CC2511 Assignment 2 - CNC Design Report

Ashley Mauro

SN: 13861236

Report developed for: Bruce Belson

Due Date: 1/11/2021

PART 1. HOW DOES THE SYSTEM WORK ?

Hardware

The PCB shield has a raspberry pi Pico integrated that has selected GPIO output pins connected to other certain chips on the board. Ten of the GPIO pins are used to connect the DRV8825 stepper motor controller chips, one GPIO pin is used as a PWM for the DC spindle motor, two pins are used as Serial TX and Serial RX for Pico debugging during the software design stage, the RUN pin on the Pico is used as a hard reset switch in case of emergency when under operation, and the VBUS and VSYS ins are used to power the Pico from the DC power supply using the Barrel jack.

Additionally, the Serial TX, Serial RX, VSYS, and GND were integrated into a breakout header so the debugging PICO could be utilised. All the axis' STEP and DIR pins also had an integrated breakout header in case of DRV8825 chip malfunction. In the event of a barrel jack or voltage regulator malfunction, a breakout header for both 18V DC power and 5V DC power were integrated into the board.

The three main stepper motors for each respective axis have their own separate DRV8825 stepper motor driver chip that is connected in a very particular way to operate the stepper motor. The chip itself was powered from the 18V DC power supply with a configuration of various capacitors and resistors in parallel to enable smooth power delivery to the chip. The DIR and STEP pins were connected to a selected GPIO pins to specify the stepper motor direction and step movements. The MODE0, MODE1, and MODE2 pins were connected to GPIO pins so the step size could be altered for a more accurate and finite stepping motion. The nENBL pin was also integrated into the PICO however, this was never used because a low is what allows stepper motor operation. Moreover, both the AVREF and BVREF were connected to their own node with an integrated breakout header so the VREF can be checked for operation. Finally, the AOUT1, AOUT2, BOUT1, and BOUT2 were all connected to a B4B stepper motor connector header to drive the stepper motors.

Regarding the DC motor, a GPIO pin with PWM output was used to integrate the spindle motor. The spindle motor is connected to the PCB by a B2P connector with a schottky diode and capacitor in parallel. This was incorporated to protect the PCB from the flyback current when the spindle motor is stopped or started suddenly. Variation of the duty cycle is used by the user to achieve a set spindle speed. This GPIO output is then used sent through a N-Mosfet with a pulldown resistor to control the spindle motor.

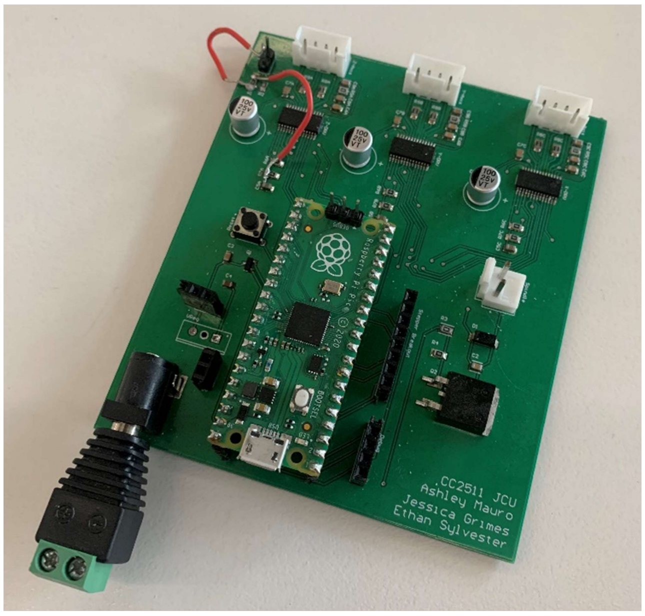


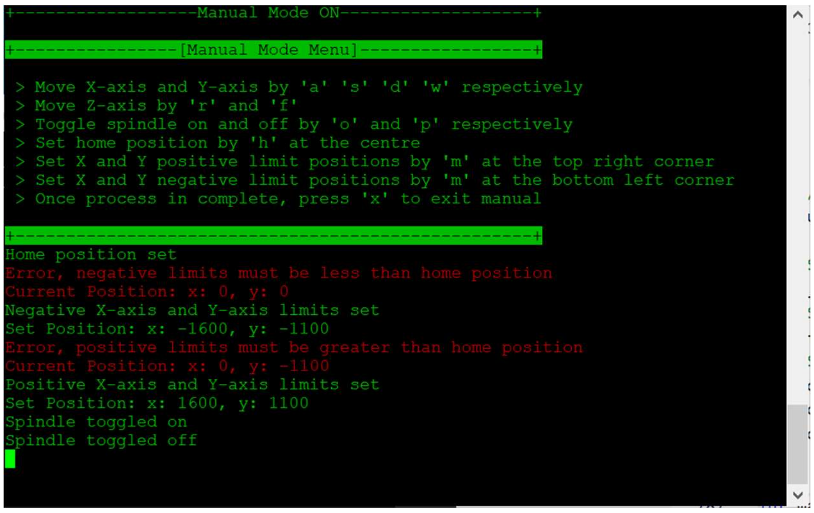
Figure 1: Final PCB shield

Software

Utilising the Raspberry pi Pico's onboard RP2040 microcontroller the software written in C can interpret G-code style coordinates. The software was written for user integration in mind. There were multiple stages of CNC operation including Manual mode, G-Code coordinate mode and automatic G-code coordinate mode. The reason G-code coordinates were chosen was due to the simplicity and user-friendly operation. The main user integration was through a putty terminal for both manual and G-code coordinate modes. Most of the calculating power was done by the micro-processor, whilst also having a UART interrupt handler integrated to handle user commands. The main code that controlled position on the CNC was a position struct. This struct had five sperate instances which were: home position, current position, set position, negative limit position and positive limit position.

Manual Mode

Manual mode was the initial mode the user was exposed to. In this mode the setup of the home position, positive and negative axis limits would be done. These processes were used to create an axis coordinate system of allowable engraving space. Main X, Y, and Z axis movements were used by the A, S, D, W, R, and F keys while the O and P keys were used toggle the spindle on and off. The main setup of the home and axis limits were done by press H which would set [X, Y, Z] = [0, 0, 0] and N and M to set the negative and positive X and Y axis limits. These processes had user errors integrated within these. This made the user specifically set the negative limits in the negative region and the positive limits in the positive region (Fig.2). Finally, after all positions were set the user then presses the X key to exit manual mode to G-code mode.



```
+------------------Manual Mode ON-------------------+

+---------------[Manual Mode Menu]-----------------+

 > Move X-axis and Y-axis by 'a' 's' 'd' 'w' respectively
 > Move Z-axis by 'r' and 'f'
 > Toggle spindle on and off by 'o' and 'p' respectively
 > Set home position by 'h' at the centre
 > Set X and Y positive limit positions by 'm' at the top right corner
 > Set X and Y negative limit positions by 'm' at the bottom left corner
 > Once process in complete, press 'x' to exit manual

+--------------------------------------------------+
Home position set
Error, negative limits must be less than home position
Current Position: x: 0, y: 0
Negative X-axis and Y-axis limits set
Set Position: x: -1600, y: -1100
Error, positive limits must be greater than home position
Current Position: x: 0, y: -1100
Positive X-axis and Y-axis limits set
Set Position: x: 1600, y: 1100
Spindle toggled on
Spindle toggled off
```

Figure 2: Terminal output for Manual mode.

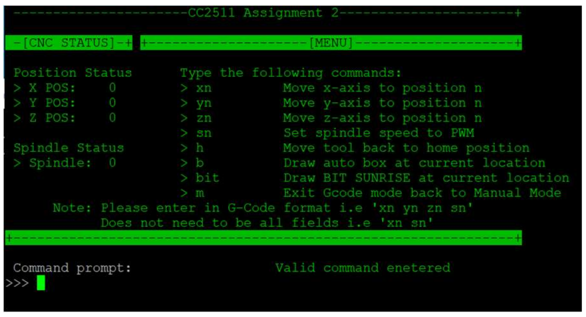
G-Code Mode

G-code mode is where the main movements integrated by the user would occur. G-code mode also had various levels of user error checking to ensure correct operation of the CNC. The user is instructed to send a command for various applications the code can do. Coordinates are sent via a 'xn yn zn sn' format as instruction on the menu (Fig.3). This command is interpreted by the interrupt and put into a buffer which is split up at the spaces by using the strtok function. Each element of the command is then error checked against a string of acceptable inputs and then compared to various commands by either using strcmp to compare strings or sscanf by compared the letter, retrieving the number adjacent and assigning that number to a particular pointer. If the command was not within the acceptable inputs and error occurs on the terminal describing this (Fig.4). Once the pointer is assigned a number, the corresponding set position element is set to that number. This is then looped until all axis elements entered by the user have been interpreted and assigned to their corresponding pointer.

Furthermore, after the set position has been achieved it is then error checked against the positive and negative axis limits either passing or failing and showing an error describing what the issue is (Fig.5 & Fig. 6). If the error checking has been achieved the terminal outputs valid command (Fig.3), the z-axis position is then moved first and then the dynamic_lines function is called with both x and y set point elements passed in as the arguments. Moreover, both the box and bit sunrise functions are integrated to showcase how a G-code coordinate file would be integrated by the code. Constantly feeding it specific coordinates to move to until a desired shape is achieved.

The dynamic_lines function utilises Bresenham's Line algorithm to control offset stepper motor function in any direction, for any length, at any angle. The algorithm determines the largest amount of distance between both axis direction and loops for that period of steps. Conditional based if statements are integrated to determine which axis has the furthest distance to travel at each specific step and is compared to what it actual should be. Once this is achieved the motor with the furthest distance to the actual is stepped by driving a high and low which are separated by two 1000µs sleeps. After the desired set position is achieved by the CNC the current position instance is then updated to the set position instance entered by the user which is displayed back to the user via the terminal.

Additionally, when the home position function has been entered it is conditional based on whether the z-axis is in home position or not. If the z-axis position is lower than the home and error will occur to stop the user from engraving unwanted lines or wrecking the engraving tool (Fig.7). Finally, by selecting m as the command the user can switch back to manual mode to reset limits and home position if needed.



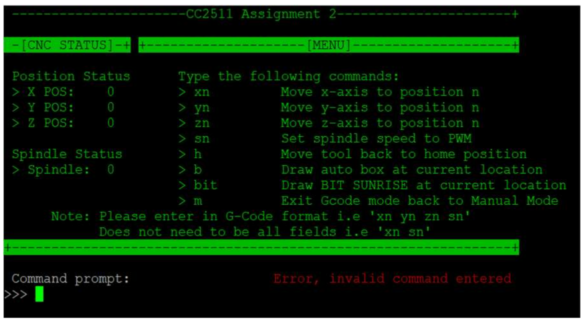
```
----------------------CC2511 Assignment 2----------------------+

-[CNC STATUS]-+ +-------------------[MENU]--------------------+

Position Status      Type the following commands:
> X POS:    0        > xn         Move x-axis to position n
> Y POS:    0        > yn         Move y-axis to position n
> Z POS:    0        > zn         Move z-axis to position n
                     > sn         Set spindle speed to PWM
Spindle Status       > h          Move tool back to home position
> Spindle:  0        > b          Draw auto box at current location
                     > bit        Draw BIT SUNRISE at current location
                     > m          Exit Gcode mode back to Manual Mode
     Note: Please enter in G-Code format i.e 'xn yn zn sn'
           Does not need to be all fields i.e 'xn sn'
+--------------------------------------------------------------+

Command prompt:                  Valid command enetered
>>> 
```

Figure 3: Terminal for valid command



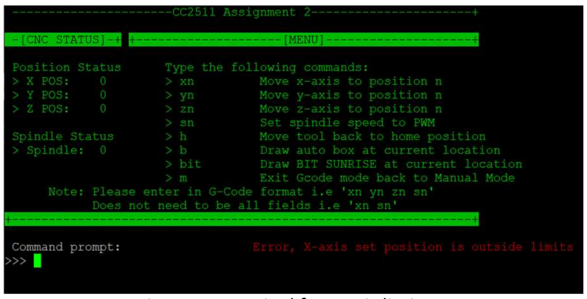
```
----------------------CC2511 Assignment 2----------------------+

-[CNC STATUS]-+ +-------------------[MENU]--------------------+

Position Status      Type the following commands:
> X POS:    0        > xn         Move x-axis to position n
> Y POS:    0        > yn         Move y-axis to position n
> Z POS:    0        > zn         Move z-axis to position n
                     > sn         Set spindle speed to PWM
Spindle Status       > h          Move tool back to home position
> Spindle:  0        > b          Draw auto box at current location
                     > bit        Draw BIT SUNRISE at current location
                     > m          Exit Gcode mode back to Manual Mode
     Note: Please enter in G-Code format i.e 'xn yn zn sn'
           Does not need to be all fields i.e 'xn sn'
+--------------------------------------------------------------+

Command prompt:                  Error, invalid command entered
>>> 
```

Figure 4: Terminal for invalid command



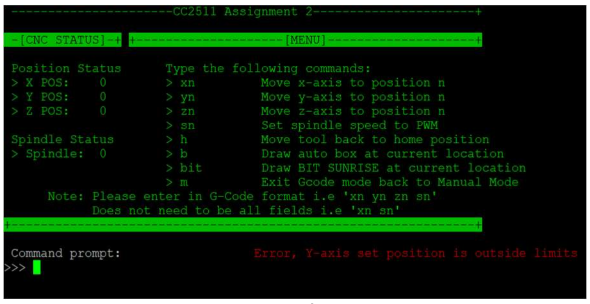
```
----------------------CC2511 Assignment 2----------------------+

-[CNC STATUS]-+ +-------------------[MENU]--------------------+

Position Status      Type the following commands:
> X POS:    0        > xn         Move x-axis to position n
> Y POS:    0        > yn         Move y-axis to position n
> Z POS:    0        > zn         Move z-axis to position n
                     > sn         Set spindle speed to PWM
Spindle Status       > h          Move tool back to home position
> Spindle:  0        > b          Draw auto box at current location
                     > bit        Draw BIT SUNRISE at current location
                     > m          Exit Gcode mode back to Manual Mode
     Note: Please enter in G-Code format i.e 'xn yn zn sn'
           Does not need to be all fields i.e 'xn sn'
+--------------------------------------------------------------+

Command prompt:                  Error, X-axis set position is outside limits
>>> 
```

Figure 5: Terminal for x-axis limit



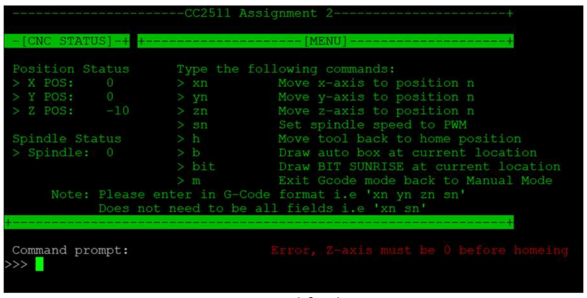
```
----------------------CC2511 Assignment 2----------------------+

-[CNC STATUS]-+ +-------------------[MENU]--------------------+

Position Status      Type the following commands:
> X POS:    0        > xn         Move x-axis to position n
> Y POS:    0        > yn         Move y-axis to position n
> Z POS:    0        > zn         Move z-axis to position n
                     > sn         Set spindle speed to PWM
Spindle Status       > h          Move tool back to home position
> Spindle:  0        > b          Draw auto box at current location
                     > bit        Draw BIT SUNRISE at current location
                     > m          Exit Gcode mode back to Manual Mode
     Note: Please enter in G-Code format i.e 'xn yn zn sn'
           Does not need to be all fields i.e 'xn sn'
+--------------------------------------------------------------+

Command prompt:                  Error, Y-axis set position is outside limits
>>> 
```

Figure 6: Terminal for y-axis limit



```
----------------------CC2511 Assignment 2----------------------+

-[CNC STATUS]-+ +-------------------[MENU]--------------------+

Position Status      Type the following commands:
> X POS:    0        > xn         Move x-axis to position n
> Y POS:    0        > yn         Move y-axis to position n
> Z POS:    -10      > zn         Move z-axis to position n
                     > sn         Set spindle speed to PWM
Spindle Status       > h          Move tool back to home position
> Spindle:  0        > b          Draw auto box at current location
                     > bit        Draw BIT SUNRISE at current location
                     > m          Exit Gcode mode back to Manual Mode
     Note: Please enter in G-Code format i.e 'xn yn zn sn'
           Does not need to be all fields i.e 'xn sn'
+--------------------------------------------------------------+

Command prompt:                  Error, Z-axis must be 0 before homeing
>>> 
```

Figure 7: Terminal for homing error

Part 2: Problems encountered and how were they fixed?

Problem 1 was an error in the schematic where the VREF header was not connected properly. The original track to the header needed to be scratched and bridging wires needed to be soldered from the 3V3 to the top node of the R1 resistor, and one from the bottom node to the VREF header to correct it (Fig. 8).

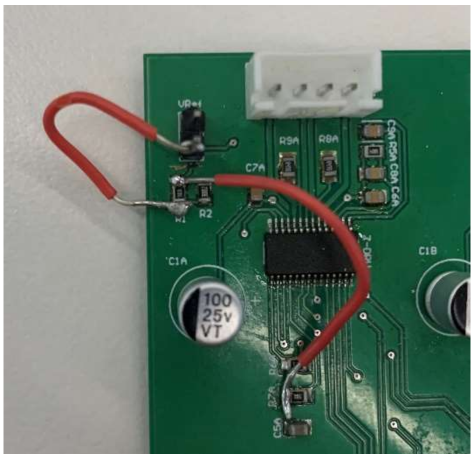


Figure 8: Bridging wires for VREF

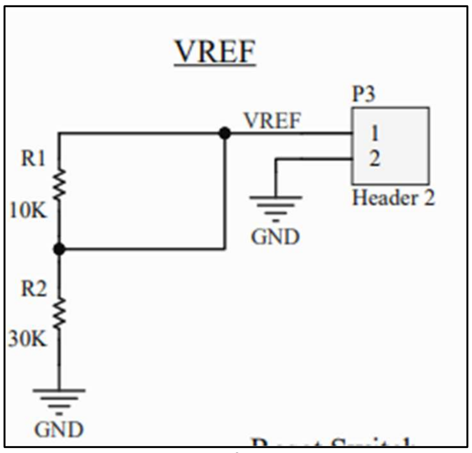


Figure 9: Error for VREF header

Problem 2 was when under operation the protection schottky diode on the Pico board was getting destroyed. This could have been caused by three potential issues being: a faulty voltage regulator, a short in the board at the voltage regulator, or current limiting the DC power supply when under operation. This problem was luckily thought ahead by the implementation of a 5V header discussed above. This allowed the Pico board to be powered separately to the other components with no further problems arising.

Finally, the last problem that occurred was software based. The software was initially written to find the maximum difference of the current point and set point and iterate axis stepping for that selected number of steps. This in turn caused the issue of not being able to do any other diagonal other than 45°. This issue was overcome by integrating the Bresenham's line algorithm.

Part 3: Recommendations

Whilst the code and PCB worked to the specific requirements there are areas for improvement. The software could be implemented with more user functionality. The main improvement being to create an actual G-code command interpreter rather than just the coordinates. This will provide the user to send specific G-code commands such as: G0 (move but not engraving), G1 (move and engrave), G2 (CW arcs), and G3 (CCW arcs). This functionality, more specifically the latter two would then need specific software to engrave arcs which should also be implemented. Additionally, user input should be implemented for the stepper controller MODE pins, which would allow the user to select a specific federate they would like to engrave or move at. Finally, a conversion between steps and a universal measurement of choice should be implemented. Converting the numbers of steps to a specific measurement such as millimetres (mm) would be ideal for higher precision user input.