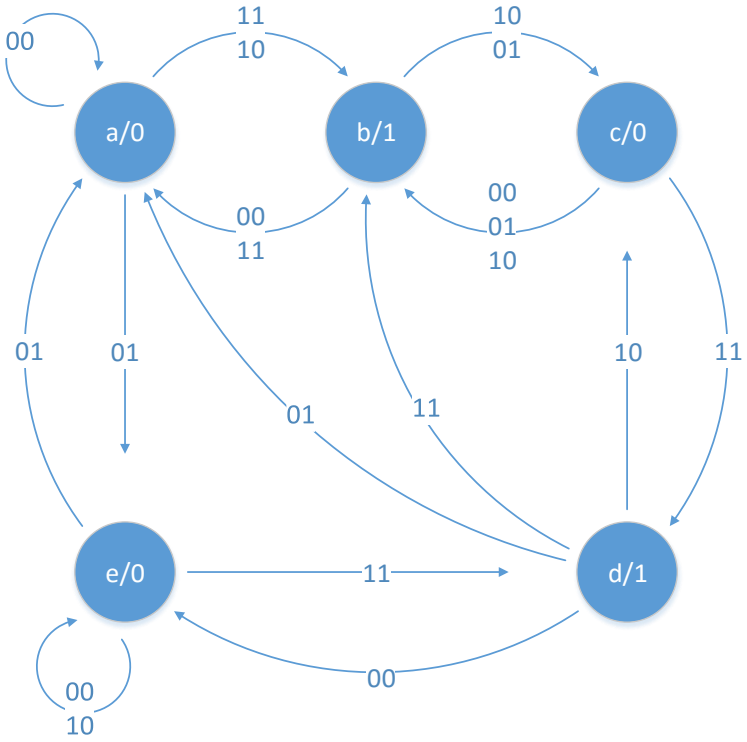
Finite State Machine

Finite state machine (FSM) will hold at a stable state at any time. One stable state corresponds to a specific output. The number of states is finite, and the transitions between states are specified.A input change will cause a transition. So once you know its current state and next input, you can always predict its next state, hence next output. Here is an example.



This is a state diagram of an FSM. This machine has 5 states, a 2-bit-wide input and a 1-bit-wide output. So there are 2^(width of input)=2^2=4 arrows coming out from each state. Each arrow stands for a possible transition caused by certain input. Assume the machine is currently at b state. So its output is 1. Now input 11, its state will jump to a. Its output will jump to 0. This behavior is called a transition.

Hardware implementation would take circuit delay into account, but in terms of software implementation, transition happens in no time.

Given its current state b, and a 2-bit-wide input sequence, corresponding transition and output will be,

Input		00	11	10	11	00	00	01
State	b(initial)	a	b	c	d	e	e	a
Output	1	0	1	0	1	0	0	0

Packet Detector

Assume there are 8 bytes per packet. A normal packet looks like this,

0	1	1	1	1	1	1	0

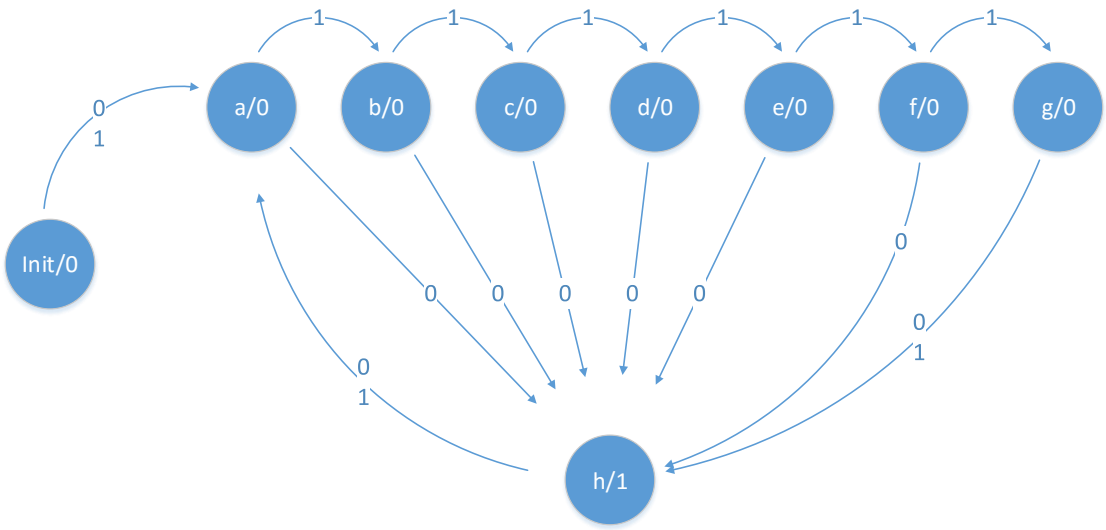
0 indicates 0x00, 1 indicates any value other than 0x00. An actual packet consists of 24 bytes, its structure is similar to the packet shown above. So take the 8-bit packet as an example to save some drawing work.

A byte stream might look like this, leftmost byte arrived first.

0	1	1	1	1	1	1	0	0	1	1	1	0

The question is, how to detect packets and divide this byte stream into packet stream. A byte might be lost, duplicated, altered from 1 to 0 or from 0 to 1, etc. An unreliable packet detector might corrupt all the following packets when certain byte error occurs.

Design an FSM to work as a packet detector. The machine takes 1-bit-wide input, gives a 1-bit-wide output. Feed the byte stream to this machine one by one. When the machine outputs 0, copy this byte into a packet buffer following the order of arrival. When the machine outputs 1, copy and push packet buffer into a packet stream, clear packet buffer, and wait for the next output from the machine.



This is the FSM I designed. State Init is the initial state. So when the byte stream shown above is feed to the FSM,

Input		0	1	1	1	1	1	1	0	0	1	1	1	0
State	Init	a	b	c	d	e	f	g	h	a	b	c	d	h
Output	0	0	0	0	0	0	0	0	1	0	0	0	0	1

The packet stream will be,

01111110
0110

The second packet is apparently an error packet. Leave it to protocol to handle. Notice that this error will not affect following good packets. There are other kinds of possible errors. See test case and test result for details. Test cases simulate some possible errors that might occur to 24-bit long packets. Test subject is the actual FSM used in ground station packet detector. Test result shows that an error packet, no

matter the type of its error, will only corrupt itself and the packet following it, at most. This FSM structure might not be the optimal structure in terms of data integrity, but at least it can make sure that any error will not corrupt more than a limited amount of following packets.