커패시티브 센싱 기술로 오류 키 활성화 피하기

By (Ryan Seguine, Product Marketing Engineer, Cypress Semiconductor Corp.)

최근 디지털 뮤직 플레이어에서 커패시티브 센싱 (capacitive sensing)의 성공은 차별적인 요소를 부가하려는 다른 산업과 제품에 관심을 불러일으키고 있다. 커패시티브 센싱 기술은 휴대폰 메뉴의 네비게이션에서 부터 자동차 프론트 패널 디스플레이 버튼에 이르기까지 모든 것들을 위해 선호되는 입력 방식이 빠르게 되어가고 있다. 커패시티브 센싱은 기존의 기계 버튼, 슬라이더, 분압기 등을 위한 많은 장점을 제공한다. 손가락은 유리나 플라스틱의 각기 다른 굵기를 통해 감지될 수 있기 때문에 커패시티브 센싱은 산업용 기기 및 백색 가전 등을 포함한 많은 애플리케이션 영역에 내구성을 더해 준다.

기계식 스위치는 큰 손가락으로 보다 작은 버튼을 실행하려 할 때의 힘인 실질적인 접속을 필요로 한다. 사용자의 입력을 기반으로 하여 어떤 버튼이 효율적인지를 현명하게 결정하기 위한 커패시티브 센서의 능력은 소비자들을 위한 제품을 설계할 때 아주 중요한 요소이다. 이 글은 잘못된 커패시티브 센서 활성화를 없애기 위한 방법들을 얘기하고 있다.

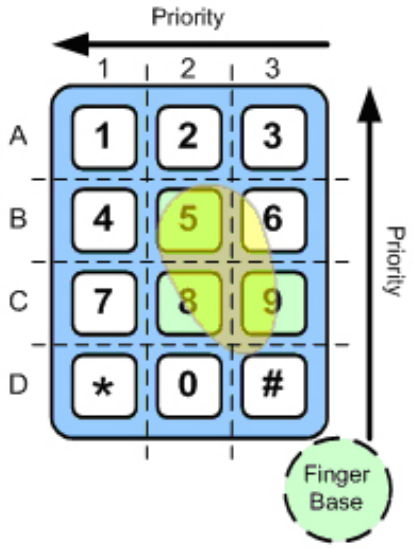
하나의 유저 인터페이스에서 버튼 상태를 결정하기 위한 아주 많은 테크닉들이 있다. 이러한 테크닉들은 센싱 디바이스가 필요한 결정을 하기에 충분한 지적 인원들이 있다면 보다 쉽게 커패시티브 센싱에 다가선다. 이 글에서는 3 가지 테크닉들이 논의되고 있다. 첫 번째는 사용 모델이 어떤 버튼에 중요성을 더하기 위한 영역을 만들어 가는 우선순위 계획(priority scheme)이다. 두 번째는 첫 번째 버튼이 활성화되기 위해 놓여질 때까지 활성화 상태로 머무는 시점 계획(timing scheme)이다. 세 번째는 “gorilla-proofing”이라 불리며 많은 센서들이 활성화 될 때 센서 활성화를 방지하기 위한 방법이다.

키의 우선순위 (Key Priority)

모든 애플리케이션은 하나의 사용 모델을 가지고 있다. 인터페이스와 함께 하는 유저 인터페이스는 어떨까? 일반적인 작동을 하는 동안 인터페이스상에 놓여진 손가락은 어디 일까? 이러한 질문에 대한 답은 행렬, 종대 혹은 열에서 어떻게 다른 키들이 우선순위가 매겨지느냐를 규정하는 것에 있다.

행렬 키의 경우, 사용자의 손가락은 일반적으로 활성화 영역 아래 놓여진다. 이는 손가락이 활성화 영역에 보다 타원으로 보일 것이라는 것을 의미한다. 한 개 이상의 센싱 요소가 동시에 활성화될 수 있다는 것은 분명 가능한 일이다. 이와 같은 경우, 손가락 출발점에서 가장 멀리 있는 활성화 된 센싱 요소가 실질적으로 누르는 버튼이다. 그림 1 이 이를 보여주고 있다.

그림 1.



그림에서 보는 것 처럼, 센서 5, 8, 9 는 모두 활성화 상태이다. 그러나 아래 테이블 1 에 나타난 것처럼, 열이나 종대로 우선순위를 배정하면 마이크로컨트롤러가 정확하게 어떤 센서로 사용자가 활성화하려는 것 인지와 관련한 결정을 할 수 있도록 해 준다.

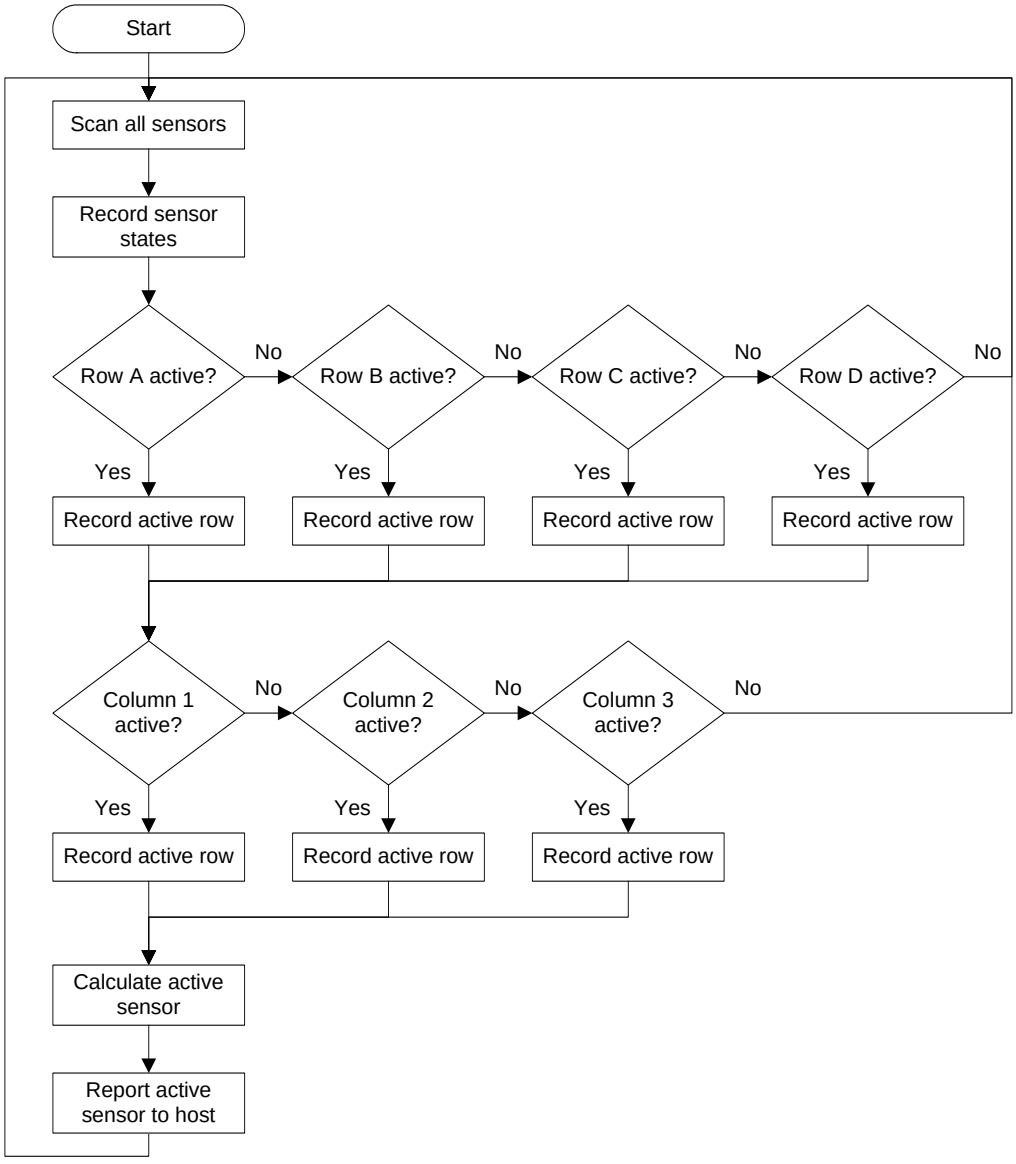
테이블 1.

Row	Column
A	1
B	2
C	3
D	

그림 1 에서 보이는 활성화와 함께, B 와 C 열은 활성 상태이다. B 가 C 보다 더 높은 우선 순위를 가지고 있기 때문에 B 열에 있는 센서는 활성으로 보고된다. B 열에서는 단 하나의 센서 활성화가 있는 관계로 (버튼 “5”) 센서는 호스트로 보고된다. 그러나, 버튼 “4”에 반응하는 센서 또한 활성화 되었다면 “4”에 대한 센서는 호스트에 활성화되는 것으로 보고될 것이다.

프로그램 작동을 위한 플로어챠트는 아래 그림 2 에서 보여주고 있다.

그림 2.

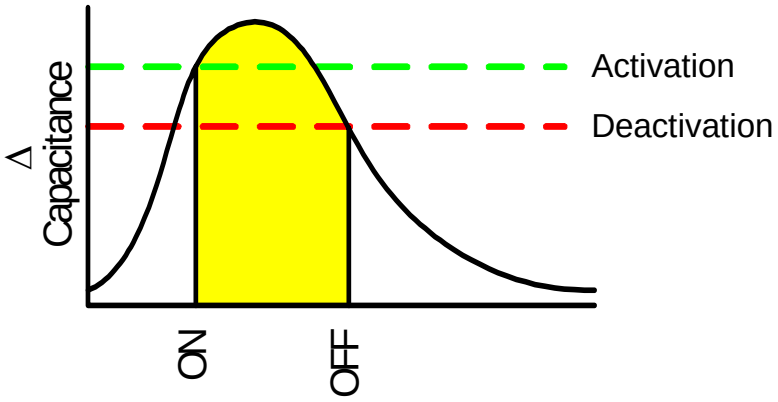


시점 계획 (Timing Scheme)

또 다른 이용 모델에서 사용자는 올바른 버튼 위에 손가락을 놓고 있으며, 그 버튼만이 활성화 된다. 제거되어야 할 오류 탐지는 손가락의 우발적인 움직임으로부터 기인한다. 손가락이 필요한 센서와의 접촉을 유지하는 동안 불필요한 센서 역시 활성화 될지도 모른다. 이와 같은 이용 모델에서 우선순위 시스템은 가장 효율적이거나 가장 효과적인 방법이 아닐 수도 있다. 필요한 것은 첫 번째 센서가 활성화 상태를 유지하는 동안 또 다른 센서가 활성화 될 수 있는 손가락의 작은 움직임을 무시하기 위한 방법이다.

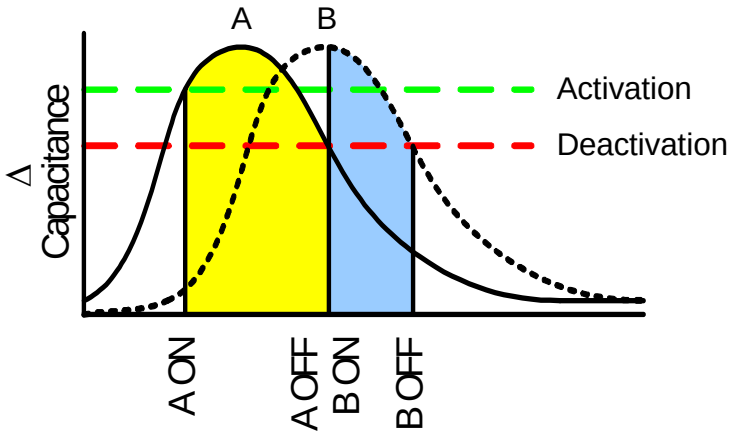
커패시티브 센싱은 종종 시작과 함께 작동한다. 이러한 시작은 손가락이 센서에 놓여질 때 뿐만 아니라 뺄 시점도 결정한다.

그림 3.



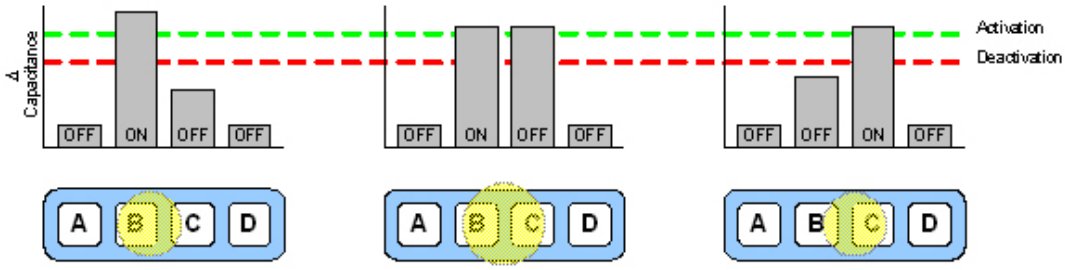
손가락의 위치로부터 커패시턴스에 있는 긍정적인 변화가 첫 번째 시작(활성화)에 이를 때 센서는 ON 으로 보고된다. 손가락이 센서의 중앙에서 떨어지거나 터치 표면에서 놓여지기 시작함으로써 커패시턴스의 변화는 줄어든다. 그러나, 센서의 상태는 커패시턴스 변화가 두 번째 시작 (비활성화) 이하로 떨어질 때까지는 변화하지 않는다. 이는 그림 4 에서 보여주고 있다.

그림 4.



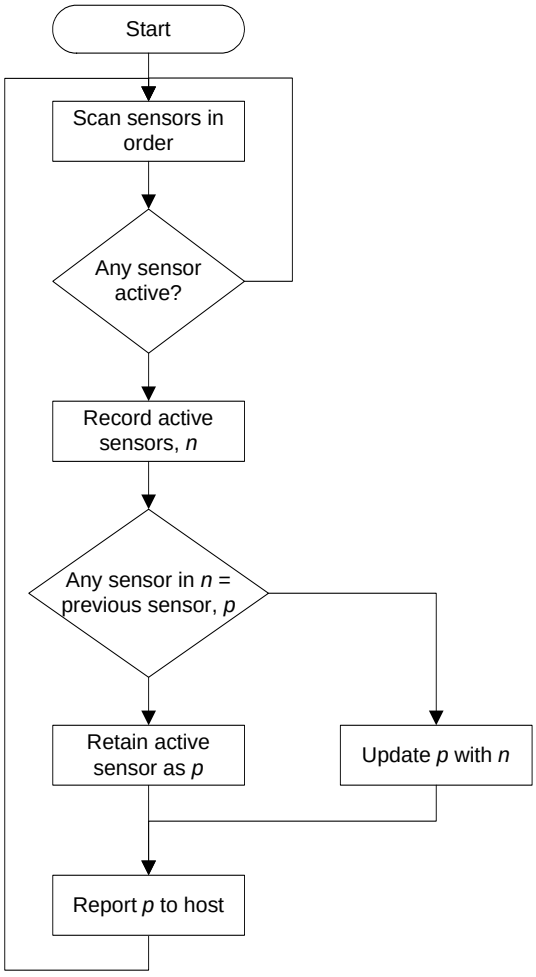
단일 열에 위치한 더 큰 숫자의 버튼상에 있는 이러한 계획을 보면 움직임 역시 탐색할 수 있다. 이는 그림 5 에서 보여주고 있다.

그림 5.



이러한 기능을 위한 플로어챠트는 그림 6 에서 보여주고 있다.

그림 6.



Gorilla-proofing

오류 센서 활성화를 없애기 위한 세 번째 테크닉은 많은 센서가 활성화 되었거나 시그널을 무시할 때 인식하기 위한 것이다. 인터페이스 상에서 손바닥을 누름으로써 기능을 테스트 하기 때문에 이를 “gorilla-proofing”이라 부른다. 이 아이디어는 어떤 센서가 활성화되는 가를 구별하기 위한 것은 아니다. 이런 경우는 커패시티브 센싱 디바이스가 주머니에 있거나 커패시티브 센싱 전화기가 얘기할 때 누군가의 뺨 옆에 놓일 때 발생하곤 한다.

활성화되는 요소는 손가락 보다 훨씬 더 크다. (위의 사례에서 활성화되는 요소가 다리나 뺨일 경우 처럼).

그림 7.

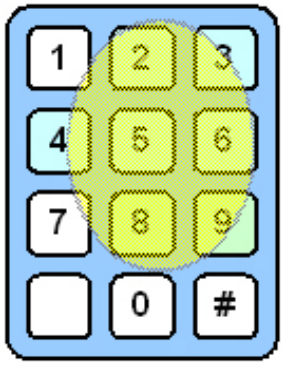


그림 7 에서, 7 개의 요소가 활성화 상태이다. 전화기 키패드의 사용 모델을 위해 7 개의 센서가 의도적으로 활성화 될 수 있을 것 같지는 않으며, 사실 3 개 이상의 센서가 활성화 되는 것 같지 않다. 그림 7 에서 보여준 경우에서 모든 센서 입력은 무시되었다. 이는 호스트에 센서의 상태를 보내기 전에 누른 버튼 수를 증가함으로써 이뤄진다. 활성 센서의 수가 3 개를 넘는 관계로 모든 버튼 상태와 동작은 취소되었다. 캡센스(CapSense)와 함께 하는 싸이프레스의 PSoC 혼합 시그널 어레이에서 이와 같은 특징을 구현하기 위한 하나의 코드 사례는 다음에서 보여주고 있다.

코드 사례;

```
void main(){
        int iActiveButtons;              // Variable for storing total 'on' buttons
        CSR_1_Start();                   // Set PSoC up to run CapSense
        LCD_1_Start();                   // Set PSoC up to run LCD
        M8C_EnableGInt;          // Enable global interrupts
        CSR_1_SetDacCurrent(0x15,0);     // Set iDAC in low mode (1.43uA)
        CSR_1_SetScanSpeed(3);           // Set PWM-high time to 1 oscillation (3-2=1)
```

```
        CSR_1_StartScan(0,7,1);  // Scan 7 buttons starting at 0
        while(1){
                while( 0 == (CSR_1_GetScanStatus() & CSR_1_SCAN_SET_COMPLETE));
                CSR_1_bUpdateBaseline(0);        // Run UpdateBaseline Routine
                // Store how many buttons are active.
                if(CSR_1_baSwOnMask[0] & 0x01) iActiveButtons++;
                if(CSR_1_baSwOnMask[0] & 0x02) iActiveButtons++;
                if(CSR_1_baSwOnMask[0] & 0x04) iActiveButtons++;
                if(CSR_1_baSwOnMask[0] & 0x08) iActiveButtons++;
                if(CSR_1_baSwOnMask[0] & 0x10) iActiveButtons++;
                if(CSR_1_baSwOnMask[0] & 0x20) iActiveButtons++;
                if(CSR_1_baSwOnMask[0] & 0x40) iActiveButtons++;
                if(CSR_1_baSwOnMask[0] & 0x80) iActiveButtons++;
                // If more than 3 buttons are active, change device status to "off."
                LCD_1_Position(01,00);
                if(iActiveButtons < 4){
                        LCD_1_PrCString("On");
                        LCD_1_Position(01,02);
                        LCD_1_PrCString(" ");
                }
                else LCD_1_PrCString("Off");
                iActiveButtons = 0;
        }
}
```

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com