Chapter 8 State Machine

State machines (more precisely finite-state machines) provide a methodology to design, analyze, and formally communicate the functionalities of embedded systems. We will see a smart bike-light as an example in this chapter. Its functionalities are explained below and are already implemented in the source code. Study the source code together with the information given below to understand how it works.

1.1 Smart Light

The smart light has three user selectable modes:

- Manual off: The light will not emit light.
- Manual on: The light keeps blinking indefinitely. In reality, until the batteries are removed or discharged.
- Automatic: The light keeps blinking as long as the motion-sensing system reports that bike is moving. In reality, until the batteries are removed or discharged.

When in automatic mode, there will be actually two situations that are not directly user selectable:

- Automatic on: Motion-sensing system reports motion, light must be blinking.
- Automatic off: Motion-sensing system does not report motion, light must be kept off.

Another situation is battery power:

- Power up: The batteries are alright. They provide the system with required power.
- Power down: The batteries are discharged or removed by user.

Note that the system cannot tell these following two situations apart: (1) the batteries are discharged; (2) the batteries are removed by user.

1.2 Work Flow

Copy “smart light” from “skeleton” (/home/TDDI11/lab/skel) directory to your local directory (we assume: userID/TDDI11/smart_light)

```
cp -r /home/TDDI11/lab/skel/smart_light /home/userID/TDDI11
```

Please note that userID is the user name that you use to login to the lab computers. Now let us change to the new directory and check it:

```
cd /home/userID/TDDI11/smart_light
ls
```

Check to see if “smart_light” directory contains the following:

- main.c

The source code is in “main.c”.

1.2.1 The Source Code

The code implements and simulates the “smart light”. The simulation include the battery situations, therefore, the infinite loop that will run on the embedded platform can be broken by “removing” the batteries.

In the main body of code, in section 1, the time is read. In section 2, the mouse button status is read. In section 3, mouse motion is processed. A preprocessing similar to the one introduced in chapter 3 is utilized. Pay attention to the time-out mechanism, in combination with section 4. In section 4, some time-based operations for blinking and for motion-detection are performed. In section 5, states are updated. In section 6, outputs are generated. Pay attention that it must be assured that the light is not kept on needlessly. Otherwise the batteries must be replaced, unnecessarily, quite often. In section 7, blinking is simulated by writing “X” for the moment that light is switched on and by writing “_” for the moment that light is switched off. In the real-world, “light_on” will directly power up the lights through a driver[1].

Compile and Run

Compile and run the code with:

```
gcc main.c -lm -lX11
./a.out
```

1.3 Evaluation

1.3.1 Assignments

Formal description of the state machine: Draw the state diagram with required details as well as tables. Identify states and events/signals that control state transitions. Identify outputs, also consider the “printf”commands as outputs since they display relevant information.

(Optional) Add code to enable an ambient light sensor. When there is a lot of light (day-time), the light must be kept off. In low-light situation, the light must keep blinking. The sensor readouts must be preprocessed (low-pass filtering / averaging). Run the code and test it. Moreover, present the updated formal representation for the state machine.

1.3.2 Demonstrations

There is no demo, except for the optional assignment.

(Optional) Show the code and test it.

1.3.3 Deliverables

- Formal description of the state machine
- (Optional) Code for updated project with light-sensor
- (Optional) Formal description of the updated state machine
- Feedback questionnaire

Email them to your lab assistant. Write in the subject: TDDI11 Chapter 8.

[1] A driver is an electric circuit that translates the low-power signal of a processing unit to a high-power signal capable of powering up the lights.