A Serpentine Robot Designed for Efficient Rectilinear Motion

A Thesis

Submitted to the Faculty

of

Drexel University

by

Richard Anthony Primerano

in partial fulfillment of the

requirements for the degree

of

Master of Science

September 2008

Table of Contents

List of Figures

List of Acronyms

ABS	acrylonitrile butadiene styrene
ADC	analog-to-digital converter
CAN	controller area network
CMRR	common-mode rejection ratio
DOF	degrees of freedom
IR	infrared
MEMS	microelectromechanical systems
MIPS	million instructions per second
PID	proportional-integral-derivative
PLL	phase-locked loop
PWM	pulse width modulation

Abstract

A Serpentine Robot Designed for Efficient Rectilinear Motion
Richard Anthony Primerano
Moshe Kam and William Regli

Robots that mimic the natural motions of animals have long been of interest in science and engineering. The primary engineering interest in such robots is in having them conduct tasks that require complicated locomotion and cognition. The biological creatures after which the human-made robots are designed manifest a remarkable degree of efficiency and agility when compared to what we have been able to mimic so far in human-made designs. For example, the small cross-section and low center of gravity of most biological snakes, coupled with their large repertoire of possible motion sequences, make their bodies very efficient when navigating confined spaces and rough terrains. To date, no "artificial" snake has been able to come close to duplicating these navigational characteristics.

In this study we concentrate on a set of motions observed in medium size (1-4m) biological snakes. There are currently several robot designs that attempt to reproduce the movements of such snakes. Almost all of these designs require the robot to articulate segments of its body in a repetitive sequence to achieve locomotion, and some even attach passive wheels to the snake's body in order to facilitate movement. As a result of these design decisions, the artificial snakes are generally slow and most (especially those with wheels) are not well suited for travel over rough terrain. We offer an alternative design that propels the snake using many small feet attached to disk-like body units ("ribs"). Due to the superior flexibility that this design provides, the resulting robot, which we have built and tested, can actually "walk" over obstacles and therefore will be much more maneuverable than existing prototypes.

1 Introduction

Snake-like robots [1] are believed to offer several advantages over conventional wheeled or legged robots. For example, robotic snakes have a low center of gravity, which makes them very stable when moving on inclines. In addition, if a snake-like robot were to fall over, it may recover by articulating its body in the proper way. Unlike their walking or wheeled counterparts, snake-like robots spread their weight out over a large area, thus exerting less force per unit area over the surface on which they are moving. This characteristic means that robots of this class are better suited for moving over soil or sand, compared to wheeled and legged robots that are more likely to get stuck in such environments.

This study details the design and construction of a robotic snake prototype that addresses many of the shortcomings found in previous robots of its type. The key element of this design is the robot's ability to execute rectilinear motion (straight line) by simply controlling a single drive motor as opposed to articulating its body in complex motions (gaits) as is typical of many other serpentine robots. During the design process, many options had to be considered, and throughout this document these alternatives will be presented and addressed. Plans for a second version of this robot will also be presented, based on lessons learned through the construction of this first prototype.

2 Background

Most biological snakes employ one of four major types of locomotion, each of which is outlined below [1].

Lateral Undulation – The snake's body forms a series of S shaped curves. The back portion of each of these curves pushes against the ground propelling the snake forward.

Concertina – the snake expands and contracts sections of its body alternately, while planting others on the ground firmly. This motion is similar to that of an inchworm.

Sidewinding – the snake contacts the ground at only two points while moving its body in a 'sinusoidal' motion. The result is that the snake moves sideways rather than forward.

Rectilinear – the snake propels itself in a straight line by moving scales on its stomach in a wave-like motion. Rectilinear motion allows snakes to access very confined spaces.

It would appear that if a robotic snake could undergo rectilinear motion, it would be very maneuverable in constricted areas. Currently, robotic snakes are available which can undergo rectilinear motion by articulating each of their segments in a repeated sequence. A 1995 paper by Chirikjian [3] outlines various methods for accomplishing this task. The approach is to create rectilinear locomotion through body movements alone. Chirikjian classifies snake-like robots as either *inextensible* or *extensible*. The

former are capable of only bending their segments with respect to each other while the latter can expand and contract like an accordion. He further outlines several locomotion algorithms for robots of these types, but does not address their construction and testing. We observe that robots of this family are expected to be slow and require extensive operator input. In addition, the precise interaction between segments, which is required for efficient locomotion, can be difficult to achieve. Downing [1] goes into great detail discussing the possible construction of inextensible snake-like robots, but does not settle on a particular design.

Our approach, developed through the study of these papers, as well as several prototypes that were built earlier in Philadelphia-area laboratories, proposes an alternative design for robotic snakes. This design does not fall into any of the previous classes. Our robot propels itself using many small "feet", with locomotion similar to that of a millipede. Additionally, the power to drive all of the snake's feet is derived from a single motor located in the tail of the robot. Moving the snake forward is a simple matter of controlling this drive motor. Comparing this approach to locomotion with that used in most other robotic snake designs, we see that most other robots of this class accomplish forward motion through a complex sequence of body movements (called a gait). While this articulation is effective in moving the robot forward, even over rough terrains, the forward motion of these types of robots is generally very slow. The main objective of our design was to construct a robot that could efficiently perform straight line motion with minimal power consumption and computation. Since this robot will actually walk, rather than dragging itself, we

expect it to be capable of navigating rough terrains more efficiently than existing prototypes.

3 Mechanical Design

3.1 Overview

Before development of this device began, several main design objectives were established. These included: (1) ability to perform efficient rectilinear motion, (2) small cross section, and (3) ability to easily lengthen or shorten the robot. The first of these needs arose from the observation that all current robot designs that are capable of operating in rough terrains perform forward motion through gaits that tend to be very slow (measured in inches per minute). We sought to develop a device that could perform in these same environments, but move at higher forward speeds (measured in inches per second). The second objective arose from the desire to develop a robot that could access confined spaces, in applications such as pipe inspection and exploration. The final objective is desirable because a robot that can easily be lengthened to suit a particular application will be more versatile and adaptable. Additionally, a design of this type suggests that if a segment were to malfunction or break, it could be removed and the robot be placed back into service. An added benefit of this type of modular design is that robot is essentially composed of many copies of the same mechanism, simplifying the development process. To see how these issues have been addressed, we first take a system level look at the robot's mechanical structures and their interactions. Subsequent sections provide a detailed look at individual component operation and design.

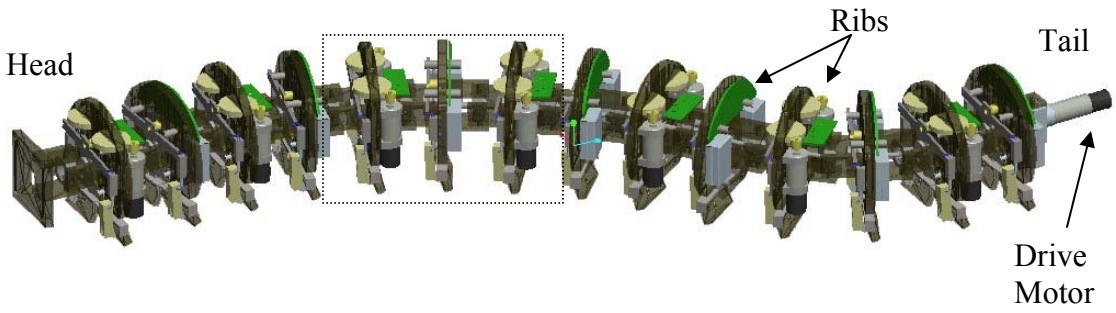


Figure 1: Computer Rendering of Robotic Snake

Figure 1 shows a rendering of the robotic snake. The robot contains fourteen ribs and is approximately 30" long. At the bottom of each rib is a pair of feet that carry the robot forward. Forward motion is powered by a drive motor located at the tail of the snake.

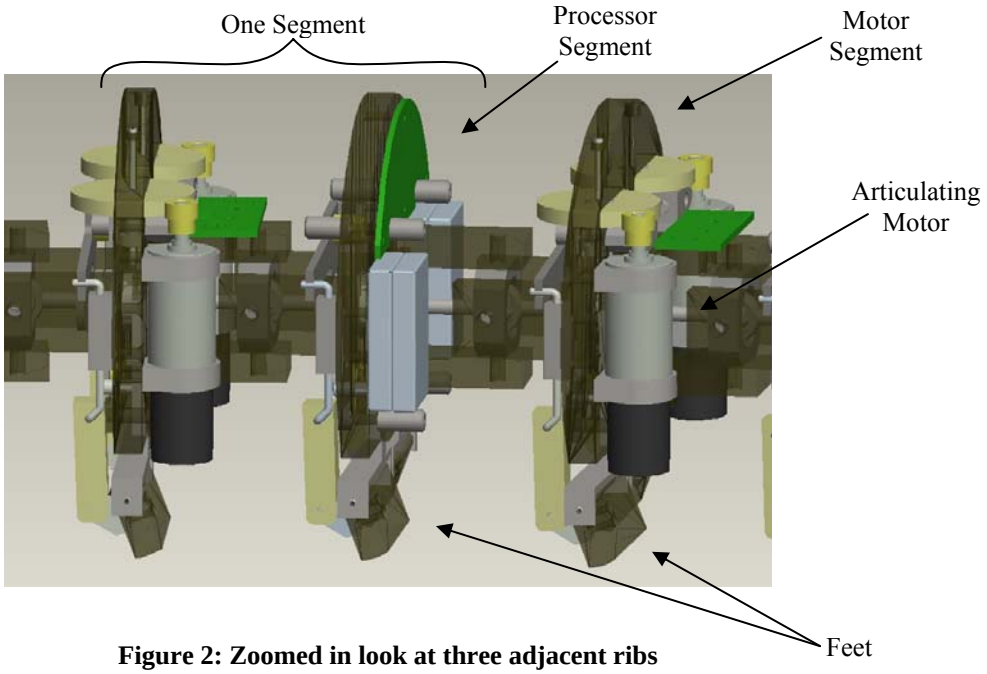


Figure 2: Zoomed in look at three adjacent ribs

Figure 2 shows a zoomed in view of three adjacent ribs. From this picture, it is clear that the robot is composed of two types of ribs. The first contains two servo motors used for articulation while the second contains a circuit board and batteries. These are

named *motor ribs* and *processor ribs* respectively. Starting from the head of the snake, ribs alternate between motor and processor type. There are a total of seven motor ribs and seven processor ribs. The design is such that (referring to *figure 2*) the motors located on a particular motor segment are driven by the circuitry located on the processor board immediately behind it. For this reason, addition and removal of ribs must be done as a pair. The motor/processor rib pair is referred to as a *segment*.

Thus far, we have introduced two types of motors found on this robot. The first is the drive motor (*figure 1*) located at the rear of the snake. This motor transmits power to all upstream ribs to power a series of "feet" on the bottom of the snake. *Figure 3* shows the components responsible for this and how they are positioned on the robot's rib. The basic function of this mechanism is to take rotary motion applied to the cam and convert it to orbital motion in the feet. The detailed operation of this device is discussed in section *4.3 U-Joint/Cam Assembly* and section *4.4 Foot Mechanism*.

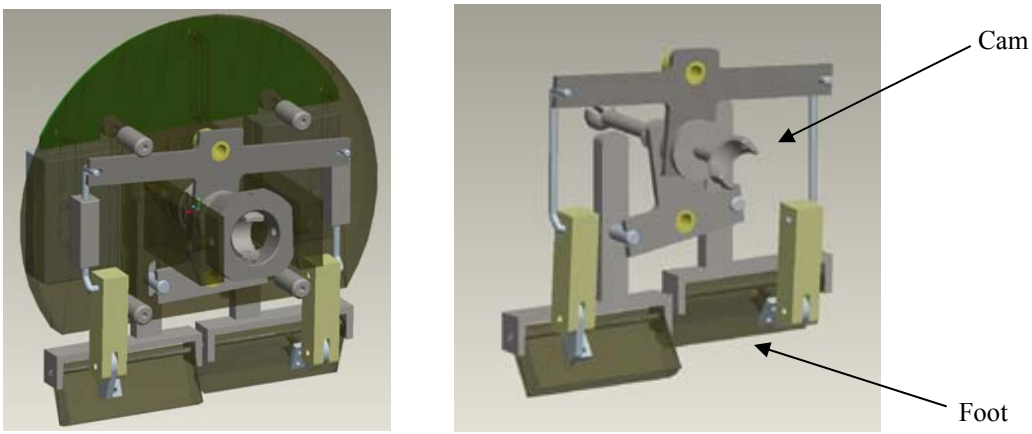


Figure 3: Components on a typical rib

The second type of motor is the articulation motor (figure 2). Each motor segment contains two such motors. These are responsible for bending (articulating) the segment on which they are located. These two motors provide two degrees of motion

per segment. The prototype snake fourteen articulation motors. Their operation is detailed in section *4.5 Servo Mechanism*.

The cross section of this robot is 3.5". The necessity to integrate a large number of moving parts in such a small area was aided by computer design and modeling. ProEngineer was used exclusively for developing solid models of each of these parts, assembling these parts and performing interference analysis on the finished assembly to ensure that parts would not interfere with each other under normal operation.

The snake has been designed such that its length can be increased or reduced to fit the particular application. Each segment is identical to all others. It carries its own batteries and processor.

3.2 Frame

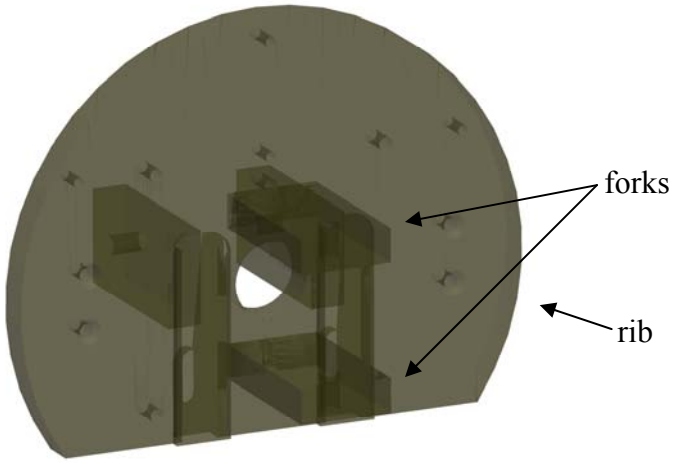


Figure 4: Snake Processor Rib

The frame of the snake is constructed from 0.231" thick acrylic ribs as illustrated in *figure 4*. Four forks, two on each side, have been solvent welded to the rib. The nominal pitch between the ribs is 2". With fourteen ribs in the prototype, the robot is

approximately thirty inches long. Acrylic was chosen because it is economical and easy to shape by machine.

3.2.1 Design Options/Future Considerations

Acrylic was ideal for building a proof-of-concept device because of it good machinability characteristics and because the ease with which composite parts (figure 4) can be assembled. The four forks shown to the rib in figure 4 are attached with a solvent based acrylic adhesive that melts the joining surfaces and forms a bond that is as strong as the base material. The ability to create such strong bonds was another factor in choosing this material for the prototype robot.

The second version of this robot (still in the deign stage) will have ribs constructed of injection molded Acrylonitrile butadiene styrene (ABS) plastic. This material has much better impact resistance than acrylic, allowing us to use thinner cross-sections for the rib construction. Not only will the ribs be stronger, but they will considerably lighter. The newly designed rib is approximately 35% lighter than the original rib. This change alone will translate into a ~20% reduction in weight of the overall robot. The table below compares acrylic with injection molded ABS. For our purposes, the most important of these values are the impact test results. For example, the *unnotched* Izod impact test shows that ABS can absorb at least four times more impact energy than acrylic before failure. Crack propagation in acrylic tends to be very rapid. Small cracks that may form during the life of the robot can propagate quickly and lead to failure of the part. This behavior is associated with *brittle failure*. On the other hand, ABS exhibits *ductile failure*, which means that cracks and defects to not spread quickly, but rather the part bends before if ultimately fails. These classifications are

reflected in the *notched* Izod impact test. Here, the impact strength of the material is reduced substantially due to the addition of a notch. ABS is available in many grades with varying mechanical properties. For the second version of this robot, we will choose as ABS that exhibits good impact strength and elasticity.

Property	Acrylic	ABS
Density	0.0415-0.043 lb/in^3	0.0368-0.0437 lb/in^3
Elongation @ break	1-30 %	2-110 %
Elongation @ Yield	4-5 %	1.7-6 %
Izod Impact, Notched	0.225- 0.375 ft-lb/in	7-12 ft-lb/in
Izod Impact, Unnotched	5.06 ft-lb/in	20 ft-lb/in to NB
Charpy Impact, Unnotched	9.04-28.6 ft-lb/in^2	23.8-114 ft-lb/in^2

Material properties of Acrylic and ABS plastics

3.3 Cam/U-Joint Device

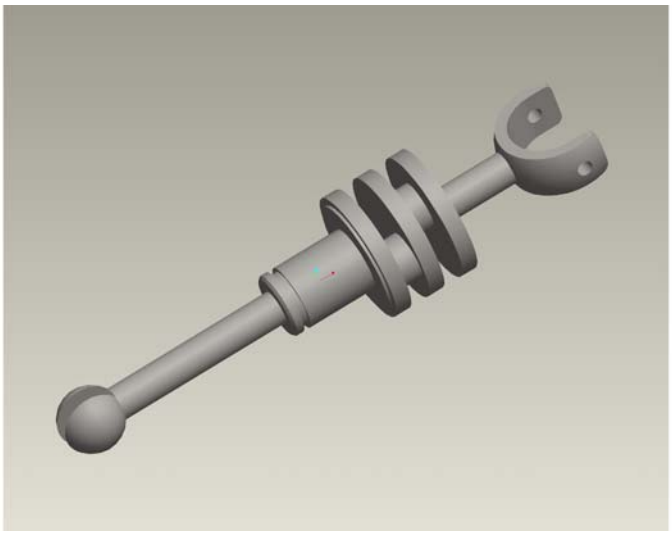


Figure 5: Cam/U-Joint Device

A 0.500” hole is located in the center of each rib (see *figure 4*) and two flanged ball bearings are pressed into it. These bearings support the cam/universal joint device shown in *figure 3.* This component has two functions; the device (1) transmits power from the drive motor, located in the rear of the snake, to all upstream segments while

allowing flexure between segments and (2) converts the drive motor's rotational motion into an orbital motion in the snake's feet by way of a set of cams.

When designing this device, the non-ideal behavior of the u-joint must be taken into account. Universal joints have the undesirable property of being *non constant velocity*. This means that for a constant input shaft velocity, and non-zero joint angle, the output shaft velocity varies approximately sinusoidally with time. This sinusoidal speed variation worsens as joint angle increases. If left uncorrected, the velocity ripple would quickly multiply as rotary motion traveled down the fourteen u-joints found in the snake. The result is that feet at the rear of the snake (near the motor) would operate smoothly, while those at the front of the snake would move with very abrupt motions. This condition would render the snake inoperable. Fortunately, there is a simple solution to this problem know as a constant velocity joint (CV Joint).

3.3.1 Kinematics of the U-Joint and CV Joint

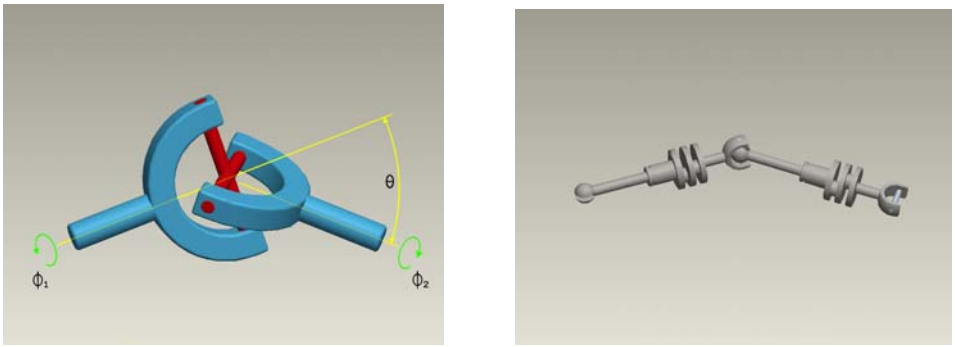


Figure 6: Illustrated U-Joint and Assembled Snake U-Joint

Figure 4 shows a conventional universal joint compared with one of the u-joints found on our snake. The latter is formed when two of the snake's cams are joined at their ends. The input/output relationship for a u-joint is defined as

$$\phi_2 = \tan^{-1} \frac{\tan \phi_1}{\cos \theta} \qquad (1)$$

Note that when $\theta = 0$, the equation reduces to $\phi_2 = \phi_1$. Physically, this means that when there is no bend in the joint, the input and output shafts move with same velocity. As the joint angle varies, however, the output shaft undergoes periodic velocity fluctuations even when input velocity is constant. Computer simulation shows the degree of speed variation that can be found in the segments of the robotic snake for even moderate joint angles. The graphic shown in *figure 7* was used as the input to ProEngineer's motion analysis software. This figure represents a snake seven segments long with each segment bending at an angle of 25 degrees.

Figure 7: U-Joint Test Setup

For our analysis, a constant angular velocity (180 rad/sec) was applied to the first cam in the assembly, and the angular velocity of each downstream cam was plotted over time. The results, plotted in *figure 8*, demonstrate the severity of velocity distortion seen in downstream cams. As expected, cam one rotates at the same speed as the servo motor, while each successive cam rotates with a more distorted motion than the

previous one. Cam seven's velocity varies by +80/-45% of the drive motor velocity each cycle. If this behavior were left uncorrected, the snake would not operate properly.

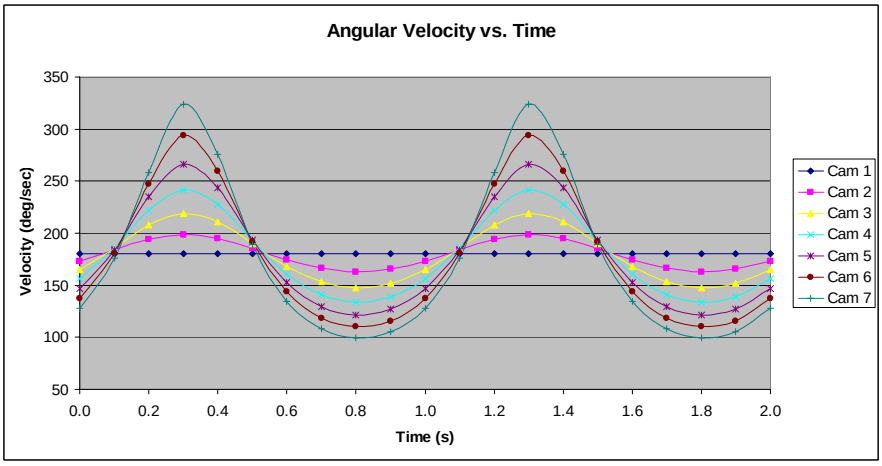


Figure 8: U-Joint analysis results

In the arrangement described above, the non-constant velocity behavior at each universal joint in the series is the accumulation of the non-CV behavior of all universal joints that precede it. With a minor revision to the design, however, this non-ideal behavior can be made to cancel between adjacent universal joints. In an arrangement known as a *constant velocity joint*, two universal joints are placed back to back (see figure 9). In this configuration, the input and output shafts rotate at the same velocity, while the linkage between the two shafts exhibits a non-constant velocity behavior. In order for the non-ideal behavior of the two u-joints to exactly cancel each other, the bending angle of each joint in figure 9 must be equal.

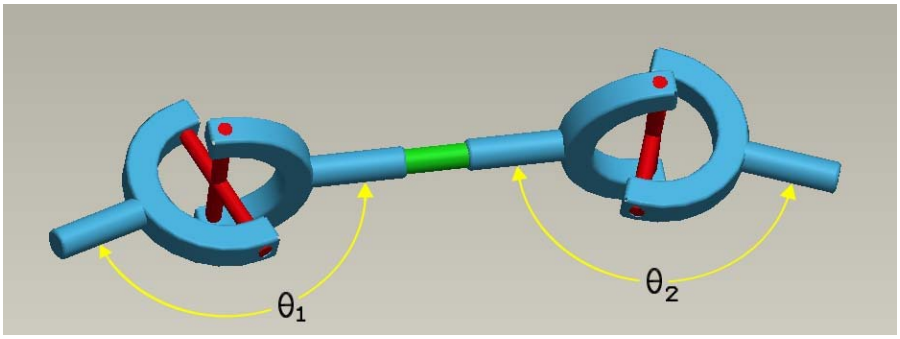


Figure 9: Constant Velocity Joint

The kinematic analysis is repeated using the modified cams. Again, there are seven cams each with a bending angle of 25 degrees. Applying a constant angular velocity at the first u-joint, we measure the angular velocity of each downstream joint. The results of this test are shown in *figure 10*, where angular velocity is plotted verses time. Note that now the non-ideal behavior is confined to even cams while the odd cams move with constant angular velocity. The velocity ripple that remains in the odd numbered cams cannot be removed, but at the joint angles produced by the snake, it will be confined to about ±10% drive motor velocity.

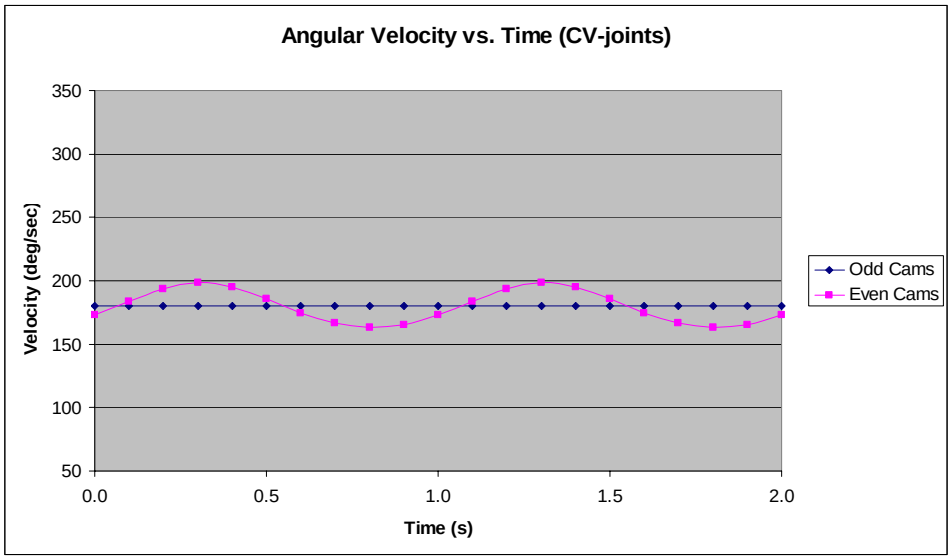


Figure 10: Constant Velocity Joint Analysis

3.4 Foot Mechanism

As the drive motor rotates, it imparts motion to the snake's universal joints (section 4.3). As the u-joint rotates, so to do the integral cams (see figure 5) which couple drive motor power to the *foot mechanism*. The foot mechanism is a group of mechanical linkages which convert rotational motion from the cam into an orbital "walking" motion in the snake's feet. Figure 11 shows a series of snap-shots of the foot mechanism over one full rotation of the cam. Viewing the figure left to right, top to bottom, observe that as the cam rotates clockwise, the left foot begins in the downward position (frame 1) and sweeps an orbital path until it returns to its original position.

Following the motion of the right foot from frame to frame, its motion is 180° out pf phase with respect to the motion of the left foot. When one foot is touching the ground, the other is elevated. Additionally, the robot designed such that the feet on neighboring ribs operate with 180° phase difference. In other words, when the left foot of one rib is touching the ground, the right foot of its neighbor is touching.

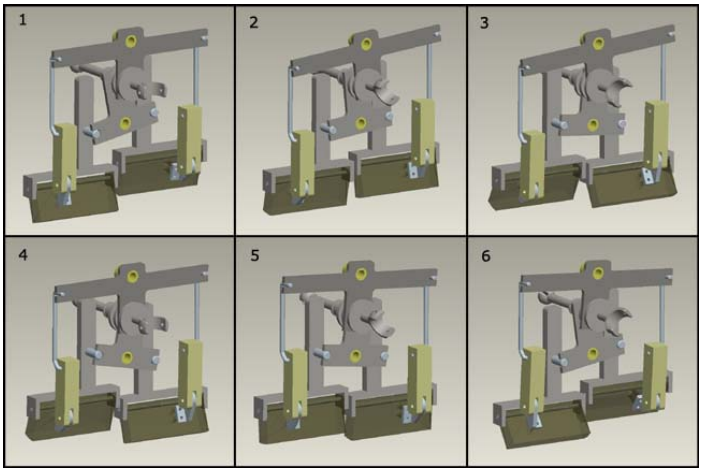


Figure 11: Foot Mechanism Detail

3.4.1 Design Options/Future Considerations

Before settling on this mechanism to impart forward locomotion, several alternatives were considered. The first, and most obvious choice was to use wheels or treads to propel the snake forward, however, wheels and treads were abandoned because they generally do not work well on rough surfaces. Another option was to use feet that were rigidly attached to the ribs and move the snake forward by articulating its body in specific sequences (gaits). This option was discounted because robots of this type move slowly, with forward speed measured in inches per minute.

After the decision was made to give the snake "feet", several methods were considered to power them. The main challenge was in converting rotary motion from the drive motor to the orbital motion observed in the foot. Moreover, we needed a device that would allow two feet to move 180 degrees out of phase from each other so that the rib was always in contact with the ground. The details in *figure 11* shows that the final solution to this problem is relatively complicated. The mechanism being designed for the next version of this robot will provide the same resulting foot motion, except that the foot mechanism has been simplified. We have reduced the part count from sixteen to eleven parts.

3.5 Servo Mechanism

While the drive motor and foot mechanism, along with integrated universal joint, are responsible for forward locomotion, the robot still requires additional mechanical components to allow for turning, cantilevering and other functions we associate with biological snakes.

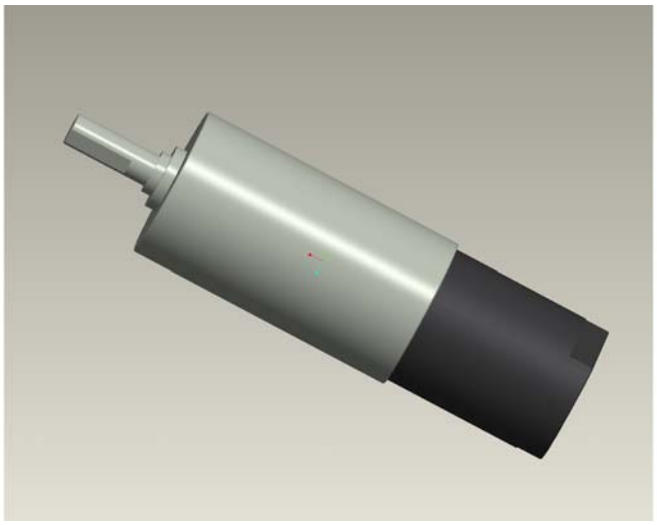


Figure 12: DC brush motor

To accomplish these functions, the prototype contains fourteen DC servomotors which allow the snake to articulate its segments with respect to each other. These motors, from Faulhaber [19], integrate a DC motor, four stage stainless steel planetary gearbox and 400 count per revolution quadrature encoder. This assembly, along with a Proportional-Integral-Derivative (PID) control algorithm, provides a complete closed loop position controller. *Figure 13* shows a block diagram of the complete servomechanism.

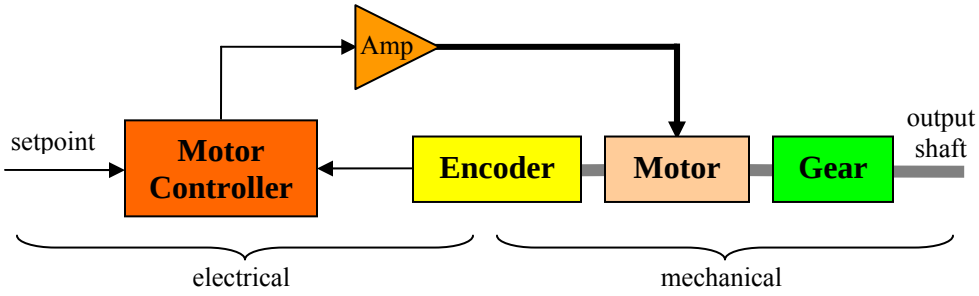


Figure 13: Servomechanism block diagram

The entire servo assembly consists of the servo motor, an additional pair of gears and a spool. As the motor rotates, the spool winds a control cable which ultimately pulls on neighboring segments, providing articulation of the robot. The control cable is 0.030" diameter nylon coated pre-stretched aircraft cable. This cable has a breaking strength of 80 lbs. and since it is pre-stretched, its elongation under load will be negligible. *Figure 14* shows the complete servo gear train.

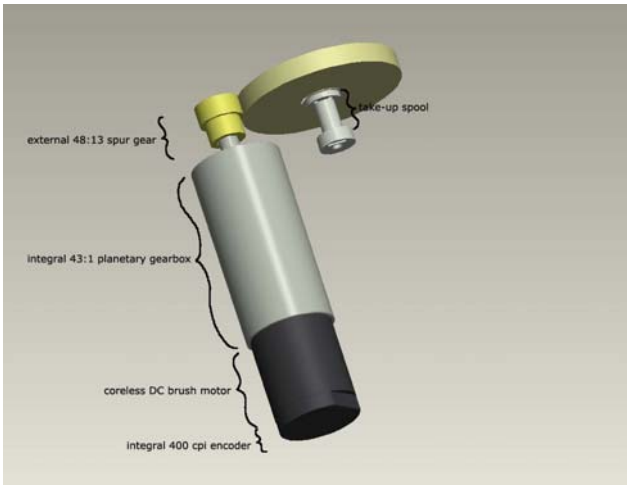


Figure 14: Servo Gear Train

For control purposes, it is necessary to relate encode counts to control cable take-up (i.e. the length of cable would around the spool). First, we will develop the input-output relationship between the encoder and spool.

$$\left(\frac{400\ counts}{revolution}\right)\left(\frac{43\ revolutions}{revolution}\right)\left(\frac{48\ revolutions}{13\ revolutions}\right)=\frac{63{,}508\ counts}{revolution}=63{,}508\ cpr \qquad (2)$$

From *equation 2*, every rotation of the output shaft (spool) requires 63,508 encoder counts. We are now interested in relating control cable take-up, in inches, to spool rotation. In other works, we will find how many inches of cable are required to wrap

around the spool one time. The radius of the take-up spool is 0.063". This will not be used in the calculation of the spool's *effective* circumference, however. Instead, we will take into account the thickness of the control cable which wraps around the spool. The effective radius of the spool will be a pitch circle that lies on the centerline of the cable as it wraps around the spool. *Figure 15*, we see that the effective radius is 0.093". With this dimension we can calculate the spool take-up per revolution *(equation 3)* and the overall relationship between motor position and cable length *(equation 4)*.

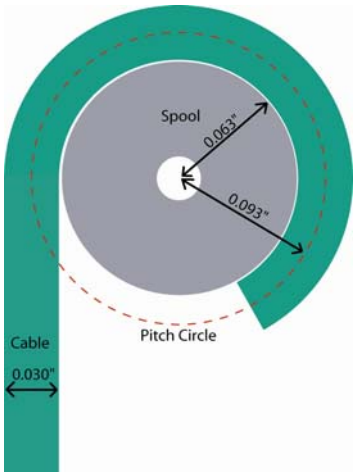


Figure 15: Spool Illustration

$$T_s = 2\pi \times 0.093" = 0.584" = \frac{0.584\ in}{revolution} \qquad (3)$$

$$\left(\frac{63{,}508\ counts}{revolution}\right)\left(\frac{revolutions}{0.584\ in}\right) = 109{,}000\ count/in \rightarrow \Delta l = \frac{C}{109{,}000} \qquad (4)$$

From equation (4), C is the number of encoder counts from home position and Δl is the resulting change in control cable length. The choice of these definitions will become clear in the following section.

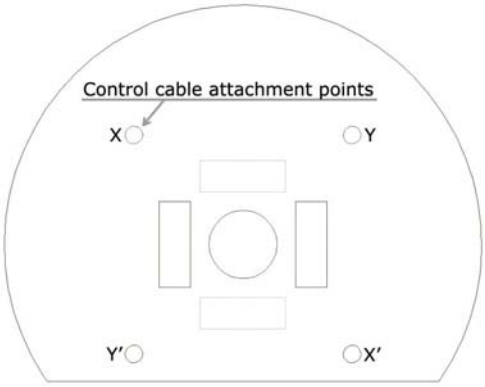


Figure 16: Control cable attachment points on a typical segment

Now we seek to develop an input-output function relating *motor position* to the *joint angle* of the segment being driven by the motor. We will first relate control cable length to joint angle. The control cables that articulate the snake's segments are oriented at forty-five degrees (45°) from the horizontal and vertical axes. Referring to *figure 16*, the cables attached at points X and X' operate as a pair. When the X-axis servo motor is activated, it retracts the X cable and extends the X' cable (or vice-versa). This causes the segment to bend in the X axis. *Figure 17* shows a side view of how these cables are rigged. The Y-axis servo operates in exactly the same manner.

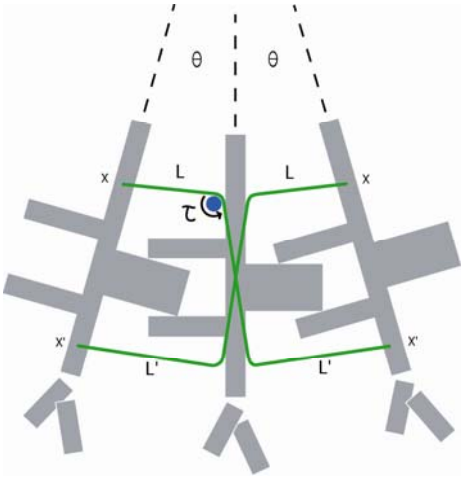


Figure 17: Control Cable Rigging

Under applied motor torque (τ) the length of the control cables vary such that the following relationships are held:

$$L = P + \Delta l, \quad L' = P - \Delta l \quad \rightarrow \quad L + L' = 2P \quad , \qquad (5)$$

where P is the pitch between two ribs (2" in our design) and Δl is the variation in cable length caused by the applied motor torque. The problem has been redrawn in a simplified form in *figure 18*.

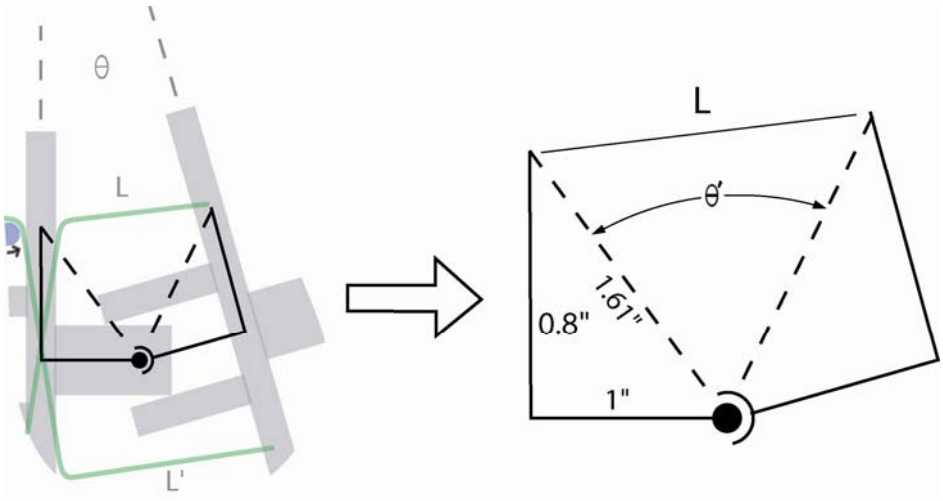


Figure 18: Geometric relationship between cable length and joint angle

From this, it is simple to relate cable length to joint angle. The result is given as,

$$\begin{aligned} &\theta' = 2\sin^{-1}(L/2\sqrt{2.6}) \qquad for: L = 2'', \quad \theta' = 76.7^\circ \\ &\theta = \theta' - 76.7^\circ \qquad \theta = 2\sin^{-1}(L/2\sqrt{2.6}) - 76.7^\circ \end{aligned} \quad . \qquad (6)$$

Combining (4), (5) and (6), we obtain the overall input-output function relating motor position (in encoder counts) to joint angle.

$$\begin{aligned} &\theta = 2\sin^{-1}\left(\frac{2 + C/109{,}000}{2\sqrt{2.6}}\right) - 76.7^\circ \\ &C = 109{,}000\left(2\sqrt{2.6} \times \sin\left(\frac{\theta + 76.7}{2}\right) - 2\right) \end{aligned} \qquad (7)$$

3.5.1 Design Options/Future Considerations

While the mechanism outlined on the previous section is effective, it has several drawbacks. The first issue is that control cable lifetime is greatly affected by bending. The 1/16" bending radius that this cables experiences in wrapping around the spool will have a significant effect on both the cable's load-carrying ability and the cable's life. Control cable manufactures recommend that pulley diameters have a radius in the range of sixteen times the cable diameter for adequate operation lifetime [10]. For a 1/32" cable, the spool radius would ideally be larger than half an inch. Due to size restrictions and the need for greater pulling force, we had to select a pulley with a significantly smaller radius. The second issue that is that the cables sustain abrasion during operation. The control cables pass through Teflon guide tubes that route the cables past obstructions on the segment. After prolonged operation, the nylon coating on the cable and the Teflon guide tube will become worn and require replacement. Regularly replacing all cabling and guide tubes would be very labor intensive.

Although the control cable method is acceptable for a proof-of-concept model, it is clearly not adequate for a second generation prototype.

Several solutions to this problem have been considered, and one has been chosen as the method to be used in the second version of the robot. Each set of control cables will be replaced with a single push-pull rod driven by a lead screw. As the lead screw rotates, the attached nut moves up or down, retracting or extending the attached control rod. This action causes articulation in the segment. The alternative design is depicted in figure 19. Not only does this method eliminate the shortcomings of the control cable method, but it also makes assembly of the robot much simpler.

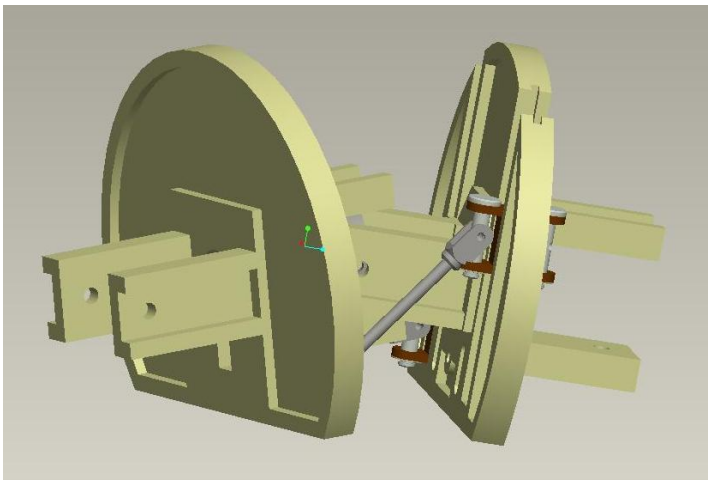


Figure 19: Push-pull rod design

4 Electrical Design

The top level view of the robot's electrical system is shown in *figure 20*. The robot consists of eight processor boards, seven *segment processors* and one *drive motor processor*. Each of these processors is connected through the Controller Area Network (CAN) bus. A PC is also connected to the network and can be used to display sensor data and issue commands. Due to the network topology employed here

(figure 17), we can implement the robot control strategy in a centralized manner, in a distributed manner, or using a hybrid technique. This will be discussed in detail in Section 5.4.

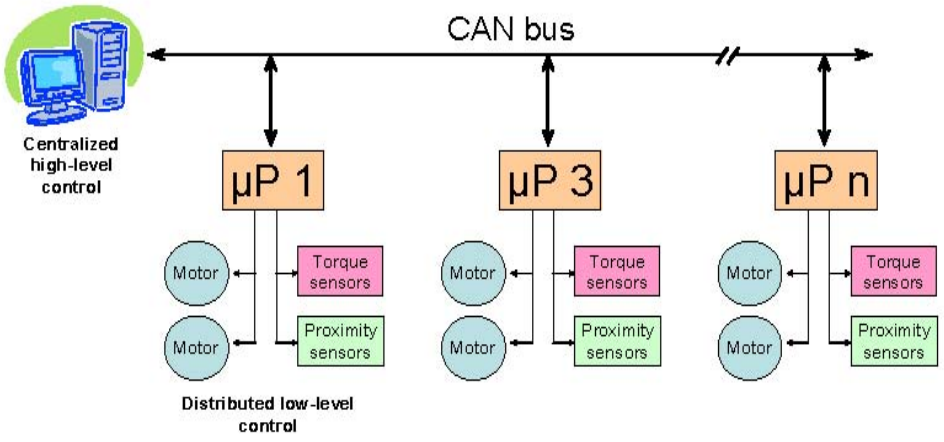


Figure 20: High-level electrical system view

The robot can be equipped with many navigational sensors including heading, torque and proximity sensors. Auxiliary sensors include temperature and humidity, pressure and vibration. In all, the robot can support approximately thirty (30) sensor inputs. With this number of sensors, and the amount of data they produce, it becomes advantageous to process and act on the data locally, rather than send it all to a central processor for analysis. For example, tactile sensors in the feet are continually sampled to monitor the robot's stability. If processors detect that the robot is not on a level footing, they can drive the servo motors in the proper direction so that the robot can achieve more uniform contact with the ground. Having much of the low level control done locally means that not all of the sensor data needs to be relayed to the operator, only that which is relevant.

4.1 Segment Processor Board

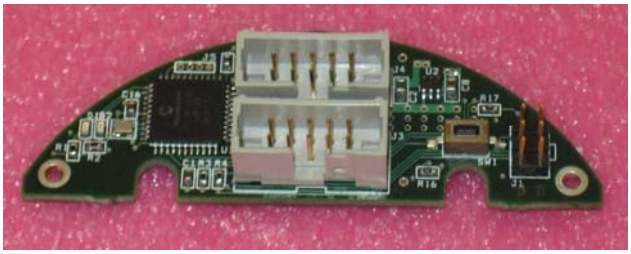


Figure 21: Segment processor PCB

Each segment of the robot contains a circuit board responsible for monitoring sensors, driving servos and performing computations related to that segment. Our robot has seven segments and as many processor boards. Each board contains a Microchip PIC18F4685 RISC processor [17] operating at 5 MIPS. The device contains a hardware Controller Area Network (CAN) transceiver that manages all network traffic. It also has internal FLASH memory, giving us the ability to reprogram the board in-circuit after firmware modifications. Currently, the main task of the processor is to calculate a two axis PID control algorithm responsible for driving the two motors located in the segment. To complete the servo motor loop, the PCB also includes two motor drive ICs and two quaderature encoder counter ICs to manage motor position. The board accepts four analog inputs, two of which are dedicated to motor current measurement. The remaining two channels can be used for other analog or digital sensor inputs. *Figure 22* is a block diagram of the hardware contained on the processor board.

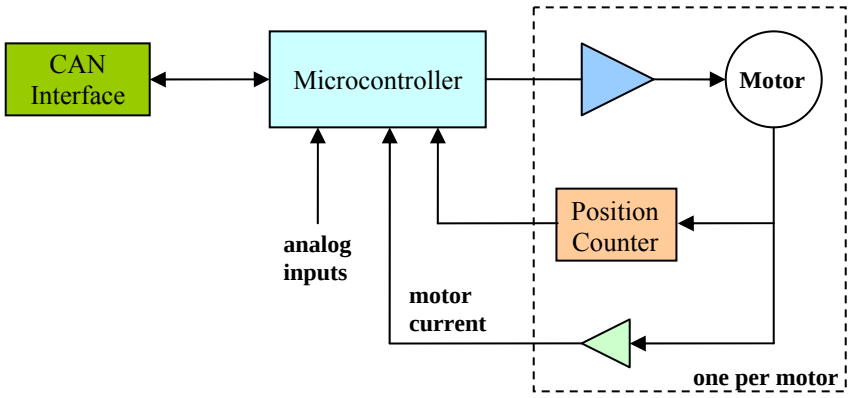


Figure 22: PCB block diagram

4.2 Drive Motor Processor Board

Figure 23: Drive motor PCB

In addition to the robot's segment processor boards, there is one drive motor processor located in the tail of the snake. The main task of this processor is to control the operation of the drive motor. In many ways, this PCB is a reduced function version of the segment processor. The code that runs on this processor is very similar to the code residing on the segment processors, with the major difference being that the drive motor board controls one motor, while the segment board controls two.

4.2.1 Design Options/Future Considerations

The microcontroller chosen for this design was chosen for its simplicity rather than processing power. This device is fast enough to provide the snake with basic functions, but does not have the resources to solve the complex kinematic and dynamic equations needed it our future investigation. Due to these limitations, the next generation of the robotic snake will contain a more powerful set of processors. The microcontroller currently being considered for use in the next generation robot is the LPC2119 from Philips [20]. This is a 60MHz ARM based microcontroller with built in dual CAN transceivers, pulse width modulation (PWM) outputs and several counter/timer channels. The 32-bit core will enable us to perform complex calculations quickly and the onboard peripherals allow us to reduce the circuit board's part count. This microcontroller also has extensive power management features including clock rate reduction, power-down modes and the ability to disable unused peripherals.

4.3 Communication Bus

This robot uses the industry standard *Controller Area Network (CAN)* [6] to move data between processors. CAN was originally developed for use in automotive applications and accordingly has been designed for noise immunity and fault tolerance. The physical bus is a differential pair, which inherently provides immunity to electrical noise because any noise induced in one wire will also be induced in the other, and this common mode signal is easily removed by the (differential) receiver amplifier. The interface between the bus and the microcontroller is accomplished with a CAN bus transceiver. This chip serves several important functions. It electrically isolates the bus from the processor boards to prevent bus noise and electrical faults

from effecting the processor board's operation. Additionally, the transceiver has the ability to automatically detect faults and disconnect the offending processor board from the bus so that it cannot render the entire bus inoperative.

CAN is a peer-to-peer protocol, meaning there is no bus master. Any node can initiate a data transfer with any other node. This is an important feature because certain distributed control topologies require the ability of segments to communicate directly with each other. Master-slave protocols like I2C [21] require that all communications are initiated by a pre-designated "master" device and thus, direct communication between two "slaves" is not possible. One slave would have to send data to the "master" and the "master" would have to relay that data to the second "slave". Obviously this arrangement is inefficient in a network that requires direct communication between any two nodes, such as ours.

CAN has a unique method of specifying which node is the recipient of an information packet. In many low-level protocols, each node would have a unique address. If node A wants to communicate with node B, A would transmit a packet onto the bus that contained node B's address in the packet header. All nodes on the bus would inspect the packet. If the packet destination address corresponded to that node's address, the node would store and process the packet, if not, the packet would be ignored. This concept is taken a step further in CAN. Rather than have a unique address, CAN nodes have multiple *acceptance filters*. Each transmitted packet has an *identifier* in its header that all nodes on the bus inspect. If this identifier matches one of the values stored in that node's acceptance filter register, the packet is received and processed. CAN allows different identifiers to have different priority levels both in the receiving

node's buffer and during bus arbitration. This addressing scheme allows for great flexibility in defining the upper layers of the communication protocol that the snake uses. More urgent messages can preempt less urgent ones and we now have the ability to send messages to one node, several or all simultaneously. Of the current commercially available low-level serial protocols, CAN seems the most appropriate for this application. The second generation of this device will also use this standard.

4.4 Control Topology

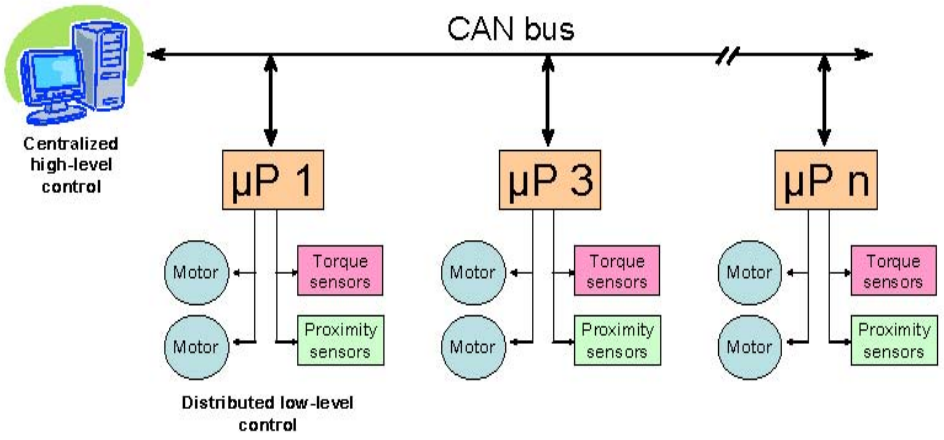


Figure 24: Control Network Topology

Figure 24 shows the control hierarchy used by the robotic snake. The robot's processor boards are networked to each other and to a personal computer (PC). In the current configuration, the PC issues commands to the robot and reads sensor data from the robot. Gait generation, sensor processing, and other "intelligent" tasks are executed centrally, by the PC. The robot's local processors are responsible for executing commands issued by the PC, but are not given the ability to process or react to sensor data. This division of labor, with the PC handling "high-level" tasks, and the snake's processors handling "low-level" tasks, is inefficient in the sense that the

snake's local processing power is not being fully utilized. Despite its inefficiency, however, this division has several benefits, especially in the early stages of control algorithm development. For example, the robotic snake consists of fifteen (15) actuators and approximately thirty (30) sensors. For this system, it is considerably simpler to implement a centralized control strategy than a distributed one. In a centralized control scheme, a single control algorithm processes all sensor data and generates all actuator signals. Numerous methods from classical control theory (e.g., lead-lag compensation, PID control) and modern control (e.g., pole placement, observer based controllers) can be implemented in this framework. Distributed control schemes, however, are often difficult to implement, partly because coupling of the system's dynamics necessitates communication between distributed control components. As a first step, we have employed a hybrid control approach, where the snake's distributed microprocessors execute a PID motor control algorithm, while the PC issues various commands to the snake and processes sensor data from the snake.

4.4.1 Instruction Set

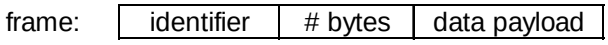
The control hierarchy currently implemented on the robotic snake divides high-level and low level control tasks between the PC and segment processors, respectively. The PC performs path planning and inverse kinematics computations and issues joint angle setpoint commands to the robot's distributed processors. These processors, in turn, take those commands and perform closed loop control of the robot's motors. In this sense there is a master-slave relationship between the high-level controller (the PC) and the low-level controller (the robot's processors), with the PC issuing

instructions, and the snake responding to them. The instruction set currently supported by our robotic system is summarized in the table below.

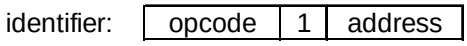
		Instruction	Description	Identifier	# Bytes	Byte 3	Byte 2	Byte 1	Byte 0
high	1	Stop	disable motors	00001100000	0				
	2	Reset	place nodes into reset	00010100000	2			0xAA	0x55
	3	Start	enable motors	00011100000	2			0xAA	0x55
	4	Report_freq	set new data report frequency	00100100000	1				freq (Hz)
	5	PID_freq	set PID update frequency	00101100000	1				freq (Hz)
	6	CAN_add	enter CAN addres set routine	00110100000	4		address	0xAA	0x55
priority	7	Setpoint	set new command position setpoint	000011aaaaa	4	motor 1 position		motor 0 position	
	8	Led_ON	turn LED 's' on	0001s1aaaaa	0				
	9	Led_OFF	turn LED 's' off	0010s1aaaaa	0				
	10	P_update	set new Proportional gain for motor 's'	0011s1aaaaa	4	proportional gain			
	11	I_update	set new Integral gain for motor 's'	0100s1aaaaa	4	integral gain			
	12	D_update	set new Derivative gain for motor 's'	0101s1aaaaa	4	defivative gain			
low	13	Store	store current parameters to ROM	011001aaaaa	2			0xAA	0x55

Figure 25: Robotic Snake Instruction Set

Data Frame Format



Every instruction issued on the CAN bus is packaged into a data frame, with each frame containing three fields as shown above. The first is the identifier field, the second is the number of data bytes in the frame, and the third is the data payload. Note that some instructions require no data payload.



The *identifier* field is the portion of the data packet that describes the instruction and recipient of that instruction. We have divided this eleven (11) bit field into three subfields as shown above. The five bit *opcode* subfield is the binary encoding of the instruction being issued. This subfield is followed by a delimiter field containing a binary '1'. This is added so that the data frame conforms to the bit stuffing rules of the CAN network (see [6] for details). The final subfield is the five bit *address* subfield, which identifies the node to which the instruction is being issued. In our system (which consists of seven segment processors and one drive motor processor),

snake processor boards are assigned addresses in ascending order, with address '1' given to the processor at the head of the snake, and address '8' given to the drive motor processor (at the snake's tail). Address '0' is reserved for *general call* instructions. When an instruction is issued to this address, it is acted on by all nodes in the network. Referring to figure 25, the listed instructions are divided into two groups. Instructions 1 through 6 are *general call* instructions, while 7 through 13 are *addressed* instructions. Due to the encoding scheme we have chosen, the system can support up to thirty two (32) general call instructions, and thirty two (32) addressed instructions. The network supports up to thirty one (31) nodes. A detailed description of each instruction is given in *Appendix B*.

4.5 Sensors

A robot of this complexity requires a variety of sensors to monitor stability, motor torque, battery life, obstacles and various environmental variables. The ability to measure these quantities allows the robot to sense its environment and adjust its operation accordingly. One of the main design goals of this project is to make a robot that is not only flexible and maneuverable, but also semiautonomous. Complex tasks, such as maintaining a stable footing when moving over uneven terrains, or ensuring that the torque exerted by a motor is within safe operating limits, should happen automatically. This allows the operator to concentrate on coarse grain control of the robot, such as telling it to move forward, look up, etc. It is "the robot's job" to ensure that while moving forward, it maintains contact with the ground, or that in looking up, it does not move its center of gravity to a location that would cause the robot to fall over. This section outlines the sensors found on this robot.

4.5.1 Tactile

Several of the semiautonomous behaviors that the snake will perform require that it be able to accurately measure the amount of force it exerts on a surface. The ground hugging behavior forces the snake to adjust its body to maintain even force distribution across all of its feet. The stability monitoring behavior requires that the snake detect situations where several of its segments begin to lift off of the ground. To accomplish tactile sensing, the foot brackets located on the bottom of each segment have been constructed to behave as miniature electronic scales. Force applied to the foot causes a deflection in the bracket. This deflection is measured by a foil strain gauge bonded to the bracket. The amplified signal from the strain gauge represents the force exerted on the ground by that foot. Figure 26 shows a segment experiencing a force on each of its feet. This force causes a small deflection in the foot bracket, which is sensed by the attached strain gauge. Note that the left and right feet have their own strain gauges and can measure force independently.

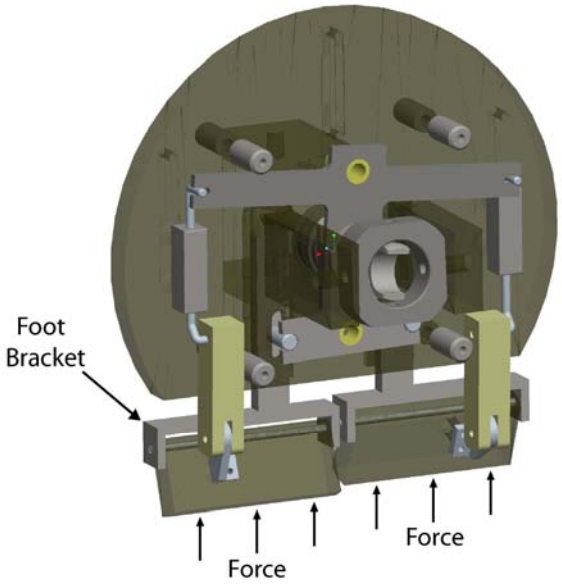


Figure 26: Force Measurement and Foot Bracket Location

Figure 27 shows a foil strain gauge [18] similar to the one used on this robot. Strain gauges of this type are made by embedding a foil measurement element in a polyimide substrate. The device is approximately 50um thick and is bonded directly to the specimen whose strain is to be measured. As the specimen experiences strain, so too does the foil element. This strain subsequently causes a change in the element's resistance which is converted to a voltage and amplified by a conditioning circuit. As was mentioned earlier, we measure strain on the foot bracket to determine applied force. To determine the optimal location of the strain gauge, we performed a finite element analysis (FEA) on the bracket. This test shows graphically where the largest strains are experienced. Naturally, to get the best sensitivity from the device, we place the gauge at the location that experiences the largest strain under applied load. Figure 28 shows the results of a FEA performed in ProEngineer. In simulation, force was applied at the bottom of the bracket, while the top of the bracket was held fixed. Under applied load, the software calculates the amount of strain experienced at numerous points on bracket, and the results are shown as a superimposed color gradient. The colors at the top of the color chart (on the right hand side of figure 28) represent regions of highest strain. Not surprisingly, we see the greatest strain at the location where the vertical and horizontal sections of the bracket meet. This is the location where the strain gauge is bonded.

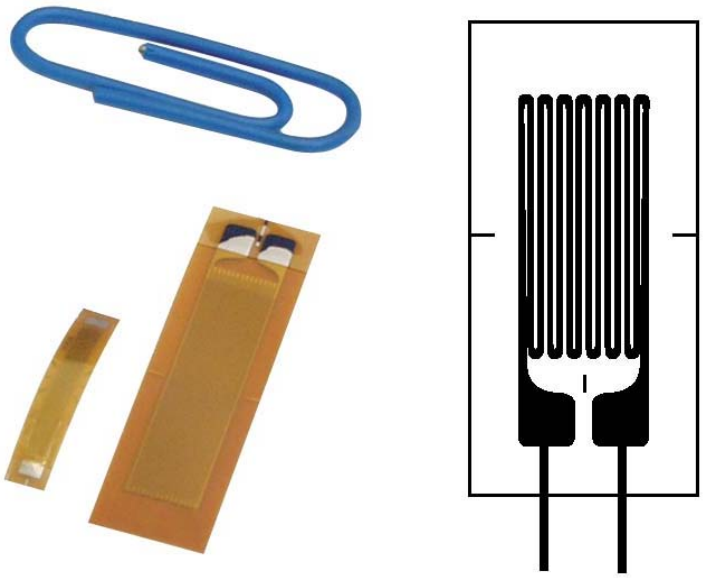


Figure 27: Strain Gauge Image and Schematic [18]

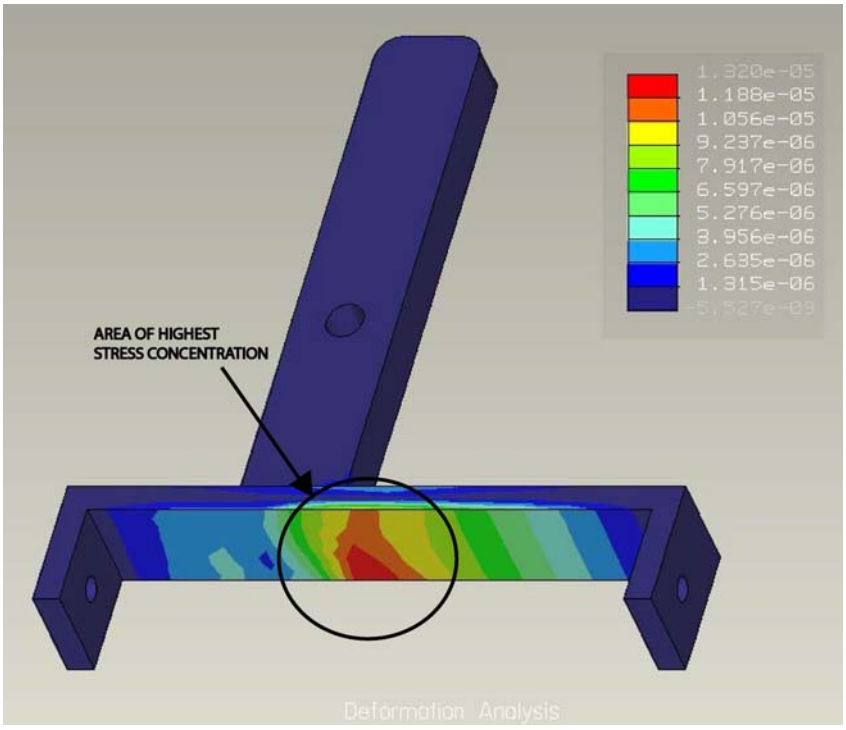


Figure 28: Finite Element Analysis of Foot Bracket (6 lb. applied)

Resistive strain gauges can be wired in one of three ways. The simplest method is to apply a single strain gauge to the specimen and have that element act as one arm in a Wheatstone bridge, as shown in figure 29. This is known as the *quarter bridge* configuration [18]. The advantage of this method is that it only requires a single sensing element, which reduces device cost and complexity. The quarter bridge sensor has several drawbacks including non-linear output and temperature sensitivity. These effects can be corrected in hardware or software. In practice, the nonlinearity of the signal is very small and can often be ignored. If accurate measurements are required, temperature compensation is generally required.

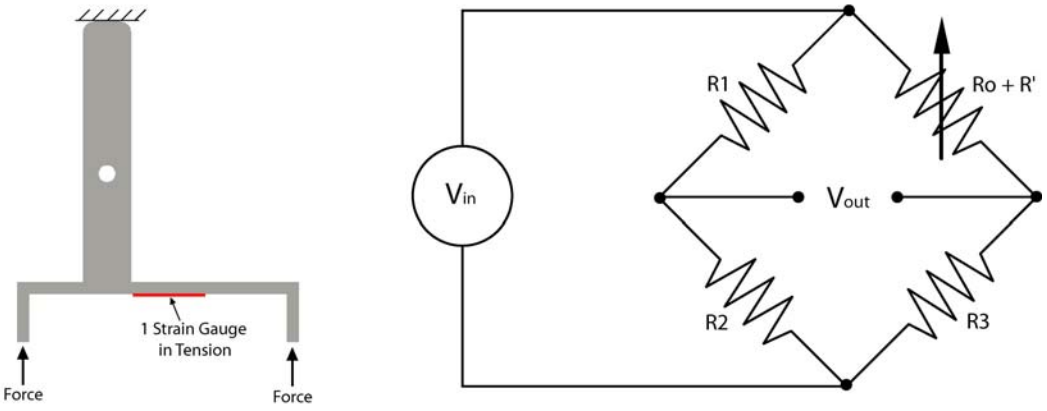


Figure 29: Quarter Bridge Configuration

An alternative wiring method is the *half bridge* configuration [18] (figure 30). This device uses two strain gauges mounted such that when one is in tension, the other is in compression. The result is that the resistance of one element increases due to applied strain, while the resistance of the other element decreases. Figure 30 shows the physical mounting of the strain gauges and a wiring diagram for the half-bridged. The half bridge configuration has several major advantages over the quarter bridge

configuration [18]. The first is that the output voltage swing using this method is approximately twice that of the quarter bridge. Additionally, the output signal is now a linear function of strain. The final benefit of this configuration is that it has built in temperature compensation. If both sensors are in the same environment (as is the case here) then ambient temperature changes will effect both sensors equally. Since the resistance of the two sensors scale equally with temperature change, the resistance ratio for a given strain remains constant. The obvious drawback of this method is the need to mount two gauges in locations that experience equal and opposite strain. In some cases this may not be possible, but the design of the foot bracket makes this type of mounting simple.

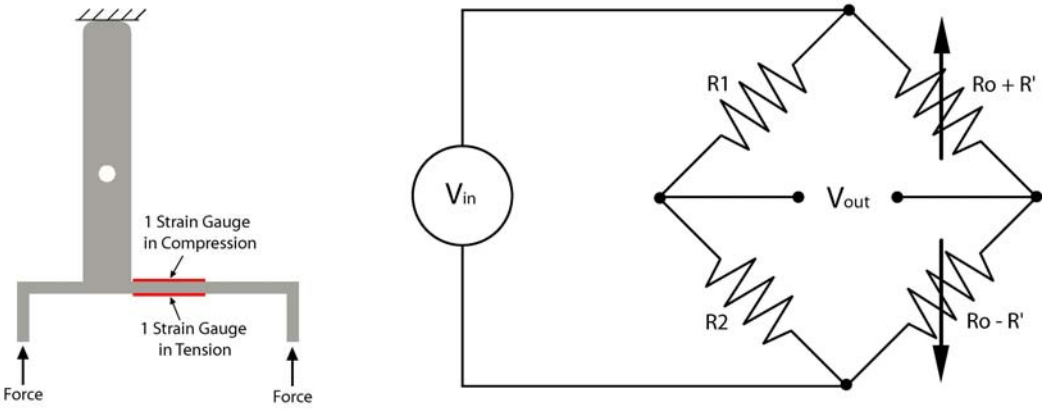


Figure 30: Half Bridge Configuration

The final configuration that can be used for the strain sensor is the *full bridge*. This device uses four active elements and is wired as shown in figure 31. The benefits of this method are similar to those of the half bridge. By using four strain gauges, we can obtain an output signal that is four times the amplitude of the (linearized) quarter bridge sensor.

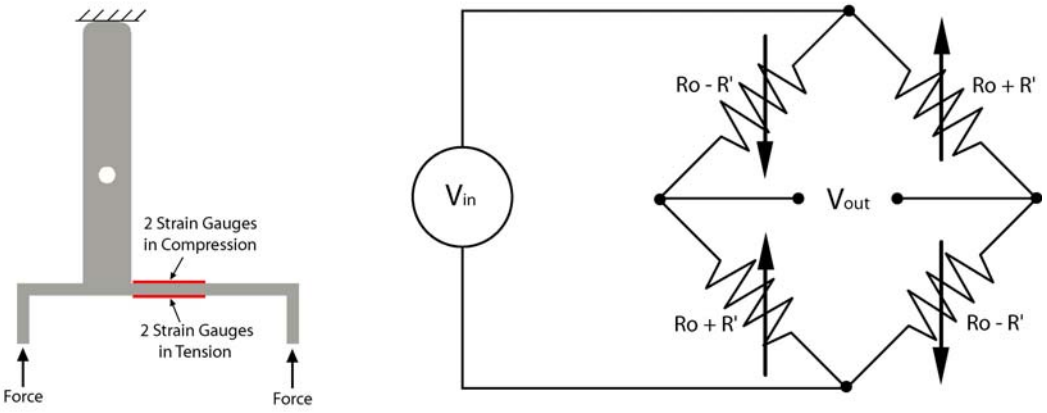


Figure 31: Full Bridge Configuration

In this design, we decided to use a half-bridge configuration because it provides a good balance between sensitivity and complexity. While the quarter-bridge configuration is the simplest alternative, the output voltage from the device is not linear with applied strain, so additional compensation and calibration would need to be performed. We also ruled out the use of the full-bridge transducer because our space constraints will not allow us to fit four strain gauges in the area of limited area of the foot bracket. On the other hand, the two element design provides a linear output and can easily fit on the foot bracket. It has twice the sensitivity of the quarter-bridge design, but only half the sensitivity of the full-bridge design.

We conclude our discussion of strain gauge configurations by deriving the equations that relate change in sensor resistance (due to applied strain) to output voltage methods. The equations are derived based on the Wheatstone bridge in figure 32. Equation 8 gives the input-output relationship of the bridge with four arbitrary resistor values.

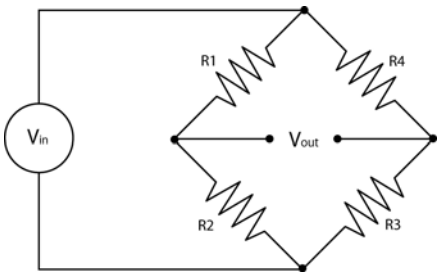


Figure 32: Wheatstone Bridge

Wheatstone Bridge:

$$v_{out} = v_{in}\left[\frac{R4}{R3+R4} - \frac{R1}{R1+R2}\right] = v_{in}\left[\frac{R4(R1+R2) - R1(R3+R4)}{(R1+R2)(R3+R4)}\right] \quad (8)$$

In a strain gauge application, one or more of the resistors R1-R4 are replaced with resistive strain gauges, resulting in the four possible configurations shown in figures 29-31. In the quarter bridge configuration, resistor R4 is replaced with a strain gauge having nominal resistance R_0. Under applied strain, the resistance of the device varies by ΔR. Resistor R3 is set to the strain gauge's nominal resistance, R_0, while R1 and R2 are set to any equal value, R. Equation 9 shows the input output relationship for the quarter-bridge. Note that the relationship is non-linear, but for small resistance variations, can be approximated by a linear equation.

Quarter – Bridge Strain Gauge:

$$R1 = R2 = R, \quad R3 = R_0, \quad R4 = R_0 + \Delta R \quad (9)$$

$$v_{out} = \left[\frac{(R_0+\Delta R)(2R) - (R)(2R_0+\Delta R)}{(2R)(2R_0+\Delta R)}\right] = v_{in}\left[\frac{1}{2}\frac{\Delta R}{2R_0+\Delta R}\right] \approx v_{in}\left[\frac{\Delta R}{4R_0}\right]$$

In the half-bridge configuration, resistors R3 and R4 are both replaced by strain gauges. Note that when the specimen is strained, one gauge experiences tension (its resistance increases) while the other experiences compression (its resistance decreases). The resistors R1 and R2 are again set to an equal value. Equation 10

shows the input-output relationship for the half-bridge. In this configuration, the output voltage is linearly proportional to change in resistance (through strain of the specimen) and its sensitivity is twice that of the (linearized) quarter-bridge.

$$\begin{aligned}&\textit{Half}-\textit{Bridge}\ \ \textit{Strain}\ \ \textit{Gauge}:\\&R1=R2=R,\quad R3=R_0-\Delta R,\quad R4=R_0+\Delta R\\&v_{out}=\left[\frac{(R_0+\Delta R)(2R)-(R)(2R_0)}{(2R)(2R_0)}\right]=v_{in}\left[\frac{\Delta R}{2R_0}\right]\end{aligned}\qquad(10)$$

Finally, in the full bridge configuration, all four resistors are replaced with strain gauges. Equation 11 shows that this configuration provides an output that is linear with strain, and it is four times as sensitive as the half-bridge.

$$\begin{aligned}&\textit{Full}-\textit{Bridge}\ \ \textit{Strain}\ \ \textit{Gauge}:\\&R1=R3=R_0-\Delta R,\quad R2=R4=R_0+\Delta R\\&v_{out}=\left[\frac{(R_0-\Delta R)(2R_0)-(R_0+\Delta R)(2R_0)}{(2R_0)(2R_0)}\right]=v_{in}\left[\frac{\Delta R}{R_0}\right]\end{aligned}\qquad(11)$$

4.5.2 Motor Current

In addition to being able to measure the external forces exerted on the snake, it is important to know the internal forces acting on the robot's segments. Specifically, we would like to know what types of torques are being applied to the segments by the servo motors. We measure these torques indirectly by taking advantage of the fact that the torque applied by a motor is directly proportional to the current drawn by the motor. In this section, we will discuss two simple methods of current measurement that were considered for use in the robot.

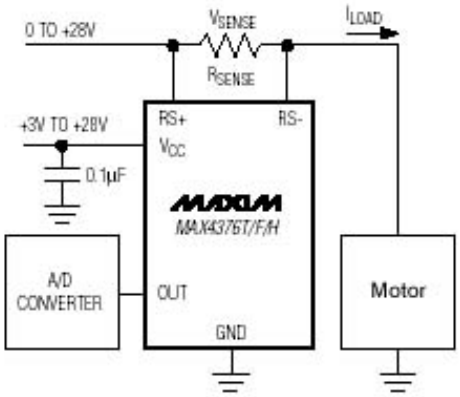


Figure 33: Typical Current Sense Amp Application [14]

The most common method of measuring small currents is by placing a small resistance in the current path and measuring the voltage drop across that element. Several manufacturers offer current sense amplifiers made specifically for this. Figure 33 shows the MAX4376 current sense amplifier [14] in a typical application. This device is made to operate as a *high side* current sense amplifier. This means that the sense resistor is placed on the positive side of the load. An alternative method is to use a *low side* configuration, where the sense resistor is placed on the low potential side of the load. The former method has the advantage that it does not introduce impedance into the ground path, a necessity for noise minimization [15]. The disadvantage of this configuration is that one must use an amplifier with high common mode rejection ratio (CMRR). The latter method has the advantage that it can be constructed of inexpensive amplifiers, but because of the location of the sense resistor, the ground path resistance is increased. This can potentially cause excessive noise in the circuit.

There is another simple method of current sensing that circumvents the disadvantages of resistive current sensing. This method uses a magnetic field sensor (often a Hall Effect sensor) convert magnetic field that results from the flow of current in a conductor into an electrical signal. Figure 34 shows a block diagram of the Allegro ACS704 Hall Effect current sensor [15]. The currents measured in our application are relatively small, on the order of tens of milliamps. Hall Effect current sensors are generally made for larger current spans. Using such a device in our design would require an additional amplifier before A/D conversion, and the signal chain would suffer from low SNR. Using a device such as the MAX4376, on the other hand, allows us to sense current and amplify the signal in one process. Additionally, resistive current sensors provide excellent noise performance. For these reasons we have decided to use resistive current sensing. We had originally intended to implement high side current sensing, but because of the construction of the motor driver IC, we were forced to use the low side sensing technique. The disadvantage of this method (added impedance and noise in the ground path) will not be an issue for us because the motor return is not shared by any of the sensitive circuitry.

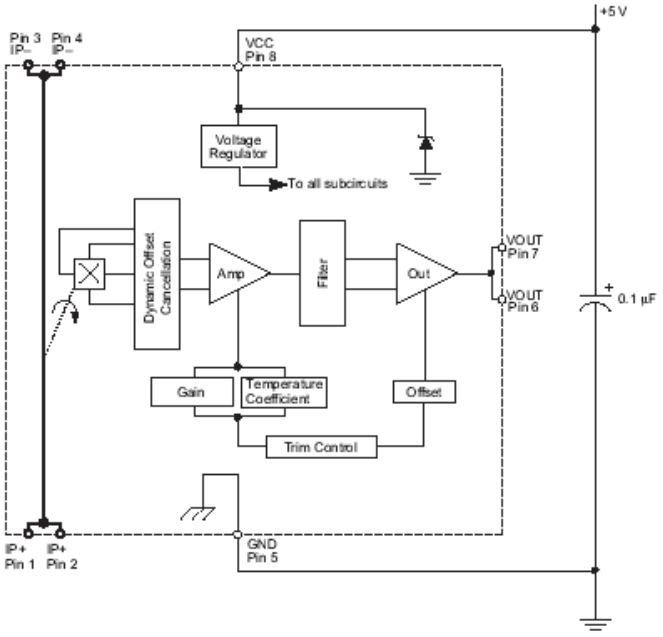


Figure 34: Block diagram of Allegro Hall Effect current sensor [15]

4.5.3 Heading/Orientation

An important feature of many autonomous robots is the ability to accurately measure its heading and orientation. This information is necessary for both navigation and control. The robotic snake accomplishes heading/orientation sensing with an *inertial measurement unit* (IMU) mounted in its tail. An optional second IMU can be placed in the head of the snake. This device consists of a solid state six degree of freedom (6DOF) sensor with 3-axis magnetometer. The 6DOF sensor consists of six components, three microelectromechanical system (MEMS) accelerometers measuring linear acceleration in the *x, y and z directions*, and three MEMS rate gyroscopes measuring angular acceleration about the *x, y and z axes*. This sensor measures acceleration in all six degrees of freedom. Finally, the three axis magnetometer has three mutually orthogonal magnetic field sensors that together resolve the direction of magnetic north. When the robot is relatively stationary, the

accelerometers can be used to resolve the direction of gravity. With these two directions determined, the robot can completely determine its heading and orientation (while stationary). When the robot is moving, heading data is derived primarily from the rate gyroscopes. Signal processing techniques are used to determine how best to combine input from all nine sensors to accurately determine the robot's heading.

4.5.4 Object Detection Sensors

As the robot navigates through its environment, it is necessary that it be able to detect obstacles in its path. The sensors that have been discussed so far are primarily used for autonomous control of the robot. We will now introduce several proximity sensors that are useful in detecting obstacles in the robot's path.

Figure 35: Sharp GP2D120 sensor [22]

Figure 35 shows the Sharp GP2D120 IR proximity sensor [22]. This device has a 1.5" - 12" sensing range. Several of these devices will be placed along the snake's length, providing object detection on all sides of the robot. The sensor gives an analog output signal that can be digitized directly with one of several available analog to digital converter (ADC) channels on each of the snake's segment processor boards. Many infrared (IR) proximity sensors provide a binary output signifying only if an object is present, not its distance. Since this robot will function as a path planning and object avoidance platform, it is often necessary that we not only detect the presence of obstacles, but how far they are from the robot.

The ability to feel the ground and detect objects gives this robot enough information to make rudimentary control decisions. For example, if the operator instructs the robot to move forward, the robot will automatically adjust its joint positions so that it "hugs" the ground as it moves forward. In addition, if the robot sees an obstacle in its path, it can automatically lift its head in an attempt to climb over the object. While these basic behaviors free the operator from many laborious control tasks, the robot still relies heavily on the human operator to provide basic instructions (e.g. "move forward") and intervene when the robot gets stuck. This requires that the operator be able to see clearly what lies in the robots path. In future versions of the robot, we intend to include a miniature CMOS video camera in the head of the snake to relay video to the operator. Figure 36 shows the placement of IR sensors and camera along the robot's length. Four forward looking IR sensors and one forward looking camera are the robot's primary means of object detection. Eight additional side looking IR sensors are also included in the design.

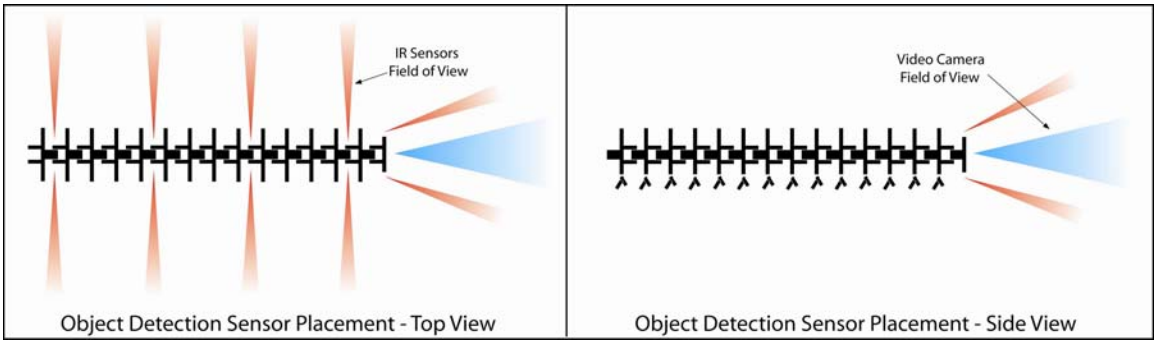


Figure 36: IR Sensor and Camera Field of View

4.5.5 Special Purpose Sensors

The sensors discussed in previous sections were necessary for the basic operation of the robot. Depending on the applications the snake will be used in, it may be desirable

to include other special function sensors intended mainly to provide the operator with environmental information following is a partial list of sensors that *could* reasonably be incorporated into the snake per specific application. In addition to this list, any sensor with a small form factor (less than approximately one inch cube) and a compatable output can be interfaced to the robot.

- Relative Humidity/Temperature
- Gas (CO, CO^2, Methane, etc…)
- Flame and Smoke

4.6 Power and Communications

During the robot's design, provisions were made to allow the device to operate un-tethered. The robot can carry batteries that give it a run-time of approximately 1.5 hours. In addition, a wireless communication module can be added to the CAN bus to allow radio control of the robot. At this stage of development, however, we provide power and communication via a tether. This was decided so that issues of communication bandwidth and battery runtime would not interfere with control algorithm development. As the design matures, we will transition to an un-tethered robot.

5 Software

The discussion thus far has centered on the physical design of the robot. This section discusses the software components being developed to control the robot, and manage data flow between the robot and the user. The processes that reside on the robot's processors are divided into two categories, *background processes* and *foreground process*. Background processes are those that provide low level control of the robot

by monitoring tactile and torque sensors, while making incremental adjustments to individual servo motors. These processes include stability monitoring, ground hugging and torque monitoring. Foreground processes are those that provide more advanced control mechanisms to allow the robot to avoid obstacles and negotiate rough terrain. Software development is still in its early stages and this section introduces the main applications that are being develpoed.

5.1 Background Processes

5.1.1 Stability Monitoring

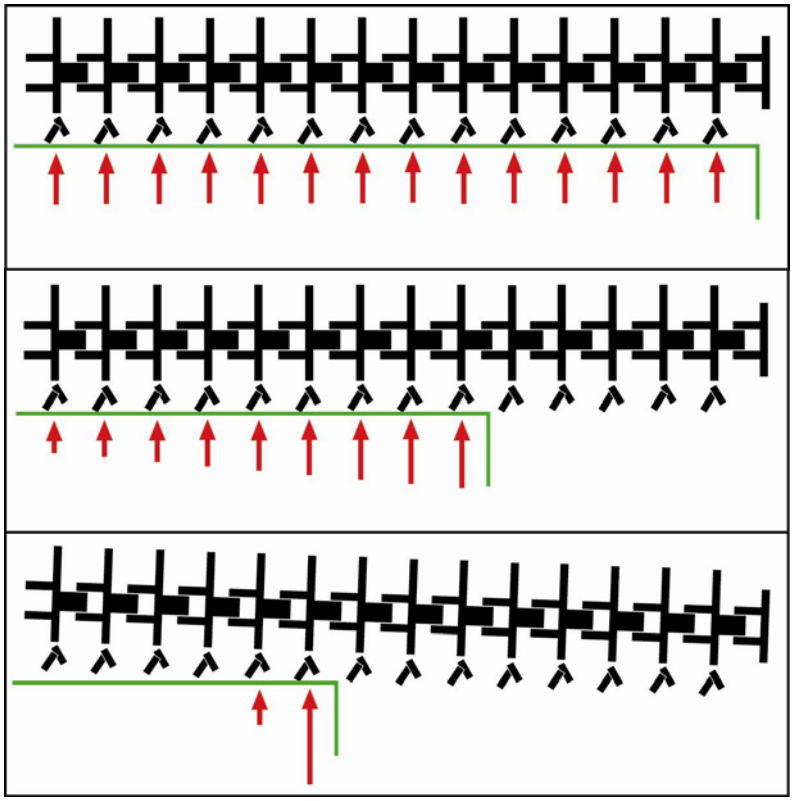


Figure 37: The Concept of Stability Monitoring

The first background process we look at is stability monitoring. Through the use of its tactile sensors, the snake continually monitors the force it exerts on the ground. If the snake senses an uneven force distribution over its segments, it will alert the operator.

Figure 37 shows an example of stability monitoring. When the snake is lying firmly on a surface, each of its feet exerts an equal force on that surface (panel 1). As the robot begins to cantilever, for example over the edge of a table, fewer feet are supporting it and the force distribution is no longer uniform. The second panel shows this case. As a result, the forces exerted by the feet closest to the edge are largest. If the snake proceeds much further, it will fall over the edge of the table (panel 3). By monitoring the forces exerted on its feet, the snake can prevent itself from falling of the table.

5.1.2 Ground Hugging

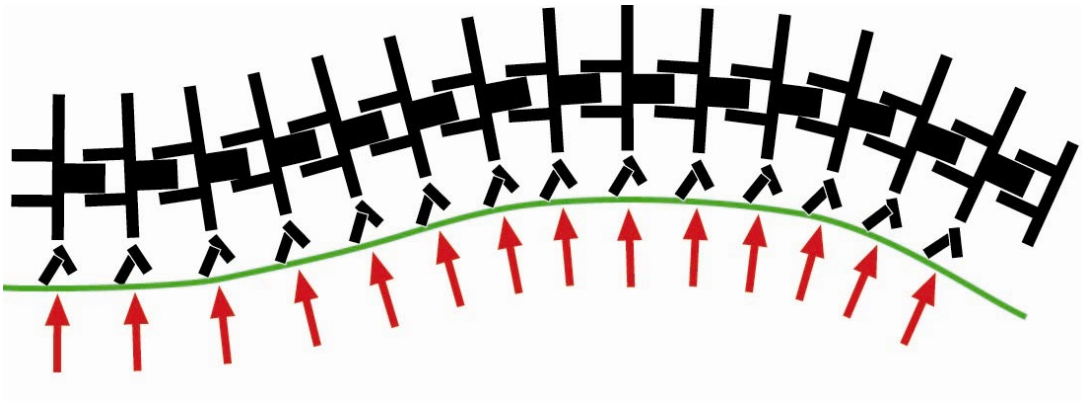


Figure 38: Ground Hugging

As the snake moves forward (as instructed by the operator) it should naturally follow the contour of the ground beneath it. This process, much like stability monitoring, requires reading the force sensors and driving servo motors so that the robot maintains uniform contact with the ground. Figure 38 shows a case when all of the snake's segments are touching the ground. As the snake crawls forward, it monitors these sensors to ensure its segments maintain uniform contact. Data reported by these sensors can also be used to form a map of the underlying ground. As the snake bend

to maintain contact with the ground over which it crawls, its shape reflects the contour of the underlying ground.

5.1.3 Motor Torque Limit

Limiting the amount of torque produced by the motors is done primarily to protect the motors and other mechanical components from damage. This is a straight forward task of monitoring the motor current sensors and taking action when the current draw exceeds some set threshold. This process can be augmented by understanding the dynamics of the robot. For example, if the snake cantilevers over the edge of a table (figure 37), large torques are required by the motors located near the table's edge. Knowledge of the robot's dynamic model can predict the amount of torque required, thereby setting a maximum cantilever distance determined by maximum motor current.

5.2 Foreground Processes

5.2.1 Kinematic Equation Calculations

Several mathematical models that describe the physical behavior of this robot have been developed. The kinematic model relates joint angles to segment positions. For example, if given the angles of all segments in the robot, the kinematic model can calculate the position, in Cartesian space, of any segment. The inverse kinematic model can take a desired segment position and determine the required joint angles to place the segment correctly. The inverse kinematic equations are necessary in performing basic locomotion and curve following tasks. In a typical scenario, the operator would specify a curve in space along which the snake is to crawl. This curve would be fed into a *fitting algorithm* that uses the kinematic equations to determine

how each of the snake's joints must be bent in order to fit to the curve. As the snake crawls forward, the fitting algorithms continually re-fits the snake with a slightly advanced starting point. At this point, tactile sensor data can be used in order to "fine tune" the snake's joint angles so that uniform contact is maintained with the ground. Simultaneously, the curve that was originally fed into the fitting algorithm is also fine tuned to reflect the contour of the ground over which the snake crawls. Currently, the processes of curve generation and snake fitting are executed on the host PC, and joint angle commands are issued to the snake incrementally as it crawls forward. We are currently working to divide the algorithm in a way that makes it suitable to execution in the distributed processing architecture of the robotic snake.

6 Specifications and Performance

The robot was designed so that its length can be adjusted to fit a particular mission. The mechanical and electrical specifications given below assume a robot seven (7) segments long.

6.1 Mechanical Specifications

Parameter	Min	Typ	Max	Units	Notes
Length	-	31.0	-	in	seven segments and drive motor
Diameter	-	3.7(1)	-	in	
Mass (single segment)	-	10.7	-	oz	
Mass (robot)	-	4.99	-	lb	including drive motor
Forward Speed	-	2	4(2)	in/sec	
Cantilever Distance	-	-	8(3)	in	based on motor strength
Bending Radius	6	-	-	in	

Notes:
1. While the diameter of each segment's rib (figure 4) is 3.5 inches, the addition of the foot increases its effective diameter to 3.7 inches.

2. This maximum forward speed has been imposed during experimentation to limit wear on mechanical components. The absolute maximum forward speed is determined by choice of drive motor.
3. The cantilevering operation is depicted in figure 37. Cantilever distance is defined as the maximum overhang distance for which articulating motors are powerful enough to support the snake. The actual distance that the snake can successfully cantilever is also dependant on its length. In the absence of motor torque limitations, the snake can cantilever at maximum, half of its body length.

6.2 Electrical Specifications

Parameter	Min	Typ	Max	Units	Notes
Supply Voltage	6	-	10	volts	
Standby Current	-	386(1)	-	mA	motors off
Running Current	-	-	3(2)	A	
Processor Speed	-	-	5	MIPS	
Communication bandwidth	-	-	1	Mbps	

Notes:
1. In standby mode, all electrical systems are powered with the exception of motor drivers.
2. Running current is dominated by motor current draw. Certain maneuvers require greater current demands than others. In experimentation, this number is the largest observed current draw.

6.3 Test Results

One of the main goals of this project was to design a robot capable of accessing confined spaces and operating on rough terrains. Several tests were conducted to assess the robot's capabilities in such scenarios. While the objective of these tests is to demonstrate basic capabilities of the robot, we will present performance measures where appropriate (e.g., time to complete maneuver, maximum current draw).

In each test, a predefined trajectory in space was defined and a MATLAB routine was used to calculate the articulations required for the snake to adhere to and crawl along the curve. In this sense, the behavior of the robot is open loop. It assumes perfect knowledge of the terrain over which is it crawling, and that there is no slip as it

moves over the surface. In future experimentation, sensor input will be used to allow the robot to cope with uncertainties in its surroundings.

6.3.1 Motion in the Plane

In this test, the robot follows a predefined planar path. The "S" shaped path, shown in figure 39, has a bending radius of approximately eight inches.

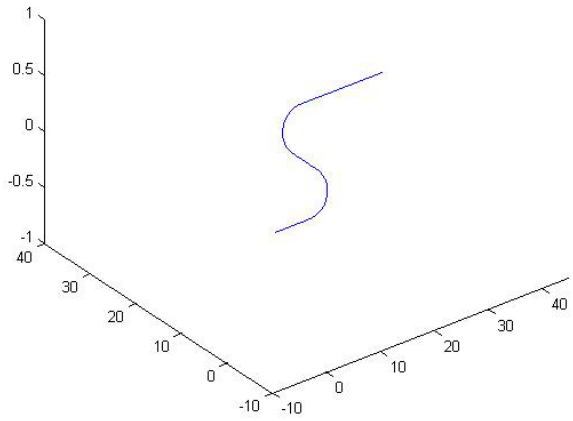


Figure 39: Trajectory followed in planar movement test

Figure 40: Robot during planar movement test

During this maneuver, the robot crawled a distance of sixty (60) inches in approximately forty (40) seconds. The average current draw was 1.2 amps.

6.3.2 Cantilevering/Gap Crossing

In this maneuver, the robot crawls across a seven (7) inch gap (figure 40). This behavior is important because it arises frequently in situations where the robot attempts to traverse uneven terrains (e.g. crawling over large rocks). While this test shows that this robot can successfully cross gaps of approximately seven inches, laboratory tests have shown that articulating motor strength is sufficient to cantilever approximately eight (8) inches of the robot's length. If the robot is not sufficiently long, its center of mass will prevent it from crossing gaps of this length. In these cases, the robot will tip over before the actuators saturate. This illustrates a situation where the operator may want to adjust the robot's length to suite a particular mission.

As the robot crosses a gap, its current draw steadily increases, due to the increased torque required by the articulating motors to keep the cantilevered section of the snake extended horizontally. In experimentation, the robot required a maximum torque of three (3) amps during this maneuver.

Figure 41: Robot crossing a seven (7) inch gap

7 Conclusion

This paper presents the design, analysis, and construction of a snake-like robot. During the design stage, simulation tools were used to verify proper operation of the mechanism, and after construction, several experiments were conducted to verify performance. A robot of this type is well suited to exploration of confined spaces, and rough terrains. This type of robot also serves as a useful research platform for testing object avoidance and path planning algorithms, distributed control algorithms, and sensor fusion algorithms. Future work with this platform will include the addition of tactile and object detection sensors and the implementation of distributed control algorithms. A second version of the device is also being designed. This version will feature wireless operation (both power and communications), overall reduction of weight, and reduced mechanism complexity.

List of References

[1] K. Dowling, "Limbless locomotion: Learning to crawl with a snake robot," doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA, 1997.

[2] I. Erkmen, "Snake Robots to the Rescue!" *IEEE Robotics and Automation Magazine*, September 2002, pp. 17-25.

[3] G. S. Chirikjian, "The Kinematics of Hyper-Redundant Robot Locomotion," *IEEE Transactions on Robotics and Automation*, December 1995, pp.781-793.

[4] C. W. Eno, "Jormungand, an Autonomous Robotic Snake," technical report, Machine Intelligence Laboratory, University of Florida, 1998

[5] H. Choset, W. Henning, "A Follow-the-Leader Approach to Serpentine Robotics," *ASCE Journal of Aerospace Engineering*, 1999.

[7] *TR1000, 916.50 MHz Hybrid Transceiver Datasheet*, RF Monolithics, Dallas, TX, 1999; www.rfm.com

[6] *CAN Specification Version 2.0*, Robert Bosch GmbH, Postfach, Germany, September 1991

[8] F. Hartwich, A. Bassemir, "The Configuration of the CAN Bit Timing," *6th International CAN Conference*, 2000

[9] B. Müller et al., "Fault Tolerant TTCAN Networks," *8th International CAN Conference*, 2002

[10] *Cable Design Guide, DG0406*, Carl Stahl Sava Industrial, Inc., Riverdale, NJ; www.savacablr.com

[11] *Dual Axis ±1.7g Accelerometer with SPI Interface*, Analog Devices, Inc., 2005; www.analog.com

[12] *Low Cost ±300 %sec yaw Rate Sensor with SPI Interface*, Analog Devices, Inc., 2005; www.analog.com

[13] *1 and 2-axis Magnetic Sensors - HMC1001/1002, HNC1021/1022*, Honeywell Sensor Products, 2000; www.ssec.honeywell.com/magnetic

[14] *Single/Dual/Quad High-Side Current-Sense Amplifiers with Internal Gain*, Maxim Integrated Products, Sunnyvale, CA, 2001; www.maxim-ic.com

[15] *Fully Integrated, Hall Effect-Based Linear Current Sensor with Voltage Isolation and a Low-Resistance Current Conductor*, Allegro Microsystems, Inc.; www.allegromicro.com

[16] *Dual Full-Bridge PWM Motor Driver, 3966*, Allegro Microsystems, Inc.; www.allegromicro.com

[17] *28/40-Pin High-Performance, Enhanced Flash Microcontrollers w/ CAN Module*, Microchip Technologies, Inc., 2004; www.microchip.com

[18] *Strain Gauge Measurement - A Tutorial*, Application Note 078, National Instruments, December 1995; www.ni.com

[19] *Coreless DC Motors – Data Sheet, Series 2232SR*, Faulhaber Group, Schönaich, Germany, www.faulhber.com

[20] *LPC2109/2119/2129 Product Data Sheet*, NXP Semiconductor, 2007; www.nxp.com

[21] *UM10204: I^2C-bus specification and user manual*, NXP Semiconductor, 2007; www.nxp.com

[22] *GP2D120 General Purpose Distance Measuring Sensor*, Sharp Microelectronics, 2000; www.sharpsma.com

Appendix A – Mechanical Components

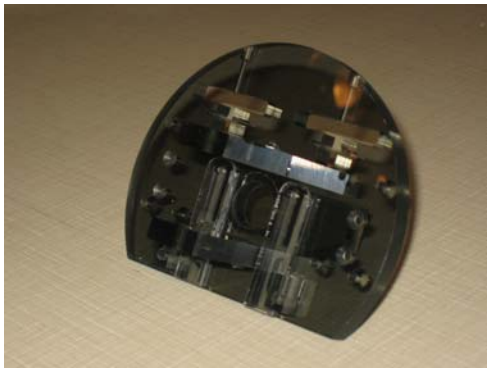


Figure 42: Motor Frame

Figure 43: Foot Assembly

Figure 44: Guide Block Assembly

Figure 45: Motor Mount

Figure 46: Pinion

Figure 47: Cam / U-Joint

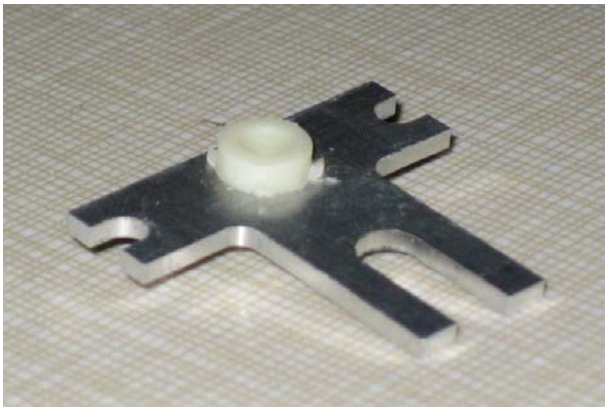


Figure 48: Lower Rocker

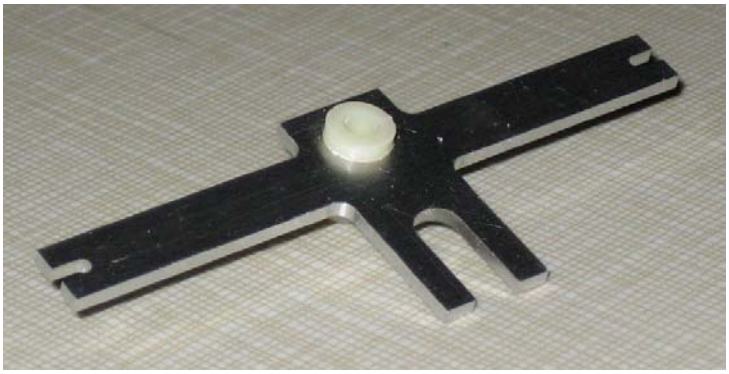


Figure 49: Upper Rocker

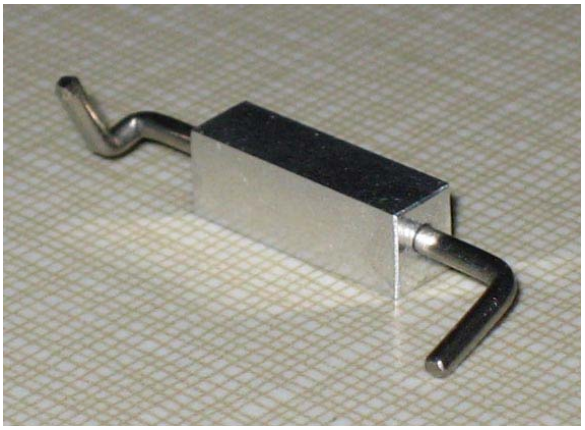


Figure 50: Transfer Arm Assembly

Figure 51: Coupling

Figure 52: Spool

Figure 53: Cable Retainer

Appendix B – Instruction Set

STOP	**Disable Motors**
type:	general call
description:	Motor drivers are disabled, but all control, acquisition, and communication processes remain active.
identifier:	00001 1 00000
data bytes:	0
data:	N/A
notes:	

RESET	**Place node into reset**
type:	general call
description:	Executes a software reset. Same effect as issuing a herdware reset. Node does nothing and waits for START command
identifier:	00010 1 00000
data bytes:	2
data:	byte 0: 0x55 byte 1: 0xAA
notes:	In RESET mode, the node's CAN address may be changed. Once START command is issued, address change is locked out.

START	**Start motors**
type:	general call
description:	Enables motor drivers. The first time this command is issued after reset, it also causes entry into normal operation mode
identifier:	00011 1 00000
data bytes:	2
data:	byte 0: 0x55 byte 1: 0xAA
notes:	This command causes the command motor position to equal the current motor position, preventing the robot from "jumping" to a prior setpoint upon motor re-enable.

REPORT_FREQ	**Set new data report frequency**
type:	general call
description:	Changes the frequency at which the node broadcases its current motor position and motor current draw values
identifier:	00100 1 00000
data bytes:	1
data:	byte 0: report frequency (in hertz)
notes:	The report frequency is sent as an unsigned integer. Data reporting can be disable by setting this value to '0'.

PID_FREQ	**Set PID update frequency**
type:	general call
description:	Changes PID loop execution frequency
identifier:	00101 1 00000
data bytes:	1
data:	byte 0: update frequency (in hertz)
notes:	The report frequency is sent as an unsigned integer.

CAN_ADD	**Change node's CAN address**
type:	general call
description:	Modifies the CAN address of a node
identifier:	00110 1 00000
data bytes:	3
data:	byte 0: 0x55 byte 1: 0xAA byte 2: address [1 to 31]
notes:	This command can only be issued immdiatly after node reset. Once the START command is issued, this feature is locked out. The address must be in the range of 1 to 31.

SETPOINT	**Set new motor command position**
type:	addressed
description:	Motor drivers are disabled, but all control, acquisition, and communication processes remain active.
identifier:	00001 1 aaaaa
data bytes:	0
data:	N/A
notes:	

LED_ON	**Turns LED on**
type:	addressed
description:	Each drive board contains two uncommitted LED that can be used for diagnostics. This command is used to turn these LEDs on.
identifier:	0001s 1 aaaaa
data bytes:	0
data:	N/A
notes:	With 's'=0, the Green LED is turned on With 's'=1, the Red LED is turned on

LED_OFF	**Turns LED off**
type:	addressed
description:	Each drive board contains two uncommitted LED that can be used for diagnostics. This command is used to turn these LEDs off.
identifier:	0010s 1 aaaaa
data bytes:	0
data:	N/A
notes:	With 's'=0, the Green LED is turned off With 's'=1, the Red LED is turned off

P_UPDATE	**Set new proportional gain**
type:	general call
description:	changes proportional gain of motor 's'.
identifier:	0011s 1 aaaaa
data bytes:	4
data:	32 bit float (IEEE-754 format)
notes:	At power-up, this value is retrieved from non-volatile memory. The contents of this memory can be modified with the STORE operation.

I_UPDATE	**Set new integral gain**
type:	general call
description:	changes integrall gain of motor 's'.
identifier:	0100s 1 aaaaa
data bytes:	4
data:	32 bit float (IEEE-754 format)
notes:	At power-up, this value is retrieved from non-volatile memory. The contents of this memory can be modified with the STORE operation.

D_UPDATE	**Set new derivative gain**
type:	general call
description:	changes derivativel gain of motor 's'.
identifier:	0101s 1 aaaaa
data bytes:	4
data:	32 bit float (IEEE-754 format)
notes:	At power-up, this value is retrieved from non-volatile memory. The contents of this memory can be modified with the STORE operation.

STORE	**Store current parameters ot ROM**
type:	general call
description:	copy the P, I, and D constants for motors 0 and 1 from RAM to ROM.
identifier:	0111s 1 aaaaa
data bytes:	2
data:	byte 0: 0x55 byte 1: 0xAA
notes:	At power-up, the P, I, and D gain coefficients are loaded from ROM to the motor control routine. This command provides a means of modifying these values so they are restored on power-up.