Hexapod

Ivan Bukreyev

Patrick Gauvin

April 22, 2012

University of Florida

Department of Electrical and Computer Engineering

EEL 4924C – Senior Design – Final Report

Instructors: Eric M. Schwartz

Table of Contents

Abstract

When I started Senior Design class I had an idea for my project – I wanted completely redesign and rebuild my earlier work - a hexapod, a 6 legged crawling robot. In my previous project I had designed and built mechanical platform, sensor array, and software package for such a robot, however I used pre-made electrical system and controller board. With experience gained from previous project, I had numerous changes I wanted to make to my new hexapod, and I thought it would be a perfect project for the Senior Design class.

For the purposes of the class I started from scratch: I built and designed completely new mechanical platform, built my own electronic boards, designed new and improved sensor array, used more complicated control algorithms and more. Amongst the biggest changes from my previous work is the use of embedded computing – my robot has a powerful ARM-based Linux board capable for tremendous computing power. In short, my new hexapod bears little resemblance (both mechanically and electrically) to my previous design and is a substantial improvement in every single way.

Introduction

The objective of my project was to create a 6 legged hexapod robot with 6 Degrees of Freedom per leg (6 DoF), a 6 DoF head, and 4 DoF tail remotely controlled by PS3 controller.

At the heart of the robot is the micro controller (uP) board. The uP board features ATxmega128A1U as the servo driver and data gathering center. In my design I use 23

independently controlled servo motors each having its own PWM channel. Since every servo is controlled independently of the CPU, very precise and smooth motion can be achieved. Aside from servo control, my uP board also collects data from numerous sensors: battery voltages, body accelerometer data, head accelerometer data, head rangefinders, and power status. The acquired data is shared with Linux board with direct USB connection.

A distinctive feature of my design is the motion controlled algorithm. I implemented a process called Inverse Kinematics (IK) to actuate the servos. In a nut shell, each of the hexapod's legs has the ability to move directly between any two points in space given the initial and final coordinates. This allows for smooth control of the locomotion. The arithmetic behind IK is rather complex, requiring numerous floating point calculations. I decided to use newly released Linux development board (ODROID-X2) to do all the heavy lifting. The bidirectional communication between ODROID and uP board is the basis of motion control: ODROID requests current sensor status, processes it, and sends back the positions for the servos. The ODROID itself is controlled by the used via a PS3 controller.

In my design power management was of a major concern. I had to have 3 separate voltage rails: 3.3 V for the microcontroller, 5V for the ODROID, and 6V for the servo motors. To accommodate lifting ability for the servos, 6V rail is capable of handling 20A of continuous current. To ensure that my main battery supply is not drained below the rated capacity, my power management board is equipped with automatic relay shut off mechanism: when the voltage of any cell reaches below the specified threshold, the power to the robot gets cut off. To warn the user of the impending doom, a TTL warning signal is sent out shortly prior to main system shutdown.

Electrical Design – Microcontroller board

At the heart of my robot is the microcontroller (uP) board. The main feature of the board is ATxmega128A1U microcontroller. It serves three main functions: servo driver, data gathering center, and communication hub. There 23 servos in my design and each is controlled by a dedicated PWM channel on the ATxmega. This configuration allows for maximum precision with minimum CPU intervention. The uP board has 3 switching voltage regulators: 3.3V for the microcontroller, 5V for the Linux board, and 6V for the servo motors. The 6V switching regulator is a high power module capable of delivering 20A continuous current to the servos. This regulator is used exclusively for servos, which allowed me to maximize power output. Parts of schematic are shown in Figures 1 and 2.

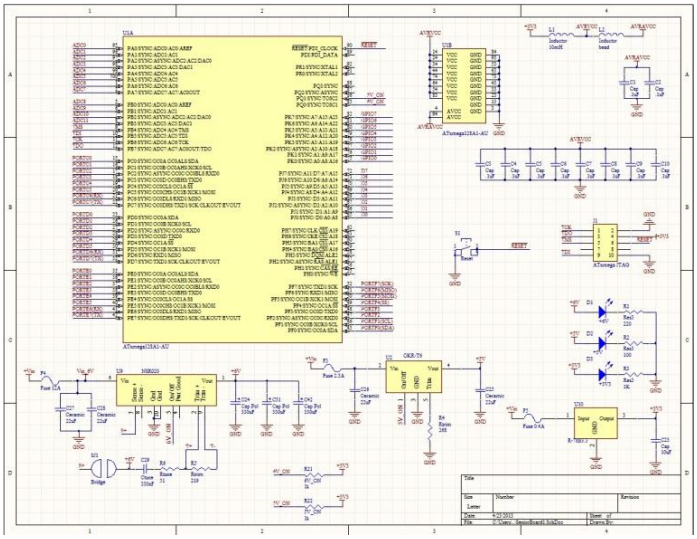

Figure 1. ATxmega and Power regulators.

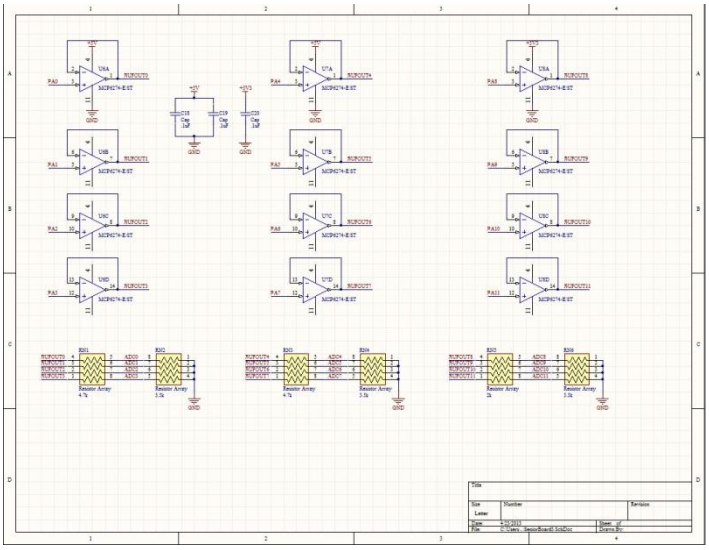

Figure 2. ADC Voltage buffers

The ADC on ATxmega is capable of reading voltages up to 2.0V, so in order to read higher sensor outputs, I had to design op-amp buffer stages for each of the channels. These buffers are feed by 3.3 V and 5V regulators. During regular operation, ATxmega continuously gathers most recent sensor data and stores it in memory. The Linux board frequently requests various sensor data from ATxmega in order to do higher level functions. In addition, Linux board also provides uP board with current servo positions, so that motion can be achieved. Final layout of the board is shown in Figure 3 below.

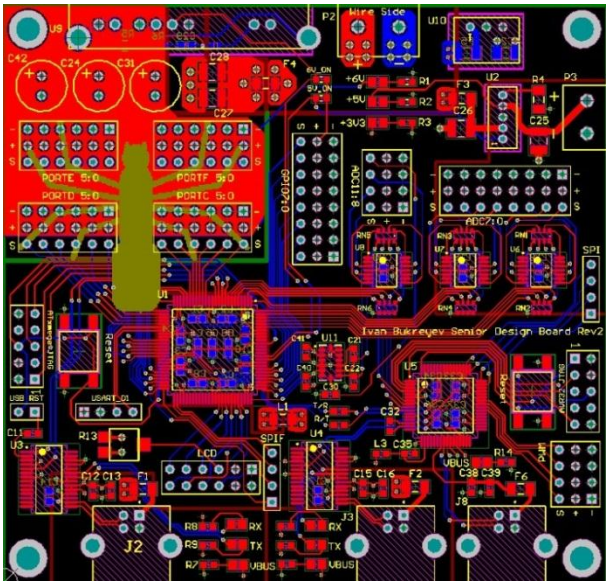

Figure 3. uP board layout, overall dimension 3.5X3.7

It took considerable time in designing the layout. I had taken care and separated high power circuitry from the microprocessor. In the end, my 4 layer design allowed simultaneously exceedingly low ripple fluctuation on the microcontroller voltage supply (~3mV peak to peak) and high current operation on the servo side.

Electrical Design – Power Router board

In my robot, I used 3 cell Lithium Polymer (LiPo) battery with 4000 mAh capacity. While very energy density efficient, LiPo batteries can pose serious threat if not handled with care. In particular, overdrawing such a battery can cause failure on the next charge cycle. In order to protect my battery (and the robot) I designed a special board that continuously monitors the status of the LiPo. If any of the cells fall below specified threshold, the circuitry will

automatically turn off relay that feeds all other boards and systems on my robot. The schematic for this board is shown in Figure 4.

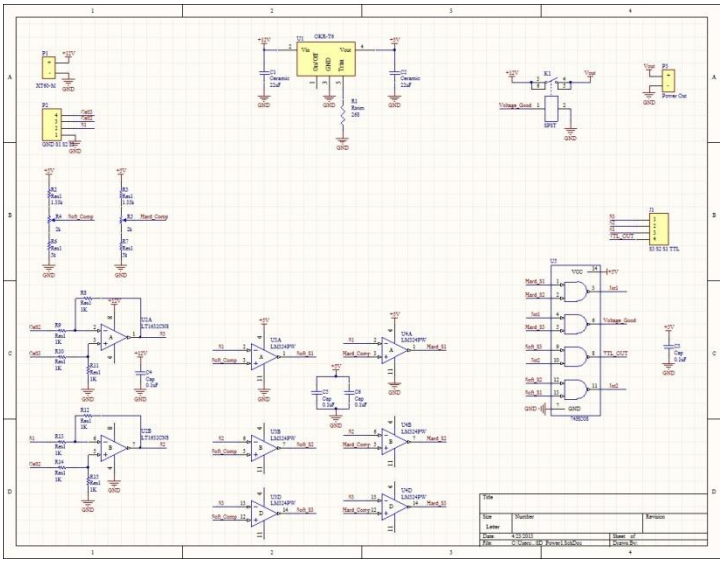

Figure 4. Power router schematic.

Additional feature of this board is the ability to drive RGB LEDs. The board houses power transistors and current limiters to drive up to 60W of power to the LEDs. The layout of this board is shown in Figure 5.

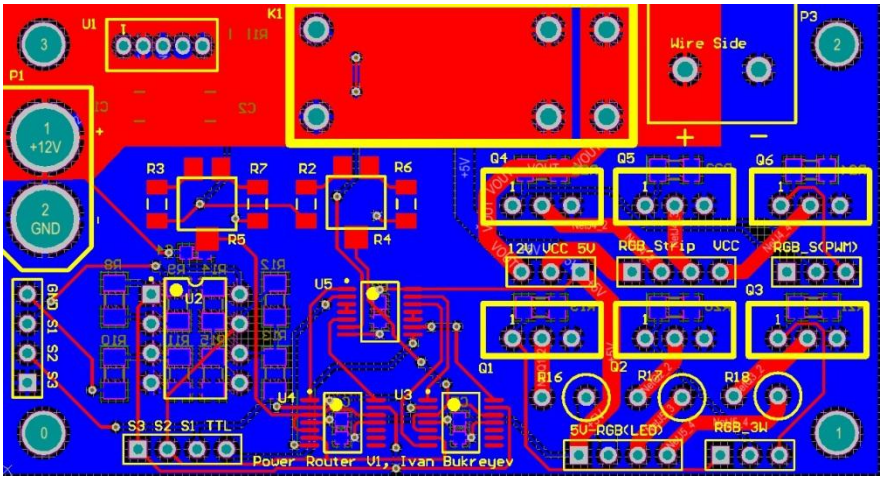

Figure 5. Power router layout.

Electrical Design – Audio Board

In order to meet analogue requirements of this course, I had to design a transistor level audio amplifier. The circuit is shown in Figure 6.

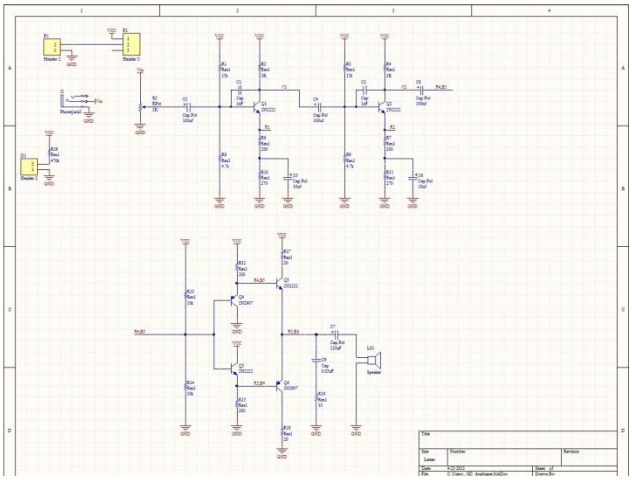

Figure 6. Audio Amplifier circuit.

My audio amplifier consists of three stages. Stage 1 and 2 are BJT common emitter small signal amplifiers. These two stages operate at relatively low current and bring the audio signal up to 6V peak to peak. The final stage is a double emitter follower that delivers output signal to the 8 ohm speaker with nominal voltage gain of about 1. At the time of the design, I did not have power BTJs available so I had to use logic level BJTs to drive the speaker. This resulted in reduced output power and lesser audio quality. The layout for the amplifier is shown in Figure 7.

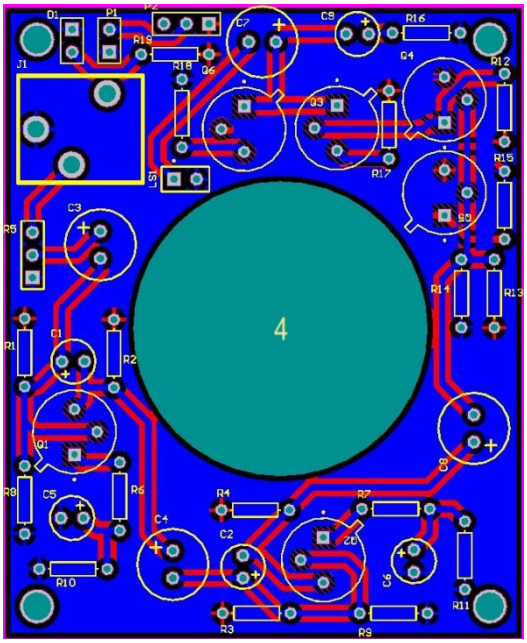

Figure 7. Audio Amplifier layout.

In the end I was able to get my audio amplifier to work reasonably well, although I was limited by my power output.

Linux Board

I used ODROID-X2 development board as my main computation platform. Originally I had planned to use AVR32 to perform Inverse Kinematics calculations, however due to the time constraint of the course, I had to move all heavy processing to my Linux board. Currently, the ODROID is running all robot control algorithms. The robot is controlled by as PS3 controller that links to the ODRIOD upon startup. From this point on, all functionality and behavior of the robot is controlled by the PS3 controller.

Communication Protocol

Microcontroller board and Linux board operate according to a sophisticated communication protocol. First, there is a heartbeat in place that will reset the program in case there is a long term communication loss. Second, the communication packet itself has several layers of protection against corrupted data. The following is the structure of my communication packet: Start, Start, Start, Message Code, Message (lengths from 2-48 bytes), CRC checksum, End. The CRC checksum that is performed on the entire data packet virtually eliminates the possibility of incorrectly decoding a message. This system is in place to make sure that my robot will never set servo angle (or any other peripheral) to an illegal value on accident.

Mechanical Platform

Although the most complicated aspect of my project, the mechanical platform has little to do with Electrical Engineering, and therefore will be mentioned briskly. Figure 8 shows right leg of my Hexapod (6 legs total).

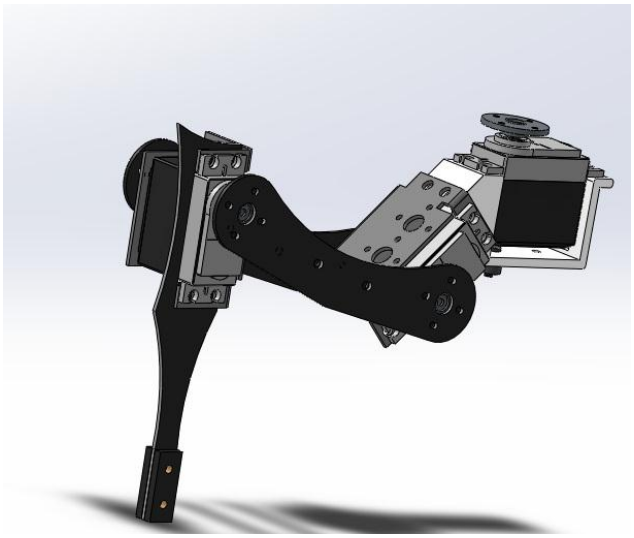

Figure 8. Leg design.

Aside from the servo and servo mounting bracket, every part of the leg was custom made for this project (per my design specification). Same goes for the entire hexapod, which is shown in Figure 9.

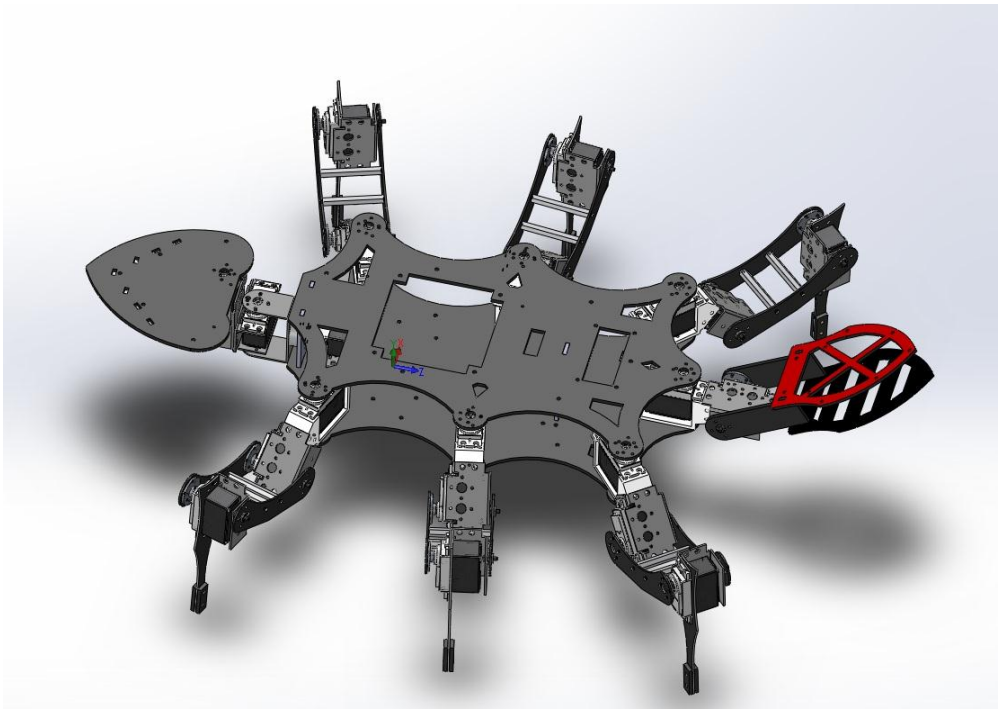

Figure 9. Solidworks design of the hexapod.

My mechanical platform features six 6 DoF legs, 6DoF head, and 4DoF tail.

Controller Algorithm

The locomotion is done by using Inverse kinematics algorithm. Traditional hexapods operate by cycling through imperially determined states which results in jerky motion. In my design, every leg has a defined coordinate system. Within this system, the foot of every leg is able to travel between any two points along a fitted curve. Basically, I can move robot's legs between two specified coordinates in a linear fashion using finite number of time steps. This gives motion silkiness and real life feel.

Conclusion

The project I picked for the Senior Designed turned out to be inadequately complicated for the time frame given. In the end I was able to get most of the design to work, although it came at the price of making sacrifices, mostly to the mechanical platform. By far the most difficult aspect of the project was the mechanical platform – 300+ screws, over a pound of wire, 23 moving parts that cannot collide and others were the reason for my poor sleep habits in the spring of 2013.

At the time this report was written, much remains to be done for my own personal satisfaction.