Embedded-Systeme mit Berührungs- und Näherungssensor-Funktionen

Das Code-Free-Konzept macht es möglich

Von Mark Lee, Principal Applications Engineer, Cypress Semiconductor Corp.

Einführung

Auf kapazitiven Berührungssensoren beruhende Benutzeroberflächen haben sich in den vergangenen Jahren in vielen Consumer-Produkten sowohl als praxisgerechter wie innovativer Ersatz für konventionelle Tasten durchgesetzt. Leichtes Berühren einer glatten Oberfläche oder das Annähern der Hand ist ausreichend - das Betätigen eines mechanischen Schalters wird überflüssig.

Kapazitive Sensormodule machen es zu einer einfachen Aufgabe, ein System durch eine auf Berührung oder Näherung ansprechende Benutzeroberfläche aufzuwerten, denn das Schreiben oder Debuggen von Programmcodes ist vollkommen überflüssig. Es ist nichts weiter zu tun, als das fertig ausgearbeitete Modul in das eigene System einzufügen. Allerdings sind die Sensormodule selbst hinsichtlich der von ihnen ausgeführten Funktionen so unflexibel, dass sie ein Produkt nur in begrenztem Umfang um neue Features bereichern können. Hier wäre eine vielseitigere Lösung wünschenswert, denn oft sind es gerade die in letzter Minute eingefügten kleinen Extras, die ein Produkt aus der Masse der Wettbewerber herausheben.

Der vorliegende Artikel beschreibt, wie sich kapazitive Sensorfunktionen in ein einfaches Embedded-System einfügen lassen, ohne auch nur eine einzige Codezeile zu schreiben. Dieses ‚Code-Free'-Konzept verbindet den Komfort fertig ausgearbeiteter Module mit der Flexibilität eines programmierintensiven Embedded-Controllers. Der gesamte Designprozess wird in diesem Beitrag an Hand einer Produktidee aus der Praxis demonstriert und deckt sämtliche Phasen vom anfänglichen Konzept bis zum funktionierenden Prototyp ab.

Möglich wird das beschriebene ‚Code-Free'-Konzept für das Systemdesign durch die PSoC Express Software und das FirstTouch Evaluierungs-Kit – beide von Cypress Semiconductor. Kreative Menschen, die per Brainstorming neue Produkte ersinnen, sind nicht mehr darauf angewiesen, das Design an die Technikabteilung weiterzureichen, wenn sie einen funktionierenden Prototypen wünschen. Wer in der Lage ist, ein Blockschaltbild seines Systems zu zeichnen, kann mit Hilfe der grafischen Benutzeroberfläche von PSoC Express auch ein reales Produkt generieren. Selbst Anwender, die noch nicht mit PSoC Express gearbeitet haben, benötigen dafür nur wenige Stunden. Die Entwicklungszeit reduziert sich hierdurch ebenso wie die Projektkosten.

Entwicklung einer besseren Mausefalle

Eine geradezu klassische Aufgabe für Erfinder ist die Entwicklung einer besseren Mausefalle. Das nachfolgend beschriebene Designbeispiel kommt diesem Ziel jedenfalls sehr nahe. Es soll ein Embedded-System entwickelt werden, das zwei Trends im Bereich der Elektronik – kapazitive Sensoren und die Digitalfotografie – miteinander kombiniert. Das Projekt basiert auf einem

intelligenten Näherungssensor, der eine Digitalkamera auslöst. Solche Systeme, die auch als ‚Kamerafalle' bezeichnet werden, haben in den letzten Jahren in wissenschaftlichen Veröffentlichungen für Schlagzeilen gesorgt, denn mit ihrer Hilfe gelang der Nachweis, dass als ausgestorben geltende Tiere wie etwa der Jaguar im US-Bundesstaat Arizona [1] nach wie vor auf der Pirsch sind. In entlegenen Gebieten wie etwa den dichten Wäldern Borneos entdeckten Wissenschaftler mit diesen Kameras sogar völlig neue Arten [2].

Allem Anschein nach bewährt sich die Technik, die bisher für Kamerafallen verwendet wird, ausgezeichnet. Eigentlich gibt es also keinen Grund, das Rad noch einmal zu erfinden, oder doch? Eine kurze Gegenüberstellung von alter und neuer Technik soll zeigen, was in der Tat für die Entwicklung einer besseren Falle spricht.

Da die alte Technik auf PIR-Bewegungssensoren basiert, gehen Tiere bei hohen Umgebungstemperaturen in der Hintergrundszenerie unter, und reflektiertes Sonnenlicht führt zu Fehlauslösungen. Die sperrige Hardware muss zudem einerseits getarnt und vor Umwelteinflüssen geschützt werden, während andererseits jegliche Sichthindernisse die Auslösung vereiteln. Hinzu kommt, dass der Auslösebereich mit einfachen Linsen schwierig anzupassen ist.

Die neue Technik beruht dagegen auf einem kapazitiven Näherungssensor und nutzt einen Draht als solchen. Dieser Draht kann unauffällig an einem Zweig, auf einem Felsen oder neben einer Wasserstelle platziert werden, und der Auslösebereich lässt sich mit einem einfachen Seitenschneider genau eingrenzen und abstimmen. Die Technik ist bei Nacht genauso funktionstüchtig wie bei hellem Sonnenlicht, und die Hintergrundwärme hat keine Auswirkungen auf die Leistungsfähigkeit. Da der Näherungssensor nur periodisch aktiviert wird und die Kamera nur dann eingeschaltet wird, wenn sich ein Tier in der Auslösezone befindet, wird sparsam mit der Batteriekapazität umgegangen.

Vergleicht man beide Verfahren zur Herstellung einer Kamerafalle, so wird deutlich, welche Vorteile das kapazitive Sensorverfahren in einigen Situationen gegenüber der gegenwärtigen Technik bietet. Die neue Lösung dürfte deshalb ihren Platz in der Erforschung der Tierwelt finden.

Der Designprozess im Überblick

Der Code-Free-Designprozess gliedert sich in sechs Schritte:

1. Beschreibung des Systems in Worten
2. Erstellen eines Blockschaltbilds
3. Definieren der Zustandslogiken, Zustandswechsel-Funktionen und Wahrheitstabellen
4. Simulation des Systems
5. Test des realen Systems
6. Feinabstimmung des CapSense-Elements

Schritt 1: Beschreibung des Systems in Worten

Das Design soll nach dem Top-Down-Prinzip angegangen werden. Es wird also zunächst auf einer hohen Abstraktionsebene in Worten die angestrebte Funktion des Systems festgelegt, um die technischen Details anschließend bedarfsentsprechend hinzuzufügen. Es folgt eine kurze, nichttechnische Beschreibung des Systems:

Das System überwacht fortlaufend, ob sich ein Tier in der Nähe befindet. Wird ein Tier registriert, wird die Kamera aktiviert und ein Bild aufgenommen. Wird kein Tier erkannt, bleibt die Kamera abgeschaltet. Um zu verhindern, dass beim Aufstellen

der Kamera Bilder vom Forscher aufgenommen werden, wird das Erkennungssystem nach einem Reset für eine bestimmte Zeitspanne deaktiviert.

Diese wenigen Sätze sind sicherlich ein guter Ausgangspunkt, doch für die Arbeit mit PSoC Express sind ein paar mehr Details erforderlich. Hier also eine etwas technischere Beschreibung der vorgesehenen Funktionsweise:

1. Aufstellen der Kamera
2. Anbringen des Drahts in der Auslösezone (z. B. Befestigung an einem Zweig). Die Farbe des Isolationsmaterials sollte in der jeweiligen Umgebung möglichst wenig auffallen.
3. Verbinden des Sensordrahts und des Steuerkabels für die Kamera mit der Steuerungsplatine.
4. Einschalten der Stromversorgung für die Steuerungsplatine.
5. Für eine Zeitspanne von 30 Sekunden signalisiert eine blinkende Leuchtdiode (LED) dem Forscher, dass er sich ohne Auslösen der Kamera aus dem Wirkungsbereich des Systems entfernen kann. Während dieser Zeit ist die Kamera noch nicht scharf geschaltet.
6. Die blinkende LED erlischt.
7. Die Auslösefunktion für die Kamera ist jetzt scharfgeschaltet und reagiert auf Tiere, die dem Näherungssensor nahe kommen.
8. Sobald ein Tier den Näherungssensor auslöst, leuchtet eine grüne LED auf. Die Kamera wird eingeschaltet und ein Bild aufgenommen.
9. Verlässt das Tier den Auslösebereich, so erlischt die grüne LED. Wenn für eine gewisse Zeit keine Aktivierung erfolgt, schaltet sich die Kamera ab.
10. Zurück zu Schritt 7.

Schritt 2: Erstellen eines Blockschaltbilds

Mit Hilfe von PSoC Express wird ein Blockschaltbild des Systems gezeichnet (Bild 1). Dies geschieht in grafischer Form nach dem Drag-and-Drop-Verfahren. In PSoC Express ist zu diesem Zweck eine Bauelemente-Bibliothek enthalten. Dieser so genannte Treiberkatalog enthält verschiedene Funktionsblöcke wie zum Beispiel Temperatursensoren, Lichtsensoren und Beschleunigungsaufnehmer. Die benötigten Blöcke werden per Maus aus dem Katalog in das Blockschaltbild gezogen. Aus Gründen der Einfachheit besitzt das hier beschriebene System als Ausgänge nur die grüne und die rote LED. Es bereitet keine Schwierigkeiten, die an die Kamera gerichteten Ausgangssignale ‚Kamera ein/aus' und ‚Bild aufnehmen' zu implementieren, indem dem Treiberkatalog zwei zusätzliche digitale Ausgänge entnommen werden.

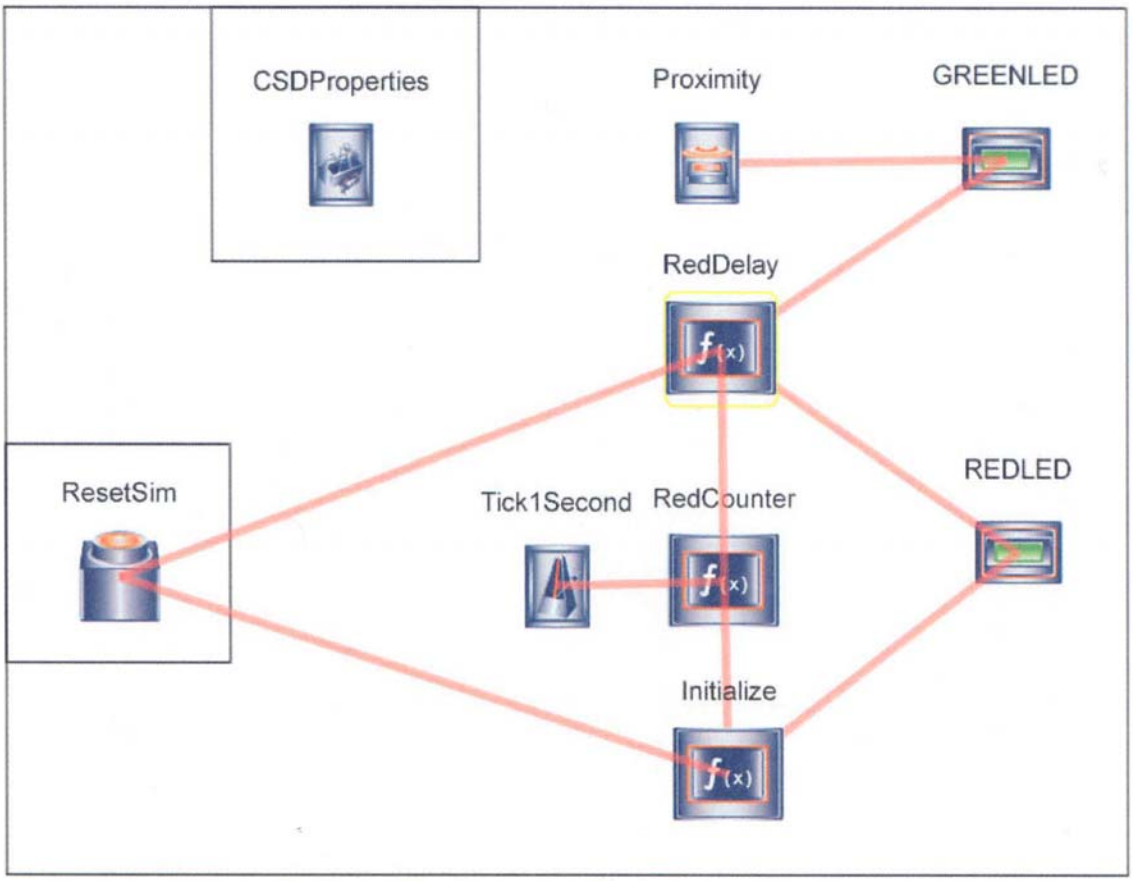


Bild 1. Blockschaltbild des Systems

Das in diesem Designbeispiel behandelte System besteht aus den folgenden Blöcken:

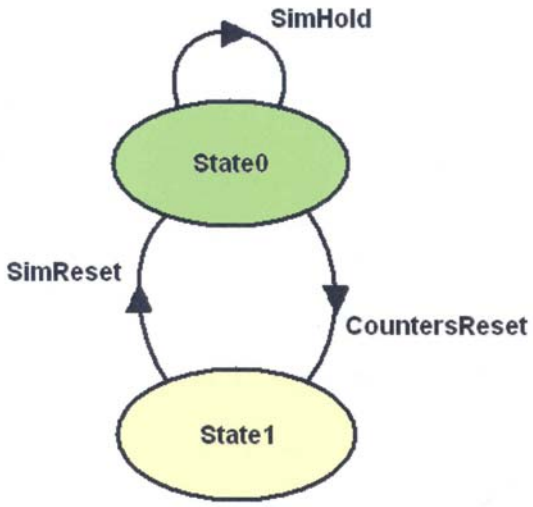
- Proximity – Dieser Näherungssensor basiert auf der CSD-Methode der CapSense-Lösung.
- GREENLED – Wird eingeschaltet, wenn der Näherungssensor ausgelöst wird.
- REDLED – Blinkt während der 30 Sekunden dauernden Unwirksamschaltung nach einem Reset.
- Tick1Second – Diese Zeitbasis gibt einen Impulszug mit einer Rate von einem Impuls pro Sekunde aus.
- RedCounter – Zähler, dessen Zählerstand sich pro Sekunde um eins erhöht.
- RedDelay – Zustandslogik, die ihren Zustand nach 30 Sekunden wechselt.
- Initialize – Zustandslogik, die den Zähler bei einem System-Reset zurücksetzt.

Schritt 3: Definieren der Zustandslogiken, Zustandswechsel-Funktionen und Wahrheitstabellen

Auch wenn mit PSoC Express kein Code geschrieben werden muss, ist es notwendig, eine logische Beschreibung des Systems zu definieren, was mit Hilfe von Zustandslogiken, Zustandswechsel-Funktionen und Wahrheitstabellen erfolgt.

Die beiden einfachen Zustandslogiken in diesem Design weisen die gleiche Grundstruktur auf. In PSoC Express werden die jeweiligen Zustandsdiagramme gezeichnet (Bild 2 und 3). Die Zustandswechsel-Logik ist in der Tabelle zum jeweiligen Bild wiedergegeben.

Die Zustandslogik ‚Initialize' ist in dem System enthalten, um die Voraussetzungen für benutzerfreundlichere Simulationen mit Hilfe des Resetschalters ‚ResetSim' zu schaffen. Wie die Zustandswechsel-Tabelle in Bild 2 zeigt, bleibt diese Logik in State0, solange der Resetschalter On-Status hat. In allen anderen Fällen befindet sich die Logik in State1, in dem der Zähler weiterzählen kann.

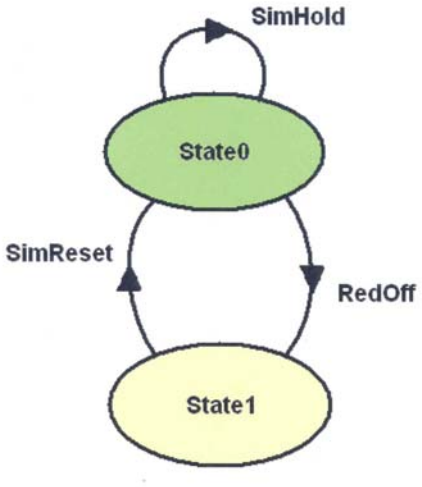


Transfer Function: Initialize State Machine

TRANSITION NAME	FROM STATE	TO STATE	EXPRESSION
CountersReset	State0	State1	ResetSim==ResetSim__Off
SimReset	State1	State0	ResetSim==ResetSim__On
SimHold	State0	State0	ResetSim==ResetSim__On

Bild 2. Zustandsdiagramm für die Zustandslogik ‚Initialize'

Die Zustandslogik mit der Bezeichnung ‚RedDelay' hat die Aufgabe, nach einem Reset den Ablauf einer Zeitspanne von 30 Sekunden zu signalisieren. Die Zustandswechsel-Tabelle in Bild 3 ist selbsterklärend.



Transfer Function: RedDelay State Machine

TRANSITION NAME	FROM STATE	TO STATE	EXPRESSION
RedOff	State0	State1	RedCounter>=30
SimReset	State1	State0	ResetSim==ResetSim__On
SimHold	State0	State0	ResetSim==ResetSim__On

Bild 3. Zustandsdiagramm für die Zustandslogik ‚RedDelay'

In dem Design sind drei Wahrheitstabellen enthalten, die von einer besonderen, als ‚Prioritäts-Encoder' bezeichneten Art sind. Dieser Encoder hat einen einzigen Ausgang, dessen Zustand von den logischen Ausdrücken in der Tabelle bestimmt wird. Die Ausdrücke werden einer nach dem anderen ausgewertet, bis sämtliche Bedingungen für einen der Ausdrücke erfüllt sind. Aufgabe dieser drei Tabellen ist es, den Zähler zu inkrementieren und die LEDs ein- und auszuschalten (Bilder 4, 5 und 6).

Transfer Function: RedCounter Priority Encoder

	EXPRESSION		
IF	Initialize_state==Initialize_state__State0	THEN	0
ELSEIF	RedCounter>=60	THEN	RedCounter
ELSEIF	Tick1Second==Tick1Second__Triggered	THEN	RedCounter+1

Bild 4. Definition von RedCounter

Transfer Function: GREENLED Priority Encoder

	EXPRESSION		
IF	(RedDelay_state==RedDelay_state__State0)	THEN	GREENLED__OFF
ELSEIF	(Proximity_Status > 0)	THEN	GREENLED__ON
ELSEIF	1	THEN	GREENLED__OFF

Bild 5. Definition von GREENLED

Transfer Function: REDLED Priority Encoder

	EXPRESSION		
IF	Initialize_state==Initialize_state__State0	THEN	REDLED__ON
ELSEIF	RedDelay_state==RedDelay_state__State0	THEN	REDLED__ON
ELSEIF	RedDelay_state==RedDelay_state__State1	THEN	REDLED__OFF

Bild 6. Definition von REDLED

Schritt 4. Simulation des Systems

Die Simulationsfunktion gehört zu den besonders leistungsfähigen Features von PSoC Express. Bild 7 gibt die Simulations-Darstellung des Designbeispiels wieder. Mit diesem Hilfsmittel lässt sich überprüfen, ob die Zustandswechsel des Systems zu den gewünschten Zeiten erfolgen. Außerdem können Was-wäre-wenn-Experimente durchgeführt werden, ohne dass irgendein Bauelement programmiert werden muss. Sobald alles wie vorgesehen funktioniert, ist es an der Zeit, einen Baustein zu programmieren und zu prüfen, wie sich das System unter Praxisbedingungen verhält.

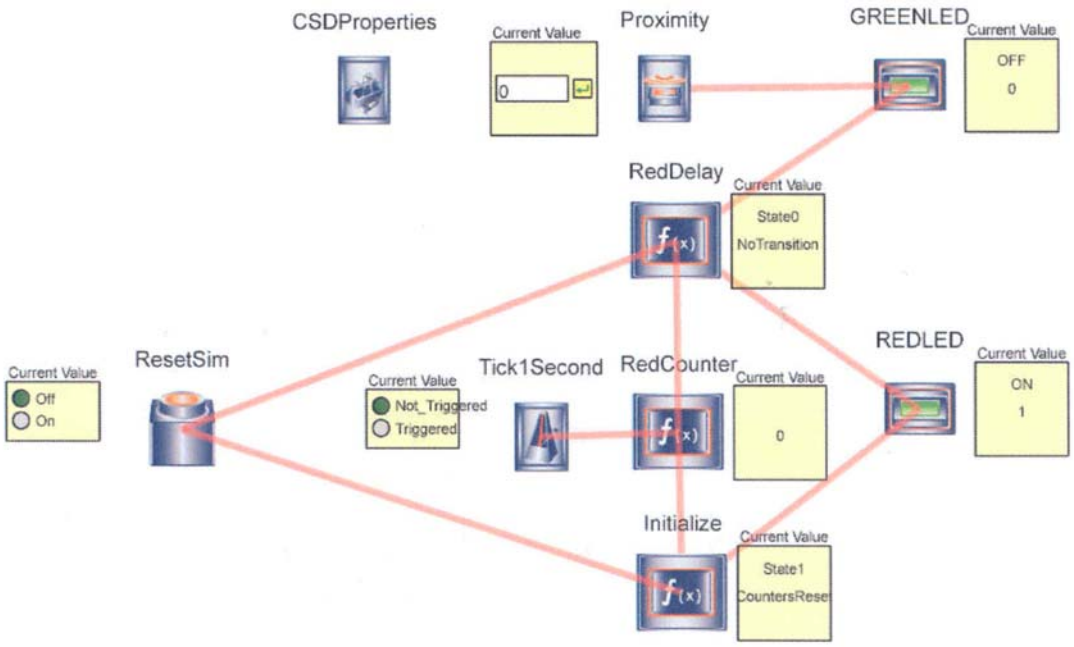


Bild 7. Simulation des Systems

Schritt 5: Test des realen Systems

Das FirstTouch Evaluation Kit, das mit ca. 30 US-Dollar sehr preisgünstig ist, hat das Format eines USB-Sticks (Bilder 8 und 9). Dennoch ist in diesem winzigen Entwicklungssystem alles enthalten, was zur Evaluierung des Designbeispiels erforderlich ist.

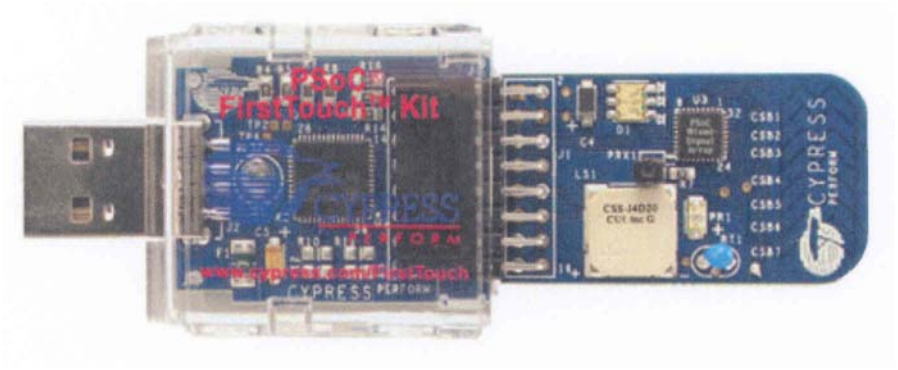


Bild 8. Ansicht des FirstTouch-Kits mit USB-Interface und Entwicklungs-Board

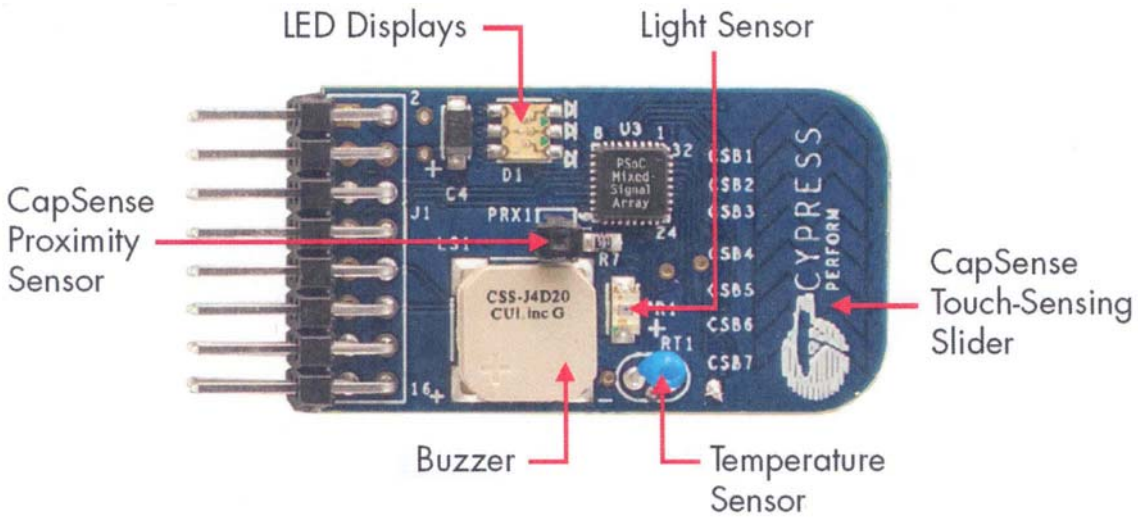


Bild 9. Detailansicht des Entwicklungs-Boards des FirstTouch-Kits mit dem Anschluss für den Näherungssensor-Draht

Zum Programmieren des FirstTouch-Kits muss zunächst das korrekte PSoC CY8C21434 ausgewählt werden, bevor wie in Bild 10 gezeigt den einzelnen Funktionen die richtigen Pins zugewiesen werden. Das Hex-File für das Projekt wurde in PSoC Express erstellt und dann in das FirstTouch-Kit programmiert. Es zeigt sich, dass das Verhalten des realen Systems genau den Prognosen der Simulation entspricht, die wiederum exakt die zuvor festgelegten System-Spezifikationen widerspiegelt. Das Projekt kann also als Erfolg verbucht werden, zumal der gesamte Prozess auch von einem Anfänger in wenigen Stunden absolviert werden konnte. Erfahrene PSoC Express Anwender benötigen sogar noch weniger Zeit.

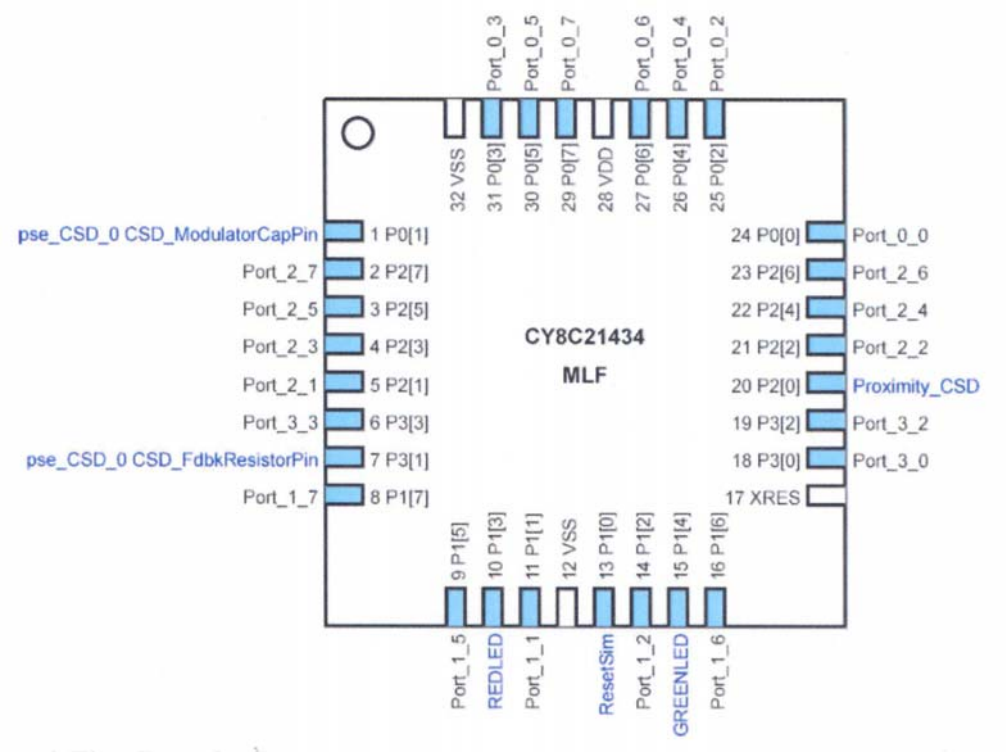


Bild 10. Pinbelegung für das erste Projekt mit dem FirstTouch-Entwicklungs-Board.

Schritt 6: Feinabstimmung des CapSense-Elements

Ein von der Simulation nicht abgedecktes Feature ist die Feinabstimmung des CapSense-Sensorblocks. Bild 11 und 12 zeigen die Betriebsarten CSD und Proximity, die sich bestens für das FirstTouch-Kit eignen. Weitere Informationen über diese beiden Betriebsarten enthält die Applikationsschrift ‚CapSense Best Practices' von Cypress Semiconductor [3].

Properties - CSDProperties

Name	CSDProperties
Driver	Properties - CSD
Cypress Certified	yes
Version	1.1.0.2
NoiseThreshold	10
NegativeNoiseThres	20
BaselineUpdateThre	200
Hysteresis	0
Debounce	3
LowBaselineReset	50
Sensors Autoreset	Enabled

Bild 11. Eigenschaften von CSD Properties

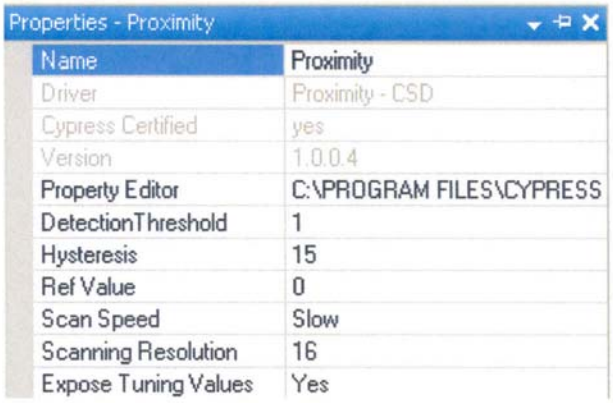


Bild 12. Eigenschaften von Proximity Sensor

Wie geht es weiter?

Es ist denkbar, dass ein Forscher Bilder ausschließlich von solchen Tieren wünscht, die sich dem Zweig mit dem Auslösedraht nähern, ohne ihn zu berühren, während Tiere, die den Zweig berühren, ignoriert werden sollen. Eine auf dem Zweig entlang kriechende Schlange oder Schnecke könnte beispielsweise den Näherungssensor triggern, doch sollen in dieser Anwendung keine solchen Lebewesen abgelichtet werden. Um ausschließlich Tiere zu fotografieren, die sich dem Näherungssensor nähern, ihn aber nicht berühren, müssen lediglich weitere Berührungssensoren entlang des Zweigs platziert werden. Um die entsprechende Änderung in PSoC Express vorzunehmen, sind CapSense-Tasten aus dem Treiberkatalog zu holen, um anschließend die Logik zum Einschalten der grünen LED zu verändern (‚Ein' bei Proximity_status > 0 und Button_status < 1). Eine wirklich intelligente Kamerafalle.

Dies war eine kurze Einführung in PSoC Express. Detaillierte Hilfestellung finden Sie online unter www.cypress.com. Ein guter Ausgangspunkt für fundiertere Studien ist außerdem die Cypress-Applikationsschrift AN2261 (Primer on PSoC Express) [5].

Literaturhinweise

[1] Will Rizzo, "Return of the Jaguar?", Smithsonian Magazine, December 2005

[2] Shaoni Bhattacharya, "Mystery Mammal Discovered in Borneo's Forests", NewScientist.com News Service, December 6, 2005

[3] Application Note AN2394, "CapSense Best Practices", Cypress Semiconductor

[4] Application Note AN2292, "Layout Guidelines for PSoC CapSense", Cypress Semiconductor

[5] Application Note AN2261, "Design Aids - PSoC Express™ 2.0 Primer: First Introduction", Cypress Semiconductor

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