Simple project of Light and switch automation with CAPL

Requirements:

If the switch is turned **on** (sys_switch = 1), the light should be turned **on**.
If the switch is turned **off** (sys_switch = 0), the light should be turned **off**.
The light's status should accurately reflect the switch state, and the system should pass if the light behaves as expected for both states.

Test Case:

Test Steps:

1. Set switch = 1 to turn the light **on**.
2. Wait for 500ms.
3. Verify that the light is on by checking the value of switch.
4. Set switch = 0 to turn the light **off**.
5. Wait for 500ms.
6. Verify that the light is off by checking the value of switch.

Expected Results:

- After setting switch = 1, the light should be on (switch = 1).
- After setting switch = 0, the light should be off (switch = 0).

CAPL code for BCM node:

```
includes
{

}

variables
{
  message BCM_msg b;

}
on sysvar light_switch::sys_switch
{
  b.BCM_signal =@light_switch::sys_switch;

  output(b);
}
```

CAPL Automation for Switch_Light:

```
includes
{

}

variables
{
 int a,b;
}
```

Simple project of Light and switch automation with CAPL

```
void Maintest()
{
 testReportFileName("automation for light");
  testModuleTitle("Testing a light" );
  testModuleDescription("testing a light when switch triggered");
  testcase1();
}

testcase testcase1()
{
  testCaseTitle("TC1","light_chk_on_off_status");
  light_on();
  testWaitForTimeout(2000);
  light_off();
}

testfunction light_on()
{
  @light_switch::sys_switch=1;
  testWaitForTimeout(500);
  a= @light_switch::sys_switch;
  if(a==1)
  {
  testStepPass("TC1","light on");
  }
  else
  {
    testStepFail("TC1","light off");
  }
}

testfunction light_off()
{
  @light_switch::sys_switch=0;
  testWaitForTimeout(500);
  b=@light_switch::sys_switch;
  if(b==0)
  {
  testStepPass("TC1","light off");
  }
  else
  {
    testStepFail("TC1","light on");
  }
}
```