers over the I²C bus

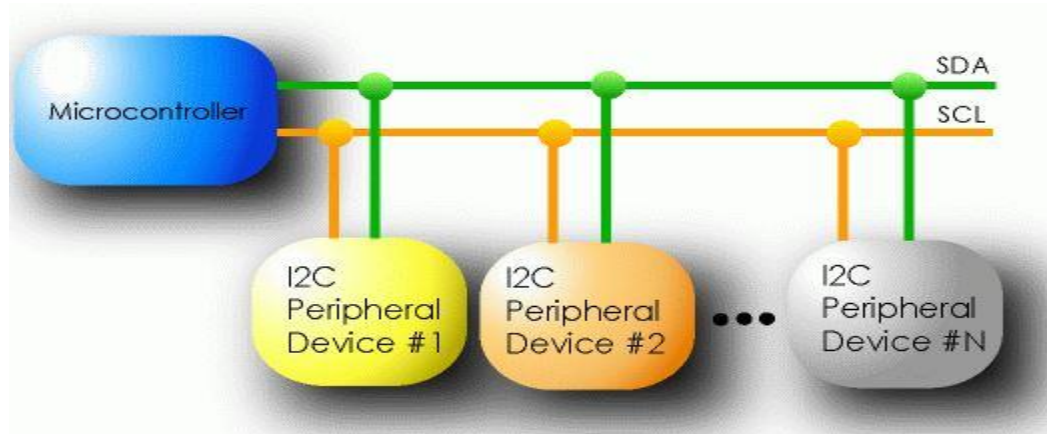


Figure 13: I²C 2-wire Interface Reprinted with Permission from Microcontroller Pros Corporation

SCL and SDA are open drain drivers which means the chip can drive its output low, but it cannot drive it high. For the line to go high, both wires must be connected to the Vcc via pull-up resistors. The device initiating data transfers and providing the clock signal on the bus is called the master. The device being addressed by the master is called the slave. There can be any number of slaves and any number of masters connected to these two signal lines, which is why I²C is a multi-master protocol.

Figure 2: I2C Protocol

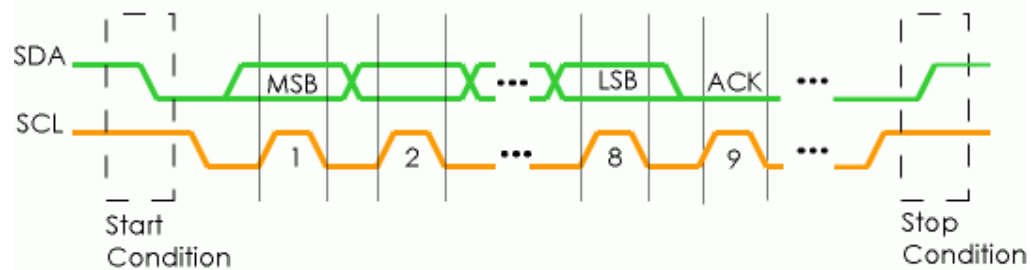


Figure 14: I²C Communication Protocol Reprinted with Permission from Microcontroller Pros Corporation

The figure above shows the I²C communication protocol. It is characterized by the following:

- 7-bits slave addresses: each device connected to the bus has a unique address
- Data is divided into 8-bit bytes
- Control bits are implemented for controlling the communication start, end, direction, and acknowledgments

The data transmission format is most significant bit first. I²C is level sensitive and therefore SDA must be stable while SCL is high. The SDA line can only change when SCL is low, with two exceptions:

- Start Condition – The master signals the beginning of a transfer by sending a 1-to-0 transition while SCL is high.
- Stop Condition – A 0-to-1 transition issued by the master while SCL is high, signals the end of a transfer

The communication procedure is as follows:

- Master issues a START condition which acts as an attention signal to all connected devices
- Master sends ADDRESS of device it wants to access, along with indication of whether the access is a Read or Write operation
- Having received the address, all IC's will compare it with their own address
- If there is no match, they simply wait until the bus is released by the STOP condition
- If the address matches, the chip will produce a response called the ACKNOWLEDGE signal

- Once master receives ACK, it can begin transmitting or receiving data
- When the master has completed its transmitting or receiving action, it will issue the STOP condition

The STOP signal states that the bus has been released and that the connected ICs may expect another transmission to start at any moment. When a master wants to receive data from a slave it proceeds the same way, but sets the Read/Write bit at a logical one. Once the slave has acknowledged the address it begins sending the requested data, byte by byte. After each data byte, it's up to the master to acknowledge the received data.

Data can be transferred at rates up to 100kbps in Standard mode, 400kbps in Fast mode, and 3.4Mbps in High-Speed mode. In Standard mode 7-bit addressing is used. In other modes, slaves can either have 7- or 10-bit addresses.

If you have many devices to connect and in addition have multiple microcontrollers in your system that can act as masters, then I²C is the interface of choice. The same holds true if you need to keep the number of interconnects, board routing and pins required for the interface to a minimum. If data must be transferred at a high speed, I²C is not the way to go since it is limited to 3.4Mbps in High-Speed mode.

I ² C	
Advantages	Disadvantages
<ul style="list-style-type: none"> • Multiple slave devices can be accessed with only 3 wires • Low cost • Easy to implement • Supports multi-master configuration 	<ul style="list-style-type: none"> • Slow speed • Short distance • Limited device addresses

Table 6: Advantages and Disadvantages of the I²C Protocol

3.5.2 Asynchronous Interfaces

Asynchronous transmission allows data to be transmitted without the transmitter needing to send a clock signal to the receiver. Instead, the transmitter and receiver must agree on timing parameters in advance and special bits are added to each word which are used to synchronize the transmitting and receiving bits. Asynchronous interfaces have the timing encoded in the data stream itself and therefore are characterized by the absence of a dedicated receive/transmit clock signal.

There is no full synchronization between the transmitter and receiver; meaning the transmitter and receiver refrains from transmitting long sequences of bits. Therefore re-syncing is usually achieved through start and stop bits (frames):

- Start bit: Marks the beginning of a new frame, alerts the receiver that a word of data is about to be sent, and forces the clock in the receiver into synchronization with the clock in the transmitter
- Stop bit: Marks the end of a frame

Frames of information must not necessarily be transmitted at equal time space, since they are independent of the clock. In short, asynchronous data is self-synchronizing. The table below summarizes some key parameters for the most popular asynchronous interfaces.

Technology	Multi-Master Support	Typical Data Rates	Network Topology	Cable	Max. # of devices/segment	Max. Cable Segment Length
Ethernet	yes	10/100/1000 Mbps	Star (Hub or Switch)	UTP3/UTP5/UTP5e	2 (point-to-point)	100m (300'). Max 5 segments (4 repeaters): 500m (1500')
Ethernet	yes	10 Mbps	Daisy Chain with terminating resistors	RG58 Coax	30	185m (555'). Max. 5 segments (4 repeaters): 925m (2775')
RS485 (UART based)	no	0.3 kbps ...1 Mbps	Daisy Chain with terminating resistors	STP, 24AWG	32 (2-wire bus), 64 (4-wire bus)	1330m (4000') @64kbps, much more with repeaters
RS422 (UART based)	no	0.3 kbps ...1 Mbps	Daisy Chain with terminating resistors	STP, 24AWG	10	1330m (4000')@64kbps, much more with repeaters
RS232C (UART based)	no	0.3 kbps ...128 kbps	Point-to-Point	Various (ribbon, STP, UTP)	2	15m (45') @ 19.6 kbps, ~1.5m (4.5') @ 128 kbps
LIN (UART based)	no	20 kbps	Daisy Chain	single wire	16	40m (120'), more with repeater
CAN	yes	20 kbps ...1 Mbps	Daisy Chain with terminating resistors	STP/UTP	128	1000m (3000') @40kbps, 40m (120') @1Mbps, much more with repeaters
USB	no	1.5/12/480 Mbps	Star	STP	2 (point-to-point), 127 per root host controller via USB hub	5m (15') Max. 7 segments (6 repeaters/hubs) 35m (105')

Figure 15: Summary of Common Asynchronous Interfaces Reprinted with Permission from Microcontroller Pros Corporation

3.5.2.1 UART Based Interfaces

To communicate with external components such as microcontrollers or computers, a controller could use a component called Universal Asynchronous Receiver Transmitter (USART). More specifically the UART is a commonly used piece of hardware which translates data between parallel and serial communication mediums. The following sections cover multiple standards used for transmitting data that the UART supports

3.5.2.1.1 Serial Data Standards

The RS232 is a standard that utilizes the RS232 ports (serial ports) on computers. Years ago, noise was avoidable in circuits due to the lack of filters and robust algorithms etc and signals were weak because of poor wiring. To compensate for the signal loss, high voltages were used. The RS232 uses +/- 12V levels on the line, but they may vary widely as the detection is specified at +/-3V. There is a line driver, usually from the MAX232 family that converts these levels to and from the internal digital signal levels of a computer or microcontroller.

In short, the RS232 standard is a pure point-to-point connection of two devices. If single master-multiple slave bus systems are needed the RS422 and RS485 standards are the way to go. It's possible to build multi-master RS242 and RS485 systems, but it requires development of your own software protocol to hand bus arbitration in case multiple masters want to transmit at the same time. Other interfaces, like CAN and Ethernet handle bus arbitration in hardware and are better suited for multi-master systems.

RS232	
Advantages	Disadvantages
<ul style="list-style-type: none">• Many compatible legacy devices• Relatively long distance• Immune to noise• Easy to implement	<ul style="list-style-type: none">• More suitable for system to system communications than chip to chip/sensor• Low speed for long distances• High part count which adds to system cost• Single master single slave system

Table 7: Advantages and Disadvantages of the RS232 Standard

RS485	
Advantages	Disadvantages
<ul style="list-style-type: none">• Very long distance• Immune to noise• Easy to implement• Higher speeds beyond 115200 baud	<ul style="list-style-type: none">• More suitable for system to system communications than chip to chip/sensor• High part count which adds to system cost

Table 8: Advantages and Disadvantages of the RS485 Standard

3.5.2.2 USB Protocol

Some things to consider about the RS232 is that it is big and bulky, the use of level shifters add parts, and most modern computers, especially laptops do not have serial ports. Nowadays more microcontrollers have support for USBs which have smaller connectors, are faster, and work with the plug and play architecture of modern operating systems. The main disadvantages are that the firmware is much more complex and the PC side requires a device driver.

USB	
Advantages	Disadvantages
<ul style="list-style-type: none">• Fast• Plug and Play• No level translators and small connectors• Ability to implement standard devices without writing a device driver	<ul style="list-style-type: none">• Complex Firmware and PC software• Requires OS specific device drivers and installers• PC application software is more OS specific• Short cable lengths may require use of an expensive hardware bus analyzer

Table 9: Advantages and Disadvantages of the USB Protocol

3.5.2.3 Ethernet Protocol

Nowadays Ethernet is one of the most widely deployed networks in offices and industrial buildings, making it ubiquitous. It is based on standards that ensure reliability of network connections and data transmission, therefore ensuring interoperability. Most microcontroller devices are equipped with USB or serial connections. Adding a network connection allows these devices to transmit data at a much higher rate than normally possible.

Ethernet	
Advantages	Disadvantages
<ul style="list-style-type: none">• Very high speed (10Mbit to 100Mbit/s)• Long cable lengths and commodity equipment• PC API is mostly OS independent• No device drivers required	<ul style="list-style-type: none">• More suitable for system to system communications, not so much for chip to chip/sensor• High part count and complex hardware• More complex firmware that either serial ports or USB• Device detection is no plug and play

Table 10: Advantages and Disadvantages of the Ethernet Protocol

The specific situation our control panel will be used for will determine the communication system that will be implemented. For our design, the Ethernet protocol will be used to connect power plant simulator to the soft panel application. UDP data transmission will be sent over this connection.

3.6 Networking with UDP Data Transmission

The soft panel application and power plant simulator need a way to communicate over the network. While our project only focuses on this single point-to-point connection, the power plant simulator is designed to communicate with a multiple number of stations. The use of multiple stations is outside of the scope of our project, but in order to support the ACTIVE research team, our design will include this possible system expansion.

In order to communicate with every station, the simulator uses multicasting to deliver feedback messages to its clients. A multicast is the delivery of a message to multiple receivers using just one single data transmission. A single transmission follows a one-to-many distribution procedure. This means that multiple devices receive the transmission simultaneously, as shown in the figure below.

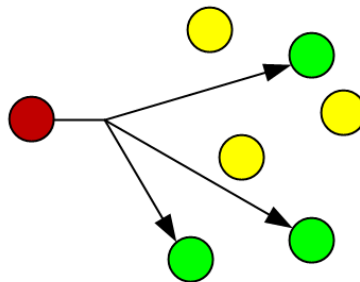


Figure 16: Multicasting

The power plant simulator will always use multicasting when sending a message. This means that all existing stations receive every message that the power plant simulator sends out. It is their duty to determine whether or not a specific message applies to them.

On the contrary, when a station sends data to the simulator, multicasting does not need to be used. Other stations don't care about this data. Only a one-to-one distribution is desired. This is known as a unicast.

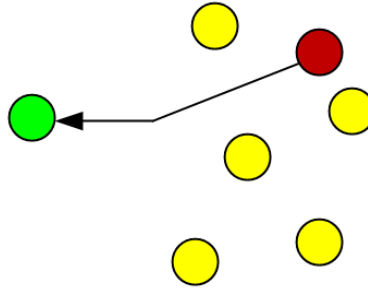


Figure 17: Unicast

The IP delivery method that will be used for all network data transmissions is the User Datagram Protocol (UDP). UDP is a simple networking method for client/server network applications, often used as an alternative to Transmission Control Protocol (TCP) in order to achieve more real-time performance. Both UDP and TCP are forms of connectionless communications, meaning that each data transfer unit is individually addressed and routed based on the information it contains.

The data transfer units used in a UDP network are called datagram packets, or datagrams for short. Datagrams have two components: a header and a data payload. The header contains all the information that is necessary for routing. It has four fields: The source port number, the destination port number, the datagram size, and the checksum. Each field is two bytes long. As one might have guessed, the port numbers are used to identify the senders and receivers. The datagram size is a count of the total number of bytes contained in the header and data payload. Because the header has a fixed size of eight bytes, the datagram size tells us how large the message size is. This is an important field as it is used by the receiver to properly parse the message. The checksum is used to ensure that there was no error during transmission. If the checksum sent does not match the checksum received, the software will know that an error has occurred.

In order to communicate with the simulator, our application will need to support UDP multicasting. This includes proper multicast addressing as well as sending and receiving multicast datagrams. More specifically, it will need to specify its port, or socket, for datagrams to be sent and received.

Every device participating in a network that uses Internet Protocol needs an IP address. This address specifies its only connection to the network. The span of IP addresses is divided into classes based on the high order bits of a 32 bit address, as shown in the following figure.

Class	Leftmost bits	Address range
A	0	1.0.0.0 to 127.255.255.25
B	10	128.0.0.0 to 191.255.255.255
C	110	192.0.0.0 to 223.255.255.255
D	1110	224.0.0.0 to 239.255.255.255

Figure 18: Class Breakdown of an IP Address

These classes determine which part of the address belongs to the network address and which part belongs to the node address. Multicast addressing falls in Class D, which means that multicast datagram destination addresses must start with “1110.” Class D needs to be used because there are no host addresses in this range. In multicasting, all hosts within the group share the group’s IP address. The application will have to choose which multicast IP to listen to and send on. This will need to be done dynamically, and the choice will depend on the computer’s IP address.

Once data comes through a network and into a computer, it needs to know where to go next. There are many applications on a computer. If an application requires a network connection, it must bind itself to an empty port on the computer. As stated before, a datagram knows what port the receiver is connected to. This is how it knows where to go after it enters the computer. The following figure shows how this works.

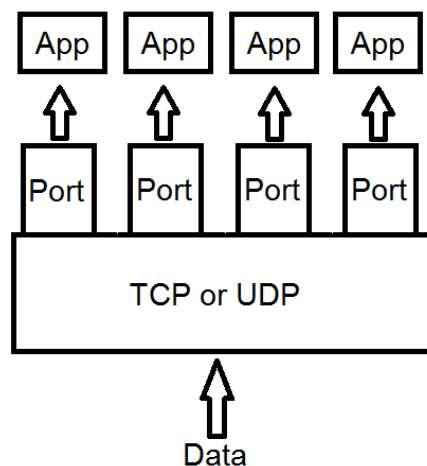


Figure 19: Port Binding

Port numbers range from 0 to 65,535 because ports are represented by 16-bit numbers. The application will bind to one of these ports and will stand by, listening to the chosen multicast IP and remain ready to send and receive data at any time.

Before the application can send data, it will also need to know how to create the packets. This includes giving them proper headers and payloads. Oracle provides classes for implementing all of the above procedures in Java. These classes are contained in the *java.net* package. The two main classes that will be used from this package are *DatagramSocket* and *DatagramPacket*. A *DatagramSocket* object will represent the socket that will be used for sending and receiving datagram packets. When it comes to packet creation, Java conveniently builds the header and payload. These classes will make handling the UDP very simple. Using the available methods, UDP multicasting can be implemented in just a few lines of code.

UDP is a fast and simple method of data transmission that will ensure quick communication between our application and the power plant simulator. But one fact that cannot be ignored is that because of its simplicity, there is potential for problems to arise. In this protocol, individual packets are allowed to be received in a different order that they were sent. Also, if a problem arises packets can be dropped and no retransmission will occur. It is true that we are designing a small system with a low amount of traffic being sent between the client and the server, so this is just a small problem to worry about at the moment. In a real power plant system, UDP networking could not be the primary method of data transfer, as proper communication is critical.

3.7 Electrical Design

This section covers our research completed on microcontrollers, our power supply and our plan for our PCB design.

3.7.1 Microcontrollers

A microcontroller is a small computer used for embedded applications. They incorporate a processing core, communication interfaces, input/output peripherals, memory for data and memory for program storage. Before jumping into the various available microcontrollers, we must first hash out the details of the system that must be researched before choosing a microcontroller.

- **Peripherals:** The two common types to define are the communication interface and the interface that will dictate the number of pins that will be required by the microcontroller (Digital inputs/outputs, analog to digital inputs, PWM's etc).
- **Software Architecture:** Must decide on the types of algorithms, frequency, and run time to estimate the amount of processing power needed and whether to use an 8 bit, 16 bit, or 32 bit microcontroller.
- **Identify programmability, memory needs, and examine costs and power constraints.**

One aspect of the Nuclear Analog to Digital design is to provide our hard panel that will emulate what an analog control panel looks like in an actual nuclear plant. The hard panel will accommodate upwards to 100 analog controls, therefore requiring the microcontroller to have sufficient input/output pins or adding the use of I/O expanders. The soft panel holds responsibilities that require it to both establish a connection and to transmit/receive data from the hard panels microcontroller, so that must be accounted for. All in all, the microcontroller must operate under low power, but also have the computing power and versatility to drive the various numbers of I/O's, support signal conversion, and provide Ethernet compatibility.

Finding the right microcontroller tailored to your project is not easy. There are a plethora of manufacturers offering a number of peripherals, architectures, power consumption, and much more. The following sections will cover in detail the number of available microcontrollers at our disposal.

3.7.1.1 Freescale HCS12 Family

The MC9S12NE64 is a powerful and robust chip operating on a 16-bit central processing unit. It is ideal for low-end connectivity applications, consists of various on-chip peripherals, and comes in multiple pin packages. This microcontroller has many features as detailed below:

- 16-bit Processor
- 64 Kbytes FLASH EEPROM
- 8 Kbytes RAM
- 80-pin or 112-pin Package
- 50 MHz Processing Speed
- Serial Interfaces: SCI, SPI, & IIC
- Ethernet Media Access Controller (EMAC)
- Ethernet 10/100 Mbps Transceiver (EPHY)
- Sophisticated timer functions that include: input capture, output compare, pulse accumulators, & real-time interrupt

The HCS12 is a high-speed 16-bit processing unit. Its high performance gives way to full 16-bit data paths and internal registers for extended math instructions. The architecture is identical to the HC12 and is upward compatible with the HC11 CPU. The HCS12 provides efficient memory access because the CPU maps all registers and peripherals into a single linear address space; this provides us with the memory needed to satisfy our applications. Enhanced indexed addressing capabilities are available which lead to new instructions and addressing modes to support high level languages.

The MC9S12NE64 comes equipped with a 64 Kbyte flash module that includes a 64 Kbyte Flash nonvolatile memory. The flash memory is ready in multiple ways: bytes, aligned words, misaligned words. The read access time is two bus cycles

for misaligned words and one bus cycles for bytes and aligned words. Our design at times will be using single-supply applications and this module is ideal for this sort of program because it allows field reprogramming without requiring external voltage sources for program or erase.

The HCS12 comes embedded with a powerful timer module (TIM) to support time-based functions. Timer functions are important for the following applications:

- Frequency Measurement
- Event Counting
- Time Delay Creation & Measurement
- Period and Pulse Width Measure
- Arrival Time Comparison
- Time-of-day Tracking
- Waveform Generation
- Periodic Interrupt Generation

If we were to implement timer functions without a timer module we would find that it consumes a substantial amount of processing power, inefficient in responding to short time period events, and is troublesome in terms of programming. With the timer module these operations can be handled more efficient. The HCS12 timer module consists of the following:

- 4 channels of input capture
 - Measures pulse-width, period, and duty cycle
- 4 channels of output capture
 - Generates time delay, trigger action, and creates complex digital waveforms
- 16-bit pulse accumulator
 - Counts occurrences from external events and measures frequency
- 16-bit counter

The group might need to make use of the features that the timer module provides to achieve functionality when the analog controls interact with the digital controls.

The 50MHz processing speed this microcontroller provides is not the fastest operating frequency at our disposal. There are other microcontrollers within the same HCS12 family that operate at speeds up to 80MHz. The group is currently unsure if the 50MHz is fast enough to perform the necessary operations and communications of our design.

A key component in our microcontroller decision is Ethernet capability. The MC9S12NE64 comes standard with an Ethernet Media Access Controller (EMACV1) and Ethernet Physical Transceiver (EPHYV2). Both modules are

IEEE 802.3 compliant and support the medium-independent interface (MII) and the MII management interface. The features of both modules are the following:

- IEEE 802.3 compliant
- Medium-independent Interface (MII)
 - Supports both 10Mbps & 100Mbps data rates
 - Data and delimiters are synchronous to clock references
 - Provides independent 4-bit wide TX/RX data paths
 - Provides a simple management interface
- Full-duplex and half-duplex modes
- 1:1 common transformer
- Address recognition
- 2 RX and one TX Ethernet buffer interfaces
- Single RJ45 connection
- 2.5V CMOS and 2.5V MII interface
- 125MHz clock generator and timing recovery

The Ethernet media access control is designed for communication with devices that support MII and is specifically optimized for 8/16 bit processors. The key components of the EMAC are the receiver, transmitter, MAC flow control, MII management, and RX/TX Ethernet buffer interfaces. The EPHY has the ability to be configured to support 10BASE-T or 100BASE-TX applications and operate in 5 basic modes: Power down/initialization, auto-negotiate, 10BASE-T, 100BASE-TX, and low power. These modules will be of great help in our design of communicating between the master microcontroller and PC application.

Another standout feature of the MC9S12NE64 is the Port Integration Module (PIM). This module establishes the interface between the various peripheral modules and the I/O pins for all the ports. The port integration module supports 10 ports and is featured in the following ways:

- Port [A, B, E, K]: communicates with core logic and multiplexed bus interface
- Port [T]: connects to the timer module
- Port [S]: communicates with the 2 SCI and 1 SPI modules
- Port [G, H, J]: connects to EMAC module
- Port [L]: connects to EPHY module

Each port in the PIM contains a set of control registers where each individual I/O pin can be configured for the following:

- Input/Output selection
- Drive strength reduction
- Enable and select of pull-up resistors
- Interrupt enable and status flags

Each pin supports general –purpose I/O capabilities. On top of that each pin can also function as an output from a peripheral module or an input to a peripheral module. This module will aid the group in dealing with pin configuration.

Another member in the same family as the MC9S12NE64 MCU is the MC9S12XD256CAG. The differences with this MCU is that it operates at a higher processor performance, more flash memory, higher pin packages, and offers the XGATE co-processor which can pump peripheral data in and out from I/O pins. A downside with this MCU is that it does not include a built-in Ethernet support. If this route is chosen, there will be a need to implement Ethernet controllers into the design.

3.7.1.2 megaAVR Family of Microcontrollers

The megaAVR family offers a wide selection of microcontrollers in terms of pin-counts, memories, and peripherals. For applications that require a substantial amount of code, the megaAVR provides plenty of program and data memories that can perform up to 20 MIPS. MegaAVR also provides picoPower for select devices, which employs multiple clock domains to enable different parts of the MCU to be individually clocked, in order to lower the clock frequency, and therefore optimize power consumption. For a design that is looking for self-programmability, efficient power consumption, high integration, and advanced analog capabilities the megaAVRs are an ideal family to MCUs to select from.

The microcontroller from this family that is under consideration is the 8-bit ATmega325. This high performance, low power MCU is embedded with an 8-bit CMOS processor that is based on the AVR enhanced RISC architecture. This MCU is able to reach throughputs upwards to 1 MIPS per MHz by executing instructions in a single clock cycle. Other features of the ATmega325 are as follows:

- 16 MHz operating frequency
- 32 Kbytes ISP Flash memory
- 2 Kbytes SRAM
- 1 Kbytes EEPROM
- JTAG interface for on-chip debugging
- 32 general purpose working registers
- 64-pin package with 53 programmable I/O lines
- 0-8 MHz at 2.7-5.5 V speed grade
- Serial interfaces: SPI, TWI (I2C), UART

The 8-bit RISC CPU utilizes Harvard architecture in order to maximize performance and parallelism. Instructions in the program memory are carried out using single level pipelining therefore, while one instruction is being executed the next is pre-fetched from the program memory. The instruction set with 32

general purpose working registers, which are directly connected to the Arithmetic Logic Unit (ALU). These registers allows for 2 independent registers to be accesses in one single instruction during a single clock cycle. As a result, the architecture is more efficient considering the code and reaching throughputs up to ten times faster than the conventional CISC microcontrollers. This is of much value to our design because in order for our MCU to execute the applications of the board and networking with the slave MCUs and server we must first optimize the speed and throughput of the CPU.

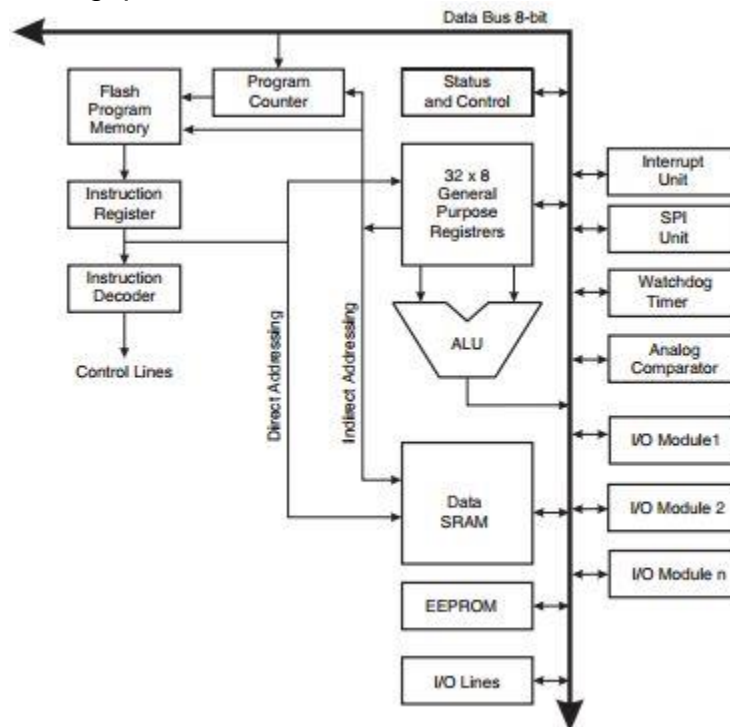


Figure 20: Block Diagram of AVR Architecture Reprinted with Permission from Atmel

There are two main memory spaces in the AVR architecture, the Data Memory and the Program Memory space. The ATmega325 also has a special feature of having an EEPROM Memory for data storage. The on-chip in-system reprogrammable (ISP) Flash memory contains 32 Kbytes of program storage. More specifically, the program storage is broken down into two other sections: Boot Program section and Application Program section. Considering the Data Memory section, there are five different addressing modes that the section covers: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. The 1 Kbytes of data EEPROM memory is organized as a separate data space. Here each single byte may be read or written with an endurance of at least 100,000 write/erase cycles.

The ATmega325 supports multiple communication interfaces as shown in the table below.

Serial Peripheral Interface	USART	Universal Serial Interface (USI)
<ul style="list-style-type: none"> • Full duplex 3-wire synchronous data transfer • Master or slave operation • LSB/MSB 1st data transfer • 7 programmable bit rates • End of transmission interrupt flag • Double speed master SPI mode 	<ul style="list-style-type: none"> • Full duplex • Asynchronous or synchronous • Master or slave clocked synchronous operation • 3 separate interrupts on TX complete, data register empty, and RX complete • Multi-processor communication mode • Double speed asynchronous mode 	<ul style="list-style-type: none"> • 2/3-wire synchronous data transfer • Data received interrupt

Table 11: Communication Interfaces

These interfaces offer a solution to communicating back and forth with the PC application and its slave MCUs.

3.7.1.3 Microchip PIC32 Family

Microchip's 32-bit family of microcontrollers is recognized for their high performance processing power, memory, and peripherals. The microchip network is worth noting because of its available software, development tools, and pin/peripheral compatibility from the 16-bit product lines.

The microcontroller we will be researching in this family is the PIC32MX664F064L. This MCU is embedded with a processing core that operates on 5 stage pipeline Harvard architecture and operates at speeds up to 80MHz. Other special features include power management modes and a debugging interface. The remaining features of the MCU are detailed below:

- 32-bit MIPS M4K Core Processor
- 512 Kbytes Flash
- 5 Stage pipeline, Harvard architecture
- 64 Kbytes RAM
- 10/100 Ethernet MAC with MII/RMII Interfaces
- Serial Interfaces: SCI, I2C, SPI
- Ethernet Media Access Controller (EMAC)

The MIPS32 M4K processor is the same throughout the PIC32 family. The responsibilities of this CPU are to fetch and decode instructions, fetch source operands, and execute each instruction and write the results to the proper destinations. The MIPS32 M4K is a 32-bit Harvard architecture with a 5 stage pipeline and consists of a set of logic blocks that are described in the following:

- Execution Unit
- Multiply/Divide Unit (MDU)
- System Control Coprocessor (CPO)
- Fixed Mapping Translation (FMT)
- Dual Internal Bus Interfaces
- Power Management
- Enhanced JTAG Controller

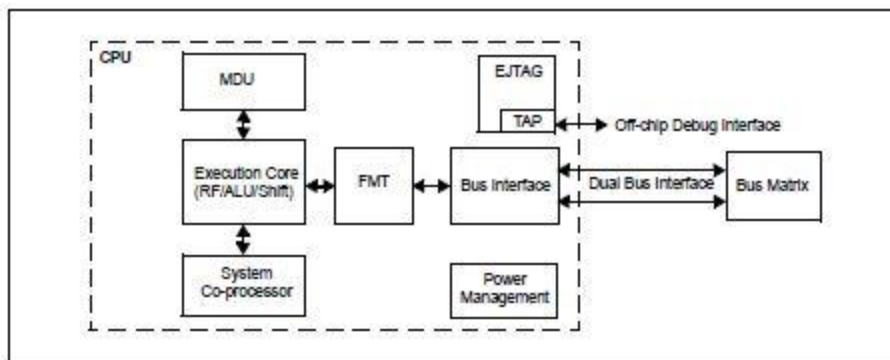


Figure 21: Processor Block Diagram Permission from Microchip via the Fair-use Copyright

These blocks work together, in parallel, to provide an efficient high-performance processing core. The operating speed of 80MHz is the fastest of the MCUs we have reviewed. This will be more than sufficient for the calculations needed in our design.

The microcontroller comes equipped with 512 Kbytes of internal Flash program memory for executing user code and other functions. There are multiple ways by which the user can program this memory:

- Run-Time Self-Programming (RTSP)
 - Performed through software execution from either Flash or RAM memory
- EJTAG Programming
 - Performed using the EJTAG port and an EJTAG capable programmer
- In-Circuit Serial Programming (ICSP)
 - Performed using a serial data connection to the device which allows faster programming times than RTSP

The 512 Kbytes of Flash and 64 Kbytes of RAM is enough memory to handle the size of the code for all the applications to be implemented.

An interrupt controller is a device that is used to combine various sources of interrupts onto one or more CPU lines. When a device has multiple interrupt outputs, the interrupt controller organizes them in order of their relative priority. For example, there are numerous mechanisms that will trigger the array of LEDs on the control board and indicate that an event needs immediate attention. This trigger that signals the LEDs to blink is the interrupt. The interrupt control module exists externally to the CPU logic and prioritizes the interrupt events before presenting them to the CPU.

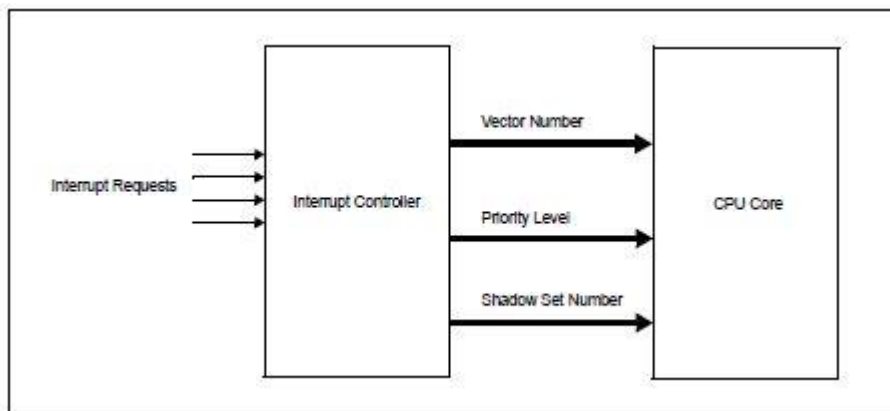


Figure 22: Interrupt Controller Module Permission from Microchip via the Fair-use Copyright

When you are able to transfer data without CPU intervention, it allows the CPU to operate in a more efficient manner. The Direct Memory Access (DMA) controller is a master bus module that is used to transfer data between multiple devices without the use of a CPU. The data is transferable between any source and destination of the memory mapped modules existing in the microcontroller or memory itself. The following are some key features of the DMA controller module:

- 4 identical channels
- Automatic word-size detection
- Fixed priority channel arbitration
- Flexible DMA channel operating modes
- Multiple DMA channel status interrupts
- DMA debug support
- CRC Generation module

This MCU pin count package is 100 pins where 85 of them are general-purpose input/output pins. These pins allow the microcontroller to monitor and control other devices. In order to add flexibility and functionality some pins are multiplexed with alternate functions, which depend on the peripheral features on

the device. The pin count meets our requirement for our control board and also adds flexibility to the design of the pin configuration.

There are a various number of external peripherals that our master microcontroller will be communicating with. The PIC32 comes standard with multiple serial communication interfaces to accomplish this task: SPI, I²C, and UART. Not only will our microcontroller have to communicate with external peripherals but it will have to communicate with a server, which is the soft panel in our design. An Ethernet controller will be used to create a network between our MCU and PC. An Ethernet controller is essentially a master bus module that interfaces with an off-chip physical layer to implement a complete Ethernet node in a system. Notable features of the Ethernet controller are described in the following:

- 10/100 Mbps data transfer rates
- full-duplex and half-duplex operation
- RMI and MII PHY interface
- MIIM PHY management interface
- Manual & automatic flow control
- Fully configurable interrupts

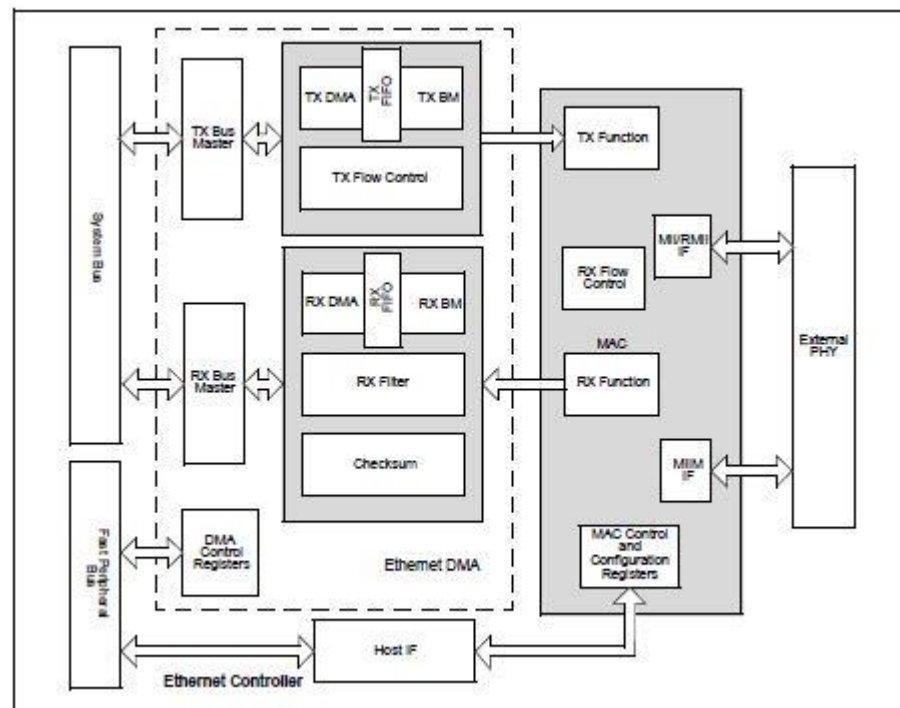


Figure 23: Ethernet Controller Block Diagram Permission from Microchip via the Fair-use Copyright

3.7.1.4 Microcontroller Decision

Choosing the correct microcontroller is one of the more crucial decisions for the entire design. Operating speed is one key factor to selecting an MCU. Both the Microchip and Freescale MCUs we researched surpassed the ATmega325 considering operating speed. Although since our design includes using one master MCU and multiple slave MCUs, the operating speed does not need to be 66 MHz and therefore the ATmega325's 16 MHz will suffice.

Another key factor is the amount of memory is at our disposal. It's not difficult to see that a control board with upwards to 105 analog controls will need a substantial amount of internal memory to carry out the necessary applications and code. Atmel's ATmega325 does not support the largest amount of memory between the MCU's researched, but considering the specifications and cost the ATmega325 remains our best solution.

Our design requires the use of slave MCU's to carry out the actions of various analog controls on the board. There will be an MCU controlling the LED array and other MCU's covering other analog controls such as the gauges and valves. Whichever MCU the group decides on, it must be able to communicate with these slave MCU's. Because the slave MCU's come from the ATmega family, it made sense to use a master MCU in the same family in order to ensure a smooth transition. The ATmega325 offers serial peripheral interface, universal serial interface, and USART interfaces.

Even though the ATmega325 may have been outperformed by the other researched MCUs, the group decided to choose the ATmega325 for other reasons. After careful review of our specifications and requirements we find that the high operating speed and large amount of internal memory we originally thought we needed does not stand true anymore. Our division of controls into subsystem of slave MCU's forced us to consider having the master MCU also in the ATmega family. Functionality and communication between MCUs, from top to bottom, is a key parameter and that along with cost is what drove the group to decide on the ATmega325.

3.7.1.5 In-System Programming

Because of the large number of controls in our complete system, it can be expected that design implementation will not be simple, especially when building the analog control panel (hard panel). This section pertains to a specific approach we will take in order to simplify the design of the hard panel.

Several microcontrollers will be used to drive the 100 embedded controls in the hard panel. This gives a sense of the complexity involved with the circuits that will be built. During this structuring, a high amount of testing will be required. It can be expected that during this time the microcontrollers will go through a large amount of reprogramming. It can be concluded that the microcontroller must be

reprogrammable, that is, there shall have no limit in the number of times software can be written and erased. This is not a very limiting factor in our choice of MCUs, as most are indeed reprogrammable.

What is a limiting factor, however, is the ability to program and reprogram our MCUs while installed within the system. This is known as in-system programming (ISP). If all of our microcontrollers support this, we will have the advantage of combining two stages into one. These stages are the programming stage and assembly stage. By allowing reprogramming of the chip without removing it from the circuit, ISP will save us time and reduce risk for error.

There are many options available when choosing an ISP supported microcontroller programmer. The decision of which one will be used will depend on the following factors: Ease of use, cost, risk, and support.

The first option to consider is a Direct AVR Parallel Access (DAPA) cable. This method involves using a parallel port on the computer your programming on to connect directly to the MCU. The follow figure shows how this works with an Atmega16.

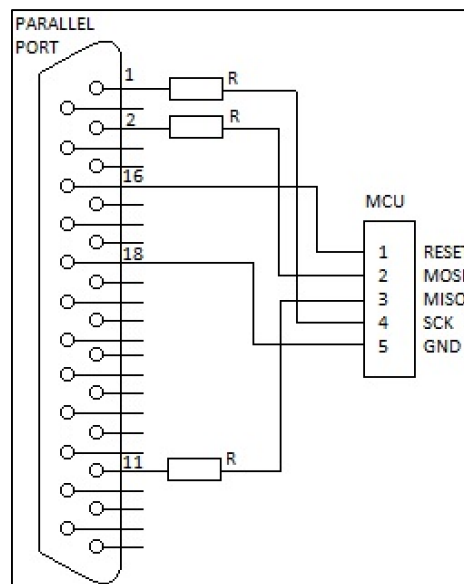


Figure 24: Using DAPA as a programmer

It is safe to say that this is the simplest and cheapest option. However, there is the possibility that the computer's port can be damaged if the voltage across the MCU exceeds 5V. This is the reason for the resistor usage shown in the figure above. This is just a minor risk to consider when choosing this option. The main reason we would consider other programmer options is because most modern computers don't have a parallel port. This would restrict us to using an older computer which is not something we desire.

A very popular in-system programmer option is an Arduino board. This is a much supported method because Arduino offers firmware to implement this action.

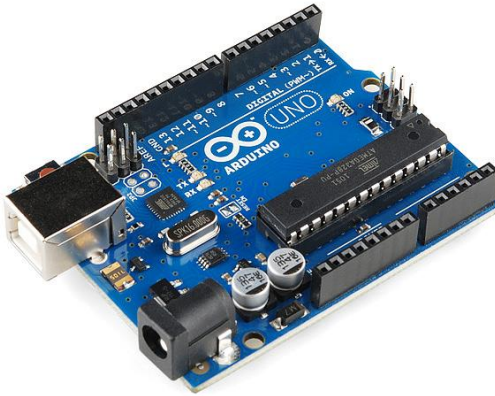


Figure 25: Arduino Uno as an ISP, image reprinted with permission from Sparkfun Electronics

One of the most powerful ISPs designed specifically for AVR microcontrollers is the AVR Dragon. This is designed for professional development and offers a lot of high level options and support. While this could be beneficial in several ways, we expect that this choice would result in a lot of wasted resources and thus is not worth the expense.

We more interested in products that are low cost and simple to use. Most importantly, they must be designed to work best with the multiple AVR microcontrollers that will exist in our circuit. Many of these programmers are available in the market today. With all the existing support, these programmers are easy to use and available at a low cost.

3.7.2 Power Supply

One of the main requirements for our analog system is that it be a plug and play device. This means that the operator should be able to simply plug the control panel into a wall outlet that supplies the American standard of approximately 115VAC with a frequency of 50 to 60 Hz. Since our devices all operate at low voltages this will need to be stepped down to a minimum of 5V and capable of at least 20 mA of current, more of course is desired. The optimum minimum voltage is 3.3V but it is not required for the performance of our device yet would be nice to reduce the amount of energy used.

In addition, we would like to transmit data using DC signals to minimize the effect of noise so the voltage will need to be converted from an alternating current to a direct current. Not only do we want the power supply to be a low voltage isolated source with DC capabilities, we would like it to be efficient by conserving power and energy as well. This means there will need to be a close look into how the AC signal is stepped down and that there is a very low generation of heat during

the process. The final requirement is that the system, once turned on, will need to provide a constant power supply to the analog system.

3.7.2.1 Power Supply Technologies

There are many techniques available for designing our power supply subsystem. The first step is to start narrowing down the options based off of our system and design requirements. The main thing to consider is whether or not the power supply will need to be linear or switched-mode and then the advantages/disadvantages of both.

Linear power supplies generally consist of a transformer that manipulates the electrical and magnetic properties of electricity to dissipate the input voltage over the many windings of a coil. In order to do this the input voltage cannot be varied because the number of windings in a transformer is very specific. If you have a 120VAC transformer that outputs 30VAC but instead you apply a much higher input voltage, not only will you not get a 30VAC output, you could do some very serious damage to you devices connected to this output. After this transformer stage where the AC voltage is lowered, it will then be rectified (either full or half wave) and run through a filter. This signal is then driven through some large electrolytic capacitors to convert it to a DC signal. The final step is to regulate this DC voltage into a smaller one. Since for our system we require a 5VDC source a LM7805 rectifier can be used to obtain the value we need. This is where the tricky part is because LM7805 have a maximum input voltage at which it will still convert the signal to approximately 5VDC. Through previous experimentation it was found that the LM7805 is able to handle converting at least 20VDC to 5VDC with an overall dropout voltage of typically 2V. However, this is highly un-recommended as a large portion of this excess voltage is dissipated as heat and the larger the difference between the input and output voltages the greater the heat index becomes. A way to minimize this is to get V_{in} as close as possible to the desired V_{out} as possible. For the LM7805 it is generally recommended to get the input voltage to as low as 9V before applying it to the rectifier. A series of voltage regulators could be used to accomplish this. The LM78XX series comes with capabilities in between 5-24VDC.

Even though linear power supplies are simpler in design, they have poor energy conservation capabilities because a lot of heat is generated when stepping high voltages down to low ones. According to one source, "a linear power supply[...] will normally operate around 60% efficiency for 24V outputs, whereas a switch-mode is normally 80% or more." This in and of itself is enough to make us not look at a linear power supply as an option even though their smaller parts count makes them more reliable. Furthermore, they tend to be larger in size because of the number of coils that are required in the transformer is more significant than those used in the switched-mode power supplies.

Switched-mode power supplies are more common than their linear counterparts. Even though they are more complex and involve more components, these components are still smaller than those found in the linear power supplies. In addition to this, switched-mode power supplies are able to handle a wide range of input voltages and can be manipulated for a various range of output values. A basic overview of how SMPS work, is that the input signal is sent through a rectifier and filter, then an inverter, an output transformer and finally an output rectifier and filter. There is an additional option of making the SMPS closed or open looped. The benefit of creating a closed loop design is that the system will be able to check itself and make any adjustments in conversion needed if the output voltage is slightly off from the desired value. This type of system removes a lot of guess work and assumptions that would have to be made by the designer in addition to providing protection to the devices by making sure the power they receive is being constantly monitored and corrected if changes do occur. Creating a feedback loop is more difficult in terms of design and computation, due to the system needing to have an extra non-switching power supply to provide it with power during stand-by. However, the added protection that a feedback loop provides is well worth it.

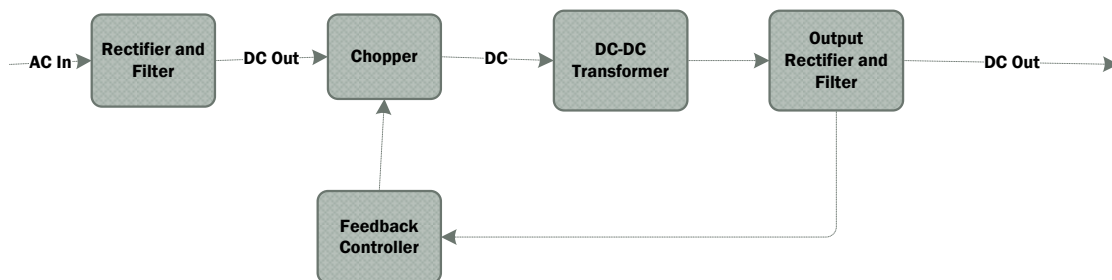
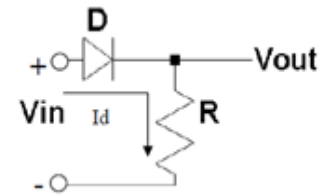
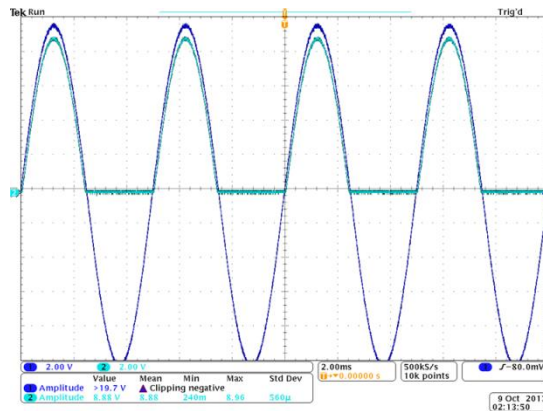


Figure 26: Block diagram of basic process flow of a closed-loop SMPS

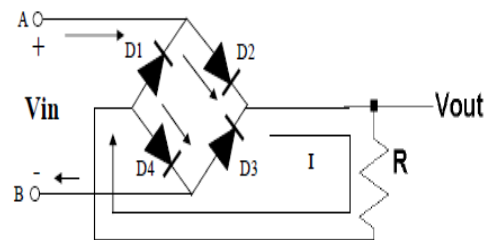
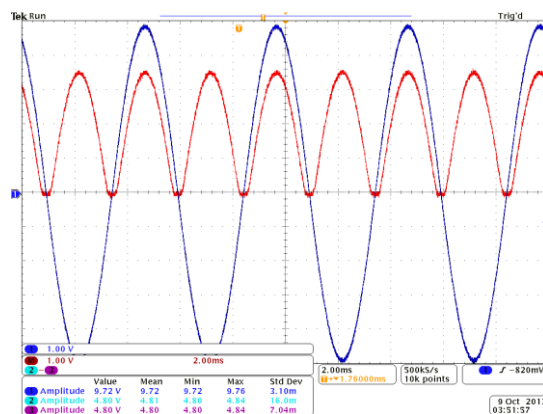
The above figure is a basic flow diagram of a closed-loop SMPS. Each stage is an important process in creating a constant low voltage DC power source for our system and are discussed in detail in the following sections.

3.7.2.1.1 Rectifiers and Filters

There are a lot of considerations that go into designing each one. The first step is the rectifier and filter stage. There are two types of rectifiers we will consider, half and full wave. As depicted in the following images, a half wave rectifier only passes half the period of the input cycle while the full-wave passes the entire signal.



(a)



(b)

Figure 27: Example oscilloscope images and schematics of (a) half wave and (b) full wave rectifiers

Both of these circuits are easily implemented with diodes and a resistor. In order to conserve energy it makes sense to use the full wave rectifier where the entire signal is used. However, there is a noticeably large difference in the ripple voltage which can create adverse effects on our signal especially since a DC signal is supposed to be as flat as possible. A way to rectify this is with a filter; hence why this is the rectifier and filter stage. A passive component filter will suffice for this step, narrowing down the options significantly to four; capacitor, inductor, inductor and capacitor, or a π filter. We will only consider the LC versus π filter shown below.

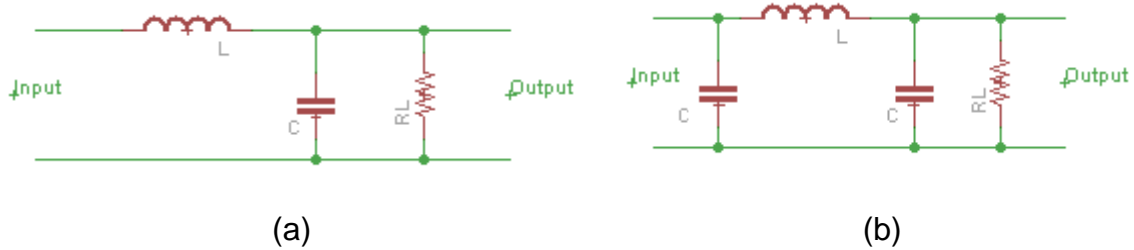


Figure 28: (a) LC Filter and (b) π or CLC Filter

They both use a combination of an inductor and capacitors before the load out resistor R_L . The combination of the two devices helps to make the ripple voltage become almost independent of the load filter. There is not much difference between the two filters besides the capacitor across the input of the inductor. This additional capacitor makes the π filter better than the LC filter because it handles the majority of the filtering leaving the other two components to remove the remaining ripples. This circuit is only very effective for low current systems which meet our requirements fine. The combination of the rectifier and filter convert the signal from AC to DC.

3.7.2.1.2 Chopper Circuit and Feedback

One of the specifications is to make the power supply isolated and this is done with a high frequency transformer. In order to accomplish this, the unregulated DC signal that was just created is fed back to a chopper circuit signified by the block called “chopper” and shown in the following figure. This signal is then used to drive the DC-DC transformer in the output transformer stage unless it is turned off by a switch. The switch is usually a variation of a field effect transistor and gives the SMPS its name. The switch has two states, on (saturation) and off (cut off). While on, the unregulated DC signal from the rectifying stage is allowed to power the transformer and it is turned off when a high frequency is detected. [2] This detection is carried out with the use of the feedback loop and an IC controller. The controller is able to identify the maximum user defined output voltage and can regulate the system based off of its readings from the feedback voltage and the input voltage. Generally they are capable of pulse width modulation so switching frequency can be varied as well. Furthermore, having this chopper circuit allows the designer to implement voltage scaling in addition to regulating the output voltage. Being able to control these features would not only help prevent power surge damages yet also, reduce the amount of energy used in our entire system.

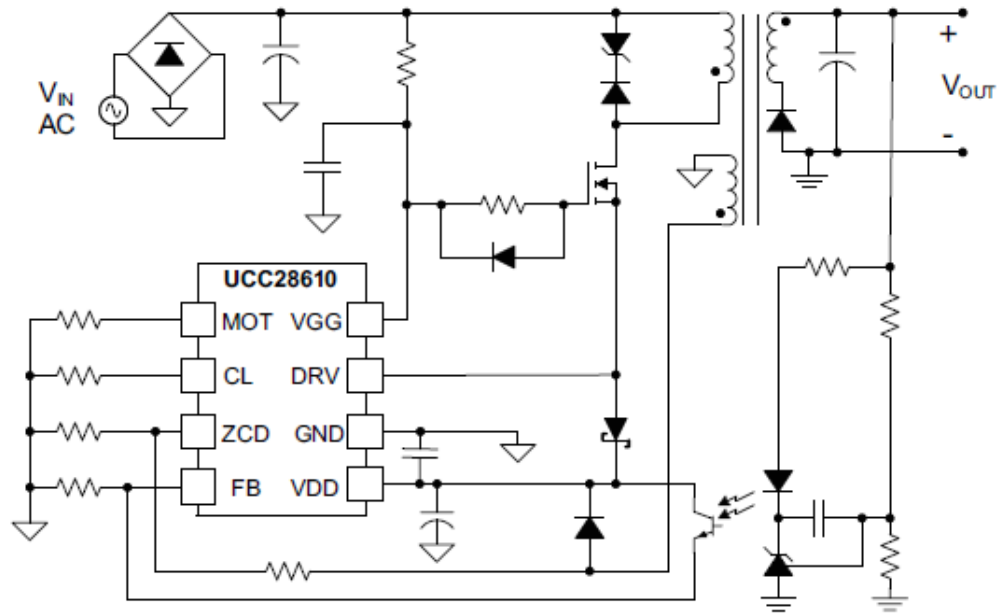


Figure 29: Example of a “Chopper” circuit reprinted with permission from Texas Instruments

In the above example of a chopper circuit one can see the rectifying and filter stage, how the signal is fed to both the transformer and the controller and the driving signal of the transformer comes directly from the controller and the drain side of the MOSFET. The image clearly shows the output voltage being fed back to the controller as well. In the following section we will discuss in detail the DC-DC transformer and the many options available for implementing one.

3.7.2.1.3 DC-DC Transformer Topologies

There are many topologies available for designing a DC-DC transformer. In order to narrow down the choices and make our system more flexible we will look at designs that are capable of handling multiple output voltages of varying values. Our system specification requires a 5V source yet many of our devices are capable of running off of 3.3V as well. It may be nice to have a higher DC source somewhere in the range of 12-24VDC. This would act as a safe guard against potential problems that may arise from the loss in DC signal strength that occurs over long distances from the built-in resistance in cable wire. However, due to cost restrictions it will be best to keep the number of outputs to a bare minimum of two. It is good to consider that there may be future issues with running our entire system off of such low voltage and that there are remedies available and have been addressed. This can be easily implemented by buying a transformer with multiple secondary windings. Since each voltage requires a separate load they cannot use the same stepped-down voltage. Furthermore, they will need a

different number of windings in order to get the DC voltage reasonably low enough to convert to the desired output size without a huge consumption of energy.

For our system we will require a step-down converter, where the output voltage is less than the input. This eliminates the boost and buck-boost topologies from our selection process because the boost increases the output voltage versus the input and the buck-boost is capable of both which is not necessary. This leaves us with a buck converter topography. A basic design of one is given below.

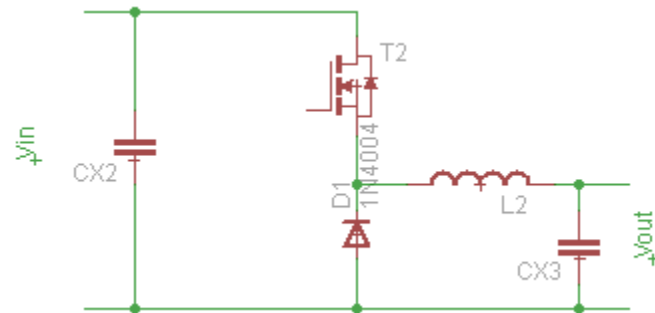


Figure 30: Schematic of simplified buck converter

Unfortunately, the buck converter is not isolated and the input/output currents are discontinuous due to the switching cycle of the transistor. In order to correct this and make the power supply more efficient at the same time is to use a flyback converter (FBT) which implements a transformer that is capable of storing energy during the on state of the system. Furthermore, the transformer isolates the output from the input. The FBT is derived from the buck-boost converter so it is capable of both operations. Below is a very simplified schematic of a FBT. As one can see, it does not require many more additional parts to implement it and the benefits far out way the cost.

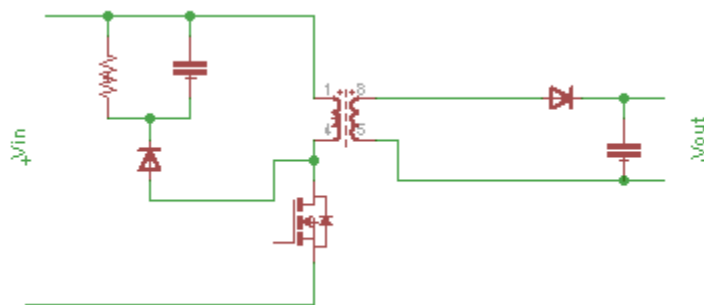


Figure 31: Schematic of simplified flyback converter

3.7.2.1.4 Output Rectifier and Filter

Depending on whether or not the designer wants to create multiple outputs of varying sizes, which is fully capable with SMPS, the output rectifier and filter will differ for each value. Once again diodes can be used for rectification and since at this point we are only dealing with DC signals, full bridge designs are not necessary. It is recommended that for lower voltages, around 12VDC and less, the use of Schottky diodes is implemented. This is because they have a lower voltage drop and faster recovery times. At this final stage, some signals might require being fed through an additional voltage regulator.

3.7.2.2 Power Supply Design Tool

Texas Instruments (TI) has created a very useful design tool called Power stage Designer Tool. The user is able to identify what type of topology they desire anywhere from an inverting buck-boost to a Weinberg circuit. Once the topography is chosen, the user is then able to state their minimum/maximum input values and their desired output voltage and current as well as a multitude of other specifications.

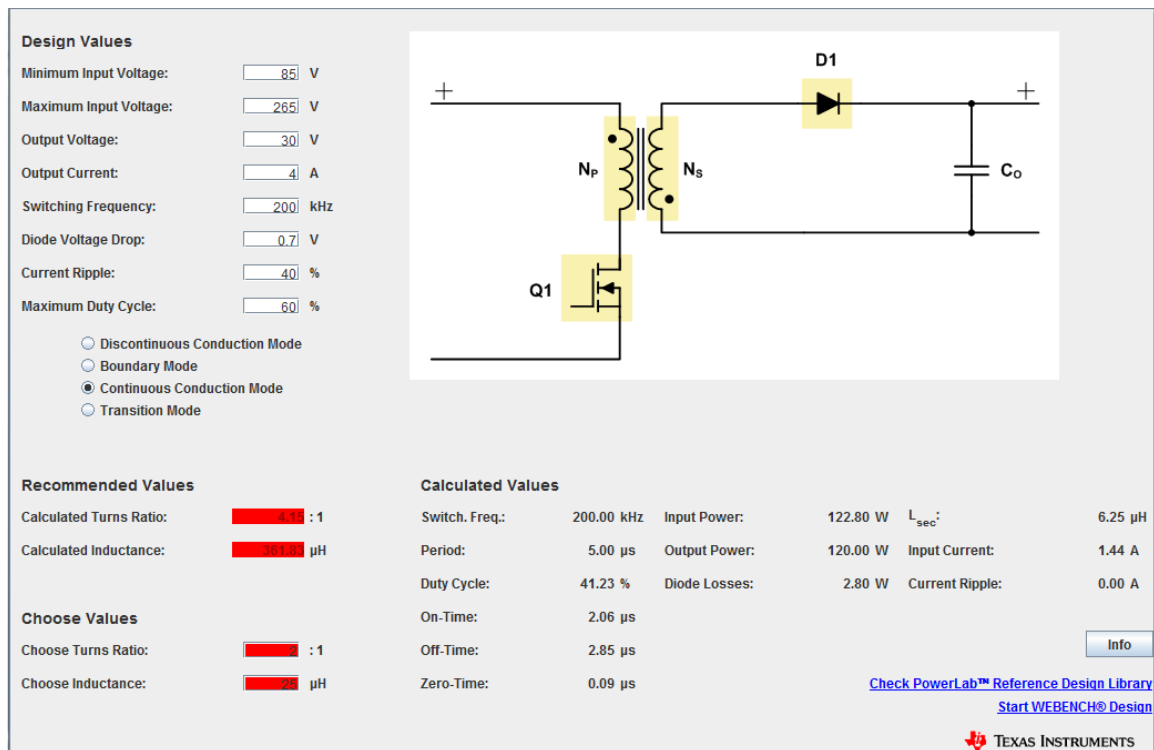


Figure 32: Screenshot image of the Power Stage Designer Tool by Texas Instruments

The software will then calculate a recommend number of turns and turn ratio for the transformer as shown in red, in order to help the designer. In addition to this,

TI has a database of over 1000 tested schematics with fully documented parts lists that meet different design requirements and are available for free through their website. The database is referred to as Powerlab Power Reference Designs Selection Tool.

3.7.3 Printed Circuit Board

Printed circuit boards (PCB) are used in the most simple to the most complex electronic devices across the world. You can find these devices in cell phones, computers, radios, and nearly every piece of electrical equipment used in our day to day lives. A PCB serves two main purposes. One is to act as a place to mount the electronic components of the device and the other is to supply an electrical current to and from each of these components. Whether it is used for large-scale manufacturing or electronic hobbyists, printed circuit boards are the optimal choice for building quality electronic devices.

3.7.3.1 Anatomy of PCB

Before jumping into the various considerations for choosing a PCB we should first gain an understanding of what materials are compounded to make a one. When PCBs are being manufactured the first step is to form a base out of some sort of solid, non-conductive material such as glass or plastic. This base material is then laminated with a copper (or other conducting metal) sheet. The copper pattern that is printed serves as an avenue for current to pass through the conducting pattern at a desired rate. The non-conductive base and printed copper pattern are the only materials that go into making a bare PCB. When we design the layout for our PCB and send it for manufacturing, the electrical connections are created by removing select portions of copper from the bare PCB.

3.7.3.2 Layers

Printed circuit boards can be manufactured as a single or multiple layered boards. Deciding the amount of stack layers needed depends on the complexity of the design. A few things to consider when deciding on the number of layers for a PCB are the board size, cost, and placement of components. It goes without saying that single sided boards (single layered) are the cheapest and are made from just the base material and sheet of conductive metal. Single sided boards are simple to manufacture and effortless to work with. Although they are straightforward to implement they cause some difficulty when crossing electrical connections because there is only one layer of metal to work with. An external jumper would be needed to rectify this problem.

In order to avoid the complications of single sided PCBs, many simple commercial and hobbyists boards are developed on double sided PCBs. Crossing electrical connections then becomes a simple task and allows for more complex designs. Over time a design may become more complex and must

make use of additional layers. This generally comes into play when a design calls for a board consisting of complex signal paths with a compact design.

3.7.3.2.1 Layer selection

When cost-efficiency is an important factor in a design you do not want to end up with an unnecessary complex design that will waste funds and cost more to produce. For our project we will make use of multiple PCBs for the various subsystems of our control panel. We initially thought that since we are not using a single PCB for the entire system we will not need a PCB with more than two layers, and that a double sided PCB will be the most cost effective method to implement our design and will leave us with the least amount of complications. After conducting more research and gathering information on how components should be placed, we decided to use four layers for the printed circuit board. The layout will be as following:

- Signal layer
 - Allows for easy accessibility to the MCU
- Ground layer
 - Provides electromagnetic shielding
 - Lowers resistance of the path to ground
 - Assists with heat dissipation across the board
- Power layer
 - Counterpart to the ground plane
 - Behaves as AC signal ground while providing DC voltage for power circuits mounted on the PCB
- Signal layer
 - Allows for easy accessibility to the MCU .

3.7.3.2.1.1 Vias

Vias are components that are necessary for multi-layered boards. This main component electrically connects one layer to another. There are three types of vias to consider when selecting a PCB:

- Through hole: a hole is drilled through the entire board and is then electroplated so that it is conductive
- Blind: used in design with more than two layers to connect a surface layer to an internal layer without going all the way through
- Buried: similar to blind via but are only used to connect internal layers

Through hole vias are the most common type and will be used in our design.

3.7.3.3 Design overview

There are many things to consider in designing a PCB. Before jumping into the physical design we must first have a clear idea of what functions the design will accomplish. Some basic parameters that are important for designing the circuits of our subsystems are choosing parts, selecting the right vendor, and sketching the connections. Once we cover these bases we can then follow through with the final preparations for a PCB layout.

3.7.3.3.1 CAD Packages

An important aspect of designing a PCB is choosing the right CAD package for the schematic connections. This is of importance because in order to have our PCB manufactured, we must first design a schematic and then send it to an appropriate vendor. The amount of CAD software at our disposal can be frightening. After some research some of the more popular PCB design software available is EAGLE, KiCad, and gEDA. All three are free, offer open source software, and come with no limitations.

3.7.3.3.2 Choosing a Manufacturer

Choosing the right vendor is more important than what most would consider. After much research we found that no two PCB vendors are alike and each one has their own limitations. Excluding PCB specifications, low cost and quick turnaround times are the two main constraints in selecting a PCB vendor. The following table was made to help decide on which vendor to choose.

Manufacturer	Information
OSH Park	<ul style="list-style-type: none">• Standard 2 Layer<ul style="list-style-type: none">• \$5 per square inch for 3 copies of your design• Orders are 2 to 3 times a week with 12 day turnaround• Standard 4 Layer<ul style="list-style-type: none">• \$10 per square inch for 3 copies of your design• Orders are 2 to 3 times a week with 2 week turnaround• Specifications<ul style="list-style-type: none">• Gold finish for solderability and environmental resistance• 2 layer boards are 1.6mm thick with 1oz copper on both sides• 4 layer boards internal copper is 0.5oz• Minimum 6 mil traces with 6 mil spaces and 13 mil drills with 7 mil annular rings• Internal cutouts are supported

Table 12: Osh Park PCB Information

Manufacturer	Information
Advanced Circuits	<ul style="list-style-type: none"> • 2 layer design <ul style="list-style-type: none"> • Same day to 3 day turnaround • 4 to 10 layer design <ul style="list-style-type: none"> • one day to 5 day turnaround • Specifications <ul style="list-style-type: none"> • Minimum .010" inner layer clearance • All hole sizes are plated through • Green solder mask with white legend • 1oz copper for inner layers and up to 2oz for outer layers • Trace/space to 5/5 mils

Table 13: Advanced Circuits PCB Information

Manufacturer	Information
ExpressPCB	<ul style="list-style-type: none"> • Standard double sided boards <ul style="list-style-type: none"> • Tin/lead plating • Perimeter routed to shape • No silkscreen or solder masks • Small orders shipped next business day • Double sided board specs <ul style="list-style-type: none"> • Max board size of 12x14 inches • Minimum square board is .64x.64 inches • Minimum trace/space width is 0.006 inches • MiniBoard PCB <ul style="list-style-type: none"> • 2 or 4 layer boards • Tin/lead plating • Minimum board size is 3.8x2.5 inches • \$51 for three 2-layer boards or \$98 for three 4-layer boards • Fast manufacturing

Table 14: Express PCB Information

3.7.3.3.2.1 PCB Specifications Example

(3 total: LED subsystem, Gauges subsystem, Master MCU subsystem)

In order to select an optimal PCB we must hash out the specifications. We will use CAD software to design the board layout and then send it, along with the

specifications, to the appropriate vendor. The vendor will then inform us if our design needs any modification.

- 4 layers
- Board length: 6in
- Board width: 8in
- Board thickness: 0.06in
- Distance between signal layers: < .01in
- Minimum trace width: .01in
- Minimum hole size: 0.032in

4.0 Project Hardware and Software Design Details

The following sections describe the details our group came up with in order to implement our design.

4.1 Aesthetics

Our project needs to look as similar as possible to an established analog control panel in a nuclear power plant. In order to meet this specification we have consulted with the ACTIVE lab on specific parts they would prefer us to use in addition to our team pursuing making our panel aesthetically pleasing. Some of the things we are working on in order to accomplish this are labels for all of the parts, plastic covers for the LED arrays, and the design of our housing unit.

4.1.1 Hard Panel Housing Unit

The analog control panel is a large part of our overall design. Since it is what will be seen most during our presentation, it is important that the design of the housing unit is taken under careful consideration. Instead of having a vendor manufacture us a housing unit we found it cost effective to design our own.

The first step was to decide what material to use for the cabinet. After much research on wood and various types of plexiglass we decided on sheet metal. The advantages of using steel to construct our panel are as follows:

- A steel panel weighs less than a panel made from the aforementioned material
- Sheet metal gives us the flexibility of various shapes and bends we can produce that other materials would not give us
- Metal gives us much more space to work with under the panel
- No more expensive than the other materials and is often a faster process

Sheet metal is not a difficult product to acquire and therefore we will most likely purchase from a local hardware to avoid shipping costs. The following are the tools needed to construct the cabinet.

- Metal Brake
- Angle Grinder with cutoff wheel
- Drill press
- Hole saw kit
- 1/16" steel plate
- Drill bits

The ACTIVE Lab has given us access to their robotics lab which has many of the equipment items needed to build the cabinet.

We will begin by pressing a large sheet of paper against the steel plate and mark the diameters of each bend, placement of the bend, and where the holes for the controls should be placed. Once all the markings are made on the plate we will use the metal brake to make the appropriate bends. We will then use the drill press and hole saw kit to cut the openings for the controls and holes for the screws respectively. Once we make sure that all the controls fit into their respective openings we will disassemble and use the angle grinder to smooth out the gouges.

4.1.2 Control Device Labels

Due to confidentiality restrictions, we cannot make an exact copy of existing device labels for our control panel. The alternative which was provided to us by the research team would be to organize our components by a seven character alphanumeric string. Each component will have a different label in order to identify it for both testing purposes and for our software implementation.

For push buttons and rotary switches, device labels will be located below each component. For LED sectors, a device label will be located on top of the casing, in which the light must be able to shine through. For gauges, the device label will be located within the casing. We will design each device label in Microsoft Visio and will have the labels printed on a laminated card stock for durability, with the exception of the LED sector labels. We require durability in order to assure that the labels will remain affixed to our control panel and will not fall off or become easily damaged. The LED labels have to be approached differently because the material for the label cannot impede the brightness of the LED's beneath the casing. In order to assure maximum brightness we will be printing these labels on a clear laminate which will be affixed to the casing. The text on the laminate also must be visible both when the LED's are engaged and not engaged. Black text should be sufficient but we will experiment with this when the time comes.

4.2 Hard Panel Devices

This section goes into detail about the analog parts that we will have to custom build, the gauges and LED's.

4.2.1 Gauges

The purpose of the gauge is to provide readings of the states of various components throughout the power plant. This section covers the design of the analog gauges that will be embedded within the control board.

Before going into design detail, the following sketches are shown to provide an exterior visualization of the gauge.

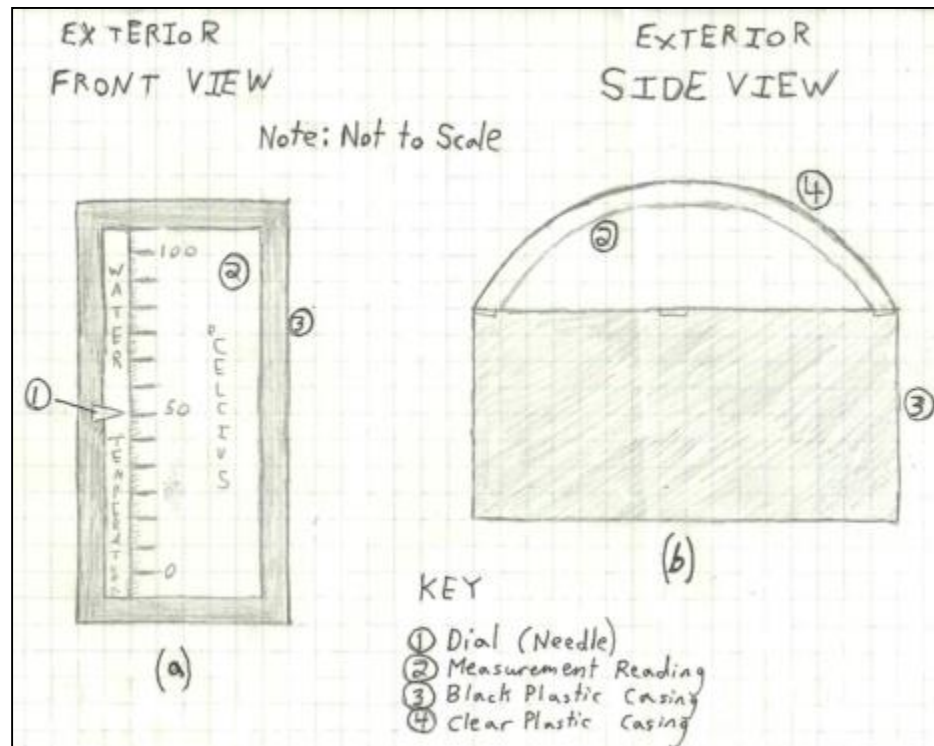


Figure 33: Exterior views of analog gauge

The reason for the semi-circular shape shown in Figure 1(b) will be explained further on.

The three main components that will be used for gauge functionality are microcontrollers, shift registers, and stepper motors.

The stepper motor is the component that moves the dial in Figure 1(a) to point to different readings on the gauge. It is a DC electric motor with a circular shape that divides a full rotation into a number of equal steps. The dial will be attached to the motor's pointer shaft which rotates accordingly. This means the dial will move along the arc of a circle, which is the reason for the gauge's semi-circular shape. A sketch of the interior gauge is shown to help visualize this process.

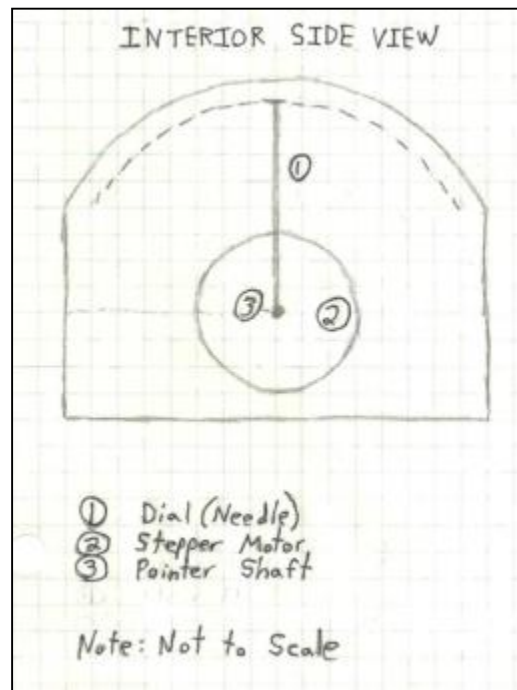


Figure 34: Interior Side View

The microcontroller will receive its commands from the master control board MCU. Listening to these commands, the microcontroller will drive the stepper motors. Our design will allow one microcontroller to drive multiple motors. These commands will specify which stepper motors need a change in state, and to what value their state needs to change to. The shift registers are used to route the data to the necessary motors.

4.2.1.1 Stepping Motor

The stepper motor must be capable of satisfying the following requirements:

- High step amount
- Dial accuracy
- Reliability
- Smooth stepping
- Low operation current
- Low power consumption

The gauges used on the control board will need to support displaying measurements with high precision. We plan to have each gauge display one hundred measurements. This means that the stepper motor must be able to rotate its shaft to one hundred different positions, all contained within the gauge viewing area. This will not be a difficult requirement to satisfy, as most motors can rotate their shafts over 300 degrees.

Our motor of choice is the Juken Switec X27.168. This motor is used in tens of thousands of automobiles today, and this fact alone supports our decision.

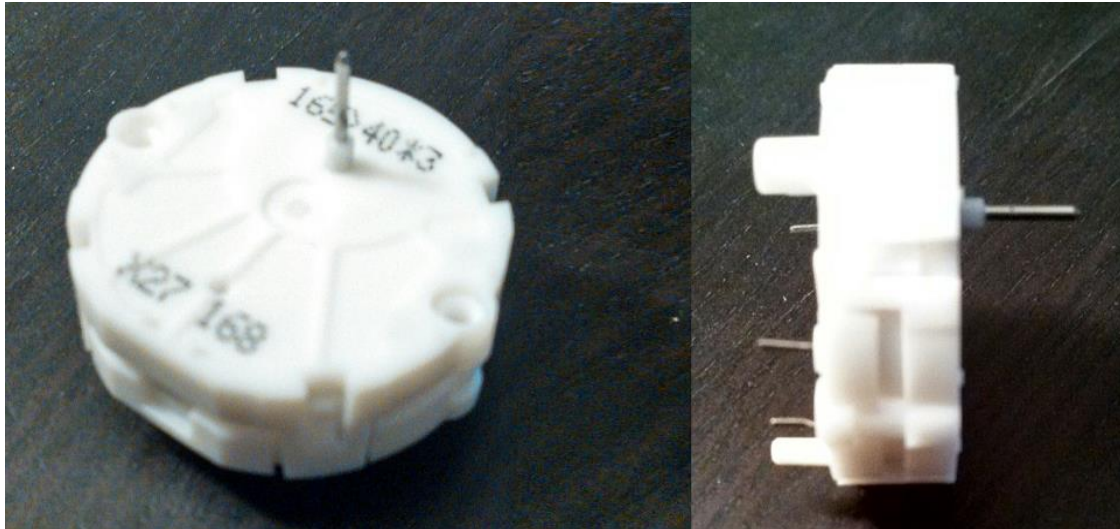


Figure 35: Juken Switec X27.168 Stepper Motor

This motor is capable of providing a 315 degree angle of rotation. It also implements partial stepping, at 1/3 degree resolution. This is a very high step amount and satisfies the precision requirement. Additionally, with proper software design, the X27 has been proven to perform smooth stepping. It can rotate its shaft at high speeds and handles acceleration and deceleration with no problem.

The following table, consisting of various technical information about our specific stepper motor, shows detailed schematics of the X27.

	Technical Data	Min	Typical	Max	Unit
1	Dynamic torque on the pointer shaft at 200 deg/s and 5 VDC supply	1	4.5		mNm
2	Maximum operating speed with appropriate acceleration			600	Deg/s
3	Angle of rotation with internal stop			315	Degree
4	Operating voltage		5	9	VDC
5	Rotating angle for an electrical period		2		Degree
6	Noise level of the motor		40		dB(A)
7	Operating temp	-40		+105	Deg C
8	Soldering temp		260		Deg C

Table 15: X27.168 Technical Data

4.2.1.2 MCU and Shift Registers

In order to drive the motors, the microcontroller must support the following requirements:

- Pulse-width modulation (PWM) to form the coil currents.
- Variable frequency generator to generate update events for the coils' current values. This helps determine rotational speed.

Our microcontroller of choice is the Atmel Atmega8. The programmable interval timer on the Atmega8 can be used as a variable frequency generator for speed control.

The shift registers that will be used are Texas Instrument's 74hc595. Each shift register will be capable of controlling two motors. The Atmega8 is capable of controlling six registers, so twelve gauges can be designed in this system. The schematic of this twelve-gauge system is shown.

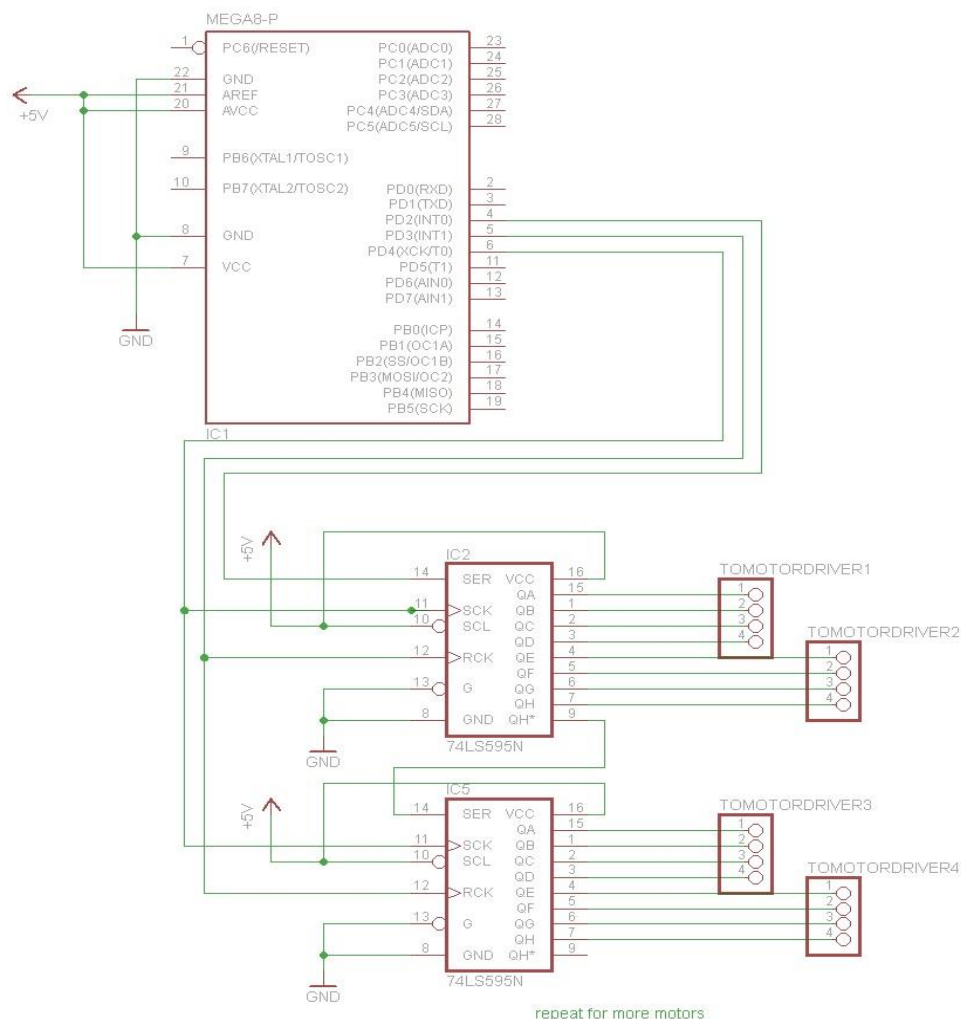


Figure 36: A gauge system supporting up to twelve gauges

One of our project requirements is to include a total of 24 gauges in the control panel. This infers that we will be building the circuit in Figure 5 twice. Two separate microcontrollers will be required to drive all 24 gauges. Following the schematic, we expect this hardware design to be easily implemented. This does however make the software design of the control board more complex, which will be covered within Section 4.7.

4.2.1.3 Detailed Component Design

Now that all parts of the analog gauge system have been selected, it is possible to specify detailed design of the component. The component itself will include four parts

- Plastic casing (Black casing and clear casing)
- The stepper motor
- The pointer needle (dial)
- Measurement readings

The following sketches are used to show the detailed design of the gauges. It will provide information about the parts and how they will be used. All measurements shown are precise and have been proven to be accurate. The unit used is millimeters. These measurements were made possible using the stepper motor's dimension specifications provided by Juken Swiss Technology. Circled numbers are shown in blue for reference.

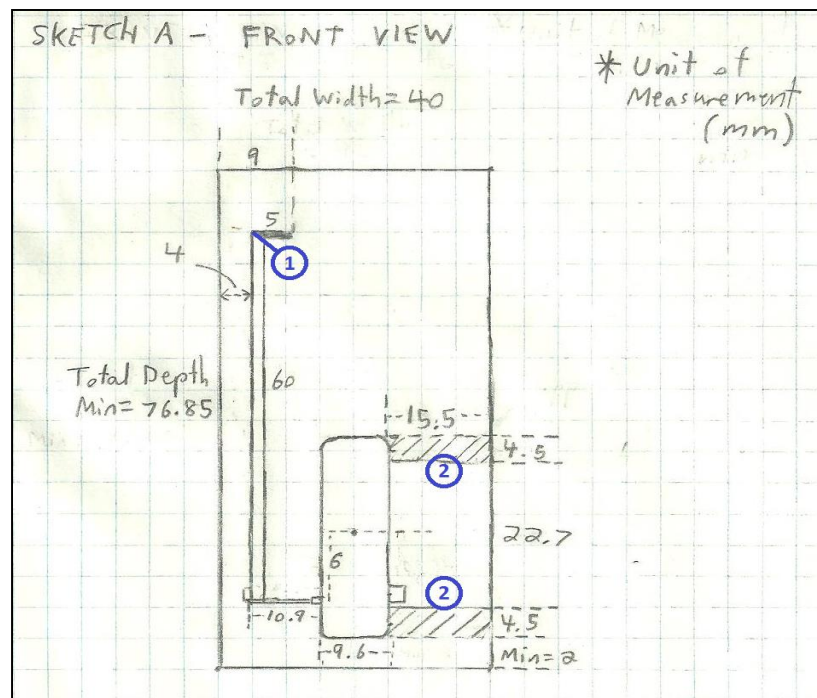


Figure 37: Sketch A - Interior Front View of Gauge

Sketch A shows the interior front view of the gauge (as opposed to top view). The leftmost side of the dial (1) will reside 4 mm from the leftmost part of the gauge. The pointer portion of the dial is 5 mm long, implying that the measurement readings will begin 9 mm from the leftmost part of the casing. The stepper motor will be secured to plastic supports (2) 15.5 mm from the rightmost part of the casing. These specifications make the total width of the gauge 40 mm.

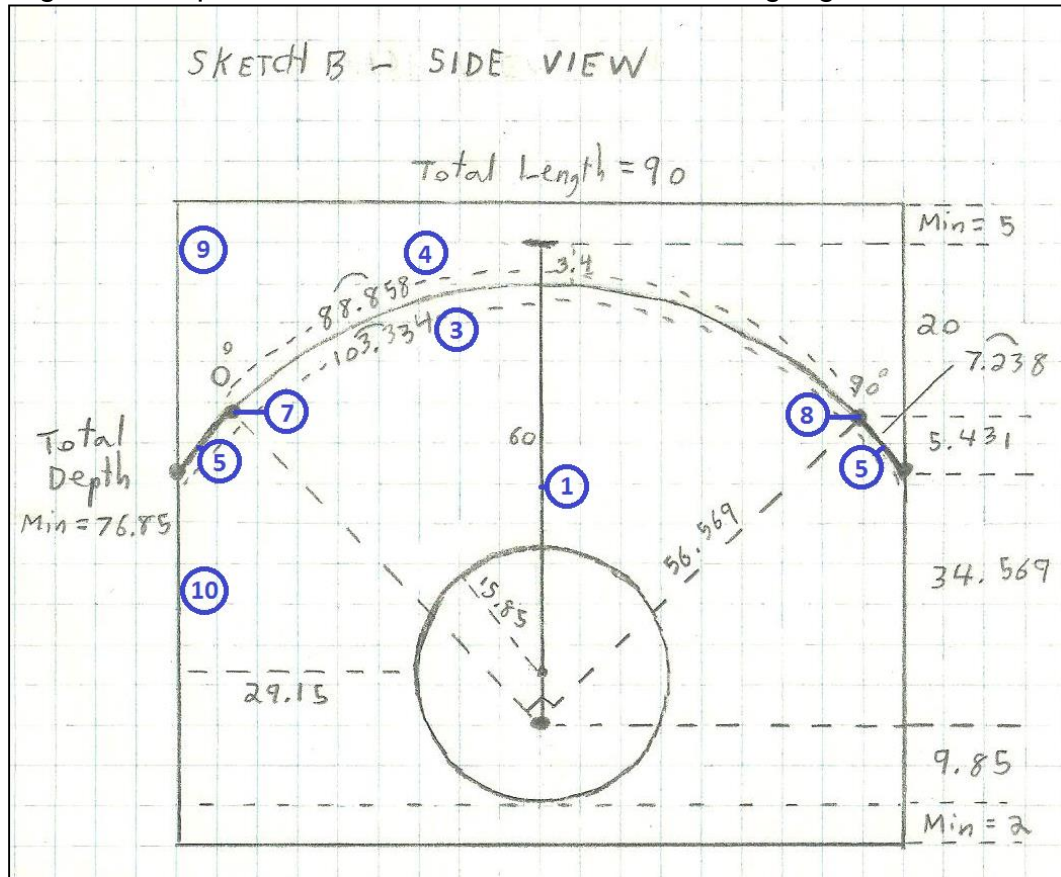


Figure 38: Sketch B – Interior Side View of Gauge

Sketch B shows the interior side view of the gauge. As shown, the total length of gauge has been designed to be 90 mm, and the total length of the dial (1) to be 60 mm. When pointed to 45°, the top of the dial will stand a minimum of 5 mm below the clear plastic casing (9). With these specifications, it can be determined that the total arc length in which the dial may travel is 103.334 mm (3). The shaded drawings (5) represent plastic clips which will be used to hold the paper measurement readings at the correct arc. These clips are the dividing point between the clear casing (9) and the black casing (10). A feature of this clip-support design is that it allows the pointer to reach the minimum (7) and maximum (8) readings without colliding into the plastic casing. The ends of each clip are the locations of the min and max readings, respectively. The arc length of these values is equal to 88.858 mm (4). This length will be used during the design and printing of the measurement readings.

An important requirement of our project is to provide precise and reliable readings at all times. The accurately derived measurements shown above will ensure complete gauge functionality. Additionally, we believe that this design will work perfectly with the complete hard panel system. Its dimension of 90x40x77 millimeters is a comfortable size with respect to the other panel components.

4.2.2 LED's

The following sections describe the design parameters for both the LED Light Box on the hard panel as well as the status LED's and in-sync LED's.

4.2.2.1 LED Assignments and Description

The LEDs assignments and descriptions to form the light box are arbitrary. As long as the names are in between 1-7 characters and the light box consists of 25 individual working sections it will suit the design requirements. There has been a question of whether or not a feature should be added so that the researchers can change the LED color for different cognitive recognition experiments. To decide if red grabs the user's attention more or a flashing blue light to determine if color actually has an effect on user performance. For this red green blue LEDs (RGB) would be needed. In addition, there is a concern that one LED will not be bright enough for then end design. The light box will be covered with an opaque material that will have the labels for each individual light as well as everything partitioned to resemble the light boxes currently in use. Furthermore, the light box will need to span the entire top of the panel. Since we have to keep the lights at a maximum of 25 we will have to make it look as if there are more lights by spreading out the lights and leaving partitioned squares empty.

LED	Name	LED	Name
1	XB7	14	JNN0
2	XB8	15	JNN00
3	XB9	16	JNKR
4	CT4	17	WNKR
5	CT5	18	DTBP
6	CT6	19	RODO
7	KBB	20	K9U
8	KBB1	21	DAI0
9	M2Z	22	EBY1
10	M3Z	23	CN7
11	M4Z	24	C4T
12	M5Z	25	BR9
13	JNN		

Table 16: Table naming each individual LED

4.2.2.2 LED Hardware Design

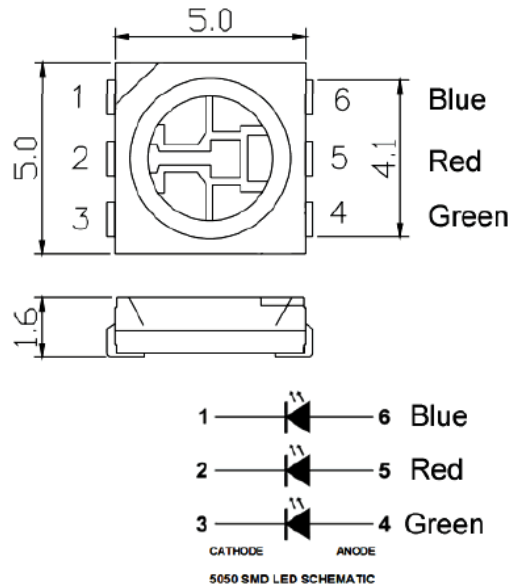
The use of a microcontroller is desired to create the individually addressable LED matrix. The microcontrollers will be programmed with an Arduino Uno, a description of this technique is given in detail in section 4.7. There are two microcontrollers under consideration for this part of the project. Their pros and cons are addressed in the chart given below.

ATmega32u4:		ATmega16:	
Advantage	Disadvantage	Advantage	Disadvantage
32 Kbytes of Flash	26 Max I/O Pins	32 Max I/O Pins	lowest operating voltage is 4.5V
USB Transceiver		SMD	16 Kbytes of Flash
USB Full Speed		Price: \$6.73	No USB Interface
lowest operating voltage is 2.7V			
Price: \$6.04			
SMD			

Table 17: Table listing the pros and cons of two microcontrollers in consideration for use

After taking the chart into deliberation, the ATmega32u4 has more desirable features than the ATmega16. The fact that it is USB compatible is a huge plus as it will make the programming much easier. Furthermore, it is faster and can operate at lower voltages which will allot for a more flexible design. There may be an issue with the number of I/Os but at the moment there is just enough to suit our needs.

The LEDs to be used are the RGB 5050 surface mounted devices supplied by Super Bright LEDs. Their compact, all in one design makes them desirable for this project. An image of the chip layout can be found below.



Notes

1: All dimensions are in millimeters.

Figure 39: RGB LED SMD chips reprinted with permission from Super Bright LEDs

The forward voltage requirements for each color as well as forward current and the current limiting resistor for each color is in the table below. The resistance values were found using Ohms law and the assumption that the supply voltage will be 5V. As added protection a zener diode will be placed in between the load and power source since the LEDs are extremely sensitive. The resistors will need to be placed in series with each LED and transistor as well.

Condition	Forward Voltage:			Forward Current:		Resistor[Ω]
	Min[V]	Typical[V]	Max[V]	IF[A]	IFpeak[A]	
R	1.8	2	2.2	0.02	0.10	150
G	3	3.2	3.4	0.02	0.10	90
B	3	3.2	3.4	0.02	0.10	90

Table 18: Table listing electrical specifications for the LEDs

The microcontroller that will be used is the ATtiny2313 by Atmel as part of their AVR Tiny family of microcontrollers.

Part	Core Size	# of I/Os	Supply Voltage Min	Supply Voltage Max	Case Style
ATTINY2313-20SU	8 bit	18	2.7V	5.5V	SOIC
	# of Pins	Program Memory Size	EEPROM	RAM	CPU Speed
	20.00	2KB	128Byte	120 Byte	20MHz

Table 19: Table listing the specifications for the mini-microcontrollers to be used

A nice feature of the 2313 is that it is capable of pulse width modulation, has a timer and a comparator. Furthermore, communication between the master MCU will be possible through either I2C, SPI or UART. This makes it a very flexible choice and will allow for smoother integration with the other subsystems in the control panel.

Furthermore, the small SMD design will allow for a more compact plan which is good because the chip only has 18 I/Os. Two of which will be needed for communication with the master MCU and two are inputs to the oscillating amplifier leaving only 14 left for the LEDs. Each LED has three inputs that need to be addressed for red, green and blue. This means that each slave MCU is capable of controlling five RGB LEDs. There are 25 LEDs in our matrix so 6 Tiny2313 MCUs will be required to drive the entire system. Each of these MCUs will contain the address for the LEDs connected to them but the brunt of the code with the instruction set will be contained in the main master MCU which will obtain its inputs from another MCU that will read in the user inputs directly. Depending on these inputs, our MCU will send a command to turn on the light, which color to make it and whether or not that light needs to be flashing.

Field effect transistors (FET), specifically metal oxide semiconductors (MOS), will be used to act as a switch to turn on the LED.

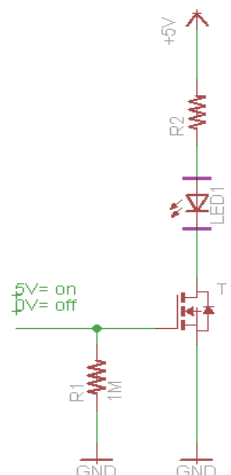


Figure 40: Schematic of an LED and MOSFET looking out of an I/O pin from the ATtiny

The right MOSFET will need to be chosen so in order to do so we must look at a very simplified version of the circuit given above which shows an NMOS connected to an LED which is connected to our current limiting resistor and then to our 5V input power supply. The resistor R1 is in place to protect the circuit during the off state, when gate voltage is supposed to be equal to zero. To pick the right NMOS we need to figure out what on-resistance is desirable to meet each LED characteristic. To do this we chose to base the design by the LED with the largest forward voltage, blue/green with $V_f=3.2V$

```
>> IDS=.02;
>> Vin=5;
>> VD=3.2;
>> VDS=Vin-VD;
>> RDS=VDS/IDS

RDS =

    90.0000
```

Figure 41: Calculations made to find on-resistance range for MOSFETs

Therefore, our MOSFET must have an on-resistance much less than 90 Ohms and a drain source current in the range of $100 < I_{DS} < 200mA$. The 2N7002 N-channel MOSFET by Diodes Incorporated is a good pick for our device. It has a low on resistance of 7.5 Ohms when the gate voltage is 5V and a maximum drain current of 210mA as well as fast switching speed and minimal input capacitance.

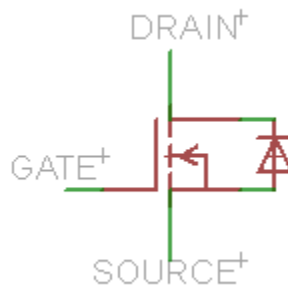


Figure 42: Equivalent circuit of the MOSFET

In addition, the LEDs will all be surface mounted. In order to save money, the PCB will be designed so that it can be cut and sectioned off for each square that will be individually attached to the panel from the back and the front will have an overlay of plastic to make it appear that the lights are a light box with well-defined sections.

4.2.2.3 LED Block Diagram and Schematic

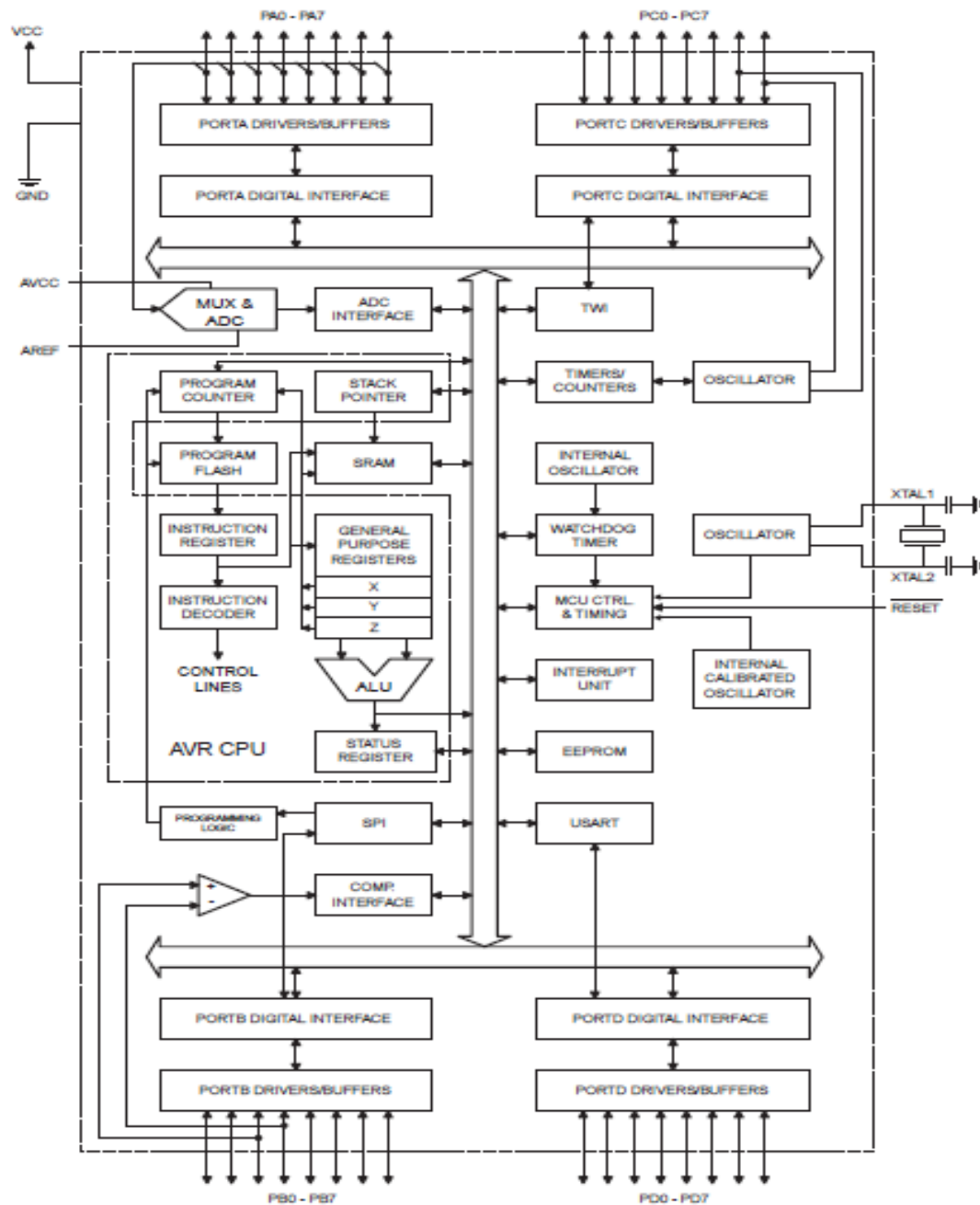


Figure 43: Block Diagram of the ATmega32u4 microcontroller reprinted with permission from ATMEL

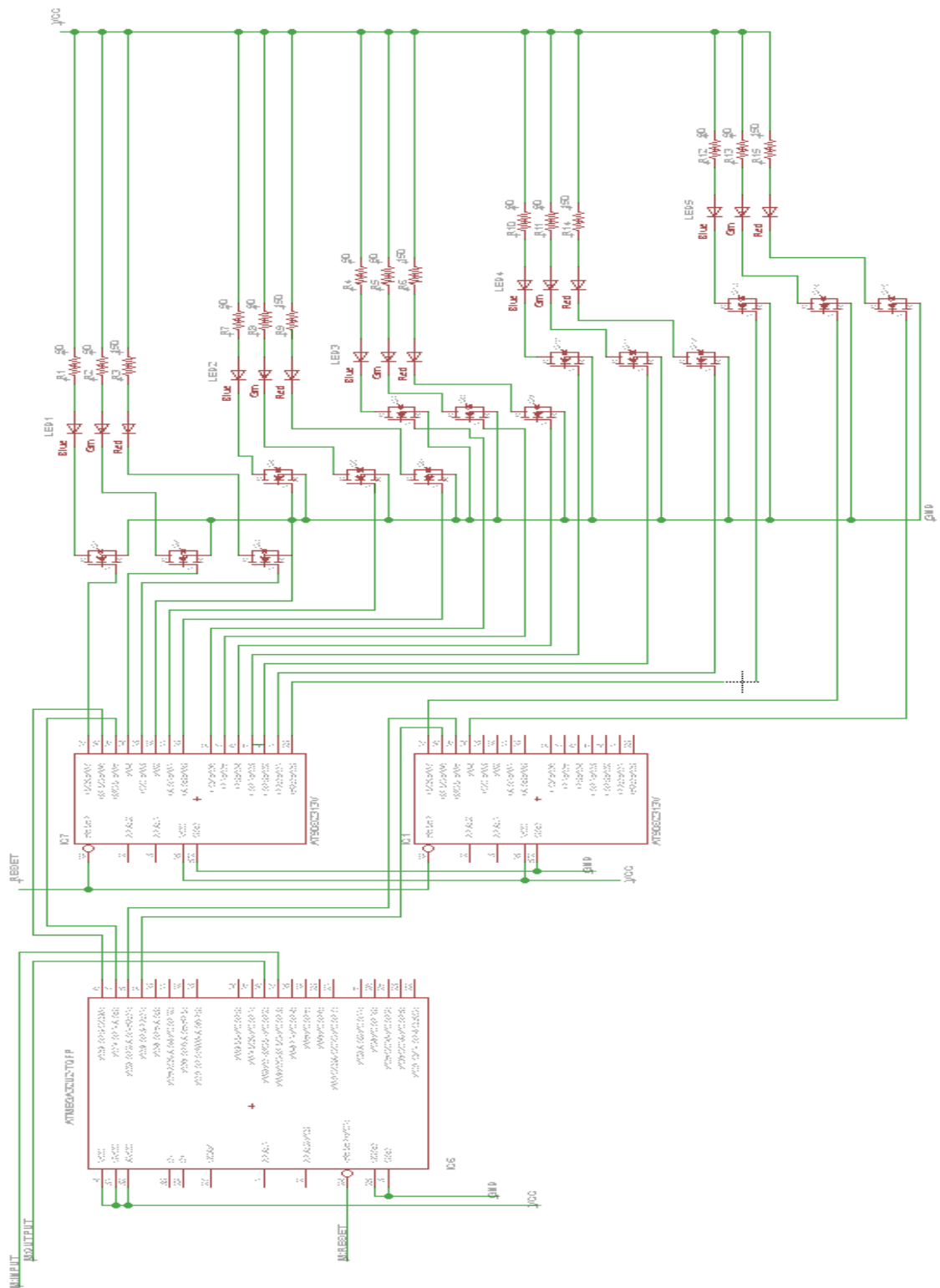


Figure 45: Light Box LED schematic showing the first five RGB SMDs wired into the slave and master controller

Since the above schematic is difficult to read and follow with only five RGB LEDs it was decided to keep the simplified version for the report so that readers will have an easier time interpreting the schematic. The other LEDs will be wired in a similar fashion with their connections going in a descending order.

4.2.2.4 Panel Mounted Light Hardware Design

It will be easy for us to implement the status lights for the rotary switches. They can be tied directly into the switches themselves since they do not need to communicate with the system and are purely for the benefit of the user. An example of how this will be done is given below.

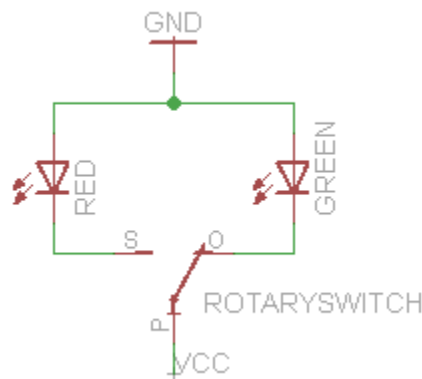


Figure 46: Simplified circuit for status lights

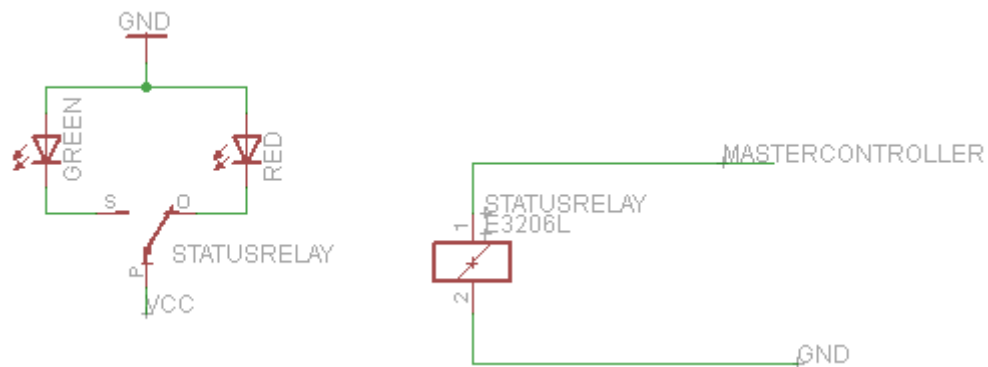


Figure 47: Schematic of the status lights that indicate synchronization between hard and soft panels

The figure above is a schematic of how the status lights to indicate synchronization between the hard and soft panels will be implemented. A relay can be used to help minimize the number of I/Os used on the main MCU. In this scenario, we will assume that the panels are normally not in sync so the status light will remain red. However, if they are, the master controller will send a signal to the relay which will flip the switch from normally closed to normally open to

power the green light. While the actual lights will be panel mounted, the relay will be surface mounted on the main PCB with the master controller.

4.3 Microcontroller

In order to establish a suitable hardware design we researched many similar models, used previous knowledge attained from our studies, and collaborated with the ACTIVE Lab for guidance. All of our research has provided us with enough insight to implement a design that meets our requirements of a hard panel with upwards to 100 controls and communication between the hard panel and soft panel, all while maintaining high throughput of data and high computational power.

4.3.1 Using the ATMEGA Family

As seen from the previous section about microcontroller research, we have chosen to have our hardware design centered on the ATmega series of microcontrollers. We have settled on the ATmega family for a number of reasons:

- The ATmega family offers a variety of I/O as well as built-in devices that our design needs. ATmega also is optimized for power consumption.
- There is a large high quality set of free and/or open source development tools. The amount of online programming support will allow us to make the code simple so we can optimize our time to focus on the many processes, number of devices to interface, and software to run.
- The ATmega family operates on the AVR modified 8-bit RISC processor which excels in throughput due to its two stage, single level pipeline design. It also offers on-chip flash memory storage which allows for in-system programming.
- Availability of ATmega devices and components from traditional component vendors (mouser.com and digikey).

The following table describes specifically which microcontrollers we have chosen to implement our design and some key parameters.

	ATmega325	ATmega32	ATmega8
Clock Speed	16 MHz	16 MHz	16 MHz
Pin Number	64-pin package	44-pin package	32-pin package
Memory	32 Kbytes Flash	32 Kbytes Flash	8 Kbytes Flash
Communication Interface	SPI, TWI (I2C), UART	SPI, TWI (I2C), UART	SPI, TWI (I2C), UART

Table 20: Microcontroller Selection Characteristics

When brainstorming on how to have the design implemented we initially sought to have all the hardware onto a single printed circuit board. After much research and deliberation we found that it would have been very difficult to have a single

PCB hold of the components and have decided to split the major subsystems onto their own PCB. Designing the layout as such allows for easier testing of components and component placement.

As previously mentioned, the major subsystems of the hardware will be placed onto their own printed circuit board. The major subsystems that are to be implemented is given in the following:

- Master MCU
 - Power supply
 - Control the push buttons and rotary switches
 - Communication interface to soft panel
 - Master/Slave configuration with the other two MCUs
- LED Control (Slave 1)
 - Control the LEDs used for the LED box
 - Shift register configuration for individual LEDs
- Gauge Control
 - Controls stepper motors for gauge control
 - Shift register configuration for individual motors

4.3.2 ATmega325 (Master MCU)

The hardware design is essentially centered around the master microcontroller because it acts as the “brain” of the entire system. The microcontroller executes all the instructions sent from the power plant simulator, which goes through the soft panel, and enables the corresponding control.

4.3.2.1 Master/Slave Configuration

Of the four types of controls on the hard panel, the master microcontroller manages two, of which are the push buttons and rotary switches. In order to execute the instructions for the LED box and gauges, the master microcontroller utilizes the serial peripheral interface (SPI), which is embedded in all the microcontrollers we have chosen to use. The SPI allows the three microcontrollers to transmit/receive data through a master/slave configuration where the ATmega325 will be the master and the other two microcontrollers will act as slaves. Through this interface the instructions will be sent to the master microcontroller and will then be sent to its respective slave to execute the command. The following figure depicts how our master/slave system will operate.

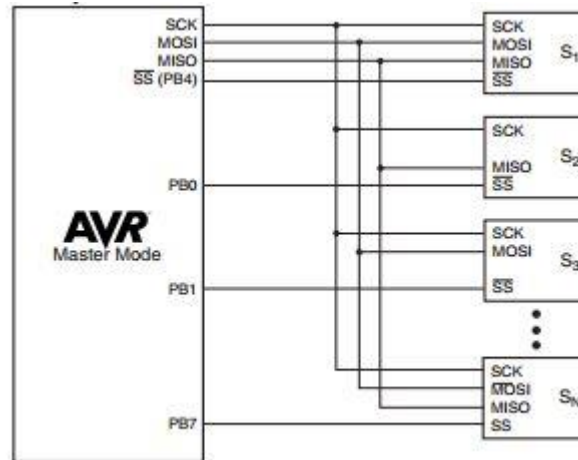


Figure 48: Single Master Multiple Slave Configuration
Reprinted with Permission from Atmel

The interconnection between master and slave using SPI is shown in the following figure. The system consists of two shift registers and a clock generated by the master MCU. The Master initiates the communication cycle by pulling the slave select (SS) pin low of the desired slave. For data to be interchanged between the shift registers, the master generates a clock pulse on the serial clock (SCK) line. The data packets are always shifted through the master out – slave in (MOSI) and master in – slave out (MISO) lines. The master then synchronizes the slave by pulling the SS pin high.

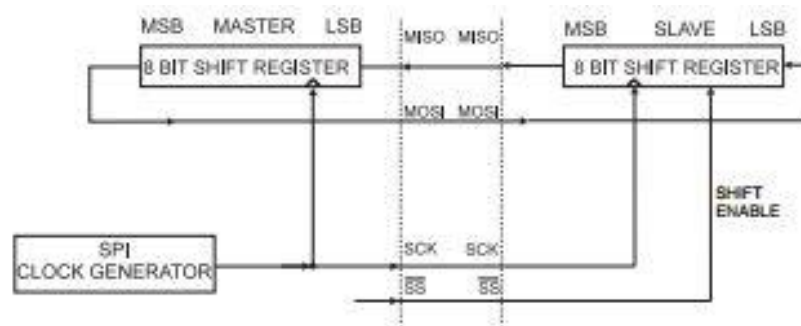


Figure 49: SPI Master/Slave Interconnection
Reprinted with Permission from Atmel

Since the ATmega325 will be configured as the master, the SPI interface will have no automatic control of the slave select (SS) line. Before communication can start we must first configure the software to meet our requirements. The SS will be configured as an input and must be held high to ensure master SPI operation. Once this is done, writing bytes to the registers begins the clock and the bits are shifted into the desired slave. After a single byte has been shifted, the clock halts in order to set a flag (SPIF) that signifies the end of a transmission. If we want to use an interrupt we must enable the SPI interrupt enable bit, SPIE, which is located in the SPCR register. The incoming data will

not be read even though the slave may continue to place new transmitting data into the SPDR. The final incoming byte will be kept inside the buffer register for later use.

The serial peripheral interface system operates under single and double buffered modes. The single buffered mode is set when the system operates in the transmit direction. If the system is double buffered it means it is operating in the receive direction. The following describes what actions are taken under either mode:

- Single buffered
 - The bytes that are ready for transmission cannot be written to the SPI data register until the entire shift cycle is complete
- Double buffered
 - In order to receive data a received character must be read from the SPI data register before the following character has been completely shifted in. Otherwise, the first byte will be lost.

4.3.2.2 Push Button/Rotary Switch Configuration

The master MCU has responsibilities other than communicating between the two slave MCUs. That is, it needs to manage the push buttons and rotary switches that will be implemented onto the hard panel. A goal for this design is to read as many buttons/switches as possible using the least amount of pins on the master MCU.

There are several techniques to read multiple switch inputs using a reduced number of microcontroller pins. The method we choose to implement allows for reading multiple pushbuttons or open/closed switches using only two digital I/O pins, counter/divider, and a timer interrupt from the microcontroller. The following figure shows the schematic used to implement the aforementioned design.

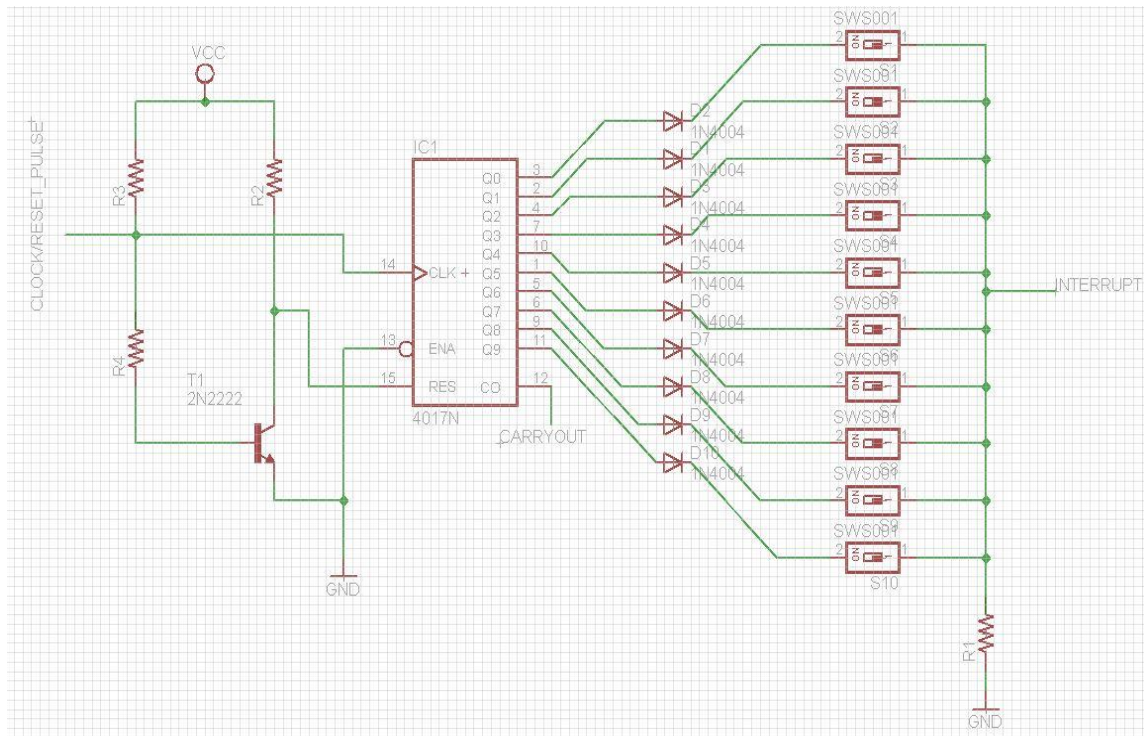


Figure 50: Schematic used to Configure Push buttons/Switches

In the case that two or more switches are closed at the same time, the schematic includes diodes that isolate the counter outputs. The schematic also includes a periodic hardware reset for the counter which drives the count to synchronization. This is important because in the event that a nearby electronic device causes an EMI or ESD, the operation will be reset and normalcy will be continued. This schematic shows that 10 switches are being used. In order to reach our goal of 20 switches and 20 push buttons, we will cascade multiple counters by using the carry-out signal and clock signal.

4.3.3 Power Supply Configuration

The power supply will also be on the same board as the master microcontroller. Our requirement for the power supply is to have the control panel plugged into a wall outlet. The power supply circuit has the purpose of stepping down the voltages in order to drive our low power devices, such as the master microcontroller. The circuit we will be exclusively using can be found in section 3.7.2 Power Supply Research.

4.3.4 PC Communication

An important component of the overall design is the implementation of a soft panel, which will be embedded into a PC. This panel will be configured the same way as the hard panel with the exception that it will be digital. One of the more important features of the PC application is that will be establishing and holding a network connection with the hard panel and transmit/receive data to and from the

master microcontroller. In order to establish this connection between the master microcontroller and PC application we will make use of an RS232 device.

Since the ATmega325 doesn't come with an RS232 dongle we will have to design and implement the necessary components to establish a serial connection with the PC application. Standard RS232 devices operate on voltage levels that oscillate between -15V to +15V. Our microcontroller operates on a range of 0V to 5V; therefore we must make use of a level translator such as the MAX232 IC. All that is left for this design is the connecting cable and a few capacitors.

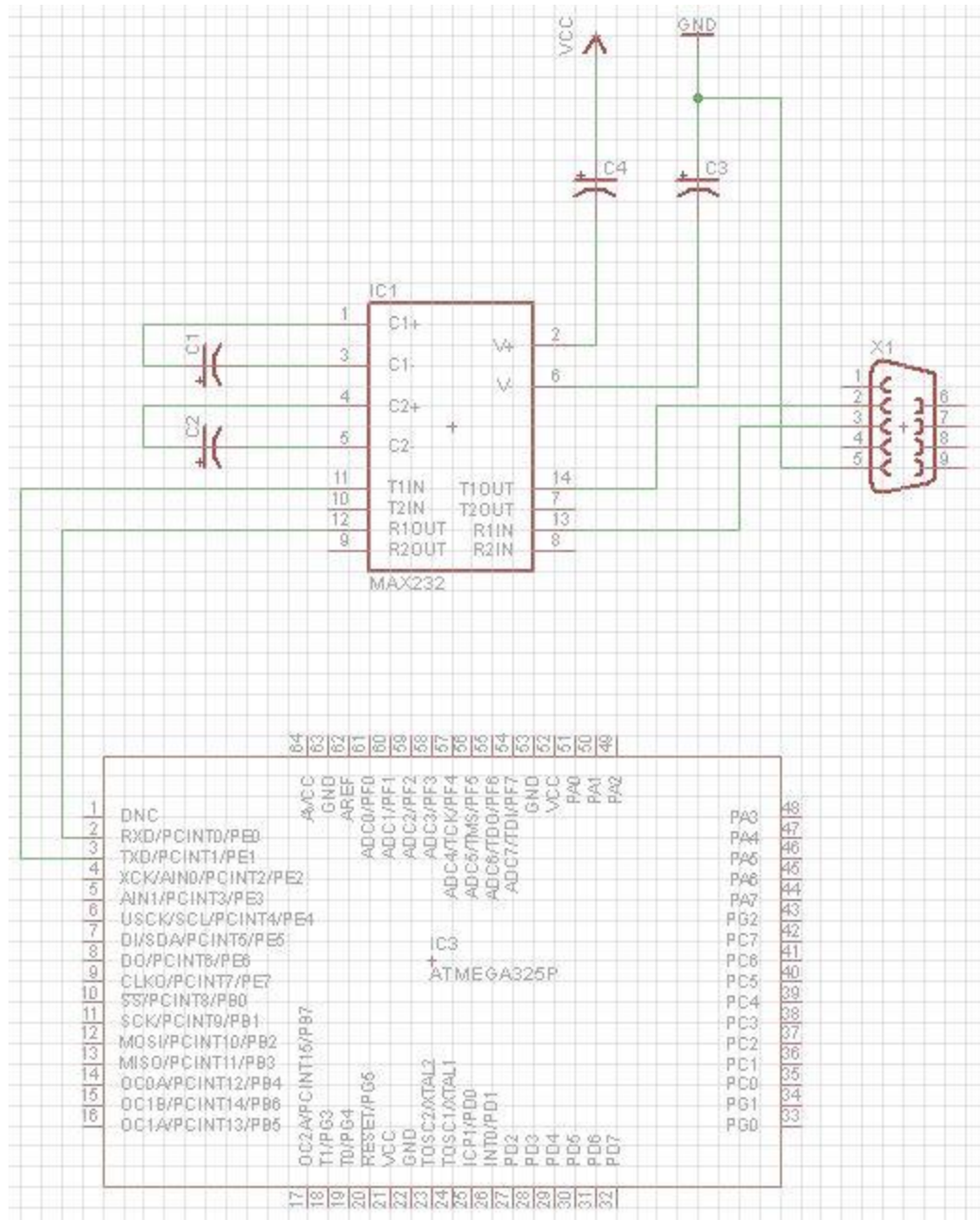


Figure 51: Schematic used to Configure the RS232 Communication

4.3.5 ATmega8 (Slave MCU #1)

The main purpose of this microcontroller is to manage the gauges that will be implemented onto the hard panel. These gauges will be reading states of various components throughout the power plant. In order to construct these gauges we will use stepper motors and use the microcontroller to drive them to the desired value. Since our design requires the use of 24 gauges, we will use shift registers to optimize pin usage. Each shift register is capable of controlling two motors and the ATmega8 is capable of controlling six registers, therefore twelve gauges can be designed in this system.

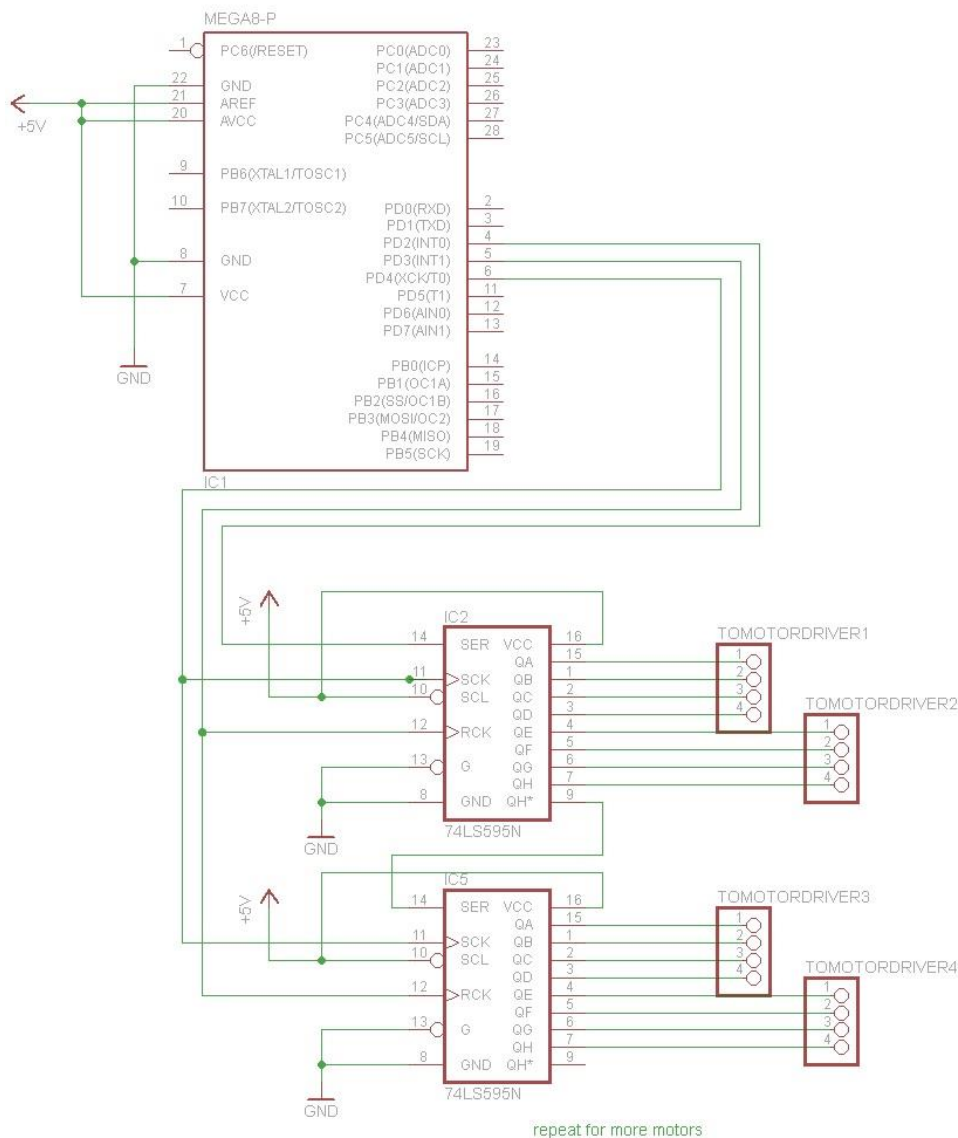


Figure 52: Schematic used to Configure Gauges

The schematic shows twelve gauges being used and in order to implement more gauges we will cascade two more shift registers onto the design.

4.3.5.1 Master/Slave Configuration

A key reason we chose to have all our microcontrollers from the same family is functionality from top to bottom, therefore the master slave interconnection is the same for each microcontroller we selected. The difference with this MCU is that it will be configured as a slave as opposed to a master.

As a slave, the serial peripheral interface will remain sleeping with master in – slave out tri-stated as long as the slave select (SS) pin is driven high. In order to update the contents of the SPI data register, we must configure the software. However, the incoming clock pulses from the serial clock (SCK) pin will not shift out the data, like in the master mode. For the data in the data register to be shifted out we must drive the SS pin low. Once a single byte has been completely shifted out the end of transmission flag (SPIF) is set. If we want to request an interrupt we must set the SPI interrupt enable bit, SPIE, which can be found in the SPCR register. Even though the incoming data will not be read during an interrupt, the slave may continue to place new data to be sent into the SPDR register. The final remaining byte will then be kept in the buffer register for later use.

The serial peripheral interface system operates under single and double buffered modes. The single buffered mode is set when the system operates in the transmit direction. If the system is double buffered it means it is operating in the receive direction. The following describes what actions are taken under either mode:

- Single buffered
 - The bytes that are ready for transmission cannot be written to the SPI data register until the entire shift cycle is complete
- Double buffered

In order to receive data a received character must be read from the SPI data register before the following character has been completely shifted in. Otherwise, the first byte will be lost.

4.3.6 ATmega32 (Slave MCU #2)

The purpose of this microcontroller is to manage the LED box that will be embedded onto the hard panel. The LEDs will be assembled into a 5 by 5 array and the ATmega32 will have the capability of individually addressing each LED. Of the multiple ways to design an individually addressable matrix, we decided to use a master/slave configuration where the ATmega32 is the master and have several slave MCUs that control individual MOSFETs which in turn will either switch the LED on or off. The schematic for this portion of the design can be found in section 4.2.3.

4.3.6.1 Master/Slave Configuration

As previously stated the ATmega32 will act as slave number two of the master microcontroller. Since it is also in the ATmega family the SPI configuration will be the same as the ATmega8. This microcontroller needs a little bit more attention because it will also be acting as a master to an array of MCUs that in turn will enable the addressed LEDs. Because of the LED box design, the ATmega32 will not be using a communication protocol, such as SPI to interface with the slave MCUs. MOSFETs will be used to drive the addressable LEDs.

4.3.7 Programming AVR Microcontrollers

This section covers the method that will be used to get the software that we write onto the microcontrollers. The biggest advantage that we have is the fact that Atmega MCUs are reprogrammable. The software that we write can be programmed to the same MCUs an unlimited number of times.

The two biggest components required to get code onto the chip are a computer and a hardware programmer. When software is ready to be used, the computer will take that code and compile into a form of digital logic that the MCU can use. It will then proceed to tell the programmer how to write that data onto the MCU.

The computer needs to run a compiler in order to do its job. Because AVR processors use Reduced Instruction Set Computing (RISC) architecture and the computers that we will be using run with either x64 or x86 architecture, a cross-compiler is necessary. This type of compiler creates executable code for a platform other than the one on which the compiler is running. This is exactly what we need since the microcontrollers that we are using do not support an operating system. The compiler that we will be using is Atmel® Studio 6 for Windows PCs. This integrated development platform (IDP) provides a very credible environment to build our applications, supporting both C and Assembly languages.

The second of the two main development components, the hardware programmer, needs software in order to run. We will be choosing the command line program, AVRDUDE. On its main website, AVRDUDE is described as “a utility to download/upload/manipulate the ROM and EEPROM contents of AVR microcontrollers using the in-system programming technique (ISP)” (<http://www.nongnu.org/avrdude/>).

To summarize the above information, the two software components used to program our Atmegas are Atmel® Studio 6 and AVRDUDE. Now what is left to be determined is the necessary hardware needed to complete our objective.

Every microcontroller needs a programmer in order to program it. After researching the many programmer options available, we have decided to use an Arduino Uno to program our AVR microcontrollers. This option supports in-system programming while designing our circuit. This means that the microcontrollers will be able to be reprogrammed while remaining in the circuit, thus saving time and reducing the risk of error. Arduino offers firmware (ArduinoISP) that supports this. This support includes tutorials and provides code to burn a bootloader onto an AVR. With this, we can ensure that all microcontrollers will be successfully programmed with little to no risk.

When performing in-system programming, six connections are required to connect a microcontroller to the programmer. These connections are Voltage, Ground, Master In Slave Out, Master Out Slave In, Reset, and Slave Clock. It is important that only one voltage source is used while programming. All other voltage sources should be disconnected from the circuit during this time.

Because we are using an Arduino Uno as a programmer, the Arduino IDE can be used to make the process very easy. It provides software that is flashed to the chip, preparing the Arduino to perform its job as a programmer. Once this is done, AVRDUDE is used to program the AVR chip with the software that we write. This process can be used as many times as we desire.

4.4 Communication Between Components and Devices

It is safe to say that because of the high amount of controls used in this project, there will be a high amount of communication throughout the system. Data transfer will occur within the hard panel, soft panel, between the panels, and between the soft panel and power plant simulator. The system must be capable of handling a heavy amount of data transfer to support the operator's needs. This means that data must be sent quickly and reliably. In order to implement these factors, we have designed our project to use one type of data object through the system. This means that all components will work in a similar way, reducing work load and unnecessary conversions. The data object that will be used over the system is a simple string, formed with one universal paradigm, called the Simple Message paradigm.

The Simple Message is a string with a predefined structure which ensures easy transmission and parsing. The basic format for the message contains several units of information with the following fields separated using the pipe character '|':

```
UUID_FOR_APP | SOURCE | DESTINATION_NAME | MESSAGE_TYPE |  
DATA
```

UUID_FOR_APP:

UUID (also commonly referred to as a UUID) stands for Universally Unique Identifier. A UUID is a 128 bit number that is used as a unique value. This will be generated in Java and used to specify which application is sending the messages. This field is required because of the possible use of multiple stations across the network.

ID_FOR_DATA_SOURCE:

This is used to identify the source of the data and to distinguish between sources that may exist on multiple computers. For example, if a “WATER_TEMP” gauge exists on multiple computers or displays, which is likely, ID_FOR_DATA_SOURCE is used to determine which instance of WATER_TEMP generated the message.

SOURCE_NAME:

The source name will be a string of the name of the data source (control component) that sent the message. It also could be used to command multiple data sources to carry out an action. An example of a source name is “WATER TEMP”, which is a gauge.

DESTINATION:

The destination is also a string which will hold the name of who should process the message. The two possible names are “ALL” and “MANAGER”. When the key value, “ALL” is used, everyone receives the message and parses it on their side to determine if it is for them. The key value “MANAGER” means that the message is only intended for the manager application, which is the power plant simulator itself.

MESSAGE_TYPE:

This is the type of message that is being sent. Two possible values include “SET” and “REPORT”. The message type SET tells the receiver to change its state to a new value, which will also be specified. A REPORT message is intended for logging a participant’s actions.

DATA:

Data will contain a list of key/value paired fields, separated using a pipe delimiter “|”. The key must come before the value, separated by a comma. However, pairs can occur in any order. All Message Keys that will be used are described in the following tables.

The following table shows the three globally used message keys. This type of message is used to handle all of a specific type of component. The tables that follow the initial description of message keys all describe different subsystem message key formats.

Constant name	text	Type of values
KEY_NAME	"widget_name"	String name of specific widget.
KEY_TYPE	"widget_type"	Defined as enum, currently either PANEL, GAUGE, VALVE, IMAGE, LIGHTPANEL, UNKNOWN
KEY_CLICKED	"widget_clicked"	Only ever "true" if in message, indicates widget was clicked

Table 21: Message Keys

Constant name	text	Type of values
KEY_TYPE_NAME	"Gauge"	Predefined string for widget type
KEY_VALUE	"gauge_value"	Percentage value, where indicator arrow is on gauge
KEY_RATE	"gauge_rate"	Integer value, rate in milliseconds to move arrow on gauge
KEY_ACK	"gauge_ack"	Only ever "true" in message, indicates acknowledgment button on gauge was clicked

Table 22: Gauge Specific Message Keys

Constant name	text	Type of values
KEY_TYPE_NAME	"Valve"	Predefined string for widget type
KEY_STATE	"valve_state"	Internal state of valve, usually represented by the light on the body of the valve. Defined as enum, currently either BLANK, GREEN, RED, YELLOW, GREENRED
KEY_HANDLE_POSITION	"position"	Position of handle based on predefined states and their associated positions. Usually, there is a CENTER position, a LEFT position, and a RIGHT position
KEY_HANDLE_angle	"handle_angle"	Angle in degrees valve handle is rotated to

Table 23: Valve Specific Message Keys

Constant name	text	Type of values
KEY_TYPE_NAME	"LightBox"	Predefined string for widget type
KEY_ROW	"lightpanel_row"	Integer row to flash. Out of bounds if -1
KEY_COLUMN	"lightpanel_column"	Integer column to flash. Out of bounds if -1
KEY_FLASH	"lightpanel_flash"	Usually "true" in message, indicates lights should flash on light box. May be "all", indicating all lights on specified light box should flash
KEY_FLASH_RATE	"lightpanel_rate"	Rate in hertz for specified light to flash
KEY_FLASH_ALL	"lightpanel_flash_all"	Only ever "true" in message, same as KEY_FLASH with "all" value
KEY_CELL_COUNT	"lightpanel_cell_count"	Integer number of cells specified in current message
KEY_CELL	"lightpanel_cell_"	In a message, light cells are indicated by a key value with a zero-based number system (i.e. lightpanel_cell_0, lightpanel_cell_1, etc.) The values for these keys is an underscore separated pair of integers representing zero-based row and column of particular cell (i.e. 1_2 indicates row 2, column 3)
KEY_COLOR	"lightpanel_color"	Hexadecimal representation of specific color for cell(s) to flash
KEY_CLEAR	"clear"	Usually "true" in message, indicates lights should stop flashing on lightbox. May be "all", indicating all lights on specified light box should stop
KEY_TEST_LIGHT_RIGHT	"lightpanel_test_light_right"	Boolean value to either turn on or off the right test lamp
KEY_TEST_LIGHT_LEFT	"lightpanel_test_light_left"	Boolean value to either turn on or off the left test lamp
KEY_TOP_ONLY	"lightpanel_top_only"	Only ever "true" in message, indicates whether only the top portion of a particular cell should flash
KEY_BOTTOM_ONLY	"lightpanel_bottom_only"	Only ever "true" in message, indicates whether only the bottom portion of a particular cell should flash

Table 24: LED Specific Message Keys

Constant name	text	Type of values
KEY_VISIBILITY	"panel_visibility"	Boolean value to make the specified panel either visible or not
KEY_CONDITION	"panel_condition"	String name of condition to run in panel

Table 25: Panel Specific Message Keys

That concludes the basic messaging format. To restate, all strings will contain the following fields: UUID for app, ID for data source, source name, destination name, message type, and the data itself. Some fields won't be required for every transmission. For example, when data is being transferred within the hard panel, the UUID_FOR_APP field is irrelevant. Another example is that output control components will only receive SET messages and never send them. Although this is the case, the same paradigm is used for all information flow. This keeps data transmission simple, quick, and reliable.

4.5 Printed Circuit Board

When we were first designing the layout of the entire hardware system we stumbled onto multiple complications. Some issues included component placement, keeping analog signals separate from digital signals, and the appropriate use of power, ground, and signal planes. We also wanted to save enough board space to run test on the different subsystems of our design. After much research and deliberation we agreed to have each subsystem placed on their own respecting PCB. Three boards will be designed; one will support the master microcontroller, gauges subsystem, and LED subsystem. Allowing each subsystem to have their own PCB cuts down on the previously mentioned complications and makes testing each subsystem user friendly. The following figure shows the PCB design of the master microcontroller.

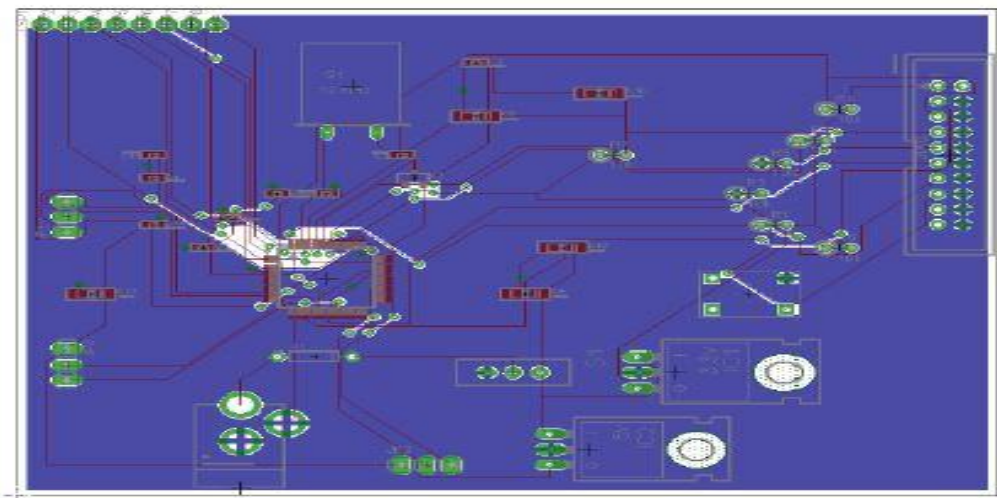


Figure 53: PCB design of the Master MCU

The schematics of each subsystem will be designed using EAGLE design software and will be manufactured to hold the main components of the hard panel. In the coming weeks we will submit the finalized designs to the possible manufactures in order to receive a proper estimate. The printed circuit boards are of high importance and the order will be quickly made so that we can begin putting the boards together.

4.6 Power Supply

With the use of TI's database of designed and tested power supplies we have decided to base our power supply off of one of theirs. Using our design requirements as guidelines and the research documented earlier in the paper it was found that the flyback buck-boost converter numbered PMP5515 would best suit our needs. The schematic provided by TI is given on the next page. The PMP5515 takes a wide range of inputs from 85-265VAC and steps down the voltage to four separate direct current values. These values are 24.4V, 12V, 5V, and 3.3V. Since we only require two continuous outputs, the circuits for 24.4VDC and 12VDC will be omitted. A new transformer will need to be acquired as well because the one documented is no longer manufactured. Therefore, even if we were to implement the exact design given by TI we would still need to find a new transformer. There are companies that exist that will make custom transformers based off of the customers design requirements. One of which is XFMRs with a sales representative located in Oviedo. For our system we will need a flyback transformer with dual parallel input/output that steps down a DC input voltage from the range of 85-265VDC to 3.3VDC. The power stage designer tool by TI recommends a turn ratio of 31.88:1 with a calculated inductance of 31737.45 μ H. However, better inductance can be achieved since the transformer TI used was rated with inductance of 570 μ H.

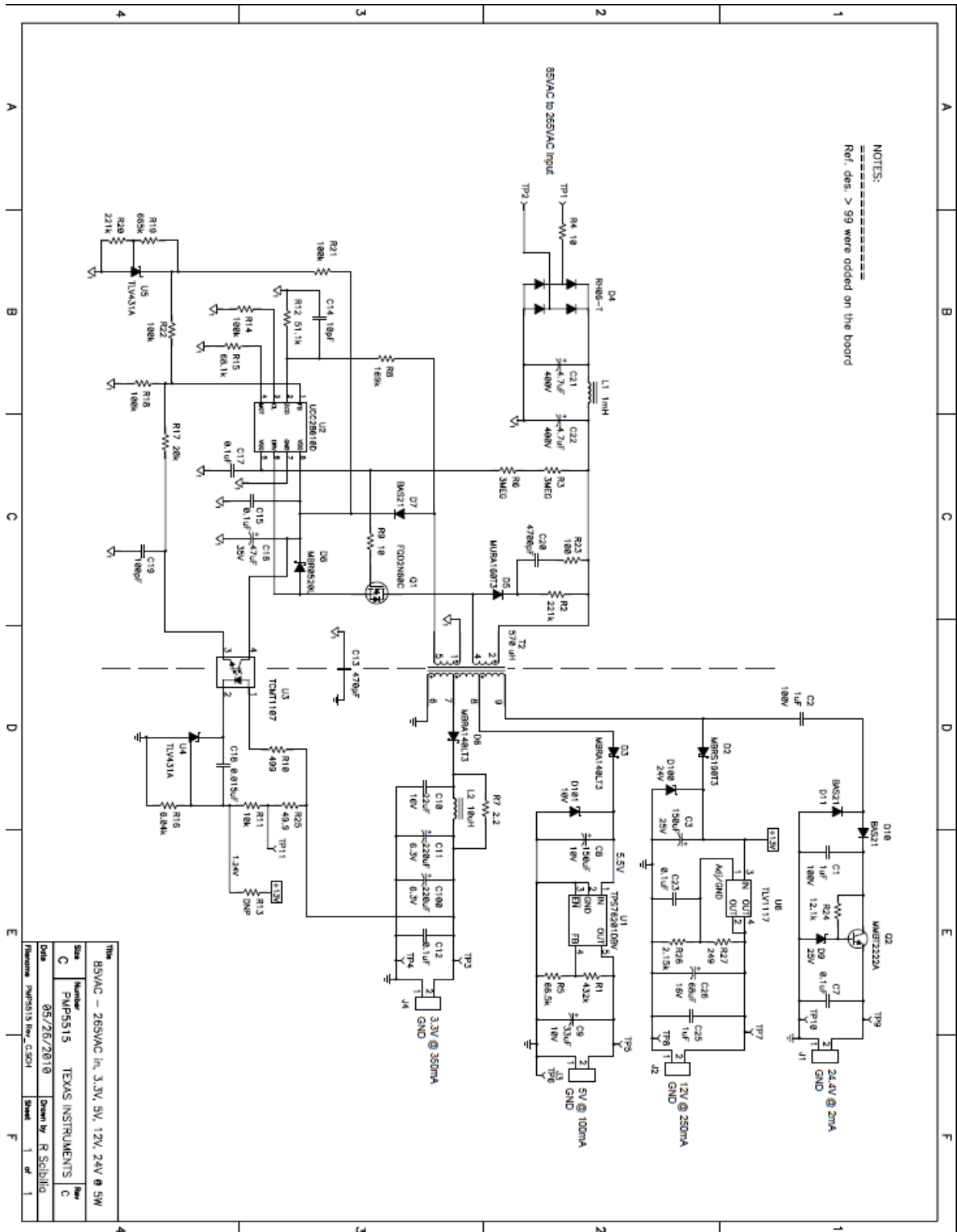


Figure 54: Power schematic reprinted with permission from Texas Instruments

Furthermore, even though the schematic is offered for free and includes a complete list of the bill of materials and full documentation of the testing results,

no alterations to the original design can be documented. That is why the schematic is shown in its entirety and without the 24.4VDC and 12VDC circuits being removed.

4.6.1 Power Breakdown

The maximum and minimum estimated power usage for each subsystem is given in the table below. The maximum estimates were made assuming that every light is on, as well as the pushbuttons and all of the microcontrollers are in active mode. While the minimum estimates were made with the assumption that the microcontrollers are in idle mode, none of the lights or pushbuttons are on. In our calculations we assumed all components operated ideally, we realize however this will not be the actual case and that our estimates will most likely change after the first prototype has been built. According to TI, the PMP5515 is capable of supplying 5W of power.

Subsystem	Power Max (W)	Power Min (W)
Pushbuttons	0.242	0
Light Box	2.505784	0.00105108
Gauge	0.0696	0.033168
Indicator Lights	1.5	1.5
In-sync Lights	0.06	0.06
Master Controller	0.00063	0.000036
Total:	4.378014	1.59425508

Table 26: Power breakdown for hard panel system

As one can see from the table above, our minimum expected energy usage is very low while our maximum falls well within our 5W limit. The largest expected consumption of energy will be the light box which was anticipated.

The graph provided below by Texas Instruments gives the efficiency of the PMP5515C design over a range of applied voltages and loads. From looking at the graph it seems that the larger the load is the lower the load current and the more efficient the circuit becomes. Once built we will have to perform our own efficiency tests by applying a constant input voltage, and varying the load resistance with a resistor box. The output voltage, current and input current will need to be measured with a multimeter and the values can be used in the following equations to determine the power efficiency.

$$P_{in} = V_{in} * I_{in} \qquad P_{out} = V_{out} * I_{out}$$

$$\text{Efficiency} = (P_{out} / P_{in}) * 100$$

Hopefully we will be able to obtain the same levels of efficiency if not higher.

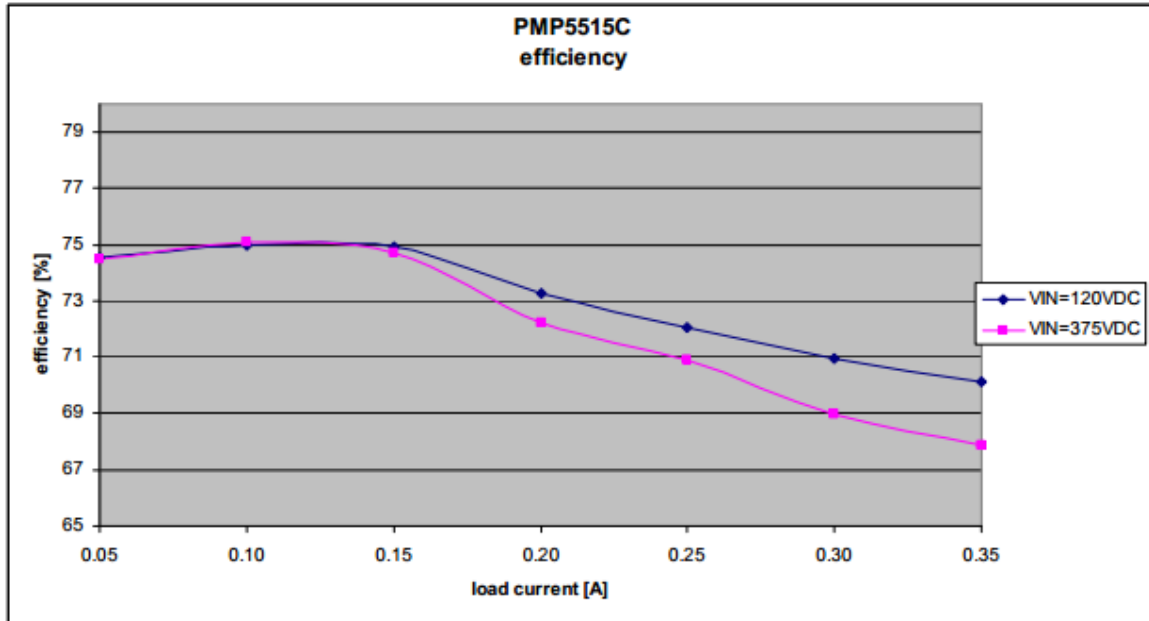


Figure 55: Power efficiency of transforming circuit provided by Texas Instruments

4.7 Software

4.7.1 Soft Panel Software

4.7.1.1 Application

The PC application can be considered as the center of this project, as it interacts with all sections of the design. Most importantly, it acts as the digital control system that the nuclear power plant industry is working on transitioning to. Containing the motivation of this entire project, the application has four main responsibilities:

1. Establishing and maintaining a network connection with both the hard panel and power plant simulator
2. Sending and receiving data to and from the analog control panel's microcontroller
3. Sending and receiving data to and from the power plant simulator
4. Displaying the entire digital control panel, which includes all components and their present status
5. Providing user interaction with the digital control panel components

This section will cover the software design approach to carrying out these responsibilities.

The initial responsibility of the application is to establish a connection with all components of the project. These components include the analog control panel and the power plant simulator. All three devices will exist and communicate

within a local area network. This networking procedure is explained in Section 3.6. Simply put, at run time the application will need to become accessible. Being accessible means that it is ready and available to send and receive data. There are two steps in becoming accessible.

The first step is to connect and bind itself to a UDP port on the computer it is running on. This is how data that enters the computer gets to the application and is also where the data leaves the application when it is sent out. In order to choose which port to bind to, the application will refer to a settings file. This will be an XML file that essentially chooses ports based upon the computer's current IP address.

The second requirement the application needs to follow for proper communication is choosing which multicast IP it should listen to. This will be done dynamically, as there are many possible addresses. The address of choice will depend on the IP address of the computer it is running on. In order to implement this procedure, a lookup table will be used. This table will exist in the XML settings file. For example, it could say that if the computer's IP address is 192.168.0.15, then listen to multicast IP address 230.100.1.1. Using this method, the same settings file could be used across all computers within the local area network. In addition to providing a successful means of communication, the settings file also enables all applications to identify themselves within the network. This is useful when multiple stations exist and it is necessary to differentiate between the group.

It is important that every part of this project is in sync at all times. The analog control panel and the digital control panel are in sync when each set of components are in the exact same state. The PC application has a big role in making this synchronization possible. The components can fall out of sync in two situations: When the user changes component's states on the analog control panel and when the user changes the component's states on the digital control panel. The application has a job to do in both situations.

Situation number one involves the user making a change to some component on the analog control board. Possible changes include pushing a button or flipping a switch. When one of these scenarios occurs, the control board's microcontroller will send new state information to the application in the form of a string. The string contains information pertaining to which component was changed and what value it changed to. The application will take this string, update the graphical user interface to display the correct value, and then send the new information to the power plant simulator. The power plant responds by telling the application to do certain things. The two possible commands are SET and REPORT. A Set command tells the application to change specific components to specific values. A Report command requests the application to return more specific information to the simulator. This SET/REPORT loop can go

on for however long the power plant simulator requires it to. A UML sequence diagram of this entire procedure is shown.

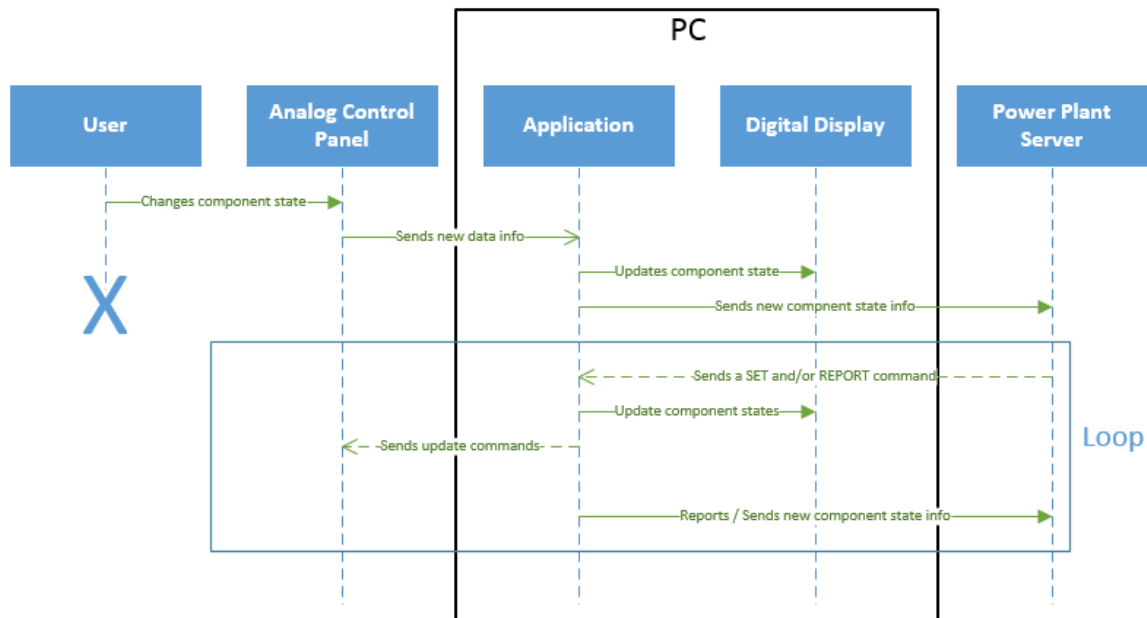


Figure 56: Sequence Diagram of User Interaction with Hard Panel

Situation number two involves the user interacting directly with the digital display. This event is handled in a very similar way, just in a different manner. The application is constantly listening to user interaction with the digital display. When it detects a new event, it updates the display and sends the new information to the power plant. This data is constructed by concatenating a string with several different pieces of information. This was covered in greater detail within section 4.4

At this point, new data has been prepared and sent to the simulator. While waiting for a response from the simulator, the application must tell the analog control panel that something has changed on the digital display. This is necessary in order to keep the analog and digital boards states synchronized. Obviously, the application itself cannot change analog states, so it must resort to telling the user to do it themselves. It is then the user's responsibility to change the necessary analog components; otherwise the system is not in sync.

When the power plant simulator responds with feedback concerning the digital display's change of state, the application will handle this in the exact same way as in the first situation. This will continue for as long as necessary. A sequence diagram of this entire scenario is shown below.

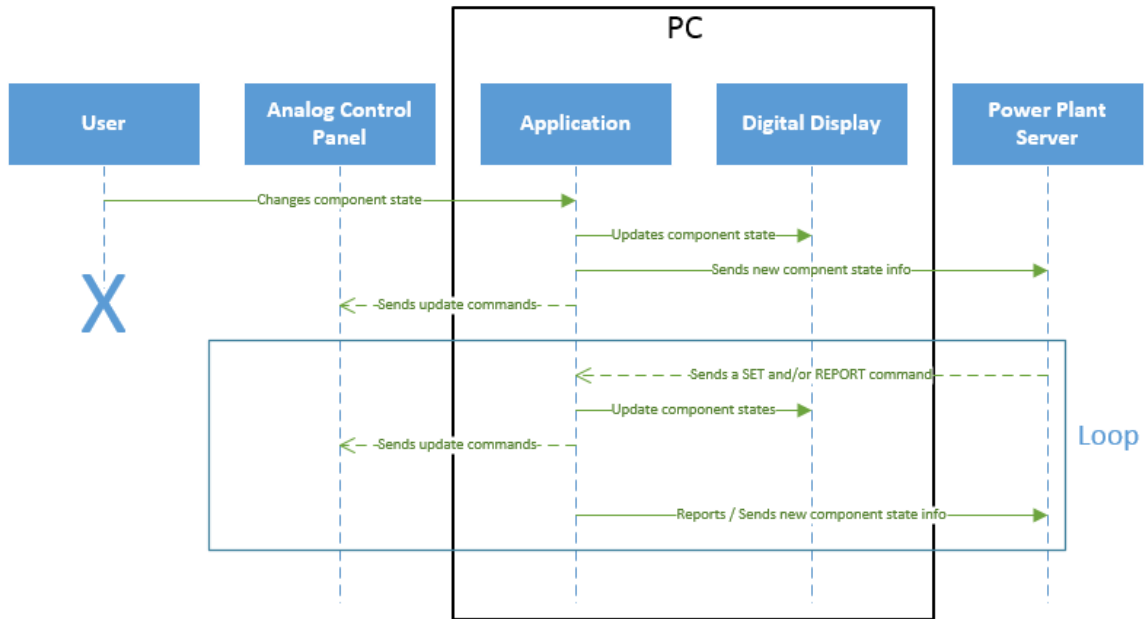


Figure 57: Sequence Diagram of User Interaction with Soft Panel

4.7.1.2 Graphical User Interface

The application is responsible for displaying the entire soft panel, which includes approximately 100 different components. It must also provide full user interaction with these components. It will do this with the use of a graphical user interface (GUI). Control panel components will be represented through graphical icons. There will be four main types of icons used which will represent the gauge, switches, buttons, and light boxes. The following figure gives a basic idea of what these icons will look like.

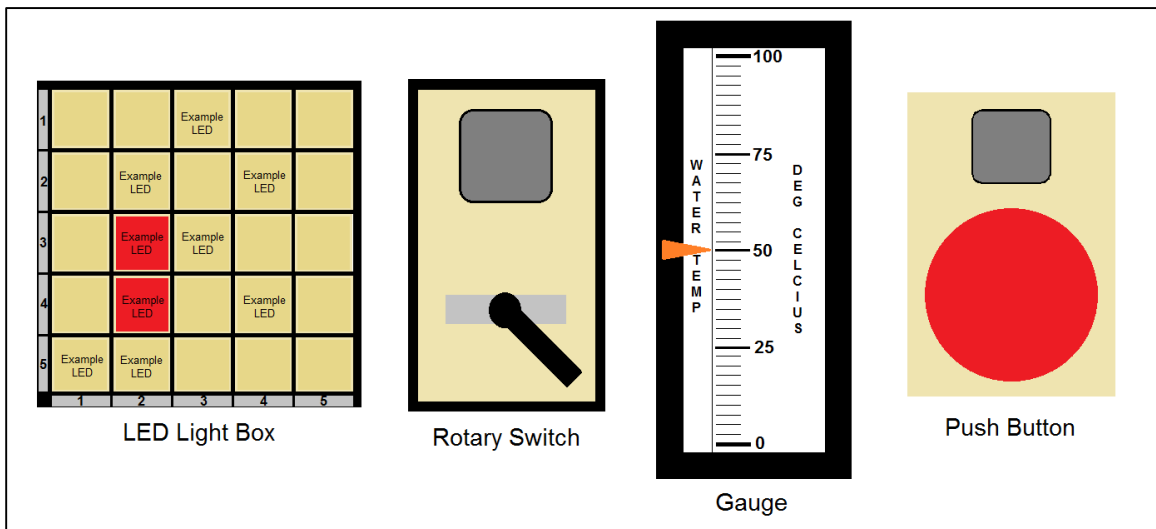


Figure 58: Tentative device icons to be used in the GUI

The project design does not rely heavily on the choice of icons, so they are subject to change.

When any component changes in state, the application will immediately update the icon representing that component. Updates include:

- Turning an LED on or off
- Flashing an LED
- Flipping a rotary switch to new position
- Moving a gauge's dial to a new position
- Setting a button to on or off

These actions will be performed with simple animations, so that the user may watch component state changes. This provides a greater user friendly experience.

The application needs to provide a method of user input for changing state of the rotary switches and push buttons. This is will done by implementing a listener for mouse clicks on the GUI. To push a button or flip a switch, the user will simply click the relevant icon with their mouse. This triggers the necessary animation and then changes the component's state.

Java Swing will be used to implement all of the GUI requirements described above. Swing is a GUI widget toolkit provided by Oracle's Java Foundation Classes (JFC). We have chosen this API because of the fact that the entire application will be developed in Java, thus making Swing fully compatible with the program.

4.7.2 Power Plant Simulator Software

The power plant simulator will exist on its own computer in the system. It will be a simple program constantly running a basic input output procedure. This procedure can be described in three basic steps:

1. Listen for incoming data from existing stations
2. Handle newly received data
3. Respond to stations with commands

The first step will be implemented using the same method in which the soft panel listens for commands. The simulator will bind itself to a UDP port on the computer it's running on, because it can assume that all data traffic flows with UDP transmission. Within the program there will be a function running that always listens for new data. As soon as data is received, this function calls the data handler segment of the program.

When receiving new data, the data handler expects it to be in the form of a string. This string contains several fields that are separated with the pipe "|" character.

After parsing the string, the data handler will have determined multiple pieces of information. This includes the following:

- Which station (application) sent the message
- The exact component the message is regarding
- The type of component the message is regarding
- Data providing a component's current state

Once this data has been determined, the handler then decides how to respond to it. It may request more information, command the control panel to change various component states, or do both. The question to answer now is how these choices of commands will be made. Our project focus is not to simulate the actions of a power plant; it is to develop effective transition methods from analog controls to digital controls. For this reason, we are able to program the simulator to decide upon various commands in a random manner. The REPORT and SET commands that will be generated will vary in no specific way. The algorithm used to create these messages will essentially contain many if-else statements, and several random functions. Different outcomes will occur depending on which station sent the message, which component changed, and current control states.

When a command is ready to be sent out, the program puts it in the form of a datagram. This is a necessary step for UDP data transmission. Oracle provides classes that can be used to complete this requirement, which are provided in the *java.net* package. The main class that will be used in this package is the *DatagramPacket* class. Once generated, a command is sent out to all existing stations. This is done with UDP multicasting. By using a universally used settings file, all stations listen to the same multicast IP. When the power plant releases new data, all stations receive it. The data that the simulator has constructed specifies which control panels it pertains to.

This is all the Power Plant Simulator program is required to do. It has the simple task of receiving new information, handling it in a random way, and returning commands for the control panel to follow.

4.7.3 Hard Panel Software

It is important that the hard panel functions exactly as it is supposed to. It should react properly to all user input, soft panel input, and simulator input. The hard panel and the soft panel are essentially the same station, just built in different forms. Therefore it is important that these two panels are in sync with one another. The hard panel is comprised of several main components all working together to get the job done. Each component has its own sets of responsibilities, which are summarized here.

- Main Microcontroller (Master)
 - Maintain every control and its state within the hard panel

- Receives information from the soft panel pertaining to:
 - Commands from the power plant simulator
 - User input that has occurred on the soft panel
- Receives new information from slave MCUs about user input that has occurred on the hard panel
- Commands slave MCUs to update their components' states
- Gauge Microcontroller (Slave)
 - Receives information from master MCU pertaining to:
 - Changing a specific gauge or set of gauges to new state
- LED Microcontroller (Slave)
 - Receives information from master MCU pertaining to:
 - Changing a specific LED or set of LEDs to new state

We feel that the best way to cover software design of the hard panel is to describe it's lifecycle through a list of operational scenarios. This list will begin with the Master MCU's responsibility and then cover the responsibilities of the Gauge MCU and LED MCU. In the case of complex scenarios, sequence diagrams will be used for summarization purposes.

Throughout these scenarios, a few key terms will be used and are defined here:

MMCU: Master microcontroller

LMCU: Slave LED microcontroller

GMCU: Slave Gauge microcontroller

Hard input: The input controls located on the hard panel. These include push buttons and rotary switches.

Soft input: Similar to hard input, but located on the soft panel.

4.7.3.1 Master MCU Operational Scenarios

Scenario 1, Hard panel powers on:

The most important job of the MMCU is to maintain every control and its state within the hard panel. When the hard panel powers off, it will be programmed to lose all control state information. This is because input controls can change during the time the panel is powered off, so it is not necessary to save this information.

When powered on, the MMCU establishes a connection with the soft panel over the RS232 standard. Without this connection, the MMCU will not know what it needs to set its output values to. The MMCU establishes a connection by sending the soft panel a message that the hard panel is powered on and ready to work. The soft panel responds by sending the MMCU a complete string of data containing every current control state.

Next, the MMCU determines the state of all hard panel input controls. This includes the push buttons and rotary switches, which are all located on the MMCU's ports. Once this information has been determined, the MMCU checks if the hard input matches the soft input.

If input does not match, the MMCU will flash its 'Not In Sync' LED and send a message to the soft panel. This message tells the soft panel that input does not match, and also contains all hard input. The soft panel will then display a list of instructions for the user to get in sync. The MMCU will not continue operation until it has received an in sync message from the soft panel. During this time, any change in input is exchanged between both panels.

When the hard input matches the soft input, the MMCU receives the current output data from the soft panel. It then commands the two slave MCUs to set to their components to their respective output values. This concludes the power on stage, which can be summarized in the following sequence diagram.

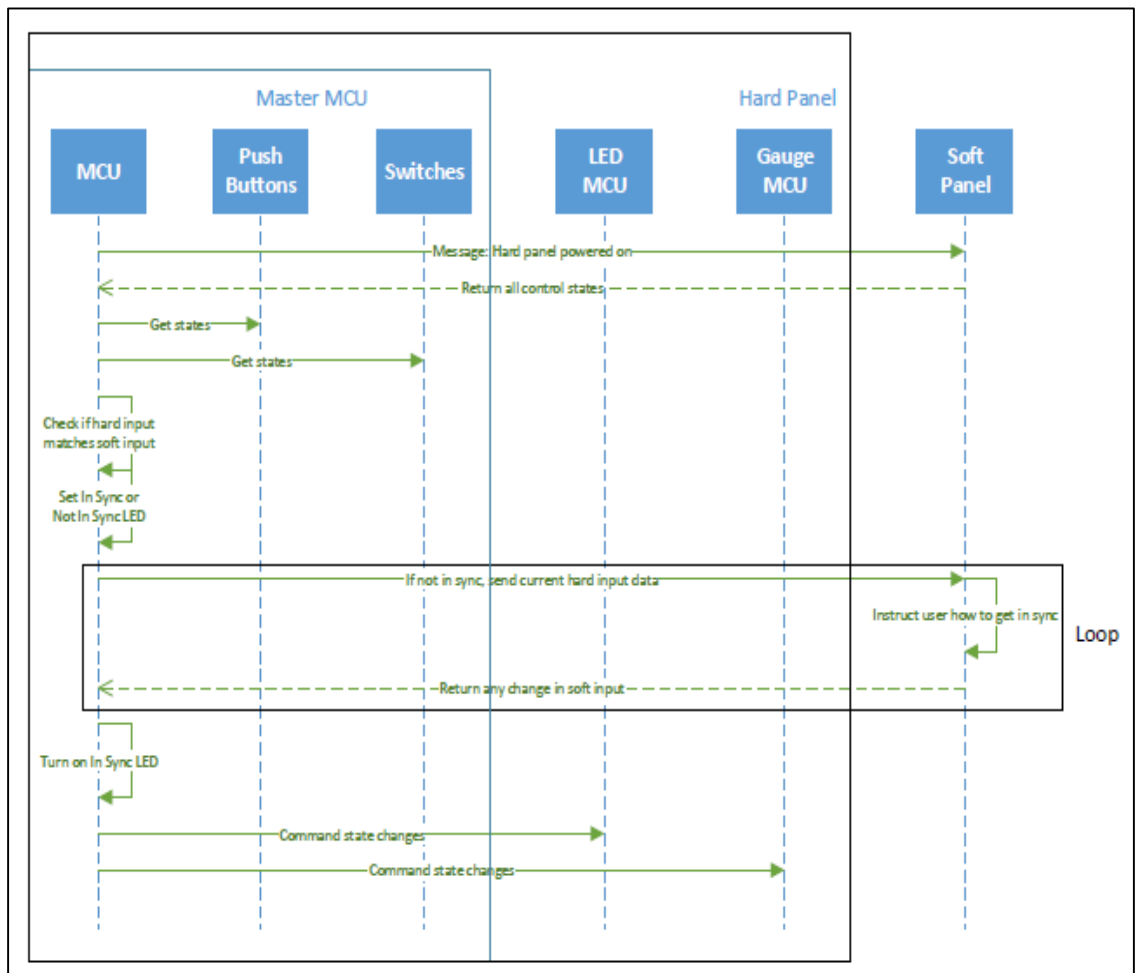


Figure 59: Hard panel powers on

Scenario 2, MMCU receives data from soft panel:

The MMCU accepts any incoming data from the soft panel. Data may consist of change in any controls or set of controls. The data specifies what needs to be changed and to what value it needs to change to. In the case that hard output needs to be updated, the MMCU passes along the message to either the LMCU or GMCU, whichever is appropriate. If the data pertains to the hard input, this indicates that something needs to be manually changed by the user to maintain control panel synchronization. The MMCU turns on the 'Not In Sync' LED and sends any change in hard input back to the soft panel until synchronization is complete. The sequence diagram below summarizes this scenario.

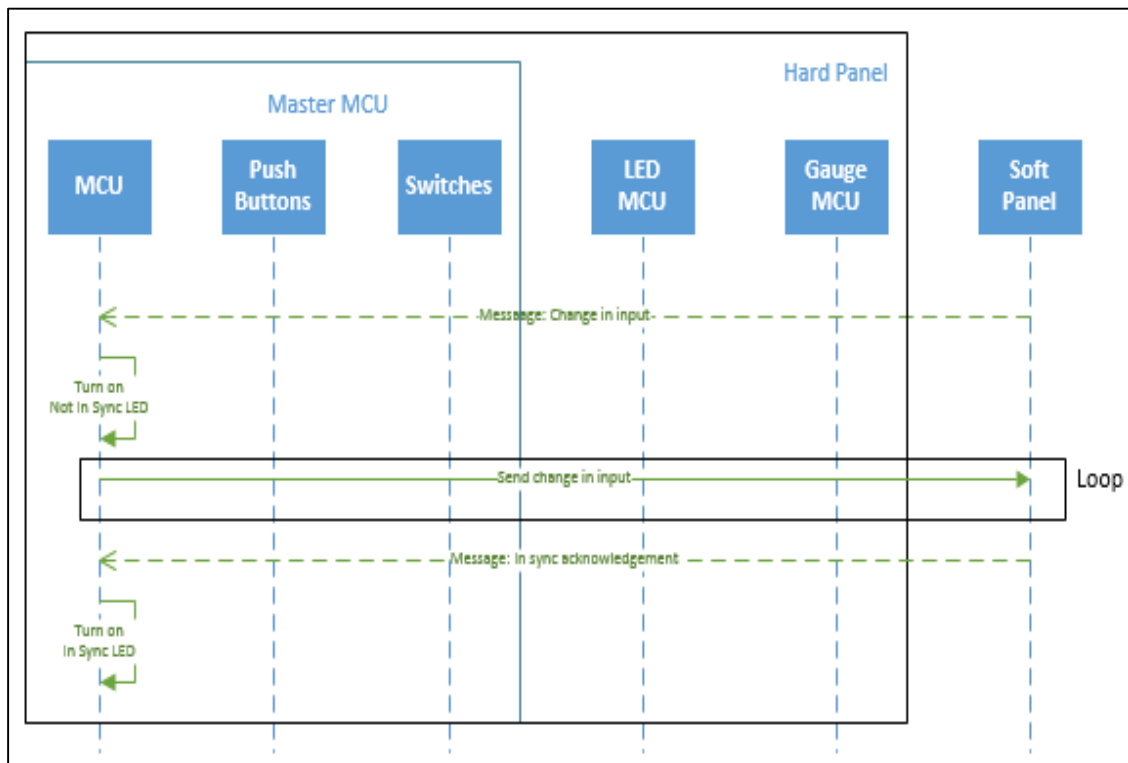


Figure 60: MMCU receives data from soft panel

Scenario 3, User changes state of hard input.

Because the push buttons and switches are located on the ports of the MMCU, the microcontroller immediately recognizes when all hard input changes. Whenever this occurs, it prepares data to be sent to the soft panel. This data tells the soft panel to update the control that changed input. Once the data is sent this scenario is complete.

4.7.3.2 LED MCU Operational Scenarios

Scenario 1, Hard panel powers on

At initial power on, the LMCU sets all of its LED components to the off state. From this point on, it waits for commands from the MMCU that specify which LEDs need to turn on, which need to flash, and which need to turn off.

Scenario 2, New data is received

The MMCU sends commands to the LMCU when LED output changes. These commands specify which LED or set of LEDs need to change, and what change needs to be made. LED changes include turning on, turning off, and flashing.

4.7.3.3 Gauge MCU Operational Scenarios

Scenario 1, Hard panel powers on

At initial power on, the GMCU does nothing. All stepper motors being driven by the GMCU remain in the same degree of rotation they were last set to.

Scenario 2, New data is received

The MMCU sends commands to the GMCU in a similar manner as it does with the LMCU, although slightly more complex. The data that the GMCU receives specifies how many steps a certain stepper motor need to make, and in what direction it needs to make it in.

4.7.3.4 Scenario Conclusion

The scenarios explained in this section have been designed to work well together, ensuring that the hard panel has complete functionality. One thing to mention is that the data that is being passed around during hard panel operation is always in the form of strings. It follows the same data structure used throughout the entire system. This data structure is covered in detail in Section 8.

5.0 Design Summary of Hardware and Software

After reading through the previous section it is easy to see that our group worked as hard as possible to get a thorough design hashed out for our paper. We were granted essentially free reign from our mentors within the ACTIVE group for our design and for the most part had to come up with requirements and specifications on our own. This allowed us the ability to use our own judgment and to gain experience in designing a product.

Below you can see our system diagram, depicting how all of our systems and subsystems will interact. Hopefully this diagram brings our design full circle and

allows the reader to visualize how everything connects. Our main aim with this diagram is to establish an understanding of the layout of our panels and the MCU's.

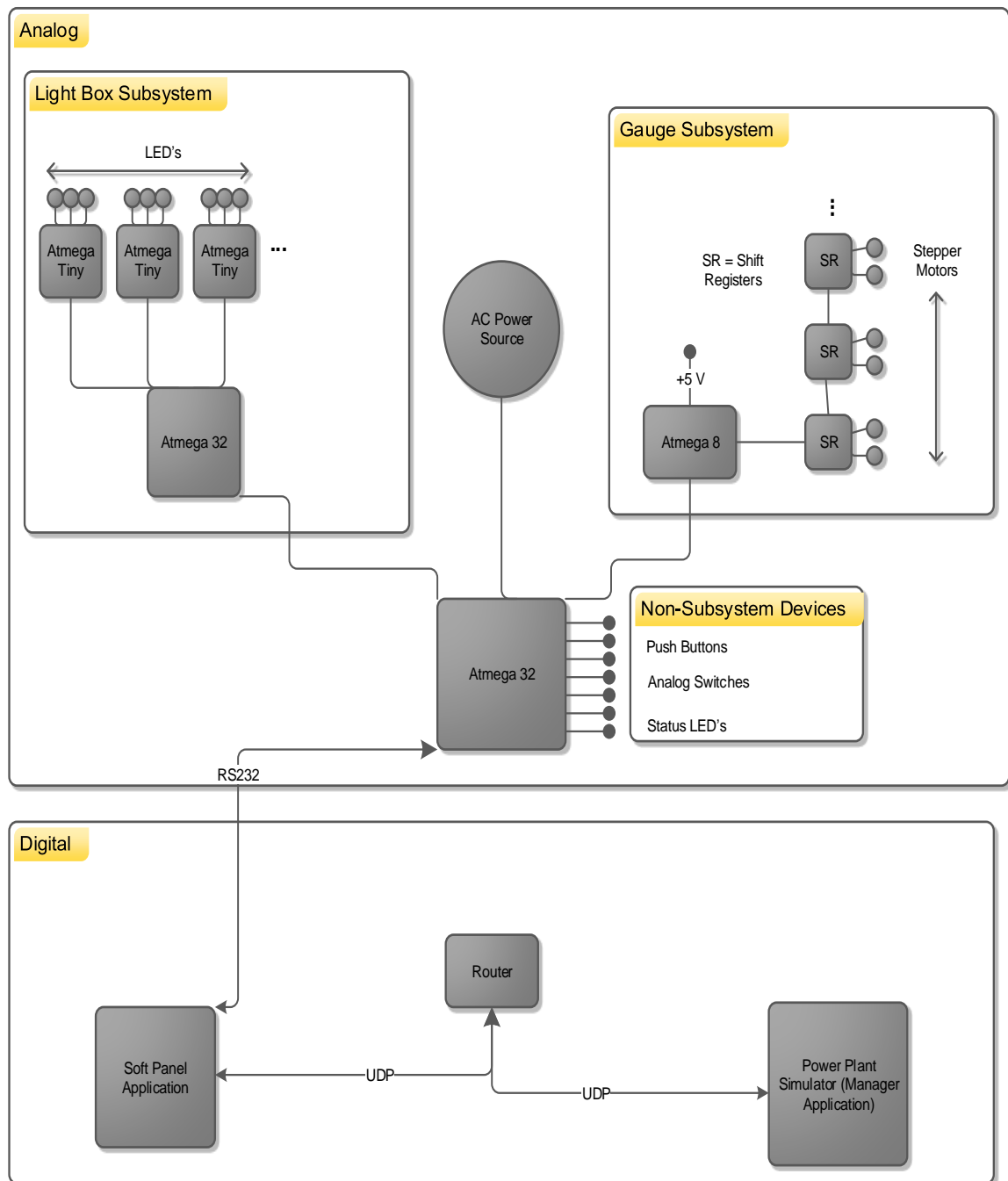


Figure 61: System Diagram

Through this diagram we have established that there will be two overall systems, the analog and digital. So far we have described the analog as our hard panel, and the digital as our soft panel. This diagram breaks down both of the panel into their subsystems and how they communicate between one another. The analog

side of things starts with the master MCU, the ATmega 32. The schematic shows the items that connect directly to it (Push Buttons, Rotary Switches and the Status LED's), the Light Box Subsystem, the Gauge Subsystem and the Power Supply. The Light Box Subsystem consists of the first slave MCU, the ATmega32, which has multiple slave MCU's connected to it, the ATmega Tiny's. The Gauge subsystem shows its slave MCU, ATmega8, and that it connects to the shift registers which in turn connect to the stepper motors. Below is a specific hardware block diagram to depict the components that make up the hard panel.

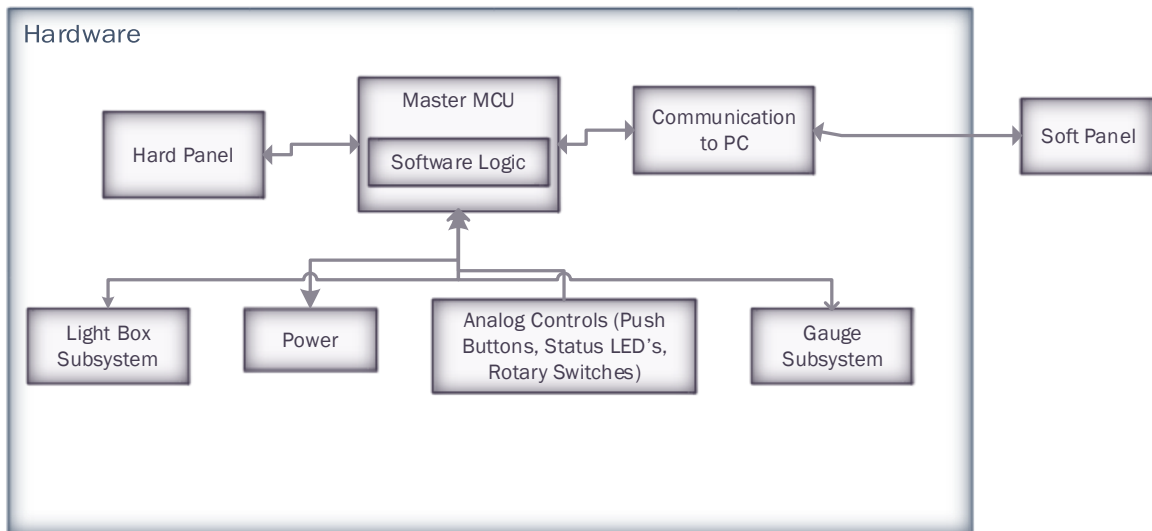


Figure 62: Hardware Block Diagram

The system diagram is a great overview of how our systems will interact, but underneath the hard panel shell will be a very complicated system of wires and circuits. We covered this breakdown briefly in our design section, primarily by paying attention to the fact that our multiple subsystems will help establish a greater sense of organization within our panel. Our goal is to make our circuit as clean and organized as possible so that simple bumps and movements cannot possible have any effect. Both of our panels have to manage to go the distance in terms of lasting because the ACTIVE lab group intends to utilize them for their research after we are completed and graduated.

Within the digital panel we have showing the communication paths in addition to how the items are connected and laid out. The Soft Panel Application is connected and communicates with the Master MCU via RS232 communication. The router both receives and sends information from the Soft Panel Application and the Power Plant Simulator via UDP Multicast.

Below we have a software block diagram showing how the code within the MCU's and the soft panel application will listen and direct the different controls, as well as communicate with the power plant simulator.

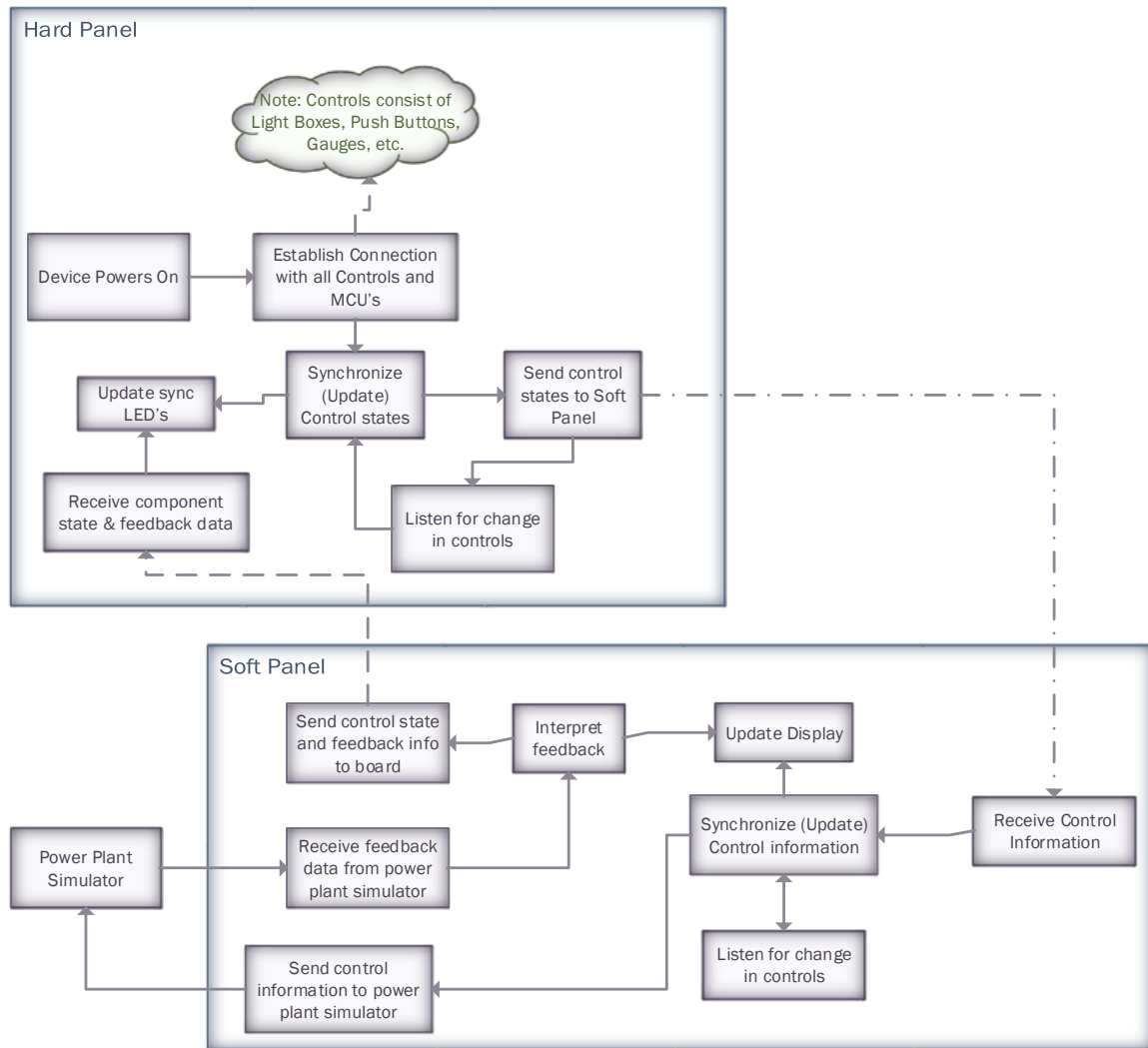


Figure 63: Software Block Diagram

In terms of parts we broke down our bill of materials below as a way to bring together all of the items we have been mentioning in our design in addition to some behind the scenes items that will be necessary for our panel. The parts below are organized by different areas of our project; the analog controls, the MCU's and then the various other parts required to bring everything together.

Item	Description	Manufacturer
Atmega8 MCU	8-bit MCU, 16MHz clock freq, 8kB flash	Atmel
Arduino UNO	Will be used for programming	
Atmega325 MCU	8-bit MCU, 16 MHz clock freq, 32kb flash	digikey
MCU	AVR tiny family MCU	ATMEL
MCU	AVR family	ATMEL

Table 27: MCU Bill of Materials

<u>Item</u>	<u>Description</u>	<u>Manufacturer</u>
Large breadboard		Sparkfun
RS232 to USB Cord	USB to RS232 cable	digikey
Female Serial connector	DB9 Female connector	digikey
Counter IC's	5-stage decade counter	digikey
Sheet Metal	3' x 2' sheet metal	Lowes
PCB	4-layer PCB	TBD
MOSFET	n-channel	Diodes Inc.
Resistor	150Ω	Panasonic
Resistor	90.9Ω	Panasonic
Relay	General Purpose 5VDC 20A	Fujitsu
Capacitor		Vishay
Diode		On Semiconductor
Terminal Block	2-pin, 6-A, 3.5mm	ED555/2DS
Inductor		Coilcraft
MOSFET	n-channel	Diodes Inc.
Resistor	SMD chip	Std
Transformer	570 uH Xfmr, ±10%	XFMR
Test Point	Test Point, Thru Hole Color Keyed	Keystone
IC	Micro-Power 100 mA LDO Regulator	Texas Instruments
IC	Flyback Green-Mode Controller	Texas Instruments
IC	Photocoupler	Texas Instruments
IC	Low-Voltage Adjustable Shunt Regulator	Texas Instruments
AC Wall Cord	Needed for Housing Unit -- Plug and Play for standalone system	Ebay
Wire	Wire, 18 AWG MTW, Stranded, 500 Ft	Carol
Wire Guard	For any cables that leave our housing unit	Electriduct

Table 28: Miscellaneous Bill of Materials

<u>Item</u>	<u>Description</u>	<u>Manufacturer</u>
Red LED Push Button	16 mm 2.2V 500 ohm, Latching	Adafruit
Green LED Push Button	17 mm 2.2V 500 ohm, Latching	Adafruit
Rotary Switch	2 Position, 380 VAC, 10 Amps	Greegoo
Stepper motor		Switec
Shift register	Counter SR, serial to serial/parallel, 3-state	Texas Instruments
LED	SMD RGB 5050	SuperbrightLEDs
Light	Panel mount indicator light green	Dialight
Light	Panel mount indicator light red	Dialight

Table 29: Analog Controls Bill of Materials

These tables are a completed and detailed list of every part we will require for our project, which we intend to start purchasing immediately. Later on in the paper, within section 7, we will break down the cost of each part for our funding summary.

6.0 Project Prototype Testing

The following sections will describe how we will run testing on both our hard panel and our soft panel.

6.1 Hardware Test Environment

6.1.1 ESD

Electrostatic discharge is a real concern for our project considering many of our subsystems contain ESD sensitive devices. Electrostatic discharge occurs between two objects that are electrically charged but with opposite fields (one positive one negative). If they come into close contact with one another a magnetic field is conducted and static electricity is created. This can cause failures in solid state devices which are the foundation for integrated circuits. Considering the heart of our hard panel is made up of ICs certain precautions need to be taken during fabrication when the devices are most sensitive since they are not properly grounded yet.

There are simple precautions that can be taken during the construction of our device in order to prevent ESD damages. We have access to a workspace with an electrostatic protective area (EPA). An EPA is where all conductive materials are grounded, there exists no highly charging materials in the vicinity and we ourselves will have to be grounded while handling the devices. How we will ground ourselves is by wearing a conducting wrist strap and working on an anti-static mat. These precautions will need to be taken while building the main control board with the master MCU, the power supply and the light box.

6.1.2 Temperature

Since our device will be operated inside under ideal room temperature conditions, temperature is not a huge concern for our project. In addition, our devices do not consume a large amount of energy so a lot of heat is not expected to be generated. However, a fan and vent will be installed in the hard panel control box to make sure that standard ventilation and cooling is performed.

6.2 Hardware Specific Testing

6.2.1 Individual Components

This section is devoted entirely to testing the hardware for the hard panel; software testing is specified in its own section and will not be discussed in this section. Each major component making up the individual subsystems will need to be tested individually before integrating all of the systems together and connecting the whole unit up to power. The major concern for hardware is that everything is connected properly with no shorts or opens, and components are biased correctly. While continuity tests are easily performed with a multimeter, extra care while installing diodes and capacitors will need to be taken.

After each subsystem has been built and the continuity for each component checked then power can be supplied to them to see if the components are working. Each subsystem will have their own expectations and desired effects when connected to power.

6.2.1.1 Push Buttons

The push buttons are easy to test. Simply apply 5VDC with a constant power supply to the positive lead and connect the negative lead to ground. If the button is pushed then the light should be turned on. If the light does not turn on, the continuity should be checked with a multimeter.

6.2.1.2 Gauges

The gauges require software in order to operate correctly. In order to test the hardware without the software element, only the actual stepper motors can be tested. To do this a variable DC power supply can be used and the inputs of each motor can be varied from 0 to 5V with each DC step corresponding to the digital value which signifies the degree to which the motor will be moved.

6.2.1.3 LED's

Each light will need to be tested individually to see if they work before connecting them. The indicator lights contain current limiting resistors so they just need to be connected to 5VDC and ground to see if they are working properly or use the continuity function of the multimeter. For the light boxes, the SMD LEDs and

resistors must be connected to the PCB. Before attaching the MOSFET check for continuity. To see if the transistor is effective in making the LED switch. Wire the schematic according to the given diagrams and apply the appropriate turn on voltage to the gate of the MOSFET. If the LED turns on then it is working properly. The LEDs have three colors, red, green and blue. Each setting will need to be tested.

6.2.1.4 Rotary Switches

The rotary switches can be tested by tying in the appropriate indicator lights for each throw of the switch. Connect the switch to power and depending on the position of the switch, the corresponding indicator light should be on.

6.2.1.5 Power Supply

The power supply will need very specific testing. Since many of our designs involve sensitive ICs it is important that our power supply is continuous and accurate. After it has been built, making sure there is continuity in all connections, a load resistance needs to be applied to each of the two outputs. With the use of either a datalogger or a data acquisition tool (DAQ) and the program DASYPOLABs, the input voltage/current and output voltage/current should be measured over an extensive period of time, at most a week the least a day. This is necessary to see if our supply is continuous, how much energy is consumed and if this changes over any length of time. A datalogger is preferred as it is more compact and can be transported easily. The DAQ requires more components and a connection to a computer so it is not ideal. Both are sufficient to acquire the desired data however.

6.2.2 Panel System

Once everything has been proven to work separately, then they can be connected together to form the entire unit. Again, continuity is very important in this step. After everything is connected together, power should be started. For this process, software specific testing will be needed to carry out the panel system hardware testing. Since everything will be connected, many of the subsystems will rely on commands from the master controller and the master MCU can be used to read whether or not it is receiving and documenting signals from the rotary switches and push buttons. If the subsystems worked correctly on their own then theoretically the hardware should work the same after integration. The overall power consumption of the device will need to be measured as well as each individual subsystems. This is necessary in order to compare to our initial estimations made and documented in section 4.7.

6.3 Software Test Environment

There are plenty things to consider regarding software testing. To be more specific, the following elements are what we will use as reference in order to successfully test the software.

- Test Strategy
 - This tells us what types of testing and the amount of testing we think will work best at weeding out the glitches that are hidden in the software
- Testing Plan
 - This is advised in order to efficiently execute testing task
- Test Cases
 - Instead of running the software blindly we will create test cases in the form of detailed examples that will be used to check that the software will actually meet its requirements.
- Test Data
 - This is what will be used when executing the test cases, which consists of both input test data and database test data
- Test Environment
 - The testing environment brings everything together by acting as the facet of which we will carry out our testing

These components will be crucial when testing the software of our overall design and if one is inadequate, our efforts will most likely fall short of what we intend to achieve. The aforementioned elements will be heavily considering for software testing, but for this section we will focus primarily on the testing environment.

The importance of testing and establishing a relevant testing environment cannot be stressed enough, for our design. We will not so much focus on if a specific feature functions accurately rather than how that specific feature affects the entire system. Our testing environment will allow us to install and configure each feature prior to its implementation into our design. This essential information given to us, through our testing environment, reduces the risk of our design being incompetent.

For us to successfully test the software of the hard and soft panel it is imperative the developers are making a collaborative effort to be on the same page. One way to go about this is to ensure that each developer is using the same Integrated Development Environment (IDE). An IDE is a software application that allows the user to maximize programmer productivity by providing extensive components with familiar user interfaces. More specifically the IDE will allow the developer to edit, comment, and understand the various sections of code. One of the key features we will be using the IDE for is to conveniently drive the code from the IDE to the MCU so that functionality is maintained throughout both panels.

Of the many IDE tools at our disposal we will be using the Arduino IDE. Not only does the Arduino IDE offer many advantages over other IDE tools, but we have already established that we are using the Arduino UNO as a programmer for the microcontrollers, which in turn will be using the Arduino IDE to aid in the process.

A few things that we require from an IDE are that it must compile C language and can be dedicated to the device so that the environment won't have to be assembled for each component. A large part of what we will be using the Arduino IDE is to aid in the programming of our microcontrollers. We already have access to an Arduino development board so along with the Arduino IDE it will make testing the microcontrollers simple. In regards to testing the microcontrollers, section 4.3.7 goes into more detail about programming the microcontroller.

Other than being free, open source, easy to use, and dedicated the Arduino IDE offers source control. Source control is a tool that tracks and provides control over changes to source code. We will be using source control extensively when testing code. Source control will allow us to take copies of the code and run tests that may or may not work on a PC. An example of how we will use source control is that if and when we make a mistake, source control allows us to retrace our steps in order start again at the initial code. There are multiple source control programs that we may choose from. One that is embedded in the Arduino IDE package is the SVN repository. An important aspect of this is when multiple members are working on the software that they only manipulate the code that is not being change by another.

6.4 Software Specific Testing

Testing the software is to not be taken lightly. Software testing is paramount for the development of all components involved in the overall design. Not only will the hard and soft panel fail to communicate with each other but if the software is not tested and debugged there will be complications with the hardware. Using the appropriate software environment along with testing the various components allows the testers to pinpoint bugs and correct them accordingly.

6.4.1 Connectivity to Hard Panel

In order to test the connectivity of the panel and tracking of data it is to be assumed that the master/slave configuration between subsystems is functioning properly. On occasion the testing of hardware and software will occur simultaneously if errors are found. Once the testing of whether the hard panel received and enabled the appropriate instruction set, we will examine if the hard panel and soft panel are synchronized. For testing, the software must be uploaded onto the development board, the Arduino Uno. The algorithms associated with each transfer of data will determine which state of synchronization the hard panel will be in. The testing must prove that once an instruction packet is sent from the PC application, the soft panel will enable the following control and send the same instruction set to the hard panel to be synchronized. Once an instruction has been confirmed the panel will remain in this state until another instruction has been set, for which the current state will be reset and the new set will be implemented. Once the testing is complete, the

hard panel will have a solid foundation for future improvements and serve as a base for further functionality.

6.4.2 Connectivity to Soft Panel

The digital display is an important component of this project and may be considered the centerpiece. A goal of the ACTIVE lab's research is to test the functionality between the operator and digital display. Our job is to provide a practical design for them to meet their goal. In order to do this our testing must be taken carefully. The testing will be roughly the same as the connectivity to control panel section. It will mainly consist of confirming that the soft panel will correctly register the instruction set sent from the PC application. The corresponding algorithms will then enable the appropriate control to be enabled. Different scenarios will be played out that reflect the various types of instructions to be satisfied. Additional software manipulation may be needed depending on the issues that arise during testing. Having the digital display have touch screen capabilities were discussed and may be an option for future improvements that will further functionality.

7.0 Administrative Content

In this section, we will describe the project management efforts used in order to organize the team and establish a schedule as well as funding.

7.1 Milestone Discussion

Our group got a late start on this project. We initially intended to pursue a gesture controlled mouse project, but we were able to get into contact with the ACTIVE Lab group and discussed potential project ideas. We both agreed that our skills would be best put to use in order to support their work on this project. We officially started work on this project on October 14th. Our work break down is shown in the following table. The way our group members have approached working on the design paper is starting as of October 21st we have individually worked to complete 7 pages each week. This plan should have us reaching 120 pages as of the week of the 18th of November. This will give us plenty of time to review our data and add any addition information to sections as needed before the deadline.

October 2013	
14	Generated funding proposal, table of contents and general idea of project scope
21	Met with ACTIVE Lab group to get project requirements
28	Started work on Design Paper. Completed preliminary sections
November 2013	
4	Continued work on research for design paper. Will submit Rough Draft design paper this Thursday (7 th)
11	Will continue work on Design Paper
18	Will continue work on Design Paper
25	Review of Design Paper – Formatting, page count, etc
December 2013	
2	Final Draft Submittal
9	Work on ordering parts, software generation
16	Continue working on software/coding generation
30	Begin prototyping

Table 30: Senior Design 1 Milestone Chart

For Senior Design 2 we will be working on implementing our work on our design paper into a working prototype which will eventually be used by the ACTIVE Lab group. The milestone chart following is an estimate of our work breakdown for the next semester.

January 2014	
6	Establish weekly goals, have all parts required in hand
13	Begin developing custom made parts (LED's, Gauges)
20	Test individual parts and pieces of software
27	Develop housing unit and power supply.
February 2014	
3	Generate PCB and work on 3-D printing for Valves
10	Start assembling completed items into housing unit
17	Start connecting Microprocessor and Microcontroller
24	Test communication between panels
March 2014	
3	Continue assembling all hardware devices
10	Continue debugging and optimizing code
17	Final software testing, final hardware generation
24	Begin work on power-point and system testing
31	Complete all system testing
April 2014	
7	Finish up Project Design, Begin work on Website
14	Present Project
21	Finals Week – Submit Project?

Table 31: Senior Design 2 Milestone Chart

7.2 Budget and Finance Discussion

We were able to create a funding proposal and breakdown very quickly in order to meet the deadline for the Duke Energy/Boeing funding opportunities provided by the college. We were very lucky to have gotten our proposal approved along with all of the other groups. Our funding breakdown which we created for the funding proposal is included below.

<u>Item</u>	<u>Quantity</u>	<u>Cost per part</u>	<u>Total Cost</u>
Red LED Push Button	13	\$1.50	\$19.50
Green LED Push Button	12	\$1.50	\$18.00
Rotary Switch	25	\$6.90	\$172.50
Stepper motor	25	Package Deal	\$70.00
Shift register	15	0.48	\$7.20
LED	25	\$0.74	\$18.50
Light	26	\$1.21	\$31.46
Light	26	\$1.27	\$33.02

Table 32: Analog Controls Materials

The devices above are necessary to accurately replicate physical analog controls seen in use today. Many of these parts are hard to find due to the lack of use of analog devices so we have to custom build many of these parts. The total cost of our analog control materials adds up to \$370.18

<u>Item</u>	<u>Quantity</u>	<u>Cost per part</u>	<u>Total Cost</u>
Atmega8 MCU	2	\$3.66	\$7.32
Arduino Uno	1	\$30.00	\$30.00
Atmega325 MCU	1	\$7.04	\$7.04
ATmega Tiny	5	\$2.56	\$12.80
ATmega32	1	\$6.04	\$6.04

Table 33: MCU Cost Breakdown

We require so many MCU's because we are working to keep our system as organized as possible, which caused us to create multiple subsystems. In addition we require the Arduino Uno to assist us in programming our microcontrollers. The total cost of our MCU's adds up to \$63.20

<u>Item</u>	<u>Quantity</u>	<u>Cost per part</u>	<u>Total Cost</u>
Large breadboard	1	\$10.00	\$10.00
RS232 to USB Cord	1	\$30	\$30.00
Female Serial connector	1	\$0.78	\$0.78
Counter IC's	4	\$0.63	\$2.52
Sheet Metal	1	\$19.28	\$19.28
PCB	4	\$50.00	\$200.00
MOSFET	75	\$0.16	\$12.00
150 ohm Resistor	25	\$0.10	\$2.50
90.9 ohm Resistor	50	\$0.02	\$1.20
Relay	1	\$2.05	\$2.05
Capacitors	20	\$0.29	\$5.80
Diodes	10	\$0.35	\$3.50
Terminal Block	8	\$0.63	\$5.04
Inductors	2	\$0.60	\$1.20
MOSFET	1	\$0.16	\$0.16
SMD chip Resistors	28	\$0.10	\$2.80
Transformer	1	\$15.00	\$15.00
Test Point	26	\$0.21	\$5.51
Power Regulator IC	1	\$0.42	\$0.42
Controller IC	1	\$1.51	\$1.51
Photocoupler IC	1	\$0.67	\$0.67
Regulator IC	4	\$0.29	\$1.15
AC Wall Cord	1	\$15	\$15.00
Wire	1	\$60.00	\$60.00
Wire Guard	1	\$5.00	\$5.00

Table 34: Miscellaneous Parts Cost Breakdown

Adding in a total of \$403.09 for the miscellaneous items we require in order to build and connect our panels, the overall total cost of our project is estimated to be \$836.47. This amount is less than what we requested which allows us some leeway with project requirements (like printing our paper and making our project board for SD2) as well as more freedom with testing parts and potential mishaps.

7.3 Work Roles and Distribution

Our group was lucky enough to consist of three Electrical Engineering students in addition to a Computer Engineering student. This allowed us to be well rounded

in our approach to this project. We have a diverse amount of experiences and the information that follows depicts the breakdown of our estimated contributions to the project design and implementation.

Mike Zellars Computer Engineering

Responsibilities:

I've taken on the role of the lead software developer of this project. This can be summed up into three main responsibilities. The first is designing and developing the computer application that runs the digital control board, which includes a graphical user interface. Responsibility number two pertains to data transfer throughout the system. The digital control panel, analog control panel, and power plant server will be sending and receiving information to each other. I will be responsible for handling this information accordingly. My third responsibility is to program the microcontrollers embedded within the analog control panel to drive all analog components. These components will hold states and may change states at any time, and the microcontrollers are the brains of this operation. In addition I took on the design of the gauges, which included researching potential stepper motors and figuring out how to connect multiple gauges within the same circuit.

Cassandra Todd Electrical Engineering (Hardware)

Responsibilities:

For the duration of this project my role has been more of a technical one. For example, I have worked on the design of the power supply and the light box as well as the other lighting aspects of the hard panel. In addition, I researched the process of analog to digital conversion and plan on helping with the smooth integration of the hardware and software. For my future roles I hope to utilize my past experience working in a maintenance shop to help fabricate the control panel and perform tests that accurately monitor the power dissipation throughout the system.

Joe Nichols Electrical Engineering (Hardware and Communications)

Responsibilities:

My responsibilities for the duration of this project echoed the responsibilities that of a hardware engineer. My main contribution was dedicated to the research and development of the main microcontroller. The key components I administered were the implementation of select analog controls, communication between the main MCU and its various peripherals, and communication to the soft panel. I also served as an aid in other facets of the project. This led to me covering the research and design of the control panel housing unit, printed circuit board, and software testing.

Kristen Berman Electrical Engineering (Systems and Project Management)

Responsibilities:

Through the course of this project I took on the responsibility of Project Manager. This put me in charge of establishing communication with the ACTIVE Lab,

organizing the team and enforcing deadlines. In addition I had experience with the nuclear industry due to an internship at a nuclear power plant so I was able to take on many of the sections of our paper describing the information related to nuclear power plants. For my technical roles, I worked on designing the plan for 3-D printing our Valves and completed the research for that. In addition I intend to pursue a role of systems integration for our second semester by debugging and making sure all components are communicating correctly.

8.0 Operation of System

This section establishes instructions for a new user to be able to operate our system and use our control panel. Preliminary user requirements will be established by the ACTIVE group, essentially what this means is that they will be selecting research participants to use our panels in conjunction with the power plant simulator in order to get the readings that were discussed in section 3.2. The majority of these people will be completely novice at using a control panel, especially a nuclear specific panel. This means that there needs to be clear and concise directions for this new user to be able to operate our system of panels. Below we go into greater detail.

8.1 Requirements

This section goes into detail for what each PC will require in order to be able to run testing. Testing consists of the hard panel and power plant simulator establishing a connection to the soft panel, where both the soft panel and the simulator are PC's.

- Requirements for Soft Panel Operation
 - Computer (Computer A)
 - Running a Windows Operating System
 - RS232 interface
 - Ethernet interface
 - Java Platform SE installed
 - Soft Panel Java Application
- Requirements for Power Plant Simulator Operation
 - Computer (Computer B)
 - Running a Windows Operating System
 - Ethernet interface
 - Java Platform SE installed
 - Power Plant Simulator Java Application
- Other equipment needed
 - Router
 - RS232 cable
 - Ethernet cable (x2)

8.2 Booting up the System

1. Power on both computers and connect them to the router with the Ethernet cords.
2. Execute the soft panel java application on Computer A.
3. Execute the power plant simulator application on Computer B.
4. When both applications have loaded, the system will function.
5. To use the hard panel, connect it to Computer A with the RS232 cable.
6. Power on the hard panel.
7. Follow the instructions on the soft panel application to synchronize the two control panels.
8. When synchronized, the complete system will be ready for normal operation.

8.3 Using the System

- Using the Hard Panel
 - The user may provide control input through the hard panel in two ways, by physically pushing buttons and flipping switches.
 - When new input occurs, it is recognized by the soft panel and forwarded to the simulator. The simulator may respond with commands, which are heard by the soft panel and forwarded on to the hard panel.
 - Output controls include gauges and LEDs and will change state as commanded by the simulator.
- Using the Soft Panel
 - The user may provide control input through the soft panel in two ways, by using the mouse to click on the digital representation of buttons and switches.
 - When the user changes input in this way, the two control panels fall out of sync. The system will not continue normal operation until the control panels get back in sync. The soft panel will provide the user with step-by-step instructions on how to do this.
 - Once the control panels are in sync again, the system will continue operation. The new input is sent to the simulator, which will respond accordingly.

Appendix A: Copyright Permissions

1. Sparkfun



SparkFun Customer Service <cservice@sparkfun.com>

Fri 11/22/2013 2:31 PM

To: kristenberman@knights.ucf.edu;

Hello,

You are more than welcome to take images and descriptions from our website for use in your project. SparkFun releases all product images under Creative Commons license. This means you are free to use our images for anything you'd like.

We only ask a few things:

1. Place a credit "From SparkFun Electronics" and also credit our photographer "Photo taken by Juan Peña" wherever the photo is shown.

2. If you are not going to be using the images but you will be using the products then you really don't have any obligation to cite SparkFun. But if you would still like to credit us as your source we would be more than grateful!

Best Regards,

Joelle Paxton

Customer Service Representative

SparkFun Electronics

www.sparkfun.com

303.945.2984 x603

2. Atmel products

RE: permission to reprint information

Chugh, Paramjyot <Paramjyot.Chugh@atmel.com>

Thu 11/21/2013 5:32 PM

To: ctodd42@knights.ucf.edu;

Cc: Mata, Cecilia <Cecilia.Mata@atmel.com>; Merken, Leo <Leo.Merken@atmel.com>;

Dear Cassandra,

We appreciate your checking with us for the permission to use our copyrighted material from our website/datasheets. Since the kind of reproduction you described, could be considered as the type of permitted copyright "fair use", no formal approval is required from Atmel. We only ask you to attribute the source of material you use in your report.

Thank you for your interest in Atmel products.

Regards,

Paramjyot Chugh | Intellectual Property Portfolio Manager

Atmel Corporation

Tel: +1 408.487.2706 | Fax: +1 408.436.4111 | Mobile: +1 408.761.7357

1600 Technology Drive | San Jose, CA 95110

3. Adafruit LED Strip

Re: [[Press/Media]] Press / media inquires

adafruit@gmail.com on behalf of Adafruit Industries <support@adafruit.com>

Wed 11/13/2013 7:46 PM

To: Cassandra Todd <ctodd42@knights.ucf.edu>;

please do!

On Wed, Nov 13, 2013 at 2:04 PM, Cassandra Todd <ctodd42@knights.ucf.edu> wrote:

contactname : Cassandra Todd

email address : ctodd42@knights.ucf.edu

contact us 2 section : press

useragent string : Mozilla/5.0 (Windows NT 6.1; rv:14.0) Gecko/20100101

Firefox/14.0.1

message text : Hello there!

I'm a senior in Electrical Engineering at the University of Central Florida and I'm in the middle of writing a design paper for one of my classes where we are considering using your Digital RGB LED Weatherproof Strips for our project. I was wondering if you could give me permission to use your image of the strip of lights all lit up located on this page, <http://www.adafruit.com/products/306>, for our report? thank you for your time!

Client IP: 132.170.175.255

4. Adafruit LED Pushbuttons

[[Press/Media]] Press / media inquires



adafruit@gmail.com on behalf of Adafruit Industries <support@adafruit.com>

Mon 11/25/2013 1:40 PM

To: Kristen Berman <kristenberman@knights.ucf.edu>;

thanks for the note, feel free to, just link to adafruit.com as the credits.

On Mon, Nov 25, 2013 at 1:37 PM, Kristen Berman <kristenberman@knights.ucf.edu> wrote:

contactname : Kristen Berman

email address : kristenberman@knights.ucf.edu

contact us 2 section : press

useragent string : Mozilla/5.0 (Windows NT 6.1; rv:14.0) Gecko/20100101

Firefox/14.0.1

message text : Hello! My name is Kristen and I'm a senior in Electrical Engineering at UCF and am working on my design paper for graduation. I am interested in using some of your parts in my project and would like to be able to receive copyright permission to use some of the images in my report.

It would be the following push buttons:

<http://www.adafruit.com/products/1440>

<http://www.adafruit.com/products/1442>

Thank you!

Client IP: 132.170.80.191

5. Super-Bright LED's
Super Bright LEDs

Steve Miller <steve-m@superbrightleds.com>

Thu 11/14/2013 12:38 PM

To: ctodd42@knights.ucf.edu <ctodd42@knights.ucf.edu>;

Hi Cassandra,

Congratulations of being on your way to completing your education! We would happily allow you permission to use any of our diagrams or schematics for your project. Please let me know if you need anything else.

Thank you,

--

Steve Miller
Director of Marketing / Sales
314.972.6200 ext 3065
866.590.3533
steve-m@superbrightleds.com

Super Bright LEDs
4400 Earth City Expressway
Earth City, MO 63045

www.superbrightleds.com

6. Zigzag LED

from:  **PRO at0mbxmb**

when: Nov 5, 2013. 8:20 AM

subject: **re: individually addressable LED matrix**

Hi,

Sure, you have my permission to use the photos if you credit them to Michelle Leonhart with a link to the instructable.

Thanks for asking! Good luck on your paper.

Original message sent 9 days ago by **dfcx15**

Hi! I'm a senior in electrical engineering and our senior design project will involve an LED matrix. W

You created a really great instructable by the way! I'm pretty excited to start working on my own soon

~Cassie

<http://www.instructables.com/id/How-to-Make-an-Individually-Addressable-LED-Matrix/step3/ZigZag/>

7. Davide Gironi – Stepper Motors



zellarsm <zellarsm@knights.ucf.edu>

Fri 11/29/2013 3:23 PM

From: Davide Gironi <davide.gironi@gmail.com>

Sent: Friday, November 29, 2013 2:02 PM

To: zellarsm

Subject: Re: Requesting permission - Stepper04multi library

you are welcome,
just use it.

when your projects will be completed, please send me some image of it.

best regards,
davide

Dear Mr. Gironi,

I'm a computer engineering student at the University of Central Florida currently working on a project involving multiple stepper motors. I have found your blog post pertaining to driving up to 12 stepper motors on Atmega using 74hc595 very helpful. In my project, I plan on building the same circuit and using your stepper04multi library and would like to thank you for providing this.


In my design paper, I would like to include the image of the circuit schematic you included in the library. I need permission from you in order to do so. It would be greatly appreciated.

Sincerely,
Michael Zellars

8. Circular Gauge Permission:
http://commons.wikimedia.org/wiki/File:Georgetown_PowerPlant_Museum_gauges_02.jpg

Licensing: [\[edit\]](#)

Joe Mabel, the copyright holder of this work, hereby publishes it under the following license:



Permission is granted to copy, distribute and/or modify this document under the terms of the [GNU Free Documentation License](#), Version 1.2 or any later version published by the [Free Software Foundation](#); with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled [GNU Free Documentation License](#).

This file is licensed under the [Creative Commons Attribution-Share Alike 3.0 Unported](#) [\[icon\]](#) license.

Attribution: Joe Mabel

You are free:

- **to share** – to copy, distribute and transmit the work

9. Microcontroller Pro's Corporation

Hi Joseph,
you have the permission to use our diagrams, as long as you cite the source.

regards
Volker
MicroController Pros LLC.
<http://microcontrollershop.com>
The World's Largest Embedded Tool Selection
phone +1-775-636-7755
fax +1-215-243-6071

10. Texas Instruments

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Appendix B: Works Cited

1. Reinerman-Jones, Lauren, Svyatoslav Guznov, Joseph Mercado, and Amy D'Agostino. *HCI NRC Final Paper 2013*. N.p.: UCF Institute for Simulation & Training, 2013. PDF.
2. *Microcontroller Interfacing Techniques*. Tech. BiPOM Electronics, Inc., 3 Apr. 2005. Web. 6 Nov. 2013. <http://www.bipom.com/applications/micro_interfacing.pdf>.
3. "Introduction to I2C and SPI Protocols." *Byte Paradigm*. N.p., n.d. Web. 30 Oct. 2013. <<http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/>>.
4. "Using the I2C Bus." *Robot Electronics*. N.p., n.d. Web. 30 Oct. 2013. <http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html>.
5. Soffel, Volker. "Synchronous Microcontroller Communication Interfaces: SPI and Microwire versus I2C." *MC Pros*. N.p., 1 May 2003. Web. 30 Oct. 2013. <<http://www.ucpros.com/work%20samples/Microcontroller%20Communication%20Interfaces%201.htm>>.
6. *8-bit Atmel Microcontroller*. Tech. Atmel Corporation, 2011. Web. 30 Oct. 2013. <<http://www.atmel.com/Images/doc2570.pdf>>.
7. *MC9S12NE64 Data Sheet*. Tech. Freescale Semiconductor, June 2006. Web. 6 Nov. 2013. <http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S12NE64V1.pdf>.
8. Lucas, Bill, and Steven Torres. *Implementing an Ethernet Interface with the MC9S12NE64*. Tech. Freescale Semiconductor, 2004. Web. 6 Nov. 2013. <http://www.freescale.com/files/microcontrollers/doc/app_note/AN2759.pdf>.
9. *MC9S12XDP512 Data Sheet*. Tech. Freescale Semiconductor, Oct. 2009. Web. 6 Nov. 2013. <http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S12XDP512RMV2.pdf>.
10. *32-Bit Microcontrollers*. Tech. Microchip, n.d. Web. 16 Nov. 2013. <<http://ww1.microchip.com/downloads/en/DeviceDoc/61156H.pdf>>.
11. Huffman, Steven. "PCB 101: Everything You Need to Know About Printed Circuit Boards." *The Technology Lounge*. N.p., 8 Feb. 2013. Web. 16 Nov. 2013. <<http://www.thetechnologylounge.com/computers/pcb-101-printed-circuit-boards/>>.
12. Leonard, Michael. "How To Design the Perfect PCB – Part 1." Web log post. N.p., n.d. Web. 20 Nov. 2013. <<http://www.michaelhleonard.com/how-to-design-the-perfect-pcb-part1/>>.

13. "How to Build a Metal Control Panel." *Snotmonkey*. N.p., n.d. Web. 20 Nov. 2013. <<http://www.snotmonkey.com/articles/how-to-build-a-metal-control-panel/>>.
14. *National Instruments*. N.p., n.d. Web. 20 Nov. 2013. <<http://www.ni.com/>>.
15. *National Programme on Technology Enhanced Learning*. N.p., n.d. Web. 20 Nov. 2013. <<http://nptel.iitm.ac.in/>>.
16. "Motors and Microcontrollers 101." *Nerdkits*. N.p., n.d. Web. 15 Nov. 2013. <http://www.nerdkits.com/videos/motors_and_microcontrollers_101/>.
17. "A Simple Java UDP Server and UDP Client." *Systembash*. N.p., 17 Sept. 2013. Web. 15 Nov. 2013. <<http://systembash.com/content/a-simple-java-udp-server-and-udp-client/#comments>>.
18. "Networking Basics." *The Java Tutorials*. N.p., n.d. Web. 15 Nov. 2013. <<http://docs.oracle.com/javase/tutorial/networking/overview/networking.html>>.
19. Abrams, Lawrence. "TCP and UDP Ports Explained." *Bleeping Computer*. N.p., 24 Mar. 2004. Web. 10 Nov. 2013. <<http://www.bleepingcomputer.com/tutorials/tcp-and-udp-ports-explained/>>.
20. Gironi, Davide. "Drive Multiple (up to 12) Stepper Motors on Atmega." Weblog post. N.p., 8 Dec. 2012. Web. 15 Nov. 2013. <<http://davidegironi.blogspot.com/2012/12/drive-up-to-12-stepper-motors-on-atmega.html>>.
21. Szczys, Mike. "AVR Programming 01: Introduction." *Hack A Day*. N.p., 23 Oct. 2010. Web. 22 Nov. 2013. <<http://hackaday.com/2010/10/23/avr-programming-introduction/>>.
22. "AVR Tutorial." *Ladyada*. N.p., 27 Apr. 2012. Web. 22 Nov. 2013. <<http://www.ladyada.net/learn/avr/avrdude.html>>.
23. "Using an Arduino as an AVR ISP." *Arduino*. N.p., n.d. Web. 22 Nov. 2013. <<http://arduino.cc/en/Tutorial/ArduinoISP>>.
24. *8-Bit AVR Microcontroller with 32 KBytes of In-System Programmable Flash*. Tech. Atmel, 2011. Web. 28 Nov. 2013. <<http://www.atmel.com/Images/doc2503.pdf>>.
25. *8-Bit Atmel with 8KBytes In-System Programmable Flash*. Tech. Atmel, 2013. Web. 28 Nov. 2013. <http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf>.

Appendix C: List of Tables and Figures

Figure 1: LED Push Buttons	12
Figure 2: Classic analog gauges	12
Figure 3: A bipolar stepper motor	13
Figure 4: Tentative gauge design	14
Figure 5: 8 by 8 LED Array	15
Figure 6: RGB LED strips	16
Figure 7: RGB LED Strips connected in zig-zag pattern	17
Figure 8: Block Diagram for Analog to Digital Converters	21
Figure 9: A graphical representation of an Analog to Digital Converter	22

Figure 10: Schematic of the on-chip A/D Converter for the ATmega16	23
Figure 11: ADC Timing Diagram of a Single Conversion	24
Figure 12: SPI Single Master Multiple Slaves Configuration	27
Figure 13: I ² C 2-wire Interface	28
Figure 14: I ² C Communication Protocol	29
Figure 15: Summary of Common Asynchronous Interfaces	31
Figure 16: Multicasting	34
Figure 17: Unicast	35
Figure 18: Class Breakdown of an IP Address	36
Figure 19: Port Binding	36
Figure 20: Block Diagram of AVR Architecture	42
Figure 21: Processor Block Diagram	44
Figure 22: Interrupt Controller Module	45
Figure 23: Ethernet Controller Block Diagram	46
Figure 24: Using DAPA as a programmer	48
Figure 25: Arduino Uno as an ISP	49
Figure 26: Block diagram of basic process flow of a closed-loop SMPS	51
Figure 27: Example oscilloscope images and schematics of (a)half wave and (b)full wave rectifiers	52
Figure 28: (a) LC Filter and (b) π or CLC Filter	53
Figure 29: Example of a “Chopper” circuit	54
Figure 30: Schematic of simplified buck converter	55
Figure 31: Schematic of simplified flyback converter	55
Figure 32: Screenshot image of the Power Stage Designer Tool	56
Figure 33: Exterior views of analog gauge	63
Figure 34: Interior Side View	64
Figure 35: Juken Switec X27.168 Stepper Motor	65
Figure 36: A gauge system supporting up to twelve gauges	66
Figure 37: Sketch A - Interior Front View of Gauge	67
Figure 38: Sketch B – Interior Side View of Gauge	68
Figure 39: RGB LED SMD chips	71
Figure 40: Schematic of an LED and MOSFET looking out of an I/O pin from the ATTiny	72
Figure 41: Calculations made to find on-resistance range for MOSFETs	73
Figure 42: Equivalent circuit of the MOSFET	73
Figure 43: Block Diagram of the ATmega32u4 microcontroller	74
Figure 44: Block Diagram of the ATTINY2313-20SU microcontroller	75
Figure 45: Light Box LED schematic showing the first five RGB SMDs wired into the slave and master controller	76
Figure 46: Simplified circuit for status lights	77
Figure 47: Schematic of the status lights that indicate synchronization between hard and soft panels	77
Figure 48: Single Master Multiple Slave Configuration	80
Figure 49: SPI Master/Slave Interconnection	80
Figure 50: Schematic used to Configure Push buttons/Switches	82
Figure 51: Schematic used to Configure the RS232 Communication	83

Figure 52: Schematic used to Configure Gauges	84
Figure 53: PCB design of the Master MCU	91
Figure 54: Power schematic	93
Figure 55: Power efficiency of transforming circuit	95
Figure 56: Sequence Diagram of User Interaction with Hard Panel	97
Figure 57: Sequence Diagram of User Interaction with Soft Panel	98
Figure 58: Tentative device icons to be used in the GUI	98
Figure 59: Hard panel powers on	102
Figure 60: MMCU receives data from soft panel	103
Figure 61: System Diagram	105
Figure 62: Hardware Block Diagram	106
Figure 63: Software Block Diagram	107
Table 1: Hardware Specifications and Requirements	4
Table 2: Software Specifications and Requirements	4
Table 3: Table describing potential LED RGB strip details	17
Table 4: A breakdown of an analog voltage output and the digital output that corresponds to it.	22
Table 5: Advantages and Disadvantages of the SPI Protocol	28
Table 6: Advantages and Disadvantages of the I ² C Protocol	30
Table 7: Advantages and Disadvantages of the RS232 Standard	32
Table 8: Advantages and Disadvantages of the RS485 Standard	32
Table 9: Advantages and Disadvantages of the USB Protocol	33
Table 10: Advantages and Disadvantages of the Ethernet Protocol	33
Table 11: Communication Interfaces	43
Table 12: Osh Park PCB Information	59
Table 13: Advanced Circuits PCB Information	60
Table 14: Express PCB Information	60
Table 15: X27.168 Technical Data	65
Table 16: Table naming each individual LE	69
Table 17: Table listing the pros and cons of two microcontrollers in consideration for use	70
Table 18: Table listing electrical specifications for the LEDs	71
Table 19: Table listing the specifications for the mini-microcontrollers to be used	72
Table 20: Microcontroller Selection Characteristics	78
Table 21: Message Keys	89
Table 22: Gauge Specific Message Keys	89
Table 23: Valve Specific Message Keys	89
Table 24: LED Specific Message Keys	90
Table 25: Panel Specific Message Keys	91
Table 26: Power breakdown for hard panel system	94
Table 27: MCU Bill of Materials	107
Table 28: Miscellaneous Bill of Materials	108

Table 29: Analog Controls Bill of Materials	109
Table 30: Senior Design 1 Milestone Chart	115
Table 31: Senior Design 2 Milestone Chart	115
Table 32: Analog Controls Materials	116
Table 33: MCU Cost Breakdown	116
Table 34: Miscellaneous Parts Cost Breakdown	117