Drug Dispenser

Teaching Assistant: Alhassan

Team Member: Abhinav Rajaseshan (1001095469)

Project URL: https://github.com/Abhiseshan/Drug-Dispenser

Introduction

For the final project of the course, I designed a machine that dispenses medicinal drugs (capsules) at preprogrammed times during a day. The machine was designed to aid elderly people who tend to take several medications through the course of the day. Keeping track of which medication to take when can be stressful for an elderly person without any help. Hence, the machine can be set up by a professional and can be used and reprogrammed anytime without troubling the user.

The current Dispenser model has 8 theoretical dispenser modules and can be upgraded to 16 or 32 with slight modifications to the code. The proof of concept dispenser designed for the course has one physical and one virtual dispenser modules enabled.

The physical dispenser is connected through the GPIO Port (GPIO_0) and uses the supply (5V) and ground (GND) from the expansion header. LEDRs were used for the virtual dispenser. The VGA out from the DE1 board was designed to display the menu and on screen instructions for different systems. When connected to a speaker, the DE1 board outputs a beep alarm during dispensing. The design was implemented using Altera Quartus II and Verilog.

Objectives

- Program a functioning clock with support for multiple alarms and the ability to set time.
- Design virtual dispensers to switch on LED at 8 am, 1 pm and 8 pm (Morning, Afternoon and Evening)
- Build physical dispenser and feed it output of virtual dispenser 1.
- Manually override to dispense the capsules on button press.
- Design UI/UX of VGA. Should be easily comprehensible by an elderly person.
- Ability to set/reset dispensers during runtime.
- Measure the quantity of tablets in a module and warn the user if medications are running low.

Tools Used

- Altera Quartus II and Model Sim – For Verilog Programming and Debugging respectively
- SolidWorks 2015 – Designing the Dispenser
- Adobe Photoshop – Designing VGA Graphics

Instructions to Operate the Machine

- Flip Switch 9 to set time. Use KEY's 3, 2 and 1 to set hours, minutes and seconds respectively.
- Flip Switch 8 to setup Dispensers. Use Switches 2 – 0 to set dispenser and 5 – 3 to set the enable signals of the dispensers
- Flip Switch 7 to go to manual mode. Switches 3 – 0 selects the dispenser and KEY 3 activates the selected dispenser
- Flip Switch 6 for about screen
- KEY 0 is reset.

The Design

Verilog and logic

The below block diagram illustrates the detailed design of the circuit.

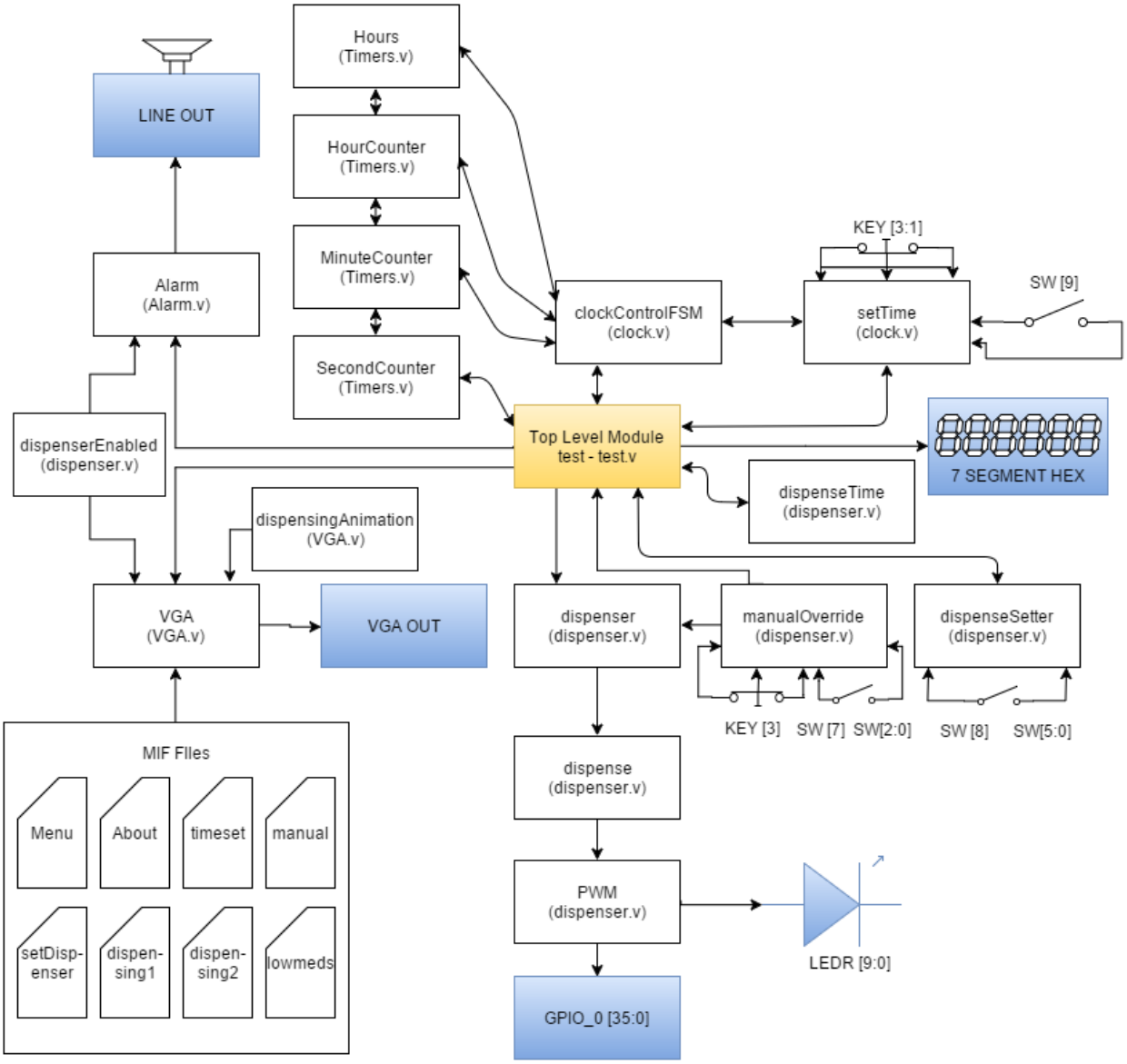


Fig 1. BLOCK Diagram (Outputs are colored in BLUE)

The project was split into multiple files based on the function they perform. These files are listed below (in alphabetical order) along with descriptions of important modules.

- **ALARM.v**

  The contents of this file describe the code in Verilog to output beeps via the audio controller on the Altera DE1 Cyclone V FPGA Board.

- **CLOCK.v**

  `module clockControlFSM();`

  This module is a FSM responsible to control the HEX Display. The different states defined in the FSM are clockMode, preSetMode, setMode, updateMode and resetMode. These states describe the different functions of the clock in the circuit. The default mode is clockMode. When switch 9 is flipped, a pulse is sent in the preSetMode state to store the current time in a register. The next state, setMode allows the user to set the time using Keys 3 to 1. When switch 9 returns to low state, another pulse is sent out but this time sets the values of the counters to the new values and the clock resumes to function from the new values.

  `module setTime();`

  This module is responsible for when the clock is in setMode. The module increments the hours, minutes and seconds when Keys 3 to 1 are pressed respectively. A keypress filter is placed to control key input, that is an input it taken only after certain time. This is useful to increment by one on each button press and to control the increment speed when the button is pressed continuously.

- **DISPENSER.v**

  ~~`module dispenseControlFSM();`~~ **(DEPRECARED)**

  This module was designed to control the setup and execution of the dispensers. The module has been deprecated and features implemented in the modules dispenseTime(), dispenseSetter() and manualOverride().

  `module dispenseTime();`

  Responsible for sending out a pulse at morning (8 am), afternoon (1 pm) and evening (8 pm).

  `module dispenseSetter();`

  Module that gets invoked during dispenser setup. Sets the selected dispenser (selected on Switches 2 to 0) to dispense at morning, afternoon or evening based on the input from switches 5 to 3 respectively.

  `module manualOverride();`

  Module controls the manual override mode to enable the dispensers. The dispenser is selected based on switches 2 to 0 and Key 3 is used to activate the dispenser. Dispenser is stopped when Key 7 goes low.

  `module dispenser();`

  Checks weather the selected dispenser should be active when the morning, afternoon or evening pulse is sent and activates it if so.

  `module dispenserEnabled();`

  When a dispenser is on, the module controls the 5 second timer for the warning screen and alarm to function.

- **TEST.v**

  The file houses the primary module of the design. Responsible for managing the circuit and connects different modules and FSM's together.

- **TIMERS.v**

  This file mainly contains timers used to control the clock. These timers have synchronous set and resets for setting the time and resetting the system.

- **VGA.v**

```
module VGA();
```

  The module is responsible to load the different images from different files depending on the current state of the machine. Also switches between images depending on the output of dispensingAnimation().

```
module dispensingAnimation();
```

  Outputs an alternating signal to control the warning animation every $1/10^{th}$ of a second.

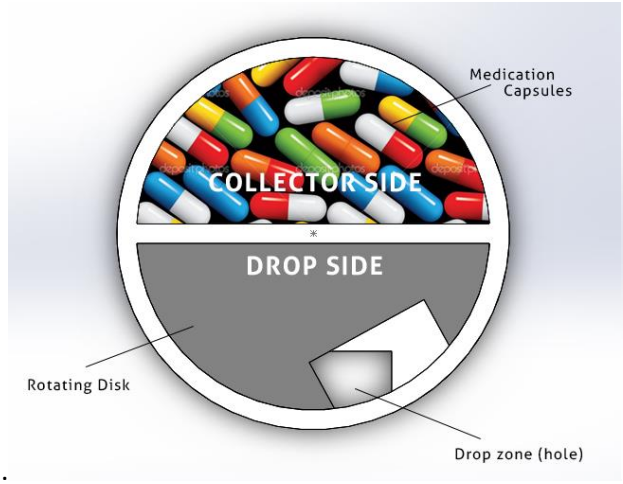
The Dispenser

The proof of concept dispenser was modelled after a candy dispenser. When the dispenser receives a 1 or a Pulse width modulated signal from the FPGA, the motor powering the dispenser activates. This spins the gears which in turn rotates the rotator plate. The plate collects a tablet from the collector side and crosses into the drop side and drops the tablet when it aligns with the hole and stops. When the dispenser receives the next on signal, it repeats the process.

An IR sensor placed at a tablet layer high sends a signal to the FPGA when it does not detect the presence of a tablet warning the user that he/she is running low on medications.

The dispenser was constructed by machining structural foam using a CNC (computer numerical control) router [Appendix B] and designed using SolidWorks [Appendix A].

The circuit diagram is available in the Appendix C.



.

Fig 2. Top View of Dispenser

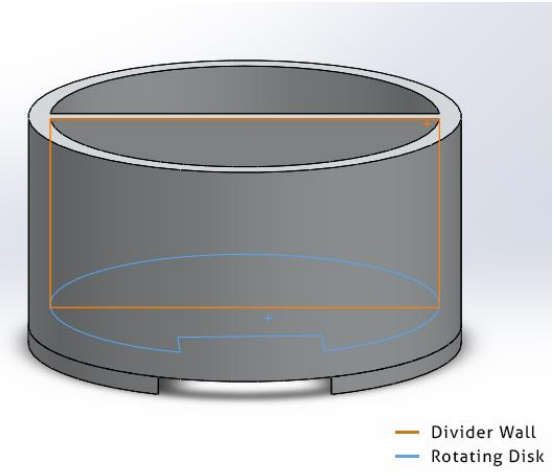


Fig 3. Iso Side View

Report on Success

Did it work?

The FPGA implementation worked as expected but the physical dispenser failed to rotate and dispense. Since the physical dispenser failed to rotate, the calibration of the rotating disk could not be completed and was set to a default value. The planned IR sensor was not completed by the end of the project. It did not play a significant part in the overall working but if an IR sensor was attached to the GPIO_0[1], it would send a signal to the VGA displaying a medication low graphic (this has been coded for).

Analyzing what went wrong, the disk was unable to rotate due to unevenness of the wall's bottom surface. This surface provided enough friction to overcome the torque produced by the motor. On adjusting the disks position, free motion of the disk was observed but maintaining that position was impossible due to the centrifugal force which shook the disk. Also during this free motion, it was observed that the disk spun at a very high speed which was undesired. Pulse Width Modulation was to be implemented to control the speed of the motor by sending pulses rather than a constant 1.

It was also observed that the friction would cause the teeth of the motor to slip from the teeth of the gears, causing a high screeching noise. The transistor would get extremely hot in the time difference from when the motor turns on and the motor slips from the gear.

Images available in Appendix D.

What would I do differently?

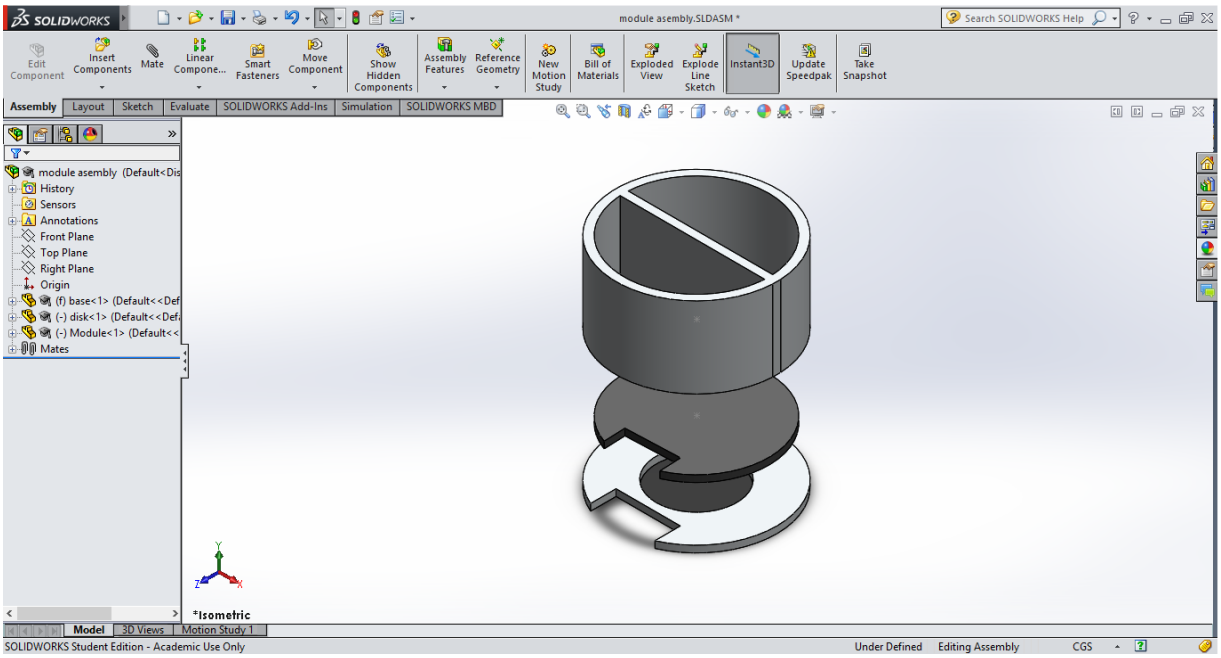
Overall the project quality in terms of the FPGA output turned out well. The hardware on the other hand turned out to look nice but did not function correctly. I would spend more time from designing and testing the dispenser to make sure I have a working prototype. Also I would choose a different motor with a lower RPM and a bigger gearbox to slow down the rotation of the disk.

The materials for the disk and the dispenser were chosen poorly and I would choose something with lesser friction and would work with a resting design instead of lighter and a suspended design which I implemented. I would also lubricate the disk to ensure smooth motion.

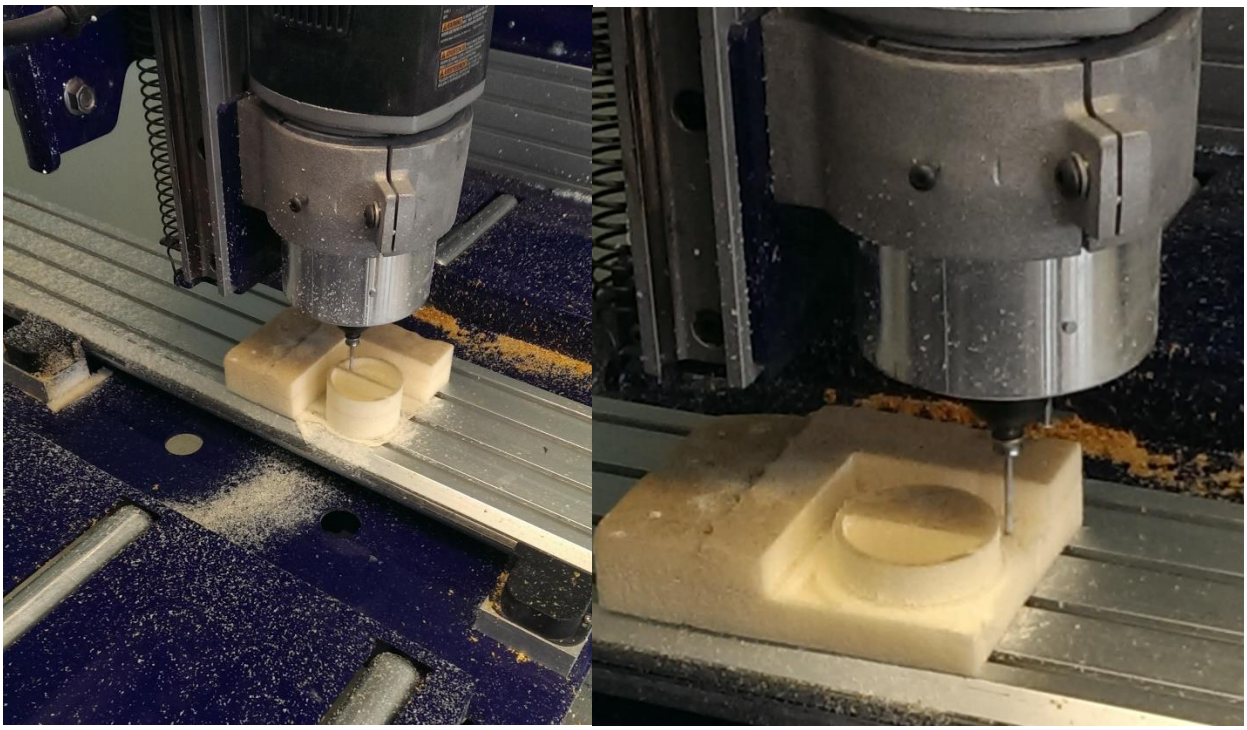
On the FPGA side, I was not quite happy with the VGA graphics and would have used 640 x 480 instead of 160 x 120 and higher bit color to produce better graphics and reduce pixilation. I would also buy an IR sensor to enable the Low Medication Alert function of the machine.

APPENDIX

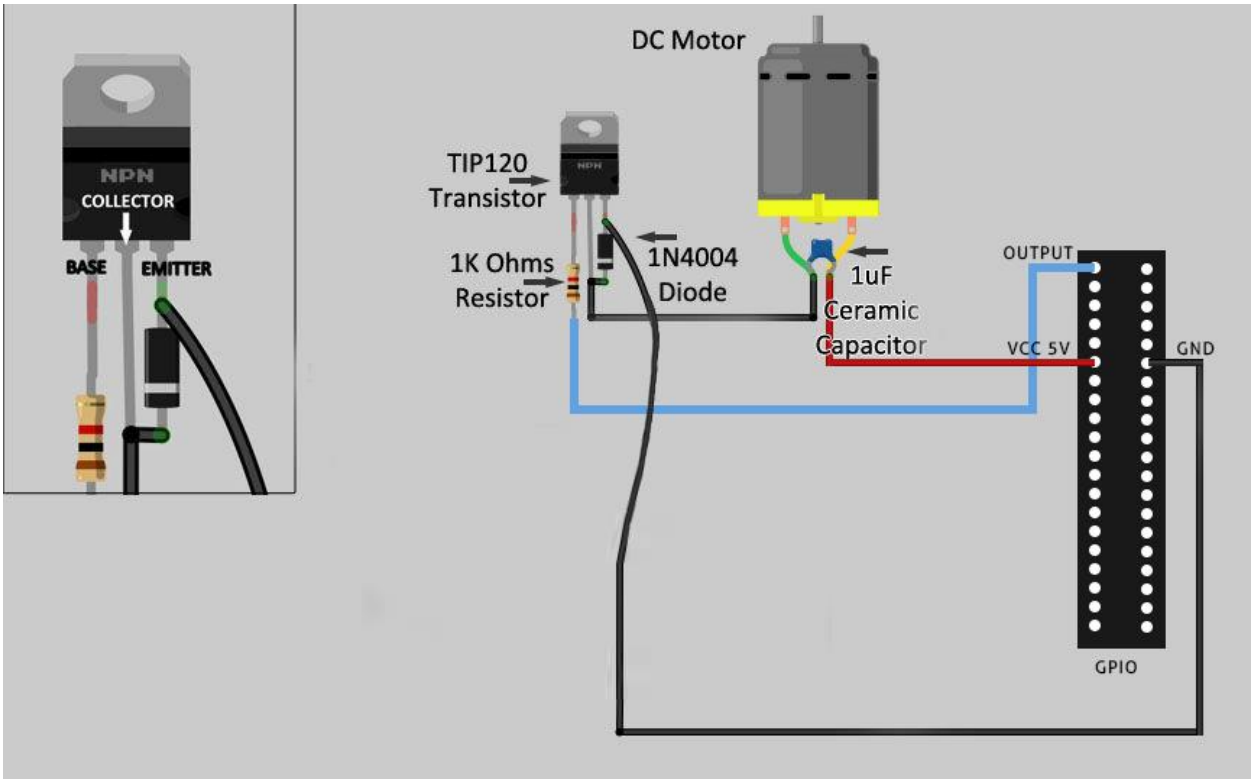
Appendix A: Designing the dispenser on SolidWorks.



Appendix B: CNC router cutting through structural foam to manufacture dispenser module.

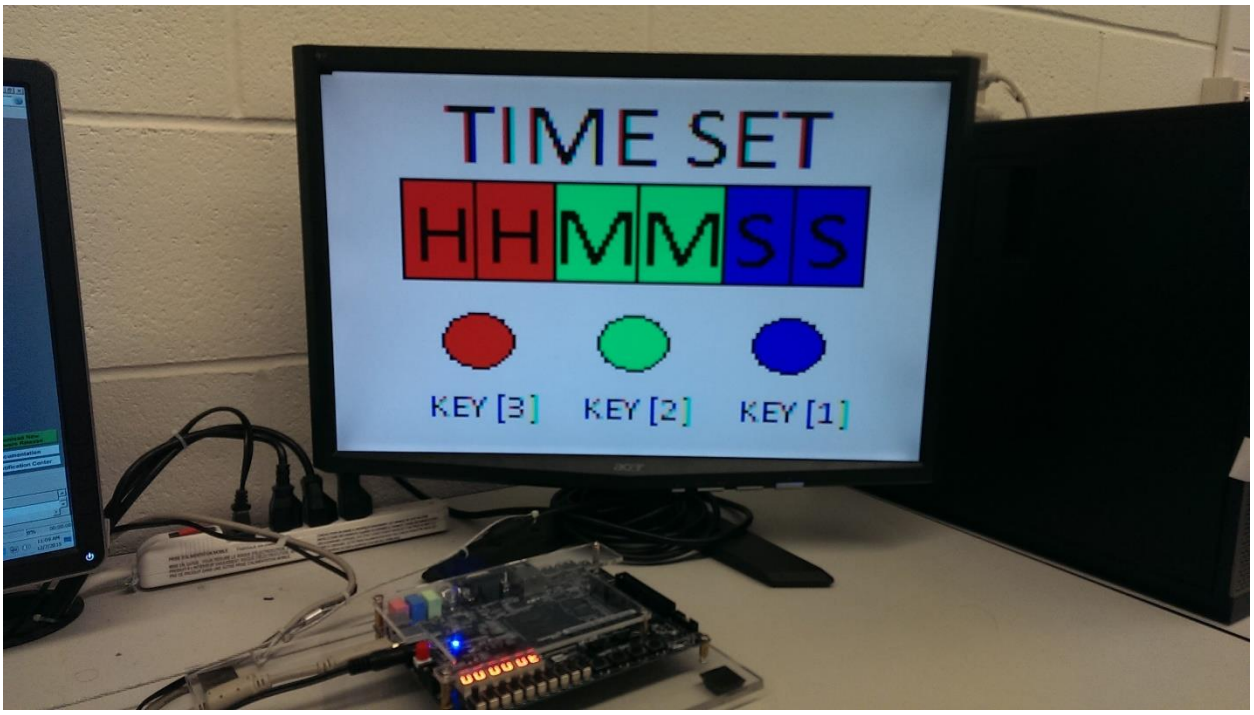


Appendix C: Circuit Diagram

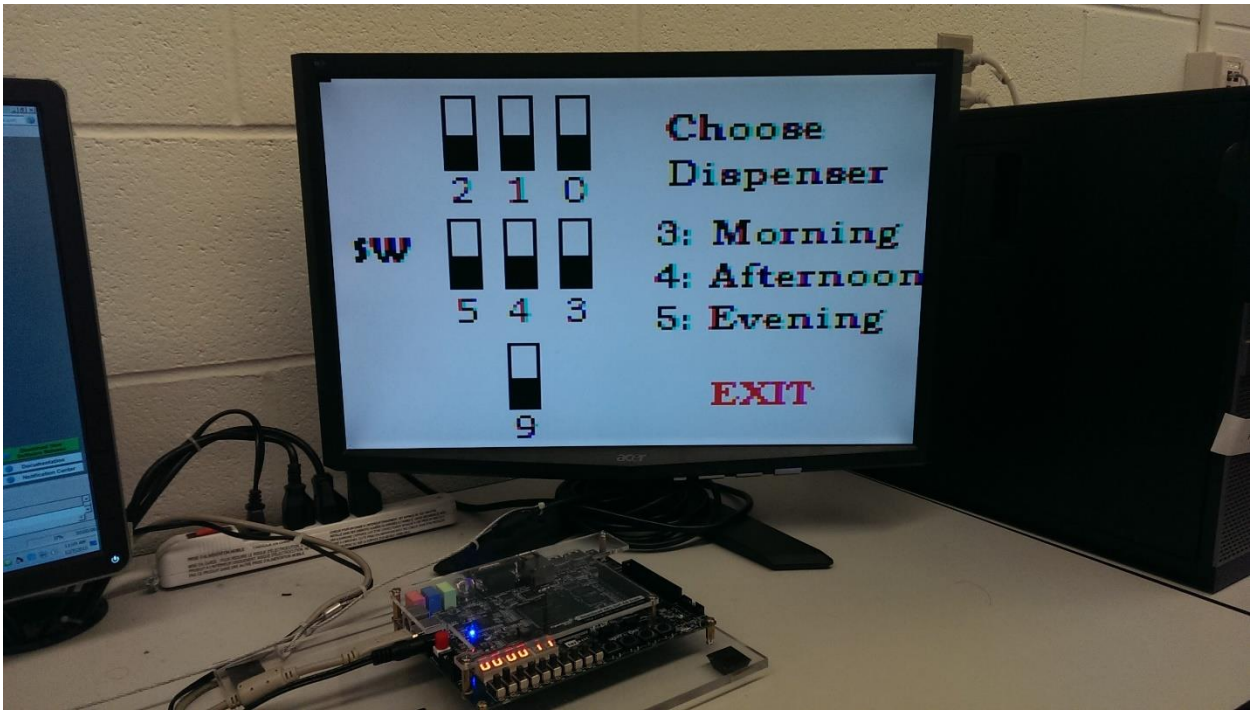


Appendix D: Final Project Demo

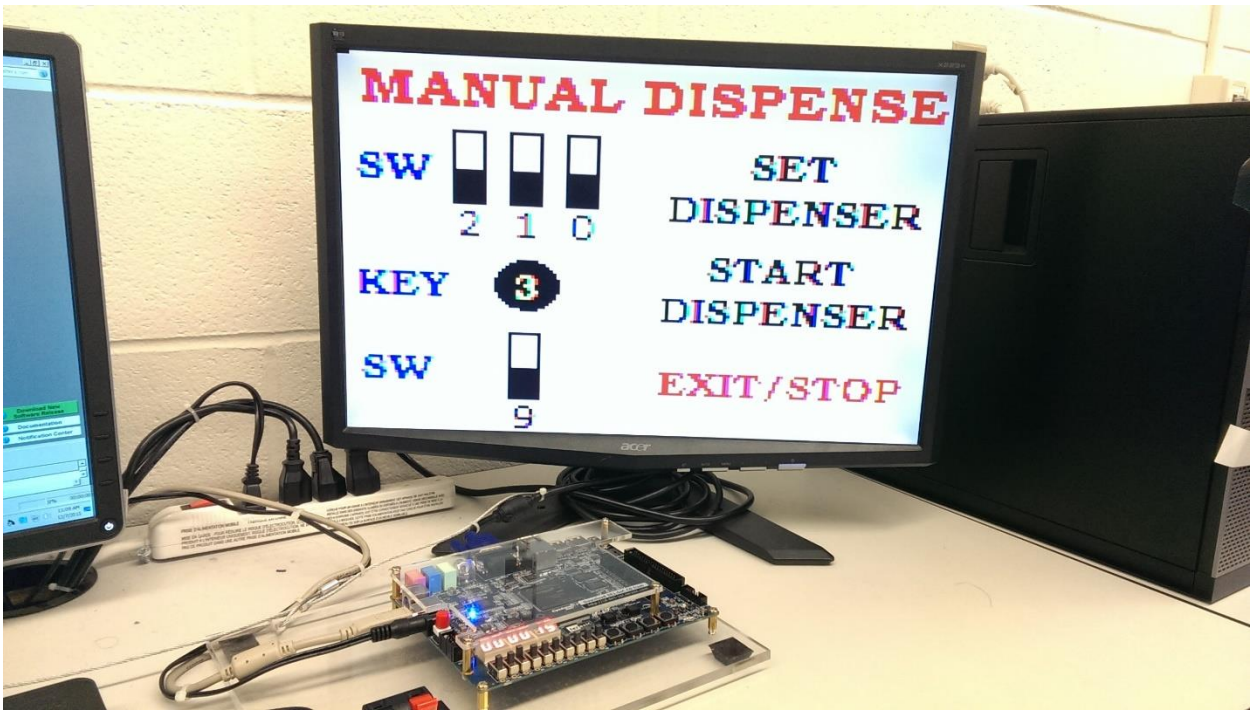
Main Menu



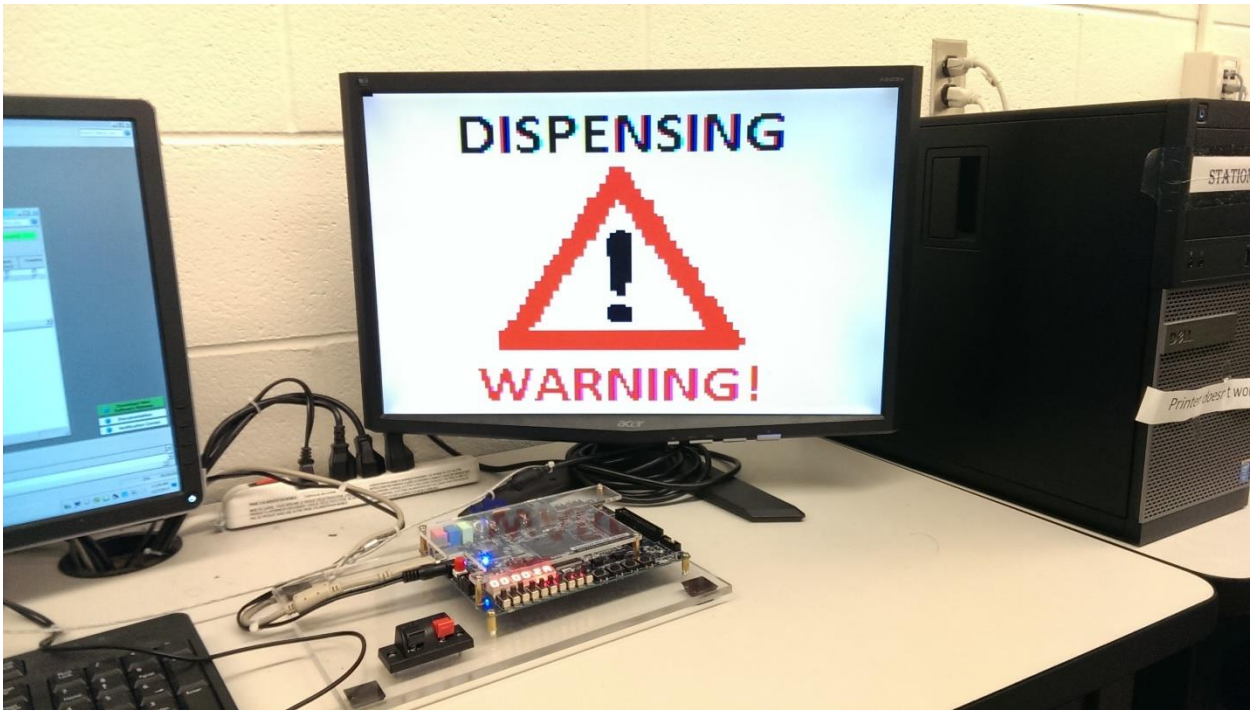
Setting the Time SW [9]



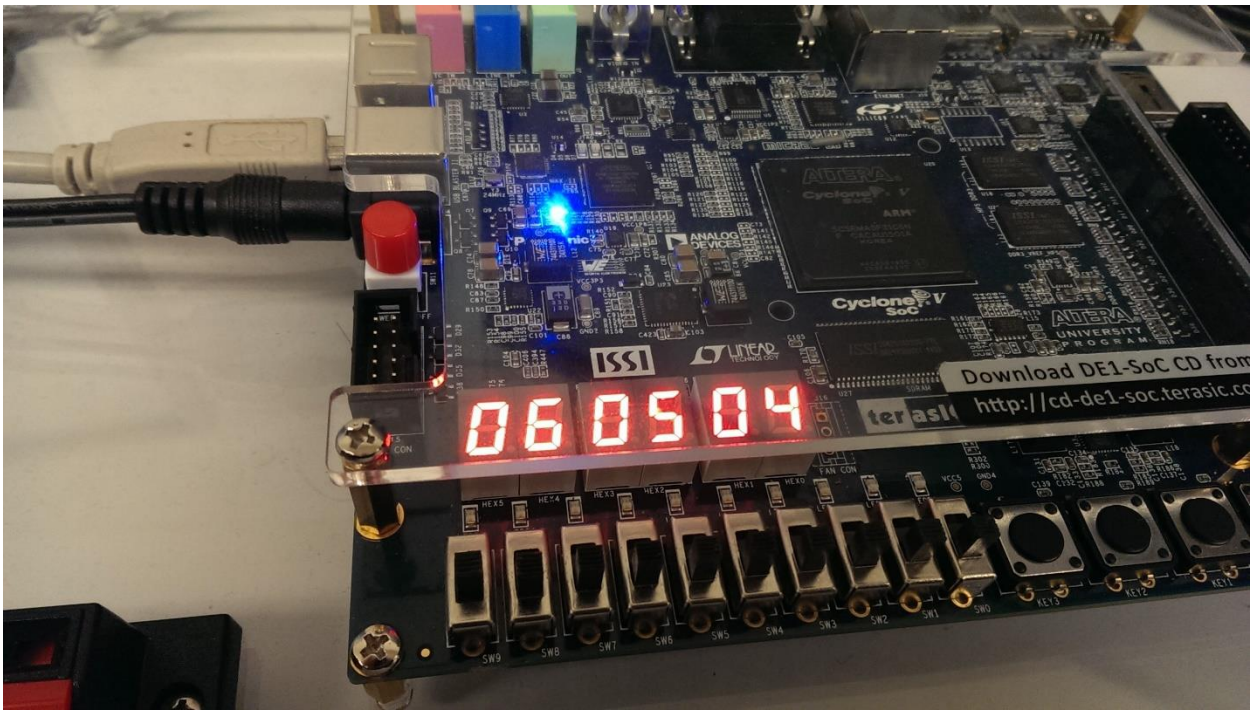
Setting up dispensers SW [8]



Manual Dispense Mode SW [7]



Dispenser Dispensing Animation



Clock Display