Modular Snake Robot: Progress Report 4

Brad Oraw & Jeremy Tinder

Computer Science and Computer Engineering
Pacific Lutheran University

Senior Capstone for B.S. in CSCE
Submitted to: Tosh Kakar

March 11, 2004
## Contents

- **Current Status**   1
- **Work Completed**   1
- **1 Construction**   1
- **2 Servo Controller Protocol: SCP**   2
- **3 Simulation**   2
- **4 Brain**   3

### Current Work

- **Future Work**   4
- **Appendices**   5
- **Time Line**   6
Current Status

The snake is moving. The Brain executes the follow-the-leader control scheme. The SCP was simplified making the communication between the Brain and the SCU’s more efficient. Accordingly, the snake’s motion is adequately smooth. Braces were added to several body segments to prevent the snake from rolling over onto its back. With this improvement the snake is sufficiently stable.

Work Completed

1 Construction

More improvements were made to the physical design of the snake. The most noticeable change is the addition of the body supports shown in Figure 1. As we were testing our movement algorithms, it was apparent that more support was needed on the snake to keep it upright. Every time the snake was started, it would initialize all the servos to their middle positions which made the overall shape of the snake a straight line. This required one to hold the snake upright until the snake began to move. Even during movement, the weight of the snake caused segments near the head and tail to fall over. The snake also managed to flip itself over a couple of times due to the instability. After adding the supports, the snake is much more stable and easier to work with. We are no longer required to keep the snake from falling over and it is now impossible for it to flip over onto its backside.

Figure 1: New body supports.

The brain was also upgraded to 20mhz this quarter. We were initially using a 4mhz crystal to clock the brain. We had trouble getting the pic to clock at 20mhz and figured out that the HS_OSC bit needs to be set in the configuration word in order to clock the pic higher than 4 mhz.

It was also discovered that the wiring of the MCLR reset switch was incorrect. I soldered the 10k resistor on the wrong side of the switch which made MCLR always high. We rectified
this problem by putting the resistor on the other side of the push button switch. Now by
pushing the switch, MCLR is set to ground effectively resetting the pic.

Another major accomplishment was the solution to the twitching and spasming of the
snake. It was initially thought that all the SCU’s were drawing too much power from the
brain when they were reading the SCU_TDX bus causing the voltage on the bus to drop and
the data to be misinterpreted. As a solution, we installed a voltage buffer on the output of
the brain. The voltage buffer is designed to keep the source voltage and the load separated.
This eliminates the effects of loading on the brain and prevents data from being corrupted.
Both the SCU_INT and the SCU_TDX buses were buffered using a 358 dual opamp.

We then connected the SCU_TDX output pin of the brain to a PC and visually analyzed
the positions what the brain was outputting to the SCU’s. It was then discovered that the
brain was persistently resetting. We added a capacitor to the power supply to filter the
incoming voltage and solved our problem. As a precaution, we also separated the five volt
sources of the servos and the pics. It was unsure if the servos would be causing current
spikes instigating a voltage drop. As a result of adding the capacitor and separating the
power source of the pics and servos, the motion of the snake has become very smooth and
fluid.

The final adjustments that were made to the snake were made to the wheels and the
servo arms. Not all of the wheels on the snake touched the ground as it moved along. Since
these wheels could not push off laterally, the motion of the snake was inefficient and wasteful.
To cure this problem, some of the wheels, and servo arms were shortened or lengthened. Not
all of the wheels fit perfectly on the ground but there differences are now tolerable.

2 Servo Controller Protocol: SCP

The Servo Control Protocol was changed again. We simplified it even more by sending all
twenty-four bytes of positions in one SEROUT command. The SCU’s receive all twenty-four
positions but only use the positions that pertain to them. Using the controller id of each
SCU, the range of positions to capture is calculated. The baud rate of the protocol was also
slowed down. We were previously transmitting at 9600 baud but have slowed to 2400 baud
in order to reduce the possibility of error. As a result of these changes, the communication
between the brain and the SCU’s is very smooth.

3 Simulation

Since we were having so many problems getting the snake to move properly more simulations
were conducted. Parts of the trace subroutine were tested in matlab to ensure that it was
properly outputting position commands. The results of the simulations concluded that
our algorithms and implementation details used in trace were correct. It was concluded
that we were overlooking a minor implementation detail with the picbasic language or the
architecture the pic. Brad also updated his Java simulator to allow us to tweak the all the
input values of the serpenoid function. This helped us to find values to implement into our
code. It saved us a tremendous amount of time in the long run since we didn’t have to guess
and check our values. A snapshot of the applet is shown in Figure 2 which is located in the Appendix.

4 Brain

Many improvements were made to the Brain. Several crucial bugs were fixed. In particular, we were handling $\alpha$ incorrectly. Once again we had trouble with PicBasic’s use of 16bit arithmetic. We made a mistake when we saved this result to 8bits. This operation takes the low order 8bits of the 16bit result and saves that to the 8bit variable. We assumed that if the 16bit result was greater than 255, the 8bit variable would always be set to 255. But, this is not the case. Therefore, we were experiencing errors when multiplying $\alpha$ in our motion equation. Once we realized what was happening, we quickly corrected the problem. The motion equation then performed as expected.

We also added a debugging port to one of the Brain’s I/O pins. On this port we are able to send serial data. We attached this output to a MAX232 voltage converter. A computer then receives the output on its serial port. We can record the data on the serial port and compare these to the expected results. This debugging feature allows us to confirm the values of internal variables in the Brain so that we can correct errors like our mishandling of 16bit arithmetic. The feature expedites troubleshooting.

Currently, the Brain is using the follow-the-leader control scheme. We focused on this scheme rather than the serpenoid scheme, because it is less complex. Follow-the-leader uses the sepenoid function without the parameter $\beta$; thus, we don’t have to tune this parameter. After correcting our mishandling of $\alpha$, the follow-the-leader scheme is working well. This algorithm is successfully moving the snake forward.

We also added directional control. $\gamma$ in the serpenoid function controls the direction of the motion. By changing this parameter, we are able to move the snake forward in different directions. We discovered that the snake has a mechanical imperfection that produces a slight tendency for the snake to turn right. We compensated for this by setting $\gamma$ to bias the output to the left. This worked very well, and the snake now moves straight.

We made a major change to our calculation of sine. We implemented our own version of a sine function since we had some trouble with PicBasic’s implementation of sine. We use a wave lookup table to generate the result of sine. The domain is from 0 to 255. The result is in the range of 1 to 255 where 128 is the median. Our implementation works well and eliminates the conversion from two’s compliment that PicBasic’s implementation used. This made our implementation much easier to work with.

Additionally, we implemented $\text{sine}^n$ where $n$ is a positive integer exponent. We discovered that a pure sine function might not produce the desired movement of the servos. For a smooth undulatory motion, the servos should spend a significant amount of time in their middle positions. This is necessary, since we want the snake to roll on its wheels. The wheels roll well when the corresponding joints angles are straight. Otherwise, if the joints are turning the wheels are used to change the direction of the motion. Thus, they are rolling forward less. If we control the angles with a pure sine function, the angles will spend most of the time in the max our min position and very little time in the middle position. Alternatively,
if a function like \( \sin^2 \) is used, the angles with spend equal amounts of time in the middle positions as in the min or max positions.

Our implementation of \( \sin^n \) uses our wave lookup table for sine to build a new table. When the Brain starts up it uses the specified value of \( n \) to build a wave table for \( \sin^n \). The motion algorithm then uses this table to generate the result of \( \sin^n \). Also, we can rebuild the \( \sin^n \) table during runtime by simply changing the value of \( n \) and recalling the table initialization subroutine.

With the successful implementation of the follow-the-leader scheme, we are confident that the serpenoid scheme will also work. We discovered that the follow-the-leader scheme is just a special case of the serpenoid scheme, since we are using the serpenoid function to control the head in follow-the-leader. In follow-the-leader the parameters \( \omega \) and \( \beta \) are combined into a single parameter. We lose independent control of \( \beta \), which specifies the phase offset between adjacent joint angles. In follow-the-leader this offset is fixed, because the motion is just shifted between the joints. Follow-the-leader use a serpenoid function where \( \omega = \beta \). Since the follow-the-leader scheme works, the serpenoid scheme should also work.

**Current Work**

Currently, we are designing the Peripheral Bus Protocol and forming its specifications. We are investigating implementation options for both the Brain and peripheral devices. Additionally, we are still testing the motion control. We are tuning parameters and evaluating speed and direction controls.

**Future Work**

After we design the PBP, we will add PBP functionality to the Brain. We will build PBP supporting devices that might be necessary to regulate peripheral communication. Then, we will construct our first Peripheral Module. We have not yet determined the function of this first Module. Most likely, it will provide collision detection and avoidance.
Appendix

Figure 2: Java simulator for Serpenoid Function.
## Time Line

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
<th>Duration</th>
<th>% Complete</th>
<th>Jan 2014</th>
<th>Feb 2014</th>
<th>Mar 2014</th>
<th>Apr 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prototype 1a</td>
<td>1/16/2014</td>
<td>3/12/2014</td>
<td>32d</td>
<td>95%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Implementation</td>
<td>1/16/2014</td>
<td>2/21/2014</td>
<td>26d</td>
<td>95%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Construction</td>
<td>1/16/2014</td>
<td>2/13/2014</td>
<td>21d</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Electrical</td>
<td>1/16/2014</td>
<td>2/21/2014</td>
<td>26d</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Testing</td>
<td>1/16/2014</td>
<td>2/13/2014</td>
<td>21d</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Analysis</td>
<td>2/20/2014</td>
<td>3/1/2014</td>
<td>4d</td>
<td>89%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>General Faults</td>
<td>2/20/2014</td>
<td>3/1/2014</td>
<td>4d</td>
<td>55%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Improvements</td>
<td>2/20/2014</td>
<td>3/1/2014</td>
<td>4d</td>
<td>55%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Prototype 1b</td>
<td>3/1/2014</td>
<td>4/30/2014</td>
<td>45d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Design</td>
<td>2/20/2014</td>
<td>3/1/2014</td>
<td>25d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Mechanical</td>
<td>2/20/2014</td>
<td>3/1/2014</td>
<td>14d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Electronics</td>
<td>2/20/2014</td>
<td>3/1/2014</td>
<td>14d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Intelligence</td>
<td>2/20/2014</td>
<td>3/1/2014</td>
<td>14d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Implementation</td>
<td>3/1/2014</td>
<td>3/28/2014</td>
<td>14d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Construction</td>
<td>3/1/2014</td>
<td>3/28/2014</td>
<td>14d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Testing</td>
<td>3/22/2014</td>
<td>4/1/2014</td>
<td>9d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Analysis</td>
<td>4/1/2014</td>
<td>4/30/2014</td>
<td>14d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>General Faults</td>
<td>4/1/2014</td>
<td>4/22/2014</td>
<td>14d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Improvements</td>
<td>4/1/2014</td>
<td>4/22/2014</td>
<td>14d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Final Analysis</td>
<td>4/22/2014</td>
<td>4/28/2014</td>
<td>6d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Project Overview</td>
<td>4/22/2014</td>
<td>4/27/2014</td>
<td>6d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Future Work</td>
<td>4/22/2014</td>
<td>4/27/2014</td>
<td>6d</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>