Cornelius van Drebbel's *Mad Design Contest*

NECTilt – A Tilt Based User Interface

©2004 John W. Peterson

Introduction

The availability of low cost, solid state accelerometers opens up great new possibilities for interacting with small electronic gadgets. NECTilt shows some creative applications for using an accelerometer with the EV9835.

An accelerometer measures the force of acceleration acting upon the device. This can be through movement, or by the force of gravity. By using gravity as the acceleration force, the device easily measures tilt angles.

The accelerometer used for this project is a Memsic MXD2004A[1]. It's a tiny, low cost, low power, solid-state CMOS part that uses minute temperature gradients across a bubble of air to measure acceleration forces in two dimensions. The device produces two pulse-width modulated outputs, one for measuring the X-axis, the other for Y. Overall the device has a range of 5*g*

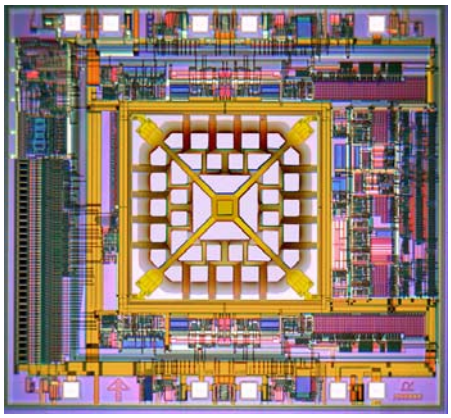


Figure 1 - Photomicrograph of the Memsic accelerometer

Two software applications for the accelerometer are presented. The first is a simple carpenter's level. This uses the tilt sensor as a traditional data source.

The other application is a rolling ball maze puzzle. Here the concept is to use the tilt sensor as a *user interface*, instead of the typical switches and buttons.

---

[1] See http://www.memsic.com

Design Considerations

The software developed works specifically with the Memsic MXD2004A sensor running with its default clock. The software could easily be enhanced to accommodate other similar sensors. Also, no attempt at temperature compensation is done. While not necessary for the simple applications presented here, systems expecting high precision measurements would require it (this is detailed in Memsic's application notes).

Microprocessor Functions Used

The design uses the LCD display, the parallel I/O ports, and the watch timer. No other external components or interface is required.

Electrical Interface

Other than the addition of the Memsic sensor, the EV9835 board is used unmodified for this project.

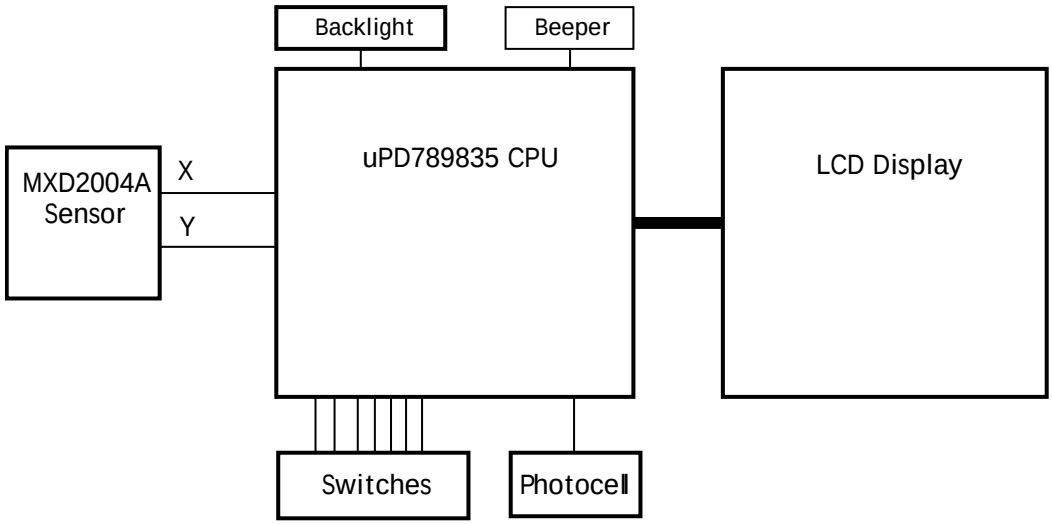


Figure 2 - System Overview Block Diagram

Connecting the sensor to the CPU is incredibly easy. The MXD2004A has a wide power supply range (2.7v – 5.2v) so it connects directly to the EV9835's power supply with nothing more than a bypass capacitor.

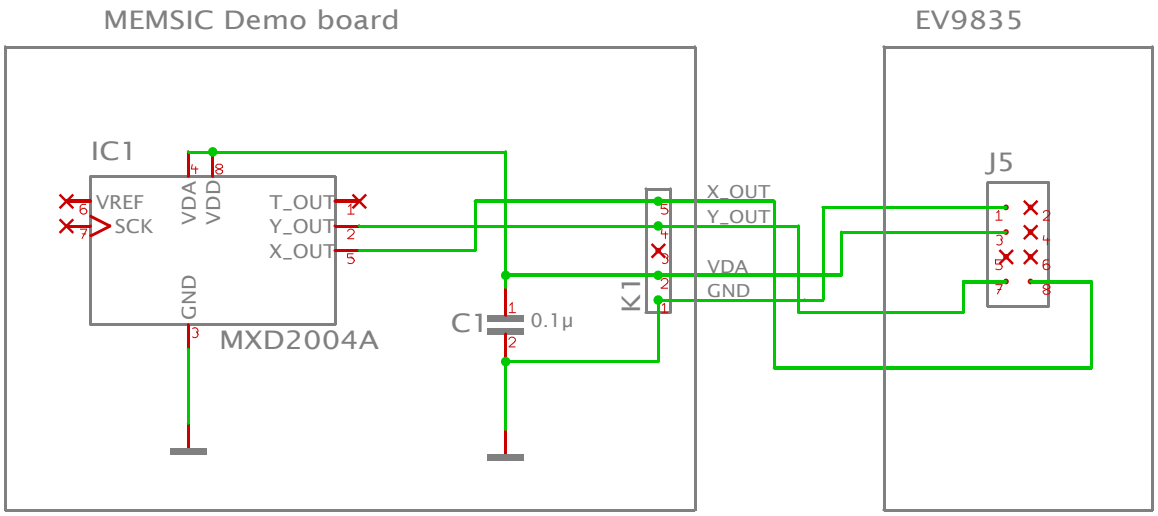


Figure 3 - Schematic Diagram

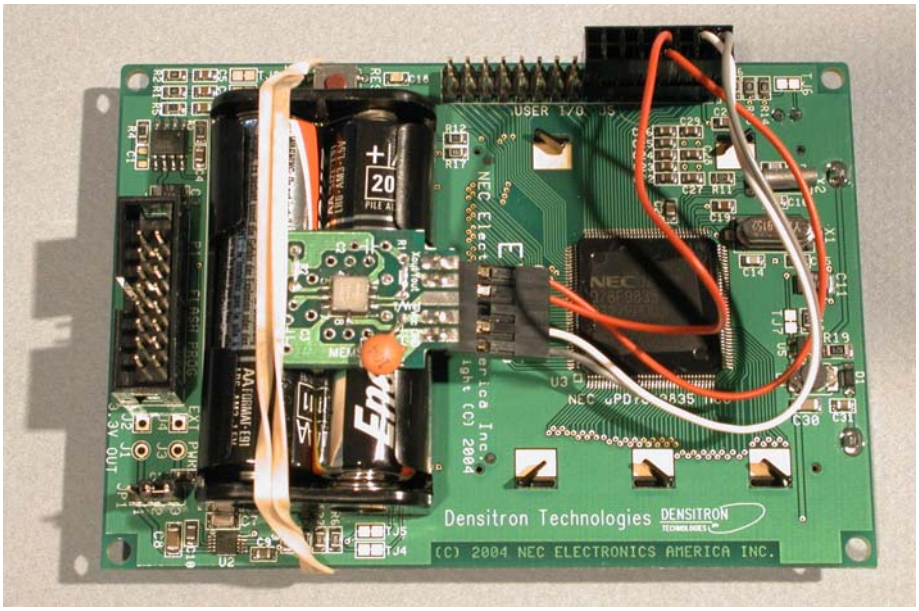


Figure 4 - EV9835 Board with Memsic Sensor Attached

The sensor was mounted on a demo board provided by Memsic. Other than assembling a simple connector for the two parts, no further electronics is needed. The PWM outputs of sensor are connected to bits 7 and 8 of port 3.

*Low-level Software**

The basic algorithm for reading the sensor line consists of measuring the pulse width. This is accomplished with a simple tight loop:

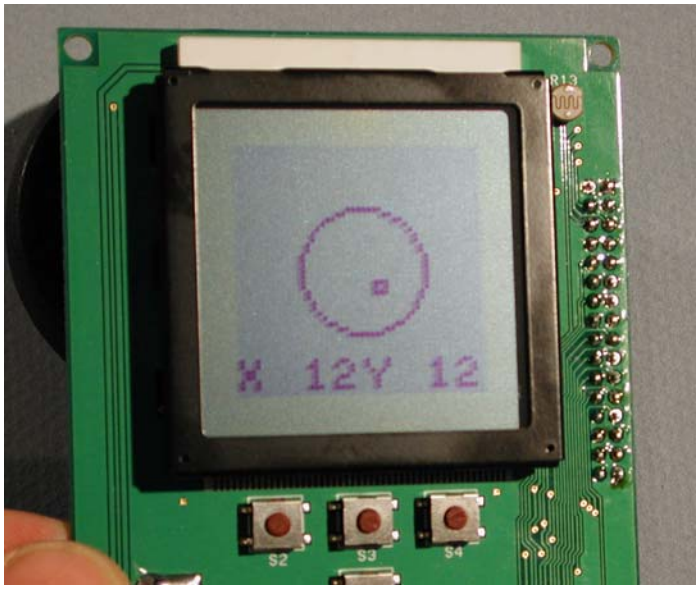
```
while (P3 & X_AXIS) {};              //Wait for line to go low
while ((P3 & X_AXIS) == 0) {};       //Wait for line to go high
while (P3 & X_AXIS)                  // Count while line is high
     count++;
```

Since the internal clock of the sensor is around a few hundred kilohertz, the count ranges from around 125 counts to 240 across 180 degrees of tilt. The counting is done with interrupts turned off, to avoid erratic results when timers go off or buttons are pushed.

To help smooth the data even further, four readings are taken and then averaged. These are then scaled down to the 0..48 range of the display, to make graphics computation work well. This is implemented in **`memsic.c`**

Application1 – Level

The Level application replicates a simple carpenter's level. In a traditional carpenter's level, a bubble in small tube of liquid indicates if the tube is level when the bubble is in the middle. Circular levels (sometimes used for leveling camera tripods) have the bubble floating under a disc, indicating a level orientation when the bubble is the center.



* Note a detailed description of the software is provided in lieu of flowcharts

Figure 5 - Level Application Display

This application is similar, except the indicator moves *towards* the tilt (instead of away from it, like a bubble level), and numeric readouts of the *X* and *Y* tilt angles are provided. This software takes advantage of the sample application provided with the EV9835. In particular, code for initialization, managing the LCD display, sound, the real time clock, and other functions where used.

The code for the level is quite simple:

```
SetLCDDisplayPattern('A');      // Set up display
SetLCDWritePattern('B');
ClearLCDPattern('A');
ClearLCDPattern('B');

while (Test_Switch() == 0)      // Pressing any switch exits
{
     ClearLCDWritePattern();

     // Read tilt data
     ReadScaledMemsic( &x, &y );

     // Draw results
     DrawRectangle( x-1, y-1, x+1, y+1, BLACK_DOT );
     DrawCircle( 24, 24, 12, BLACK_DOT );
     draw_angle( x, 0, 'X' );
     draw_angle( y, 4, 'Y' );

     // Swap buffers
     SwapLCDWritePattern();
     SwapLCDDisplayPattern();
}
```

The `draw_angle` routine formats the numerical result at the bottom of the display. The code uses "double buffering", where the graphics are written into an offscreen buffer before being swapped with the displayed one. This helps avoid any flicker in the display. As shown in the attached movie, the display is quite smooth. The level application is implemented in the file **`level.c`**

Application 2 – Tilt Maze

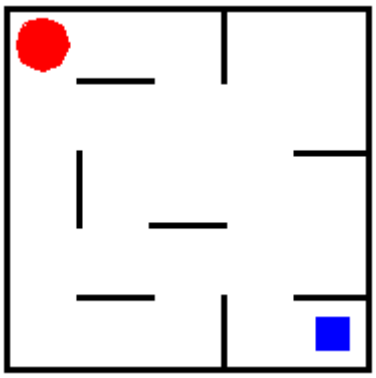


Figure 6 - A TiltMaze

The Tilt Maze is a puzzle created by Andrea Gilbert[2]. The idea is to roll a ball from its initial start position to a goal (marked by a small square). The ball always rolls until it comes to a wall, and only rolls horizontally or vertically. The puzzles are actually trickier than they appear, since the movements must go in a particular order to solve the puzzle.

In NECTilt, you tilt the EV9836 to move the ball, just like you would a real puzzle. In addition, some gestures are added to control the game. Shake the puzzle vertically, and it resets (useful if you get stuck). Shake the puzzle horizontally, and it switches to the next maze.

The maze is implemented as an array of bytes, one byte for every grid cell in the maze. Each byte has a bit set corresponding to any "walls" in the cell. In the sample above, for example, the top left cell would have the bits set for the North and West walls. A draw routine walks through the array of cells and draws lines corresponding to the bits set.

The code monitors the tilt sensor and starts the ball rolling (either North, South, East or West) when it's been tilted far enough. When a roll starts, the software examines the cells in the path of the roll direction, and stops when it encounters a wall bit corresponding to the direction of travel. The roll is animated (using a double-buffer scheme similar to the level application), then the code waits for a tilt in a new direction. When the goal is reached, it beeps and blinks the backlight.

To implement the shake gestures, the routine for reading the Memsic sensor was extended to record every time the it reached "full scale" maximum and minimum values in each direction (X and Y). The watch timer ISR was modified

[2] See http://www.clickmazes.com/newtilt/ixtilt2d.htm

so every second it saves these maximum and minimum counts, and then resets them. To see if a shake occurred, the last count of both limit values is read. If both limits were hit for a particular direction within the last interval, a "shake" occurred.

Like the level, the TiltMaze application also leverages code from the sample application for graphics, sound and RTC control. It is implemented in the file **`tiltmaze.c`**

Other Software details.

The folder "Application" contains the full source code described above. The demo application supplied with EV9835 was used as a starting point. The applications in it were removed, but the library code was used and extended. The applications described above were added to this frame work.

A simple test harness selects the applications; pressing switch S3 selects *Level*, pressing S4 selects *TiltMaze*, and pressing S2 displays some debugging information used to get the sensor code up and running.

Conclusion

NECTilt demonstrates using a tilt sensor for both measurement, and as a user interface replacing the traditional buttons and switches. Using a tilt sensor for user interface opens the door for a range of applications from novel games to specialized tools. For example, in environments where buttons are clumsy awkward to use (for example, while wearing heavy gloves) a gesture based interface implemented with an accelerometer is a valuable alternative.

Supplied Files

`Application`	Source code in C for the PM Plus IDE
`LevelDemo.mov`	Quicktime movie of the Level application
`TiltMazeDemo.mov`	Quicktime movie of the TiltMaze application
`Shake_Gesture.mov`	Demonstration of the shake gesture
`Pictures`	Photos (jpg format) of the system

Contact

John W Peterson - necdesign@saccade.com - 650 854 8538
12 Bishop Lane
Menlo Park, CA 94025 USA