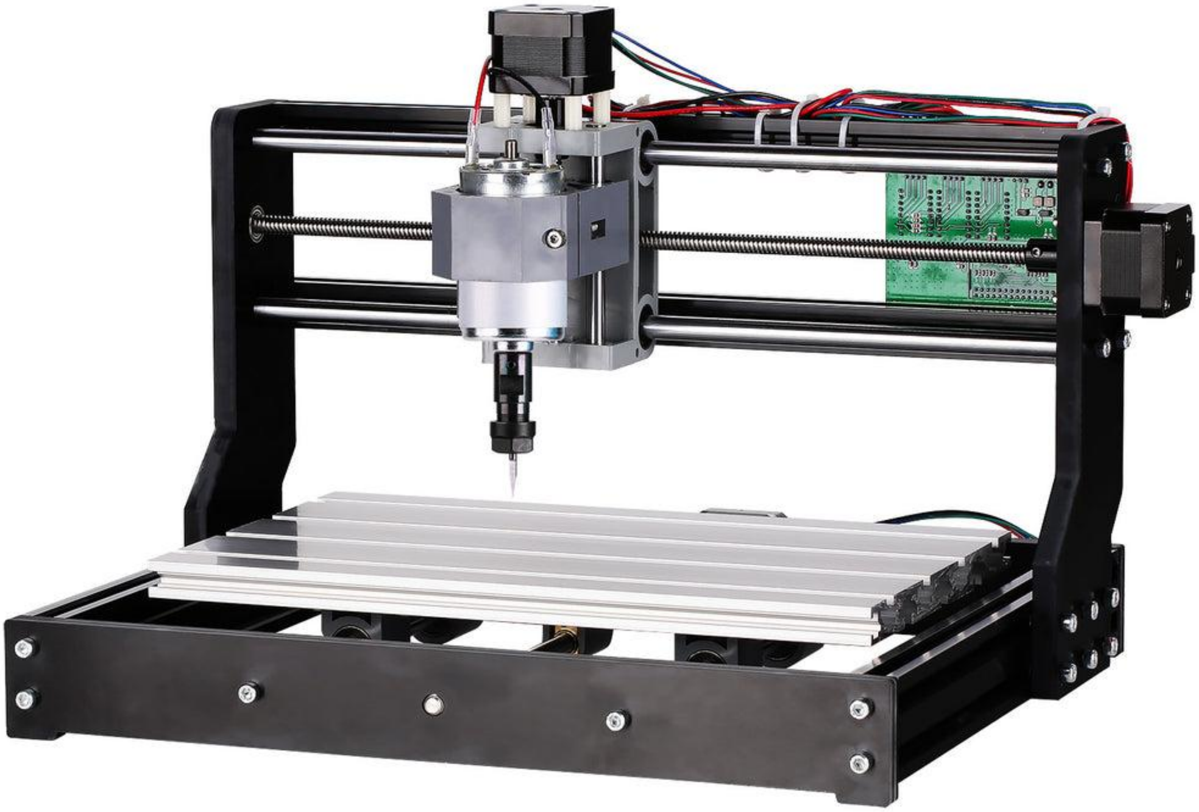


https://www.sainsmart.com/products/sainsmart-genmitsu-cnc-router-3018-pro-diy-kit

ECE 544 Final Project Report
Manual CNC/Hot Wire Cutting Machine

Team members
Alex Beaulier, Josiah Sweeney

Introduction

The Manual CNC/Soldering and Cutting machine is a three axis driven machine with a tool for manufacturing purposes. Each axis is driven by a stepper motor for full XYZ motion for an end of arm tooling(EOAT) to be used. This tooling could potentially be a router, drill bit, hot wire cutter or another actuator for any purpose. The goal of this project is to implement the three axis movement and explore two new pmods. The machine will feature an OLED displaying information such as user selected traversal speed, XYZ coordinates and other useful information. There shall be a PMOD Joystick for manual movement in XY directions. Switches for turning on the EOAT such as a heating element or cutter which would be wired to a relay also integrated on a PMOD port. Switches can also emergency stop the platform and reset the relative XYZ coordinates. The project will also use FreeRTOS for real time control of a motor application.

Stretch goals include Android IOT communication, a z-axis sensor such as conductive touch sensor for calibration, emergency stop sensors on each axis, CSV file control for true CNC like capability.

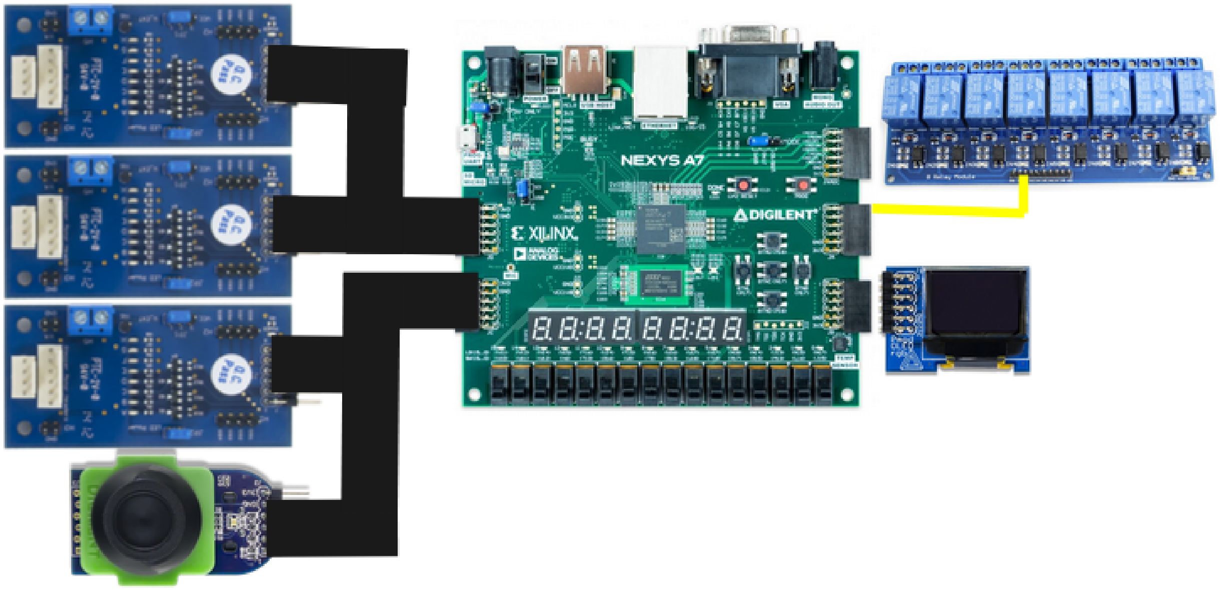


Figure 1: **Basic Hardware Layout**

Hardware

For this project, we leveraged the PmodOLED IP that was utilized in Project 1 and Project 3, while adding several new hardware components. In addition to using the PmodOLED, we added in 3 PmodSTEP devices along with a PmodJSTK and set up one Pmod port as a relay port. Since we worked in RTOS, we kept the watchdog timer from Project 3 along with the axi_timer interrupt so that our FreeRTOS would continue to function. Additionally, to make the relay port function, a GPIO was added, as well as one gpio for each axis for counting the steps of the stepper motor. As the PMOD step had no IP drivers, this had to be created.

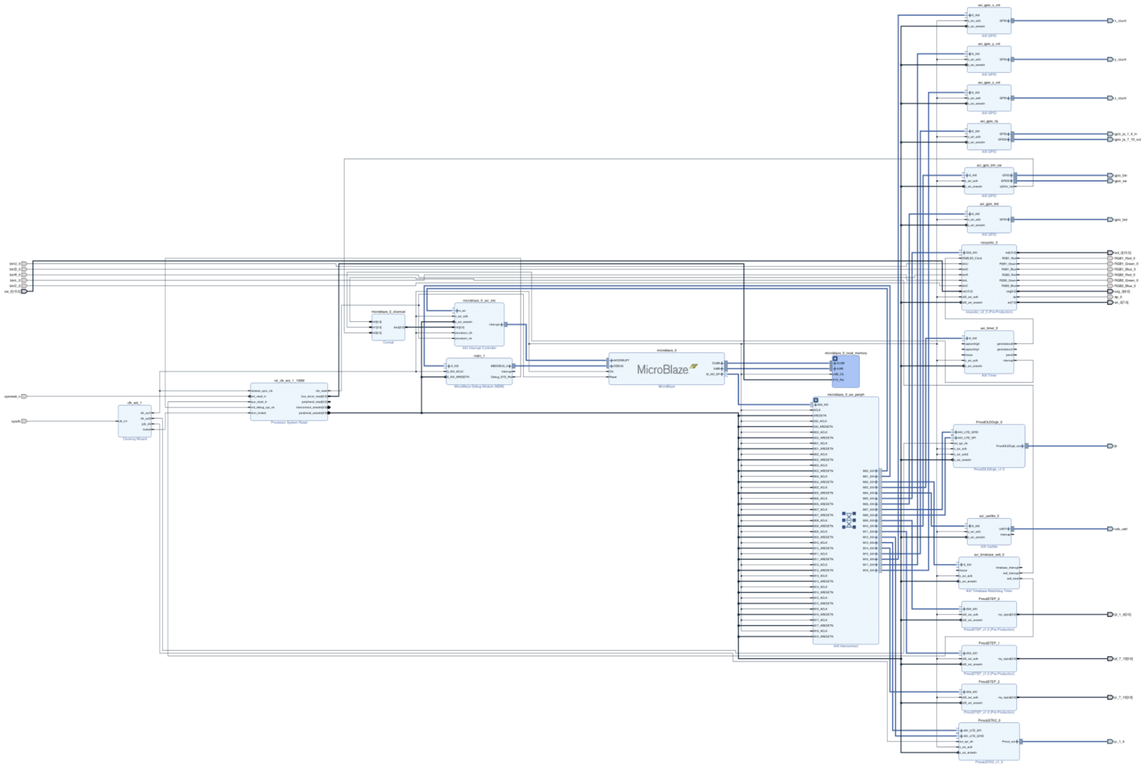
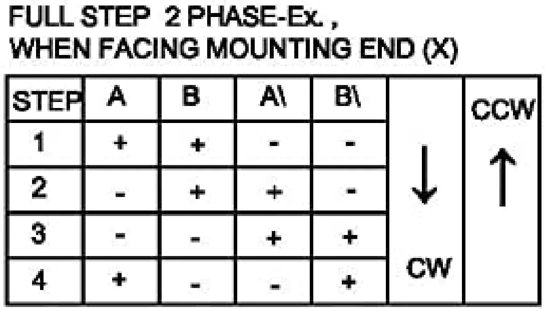


Figure 2: **Embsys Diagram for the Entire System**

IP Creation

In order to step through our motors correctly, we had to write hdl to then turn into IP for the PMOD step.

To step through a bipolar stepper motor on a 4 pin set-up, you have to drive the pins in a certain sequence to charge the coils correctly for clockwise and counter-clockwise direction. In Figure 3, the sequence of what needs to be driven to the stepper motors is seen, as it steps through alternating power to coils in the motor.



FULL STEP 2 PHASE-Ex. ,
WHEN FACING MOUNTING END (X)

STEP	A	B	A\	B\		CCW
1	+	+	-	-	↓	↑
2	-	+	+	-		
3	-	-	+	+		
4	+	-	-	+	CW	

Figure 3: **Bipolar Stepping Sequence**

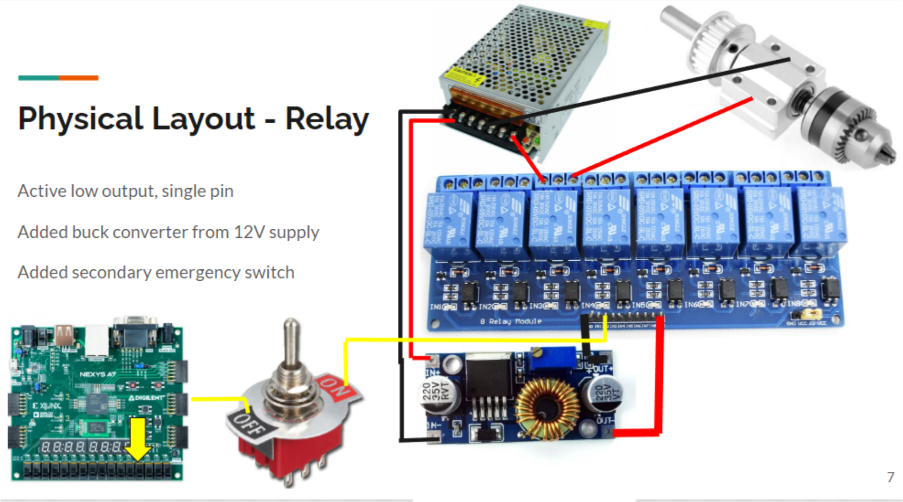
This was repeated in HDL where based on the direction, and whether or not the PMODstep was enabled, you would drive a state to the stepper motor. This is the relevant portion of the HDL code in the driver that was sent to the 4 bit register driving the stepper motors.

```
always @(posedge S_AXI_ACLK)
  begin
        if (S_AXI_ARESETN == 1'b0)  slv_reg3[3:0] <= 4'b0000;
        else
        begin
        case (my_state)
                3'd0: slv_reg3[3:0] <= 4'b0000;
                3'd1: slv_reg3[3:0] <= 4'b0011;
                3'd2: slv_reg3[3:0] <= 4'b0110;
                3'd3: slv_reg3[3:0] <= 4'b1100;
                3'd4: slv_reg3[3:0] <= 4'b1001;
        endcase
        end
  end
```

Additionally to the PmodSTEP, the PmodJSTK had to have its IP edited as it was not compatible with Vivado 2020.2 and the Makefile had to be re-made.

Relay and Power Supply

The rotary tool for the machine operates on a 12V supply. To ensure some level of safety, some minor features were added. This included a relay, initialization of off state to the gpio and a switch for manual control of whether the tool head is enabled. The relay is active low which required a switch as rebooting and initial startup of the NexysA7 would pull the relay low enabling the tool. During the GPIO initialization a signal for the relay is immediately sent after the gpio is created. The relay turns on if switch 4 is high and pushes an active gpio out on pin 7 of the JA pin 7 if switch 4 is low. Figure x below depicts the relay system. A buck converter was also added as the relay uses 5V supply to allow 3v from the Nexys to drive the relay channels.



Physical Layout - Relay

Active low output, single pin

Added buck converter from 12V supply

Added secondary emergency switch

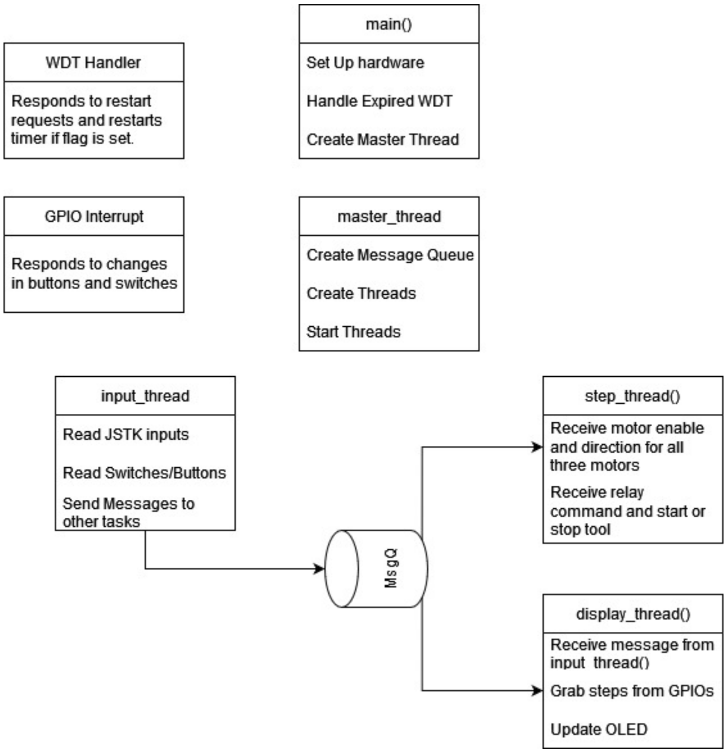
7

Figure 3: **Relay and power supply depiction.**

Software

Threading Model

For this project, we followed a very similar threading model to Project 3, where we had an input thread, a display thread, and a thread that controlled the motors and drove the relay.



WDT Handler
Responds to restart requests and restarts timer if flag is set.
main()
Set Up hardware
Handle Expired WDT
Create Master Thread
GPIO Interrupt
Responds to changes in buttons and switches
master_thread
Create Message Queue
Create Threads
Start Threads
input_thread
Read JSTK inputs
Read Switches/Buttons
Send Messages to other tasks
MsgQ
step_thread()
Receive motor enable and direction for all three motors
Receive relay command and start or stop tool
display_thread()
Receive message from input thread()
Grab steps from GPIOs
Update OLED

Project Deliverables

- Writeup
    - Project overview, circuit function
    - Block diagram of circuit
    - Design details, theory of operation, state transition diagrams
    - Results
    - Contributions
    - 10 Page Limit
- Source Code
    - All HDL files, including test benches
    - Lists of program source code
    - Block Diagram of embedded system, .mss file, .xdc file.
    - Code comments for variables, signals and descriptions

Components

- Nexys A7 100T
- 3 Stepper Motors with Linear Rail system
- Large Power Supply
- 3 PmodStep
- 1 PmodOLEDrgb
- Element Relay
    - Cutter/Heating Tool
- Joystick for XY control
- (Possible Microcontroller iOT)
- (Possible remote application control device)

Tools

- Vivado 2020.2
- Vitis
- (Possibly Atmel)
- (Possibly Android Studio)

Building

The team ordered components immediately. Josiah built the entire CNC machine and Alex built the hardware board layout and pin connectors.

Successful Scope/Results

A successful result shall be capable of driving the machine’s XYZ axis through the joystick, enabling the cutting or heating element. A backup plan would be one axis working with a forwards and backwards direction on the joystick.

The project results went as planned with the machine driving all XYZ axis through the joystick and pushbuttons with a cutting element. A presentation was shown cutting the members' names into a banana using the machine. Emergency stops features were added, the heating tool was not demoed but could be placed in the cutting holder. Traversal speed wasn’t implemented due to issues with the stepper consistency and no encoder feedback but counts were made for assumptions of the driving steps and displayed on the OLED. One other change to the plan was traversal speed as we found only certain frequencies and voltages to have consistent operation. Stepper pmods or peripheral hardware for higher currents are required to allow for various speeds. The team took significant risks and still accomplished a successful project.

Project Responsibilities Actual

Person A - Josiah	**Person B - Alex**
Hardware Stepper motor PMOD Joystick PMOD Buttons E stop Switches Cutter OLED Software Calibration of Steppers FreeRTOS Buttons WDT OLED Switches FreeRTOS Relay Assembly CNC Machine Wiring Debugging System Stepper motor voltage trials	Hardware E stop Cutter PMOD port connections and debugging Stepper motor PMOD tuning modifications Software Stepper case logic Buttons Z axis OLED Switches FreeRTOS Relay Assembly Assembly of machine Wiring Debugging System Stepper tuning Display clearing FreeRTOS changes Switch handling Stepper

Milestones

5/21/22 - Begin Project

- Stepper motor hardware achieved

- Joystick hardware achieved
- Switches implemented
- Basic C program
    - general model made

WEEK 1 END 5/29/22

- System Verilog Hardware fully debugged
- **All Ports Complete for integration**
- **Vivado Integration from both team members to single .xsa file**
- Buttons implemented
- Software started
- Software FreeRTOS
- Project Meeting with Kravitz and Burnette
- Manual movement fully implemented
- Relay implemented
- Heating element implemented
- Cutting tool implemented
- **Vitis fully implemented and integrated between team members**

Week 2 END 6/05/22

- Build presentation
- 6/7/22 - Practice
- 6/8/22 - Project Presentation

Class Ends: **6/9/22 - Project Report and Deliverables**

Project Risks

- Stepper motor 5 or 6 pin compatible to 4 pin? Preorder machine Stepper risk
- Power supply
- Order arrival

Stretch Goals

Z calibrating tool - Proxy sensor - Attempted the ultrasonic sensor integration and found the sensor required 5V input whereas the NexysA7 cannot drive that high. Due to time constraints we couldn't implement another buck converter and develop the communication protocol.

CSV pseudo file control -> Solidworks - There were plans to calculate traversal distance based on step count and make assumptions about distance to then read in a file. The stepper inconsistencies did not allow for us to implement this.

Conductive proximity sensors, ultrasonic, lidar
Android button control(IOT) - Time limited but based on other's projects one could implement the ESP32 through MQTT.
Grid draw, XY coordinates

Miscellaneous Notes:

https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000PAkPSAW