VersaPatch: A Low Cost 2.5D Capacitive Touch Sensor

Ray Bittner and Mike Sinclair

Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{raybit,sinclair}@microsoft.com

Abstract. Capacitive input devices are becoming increasingly prevalent in consumer devices. This paper presents the hardware and algorithms for the low cost implementation of a capacitive 2.5D input device. The low cost and low power consumption of the device make it suitable for use in portable devices such as cellular phones. The electrical properties used are such that the pre-existing snap dome technologies in such devices can be used as capacitive sensing elements, further reducing the cost and size impact of the capacitive sensor.

Keywords: 2D Pointing, Capacitive Sensing, Interface, Low Cost, Low Power.

1 Introduction

Capacitive sensing is becoming a hot topic in embedded devices; currently popularized by Apple's iPod and iPhone. In this paper, we will describe a very cost effective technique for producing a two and a half dimensional capacitive touch sensor. The sensor will give absolute X-Y position, as well as some useful pressure information that can be used to emulate button presses and/or provide a simplistic Z coordinate.

The basic circuit used for capacitive sensing is illustrated in Figure 1. The capacitive sense pad is connected to a large pullup resistor and a GPIO pin on the microcontroller. The object is to measure the capacitance between a finger, or other pointing device, and the sense pad. As a finger approaches the sense pad, the capacitance at the sense pad increases. This change in capacitance is measured by using the GPIO to first ground the sense pad, and then tri-state it (change it to be an input). A timer

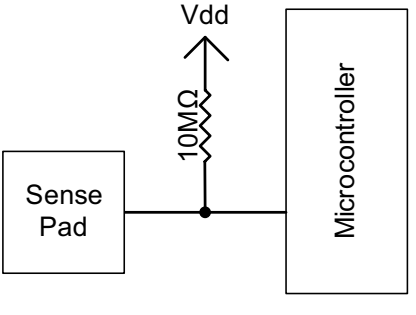

Fig. 1. Sensing Circuit Instance

J.A. Jacko (Ed.): Human-Computer Interaction, Part II, HCII 2009, LNCS 5611, pp. 407–416, 2009.

inside the microcontroller then measures the amount of time needed for the total capacitance attached to the GPIO to charge and register as a logical 1. The time elapsed is an indication of the combined capacitance of the sense pad, the GPIO pin and the rest of the circuit. The characteristic voltage curve of such a circuit follows the typical exponential curve for capacitive charging, resulting in a non-linear relationship between timer counts and actual time elapsed, but it is sufficient for our purposes. This is similar to what is discussed in [1]. The value of the pullup resistor allows the designer to optimize the circuit for more sensitivity (higher values) or more noise immunity (lower values). A value of 10MΩ is shown, but a wide range of values are possible depending on the application.

The system is exceedingly inexpensive because all that is required is a single microcontroller with multiple available GPIO pins, one resistor per sense pad, and the sense pads themselves. The sense pads can be fabricated in etched copper or other conductor directly on the circuit board for the cheapest possible implementation. The conductors used for the sense pads do not need to be very low resistance because the pullup resistor is of very high value. This suggests many possible materials and modalities for use of the basic principle.

We have used a Texas Instruments MSP430 as the microcontroller. These processors are useful because of the very low leakage current exhibited by their GPIO pins; less than 50nA as specified in the datasheet [1]. The low leakage current allows a large value pullup resistor to be used, which increases the effective resolution of the timer reading. Other processors could be used, but we have found the MSP430 to be exceptional in this regard. Further, this entire application was made to work in the Flash-based MSP430F2011, which currently has a list price of $0.80 in 100 unit quantities. It is generally known that this price can be much lower in higher quantities; particularly if a production ROM part is chosen. Since the microcontroller is at least 90% of the cost of the system, its cost can be used as an estimated implementation cost.

The average current needed is less than 40μA at 3.3V at with a 16 MHz clock, owing to the low power capacitive sensing method, and the low power performance of the MSP430. Lower power consumption is possible by lowering the operating voltage or sampling frequency, but the choice of operating voltage is often a cost issue in consumer devices, and 3.3V is a reasonable choice in today's technology.

2 Sense Pad Design

The PCB electrode layout is shown in Figure 2 where E1…E9 are the sensing electrodes. With this solution we want to use a low-cost microcontroller which usually means one with a low pin count. Of course, if dozens of pins (and electrodes) were available with which to construct the sensor, 2D finger sensing would be much easier since the electrode size could be made small with respect to the finger size and the sensing would not alias with the sample size of the electrode. In our implementation, the finger size (the size of the finger pad in contact with the sensing surface) is on the order of the electrode size so some form of anti-aliasing technique is required for subsample positions to be detected and properly processed.

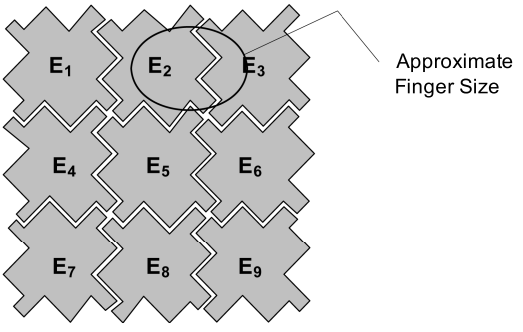

Fig. 2. Anti-Aliased Electrode Layout

In the imaging world, anti-aliasing is usually accomplished with a 2D anti-aliasing filter placed over the sensor which effectively blurs the boundary between pixels. We used a ragged pattern on the PCB to create anti-aliasing in our design. Figure 2 shows this pattern, which creates a 'blurring' or a smooth transition of capacitances between the electrodes.

The device is built by connecting each pad-resistor pair to a single microcontroller as shown in Figure 1. The microcontroller firmware cycles through the ground, tri-state, sense logic 1 cycle as described above. In the prototype device, the microcontroller's internal clock rate is 16MHz, and all 9 sense pads are connected to interrupt-capable GPIOs, allowing precise and repeatable timing within the firmware.

3 Mean Zero Value Calculation

The coordinate calculation algorithm requires a known zero reference capacitance for each sensor, which will be termed the mean zero value. This mean zero value is the summation of timing delays in the firmware, plus the inherent capacitance of the sensing pin and the PCB design. Calculation of the mean zero value must be done per pin, per design, and possibly per device instance since each design and sensing chip will be slightly different, and because these values change with device aging and environmental effects. To combat these problems, an adaptive algorithm was developed to calculate the mean zero value at run-time per pin in each device.

The problem is to find a value as close as possible to the true mean zero value in the presence of noise during normal use. Difficulties are encountered since the sensors may be covered by an object that registers as a high capacitance such as a pen or thigh, or someone may be attempting to use the device. What is needed then is an algorithm that can make successive approximations of the true mean zero value while classifying sensed capacitive events. We have developed an algorithm for this task called the Method of Split Averages.

Two running averages are kept for each capacitive sensor. One of these, the average high value, represents an average of all the readings that are believed to have been taken when a capacitive event is registered. The average low value is the average of all readings that are believed to have been taken when no capacitive stimulus was present. At initialization, the average low value may be set to zero, or to some value

that is considered to be reasonable for that device. The average high value is initialized to be some increment above the average low value, where the increment is greater than the expected noise levels for the device. The following steps are then performed per capacitive sample taken:

1. Compare the new sample to average high value and average low value.
2. If the new sample is closer to the average high value, average it into the average high value and set the "finger present" flag.
3. If the new sample is closer to the average low value, average it into the average low value and clear the "finger present" flag.

Over time, the average low value will approach the true mean zero value for that capacitive sensor. The capacitive values that are fed into the algorithm are already pre-averaged to some small extent, rather than using the raw readings. This helps the algorithm in the discrimination process in a noisy environment. The raw capacitive values are averaged using an exponential filter, using the shift and add method, so that the memory and processing requirements are manageable in a microcontroller environment. Typically, the initial averaging window is over two to four values at a sampling rate of 80 Hz.

The window length used for the algorithm's average high and average low values is longer. If this window too short, then the two averages will track noise spikes and finger events too closely. If the window is too long, the averages will not correct themselves quickly enough at initialization or in the event that a long series of abnormal readings occurred. We have found that a shift by 12, giving an averaging window of approximately 4096 values, works well.

In a production environment, the initialization values for the average high value and average low value may be standardized across all devices based on values that are found to be generally applicable to the system at hand. This will reduce the initial calibration time so that it should not be noticeable to the end user.

The "finger present" flag is used to trigger the coordinate calculation algorithm, and may also be used to implement capacitive button press detection. Finally, the difference between the average high value and average low value is the dynamic range of the capacitance reading, which is a concept that will be used later.

4 Memory Saving Technique

Microcontrollers typically have a very small amount of RAM and as presented the Method of Split Averages would require two separate running sums to be kept for each sensor pad. Each running sum is likely a 24-bit or 32-bit quantity; requiring significant space when multiplied by the number of sensor pads. The number of bytes required is equal to the number of pads multiplied by the width of the sums times two.

If the dynamic range of the pads may be assumed to be roughly similar, then significant memory may be recovered by maintaining only a single average dynamic range for all of the sensor pads and eliminating the average high value completely. In that case, a separate average low running average is kept for each sense pad, and then a single common average dynamic range is kept for all of them. This reduces the memory needed by nearly a factor of 2. In practice, the assumption of similar dynamic range turned out to be a good one for our applications and so this optimization was used.

5 Coordinate Calculation Algorithm

A 3x3 sensor array will be used as an example, but the method scales to larger sensor grids as well. Figure 3 shows a logical view of the capacitive array and not necessarily the exact dimensions or pad shapes used in a real design. The goal of the algorithm is to produce X-Y coordinates in the range of 0 to 1, so that these can be multiplied by a scaling factor to achieve whatever integer range is desired.

In order to generate the 0 to 1 fractional scale, each row and column of the array is assigned a weight. Intuitively, the columns would be assigned weights 0, 0.5 and 1, as would the respective rows, representing the desired range of output values. However, a weight of 0 is problematic because when used as a multiplicative factor it will zero out the contribution of that row or column in the final result. For that reason, a value of 1 is added to all weights, making the range 1 to 2, which is what is depicted in Figure 3. The added 1 can be subtracted back out later.

The calculation of the X-Y coordinates uses these weights in a weighted average to give far more resolution than the apparent 3x3 array. The X position is given by:

$$X = 1 * \frac{\sum_i C_{i,0}}{C_{Total}} + 1.5 * \frac{\sum_i C_{i,1}}{C_{Total}} + 2 * \frac{\sum_i C_{i,2}}{C_{Total}} - 1 \quad (\mathbf{1})$$

Where $C_{i,j}$ represents the adjusted capacitance value from the associated pad and C_{Total} is the sum of all the adjusted capacitances. As can be seen, the summation multiplies the weight of each column by the fractional contribution of that column to the total capacitance measured. The -1 on the right had side adjusts the output range back down to a 0 to 1 fractional scale. The resulting X value may be multiplied by any scaling factor, such as 256, to obtain a useful integer range. By carefully performing all summations and multiplications before any divisions, floating point math may be avoided. A power of 2 scaling factor further simplifies the amount of code needed. As an example, the formula may be rewritten as:

$$X = 256 * \frac{\sum_i C_{i,0} + 3 * \frac{\sum_i C_{i,1}}{2} + 2 * \sum_i C_{i,2}}{C_{Total}} - 256 \quad (\mathbf{2})$$

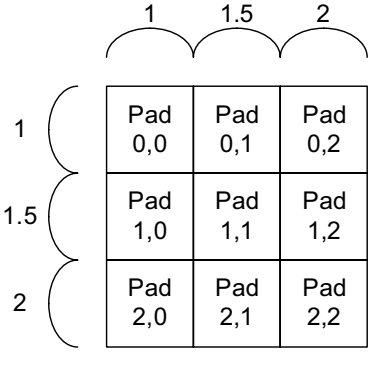

Fig. 3. Row/Column Weight Assignment

The corresponding unscaled equation for the Y coordinate is:

$$Y = 1 * \frac{\sum_j C_{0,j}}{C_{Total}} + 1.5 * \frac{\sum_j C_{1,j}}{C_{Total}} + 2 * \frac{\sum_j C_{2,j}}{C_{Total}} - 1 \quad (3)$$

Except in this case, the sums are computed across the rows. By using the proportional contribution from each pad, rather than absolutes, the weighted average avoids problems with moment to moment sensing differences such as humidity, temperature, the actual distance between the finger and the sense pads, etc.

Note that this same method may be used for other array dimensions by assigning more intermediate weights to the rows and columns of the array. For example, if a 4x4 grid is desired, then starting weights of 0, 0.33, 0.66 and 1 could be used.

A larger grid may be desired in the case where the finger size of the user is smaller than the size of a sense pad. In that case, the algorithm may not have enough capacitive contribution from adjacent pads to give sufficient resolution. In that case, larger array dimensions may be needed to make the size of the sense pad commensurate with, or smaller than, the size of the user's finger.

The algorithm expects the dynamic range of all pads to be roughly equal under a given set of conditions. This translates to making the sensing pads all have roughly the same surface area. Though, even if they are not, the disparity could be adjusted by individual pad weights.

6 Experimental Results

In order to show the effectiveness of this approach, we placed the device on a precision stepper table and used a brass finger as the pointing device. The stepper table was moved in 0.1" increments in both X and Y. At each step, the finger remained stationary briefly while sample data was collected at the rate of 80 samples per second. The finger was electrically connected it to the operator, but was not grounded through any other path. In practice, we have found that a real finger performs better than the brass one, but this provides a worst case scenario. The individual sensor pads measured 0.545" square from point to point, and the overall 3x3 pad measured 1.4" square from point to point.

The left hand graph of Figure 4 shows the results of skimming the finger across the top of the plastic keys on the keypad, which places the finger 0.060" off of the top of the sensors on the circuit board. This simulates normal use most closely, where the finger is in contact with the plastic buttons. A scaling factor of 256 was used, giving a nominal response of 0 to 255 along both axes. The averaging being applied on a sample by sample basis uses a shift by 2 in this case, resulting in a moving average window over approximately 4 capacitance values per sensor. The slower moving averages used to track the mean zero value for each pad used a shift by 12.

There is some noise in the system, which was characterized by a standard deviation of 2.16 in XY coordinate space. With a scaling factor of 256, this gives a true XY resolution of roughly 100x100. The system is quite useable as a pointing device in this state, as we have demonstrated in the lab by emulating a serial mouse and using it to drive a Microsoft Windows based PC. The pad is also very responsive with shift by 2 averaging, and if lower noise is desired the pad would still be very useable if a shift by 3 or 4 were employed. Also evident is the fact that the device was rotated slightly clockwise when the samples were taken.

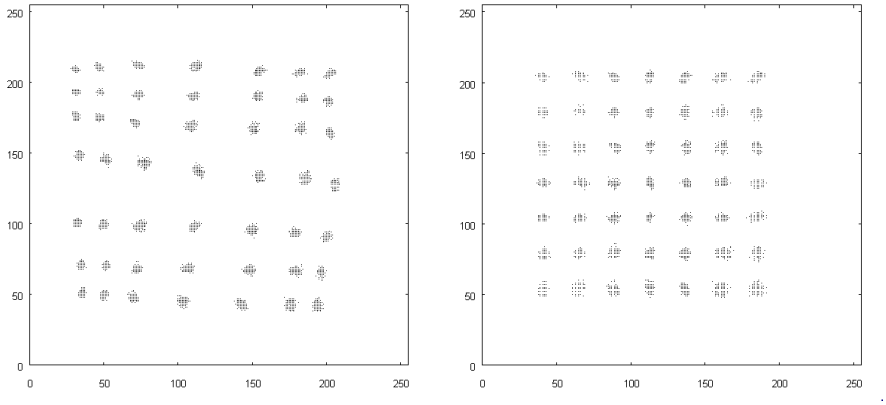

Fig. 4. XY sample data at keypad surface, before linearization (left), after linearization (right)

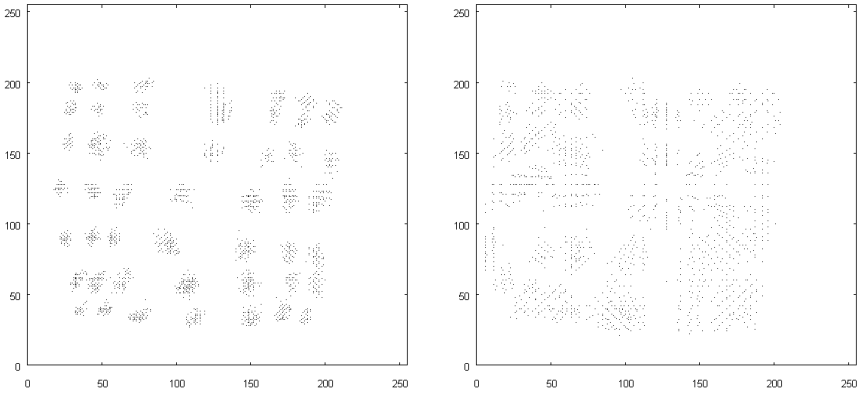

Fig. 5. XY sample data, 0.050" above keypad (left), 0.100" above keypad (right)

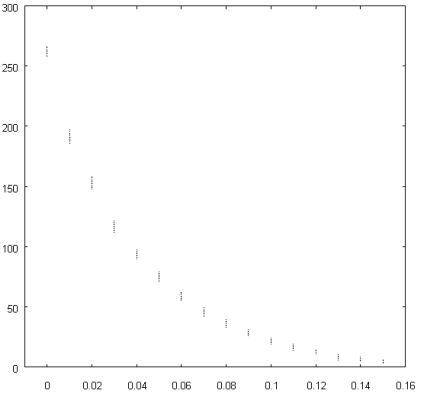

Fig. 6. Z sample data from key surface in 0.010" increments

The distribution of the points shows that the response of the pad is not linear in XY, as expected by the non-linear response of the capacitive charging curve, and due to the electrical fringing fields that are present on the board. Under our test mouse conditions, this did not prove to be overly difficult to use; likely due to the fact that hand-eye coordination can be used to overcome the non-linearity. However, if needed, the non-linearity can be corrected by using a discrete 2D lookup table with linear interpolation between table values, as shown in the right hand graph of Figure 4, where a 7x7 lookup table was used. Due to quantization error, the standard deviation increases to 2.32, but the results give a very good rectangular response. Of course, this comes at the cost of more code and data space in the microcontroller.

The Z performance of the VersaPatch was tested in two ways; first by holding the finger at different heights above the keys and applying the same scanning pattern, and then by showing the Z readings resulting from hovering over a single key and raising the finger by successive steps. The left half of Figure 5 shows the data from scanning the finger 0.050" higher off of the keypad. The right half of Figure 5 shows the same test with the finger scanning 0.100" off of the top of the keys. Both tests show increased noise as detailed in Table 1.

Table 1. Summary of 256x256 XY Testing Data

Altitude	# Points	Non-Linear Std. Dev.	Linearized Std. Dev.
On Sensors	6,945	0.874	0.7983
On Keys	11,574	2.16	2.32
0.050" Above Keys	9,050	3.99	4.592
0.100" Above Keys	9,541	6.993	10.10

Figure 6 shows the results of holding the finger over the center pad and then taking a series of data points at 0.010" increments rising off of the surface of the plastic keys. The exponential fall off of the electric field can be seen as a function of distance. The X axis shows height above the plastic keys in inches, and the Y axis shows the total capacitance value. Each altitude point is shown as a vertical bar to indicate the distribution of total capacitance values at that altitude.

It is important to note that our test finger does not perform as well as a real finger because a real finger deforms across the keys to give a larger cross sectional area. Also, body chemistry seems to reduce the overall noise seen by the sensor as compared to the brass finger. However, in order to maintain the best repeatability and accuracy in placement, we settled on the metallic finger for use as a stylus.

7 Implementation Issues and Discoveries

The main issue noted in our experiments is a sensitivity to electrical noise in the power system. This is not surprising since the input high threshold of the GPIO pin determines the detection point. Movement of that threshold causes fluctuations in the capacitive timer values making the system behave erratically. However, we have found that a linear regulator will negate local power supply noise issues.

The system is also somewhat sensitive to PCB layout issues, and best results will be obtained when the traces from the sensing pads are run over a ground plane on the PCB. Our experience is that using wire wrap wire to connect the sense pads to the microcontroller will make the system unusable due to electrical noise.

The C_{Total} quantity can be used for emulated button presses, as well as fairly coarse grain finger altitude detection. Although, the Z detection falls off quickly with the electric field as a function of $1/r^2$.

The fact that the conductor used for the sense pads can be of relatively poor quality has suggested the use of clear conductors overlaid directly on a display to allow for finger based device navigation. This is something that we have not experimented with as yet, but will be the subject of future investigations.

An implementation possibility that we have explored is the use of snap domes as the capacitive sensing element. These are commonly found in electronic devices, such as cellular phones, mounted on the PCB beneath the plastic buttons that the user sees. The snap domes make electrical contact for the switch, as well as producing the audible click and tactile feedback that is normally expected. Since these domes are metallic, the device can be built so that each dome is at the center of one of the sense pads shown in Figure 2, and the dome becomes part of the sense pad. The same algorithms described transform the normal keypad into a gesture or cursor positioning device at very little cost, while maintaining button functionality as normal. In our design, the dome electrically "hides" the normal button down contact under the dome so that it does not interfere with the capacitive sensing. The button down contact is grounded so that when activated, the sense pad appears to be stuck to ground, which is captured as a timeout by the firmware and reported as a button press. We have implemented this in a prototype device and have found it to work quite well.

8 Prior Work

A Texas Instruments application note describes the use of an MSP430 microcontroller to implement a capacitive touch slider [1]. The mean zero value is established in a somewhat similar way, but the method employed does not attempt to track the dynamic range of the capacitance values. Also, the methods discussed only work for a 1D capacitive sensor strip, rather than creating a 2D pad.

The most complete reference for capacitive sensing that we are aware of is a book by Baxter [2]. That book details a large number of circuits and measurement techniques and is a very good reference.

A relaxation oscillator based capacitive measurement circuit has been proposed by van der Goes, et. al., [7], and is believed to be at the heart of many commercial capacitive sensing devices. By using the capacitance of interest to vary the frequency of an oscillator, frequency measurements allow detection of changes in capacitance of less than 100aF. While the sensitivity of that method is startling, a custom ASIC was needed to realize it. We have found our method to have sufficient sensitivity for practical use as an input device, while still using an off the shelf microcontroller.

In [4], a technique is discussed for measuring small capacitances where the small capacitance to be measured is repeatedly charged and then switched in parallel with a larger capacitor of known value. The number of switches needed to charge the large

capacitor gives the size of the smaller capacitor. The added circuit complexity of that method would add to the overall cost of the solution as compared to the RC method proposed in this paper, both in terms of components and pins on the microcontroller.

References

1. Albus, Z.: PCB-Based Capacitive Touch Sensing With MSP430. Texas Instruments Application Report SLAA363A, Dallas, TX (2007), http://www.msp430.com
2. Baxter, L.K., Herrick, R.J.: Capacitive Sensors: Design and Applications. IEEE Press Series on Electronics (August 1, 1996)
3. Dietz, Paul, A.: Pragmatic Introduction to the Art of Electrical Engineering, Parallax, Rocklin, CA (1999), http://www.stampsinclass.com
4. Geiger, R., Allen, P.: VLSI: Design Techniques for Analog and Digital Circuits. McGraw-Hill, New York (1990)
5. Sinclair, M.J., Hinckley, K.P., Kajiya, J.T., Sherman, N.C.: Capacitance Touch Slider. US Patent #7,158,125, Issued, January 2 (2007)
6. Texas Instruments: MSP430x20x1, MSP430x20x2, MSP430x20x3 Mixed Signal Microcontroller, Dallas, TX (2007), http://www.msp430.com
7. Van der Goes, F.M.L., Meijer, G.C.M.A.: Novel Low-Cost Capacitive-Sensor Interface. IEEE Transactions On Instrumentation And Measurement 45(2), 536–540 (1996)