# Discussion of State Machines in Sensact Version 3

## Introduction

This document is being written to explore the use of states in the triggers of the Sensact system, which was introduced in version 3 of the software.

The use of states allows for the creation of “state machines” otherwise known as “menu systems”. There are good and bad aspects of using states. The state mechanism introduced in version 3 allows for great flexibility. The system can emulate anything done in previous versions and can do much more. On the other hand, it is difficult to do anything – even fairly basic things – without some understanding of state transitions. Even conceptually simple menu systems require a complex design of states and transitions that will be beyond our target user community (i.e. medical staff responsible for configuration).

The state system introduced in version 3 is an experiment and a demonstration of what might be possible. The challenge going forward is to determine which capabilities we want to maintain and how to present them in a robust and easy-to-use package.

## Notation

The state diagrams in this document make use of a number of symbols which are defined here.

1.  A cyan circle with a number in it indicates a state.
2.  A yellow box indicates an action by the user. The actions cause state transitions. The letter in the box can be either ‘L’ for “low” or ‘H’ for “high”. This is not an indication of electrical activity, but rather of user activity. ‘H’ indicates that the user is doing something – pressing a button, swiping over a sensor etc. ‘L’ indicates the user is at rest.  
   It is also possible for a plus sign to appear in the box:  This indicates that the activity must continue for some time (unspecified) before the resulting actions and/or state change will trigger. Both ‘L+’ and ‘H+’ are used at times.
3.  The magenta diamond indicates an action that will occur. Actions are numbered. In addition to ‘A’ in the diamond an ‘M’ may be used. This indicates that a menu item will be presented to the user (this could be done visually or aurally)

## Simple Systems

The most basic system is simply one where the user activates the system and something happens. There is no state transition. The state diagram for this system would be:



### Refractory Time

Version 1 contains the notion of a refractory time. This is a mechanism which will prevent an action from repeating too rapidly. This can be important for users who may not be able to smoothly control the press and release of a button.

We could (and perhaps should) hard-code this into the system, it is also possible to create a simple configurable two-state system which accomplishes the same thing.



Here the user’s activity causes action A1 and a transition to state 2. The action cannot be repeated until there is a return to state 1, and this can only be accomplished by a period of inactivity.

### Two-Action Button

Version 2 introduced the idea of a two-action button. One action would occur when the user pressed a button and released quickly. A different action would occur if the button was pressed and held for a time.

(Note: I am referring to the pressing of a button, but this could of course be activation of the system by a variety of mechanisms. I will continue to refer to “pressing a button” for simplicity – always meaning “activation by whatever means is available).

Version 2 hard-coded the state transitions (and the timing) necessary to make these two-actions possible. Version 3 can do the same thing using states. The state transitions look like this.



Activity by the user causes an immediate transition to state 2. If the user then releases the button action 1 is performed. One the other hand, if the user wants action 2, he holds the button for a while and waits until it happens.

This fairly simple system still requires three states and four triggers.

## Transitions and Bifucations

The following diagram illustrates a system that will not work.



The intent is that the user will press the button and then release to do action 1, or continue to press to do action 2. There is, however, no time factor applied to the transition from state 2 to state 3. Since we sample sensors 20 times a second the transition from state 1 to state 2 will be followed instantly (in the user’s perception) by the transition from state 2 to state 3.

It turns out that in order to split the flow of activity into two parts you need one branch which is taking on a change of activity and one branch which is taken after an existing activity is sustained. Thus the only two possibilities are this

 and this 

## A Basic Menu System

So how do we build a menu system?

Here is a simple two-action menu system. The user presses a button and holds it until the menu (what-ever it is) indicates the desired action. The user then releases the button and the action occurs.



This can, of course, easily be extended to have as many menus as you like. Two transitions are required per action – one to generate the menu alert and one for the action.

## A Menu System with Latching Actions

Another possibility is a system where the user presses and holds to select an option and then that option continues to be available on subsequent button presses. For example the user might transition to a “TV volume-up” action and want to activate “volume-up” several times without having to traverse the whole menu system each time.

The following diagram illustrates a state machine which can accomplish this.



This is getting a bit complicated. When the user releases the button after seeing (or hearing) a particular menu item the state machine goes to a state where a button press will cause an action and return to the same state – a simple loop. To return to the start of the menu the user releases the button and does nothing for a time. A menu item is needed to inform the user that the system has reset to state 1.

## An Alternate Menu

What if the user cannot hold a button – or the sensor system being used is one that will simply trigger and then turn off again. A menu system that requires the holding of an active state will not work for such a user. Can a menu system be built without using any ‘H+’ boxes.

Of course! Here it is.



The user clicks the button to start the menu selection. The menu continues to display until the user clicks again. Then the appropriate action is taken. A period of inactivity then returns the system to the initial state.

It is also possible to do a menu system of this sort in which the actions “latch” and can be repeated – but I will spare you another diagram.

## The Toggle

Sometimes it is useful to have a button which does two things, and toggles back and forth between the two functions. This can be achieved using the follow set of state transitions.



When a button is pressed the system goes to state 2. On release it continues to state 3 so that the next time the button is pressed it goes to state 4.

Of course we need to add some actions.



These will usually be complementary actions – like mouse up and mouse down. Press the button and the mouse goes up. Release the button and press again and the mouse goes down. With two buttons (one up/down and one left/right) you can get full mouse control.

You can, of course, create a system which steps through multiple actions. You could have each action be changing to a specific TV channel. Then, with one button a user could flip through his favourite channels.

## Doing Multiple Actions in One State

Prior to discussing multiple actions it would be good to discuss how triggers are evaluated. There are two possible ways to evaluate triggers.

### 1. Maintain state for each pass

Triggers are evaluated in the order they are listed on the screen. One approach to handling states is to do all the actions which apply to the state that was in effect at the start of the pass. If some trigger asks for a state change then you change to that state only after all triggers have been evaluated. If two triggers ask for different state changes – well, you pick one – but it indicates that the logic is messed up.

### 2. Change state as you go

Another way of evaluating triggers is to change state as you pass through the list of triggers. With this type of evaluation you can potentially go through a number of state transitions in one evaluation pass.

After some experimentation SensAct has been set to use the “Change state as you go” method. Let us examine the subtle differences in how these function.

For the following tables assume what is shown is a list of all the triggers that should be active according to the current signal level.

**Example 1**

|  |  |  |
| --- | --- | --- |
| Starting State | Action | Ending State |
| 1 | Action 1 | 2 |
| 1 | Action 2 | 1 |

In a “Maintain state for each pass” system Action 1 and Action 2 will both happen and at the end of the evaluation the state will be changed to 2.

In a “Change state as you go” system Action 1 will be performed and the state will change to 2 before the next trigger is evaluated. As a result Action 2 will never happen.

**Example 2**

|  |  |  |
| --- | --- | --- |
| Starting State | Action | Ending State |
| 1 | Action 1 | 2 |
| 2 | Action 2 | 3 |
| 3 | Action 3 | 4 |

In a “Maintain state for each pass” system Action 1 will be performed on the first pass and the state will be changed to 2. During the next pass Action 2 will be performed and the state will be changed to 3. On the third pass Action 3 will be performed. If you re-order the trigger list it will work the same way.

In a “Change state as you go” system all three actions will be performed in one pass, as the state is changed from 1 to 2 to 3 and finally to 4. If you re-order the trigger list it may still work, but it will take multiple passes to complete.

In reality there is not much noticeable difference between these. SensAct does about 100 trigger evaluation passes each second. A delay of a few 100ths of a second is not perceptible by the user.

### Handling Multiple Actions

At times you may want multiple actions in response to a trigger. Here are some examples:

. activating the call bell and sounding the buzzer

. changing the state of both axis of a joystick.

. sending the digits of a channel-change request via the IR

Unfortunately you can only associate a trigger with a single action, so you need multiple triggers for multiple actions.

If what you want are simultaneous actions – like this:



the triggers table should look like this.

|  |  |  |
| --- | --- | --- |
| Starting State | Action | Ending State |
| 1 | Action 1 | 2 |
| 2 | Action 2 | 2 |
| 2 | Action 3 | 3 |

Note that the state change is made on the last trigger in the set of actions.

You may need a bit of time between actions (e.g. when sending channel digits to a TV).

In that case you will need more states and your trigger list will be:

|  |  |  |  |
| --- | --- | --- | --- |
| Starting State | Delay | Action | Ending State |
| 1 | 0 | Action 1 | 2 |
| 2 | 100 ms | Action 2 | 3 |
| 3 | 100 ms | Action 3 | 4 |

Because of the delays these triggers will be evaluated is several passes.

This is pretty rough still. Sorry if it is confusing. I just wanted to get some of this down.