Finite State Machine with Arduino

Session 3 (24/01/14)

A finite state machine describes the different states that a system should adopt according to predefined conditions. State machines are used in every automation application, ranging from simple vending machines to very complex processes.

The main tool used to make a state machine in C is called a switch/case statement.

Like **if** statements, **switch...case** controls the flow of programs by allowing programmers to specify different code that should be executed in various conditions. In particular, a switch statement compares the value of a variable to the values specified in case statements. When a case statement is found whose value matches that of the variable, the code in that case statement is run.

The **break** keyword exits the switch statement, and is typically used at the end of each case. Without a break statement, the switch statement will continue executing the following expressions ("falling-through") until a break, or the end of the switch statement is reached.

Example

```
  switch (var) {
    case 1:
      //do something when var equals 1
      break;
    case 2:
      //do something when var equals 2
      break;
    default:
      // if nothing else matches, do the default
      // default is optional
  }
```

Syntax

```
switch (var) {
  case label:
    // statements
    break;
  case label:
    // statements
    break;
  default:
    // statements
}
```

Parameters

var: the variable whose value to compare to the various cases

label: a value to compare the variable to

In the *sketch*, define 4 constants that represent the 4 possible states:

```
#define STATE_IDLE  0
#define STATE_1     1
#define STATE_2     2
#define STATE_3     3
```

and a variable to store the current state:

```
int fsm_state;
```

In the setup() it’s very important to define the **initial state**:

```
fsm_state = STATE_IDLE;
```

In the loop(), use the Switch *statement* to identify the actual state, perform the operations and check if the conditions are good for a change of state:

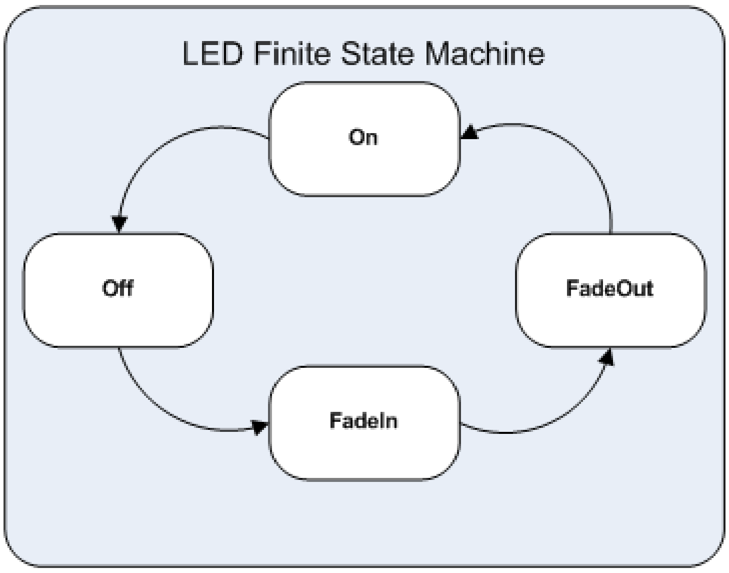
```
#include <TinkerKit.h>    // include the TinkerKit library
#define DEBUG 1
#define STATE_IDLE  0
#define STATE_1     1
#define STATE_2     2
#define STATE_3     3

#define TIME_DELAY 3000 //sets delay value in all delay() calls

int fsm_state; // our state variable

void setup() {
  Serial.begin(9600); // start serial communication at 9600 bps:
   if(DEBUG){
   Serial.println("Serial communication started");
   }
   fsm_state = STATE_IDLE; // state variable initialization
}

void loop()
{

  switch(fsm_state) { // start of switch case

   case STATE_IDLE:

  //do instructions in STATE_IDLE
     delay(TIME_DELAY);
   if(DEBUG){
   Serial.println("I am IDLE ");
   }
  fsm_state = STATE_1;

  break;

  case STATE_1:

  //do instructions in STATE_1
     delay(TIME_DELAY);
   if(DEBUG){
   Serial.println("I am in STATE_1 ");
   }
  fsm_state = STATE_2;

  break;

  case STATE_2:

  //do instructions in STATE_2
     delay(TIME_DELAY);
   if(DEBUG){
   Serial.println("I am in STATE_2 ");
   }
  fsm_state = STATE_3;

  break;

  case STATE_3:

  //do instructions in STATE_3
     delay(TIME_DELAY);
   if(DEBUG){
   Serial.println("I am in STATE_3 ");
   }
  fsm_state = STATE_IDLE;

  break;

  }//end of switch case
}// end of loop()
```

Observe the code and notice how the state variable fsm_state influences the messages being displayed. You have just started your first state machine!

Notice how

Exercise 1:

Create a programme which controls the brightness of an LED according to the following state machine:



Exercise 2:

Add another state to your state machine, in this state, the Arduino controller will be IDLE and wait for you to send the message “start” via serial. Once this message is received your Arduino will start the same sequence as the previous exercise. The state which follows the IDLE state will be the FadeIn state. There should be a 2 second delay before the transition from one state to another.

The LED sequence should run indefinitely until you press the reset button on the Arduino.

Exercise 3:

Modify the code in each of your states so that you can stop the LED sequence when you press the push button once.

References:

http://arduino.cc/en/Reference/SwitchCase

http://playground.arduino.cc/Code/FiniteStateMachine#Example