Table of Contents

ABSTRACT

The project is to design and construct horizontal blinds that will open, close, raise and lower depending on the buttons pressed on a numeric keypad. The movements of the blinds will be performed by stepper motors driven by a 68HC711D3 microprocessor. In addition, the blinds will also have an automated option such that the blinds will open and then be raised or lower and then be closed depending on the nearby light measured by a phototransistor.

PHSYICAL BLINDS CONSTRUCTION

Parts List

- Wood
- JB Weld
- (6) 3/8” x 2” hangar bolts
- (18) 3/8” wing nuts
- 2 Motors
- 1 Lever Switch
- 1 Toggle Switch
- (6) 3/8” x 2” bolts

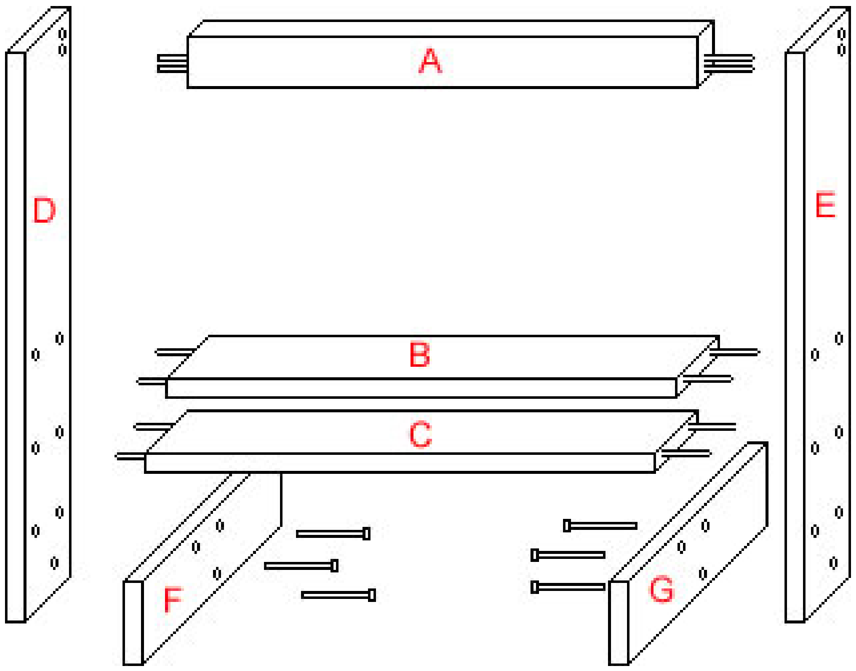


Figure 1 - Basic components of blinds

Table 1 - Wood Pieces and Dimensions

Component	Piece Dimensions	Number of Pieces	Function
A	1" x 4" x 15"	1	Blind Attachment Point
B, C	1" x 4" x 15"	2	Window Sills
D, E	1" x 4" x 27"	2	Arms
F, G	1" x 4" x 11"	2	Feet

Basic Assembly Procedures

To be able to store our project in our tub, it was designed to be easily collapsible. This was accomplished by using bolts and hangar bolts (one threaded end and one bolt end) in conjunction with wing nuts.

First, a rough sketch of the windowsill was done. Then one 5’ board of 1” x 4” wood was purchased and cut to the above dimensions. Part A had two holes drilled toward one end on both sides, to ensure half of the board would sit above the arms (D and E). This was to ensure that our motors would have enough clearance above the arms to operate properly. Two hangar bolts were then drilled into the holes, leaving a threaded end exposed.

Parts B and C were also pre-drilled, but these holes were placed 1 inch apart. Two more hangar bolts were then installed on each end.

Parts F and G were cut to length and three holes were drilled in a triangular pattern, to increase stability. Three 3/8” x 2” bolts were then fitted through these parts and parts D and E.

Parts D and E, the arms, were cut to the specified dimensions and predrilled to correspond with the preexisting hangar bolts and bolts. To join our windowsill together, all bolts where slid through the arms and were attached with the wing nuts.

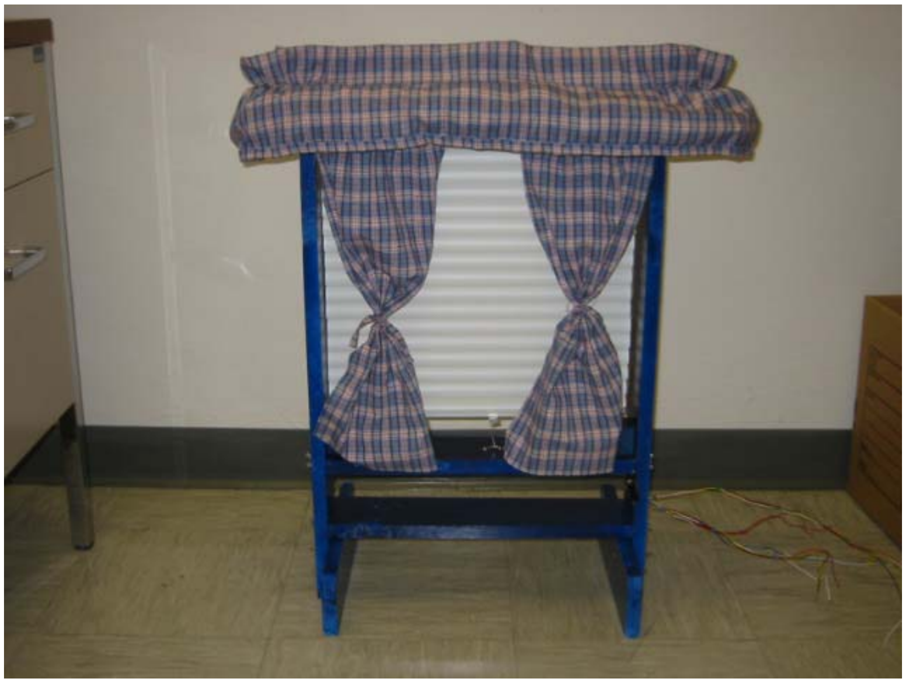


Figure 2 - Picture of front view of blinds

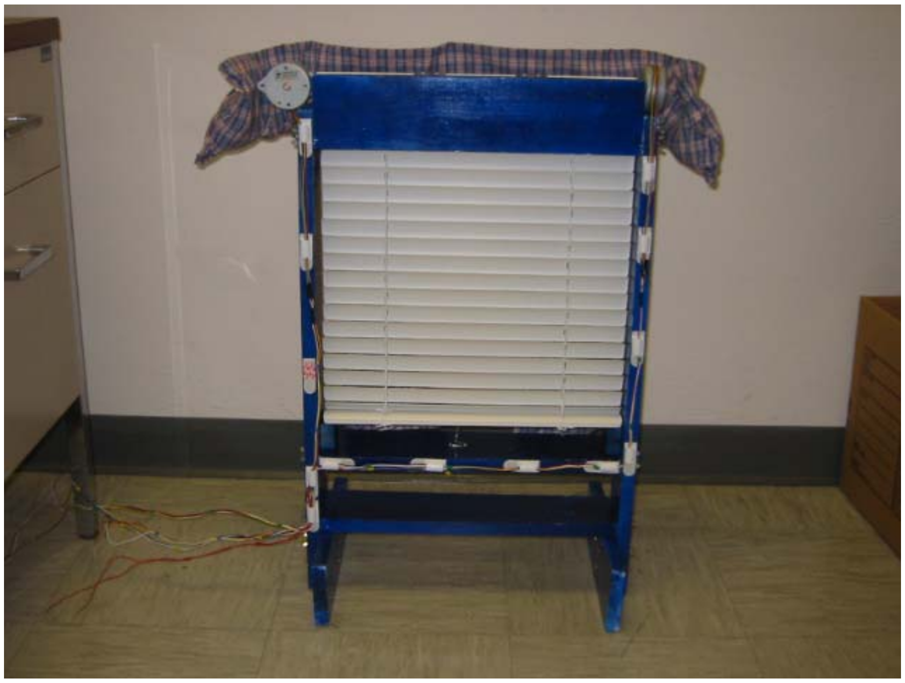


Figure 3 - Picture of back view of blinds

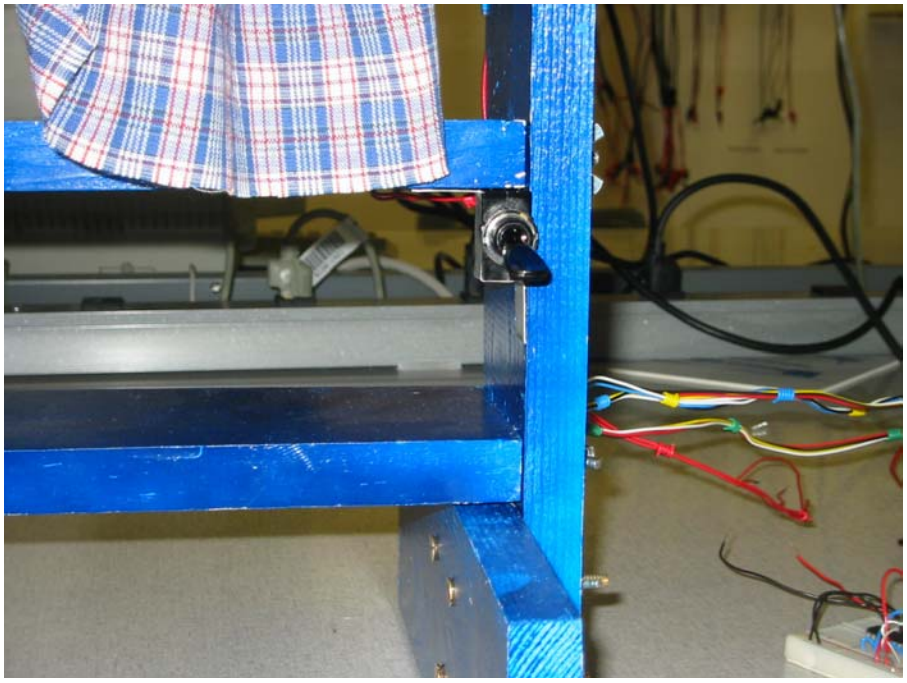


Figure 4 - Picture of automation switch

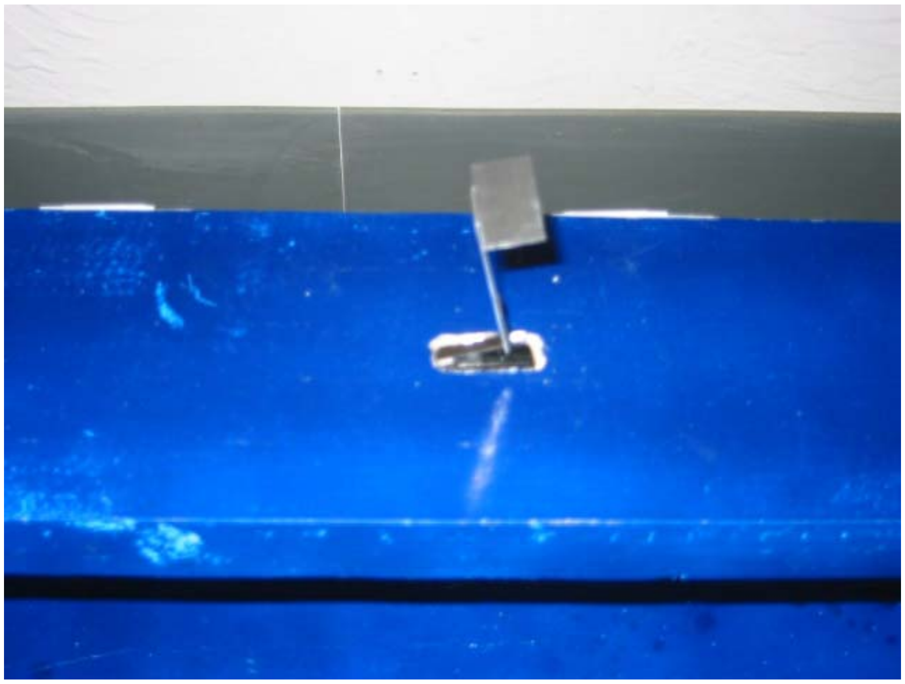


Figure 5 - Picture of windowsill switch

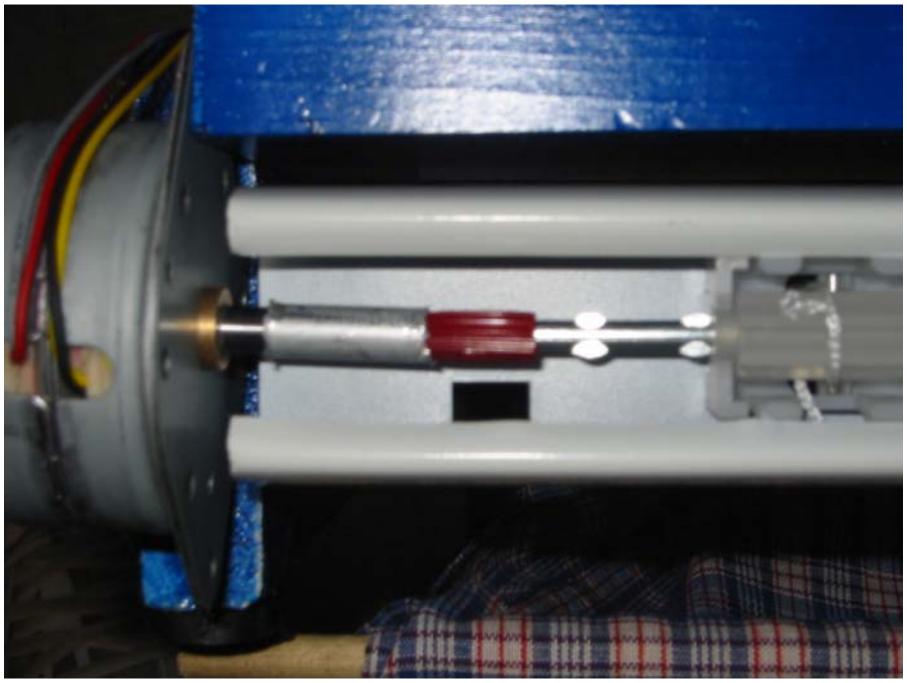


Figure 6 - Picture of Open/Close motor connection

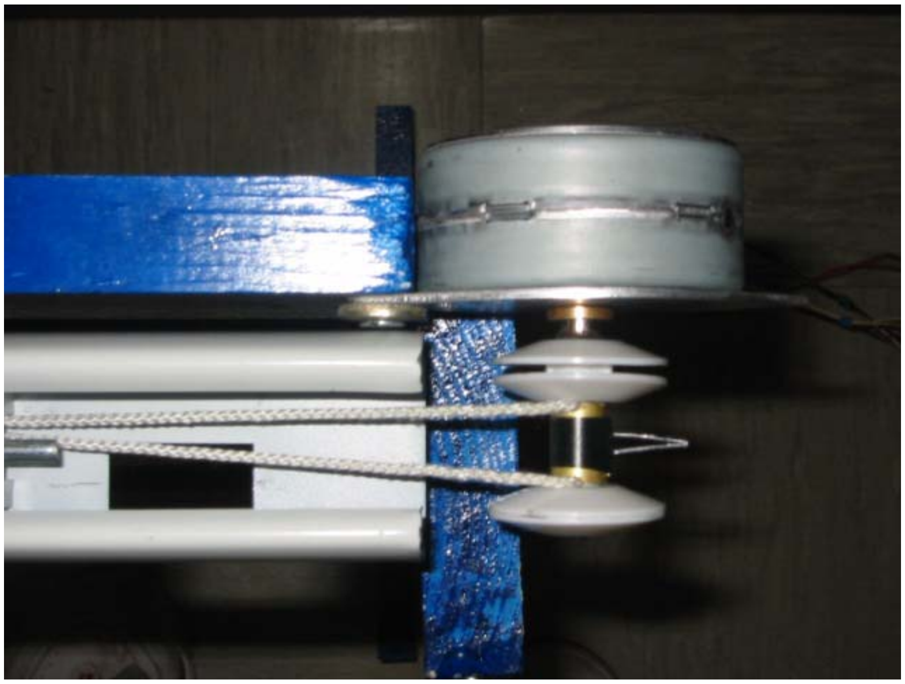


Figure 7 - Picture of Raise/Lower motor connection

HARDWARE OVERVIEW

The Motors

Two identical bipolar motors were used to move the horizontal blinds. One motor was used to open and close the blinds and the other was used to raise and lower the blinds. The important characteristics of each motor are that its holding torque is high, 33.5oz per inch, and in order to operate the motor must be powered by 5V DC and 1A of current. In order to provide such a high current and to pulse the motors to rotate in forward or reverse, an H-bridge was attached to each coil of the motor, see Figure 10. The H-bridge passed current through the coils of the motor. The direction of the current, which also determined the rotation of the motor, was specified by the H-bridge control inputs A and B. Diodes were also connected to the H-bridge to prevent back EMF current, which could damage the hardware components.

Keypad

The keypad was the control device that allowed the user to select which operation to perform on the blinds. Four of the buttons were activated on the keypad and each performed a unique operation. Button 1 opened the blinds, button 2 closed the blinds, button 3 raised the blinds and button 4 lowered the blinds. The keypad pins were connected to a 10-to-4 priority encoder whose outputs provided signals to PORTD of the microprocessor, see Table 2.

Table 2 – Keypad

10 to 4 Encoder Inputs									10 to 4 Encoder Outputs Ports PD0	PD1	PD2	PD3	
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**PORTD INPUT**
0	1	1	1	1	1	1	1	1	0	1	1	1	00001110
1	0	1	1	1	1	1	1	1	1	0	1	1	00001101
1	1	0	1	1	1	1	1	1	0	0	1	1	00001100
1	1	1	0	1	1	1	1	1	1	1	0	1	00001011
1	1	1	1	1	1	1	1	1	1	1	1	1	00001111

Keypad Button Pushed - operation performed
1 - open blind
2 - close blind
3 - raise blind
A - lower blind

Windowsill Switch

A switch was attached to the bottom windowsill of the blinds. The purpose of this switch was to detect whether or not the blinds were located at the bottom of the window. When the switch was pressed, a high input was provided to PA0 of the microprocessor; otherwise the pin received a low input.

Automation Switch

The automation switch was mounted on the front of the blinds and indicated whether or not the blinds would be automated. When the switch was flipped down, the automation option was considered off and pin PA1 of the microprocessor was provided 0V. When the switch was flipped up, the automation option was activated and PA1 received a high input of 5V. NOTE: Automation indicates that the movements of the blinds would be determined by its surrounding light instead of the keypad buttons.

Phototransistor

The phototransistor was required for the automation of the blinds and was located on the large circuit breadboard. The phototransistor always detected the light around its surroundings. If it detected bright light, the phototransistor would conduct current. The current was passed through a resistor, creating a voltage drop, and inputting a low signal to pin PA2 of the microprocessor, see Figure 13. If the phototransistor detected little or no light, no current was conducted and providing a high input to PA2. Though the phototransistor always performed the operation, its effect on PA2 was only analyzed when the automation switch was activated.

CIRCUITRY COMPONENTS

More details can be found in Appendix B

55M048D Bipolar Stepper Motors

Brief Description: These motors provide very high holding torque, 33.5 oz per inch. The motors require 5V DC and 1A from a power supply.

Application: one stepper motor was used to raise and lower the blinds, and the other one was used to open and close the blinds.

SN75LS147 10-Line-to-4-Line Priority Encoder

Brief Description: This component receives 10 inputs and specifies which input is low by providing a 4-bit output. If more than one input is low, then the highest number input gets priority.

Application: This component detected which keypad button was pressed, and then provided the microcontroller a 4-bit representation of this information.

SN754410 Quadruple Half-H Driver

Brief Description: This chip contains four half H-bridges that are capable of driving 1 A to a load resistance.

Application: Each pair of half bridges was connected to create a full H-bridge. Then, the motor driver application schematic provided with the component was implemented. The driver connected each coil of the bipolar motors to an H-bridge and had two control inputs A and B. The inputs were pulsed in order to manipulate the direction of the current through the coils. A clockwise rotation sequence is the following, and by reversing its order a counterclockwise rotation is created:

Clockwise Rotation

A – 1 B – 1
A – 1 B – 0
A – 0 B – 0
A – 0 B – 1
A – 1 B – 1

1N4007 Diodes

Brief Description: This component has high surge current capacity.

Application: These diodes were connected to the motor driver in order to prevent back EMF current ruining circuit components.

74HCT373 Octal D-type transparent latch

Brief Description: This is a high-speed CMOS device that captures input when latch enable receives a high, and stores the input when latch enable is low. When output enable is low, the contents are available at the outputs.

Application: The microprocessor was connected to the latch in order to transfer and receive information. The 8 LSB of the microprocessor were connected to the input and output lines of the latch. When the microprocessor wanted to read data from the EPROM, the latch was provided the address from the microprocessor. Then the latch outputted to the EPROM specifying the address location to the EPROM. The EPROM's data would then be read by the microprocessor.

Am27256 EPROM

Brief Description: 256 Kbit, ultraviolet erasable programmable read-only memory. It is capable of holding 32K words.

Application: This component contained the assembly program that was written to implement the Automated Blinds Project.

Dm74LS00 Quad 2-Input NAND Gates

Brief Description: This component contained 2-input NAND gates.

Application: Used to enable the EPROM by NANDing the 15th address line and the address strobe. In addition, it NANDed the E clock.

DM74LS244 Octal 3-State Buffer

Brief Description: This component receive up to 8 inputs. It provides an output with the same signal of the input but with a higher voltage and larger current.

SN74LS04 Hex Inverter

Brief Description: The component contained six inverters.

Application: This component received the input control signals for the motor driver.

Motorola 69HC711D3 Microprocessor

Brief description: This device had a nominal bus speed of 2Mhz and contained an 8-bit CPU. There are 7 CPU registers available: two 8-bit accumulators, two 16-bit index registers, a stack pointer, a program counter, and condition code register. The HC11 uses a memory-mapped architecture, so I/O pins can be referenced by the 64-kilobyte memory map.

Application: The device received inputs from the toggle switch, flip switch, phototransistor, and the 10-to-4 line encoder. It also provided outputs for the motor drivers.

16 Button Keypad

Brief description: 4 x 4 matrix keypad.

Application: Four buttons on the keypad were scanned in order for a user to provide inputs to determine the operation to be performed on the blinds: open, close, raise and lower the blinds.

IFD-5 Infrared Diode/Transistor

Brief description: The transistor becomes active and conducts current when light is detected, otherwise no current is conducted.

Application: It was used to provide an input pin on the microprocessor with a low or high signal depending on the amount of light detected.

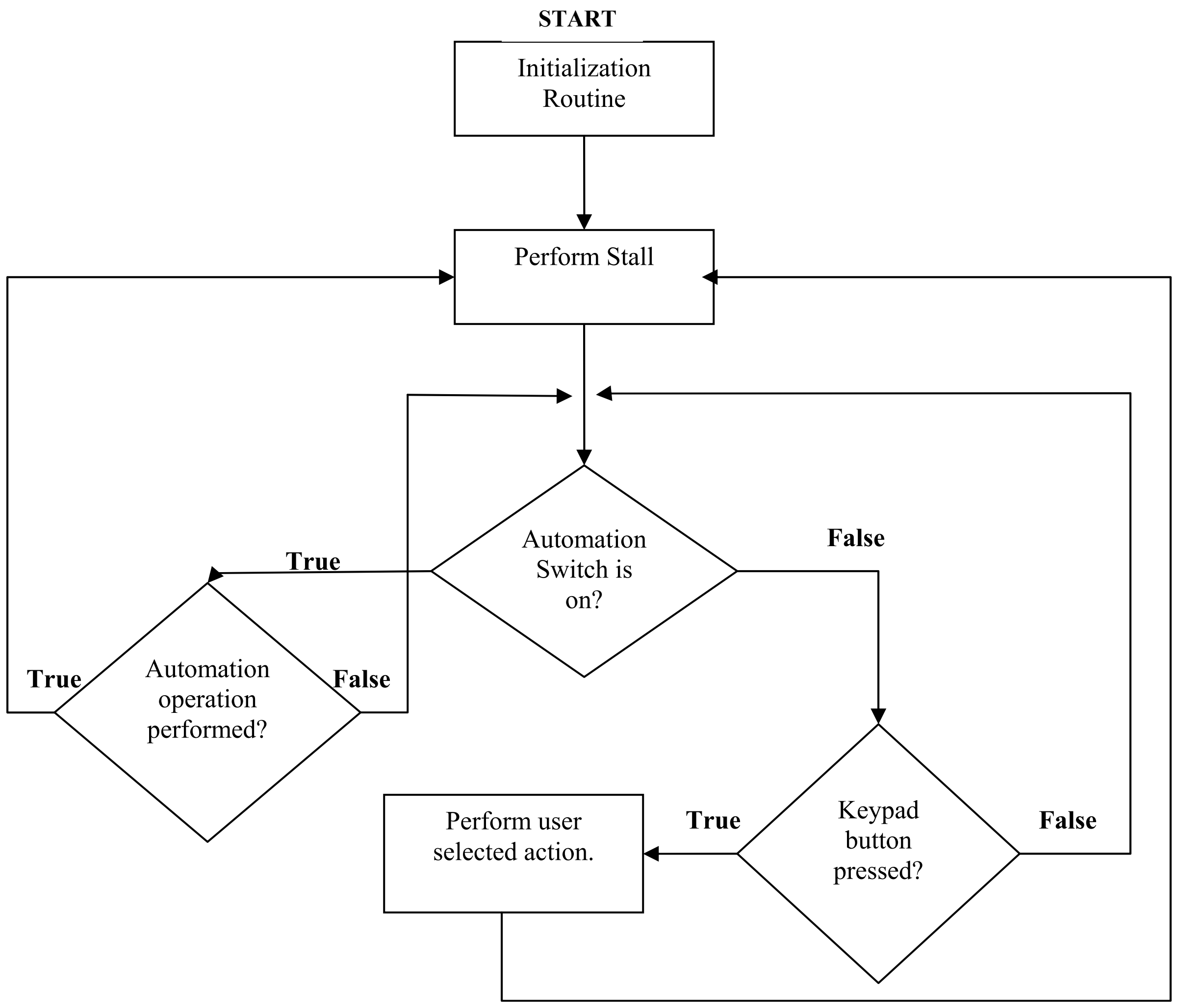
Toggle and Flip Switch

Brief Description: The toggle switch had continuity between pin 1 and pin 2 when not pressed, and pin 1 and pin 3 when pressed. The flip switch simply connected pin 1 and pin2 when the switch was flipped towards pin 1, otherwise they were separated.

Application: The toggle switch was pressed by the blinds when they were lowered all they way, in turn creating a high input to A0 of the microprocessor. The flip switch would input a high signal to A1 when it was flipped towards pin 1, and indicated whether or not to automate the blinds.

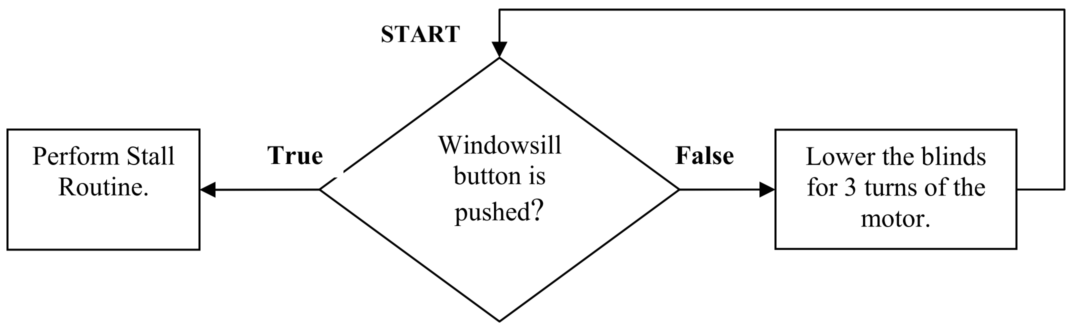
SOFTWARE OVERWIEW

Abstract View



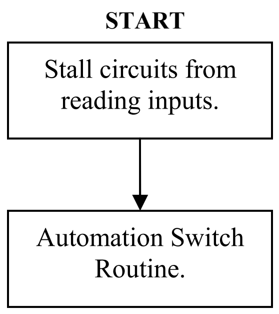
The abstract view of the program provides the general concept of the program used to implement the Automated Blinds Project.

<u>Initialization Routine (*initial*)</u>


START
Perform Stall Routine.
True
Windowsill button is pushed?
False
Lower the blinds for 3 turns of the motor.

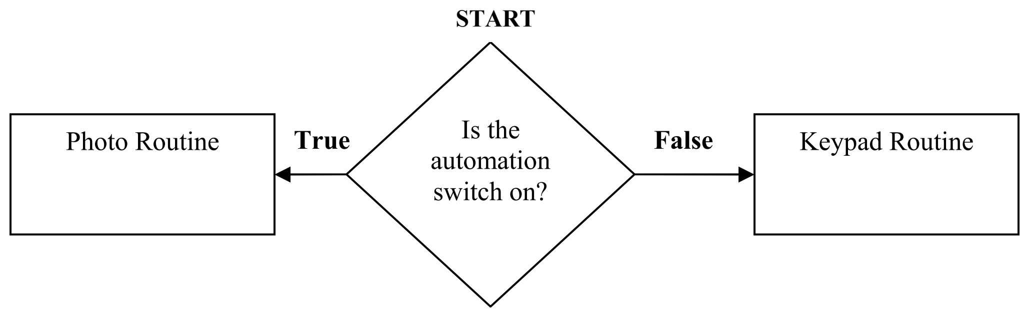
The initialization routine lowers the blinds to the very bottom of the window when the circuit is turned on. It accomplishes this by analyzing the input of PA0. If PA0 is high, then the windowsill switch is pressed and the blinds are at the bottom of the window. This means no actions on the blinds need to take place. Thus, register Y is set to 0 to indicate the bottom location of the window, and then a Stall Routine is performed. On the other hand, if PA0 is initially low, then the blinds will be lowered the same height as one press of the button L on the keypad would perform. Then, the input of PA0 will be checked again. The lowering continues until PA0 is high, which indicates the blinds have been lowered all the way—again when a high is read, register Y will be set to 0 and a Stall Routine is performed.

<u>Stall Routine (*stall*)</u>


START
Stall circuits from reading inputs.
Automation Switch Routine.

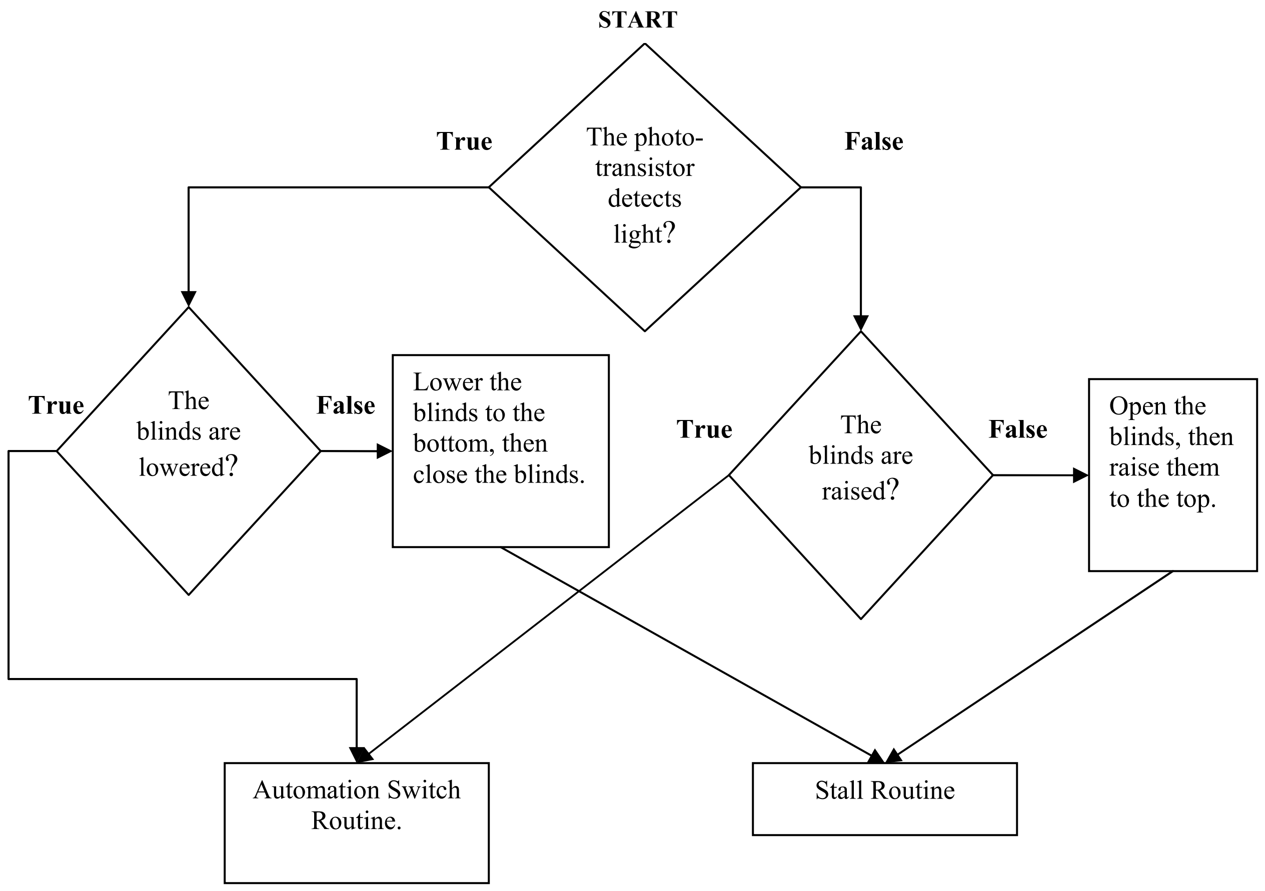
The Stall Routine performs a stall by performing 4095 nop commands. This takes place to prevent any motor from changing directions too quickly. Once the nop commands have been performed, the Automation Switch Routine is performed.

Automation Switch Routine (within *bcheck)*



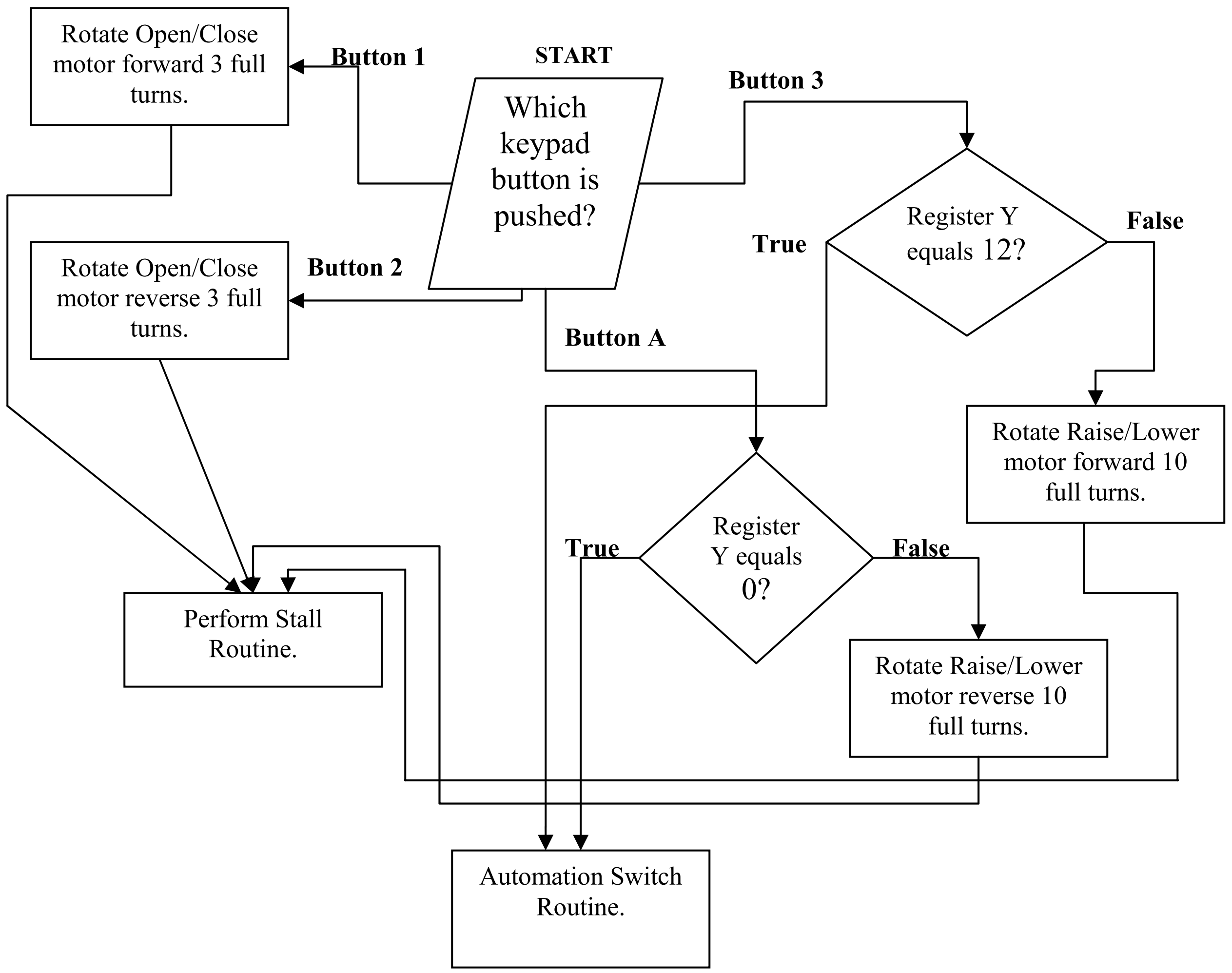
The Automation Switch Routine checks whether or not the automation switch is on, in turn determining whether or not to analyze the phototransistor readings. If the automation switch is on, then the Photo Routine is performed, otherwise the Keypad Routine is performed.

Photo Routine (*photo*)


START
True
The photo-transistor detects light?
False
True
The blinds are lowered?
False
Lower the blinds to the bottom, then close the blinds.
True
The blinds are raised?
False
Open the blinds, then raise them to the top.
Automation Switch Routine.
Stall Routine

The Photo Routine determines which operations will be performed on the blinds depending on the amount of light detected by the phototransistor. It accomplishes this by analyzing the input of PA2. If PA2 is high, this indicates that the phototransistor is not detecting much light. When there is little light, it is desired that the blinds are open and raised. If register Y equals 12—indicating the blinds are at the top of the window—then no action takes place on the blinds and the Automation Switch Routine is performed. If register Y equals 0—an indication the blinds are at the bottom of the window—then two actions take place. The blinds are opened and then the blinds are raised to the very top of the window, then a Stall Routine takes place. On the other hand, if PA2 is low, this indicates that the phototransistor is detecting a significant amount of light. When this happens, it is desired that the blinds are closed and lowered. If register Y equals 12 then two actions take place. The blinds are first lowered to the bottom of the window and then they are opened, which is also followed by a Stall Routine. If register Y equals 0, then no action takes place and the Automation Switch Routine is performed.

<u>Keypad Routine (within *bcheck*)</u>


START
Which keypad button is pushed?
Button 1
Rotate Open/Close motor forward 3 full turns.
Button 2
Rotate Open/Close motor reverse 3 full turns.
Button 3
Register Y equals 12?
True
False
Rotate Raise/Lower motor forward 10 full turns.
Button A
Register Y equals 0?
True
False
Rotate Raise/Lower motor reverse 10 full turns.
Perform Stall Routine.
Automation Switch Routine.

The Keypad Routine checks for which button of the keypad was pushed. If the 1 button was pushed, then the open/close motor turns forward 3 full turns and then performs a Stall Routine. If the 2 button was pushed, then the open/close motor turns forward 3 full turns and then performs a Stall Routine. If the A button was pushed, then the program checks if Register Y equals 0. If this is true, then the blinds are already at the bottom of the window, so no action takes place and the Automation Switch Routine is performed. Otherwise, the raise/lower motor turns forward 10 full turns and then performs a Stall Routine. If the 3 button is pressed, then the program checks if Register Y equals 12. If this is true, then the blinds are located at the top of the window, thus nothing is performed on the blinds and the automatic Switch Routine is performed. Otherwise, the raise/lower motor rotates in reverse for 10 full turns.

The code to implement the model was written in Assembly using TeXAS version 1.23. Given the hardware setup, the following Assembly code was able to implement the Automated Blinds Project.

SOFTWARE LISTING

```
PORTA equ $0000     ; PORTA is an alias to the memory location $0000
PORTD equ $0008     ; PORTD is an alias to the memory location $0008
DDRD  equ $0009     ; Data Direction Register for port D - alias to the
                    ; memory location $0009
PACTL equ $0026     ; PACTL is an alias to the memory location $0026
CONFIG equ $003F    ; define config register, contains watch dog timer at bit 3
Main   org $F000    ; main program is stored starting from location $F000
       ldaa #$04    ; store 00000100 at location $003F to disable the
       staa CONFIG  ; Watch Dog Timer
       ldaa #$30    ; store 00110000 at location $0009 to configure PD0, PD1, PD2,
                    PD3 as input
       staa DDRD    ; and PD4, PD5 as outputs, 0 means input 1 means output
       ldaa #$88    ; assign PA7, PA3 as outputs, 1 means output, 10001000
       staa PACTL   ;
       jmp initial  ; go intialize blinds
intdoneldy #$00     ; use y register as counter for location of blinds, load with 0
       jmp stall    ; stalls after initialize complete, then checks buttons
bcheck ldaa #$02    ; load 00000010 into accumulator, checks to see if automation is
                    switched on
       ldab PORTA
       andb #02     ; and with 00000010 to only evalute PA1
       CBA          ; compare A to B, if equal then switch is turned on
       beq photol   ; branch to photo loop to determine whether or not to raise and
                    lower blinds
       ldaa #$0F    ; load 00001111 into accumulator, which represents no buttons are
                    being pressed
       ldab PORTD   ; load data from PORTD into accumulator b
       andb #$0F    ; AND B with 00001111, to make zero all outputs but maintain all
                    input values
       CBA          ; compare A to B
       beq bcheck   ; if A equals B, then it means there are no buttons being pressed,
                    branch to bcheck
       ldaa #$0E    ; load 00001110 into accumulator, which represents "open blind"
```

```
        ldab PORTD
        andb #$0F       ; and B with 00001111, to make zero all outputs but maintain all
                        input values
        CBA             ; compare A to B
        beq open        ; if they are equal, open button pushed and go to open loop
        ldaa #$0D       ; load 00001101 into accumulator, which represents "close blind"
        ldab PORTD
        andb #$0F       ; and B with 00001111, to make zero all outputs but maintain all
                        input values
        CBA             ; compare A to B
        beq close       ; if they are equal, close button pushed and branch to close
        ldaa #$0C       ; load 00001100 into accumulator, which represents "raise blind"
        ldab PORTD
        andb #$0F       ; and B with 00001111, to make zero all outputs but maintain all
                        input values
        CBA             ; compare A to B
        beq raise       ; if they are equal, raise button pushed and branch to raise
        ldaa #$0B       ; load 00001011 into accumulator, which represents "lower blind"
        ldab PORTD
        andb #$0F       ; and B with 00001111, to make zero all outputs but maintain all
                        input values
        CBA             ; compare A to B
        beq lower
        bra bcheck      ; always go back to see what button is being pushed

photol  jmp photo       ; pc relative addressing range error, goes to photo

open    jmp openl       ; used because too long of a PC relative address to get to openl and
                        closel for BNE
close   jmp closel

lowint  ldab #$03       ; counter for total number of turns of stepper motor
reverse ldaa #$30       ; represents 00110000
        staa PORTD
        ldx #$2710      ;10000 decimal value
inner1  nop
        dex
        bne inner1
        ldaa #$20       ; next step, represents 00100000
        staa PORTD
        ldx #$2710
inner2  nop
        dex
        bne inner2
        ldaa #$00       ; next step, represents 00000000
        staa PORTD
        ldx #$2710
inner3  nop
        dex
        bne inner3
        ldaa #$10       ; next step, represents 00010000
        staa PORTD
        ldx #$2710
inner4  nop
        dex
        bne inner4
        decb            ; decrement b
        bne reverse     ; keep doing loop until b is 0
        jmp initial     ; go check if blinds are down yet

raise   cpy #$000C      ; make sure blinds aren't all the way raised, 12 times, if so check
                        for next button
        beq bcheck      ;
        iny             ; increment y because lowered blinds
        ldab #$0A       ; use b as a counter, 10 turns of stepper motor
raiser  ldaa #$30       ; represents 00110000
        staa PORTD
        ldx #$2710      ;10000 decimal value
raise1  nop
        dex
        bne raise1
        ldaa #$10       ; next step, represents 00010000
        staa PORTD
        ldx #$2710
raise2  nop
        dex
        bne raise2
        ldaa #$00       ; next step, represents 00000000
        staa PORTD
        ldx #$2710
raise3  nop
        dex
        bne raise3
        ldaa #$20       ; next step, represents 00100000
        staa PORTD
        ldx #$2710
raise4  nop
        dex
        bne raise4
        decb            ; decrement b
        bne raiser      ; keep doing loop until b is 0
        jmp stall       ; go check next button being pushed
```

```
lower   cpy #$0000    ; make sure blinds aren't all the way down, if so check for next
                      button
        beq bcheckl   ;
        dey           ; decrement y because lowered blinds
        ldab #$0A     ; use b as a counter, 10 turns of stepper motor
lowerr  ldaa #$30     ; represents 00110000
        staa PORTD
        ldx #$2710    ;10000 decimal value
lower1  nop
        dex
        bne lower1
        ldaa #$20     ; next step, represents 00100000
        staa PORTD
        ldx #$2710
lower2  nop
        dex
        bne lower2
        ldaa #$00     ; next step, represents 00000000
        staa PORTD
        ldx #$2710
lower3  nop
        dex
        bne lower3
        ldaa #$10     ; ext step, represents 00010000
        staa PORTD
        ldx #$2710
lower4  nop
        dex
        bne lower4
        decb          ; decrement b
        bne lowerr    ; keep doing loop until b is 0
        jmp stall     ; go check next button being pushed

openl   ldab #$03     ; use b as a counter, 3 turns of stepper motor
openr   ldaa #$88     ; represents 10001000
        staa PORTA
        ldx #$2710    ;10000 decimal value
open1   nop
        dex
        bne open1
        ldaa #$08     ; next step, represents 00001000
        staa PORTA
        ldx #$2710
open2   nop
        dex
        bne open2
        ldaa #$00       ; next step, represents 00000000
        staa PORTA
        ldx #$2710
open3   nop
        dex
        bne open3
        ldaa #$80       ; next step, represents 10000000
        staa PORTA
        ldx #$2710
open4   nop
        dex
        bne open4
        decb            ; decrement b
        bne openr       ; keep doing loop until b is 0
        jmp stall       ; go check next button being pushed

bcheckl  jmp bcheck     ; just for lower, so it will go to bcheck, not enough address space

closel  ldab #$03       ; use b as a counter, 3 turns of stepper motor
closer  ldaa #$88       ; represents 10001000
        staa PORTA
        ldx #$2710      ;10000 decimal value
close1  nop
        dex
        bne close1
        ldaa #$80       ; next step, represents 10000000
        staa PORTA
        ldx #$2710
close2  nop
        dex
        bne close2
        ldaa #$00       ; next step, represents 00000000
        staa PORTA
        ldx #$2710
close3  nop
        dex
        bne close3
        ldaa #$08       ; next step, represents 00001000
        staa PORTA
        ldx #$2710
close4  nop
        dex
        bne close4
        decb            ; decrement b
        bne closer      ; keep doing loop until b is 0
        jmp stall       ; go check if blinds are down yet

bcheck2 jmp bcheck      ; for phototransistor branches

stall   ldx #$0FFF      ; decimal value 4095
stalli  nop
        dex
        bne stalli
        jmp bcheck      ; stall completed, go check next button being pushed

lowintl jmp lowint      ; jumps to low int, because of PC relative problem

initial ldaa PORTA      ; load porta to check if blinds are down
        anda #$01       ; and with 00000001 to maintain only PA0 input
        ldab #$00       ; load b with 00000000
        CBA             ; compare A with B
        beq lowintl     ; if they are equal then blinds aren't lowered
        jmp intdone     ; otherwise blind is initialized

photo   ldaa #$04       ; 00000100 to see if PA2 is high (which means no light) and open
                        and raise blinds
        ldab PORTA      ;
        andb #$04       ; AND accumulator B with 00000100 to only maintain PA2 input
        CBA             ;
        beq photonl     ; they are equal, no light, go open and raise blinds
        cpy #$0000      ; compare y to 0
        beq bcheckl     ; if they are equal, blinds are lowered all the way, so don't do
                        anything
        ldab #$80       ; use b as a counter, 10 turns of stepper motor  * 13 loops = 130
                        (should be 82)
        ldy #$0000      ; this ensures that it will know it's at the bottom after it has lowered
plowr   ldaa #$30       ; represents 00110000
        staa PORTD
        ldx #$2710      ;10000 decimal value
plowr1  nop
        dex
        bne plowr1
        ldaa #$20       ; next step, represents 00100000
        staa PORTD
        ldx #$2710
plowr2  nop
        dex
        bne plowr2
        ldaa #$00       ; next step, represents 00000000
        staa PORTD
        ldx #$2710
```

```
plowr3 nop
        dex
        bne plowr3
        ldaa #$10       ; next step, represents 00010000
        staa PORTD
        ldx #$2710
plowr4 nop
        dex
        bne plowr4
        decb            ; decrement b
        bne plowr       ; keep doing loop until b is 0
        ldx #$0FFF      ; for waste
waste2 nop              ; kill some time
        dex             ;
        bne waste2      ; branch to waste2
        ldab #$08       ; use b as a counter, 3*4 = 12 turns of stepper motor, 8 is good
pcloser ldaa #$88       ; represents 10001000
        staa PORTA
        ldx #$2710      ;10000 decimal value
pclose1         nop
        dex
        bne pclose1
        ldaa #$80       ; next step, represents 10000000
        staa PORTA
        ldx #$2710
pclose2         nop
        dex
        bne pclose2
        ldaa #$00       ; next step, represents 00000000
        staa PORTA
        ldx #$2710
pclose3         nop
        dex
        bne pclose3
        ldaa #$08       ; next step, represents 00001000
        staa PORTA
        ldx #$2710
pclose4         nop
        dex
        bne pclose4
        decb            ; decrement b
        bne pcloser     ; keep doing loop until b is 0
        jmp stall       ; always stall

bcheck3 jmp bcheck
```

```
photonl cpy #$000C   ; compare y to12, the ceiling of the blinds height
        beq bcheck3  ; equal, so don't raise blinds jump to bcheck3, which goes to
bcheck
        ldy #$000C   ; load y with C now, to show it's going to the top
popenl  ldab #$08    ; use b as a counter, 12 turns of stepper motor (3 turns * 4 times )
                     (8 for now)
popenr  ldaa #$88    ; represents 10001000
        staa PORTA
        ldx #$2710   ;10000 decimal value
popen1  nop
        dex
        bne popen1
        ldaa #$08    ; next step, represents 00001000
        staa PORTA
        ldx #$2710
popen2  nop
        dex
        bne popen2
        ldaa #$00    ; next step, represents 00000000
        staa PORTA
        ldx #$2710
popen3  nop
        dex
        bne popen3
        ldaa #$80    ; next step, represents 10000000
        staa PORTA
        ldx #$2710
popen4  nop
        dex
        bne popen4
        decb         ; decrement b
        bne popenr   ; keep doing loop until b is 0
        ldx #$0FFF
waste   nop          ; kill some time
        dex          ;
        bne waste    ; branch to waste
        ldab #$80    ; use b as a counter, 10 turns of stepper motor  * 13 loops = 130
                     (should be 82)
praiser ldaa #$30    ; represents 00110000
        staa PORTD
        ldx #$2710   ;10000 decimal value
praise1 nop
        dex
        bne praise1
        ldaa #$10    ; next step, represents 00010000
        staa PORTD
        ldx #$2710
praise2 nop
        dex
        bne praise2
        ldaa #$00       ; next step, represents 00000000
        staa PORTD
        ldx #$2710
praise3 nop
        dex
        bne praise3
        ldaa #$20       ; next step, represents 00100000
        staa PORTD
        ldx #$2710
praise4 nop
        dex
        bne praise4
        decb            ; decrement b
        bne praiser     ; keep doing loop until b is 0
        jmp stall       ; go check next button being pushed
        jmp bcheck      ; repeat the process forever
        org $FFFE       ; point the PC to location FFFE and store the Main
        dc.w Main       ; program Address
        end             ; when 6811 resets, it points to FFFE and replaces the
                        ;current value of PC with the contents at location FFFE &FFFF
```

TESTING

Testing was a critical factor to the success of the Blinds Project. Physical materials, hardware circuits and software were all tested and retested to make certain the blinds project would perform as desired.

Physical Blinds Testing

Once the blinds were constructed, only testing whether or not the raise/lower motor would be able to support the weight of the blinds was tested. This was done by simply tying the strings of the blinds to the shaft of the motor. Then, the motor was rotated until the blinds reached the top of the window; a position where the greatest amount of torque was required. Luckily, the motor was able to support the weight and there were no more tests needed.

Hardware Circuits Testing

The entire circuit created for the Blinds Project was built by attaching subcomponents individually. The major components were the motor driver, keypad, windowsill switch, phototransistor and automation switch. First, the keypad was constructed and attached to the circuit. To verify that it was built correctly, LED's were attached to the pins that would eventually be attached to the motor driver control inputs. Then, each activated button was pushed to see if it caused the LED's to flash, which verifies a correct response to the keypad input. Next, the two motor drivers were built. The motor drivers were tested by attaching a motor to each one. Then, the motor driver inputs were pulsed in sequences by hand by plugging them in and out of 5V and ground. The motor rotated in the correct direction, and was deemed ready. The windowsill switch was also tested before being attached to the circuit. To verify that it worked, a DMM was used to measure the voltage from pin 2 of the toggle switch. Pin 2 showed a reading of 5V when the switch was pressed and 0V otherwise, thus it was verified as working. A DMM was also used to test if the phototransistor circuit was working correctly. When the phototransistor detected light, a 2.5 voltage drop was measured across the resistor. When the phototransistor didn't detect light, no voltage drop occurred across the resistor and the phototransistor circuit was considered working. Finally, the automation switch was tested and attached to the entire circuit. A DMM was used to measure the voltage at pin 2 of the switch. 5 V was measured at pin 2 when the switch was on and 0V otherwise, thus it was constructed correctly. By verifying the subcomponents individually before attaching them, the cause of hardware circuit errors were easier to discover and much easier to rectify.

Software Testing

The assembly code was written and tested using TExaS Version 1.23. Throughout the process of creating the blinds, the assembly code was revised various times to accomplish the project goals. However, upon completion of each revision, the assembly was simulated and tested within TExaS. With TExaS, it was possible to attach I/O devices to specified ports of a simulated 68HC711D3 microprocessor in order to reflect the actions and response of hardware. To analyze if the motor driver inputs were receiving the correct pulse sequence, LED's were attached to the appropriate simulated pins. When the LED lit up, it was known that a high output was provided from that pin, when the LED was off, it was known that a low output was provided. To analyze if the microprocessor responded correctly to keypad inputs, switches were attached to the appropriate simulated pins. When the switch was closed, it was known the pin read a high input value, when the switch was open it was known the pin read a low input value. Overall, the simulator provided an efficient way to test the software and saved a lot of frustrating lab time.

PROBLEMS AND SOLUTIONS

Construction Problems

Windowsill switch

1. **Problem**: The contact between the blinds and the windowsill switch was not always detected during the initialization sequence. Due to this, the blinds would raise after passing the bottom of the window because the raise/lower motor would continue to rotate in reverse. The contact wasn't always recognized for two reasons. First, the blinds could not balance on top of the small surface area of the switch, thus requiring the software to check the switch at the exact moment the blinds hit the switch. Second, the blinds were lowered in large increments only permitting a small frequency for the switch to be checked.
   Solution: A large plate was connected to the windowsill switch in order to create a larger surface for the blinds to contact and remain balanced upon. Also, the

lowering of the blinds was performed in smaller increments, increasing the frequency that the switch was checked.

Open/Close motor connection

2. **Problem**: The connection between the shaft of the motor and the rod that opened and closed the blinds wasn't centered and wouldn't turn. At first, the connection between the shaft of the open/close motor and the rod consisted of two, different sized diameter, hollow, aluminum poles. One end of the smaller aluminum pole was attached to the shaft of the open/close motor and the other end was JB welded inside a larger aluminum pole. The other end of the larger aluminum pole was clamped onto the rod which turned to open/close the blinds. However, the small aluminum pole wasn't centered within the larger aluminum pole, causing the rod not to turn correctly.
   Solution: A plastic ribbed anchor was used with only the small aluminum pole to connect the motor shaft to the rod. The larger aluminum pole was eliminated because it just caused more problems. A longer piece of the smaller aluminum pole was attached to the motor shaft. A plastic ribbed anchor was screwed into the other end of the aluminum pole. The ribbing on the anchor ensured that the pole and the anchor turned in unison. The other end of the anchor contained a center hole, usually used for a screw. Since this hole was centered, the rod was jammed into the hole without any worry of it not turning correctly.

Raise/Lower motor connection

3. **Problem:** The blinds did not always rise evenly because of the connection between the strings that raised the blinds and the raise/lower motor shaft. This was caused for two main reasons. First, after attaching the strings to the motor shaft, there wasn't a smooth surface for the strings to wind around. However, a smooth surface is required so that the strings wrap around the same size circumference, in turn raising the blinds evenly. The second obstacle was that the strings didn't always wrap around symmetrically on the motor shaft, again, preventing an even rise of the blinds.
   Solution: The strings were attached to the motor shaft using electrical tape. Then, a larger hollow aluminum pole slid over the strings and electrical tape to provide a smooth surface for the strings to wind around. Also, a small, angled, metallic plate split the center of the aluminum pole to maintain symmetric winding of the strings. In other words, the metallic obstruction prevented the strings from crossing the center of the aluminum pole.

<u>Circuit Problems</u>

Current supply for bipolar motors

1. **Problem**: When purchasing the bipolar motors, only the holding torque specification was considered. It was discovered later that each bipolar motor requires 1 A of current, however, an H-Bridge that provides such a high load current is difficult to find.
   Solution: The SN754410 quad half-bridge chip was found on the Internet and purchased. This chip provides a load current of 1A.

Output signals not recognized from microprocessor

2. **Problem**: The control inputs for the SN754410 circuit from the 68HC711D3 microprocessor weren't recognized.
   Solution: It was discovered that the output signals from the microprocessor weren't strong enough for the hex inverter to detect. Therefore, the output signals passed through a DM74LS244 Octal 3-state Buffer before going into the hex inverter.

Not enough voltage and current applied to motor drivers

3. **Problem**: The bipolar motors did not have enough torque while raising the blinds.
   Solution: A separate power supply was needed for each motor driver in order to provide sufficient current and voltage for the SN754410 circuits.

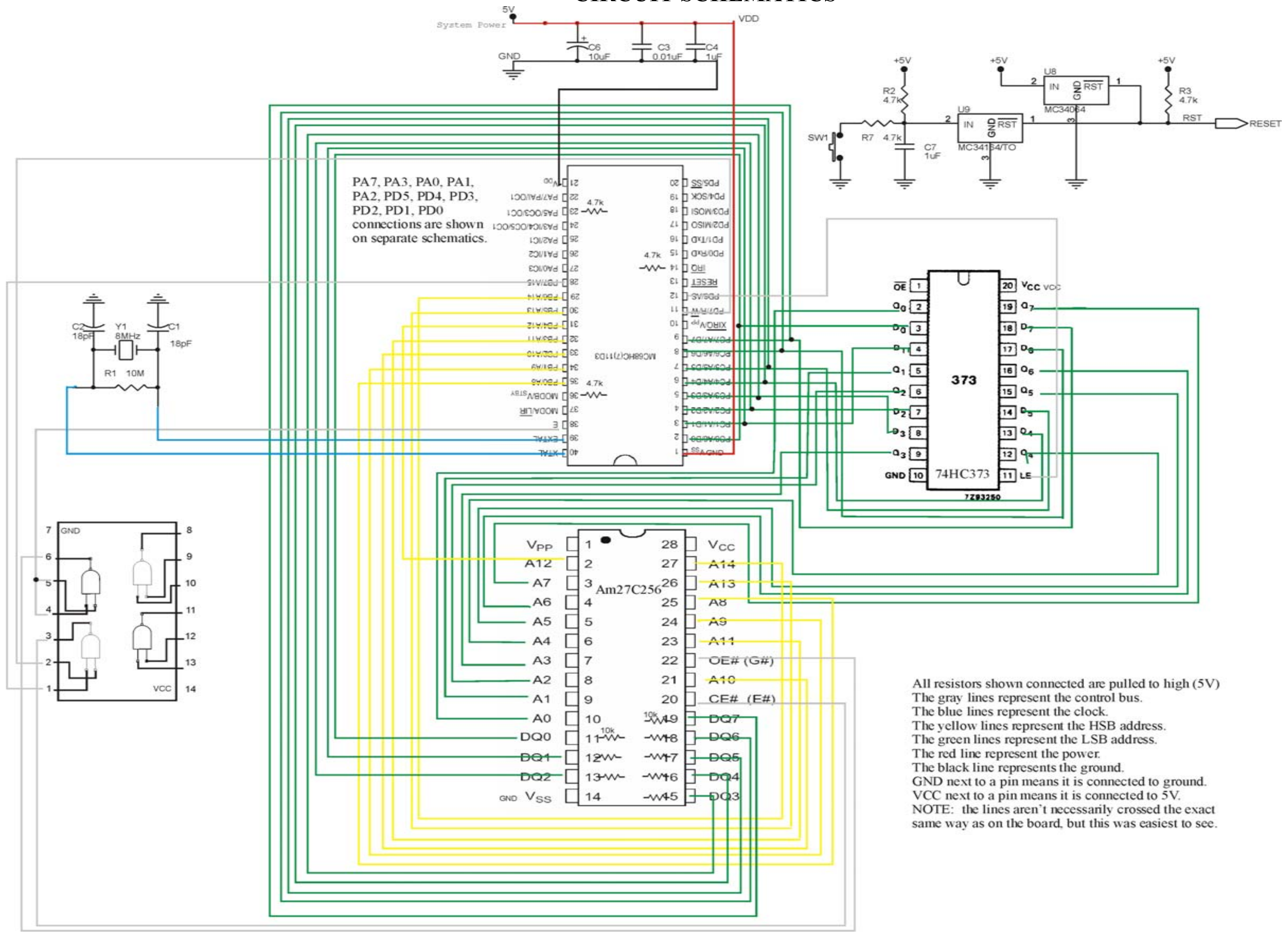
Common ground wasn't implemented

4. **Problem**: The bipolar motor did not turn correctly, even though the correct input sequence to controls A and B of the motor driver was applied.
   Solution: A common ground for the entire circuit was created. Earlier, each motor driver had its own power supply and its own ground, and this was the source of the problem.

CONCLUSION

All of the goals for this project were accomplished, including the optional automation of the blinds. A user was able to open, close, raise or lower the blinds by pushing unique buttons on a keypad. A toggle switch on the bottom of the windowsill allowed the location of the blinds to be tracked in order to prevent trying to raise or lower the blinds too much. Furthermore, by flipping a switch mounted on the blinds, the blinds became automated and would lower and close or open and rise depending on the amount of surrounding light. The keypad, bipolar motors, switches and phototransistor successfully communicated with the 68HC711D3 microprocessor to fulfill the goals of this project. Even though the Automated Blinds Project did require an abundant amount of lab time and troubleshooting, the project's eventual success was extremely gratifying. Each group member feels as though they have learned a tremendous amount about the demanding process of creating a project.

CIRCUIT SCHEMATICS



All resistors shown connected are pulled to high (5V)
The gray lines represent the control bus.
The blue lines represent the clock.
The yellow lines represent the HSB address.
The green lines represent the LSB address.
The red line represent the power.
The black line represents the ground.
GND next to a pin means it is connected to ground.
VCC next to a pin means it is connected to 5V.
NOTE: the lines aren't necessarily crossed the exact same way as on the board, but this was easiest to see.

Figure 8 – Main Board Schematic

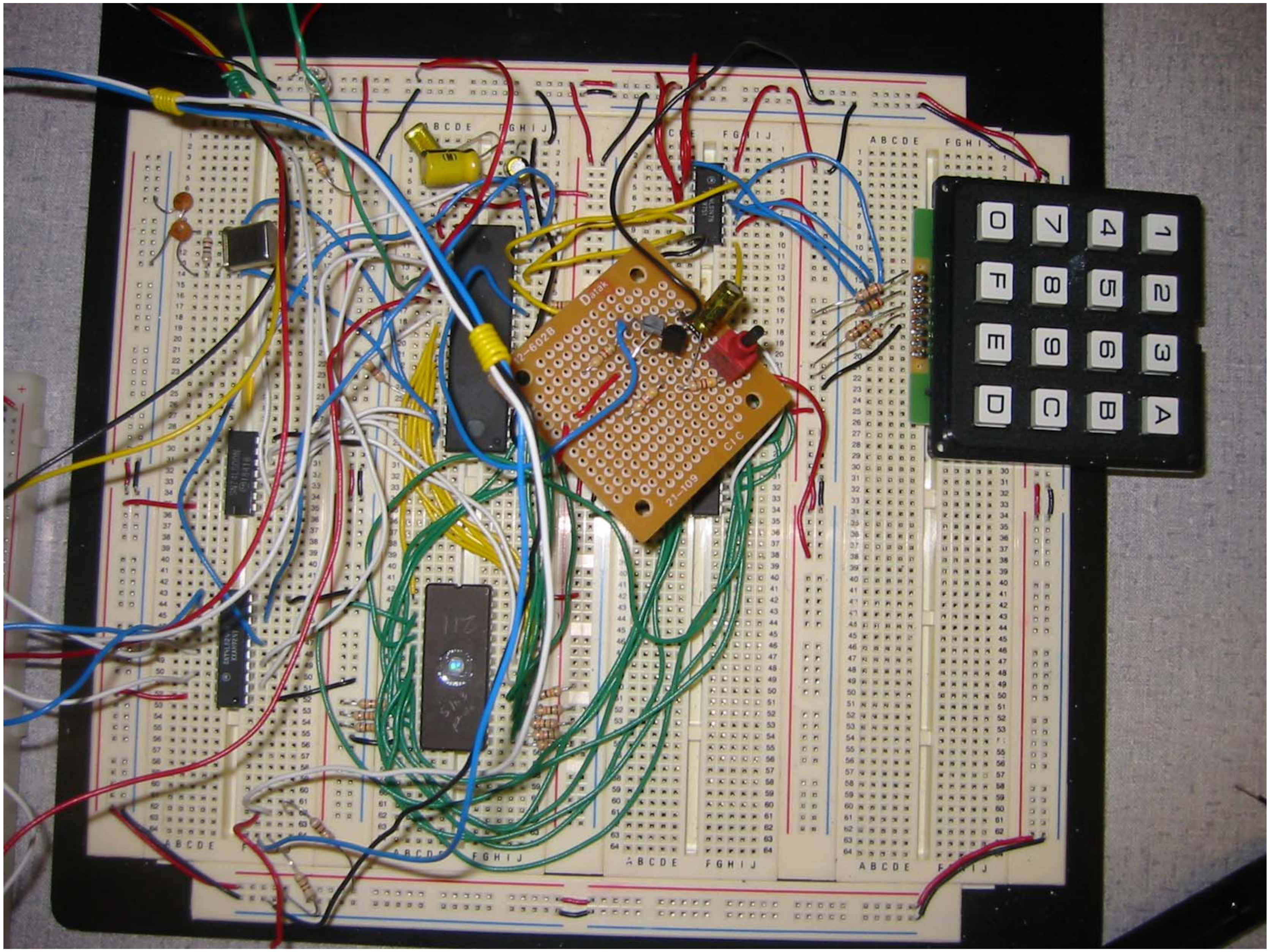
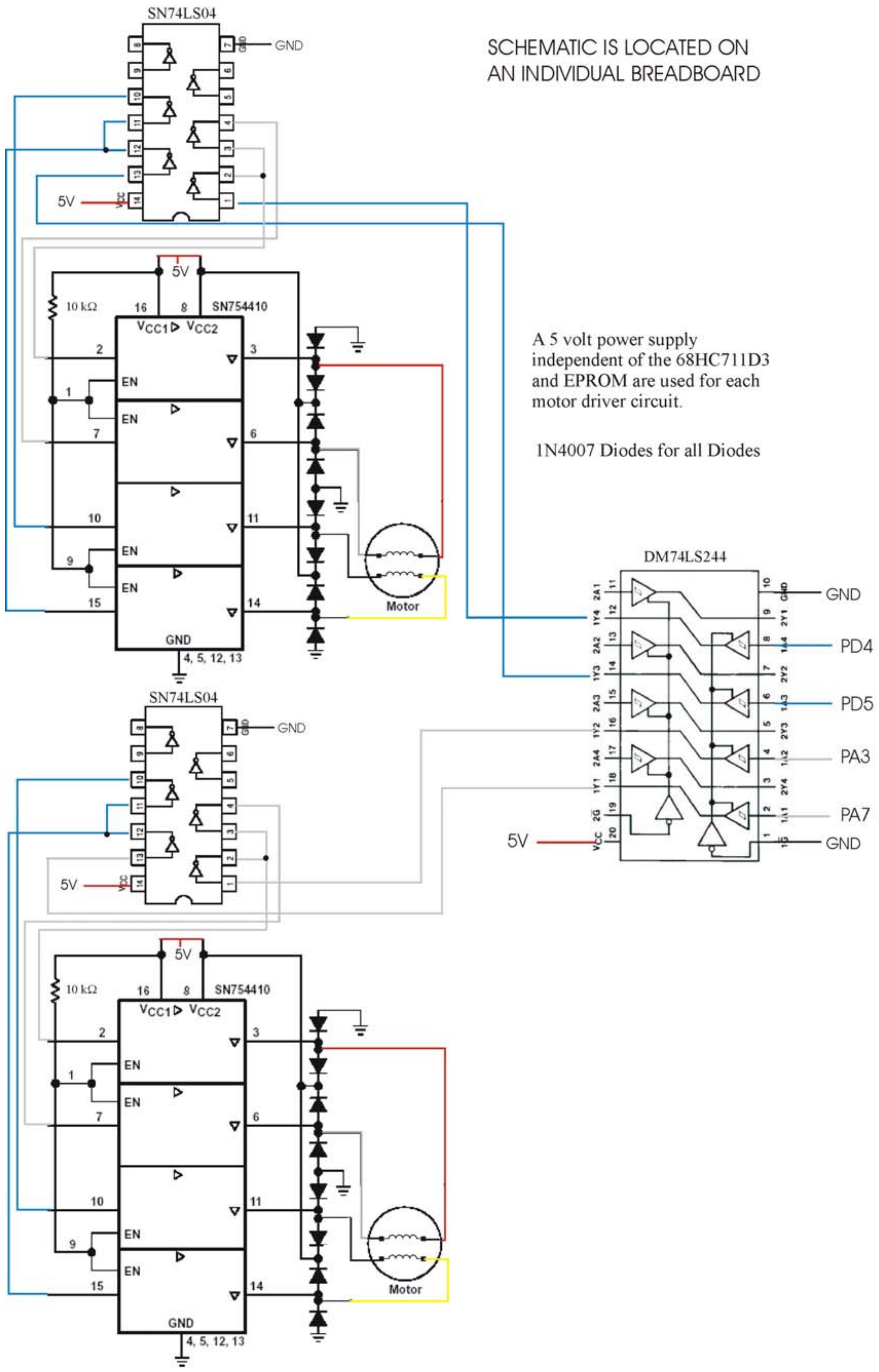


Figure 9 – Picture of main board



SCHEMATIC IS LOCATED ON AN INDIVIDUAL BREADBOARD

A 5 volt power supply independent of the 68HC711D3 and EPROM are used for each motor driver circuit.

1N4007 Diodes for all Diodes

Figure 10 – Schematic of motor drivers

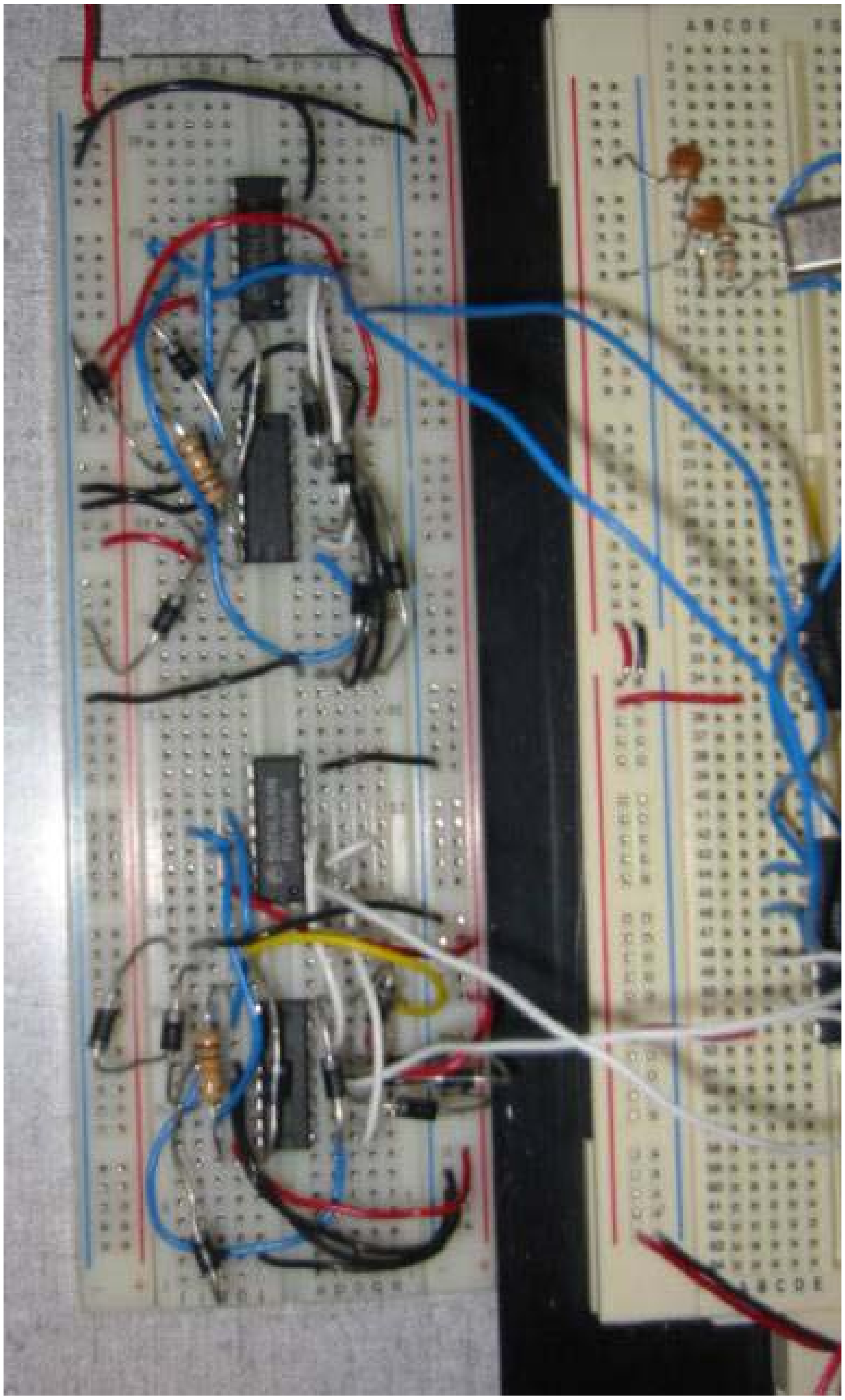


Figure 11 – Picture of motor drivers

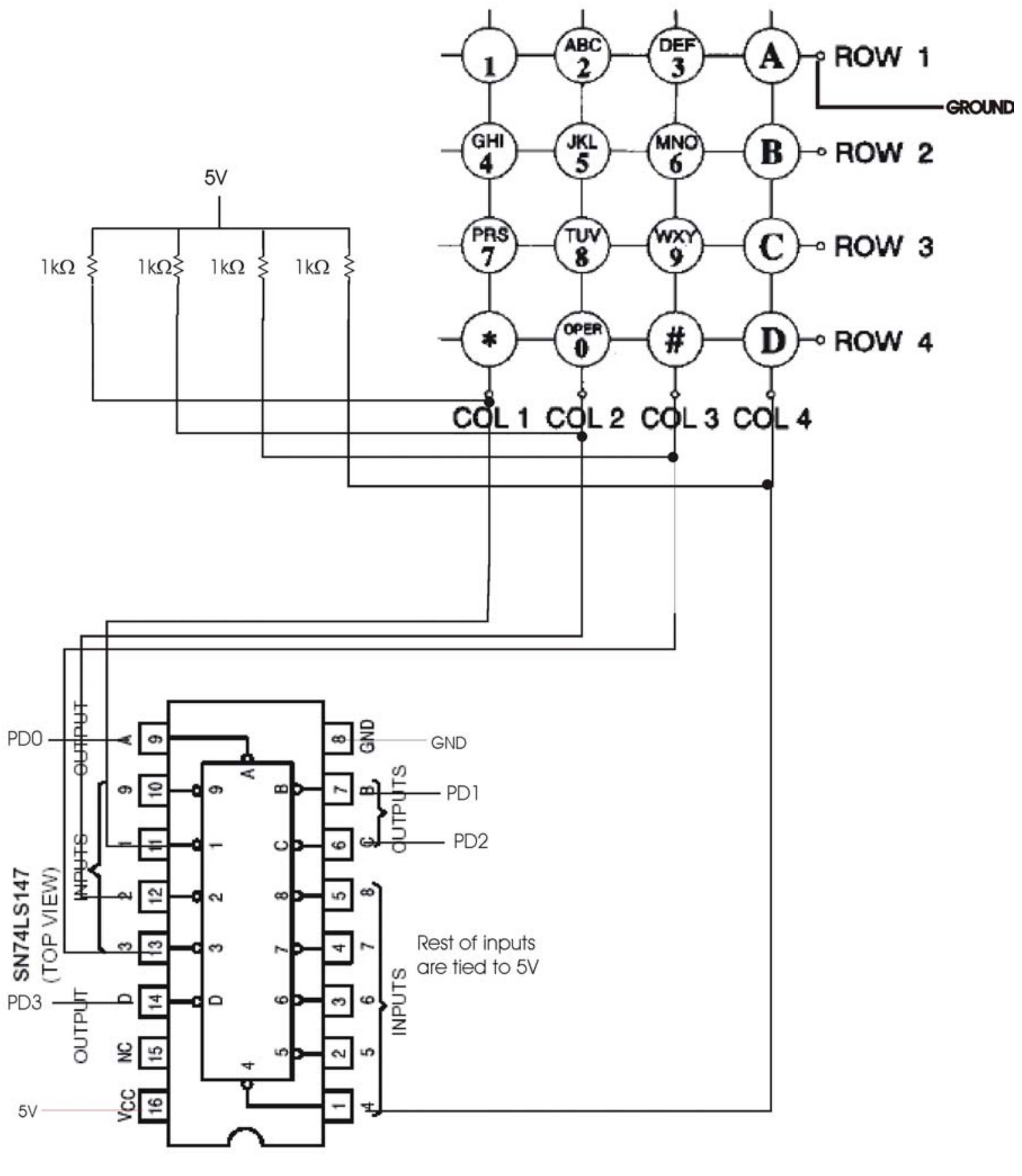


Figure 12 – Schematic of keypad

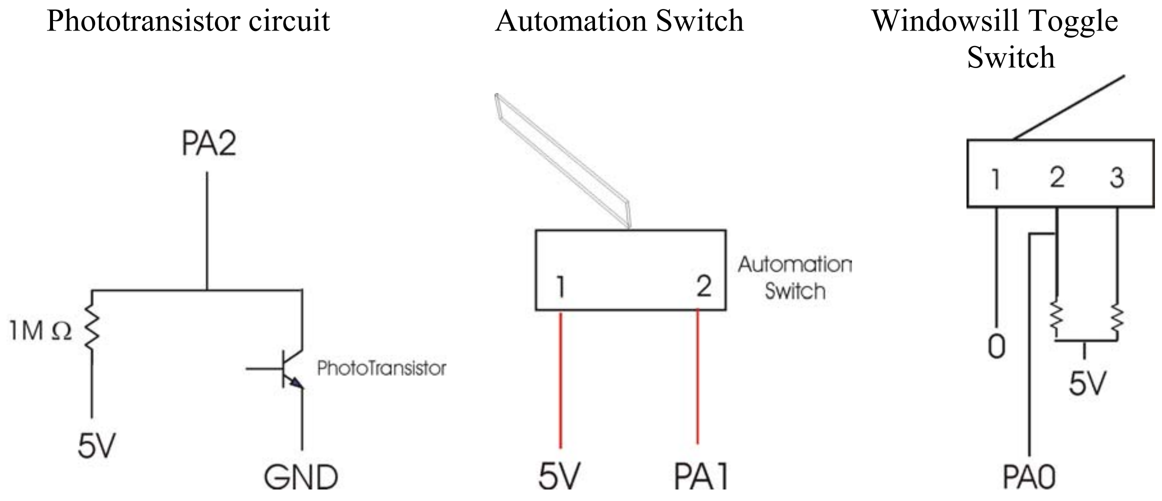


Figure 13 – Schematics of switches and phototransistor

COMPLETE PARTS LIST AND BUDGET

Part Name	Part Number	Quantity	Cost	Location
Microcontroller	68HC711D3	1	Free	ECE Stockroom
EPROM	Am27C256	1	Free	ECE Stockroom
Quad 2-Input NAND Gate	DM74LS00	1	Free	ECE Stockroom
8 MHz Clock	--	1	Free	ECE Stockroom
Diode	1N4007	16	Free	ECE Stockroom
Hex Inverter	SN74LS04	2	Free	ECE Stockroom
Octal 3-state Buffer	DM74LS244	1	Free	ECE Stockroom
10-line to-4 line Encoder	SN74LS147	1	Free	ECE Stockroom
Octal D-type Latch	74HC373	1	Free	ECE Stockroom
Undervoltage Sensing Circuit	MC34064	1	Free	ECE Stockroom
Micropower Undervoltage Sensing Circuit	MC34164	1	Free	ECE Stockroom
Reset Switch	SW1	1	Free	ECE Stockroom
Numeric Keypad			Free	ECE Stockroom
Photo Transistor	IFD-5	1	Free	ECE Stockroom
Undervoltage circuit	MC34064	1	Free	ECE Stockroom
Micropower undervotlage circuit	MC34164	1	Free	ECE Stockroom
Capacitors	Various	Several	Free	ECE Stockroom
Resistors	Various	Several	Free	ECE Stockroom
Wires	Various	Several	Free	ECE Stockroom
Plastic Ribbed Anchor	--	1	Free	ACE Hardware
Spray Paint	--	1 can	Free	Leftover paint
Bipolar Stepper Motor	55M048D1B	2	$36.00	Digikey.com
Quad Half H-Driver	SN754410	2	$5.40	Digikey.com
Lightweight Blinds, Hardware for window frame, wood for window frame	--	--	$16.43	Home Depot
Aluminum/Copper tubing	--	1	$2.14	HobbyTown USA
JB Weld	--	--	$3.47	Home Depot
Lever and Toggle Switch	--	1 of each	$3.65	Elliot Electronics
			Total	$67.09