The playground is a publicly-editable wiki about Arduino.

Manuals and Curriculum
Installing Arduino on Linux
Board Setup and Configuration
Development Tools
Interfacing With Hardware

- Output
- Input
- User Interface
- Storage
- Communication
- Power supplies
- General

Interfacing with Software
User Code Library

- Snippets and Sketches
- Libraries
- Tutorials

Suggestions & Bugs
Electronics Technique
Sources for Electronic Parts
Related Hardware and Initiatives
Arduino People/Groups & Sites
Exhibition
Project Ideas
Languages

PARTICIPATE

- Suggestions
- Formatting guidelines
- All recent changes
- PmWiki
- WikiSandBox training
- Basic Editing
- Cookbook (addons)
- Documentation index

edit SideBar

Capacitive Sensing Library

by Paul Badger

Download

Download CapSense04.zip

Download CapacitiveSense003.zip

Overview

The capSense library turns two or more Arduino pins into a capacitive sensor, which can sense the electrical capacitance of the human body. All the sensor setup requires is a medium to high value resistor and a piece of wire and a small (to large) piece of aluminum foil on the end. At its most sensitive, the sensor will start to sense a hand or body inches away from the sensor.

Version 04 adds support for Arduino 1.0, and fixes an obscure possible race condition with Tone, Servo and other libraries that perform I/O in interrupt context.

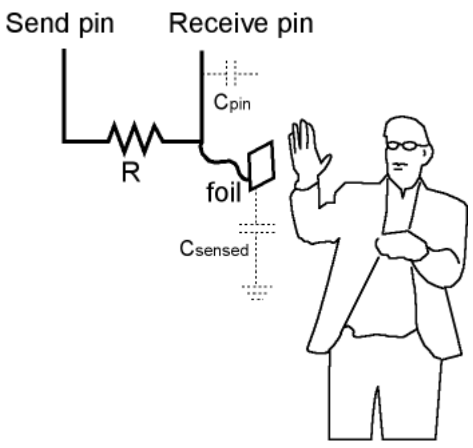
Version 03 has been updated to C++ and supports multiple inputs. It also includes some utility functions to make it convenient to change timeout values.

Applications

Capacitive sensing may be used in any place where low to no force human touch sensing is desirable. An Arduino and the library may be used to sense human touch through more than a quarter of an inch of plastic, wood, ceramic or other insulating material (not any kind of metal though), enabling the sensor to be completely visually concealed.

A capacitive sensor covered with paper or other insulator also acts as fairly good (human touch) pressure sensor with an approximately logarithmic response. In this regard it may surpass force sensing resistors in some applications.

How it works



The capSense method toggles a microcontroller *send* pin to a new state and then waits for the *receive* pin to change to the same state as the send pin. A variable is incremented inside a *while* loop to time the receive pin's state change. The method then reports the variable's value, which is in arbitrary units.

Watch a short video demonstration (YouTube)

The physical setup includes a medium to high value (100 kilohm - 50 megohm) resistor between the send pin and the receive (sensor) pin. The receive pin is the sensor terminal. A wire connected to this pin with a piece of foil at the end makes a good sensor. For many applications, a more useful range of values is obtained if the sensor is covered with paper, plastic, or another insulating material, so that users do not actually touch the metal foil. Research has shown that a small capacitor (100 pF) or so from sensor pin to ground improves stability and repeatability.

When the send pin changes state, it will eventually change the state of the receive pin. The delay between the send pin changing and the receive pin changing is determined by an RC time constant, defined by R * C, where R is the value of the resistor and C is the capacitance at the receive pin, plus any other capacitance (e.g. human body interaction) present at the sensor (receive) pin. Adding small capacitor (20 - 400 pF) in parallel with the body capacitance, is highly desirable too, as it stabilizes the sensed readings.

Library Methods

The library contains three main methods and some utility methods:

CapSense CapSense(byte sendPin, byte receivePin)

CapSense creates an instance of the library (please note the capital letters, this is not the same method as below)

long capSenseRaw(byte samples)

capSenseRaw requires one parameter, *samples*, and returns a long integer containing the absolute capacitance, in arbitrary units. The samples parameter can be used to increase the returned resolution, at the expense of slower performance. The returned value is not averaged over the number of samples, and the total value is reported.

capSenseRaw will return -2 if the capacitance value exceeds the value of CS_Timeout_Millis (in milliseconds). The default value for CS_Timeout_Millis 2000 milliseconds (2 seconds).

long capSense(byte samples)

capSense requires one parameter, samples, and returns a long containing the added (sensed) capacitance, in arbitrary units. capSense keeps track of the lowest baseline (unsensed) capacitance, and subtracts that from the sensed capacitance, so it should report a low value in the unsensed condition.

The baseline is value is re-calibrated at intervals determined by CS_Autocal_Millis. The default value is 200000 milliseconds (20 seconds). This re-calibration may be turned off by setting CS_Autocal_Millis to a high value with the set_CS_AutocaL_Millis() method.

void set_CS_Timeout_Millis(unsigned long timeout_millis);

The set_CS_Timeout_Millis method may be used to set the CS_Timeout_Millis value, which determines how long the method will take to timeout, if the receive (sense) pin fails to toggle in the same direction as the send pin. A timeout is neccessary because a *while* loop will lock-up a sketch unless a timeout is provided. CS_Timeout_Millis' default value is 2000 milliseconds (2 seconds).

void reset_CS_AutoCal()

reset_CS_AutoCal may be used to force an immediate calibration of capSense function.

void set_CS_AutocaL_Millis(unsigned long autoCal_millis)

The method set_CS_AutocaL_Millis(unsigned long autoCal_millis) may be used to set the timeout interval of the capSense function. Re-calibration may be turned off by using set_CS_AutocaL_Millis to set CS_AutocaL_Millis to "0xFFFFFFFF".

Resistor Choice

Here are some guidelines for resistors but be sure to experiment for a desired response.

- Use a 1 megohm resistor (or less maybe) for absolute touch to activate.
- With a 10 megohm resistor the sensor will start to respond 4-6 inches away.
- With a 40 megohm resistor the sensor will start to respond 12-24 inches away (dependent on the foil size). Common resistor sizes usually end at 10 megohm so you may have to solder four 10 megohm resistors end to end.

- One tradeoff with larger resistors is that the sensor's increased sensitivity means that it is slower. Also if the sensor is exposed metal, it is possible that the send pin will never be able to force a change in the receive (sensor) pin, and the sensor will timeout.

- Also experiment with small capacitors (100 pF - .01 uF) to ground, on the sense pin. They improve stability of the sensor.

Note that the hardware can be set up with one sPin and several resistors and rPin's for calls to various capacitive sensors. See the example sketch.

Grounding and other known issues

The grounding of the Arduino board is very important in capacitive sensing. The board needs to have some connection to ground, even if this is not a low-impedance path such as a wire attached to a water pipe.

Capacitive sensing has some quirks with laptops unconnected to mains power. The laptop itself tends to become sensitive and bringing a hand near the laptop will change the returned values.

Connecting the charging cord to the laptop will usually be enough to get things working correctly. Connecting the Arduino ground to an earth ground (for example, a water pipe) could be another solution.

Another solution that seems to have worked well on at least one installation, is to run a foil ground plane under the sensor foil (insulated by plastic, paper, etc.), and connected by a wire to ground. This worked really well to stabilize sensor values and also seemed to dramatically increase sensor sensitivity.

Scroll Wheels (well, slide pots anyway)

Experiments with a slide pot type linear sensor have been successful with just two pins and a resistance ladder. The basic layout is shown in the Quantum Scrollwheel sensor datasheet.

The code uses this type of arrangement

```
CapSense Left32  = CapSense(3, 2); // wire from pin 2 to left side of resistor ladder\
CapSense Right23 = CapSense(2, 3); // wire from pin 3 to right side of resistor ladder
```

Where the pins switch their send and receive positions. With a linear resistance ladder, a finger closer to the send pin will report lower values because resistance downstream from the capacitance is basically out of the circuit.

So in this manner when a finger is moved from one pin to the other the two calls to capSenseRaw will report complementary values that have an approximately constant value to them. The complication comes in when trying to deal with how much contact (capacitance) is present, which raises (or lowers) both values, but not necessarily in a linear manner.

At some point I'll get the sketch posted here.

Error Messages

capSense and capSenseRaw will return -1 with an invalid choice of pin parameter, but it appears that this feature is not working at this writing. Engineers are working on this, stand by...

capSense and capSenseRaw will return -2 if the methods timeout. This is caused by the count exceeding the value of CS_Timeout_Millis, which is set at a default value of 2000 milliseconds (2 seconds). This is most often caused by a missing resistor or the resistor in the wrong pin. It could also be caused by a sensor that is grounded or connected to +5 V.

A timeout is necessary because the *while* loop that does the timing in the CapSense method, will lock-up the sketch (the function will never return) if, for example, the resistor between sendPin and receivePin becomes disconnected.

Installation

- Download CapacitiveSense003.zip
- Unzip, and add to Arduino/hardware/libraries/
- To add capSense to a new sketch choose Sketch->Import Library->CapSense

Demo Sketch

```
#include <CapSense.h>

/*
 * CapitiveSense Library Demo Sketch
 * Paul Badger 2008
 * Uses a high value resistor e.g. 10 megohm between send pin and receive pin
 * Resistor effects sensitivity, experiment with values, 50 kilohm - 50 megohm. Larger resistor values yield lar
 * Receive pin is the sensor pin - try different amounts of foil/metal on this pin
 * Best results are obtained if sensor foil and wire is covered with an insulator such as paper or plastic sheet
 */


CapSense   cs_4_2 = CapSense(4,2);        // 10 megohm resistor between pins 4 & 2, pin 2 is sensor pin, add wir
CapSense   cs_4_5 = CapSense(4,5);        // 10 megohm resistor between pins 4 & 6, pin 6 is sensor pin, add wir
CapSense   cs_4_8 = CapSense(4,8);        // 10 megohm resistor between pins 4 & 8, pin 8 is sensor pin, add wir

void setup()
{

   cs_4_2.set_CS_AutocaL_Millis(0xFFFFFFFF);     // turn off autocalibrate on channel 1 - just as an example
   Serial.begin(9600);

}

void loop()
{
    long start = millis();
    long total1 =  cs_4_2.capSense(30);
    long total2 =  cs_4_5.capSense(30);
    long total3 =  cs_4_8.capSense(30);

    Serial.print(millis() - start);        // check on performance in milliseconds
    Serial.print("\t");                    // tab character for debug windown spacing

    Serial.print(total1);                  // print sensor output 1
    Serial.print("\t");
    Serial.print(total2);                  // print sensor output 2
    Serial.print("\t");
    Serial.println(total3);                // print sensor output 3

    delay(10);                             // arbitrary delay to limit data to serial port
}
```

Bugs, suggestions, applications

Please post bugs, suggestions, amazing feats on this forum thread

Reference/Libraries

Share |