Useless Machine

The software is makes use of a **state** machine. Correct use of a state machine guarantees predictable and safe behavior. This is accomplished by handling **events** which triggers an action, depending on the current state. Such action can be turning on a light, but an event can also cause a **transition** between states. See Wikipedia for more information.

For this project, two events are possible:

- SWTICH → when the user moves the switch lever up.
- TICK → can be seen as a clock pulse and is fired when no switch event took place.

For this project, five state are possible:

- IDLE → nothing happens, box is closed.
- OPENING → during this state, this lid will open whilst the arm moves in.
- PEEKING → during this state, the arm moves close to the switch.
- SWITCHING → during this state, the arm actually toggles the switch.
- CLOSING → during this state, the lid will be closes.

Tick mechanism

A mechanism of ticks is chosen to prevent blocking code and get accurate control over the servos. A single tick can be seen as a clock pulse.

SlowServo is a derived class from **Servo** with the ability to move the servo 1 degree per tick. After a number of ticks, the servo has reached its goal and a true will be returned.

The shutdown timer makes also use of ticks. On every tick, the shutdown counter will be increased by 1. When the counter reaches its limit, the state machine will exit.

Since the code is non-blocking, one can assume that the run time is close to 0. This means that the speed of the program is largely determined by the delay between ticks. Shorting the delay results in a faster program. Note: the delay may never be smaller than the time it takes for the servo to move 1 degree. This results in unpredictable behavior.

Decision making

What is more fun than a useless machine? A useless machine with character! To accomplish this, a decision-making algorithm is designed. Sounds fancy? It's just as simple as 'coin or head'. Given two possible options, a random one is picked. This can be influenced by weight; the lighter side of the coin had more chance to face upwards. To translate this to code: the default behavior of a useless machine is preferred but the program may deviate in a controlled manner. No spoilers here, build the useless machine and try it out!

State behavior

Every state accepts all events, though it may ignore one. The behavior on such event is described in the following section. See also Figure 1.

IDLE

- SWITCH
  - Reset switched boolean
  - Reset shutdown timer
  - **Transition** to OPENING
- TICK
  - Increase shutdown timer
  - On shutdown: move lid servo to off position

OPENING

- SWITCH: ignored
- TICK
  - Move lid servo to open position
  - Move arm servo to close position
  - When movements are done:
    - **Transition** to PEEKING & turn on LED (preferred when switch is still on) (decision)
    - **Transition** to CLOSING (proffered when switch is off)

CLOSING

- SWTICH
  - Reset switched state
  - Reset shutdown timer
  - **Transition** to OPENING
- TICK
  - Move lid servo to close position
  - When movement is done:
    - **Transition** to IDLE (preferred) (decision)
    - **Transition** to OPENING (mandatory when switch is still on)

PEEKING

- SWITCH: ignored
- TICK
  - Move arm servo to peek position. That is just before switching.
  - When movement is done:
    - **Transition** to SWITCHING (preferred) (decision)
    - **Transition** to OPENING (mandatory when switched is off)

SWITCHING

- SWITCH: ignored
- TICK
    - Move arm to switch position. Effectively toggling the switch.
    - When movement is done:
        - Set switched boolean
        - **Transition** to OPENING

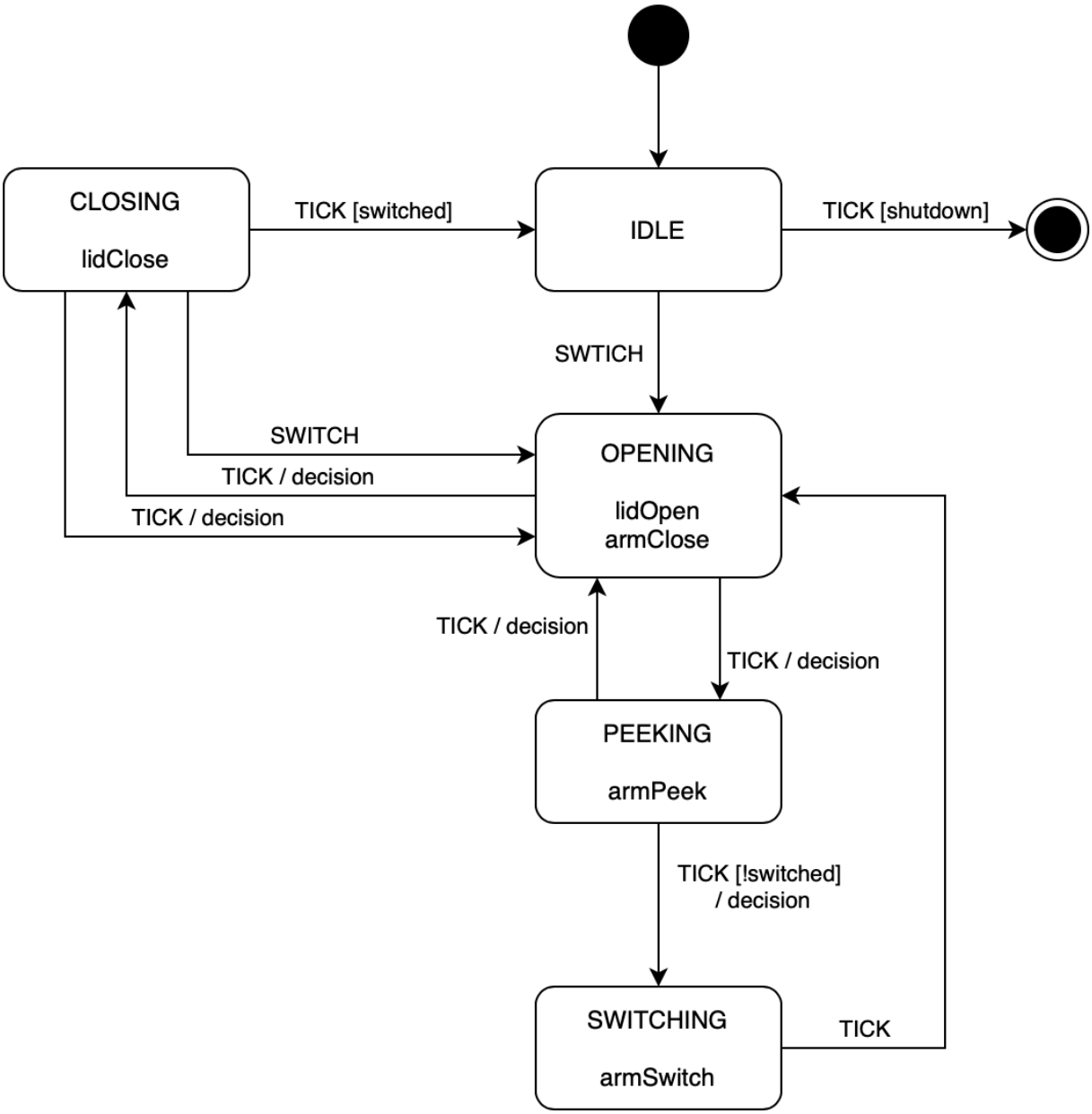

Figure 1: State machine diagram