

# Design Exploration of a Human-machine Interface Application

**Francis Li, Sam Madden**

University of California, Berkeley  
Berkeley, California 94720-1776  
{ fli, madden }@cs.Berkeley.edu

## ABSTRACT

A data glove is a human-machine interface designed to capture the motion of a computer operator's hand for real-time input. Current implementations of such devices are commonly bulky, wired, and have high power requirements. It has been proposed that a data glove could be implemented with low power, integrated sensing and processing components with wireless communications. In this paper, we evaluate an existing lower-power data-glove implementation. Then, with minimizing power requirements as a goal, we evaluate design alternatives that tradeoff application specific local processing with communication. Finally, we compare the performance of those software alternatives on several general purpose microcontrollers and DSPs.

## Keywords

Smart dust, MEMS, input devices, HMI, exploration

## INTRODUCTION

There is a significant commercial and research trend to move computing applications "beyond the desktop." Both existing and new applications are being developed that take advantage of a rapidly expanding computing infrastructure where processing occurs in embedded systems in the local environment or on wide-area systems accessed over a network. However, these applications require new forms of human-computer interaction. For example, email communication may occur on a handheld wireless device. In this case, it is clear that existing human-machine interface devices such as mice and keyboards are not suitable.

A data glove is a human-machine interface designed to capture the motion of a computer operator's hand for real-time input. It is commonly used for interacting with virtual environments, but may also be used as a general purpose pointing device. Current implementations of such devices, such as the 5DT Data Glove 16 seen in Figure 1, are often bulky, wired, and have high power requirements [1]. The U.C. Berkeley SmartDust research group has proposed an alternative design that uses integrated sensing and processing components with wireless communication capability [5]. Such components, or "motes", when scaled down to a  $1 \text{ mm}^3$  package, would draw only  $10 \mu\text{A}$  of current. At that size, the motes could be placed on the tips

of fingers and on the back of the hand for a much less obtrusive data glove interface.

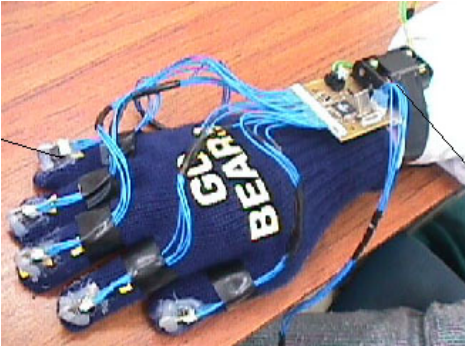
In this paper, we first evaluate the design and power requirements of the proof-of-concept prototype glove developed by the SmartDust group, called the Acceleration Sensing Glove (ASG). Then, based on the results of the analysis, we define an exploration methodology for evaluating alternative implementations of the ASG. In particular, since SmartDust motes have processors integrated with the sensors, we investigate the tradeoff between local computation and communication. We perform this evaluation in the context of three different processor architectures: the Atmel AVR 8-bit RISC microcontroller [2], the Texas Instruments MSP430 16-bit microcontroller [9], and the Texas Instruments TMS320C5000 DSP platform [12]. Finally, we discuss the results of the analysis, some directions for future work, and conclusions.

## EXISTING PROTOTYPE APPLICATION ANALYSIS

The existing ASG is shown in Figure 2. It consists of a single Atmel AVR 8535 microcontroller wired to six Analog Devices XL202 accelerometers: one per finger and one on the back of the hand. The Atmel continuously samples the accelerometers and relays their values via a RFM 916.5 MHz wireless radio connection to a host PC. The PC computes an average and polar transform on the data, then does gesture recognition via template matching and reports the gesture to the user or causes the mouse pointer to move. Thus, there were two important analyses required for the existing prototype: understanding the application and the computation involved, and



**Figure 1.** 5DT Data Glove 16-W (150mA, 9V)



**Figure 2.** Acceleration Sensing Glove (ASG) Prototype

characterizing the communication costs of this specific hardware implementation.

### Computation

The computation can be expressed via the following pseudocode:

```

Do N times per second:
  For each of k accelerometers:
    (x, y) ← 10 bit accelerometer readout
    Xavg ← Average(x, Xavg)
    Yavg ← Average(y, Yavg)
    (r, Θ, ...) ← CalcPolar(Xavg, Yavg)
  Periodically
  Compute bestG from set of gesture
  templates G where
    bestG ∈ G s.t.
      FindGestureError(r, Θ, ..., bestG) ≤
      FindGestureError(r, Θ, ..., g) ∀ g ∈ G

```

$N=20$  and  $k=6$  for the analyses in this paper. "Periodically" means at random intervals; we assume those intervals are no smaller than 1 second. The Average function is a rolling average with a window size of 48 samples. The  $(r, \Theta, \dots)$  tuple is a set of aggregated statistics, including polar coordinates, average angular velocity, and angular acceleration; these statistics are computed via floating point math and inverse trigonometric functions. The FindGestureError method uses Euclidian distance from stored  $(r, \Theta, \dots)$  values for each of the gesture templates as a scoring function.

### Communication

Communication cost in the above algorithm is dominated by relaying the  $(x,y)$  pair to the host PC via the radio. The number of bits is as follows, assuming  $x$  and  $y$  are both extended to 16 bits:

$$n = N \text{ samples/sec} \cdot \text{accelerometer} \cdot k \text{ accelerometers} \cdot 2 \cdot 16 \text{ bits/sample} = 3840 \text{ bits/sec}$$

The radio is 116 kbits. The total transmit percentage is thus:

$$n/116,000 = 3.31\%$$

The total transmission cost is then:

$$W = .0331 \cdot (\text{active radio cost}) + .9669 \cdot (\text{idle radio cost})$$

According to the radio data sheet [7] the radio draws 36mW in active mode, and 15uW in idle mode.  $W$  is thus at least 1.2 mW. This cost, however, does not include processor overhead or cost of analog support hardware on the radio; experimental results from [3] indicate an actual cost of 4317 nJ/bit, or a total cost of 16.5 mW to drive the radio.

### EXPLORATION METHODOLOGY

The results of the prototype analysis provide a basis for iterating upon the design of the ASG. Given the communication cost analysis above, we evaluate the possibilities for alternative methods of computation in terms of both the application architecture and hardware platform.

#### Application

Looking at the application design, there is a significant amount of energy spent communicating information from the glove to the host. By moving some of the computation from the host to the glove, we can reduce the amount of communication.

In this *centralized* design, we assume that the glove processor performs two Average and one CalcPolar calculations for each accelerometer at the sampling rate of 20 hz. Then, at a rate of one gesture per second, the processor returns the currently recognized gesture by calculating the FindGestureError value for each of ten gestures. The resulting communication to the host is then just one gesture value (8 bits) per second.

#### Hardware

From a hardware perspective, the design of the prototype ASG does not accurately represent the SmartDust vision in which each accelerometer is packaged with its own processor. If we assume such an implementation and distribute the computation across multiple processors, we may be able to reduce communication costs further.

In this *distributed* design, we assume that each of the processors performs the two Average and one CalcPolar calculations for its accelerometer input at the sampling rate of 20 hz. The resulting communication to the host is then one average coordinate and polar coordinate per second per processor, where the host performs the gesture recognition.

#### Processor Architecture

As a further exploration from a hardware perspective, we evaluate the centralized and distributed glove designs using three different processor architectures:

- Atmel AVR AT90LS8535 8-bit RISC microcontroller
- TI MSP430C112 16-bit RISC microcontroller
- TI TMS320UC5402 16-bit fixed-point DSP
  - TI TMS320C55x 32-bit fixed-point DSP (based upon preview data)

For each processor, we compile and simulate the execution of the Average, CalcPolar, and FindGestureError functions using existing software development tools to obtain cycle

cost estimates for those functions. Table 1 summarizes the simulation results for each of the processors.

Then, for both the centralized and distributed cases, we calculate the total computation cost in cycles as described previously. We choose the appropriate clock frequency and supply voltage for the processor to perform the total computation to calculate the power consumption. For the distributed case we then multiply this value by 6 for an estimate of the power consumption of the entire glove. Tables 2 and 3 summarize the results of these analyses.

#### ATMEL

The Atmel AVR AT90LS8535 microcontroller is a widely used, very easy to program low power microprocessor [2]. It is currently used in the SmartDust group as the CPU for the "macro mote". It can be driven down to 2.7V at speeds up to 11 Mhz, with an active power consumption of 10.8 mW at 4 Mhz. (2.7 mW/MIP). It has a sleep mode which draws about 10  $\mu$ A with a wakeup time of 1ms. It executes one instruction per cycle, except for memory instructions which require two cycles.

The AVR GCC tools for Linux were used to compile code for the Atmel processor. The open source simulator avrsim was used to generate instruction counts for the application, and those instruction counts were converted to cycles by accounting for the frequency of memory operations.

#### TI

The Texas Instruments MSP430P112 is an ultra-low power 16-bit RISC microcontroller [9]. In contrast to the Atmel processor, the TI draws only 330  $\mu$ A/MIP with a minimum supply voltage of 2.5 V and has a 1.5  $\mu$ A standby mode (with a 6 ns wakeup latency).

The analysis was performed using the IAR Systems Embedded Workbench and C-SPY development tools [4]. Program source was compiled using the included MSP430 C compiler and math floating-point package with medium speed optimization. The C-SPY debugger with the MSP430 simulator driver reported estimated cycle counts for each of the functions.

#### TI DSP

The Texas Instruments TMS320UC5402 16-bit fixed-point DSP is a low power member of the C54x DSP family, with a 1.8 V core supply voltage (vs. 3.3 V for most other actively available models) [10]. At this supply voltage, the processor draws 2/3 of the usual current of about 1 mA/MIP. The processor has three idle modes, each drawing less current than the previous [12]. In IDLE1, the CPU is disabled, leaving the timer and other peripherals running. Since we utilize the timer for performing regular sampling of the accelerometer, we use this IDLE configuration.

The analysis was performed using the TI Code Composer Studio tools, including their C compiler and simulation drivers. Table 1 shows the highest cycle count estimates

**Table 1. Individual Computation Costs**

	Atmel	TI	54x	55x	
Average	2556	2309	4012	717	cycles
CalcPolar	21870	24225	71566	17758	cycles
FindGestureError	37758	27589	79931	16737	cycles

**Table 2. Centralized Computation Cost**

Total Cycles = (2 • Average + CalcPolar) • 20hz • 6 sensors + FindGestureError • 10 templates

	Atmel	TI	54x	55x	
Total Cycles	3.62e+6	3.74e+6	1.04e+7	2.47e+6	cycles
Clock	4.00	4.00	11.00	2.00	MHz
Supply	2.70	3.30	1.80	0.90	V
Current (active)	4.00	1.32	7.37	0.10	mA
Current (idle)	1.80	0.00	1.32		mA
<i>Total Power</i>	10.23	4.07	12.62	0.09	mW

**Table 3. Distributed Computation Cost Per Processor**

Total Cycles = (2 • Average + CalcPolar) • 20hz

	Atmel	TI	54x	55x	
Total Cycles	5.40e+5	5.77e+5	1.59e+6	3.84e+5	cycles
Clock	1.00	1.00	2.00	1.00	MHz
Supply	2.70	2.50	1.80	0.90	V
Current (active)	2.00	0.33	1.34	0.05	mA
Current (idle)	0.50	0.00	0.24		mA
<i>Power</i>	3.54	0.48	2.01	0.05	mW
<i>Total Power*</i>	21.21	2.86	12.05	0.27	mW

\* for six processors

for this processor, dominated entirely by the floating-point arithmetic package. A cursory review of the code for a few floating-point routines did not reveal the source of the high cost. Since each instruction can take anywhere from 1 to 6 cycles depending upon the location of the operands, a much finer grained analysis would be required for further investigation.

The C55x is TI's next generation low power DSP in the same C5000 family [11]. By configuring Code Composer Studio with the C55x compiler and target simulator, we were able to obtain the cycle count estimates shown in Table 1. They are significantly lower than those of the C54x. We attribute this greater efficiency to their next generation compiler as well as a floating point package that is likely optimized to take advantage of the increased parallelism of the C55x (including a second parallel 16-bit ALU, a second parallel MAC unit, two additional accumulators, and more independent buses). There is no detailed power analysis for the C55x family available. All power calculations are based on the highly advertised 0.05 mA/MIP at 0.9 V ratings.

#### DISCUSSION

Figure 3 invites some interesting comparisons between the proposed architectures. First, power consumption is 40% less in the centralized Atmel architecture over the Host PC.

This is due solely to reduced communication costs: even though the processor is much more active in the centralized scenario, driving the radio is so expensive that moving the processor from near 0 utilization to near 100% utilization is still cheaper than transmitting 3 kbits/second.

A second observation is that the TI processor is a factor of two less power-hungry than the Atmel, even though the instruction counts between the two processors are similar. This is due primarily to the extremely power-efficient nature of the TI.

Finally, the DSP architectures use many fewer instructions than either the TI or Atmel. The 54x is a less power-efficient processor, and thus requires more power than the microprocessors. But the newer 55x, which is built to be extremely low power, far outperforms any of the other architectures. Low-power DSPs are very well suited to the ASG task because their MAC units are optimized for the repetitive mathematical operations being performed.

### Distributed vs. Centralized

Looking at Figure 3, it's immediately apparent that the distributed case is always worse than the centralized case. However, in the case of the TI and the 55x, this is due to the increased communication cost in the distributed case. Surprisingly, when comparing just computation costs, voltage scaling effects from running the processors at a lower frequency make six processors more power-efficient than just one. This effect is not seen in the Atmel because it is running at the minimum rated voltage of 2.7V in all cases. Similarly, voltage-to-clock-rate tables aren't available for the unreleased 55x DSP.

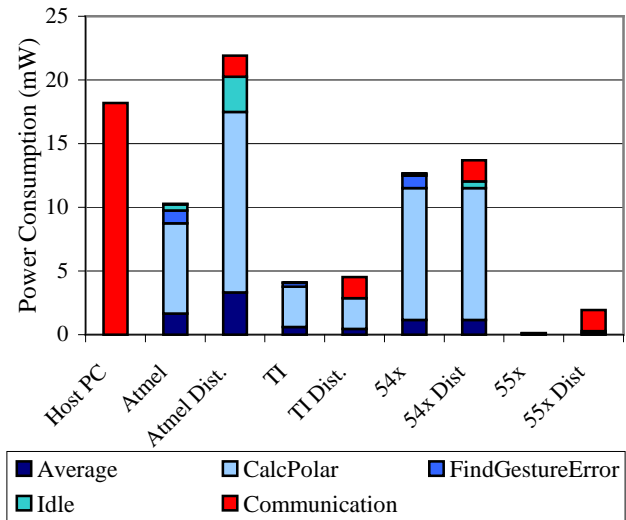
### Computation Breakdown

The total computation is completely dominated by the CalcPolar routine. Some further analysis reveals that about 50% of the computation is spent in two calls: `acos` and `atan`, which are used to do cartesian to polar coordinate conversion. Another 20% is spent in calls to `fsqrt`, and much of the remainder is consumed by floating point multiplies and adds. This suggests an obvious optimization: switch to fixed point arithmetic and lookup table based interpolation for the transcendental functions. This was not done as a part of this analysis since our goal was to compare the relative performance of various architectures and implementations on the same code base, not to optimize the code base for a particular architecture.

### FUTURE WORK

As in any optimization process, there were a very large number of options for exploration over the course of this project. We chose to deal with only a few of them here; there are a number of other areas where significant performance gains could certainly be realized:

*Dedicated Hardware:* By building ASICs or using a reconfigurable processor, it would have been possible to reduce the computation costs of the CalcPolar and Average functions significantly.



**Figure 3.** Summary and Decomposition of Power Costs

*Fixed Point/Lookup Tables:* As mentioned previously, the cost of computation is largely dominated by floating point math and trigonometric computations. Via careful analysis of the typical data sets, lookup tables for trigonometry and fixed point math would probably yield similar precision in an order of magnitude fewer instructions.

*Alternative Radios:* The RFM is an older generation radio which is clearly inferior to current technologies such as Bluetooth. Estimates indicate a cost per bit of communication in Bluetooth at 100nJ/bit [Rab], which is a factor of 40 better than the RFM. Also, it might have been possible to switch to very low range, very low power radios for the distributed architecture, which might have made it a feasible alternative

*Other hardware:* We didn't explore the power consumption of the accelerometers or analog devices around the radio. In the original ASG implementation, this accounted for 22 of the 45mW of power, so in a real world implementation where the goal is to minimize total energy use, this would be a necessary step.

### CONCLUSIONS

By following the design methodology outlined above it was possible to optimize the implementation of the ASG prototype in a systematic and informative way. We saw that communication costs can be driven down by performing local computation of aggregate statistics. We learned that several distributed processors can perform the same computation at a lower power cost as a result of voltage scaling effects, but that communication in those distributed environments may be too high to make them feasible. We saw that DSPs use less instructions to perform the same work in math-intensive applications like ours. Finally, we saw that even when two processors are similar in instructions and features (TI vs. Atmel or 54x vs 55x) one may be significantly more power efficient; fully exploring the space of available architectures is a worthwhile endeavor. To illustrate the potential benefit of these

observations, compare the original power consumption of the Atmel and RFM radio at 27mW to the centralized 55x scenario at .1mW. A 270x power reduction simply by reformulating the application for the right processor architecture *without* optimizing assembly language or designing ASICs is remarkable.

#### ACKNOWLEDGEMENTS

We would like to thank Prof. Kris Pister for giving us a lot of initial ideas and brainstorming with us and Brett Warneke, John Perng, and Seth Hollar in the SmartDust group for speaking with us in the early stages and sending us their code for the ASG. Finally, we'd like to thank Robert Szewczyk and Andras Ferencz for their detailed analysis of the communication costs of the RFM radio.

#### REFERENCES

1. 5DT <Fifth Dimension Technologies>, <http://www.5dt.com/>
2. Atmel Corporation. Atmel AVR Datasheet. <http://www.atmel.com/atmel/acrobat/doc1041.pdf>
3. Hill, Szewczyk, Woo, et al. System Architecture Directions for Network Sensors. In Review. April, 2000.
4. IAR Systems, <http://www.iar.com/>
5. Perng, J.K., Fisher, B., Hollar, S., and Pister, K.S.J. Acceleration Sensing Glove. In *Proceedings of the 3<sup>rd</sup> International Symposium on Wearable Computers*, 1999.
6. Rabaey, Jan. Embedded System Design for Wireless Applications. *UC Berkeley CS252 Lecture 24*. April, 2000.
7. RFM Corporation. RFM TR1000 Datasheet. <http://www.rfm.com/products/data/tr1000.pdf>
8. Texas Instruments Incorporated, <http://www.ti.com/>
9. Texas Instruments Incorporated. MSP430x11x Mixed Signal Microprocessors. Literature number SLAS196B. December 1998, revised April 2000.
10. Texas Instruments Incorporated. TMS320UC5402 Fixed-Point Digital Signal Processor. Literature number SPRS096A. April 1999, revised December 1999.
11. Texas Instruments Incorporated. TMS320C55x Technical Overview. Literature Number SPRU393. February 2000.
12. Turner, Clay. Calculation of TMS320LC54x Power Dissipation. Literature number SPRA164. Texas Instruments Incorporated, June 1997.