University of Puerto Rico
Mayagüez Campus
Mayagüez, Puerto Rico
Department of Electrical and Computer Engineering

Phoenix DLP 3D Printer
Progress Report 3

by

Fernando R. López Domínguez - Project Leader
José L. Acevedo Flores
Héctor Y. Franqui Pagán
José R. Oyola Cabello

Professor: Manuel Jiménez
Course: ICOM 5217
March 5, 2013

Contents

1 Power Analysis

1. Voltage Compatibility

   a. Table 1

Device	**Operating Volatage**	$\mathbf{V}_{OH}$ (min)	$\mathbf{V}_{OL}$ (max)	$\mathbf{V}_{IH}$	$\mathbf{V}_{IL}$
MSP430F5528	3.3V	2.4V	0.4V	2V	0.8V
LightCrafter	5V/3.3V(Logic)	2.4V	0.4V	2V	0.8V
LCD Display HD44780	5V	3.75V	0.4V	2.2V	0.6V
DRV8824	3.3V	3.3V	0.5V	2V	0.6V

   b. As shown in the table above, the following voltage compatibility requirements are met for each of the components, except for the VOH for the LCD Display, but this does not affect the MCU, as the LCD is operated in write-only mode.

      i. $V_{OH} \geq V_{IH}$

      ii. $V_{OL} \leq V_{IL}$

   c. Electrical Characteristics of devices not used for logic operations

Device	**Operating Voltage**
Stepper Motor	7.5V @ 0.5A
LED	1.1V
Buzzer	1V-3V
Cooling System	12V

2. Driving Capability and Power Supply Requirement Calculations

   a. 3.3V

Device	**Current Requirement**
MSP430F5528	$10mA$
DRV8824 Driver (Logic)	$100\mu A$
LightCrafter (Logic)	$100\mu A$
Total Current	**10.2mA**

   b. 5V

Device	**Current Requirement**
LCD Display / HD44780	$1.2mA$
LCD Display Backlight	$72mA$
LightCrafter$*$	$2A$
LightCrafter Cooling Fan$**$	$50ma$
Total Current	**73.2mA**

$*$ The TI LightCrafter powered by an external 5V, 2A power supply.
$**$The LightCrafter cooling fan is powered by the LightCrafter module.

c. 12V

Device	**Current Requirement**
DRV8824 Driver (Motor Supply)	$0.5A$
DRV8824 Driver TI Chip	$8mA$
Coling Fan	$150mA$
Total Current	**658mA**

d. Power Supply LDOs

Device	$\mathbf{V}_{OUT}$	$\mathbf{V}_{IN(max)}$	$\mathbf{V}_{DO}$	$\mathbf{I}_{OUT}$
LM1117n (x2)	3.3V, 5V	15V	1.2V	800mA
LM1085	12V	27V	1.3V	3A

A 120V-to-25.2V transformer was chosen that supplies a maximum current of 2A to the 12V LDO. Due to the fact that each LDO has sufficient headroom in both input voltage as well as input current, the LDOs will be cascaded in order to meet the required maximum input voltages as well as improve the PSRR leading to the MCU.

e. Worst-case total power consumption of the system:
   i. $P_T = 3.3V * 10.2mA + 5V * (73.2mA + 2A) + 12V * 658mA = 18.3W$

2 Thermal Analysis

Calculating the power dissipation considering a worst-case operation of the system, we tabulated a total of 18 I/O pins functioning as outputs. With an operating frequency of 8MHz, each pin offers a maximum of 2 mA of output current, for a total of 36 mA. Besides the power dissipated by the 18 output pins, we tabulated 6 input pins which draw a maximum of 2 mA each, for a total of 12mW. The total power dissipation is 166.518 mW. This shows that even in the worst-case, the MCU does not exceed the maximum power dissipation of 1.27 W and does not require additional cooling or heat management measures. In addition, due to the VQFN 64-pin package of the MCU, it offers an additional heat dissipation pad on the bottom, which helps with controlling the heat. However, we also have a cooling fan system, which, although meant for the TI LightCrafter, it helps maintain an airflow to dissipate heat from the electrical components inside the case.

$$P_{diss(max)} = \frac{(T_{J(max)} - T_A)}{\phi_{JA}} = \frac{95°\text{C} - 25°\text{C}}{55°\text{C}/W} = 1.27W \tag{1}$$

$$P_{diss} = V_{DD} \cdot \left(I_{DD(avg)} + \sum_{allpins} |I_{IO(avg)}| \right) = 3.3V\,(2.46mA + (18_{out} + 6_{in}) \cdot 2mA) \tag{2}$$

$$= 166.518mW$$

3 Timing Analysis

For this project, there are two components that are time sensitive:

3.1 LCD

- LCD Enable Signal
- LCD RS Signal
- LCD Data Lines

The RS line on the LCD has to be in the logic one or logic 0 for at least 100ns before the Enable line is set. The Enable line has to be in logic one for at least 300ns. This will ensure that the data found in the Data Lines (Port 6) is read correctly by the driver. It takes the driver 10ns to fetch the data from the Data Lines. This implies that the enable signal should remain low for at least 10ns. It is worth noticing that the Data Lines will hold the logic levels for more than 10ns since this is the first operation in the subroutine in charge of writing to the LCD.

3.2 Stepper Motor Driver

- Step Signal
- Direction Signal
- nEnable Signal

For the stepper motor driver to perform a correct and precise movement, the Direction signal should be either logic 0 or logic 1 for at least 200ns. After that, the nEnable signal should go low (logic 0) for at least 200ns. Once both signals, nEnable and Direction, have been at their logic states for the required amount of time, the Step signal should toggle with a frequency less than 250 khz with a 50% duty cycle.

The slowest component on the system is the stepper motor which requires the step signal to be set for a period of at least 2 us (250 KHz signal, 4 us period, and at 50% duty cycle, that would be at least 2 us), however the selected frequency for the system master clock is set to 8 MHz. A clock signal of 8 MHz has a period of 125 ns, therefore, it would be necessary to perform a delay of at least 16 CPU cycles. This delay will be performed using a software loop and not an interrupt. The reasons specified for this design decision are as follows. The operation being performed here is a very sensitive one: the movement of the stepper motor has to be timed precisely with the resin drying time in under to avoid side effects that prolonged exposure may cause, like the undesired drying of nearby resin. Since this is time sensitive, a timer is not suitable. Servicing an interrupt takes at least 6 cycles (see the User Manual, page 215) and returning from interrupt takes 5 cycles. In only servicing and returning from interrupt, 11 cycles are wasted, for a total time of 1.375 us, which is completely undesired in the system being considered. Therefore, is better to waste 16 cycles and make sure that the timing is just correct to ensure proper system operation.

4 Software Plan

4.1 Printer Side Operation

4.1.1 Functions

Main() function:

1. Initialize Stack Pointer
2. Initialize Watchdog Timer
3. Call **InitializeGPIOPorts()** function to initialize ports
4. Call **InitializeDisplay()** function to initialize LCD Display
5. Call **ResetMotorToBottom()** function
6. Call **UVShieldClosePrompt()** function
7. Call **InitializeUSBComm()** function
8. Enable interrupts
9. Set **printer status** as 1 (ready)
10. Call **LCDWriteString()** function with parameter 'Printer Ready∧'
11. Send ready signal to host computer through USB
12. Wait for printing request
13. If a printing request is received, the interrupt will be served through the **Data-Transfer()** function/routine
14. If **Transfer Complete** flag is set, then call **PrintFigure()** function
15. Disable interrupts
16. Poll reed sensor to know the status of the UV-shielded lid. (Note: lid should have remained closed up to this moment).
    a. If the lid is opened, then go to step 16
    b. If the lid is closed, then turn off the buzzer LED and continue
17. Enable interrupts
18. Set **printer status** as 4 (done printing)
19. Call **LCDWriteString()** with argument "Press '>' to complete∧" operation.
20. Wait for 30 seconds (may change)

21. If no interrupt is received, perform a power on reset.

JobCancel() function:

1. Call **ResetMotorToTop()**
2. Call **LCDWriteString()** with argument "Printer will∧reset promptly".
3. Wait for 2 seconds (may change)
4. Perform a power on reset.

InitializeGPIOPorts() function:
Parameters: None
Configure Port 1 to service as input for: Reed sensor, Motor Reset position sensors(top, bottom), User Buttons (2 buttons), LCr trigger in. (6 pins)

1. Configure Port 1.0 as input for the Reed Sensor with interrupt on High to Low Transition.
2. Configure Port 1.1 as input for top position sensor. Configure interrupt to be generated on high to low transition.
3. Configure Port 1.2 as input for bottom position sensor. Configure interrupt to be generated on high to low transition.
4. Configure Port 1.3 as input for one user button. Configure interrupt to be generated on high to low transition.
5. Configure Port 1.4 as input for one user button. Configure interrupt to be generated on high to low transition.
6. Configure Port 1.5 as output for LED buzzer.
7. Configure Port 1.6 as output for Buzzer.
8. Configure Port 2.0 as input for LCr trigger.
9. Configure Port 2.1 as output for LCr trigger.
10. Configure Port 3.0 as output for Direction on Motor Driver.
11. Configure Port 3.1 as output for Step on Motor Driver.
12. Configure Port 3.2 as output for M0 (Microstepping mode) on Motor Driver.
13. Configure Port 3.3 as output for M1 (Microstepping mode) on Motor Driver.
14. Configure Port 3.4 as output for M2 (Microstepping mode) on Motor Driver.
15. Configure Port 5.0 as output for LCD enable signal.

16. Configure Port 5.1 as output for LCD rs signal. Configure Port 6.0 - 6.7 as output for LCD data lines.

InitializeDisplay() function:
Parameters: None

1. Clear LCD display
2. Enable 8 bit mode using the 2 lines
3. Turn on underline Cursor
4. Enable left to right autoincrement entry mode
5. Return to caller

LCDWriteCommand() function:
Parameters: An integer number in hexadecimal format that contains the command for the LCD

1. Set the Command Bit (RS line) to 0
2. Load command in to the LCD Data Bits
3. Set the LCD enable pin high
4. Wait for approximately 350ns
5. Set the LCD enable pin low to trigger the LCD
6. Return to caller

LCDWriteString() function:
Parameters: A string of characters to be printed (format of string, line delimiters)

1. Call **LCDWriteCharacter()** for the first character in the given string
2. Wait for 10ns (delay required by LCD driver to acknowledge data on bus)
3. Go to step 1 until all characters are written
4. Return to caller

LCDWriteCharacter() function:
Parameters: A character to write to the LCD

1. Set the Data Bit (RS line) to 1
2. Load character in to the LCD Data Bits
3. Set the LCD enable pin high
4. Wait more than 300ns

5. Set the LCD enable pin low to trigger the LCD

6. Return to caller

ResetMotorToTop() function:
Parameters: None

1. Check **Top motor reset position** sensor

2. If the top reset position sensor is not pressed:

   a. Set direction signal with UP

   b. Set enable signal

   c. Wait for at least 200ns

   d. Enable Step signal

3. Wait $4\mu s$ (delay required for stepper motor to acknowledge STEP signal)

4. While the top motor reset position sensor is not pressed go to step d.

5. Return to caller

ResetMotorToBottom() function:
Parameters: None

1. Check Bottom motor reset position sensor

2. If the bottom reset position sensor is not pressed:

   a. Set direction signal with DOWN

   b. Set enable signal

   c. Wait for at least 200ns

   d. Enable Step signal

3. Wait $4\mu s$ (delay required for stepper motor to acknowledge STEP signal)

4. While the bottom motor reset position sensor is not pressed go to step d.

5. Return to caller

MoveStepperMotor() function: Uses parameter to generate the Step signal the required amount of times to move the stepper X mm up (X being the parameter), this depends on the resolution of the image. Direction of movement also is parameter. **Parameters**: Direction of movement (up or down), magnitude of movement (in millimeters)

1. Calculate how many times the STEP signal must be generated based on the **layer thickness.**

2. Set stepper Direction to UP or DOWN (given as a parameter)
3. Set the Enable signal
4. Set the STEP signal
5. Wait $4\mu s$ (delay required for stepper motor to acknowledge STEP signal)
6. Count how many steps have been done so far
7. Check if all the STEP signals have been generated (compare with steps done so far)
8. If not, go to step 2
9. Otherwise the platform is at the desired position
10. Return to caller

UVShieldStatusCheck() function:
Parameters: None

a. Check UV shield reed sensor

b. If its not closed:

   (a) Call **LCDWriteString()** function with argument 'Close Lid∧'

   (b) Go to 1

c. If closed return to caller

LCDSwitchStatus() function:
Parameters: None

1. Check **LCD Status** counter.

   a. If the counter equals zero:

      i. Get current layer being printed and call **LCDWriteString()** with argument "Printing layer ∧ " + **current layer** + " of " + **layer quantity**.

      ii. Increase **LCD Status** counter

      iii. Return to caller

   b. If the counter equals one:

      i. Get current time and calculate elapsed time by comparing with **start time** and call **LCDWriteString()** with argument " Approximately ∧" + **elapsed time** + " elapsed".

      ii. Increase **LCD Status** counter

      iii. Return to caller

   c. If the counter equals two:

i. Calculate remaining time by: multiplying **layer quantity** by **resin drying time** and subtracting it to elapsed time (current time - **start time**)
ii. Call **LCDWriteString()** with argument " Approximately ∧" + remaining time " remaining".
iii. Increase **LCD Status** counter (depends; if theres another status of current file name being printed, then it should increase. Else, it must return to zero)
iv. Return to caller

d. If the counter equals three (name of current file being printed)
i. Get current layer filename and call **LCDWriteString()** with argument " Printing file ∧" + **current layer filename**. Set **LCD Status** counter to zero.
ii. Return to caller

BuzzerAlert() function:
Parameters: None

1. Set timer to get fed by a 1024Hz clock
2. Load terminal count with 2
3. Turn on Buzzer LED
4. Return to caller

PrintFigure() function:
Parameters: None

1. Call **UVShieldStatusCheck()** function
2. Call **ResetMotorToBottom()** function
3. Set **printer status** to 2 (printing)
4. Initialize **LCD Status** to zero
5. Store the **start time** of the printing operation
6. Trigger LightCrafter to display layer
7. Increase **current layer** count
8. Call **LCDSwitchPrintStatus()**
9. Wait a predefined amount of time (**resin drying time**) to allow for the layer to cure
10. Check if this **current layer** is the last layer by comparing with **layer quantity**

11. If this **current layer** is not the last one:

    a. Call **MoveStepperMotor()** with arguments direction UP and magnitude **layer thickness**

    b. Go to step 6 (trigger LightCrafter to show next layer)

12. Once all the layers are printed Call **ResetMotorToTop()** function.

13. Call **LCDWriteString()** function with argument 'Finished Printing∧'

14. Check if the **audible alert** flag is set, then call **BuzzerAlert()** function

15. Wait some specific amount of time and turn buzzer off (turn off the timer of the buzzer)

16. Return to caller

4.1.2 Interrupt Service Routines

This function is called when a printing request is received after the MCU sends the ready signal **DataTransfer() (ISR)**:

1. Initialize LightCrafter

2. Send Print acknowledgement package to host computer from MCU through USB

3. Call **LCDWriteString()** with argument 'Data is being received'

4. Receive all printing information (**layer thickness**, **layer quantity** and if an **audible alert** will be produced at the end of printing) into MCU for proper control of the stepper motor and the LightCrafter

5. Send Layer-info-transfer-complete acknowledge to host PC

6. Switch USB hub so that LightCrafter can receive images: Alternatives

    a. Have two USB inputs into the casing of the printer. Then instruct the user to first connect to one of them (the MCU USB) and then to connect it to the other port (USB LightCrafter)

    b. Have a manual hardware switch that chooses direction of data (from host PC to MCU or from host PC to LightCrafter)

    c. Implement one custom USB port that basically acts as a multiplexer, and by using software control to where the four lines of the USB go.

    d. Use TI DVSDK.

7. Set **Transfer Complete** flag. Many ways to do so:

a. Hopefully, communication between LightCrafter and MCU will be possible. Flag is set through software.

b. If hardware switch is implemented, then the switch status must be polled and interpreted as follows: initially, the switch is in host PC to MCU position. Then the switch must change to host PC to LightCrafter position. If this sequence is followed, then when the switch is placed back in the MCU position, the **Transfer Complete** flag should be set.

8. Return from interrupt

BuzzerToggle() (ISR):

1. Toggle Buzzer pin
2. Clear Timer IFG
3. Return from interrupt

UVShieldOpenedDuringJob() (ISR):

1. Call **BuzzerAlert()** function
2. Call **LCDWriteString()** function with arguments "Please Close∧UV Shield"
3. Check reed sensor
4. If UV lid not closed, go to step 3
5. If reed sensor signals the UV-lid has been closed:
   a. Deactivate buzzer
   b. Call **LCDSwitchState()**
6. Return from interrupt

ArrowButtonPressed() (ISR):

1. Check **printer status**
   a. If **printer status** is 1 (ready), return from interrupt (nothing to do)
   b. If **printer status** is 2 (printing), call **LCDSwitchState()**
   c. If **printer status** is 3 (canceling), call **JobCancel()**
   d. If **printer status** is 4 (done printing) call **JobCancel()**
2. Return from interrupt

CancelButtonPressed() (ISR):

1. Call **LCDWriteString()** with argument "Press '>' to∧cancel job"
2. Set **printer status** to 3 (cancelling)
3. Return from interrupt

4.2 Host Computer Side

Phoenix3D application

1. Wait for user to import the stl file that is wished to be printed.
2. Wait for user to select layer thickness.
3. When step 1 and step 2 are done generate the cross sections that will be projected by the LCr. For this call Slice STL function.
4. Verify if the printer is connected and ready.
5. If the printer is connected and ready enable the print option in the host computer application.
6. When the user clicks the print button Send print request to the MSP430.
7. After the print request is sent wait for a print acknowledge from the MSP430.
8. When the print acknowledge is received send printing parameters: layer thickness, filename, total number of layers and if an audible alert will be performed when printing is finished.
9. When all the layers are sent, the MCU will confirm the data arrived.
10. At this point the printer will start the printing process and the computer would be safe to disconnect.

Slice STL function:

1. Save layer thickness and path to STL file selected by user.
2. Invoke FreeSteel script with the saved arguments.
3. Generate the files on relative path layers/
4. Return to caller.

5 Block Diagram

This section provides the system block diagram of the Phoenix3D Printer. This system has passed three revisions. The most significant additions since version 2 are:

1. Detail description of power supplies [1].
2. Components are now labeled with their respective part numbers.
3. The addition of a second limit sensor was proposed and added for the bottom part of the platform.

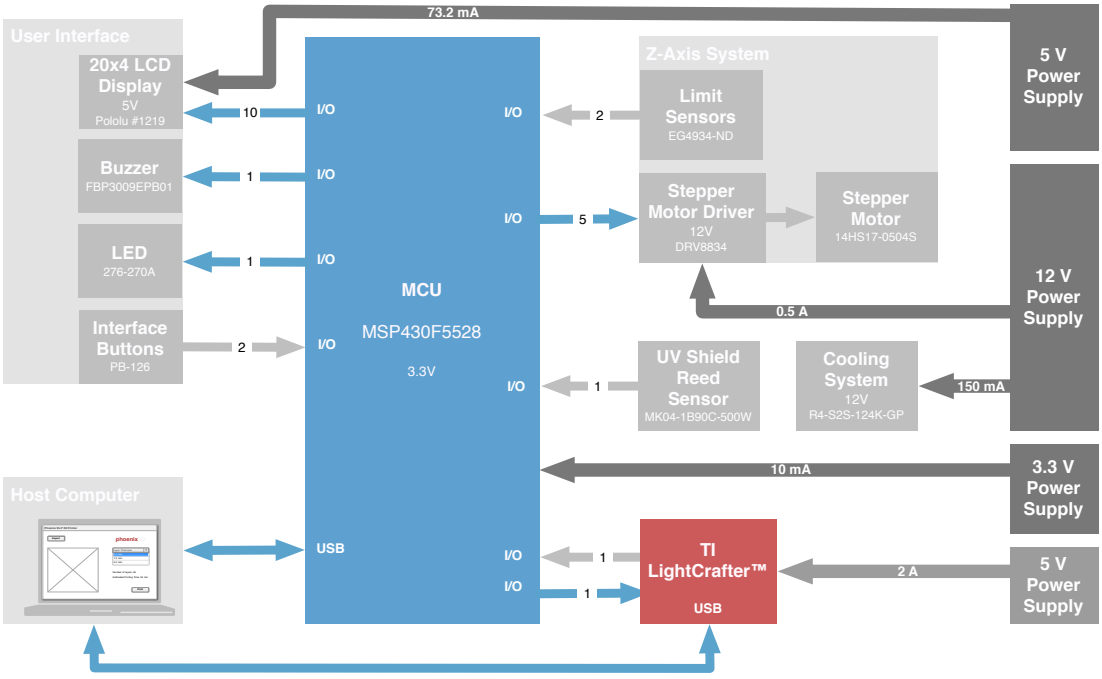

Figure 1: Block Diagram Version 3

6 Schematics

In this section the reader will find the system's schematic. This schematic was constructed using EAGLE, an industry grade software that provides useful tools for schematic designs.

[1] The TI LightCrafter is powered by an independent power supply

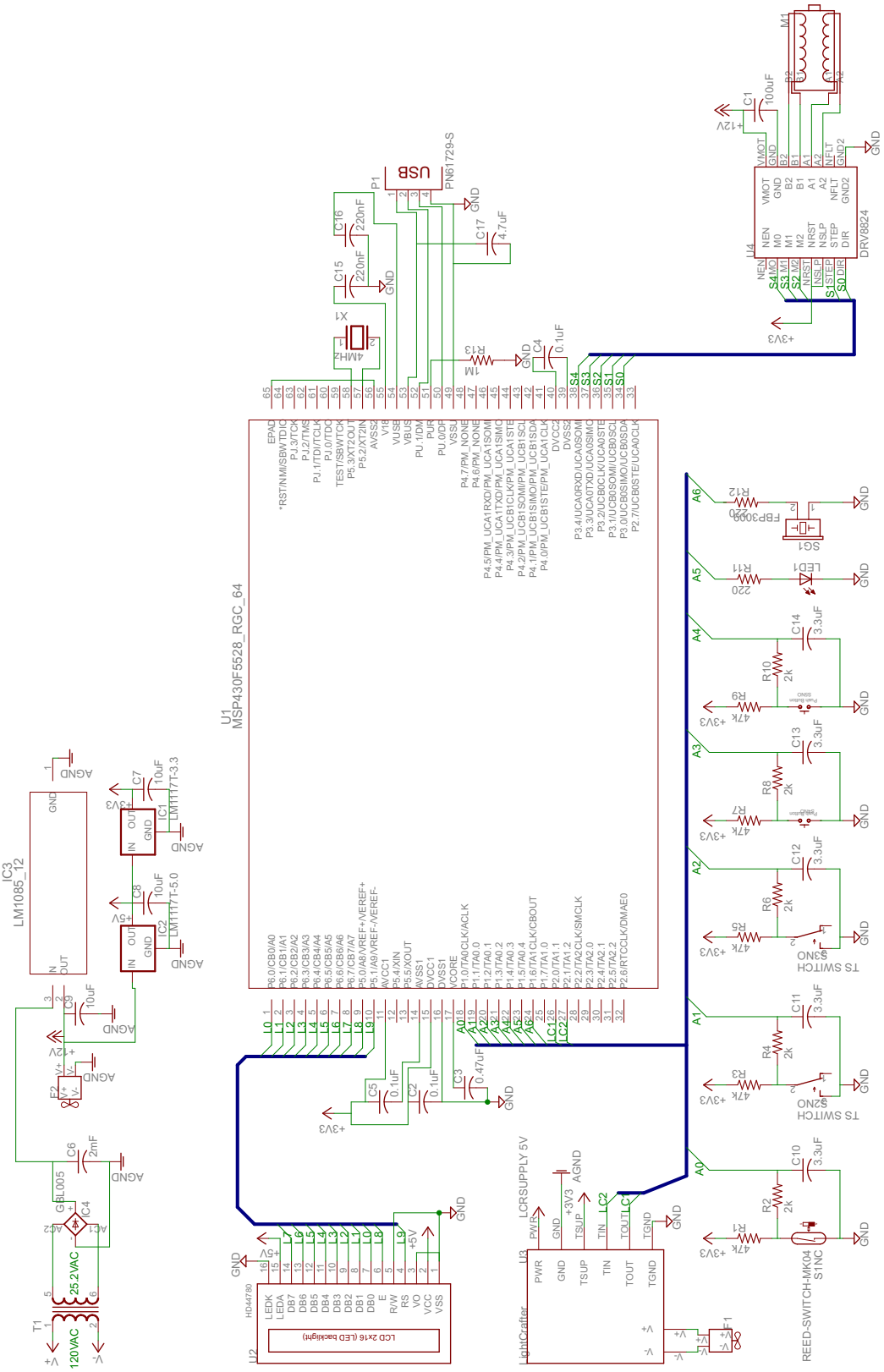

Figure 2: System Schematics

7 Parts List

This section shows a detailed outline of all the electric components that were used in the schematic design.

Part	Value / Part Number	Device
C1	100uF	C-US075-032X103
C2	0.1uF	C-US075-032X103
C3	0.47uF	C-US075-032X103
C4	0.1uF	C-US075-032X103
C5	0.1uF	C-US075-032X103
C6	2mF	C-US075-032X103
C7	10uF	C-US075-032X103
C8	10uF	C-US075-032X103
C9	10uF	C-US075-032X103
C10	3.3uF	C-US075-032X103
C11	3.3uF	C-US075-032X103
C12	3.3uF	C-US075-032X103
C13	3.3uF	C-US075-032X103
C14	3.3uF	C-US075-032X103
C15	220nF	C-US075-032X103
C16	220nF	C-US075-032X103
C17	4.7uF	C-US075-032X103
F1	5V Fan	5V Cooling Fan
F2	R4-S2S-124K-GP	12V Cooling Fan
IC1	LM1117n-3.3	3.3V LDO
IC2	LM1117n-5.0	5V LDO
IC3	LM1085_12	12V LDO
IC4	GBL005	Bridge Rectifier
LED1	276-270A	Red LED
M1	14HS17-0504S	7.5V 0.5A Stepper Motor
P1	PN61729-S	USB Module
R1	47kΩ	R-US_0207/10
R2	2kΩ	R-US_0207/10
R3	47kΩ	R-US_0207/10
R4	2kΩ	R-US_0207/10
R5	47kΩ	R-US_0207/10
R6	2kΩ	R-US_0207/10
R7	47kΩ	R-US_0207/10
R8	2kΩ	R-US_0207/10
R9	47kΩ	R-US_0207/10
R10	2kΩ	R-US_0207/10
R11	220Ω	R-US_0207/10
R12	220Ω	R-US_0207/10
R13	1MΩ	R-US_0207/10

Part	Value / Part Number	Device
S1-NC	MK04-1B90C-500W	Reed Switch
S2-NO	EG4934-ND	Limit Switch
S3-NO	EG4934-ND	Limit Switch
S4-NO	PB-126	Push Button
S5-NO	PB-126	Push Button
SG1	F-B-P3009EPB-01 LF	Buzzer
T1	273-1512	120VAC 2A Transformer
U1	MSP430F5528_RGC_64	MSP430F5528_RGC_64
U2	Pololu #1219	20X4 LCD w/ Backlight
U3	LIGHTCRAFTER	TI LightCrafter
U4	DRV8824	Stepper Motor Driver
X1	XT6S20ANA4M	4MHz Crystal