EE 330 Project List
Fall 2012

The project is to be worked on in groups of no more than two students. Select a project from the following options. The projects are available on a first come first serve basis per lab section. No two groups per section will be allowed to work on the same project and groups must be chosen within one's section. Partially completed projects will not be accepted.

Expected Deliverables

A project report with appropriate organization is required. You will need to show any analytical work used to design your system. Include the schematic and also provide the layout for the circuit. The layout must pass DRC and LVS to be considered complete. Simulations should also be performed to verify the functionality of the circuit, and they should be thorough and automated if possible. In addition, you will be presenting the final outcome of the project to the instructor/TA. Use any visual aids as deemed effective.

1. The Verilog code
2. The test bench code
3. The simulation results (multiple results are necessary)
4. RTL to Gate-level Netlist synthesis
5. Run place and route tool for the synthesized netlist using Cadence Encounter.
6. Import Encounter layout into Virtuoso in order to run DRC/LVS.
7. Thorough report of the design process and final completed design. This is a formal, professional report that will indicate to your boss whether you deserve to stick with the company during these tough economic times.
8. Presentation (10-15 minutes) – semi-formal (business casual or above). Outline the project, with important details. Then present the design with some of the simulations that show the requirements were met. Each student per group must present. Finally, give the final layout including the size and number of gates used. Extra credit will be awarded to those who explore the intricacies of digital timing and present results related to this work.

Project 1 - Smart Responder (Answering device) design

There are four persons A, B, C, D who are part of a mathematic competition. For this competition, they have to use the smart responders to answer the question (think Jeopardy but only numerical answers). The first person to trigger his responder will have a chance to answer the question. If that competitor types in the correct answer within a required time limit then he/she will get the points. Otherwise they will lose a point. There is also a judge who will control the switch of everyone's smart responder with a reset button. If and only if the judge pushes reset button, the four responders will be able to work. The objective of this project is to use hardware description language (VHDL) to design a smart responder for this competition as described above.

<u>**Design content:**</u>

There are four main parts of this responder design.

1. **To test who is the first one to press their responder**
   After judge triggered the reset button, four competitors start to push their responders and the fastest one should be picked up (we assume four persons cannot press responders simultaneously with a same speed, only one person is fastest). If person "A" firstly presses the responder, "A"'s responder is trigger ON and then "A" will be allowed to provide answer signal. The else person's (B, C, D) responders will be locked to OFF.

2. **To design a timer which can count down the time limit. Assume a set limit of 15 seconds at first and then consider adding the ability to change this time with external inputs.**
   After the competitor's responder is triggered, the timer starts to count down from a certain number until zero. Once the number reaches zero, the input stop is triggered to 1 and at the same time that contestant's responder is turned OFF. If at any time a wrong answer is entered, the timer stops and one point is lost by that contestant. One point is also removed if time expires without a correct answer. If the person's answer is right, the input stop is triggered to 0 and two points will be earned. Assume that once a new question is given, the judge presses the restart button to signal a new question.

   A 60 second master timer should also be present for each question. If nobody responds in that time, no points are awarded and all responders are turned OFF until the judge presses restart after asking the next question.

3. **To record the total points each competitor obtained**
   If the competitor types the right answer in the limiting time period, two points will be awarded but if they type in the wrong answer or fail to answer in a required limiting time, one point will be taken off.

4. **To compare input answers to the master solution (Question input)**
   Each question asked has an 8-bit solution with QUESTION input. Assume this solution comes from a database but is fed to the hardware as an 8-bit number when the question is asked. If the ANSWER typed by a contestant (also an 8-bit input to hardware) matches QUESTION, then the contestant receives the two point reward. If the answer does not match, one point is removed and the question is still up for grabs by the other remaining contestants until the 60 second timer expires.

Comment:

1. **The system frequency is required to be 10 KHz so you need a frequency divider when you design the timers.**
2. **Below are the main inputs and outputs although other secondary outputs may exist depending on how you decide to control the responders.**

Name of Input	Symbol	Size
Clock	CLK	1 bit
A	A	1 bit
B	B	1 bit
C	C	1 bit
D	D	1 bit
Question	Q	8 bits
Anwer	Ans	8 bits
Judge reset	RESET	1 bit

Name of Output	Symbol	Size
score count1	C1	8 bits
score count2	C2	8 bits
score count3	C3	8 bits
score count4	C4	8 bits

Project 2 - Video Poker (5 card stud)

Part 1: A working game

In this project you are to create a fully functioning video poker game that works through hardware instead of software (for security purposes). Become familiar with the link below on the standard rules of the game. Essentially this is a casino game in which a player is given 5 cards, and attempts to keep the best cards for the highest payout. The player plays against the machine. For this implementation we are assuming a standard deck, and rules as shown in the link. The starting amount for a player is $100 and bets can be made between $10 and $100 (if money is available) by $10 increments. The game ends when the player asks for 'payout' or runs out of money/ cannot complete another bet.

Functionality to consider

- Keyboard input for cards 1-5 – allows the user to decide to keep/fold a given card.
- Players can 'hold' between 1 and 5 cards, and should be able to select/deselect in any order before continuing for new cards.
- Keyboard input to signal that choice is final – after which those cards to remove are replaced with cards from the deck. This is the final 5 card hand.
- Repeat bet option – after a hand, this input allows the same bet to be quickly made for the next deal
- Cards must be randomized – because this is hardware and not real-life, pseudo-random algorithms should be used.
- $1 is equal to 1 credit
- How to complete a payout?
- Indicators for a 'win' and 'loss'
- The player should never run out of cards, (the game doesn't end when you go through 52 cards). Consider implementing this game such that each hand the computer "shuffles" the deck and restarts. This is not like Blackjack where the deck is used until one runs out of cards (thus allowing some to get good at catching the deck while it is hot).

Given:

- Standard casino rules Full-pay games (http://en.wikipedia.org/wiki/Video_poker)
- Use a standard deck of playing cards
- Starting money at $100
- Min bet at $10, max at $100
- Assume the keyboard at the station can have as many keys as necessary even if it looks horribly ugly. So one key might exist to select bets of $10, $20, $30, etc. Better implementation will use other methods.

Part 2: Let's make more money (extra credit)

Unfortunately the casino owner is very greedy, and doesn't like the house to lose money. He has hired you to modify the game strategy to give the house a bit of an advantage.

If a player has won more than 5x his total bet in the last 10 rounds, the card strategy should be tweaked. As in, if the player is doing well (by the owner’s standards), the probability he will keep winning keeps going down, and the house will start winning again.

<u>Functionality to consider</u>

- What type of algorithm can you make to accomplish card tweaking? Consider ‘peaking’ at the next card
- How much money is the house making?
- Is this tactic legal? Who cares it’s on private land and you’re getting paid.

Project 3 - Extruder Project

An extruder is a machine designed to transform raw material into some specific form by forcing the material through a die. The material can vary from metals, to plastics, to foods. The process is usually controlled by a programmable logic controller (PLC), but your company is experimenting with a less expensive control system. You have been asked to develop a controller for a specific extrusion process.

A typical extruder uses a screw to force a material through a die. In this particular case, the product must be heated and mixed with a solvent before it can flow into the extruder. This is accomplished by mixing a dry stock material with a solvent. The dry material is moved by a screw conveyor driven by motor 1 into the mixer, while the solvent is added by motor operated valve #1 (MOV1). The fill level is monitored by a level sensor (an ultra-sonic device). Stage 1 involves filling the mixer and bringing it to a specified temperature.

Upon reaching the right temperature, stage 2 begins. MOV2 opens a valve that allows the mixture to flow into the extruder. The extruder is operated by motor 3, and the product is pushed through a die onto a conveyor belt ran by motor 4.

Each motor is operated from a variable frequency drive to control its speed based on a 10-bit input from the controller. The machinery is all belt driven by the motors, so to protect the equipment, there are 8-bit speed detectors to verify that the belts are not slipping. The MOVs are controlled by an 8-bit input from the controller. There are 2 temperature sensors, each with an 8-bit output to the controller. Finally, the level sensor provides a 10-bit input to the controller.

Inputs:

Speed Detectors (detect belt slipping) (x4)	8 bit
Temperature Sensors (x2)	8 bit
Level Sensor	10 bit
Flow sensor (Flow or no flow)	1 bit

Outputs:

Variable Frequency Drive (x4 motors)	8 bit
MOV controllers (x2)	8 bit
Alarm	1 bit

Stage 1 - Load the Mixer

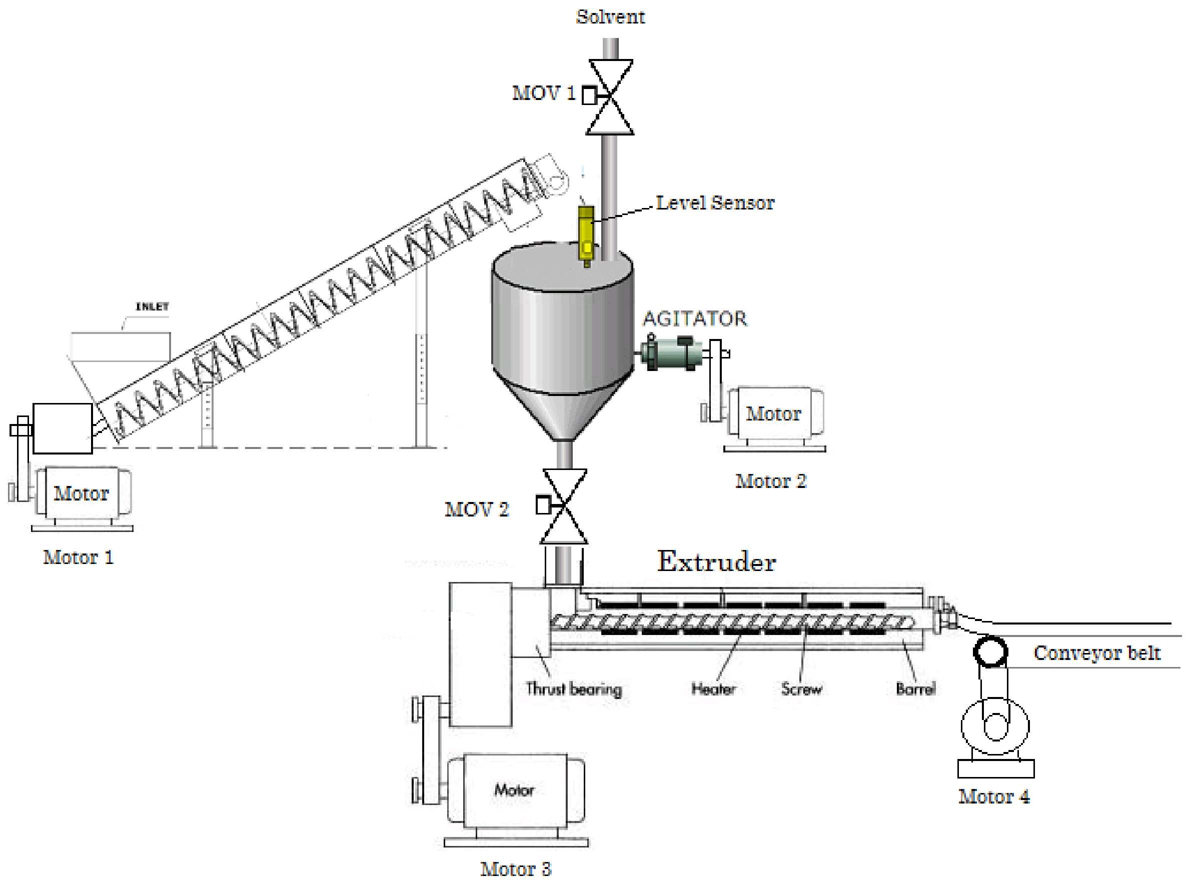
Turn on the heaters and fill the Mixer until the level sensor detects a distance of 0.5meters. Its output is scaled by value=1/3 distance in centimeters. Motor 1 should run at 1800RPM. It is scaled linearly such that an input to the VFD of 512 will give 1800 RPM. MOV1 should always be sent a value that is 6.25% of motor RPM. It has been calculated that this will result in the correct ratio of solvent to material. The temperature sensor must reach and maintain 800^{o} (134 from its ADC output) before beginning stage 2.

Stage 2 - Begin Extrusion

Motor 3, motor 4, and MOV2 should turn on. The extrusion speed is controlled from an 8-bit input such that Motor 3's RPM will be 14 times this value. Motor 1 will be 120% the speed of motor 3 with MOV1 at 6.25% of motor 1. The motors should all be synchronized, but if the mixer level reaches 120% of full, motor 1 and MOV1 should stop. Conversely, if the mixer reaches 80% of full, motor 1 and MOV1 should increase by 20%.

Safeties

If any of the speed sensors should drop below 90% of their speed input to the VFDs, the system should shut down. If a temperature sensor on the extruder should drop below 800°, the system should shut down except for motor 3 and motor 4 to clear the extruder. A flow sensor will verify that material is flowing from the mixer to the extruder, if no flow is sensed, the system should shut down as it did for the temperature sensor. Any of these circumstances will signal an alarm and await a system reset.



Project 4 - Security door

The system that allows students into buildings and labs needs to be redone. This requires each door to have a place for students to slide their cards and then depending on how the room is programmed, it may or may not allow the student to be allowed entrance. The inputs and outputs for this system are shown below. The system will be able to be programmed by the method shown below. There will be a 100 by 24-bit memory to store all of the student's ID numbers that have credentials to enter a room.

Inputs:

Program Mode (2-bit): There are four different access modes: 1) Allow access at all times. 2) Allow access during the day 7am-12pm, and allow people with credentials at anytime. 3) Allow access during the weekdays 7am-12pm, and allow people with credentials at anytime. 4) Only allow access to credentials at any time.

Program Mode Enable (1-bit): Will allow the mode to programmed when high.

Program those with credentials (24-bit): The number that needs to be stored into memory as a student ID with credentials to get into the room. Each number can only be entered once into this memory so a repeat should have no effect.

Program those with credentials enable (1-bit): Will allow a new number to be entered when high.

Remove those with credentials (24-bit): The number that needs to be taken out of memory as a student ID with credentials to get into the room.

Remove those with credentials enable (1-bit): Will allow an existing number to be deleted when high.

Return Memory Enable (1-bit): When high will allow the stored ID numbers stored in memory to be cycled through.

Card Swipe (24-bit): This will enter the 7-digit university ID number that will be able to be used to check if the student has access by checking the internal memory to see if there is a match.

Door Open/Closed (1-bit): Reads if the door is open or shut.

Time: An input that displays the current hour in 24-hour notation.

Day: An input that displays the current day of the week.

Outputs:

Green LED (1-bit): Shines bright for the five second period that the door is unlocked for if the credentials are accepted.

Red LED (1-bit): Shines bright for five seconds if the credentials are denied.

Alarm (1-bit): Goes high if the door is opened when there is no access.

Memory Output (24-bit): When Memory Enable is high, this output will cycle through all of the memory one entry at a time.

Access (1-bit): Goes high when access is allowed whether by default time or by a successful credential.

Project 5 - Four-Way Traffic Light Controller

In the traffic light system (Figure 1), an intersection has four ways, **WAY 1, WAY2, WAY 3, and WAY 4.** For each way there are four traffic lights (**red, yellow, green and left-turning**). **Assume that red light holds 60 seconds, yellow light holds 5 seconds, green light holds 40 seconds, and left-turning light holds 10 seconds**. In general condition, the traffic lights are controlled automatically, but sometimes they are also needed manually to control.

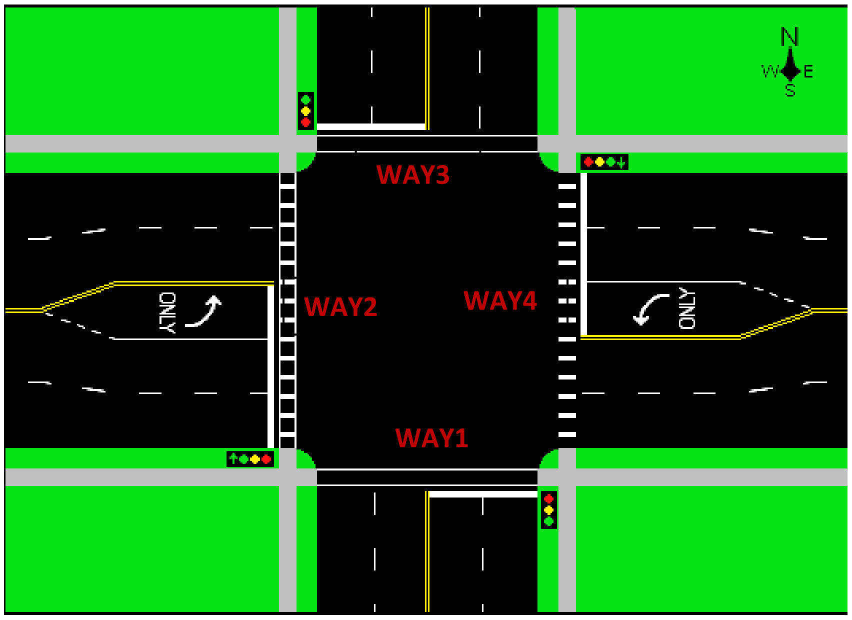


Figure 1: 4-way Intersection

Design Requirement

In this project, it is required to consider all four directions. There are two modes for the traffic light controller system. Mode 1 is automatic operation with the timing given above. Mode 2 is manual control during emergency conditions, road work, and when events change normal traffic flow. In manual mode, there are five combinations of operation (Table 1) that can be set at traffic control. In automatic mode, these combinations cycle as makes sense. A system reset is also included in case something goes wrong and the system needs a reboot.

Combination	WAY 1/3	WAY 2/4
1	Green light	Red light
2	Red light	Green light
3	Red light	Left-turning light ONLY
4	Red light	Left light/Green on 2 – Red for 4
5	Red light	Left light/Green on 4 – Red for 2

Table 1: Working Modes of traffic light controller

Assume the mode (auto or manual) are controlled by a switch and another 3-bit input controls the manual mode. When in auto, this 3-bit input does nothing.

Further, pedestrians can alter the traffic lights. At each of the four corners, there are 2 buttons that can be pressed by pedestrians to request to cross the road. Once pressed for a given direction, the traffic controller must not let the pedestrian stand for more than 60 seconds without being given a clear path to cross. Traffic lights should change, if necessary, to allow the pedestrian to cross the road. This assumes the motorists are also paying attention. For example, a pedestrian can cross the road in front of turning traffic as all motorists should be clearly watching for this as part of their duty as drivers. But pedestrians cannot cross in front of straight, green-light traffic. Just watch a busy intersection for a few cycles if any of this is confusing. Then you will appreciate the logic that goes into these controllers.

Finally, the outputs are the lights (14 in total) plus the 8 pedestrian crosswalk indicators. We can assume that unlike in real life, the pedestrian indicators are either green for WALK or not lit for NO WALK.

Project 6 - Autonomous Car – Emergency Hardware Control

Google and many other private companies have been developing vehicles that are driven autonomously, without human control. While the algorithms and sensors to control these vehicles are complicated and take years to develop, in this project, we will create a simple algorithm that is used when the software fails. This algorithm will be coded into hardware so that even when the main software fails, this backup emergency system will keep the car on the road until it can safely exit traffic and be repaired.

In the world this car moves in, there are no cars. It is a road at a facility created for testing the algorithm. If this algorithm works, further additions will be made to make the car safe in the real world. The road is also fairly straight. The car does not take any sharp turns after one of the main systems fails.

Assume all points on the road have lines that mark the edges of the lane, even when turning.

There are 3 tests that the hardware must pass to be considered a success.

1. Driving on a highway when the software fails or when the GPS and/or cameras lose signal. While the signal is lost, the car is to remain at the speed limit in the lane where the car is currently.
    a. To avoid colliding with cars in the lanes, a car without software, GPS, or cameras must stay in the same lane and use line sensors to stay within the lane. There is one sensor on each side of the car mounted near the headlights that has built-in line sensing hardware. The sensors return a 1 when a painted line in the road cross directly in front of the sensor from the right. When the line comes from the left, a 2 is returned. Otherwise, a zero output indicates that the car's position to a line is mostly unchanged.
    b. If one line sensor switches from 0, the car is accidently swerving into another lane. Depending on the output, the correct logic should turn the wheel for 2 seconds at a 45 degree angle from vertical in the correct direction to stay in the same lane. Hopefully, the lanes are not so tight that this results in over-correction. For now, let's assume that is not the case. After correction, resume heading by turning the wheel to normal position.
2. Driving in a city setting, GPS is often lost around skyscrapers and under tunnels. Sensors and cameras are used to drive the car in these situations. Software is used to direct the vehicle using these cameras and sensors. When the software fails or the cameras lose sight, a hardware backup is needed based on the sensors alone.
    a. Unlike on the highway, in the city there is too much going on to make correct decisions without cameras or GPS. Thus, if in the city, the car should slow to a stop immediately wherever that may be. This may disrupt traffic temporarily but at least other motorists can avoid a stationary car instead of being hit by an out of control one. Slow the car by progressively increasing the brakes from 0% to 100% over 5 seconds. Hold brake at 100% until GPS/cameras return or the user takes over.

3. If even the backup sensors lose signal, the car is effectively blind. The car will almost certainly crash unless it stops or a human takes control. A loud alarm shall sound in the car so that the human occupants can take control. After 15 seconds, if the sensors do not resume operation, the car will be slowed to a stop no matter where it sits, even if on the highway. This may not seem very intelligent, but neither is driving blind at 70mph even without other cars on the road.

The future of autonomous vehicles is scary if they don't have the required data. Try to add any additional intelligence to avoid chaos when the car is without GPS, cameras, or line sensors.

Inputs:

- (2) Line sensors (2-bit each)
- Camera Fail indicator
- GPS no signal indicator
- Software interrupt (software fail) indicator
- City vs. Highway mode (based on speed and previous GPS data). Assume this comes from the GSP and is in one of two states.

Outputs:

- Brakes (assume all four wheels controlled by another system) – Assume a 4-bit number scaled linearly with 0000 as 0% brakes and 1111 as 100% brakes.
- Accelerator (4-bit) – similar to brakes with linear scale – Full gas provides acceleration of 10mph/s (one second the car increases speed by 30mph). (This isn't a Lamborghini by any means). At top end, 1111 will get to 80mph. So, 0111 will accelerate the car at 5mph/s and the car will go 40mph (steady) if held at that value over time.
    - If held at the same value, the acceleration is zero, meaning constant speed.
    - Don't get too involved with the math, just assume full acceleration when speeding up, no acceleration when braking and constant when staying the same speed.
- Wheel – Assume -90 degress to +90 degrees rotation (no major turns here) – this is scaled by an 8-bit number.
- Alarm for requesting human intervention

Project 7 – Digital Alarm Clock

College students dread their alarm clocks. They signal the arrival of a new day, but also that productivity must resume, regardless of how much sleep was attained in the previous hours. In this project, design an alarm clock that is unable to be “tricked” into being shut off by students looking to fall back asleep. Many of us do this, only to miss a class, but this alarm clock will not let that happen. By receiving inputs from a portable EEG machine, this alarm clock can always monitor the brain waves of a student and thus cannot be fooled into thinking a sleep-deprived insomniac is awake and showering instead of dreaming of post-degree paychecks.

Requirements:

1. The alarm time is programmed by the user. The time is input as typical of most digital alarm clocks with the buttons on the device.
2. The time is established by the user as well. Assume when power goes out that there is a battery backup and data is not lost (don’t you hate when that happens and you have to reset everything)
3. When the time matches the alarm time, an alarm is sounded until being turned OFF or SNOOZE is pressed.
4. SNOOZE delays the alarm for 10 minutes at which the alarm sounds again.
5. OFF is not a button as is typical on most alarms. Instead it is logic based on brain waves from EEG. If the brain waves show a return to sleep, the alarm will sound 10 minutes after that point. Once the EEG shows the brain has been active for 10 minutes, then the alarm will shut OFF.
6. Finally, design logic for waking the sleeping person during REM sleep instead of deep sleep to provide the best waking experience. This may mean overriding the user’s pre-programmed alarm. Maybe the user can select a mode for alarm operation: (1) Exact time (2) Within 1 hour based on sleep cycle (3) After 8+ hours with waking at next REM cycle. An alarm like this would be loved by many doctors and students looking to reduce morning grogginess.

Assume the EEG is not a bunch of annoying wires on the head, but a clever wrist-worn device that cannot be accidently removed by an unconscious sleepy head.

Inputs:

- EEG output (2-bit) – brain waves can be detected as REM sleep, deep sleep, or active (awake). Many other patterns are possible but these are the only outputs we care to detect. Poor signal is the fourth option where a pattern cannot be determined
- Alarm Program enable – allows alarm to be programmed
- Time program enable – allows time to be programmed
- 10kHz oscillator – to get this into usable periods, frequency dividers are necessary
- UP button – for adjusting hours/min of time or alarm up one count
- DOWN button – same as above but down
- SELECT button – Accepts value and stores if in program mode. If not in a program mode, this button is used as snooze, which delays the next alarm for 10 minutes, ignoring EEG signals.

When programming, hours are selected first, and once SELECT is pressed, the minutes can be altered. A second press of SELECT resumes operation as normal.

- SNOOZE – The best button on any alarm. When pressed, the alarm is delayed 10 minutes from when it last went off. When pressed prior to the alarm going off (same 12 hour span), the initial alarm is delayed by 10 minutes.
- MODE – for the REM logic that is devised by the designer

Outputs:

- Clock display common of digital clocks including hours and minutes. Assume 24 hour military clock.
- Alarm – hooked to whatever annoying device will get the person awake