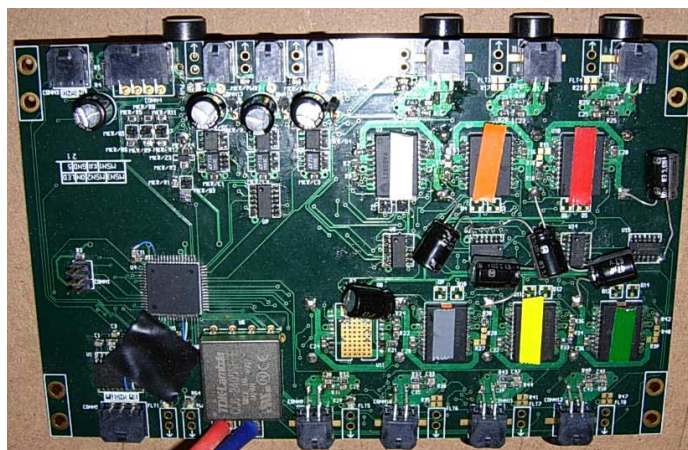


Actuator Control Data Sheet



Description:

The actuator control board communicates with the computer through a RS232 connection and listens for commands to drop markers, change thruster speed, or request current status. The mission control switch can override the computers requests and hard kill all the actuators.

The current design is able to control three marker droppers and seven thrusters. The marker droppers are either enabled or not and share a single fuse. Each thruster has an H-Bridge which has its direction and speed controlled by the microcontroller. Each thruster has its own fuse and current monitor.

Connections:

1-7. Thrusters :

1,2=PWRA (two pins for each signal because of the high power)
3,4=PWRB

8. Power from aft power:

1=Vbatt
2=GND

9-11. Droppers :

1,2=PWR1 (two pins for each signal because of the high power)
3,4=GND1
5,6=PWR2
7,8=GND2

12. Mission Switch :

1=5V
2=GND
3=KILL_L (hard kill)
4=Mission_Switch
5=Mission_LED
6=ON_L
7=Mission_Switch2 (these two might be used to select between multiple missions)
8=Mission_Switch3

13. On signal :

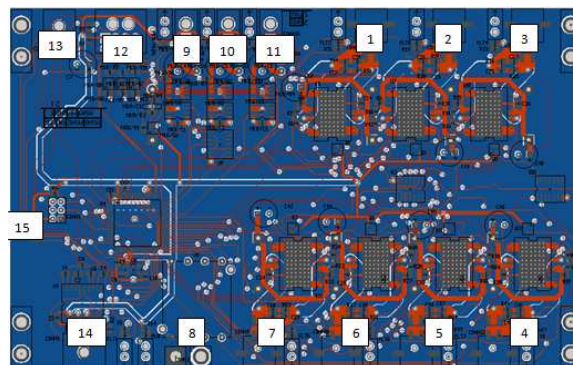
1=Iso GND
2=ON_L

14. Serial to computer:

1=RX
2=GND
3=TX

15. Programmer: (header not used during normal operation)

1=MISO
2=VCC
3=SCK
4=MOSI
5=RESET
6=GND



User Documentation

Installation:

The board should be placed away from sensors due to the amounts of interference from the relatively high voltage outputs the board produces. The connections for the board are diagrammed in the data sheet. During normal operation all the connections should be made aside from the programmer and one of the thrusters. To be compatible with this board any thrusters from Triton must have their internal boards bypassed before connecting.

Usage:

Once the board is hooked up the actuators are controlled by a daemon on the sub computer and the mission switch. An alternative control program will also be available for testing without the sub computer. The daemon takes the serial port that the actuator board is connected to as its input and monitors shared variables that describe the actuator states. It periodically sends packets to the board to keep it in the desired state. The speed can be either positive or negative to indicate direction and must have an absolute value less than 256. A low on the kill input from the mission switch will disable all actuators regardless of software state. There is a red LED associated with each thruster. These do not work in the first revision and should be ignored. In the second revision they indicate a blown or missing fuse.

Safety:

There are two main concerns for this board, battery voltage and thruster current. The max voltage input is 36V and there is no reverse voltage protection so mixing power and ground will kill the board. The H-bridges can take up to 5 amps continuously and should provide protection against higher spikes, but caution should be taken if there is reason to believe a thruster will be drawing currents higher than this.

Technical Documentation

Abstract:

This board allows the computer to monitor and control the actuators. Only minor changes were needed from last semester's design. First the Atmega64 had a nonstandard assignment of its ISP pins. This meant that the connections between the MCU and the ISP connector had to be revised. To reduce voltage swings on large speed changes 47uF capacitors were added to each thruster. The marker droppers were given separate connectors and a third was added. The marker dropper current monitoring was removed. Also to avoid the need for software PWM generation the number of thrusters was reduced to 7. To implement these changes and accommodate issues in connector spacing the layout for the board was redone from scratch using 4 layers instead of two to allow for wider traces.

Software for both the computer and the board itself is needed. The board uses an Atmega64 while the computer daemon is written in Python. The software uses the standard packet protocol used on the Triton thrusters to communicate with RS232.

Requirements:

- Use VBatt for both powering the actuators and the control logic
- take in computer level serial and parse it to determine thruster and marker dropper states (speed, direction, kill)
- isolate serial ground from battery ground
- provide power to thrusters to run them at desired speeds and direction
- provide power to marker droppers
- provide fail safes to regulate current to actuators and prevent damage to sub in event of failure
- talk to the mission switch powering it and getting hard kill
- not interfere with the sensors
- have H-bridge segments that can be easily replaced
- clearly indicate fault states (communication errors, blown fuses) to facilitate debugging

Previous Designs:

This board has gone through three revisions this semester. The first design (version 1) was a major step away from the Triton thruster control by combining all the actuator control on one board and using IC H-bridges. The first version was never produced due to concerns about connector placement and resistance from small traces. The second revision (version 2) required a complete redesign of the PCB and was produced and tested. Three problems were found and each was addressed in version 2.1:

- nonstandard ISP ports – while most AVR chips use their SPI pins for programming the Atmega64 uses the primary USART pins even though the ISP refers to them by SPI names. This was easily fixed by splicing traces and has been corrected in the new design.

- errors caused by floating outputs in LED error indicators – to interpret multiple fault sources a blown fuse indicator was ANDed with a fault indicator from the H-bridge for each thruster. These lines did not have the appropriate pull up resistors and did not work correctly. This has been corrected by simplifying the fault detection, reporting H-bridge faults to the MCU and having the LEDs only indicate blown fuses.
- brown out from large changes in thruster speed – This problem occurred do to limited current output from the bench power supplies being unable to handle the spikes from the motor speed jumping. This was fixed by adding 47uF capacitors decoupling the input to the H-bridges. Also as an extra precaution a ramping function was added to the firmware.

Design:

H-Bridge control –

I decided to use the integrated H-bridge MC33887 for the thruster control. It provided many useful features like fault detection and current monitoring, while being able to provide enough power for the thrusters.

The direction of the thrusters is controlled by the microcontroller by setting a single output high or low this signal go through a inverter to control both IN1 and IN2 on a H-bridge. This will set the bridge either forward or reverse.

The speed is controlled by setting the PWM duty cycle for each H-bridge's EN input. This must be below 10kHz due to limitations on the H-bridge.

Kill –

The hard kill line is controlled by the mission switch and is meant to disable all the actuators regardless of the microcontroller. For the thrusters this is accomplished by having this line control /D2 on the H-bridge. This input into the H-bridge sets the outputs to high impedance disabling the thruster. There must be a high input on the Kill_L line to enable the thrusters. In testing this is done by setting the microcontroller port that monitors Kill_L to an output.

Marker Dropper –

The marker dropper circuit is based on the one from Triton. It uses enable lines that have already been ANDed with the kill signal to trigger dropping. The three marker droppers share a fuse. A green LED indicates when the dropper is mid drop and a red LED indicates a blown fuse.

Current monitoring –

Current monitoring is done with the Atmega64's built in ADC. It uses the 1/375 scaled current output from each H-bridge. To measure this I used a 200 ohm resistor making a voltage range of 0-4 volts.

Power –

5 volts

I decide to go with the CC6-24 without a filtering circuit on it to provide 5 volts. I calculated the max power draw to be around 800mW which the CC6 meets. This device was fairly expensive, but most of the team is using TDK DC/DC's. Also unlike cheaper DC/DC's this one is shielded. I made the grounds common so the microcontroller can output a PWM to the motor IC's.

Battery

By far the biggest power draw from this board is the thrusters. The H-bridge has a limit of 5 amps to be continuously supplied. Besides that assuming the battery can give enough power, this board doesn't have any other major limitations on battery draw.

Serial –

Instead of the MAX233 I am using the ADM325. This does the voltage level conversion as well as isolating the signal ground.

Mission Switch –

The switch gets its power from this board as well as using it for communication with the rest of the sub. The switch controls kill as well as passing an ON_L signal to the aft power board. The mission switch now has 3 inputs to provide for the possibility of switching between multiple missions.

Status indicators –

The status LED's fall into 3 main categories: activity indicators, power indicators, and problem indicators. Each of the actuators has a power and problem indicator. The problem indicator is a red LED that can will be enabled by a blown fuse. The board's heartbeat will get an activity indicator as a blue LED. A green LED indicates that the board has power and each marker dropper has another to indicate power flowing to its connector.

Software:

The software is in two parts; the Atmega64 firmware and the sub daemon.

Firmware –

The firmware is relatively simple. When the thrusters are enabled timers 0,1, and 3 are used to produce the PWM signals in hardware to control the speed. Timer2 is used for any time driven events like the heartbeat and the ramp function which linearly adjusts the speed to a desired input of a set time interval. The bulk of the code is to parse and respond to packets from the sub computer. I used AVRlib to buffer incoming serial data which is then parsed during the main while loop.

Daemon –

The Daemon is currently very simple, it does not even check for responses from the actuator board. It simply updates the board periodically with desired speeds and kill states.

Known problems:

Currently the only glitch is the ADC on the populated version 2 board is not operational. It is unknown if this is from damage during testing or a design error. Since this is a noncritical feature it will only be worked on during the version 2.1 board testing.

Current Status:

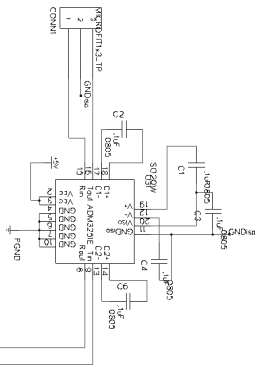
There is a fully operational version 2 board. The daemon and firmware are at a point where the sub can fully control the thrusters. Code needs to be added to handle the marker droppers, but hardware wise they are functional. The board has some rework and a nonfunctional ADC. Despite this it can be used in Nova and will be kept for testing and backup once the V2.1 board is finished. The V2.1 board and its parts have been ordered, but we are waiting for their arrival. The daemon, though full featured, could use some cleanup to make it watch for potential errors that it currently ignores.

Future improvements:

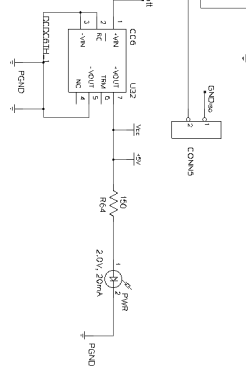
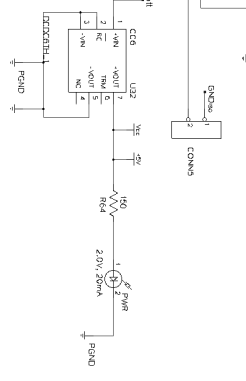
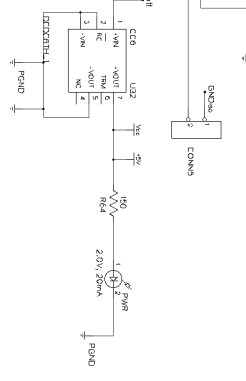
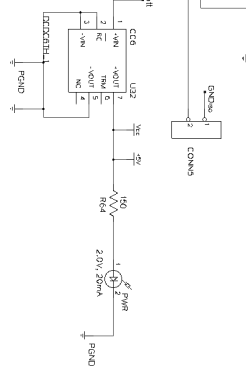
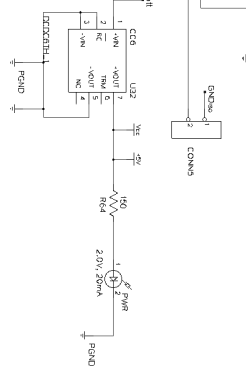
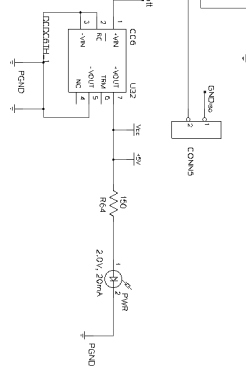
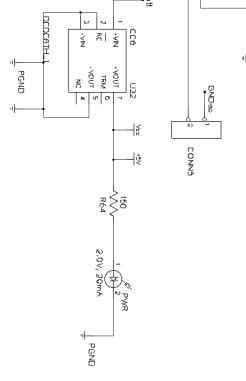
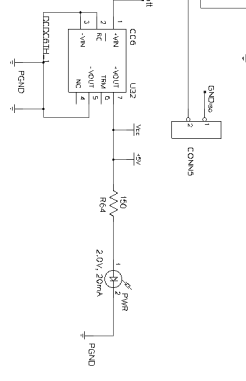
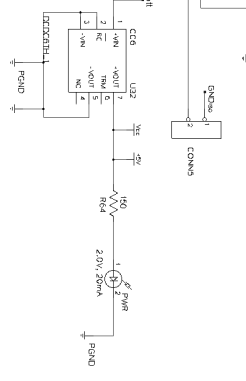
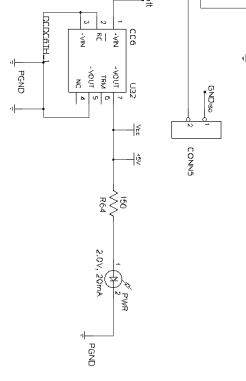
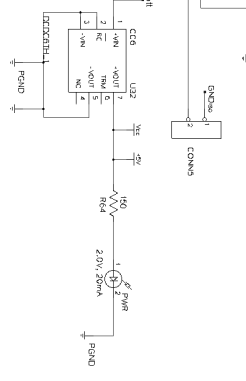
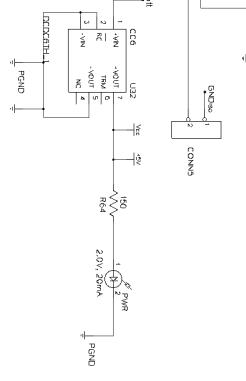
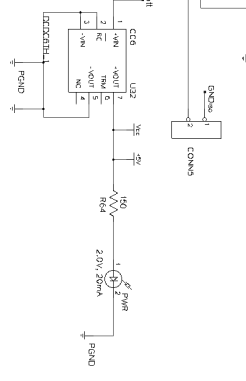
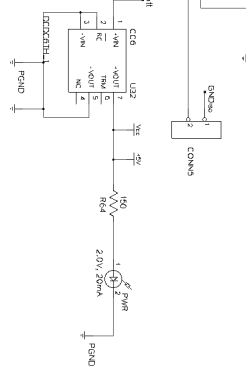
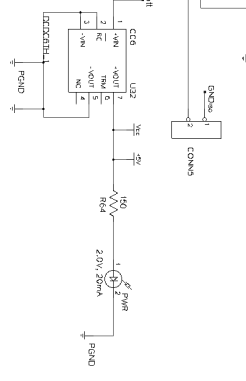
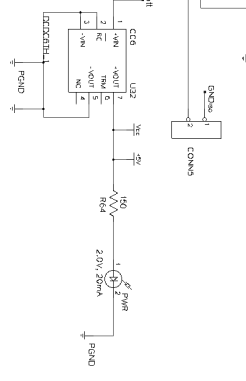
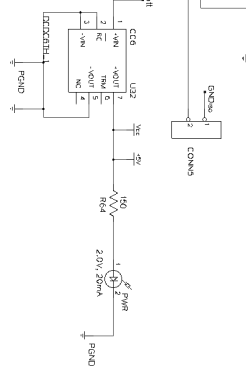
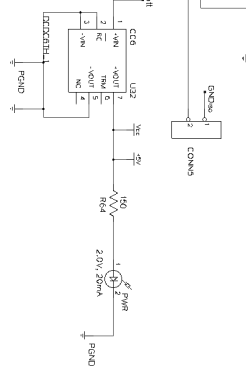
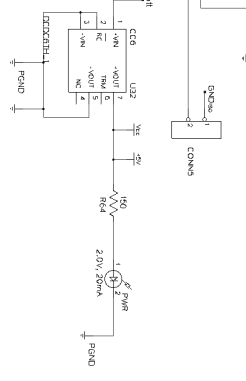
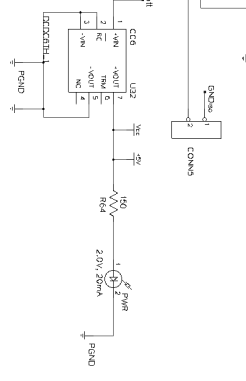
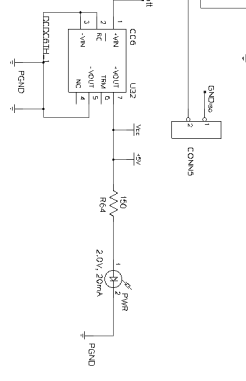
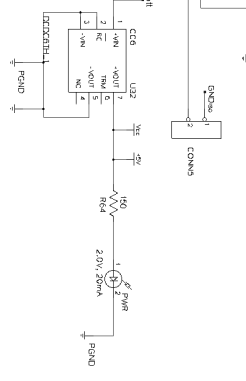
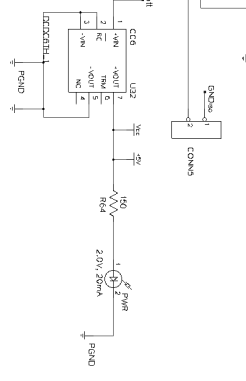
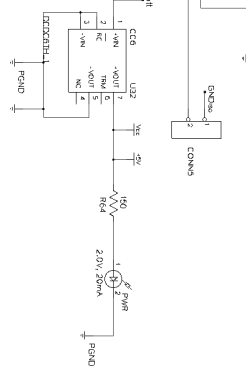
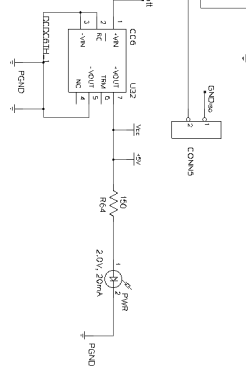
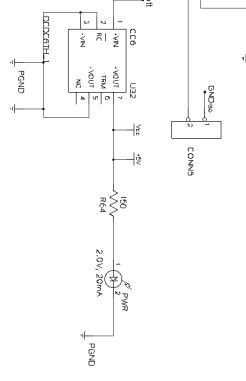
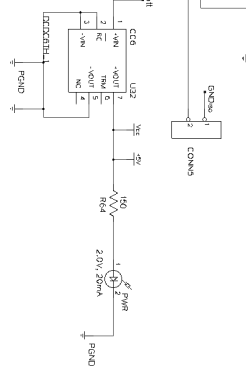
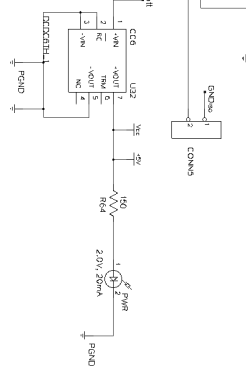
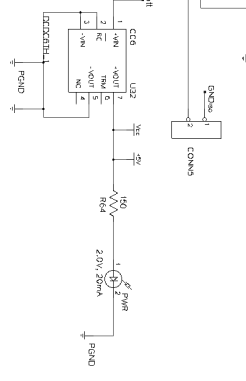
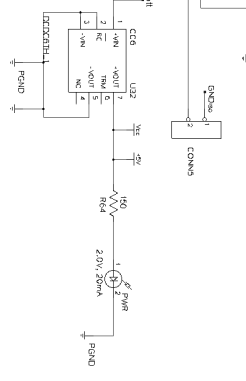
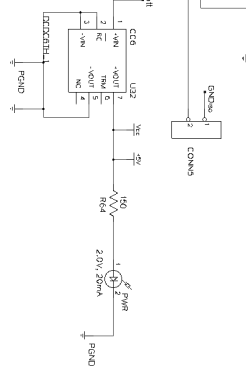
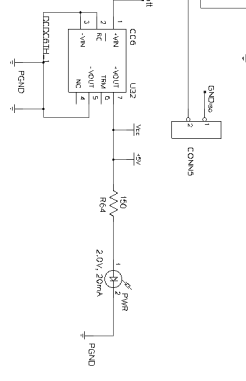
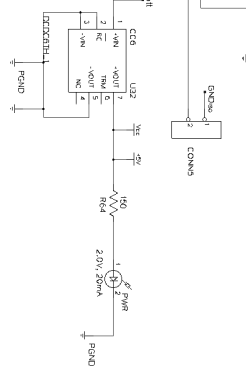
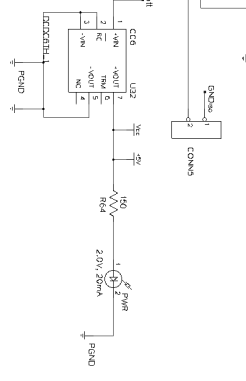
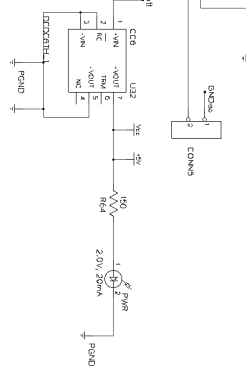
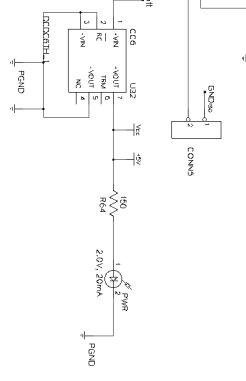
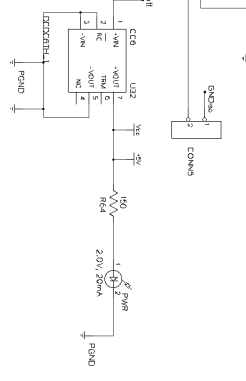
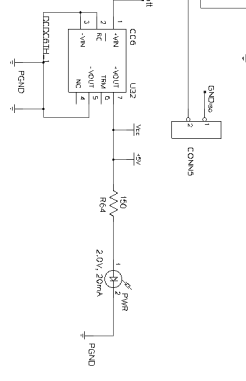
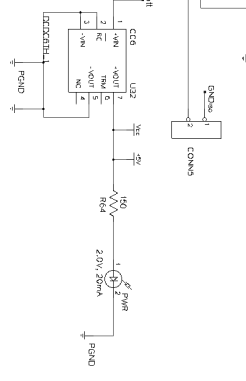
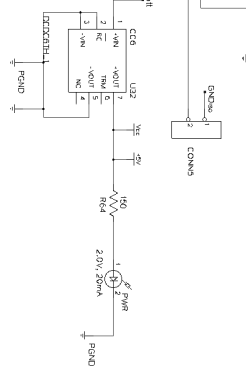
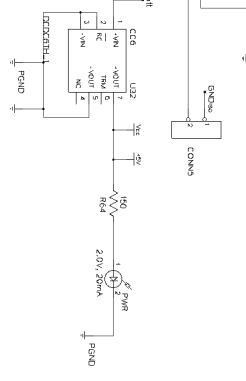
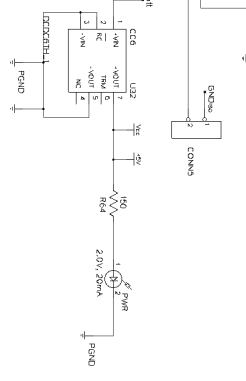
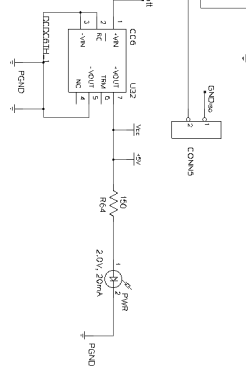
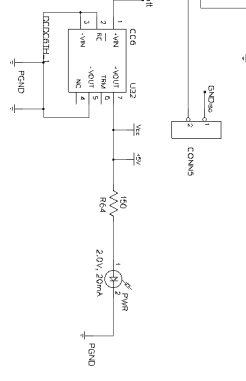
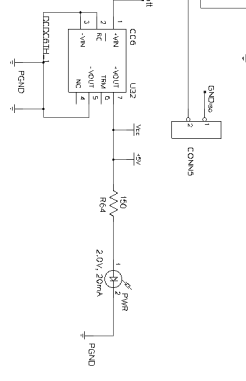
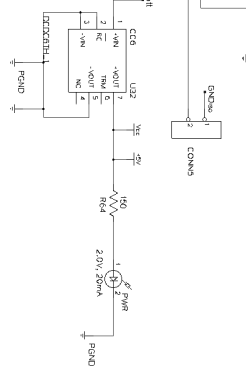
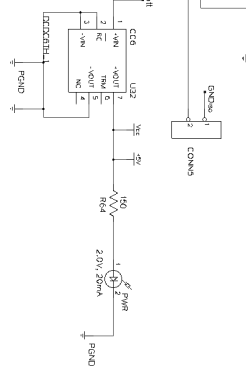
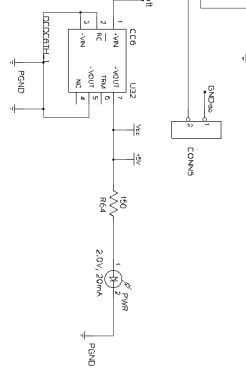
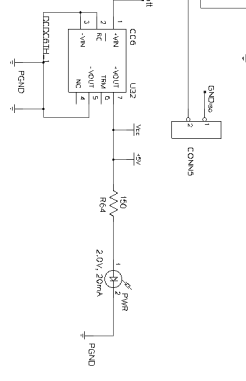
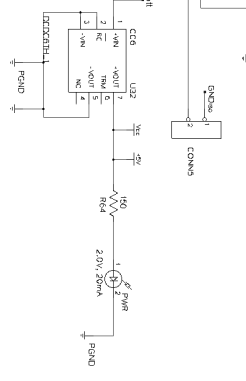
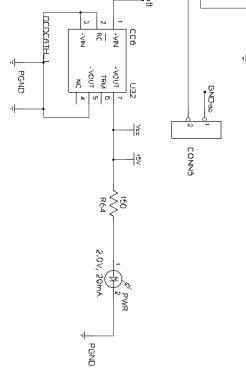
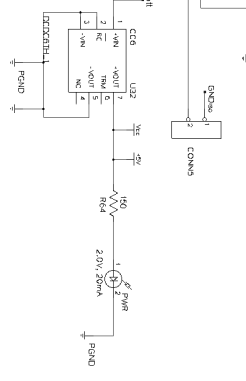
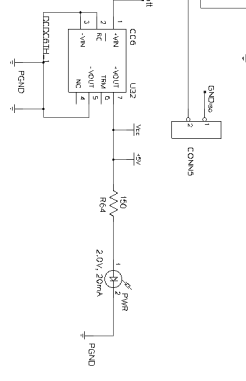
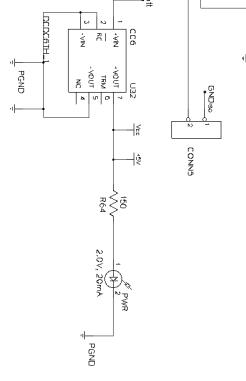
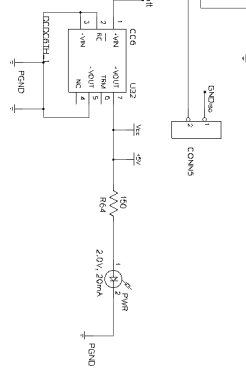
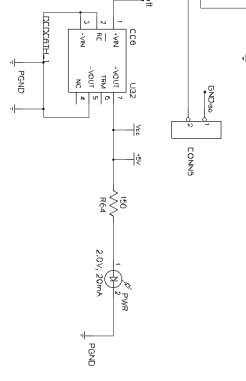
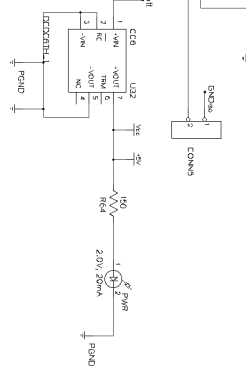
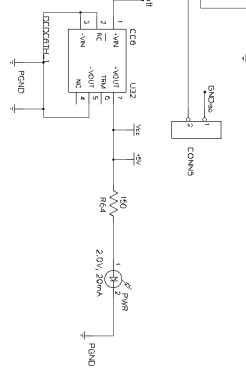
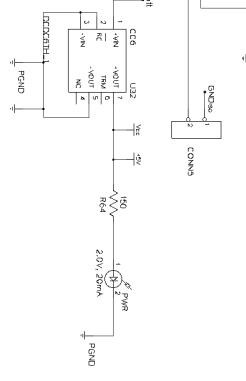
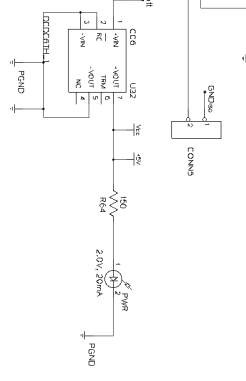
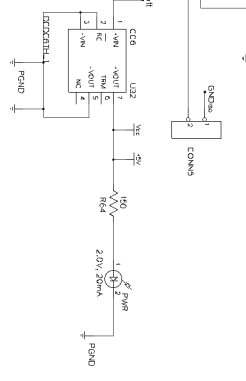
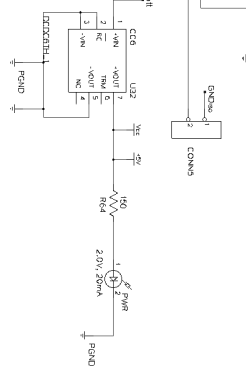
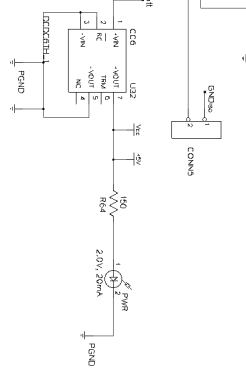
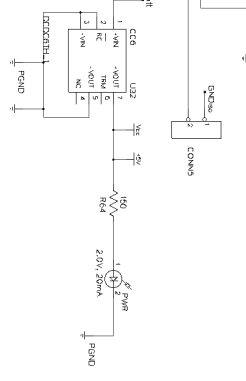
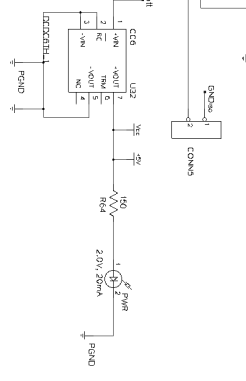
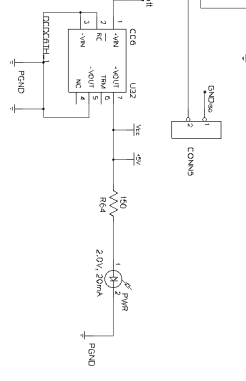
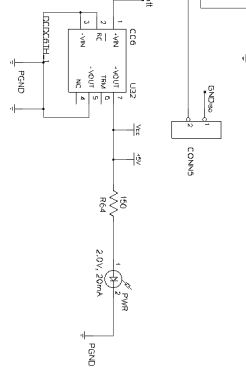
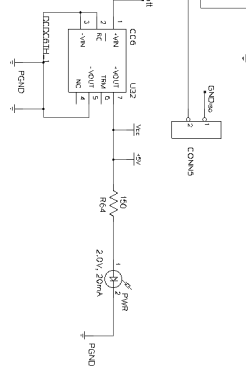
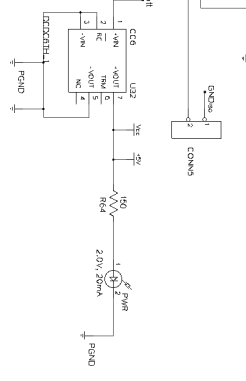
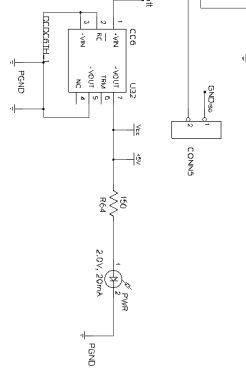
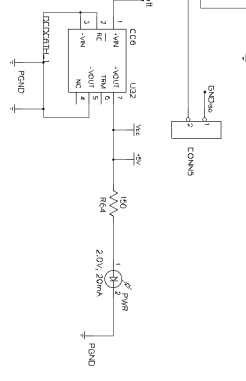
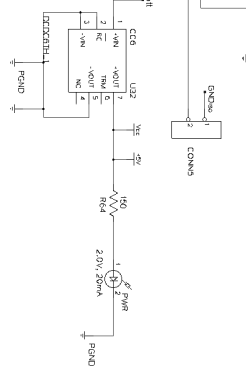
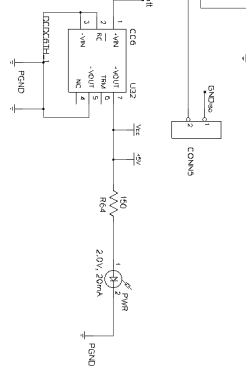
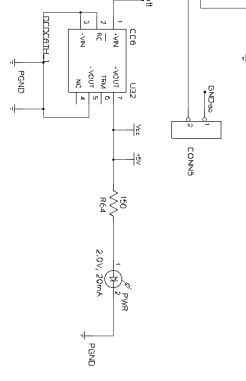
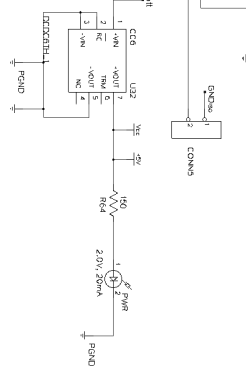
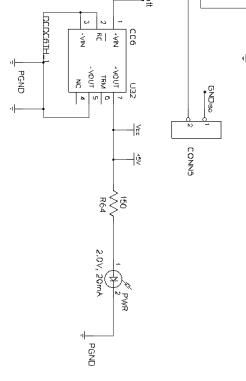
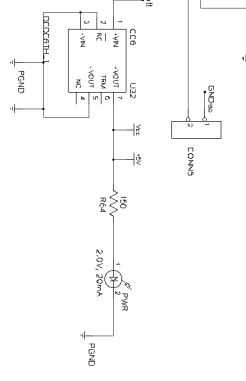
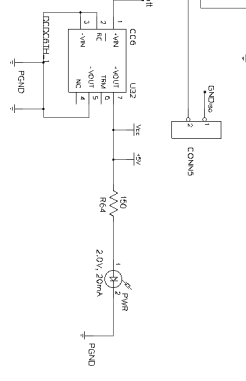
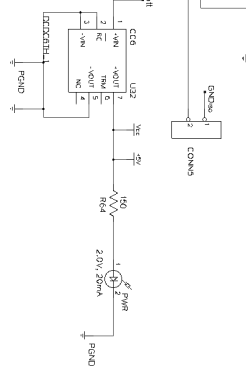
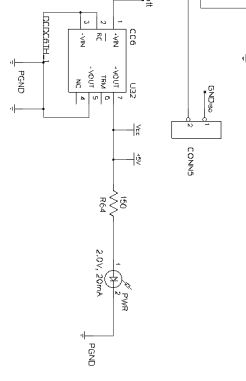
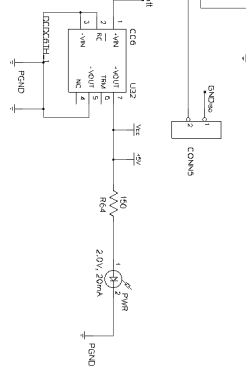
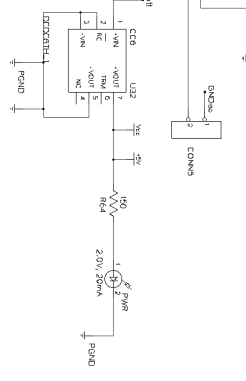
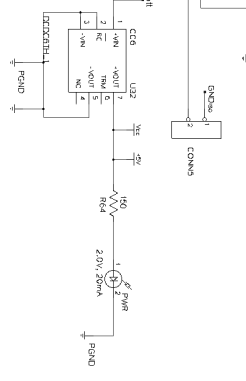
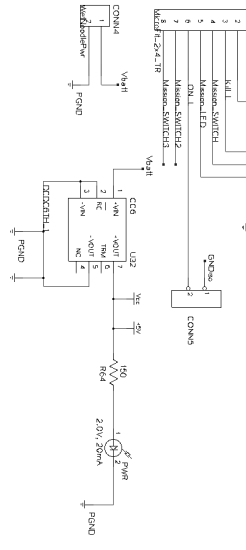
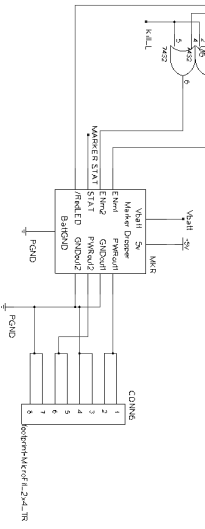
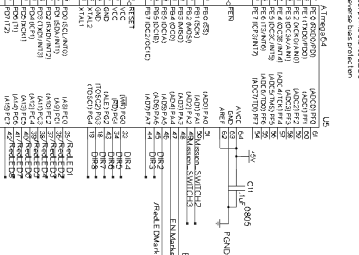
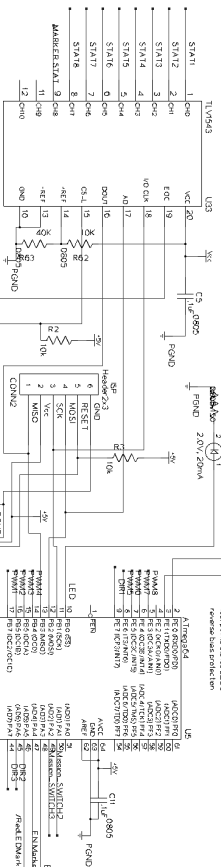
The thrusters are at a very full featured state. I currently have no planned improvements aside from finishing all the previously mentioned functionality.

Apendix:

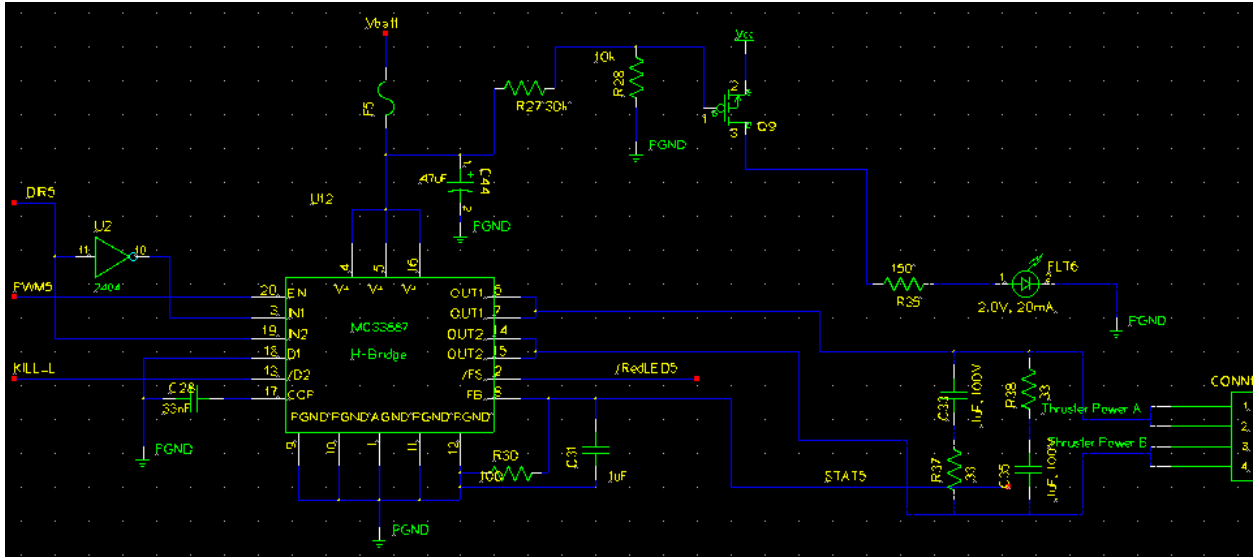
Microcontroller (The main processing section of the board)



TO DO:
Add the cost to the budget v.
Set board size LEDs
Remove the switches



H-Bridges (7 of these connect to the microcontroller)



PCB layout

