소프트웨어 공학개론

-Code Presentation

Team 1

고수창
김동언
박종엽
이선엽

목차

1. 코드 구현 구조

Concept

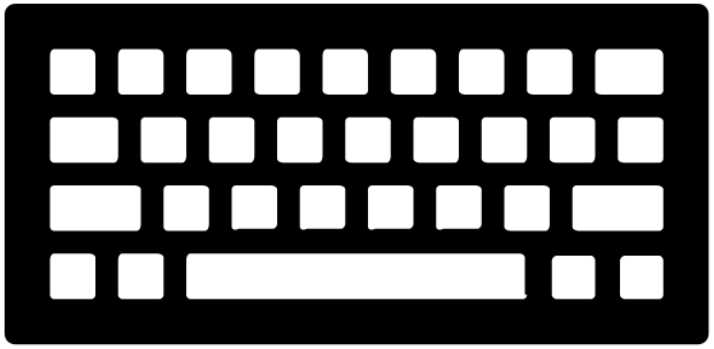
Button

Sensor

Data Process

1. 코드 구현 구조

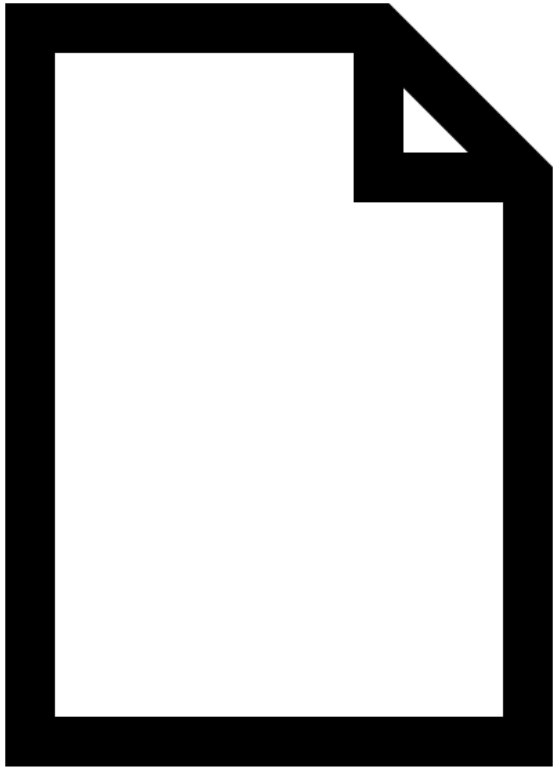
Button



Created by useiconic.com
from Noun Project

1. 코드 구현 구조

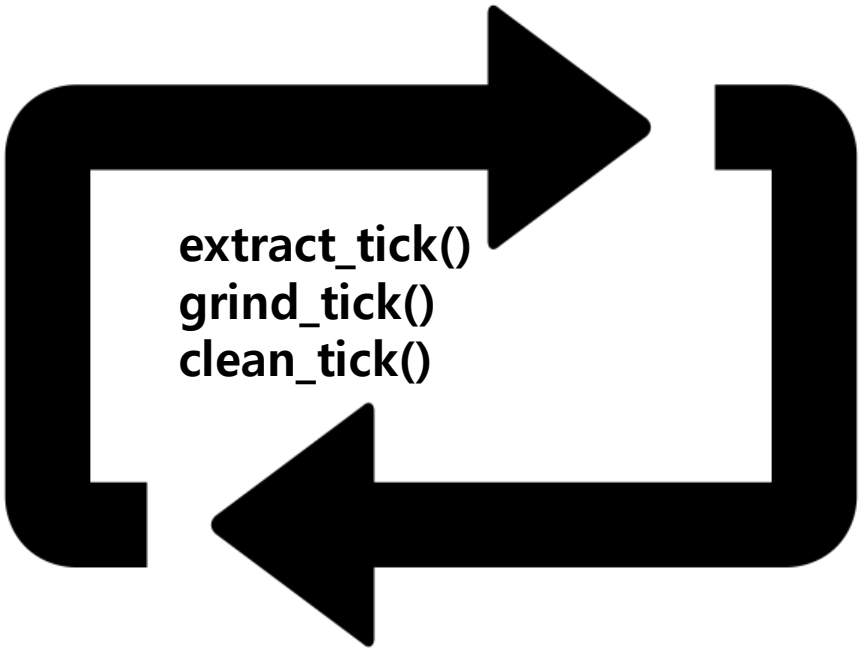
Sensor



Created by Galaxicon
from Noun Project

1. 코드 구현 구조

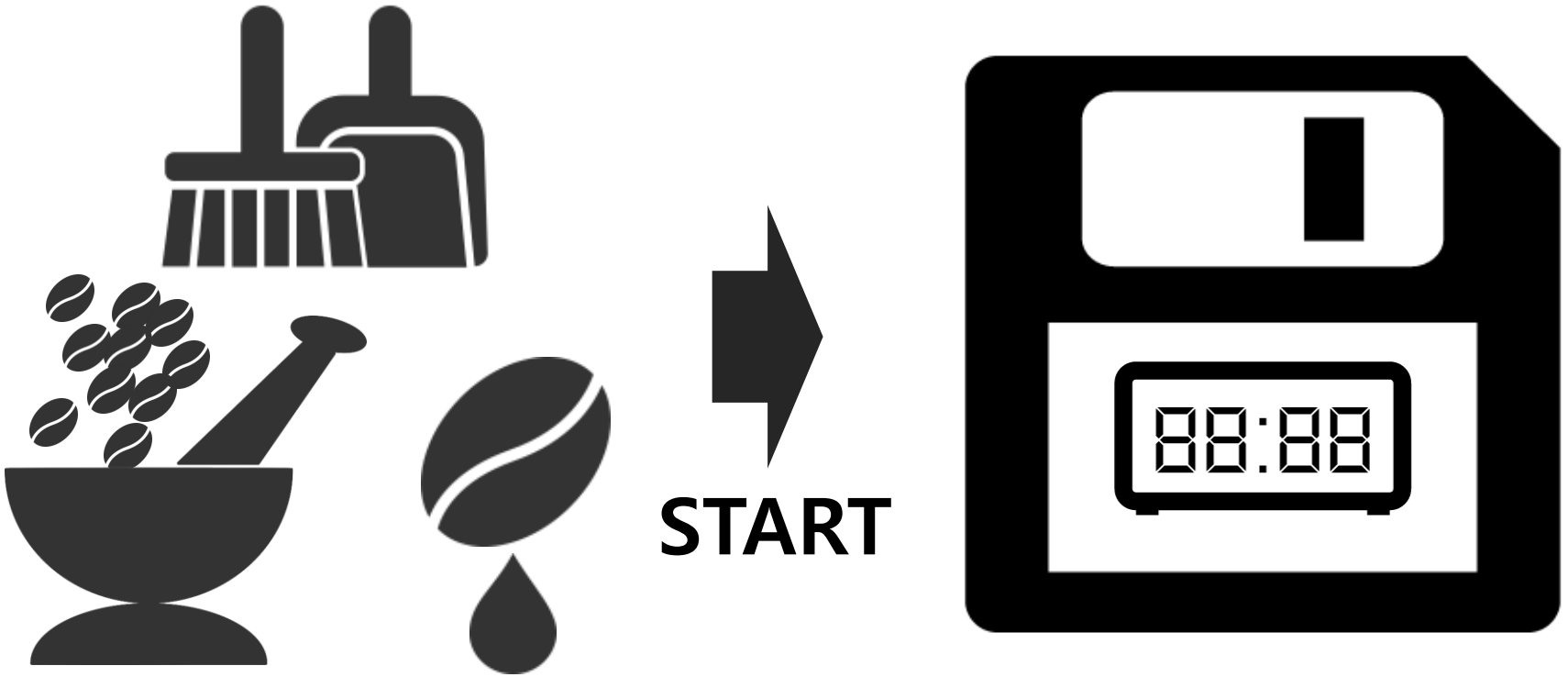
Data Process



Created by useiconic.com
from Noun Project

1. 코드 구현 구조

Sleeping



1. 코드 구현 구조

Sleeping

END

1. 코드 구현 구조

Consistent tick

One thread

Non-blocking

Well-ported

1. 코드 구현 구조

Consistent tick

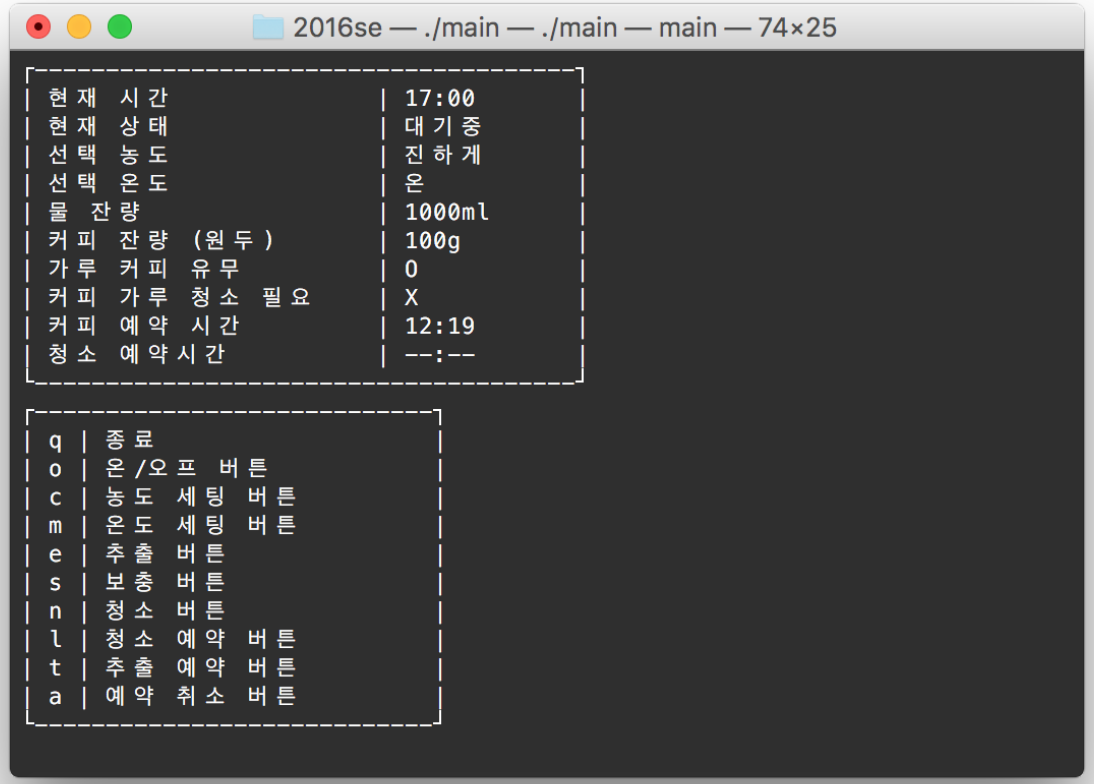
```
int mysleep_init(int *id) {
        *id = -1;
        return 0;
}

int mysleep(int *id, int duration) {
        if(*id == -1) {
                *id = mytime();
        } else if(mytime() - *id >= duration) {
                *id = -1;
                return 1;
        }

        return 0;
}
```

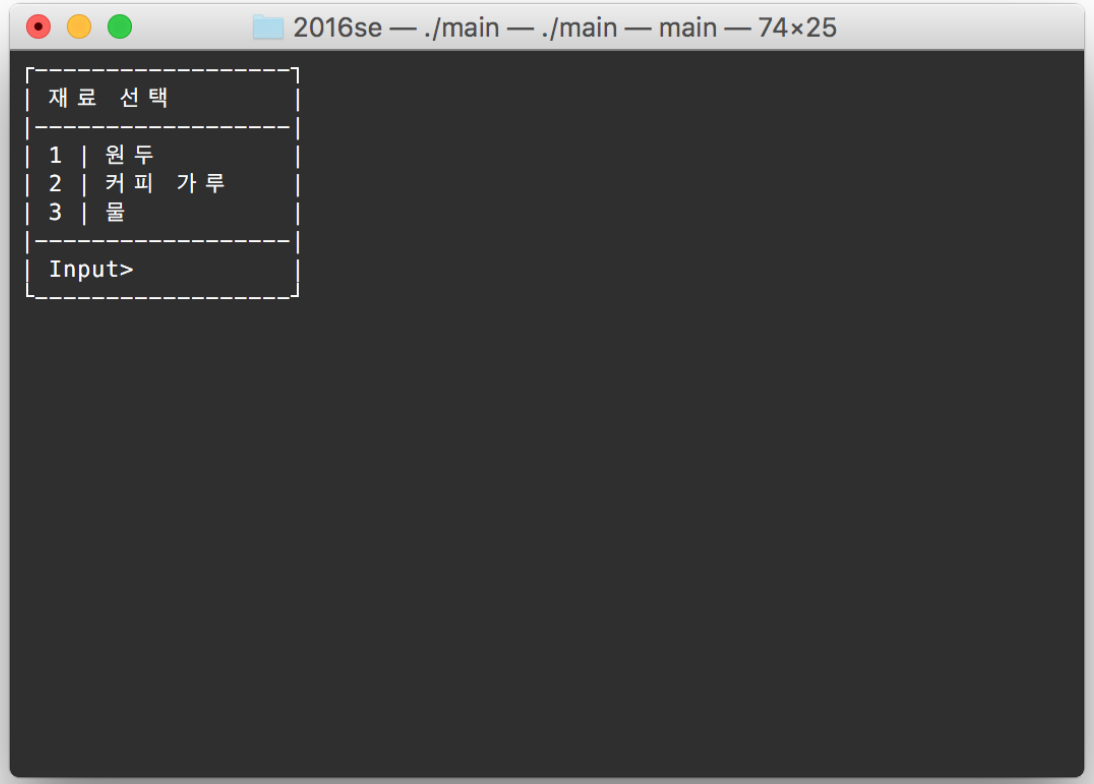
1. 코드 구현 구조

Feedback Module



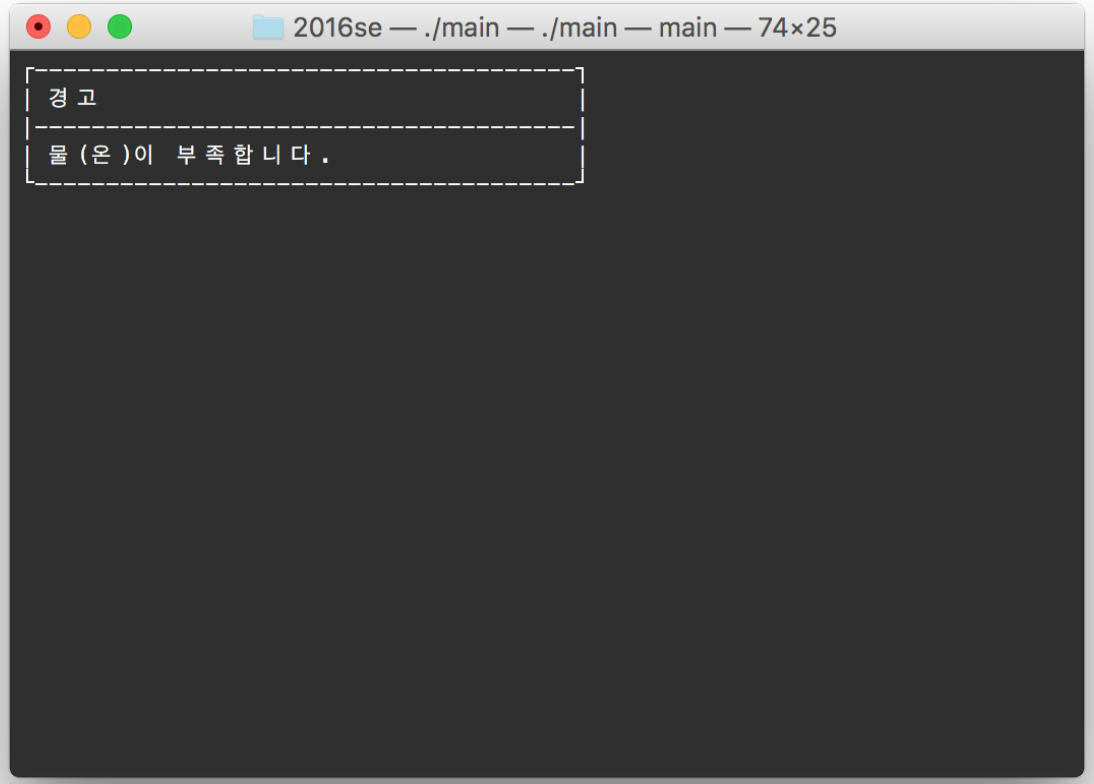
1. 코드 구현 구조

Feedback Module



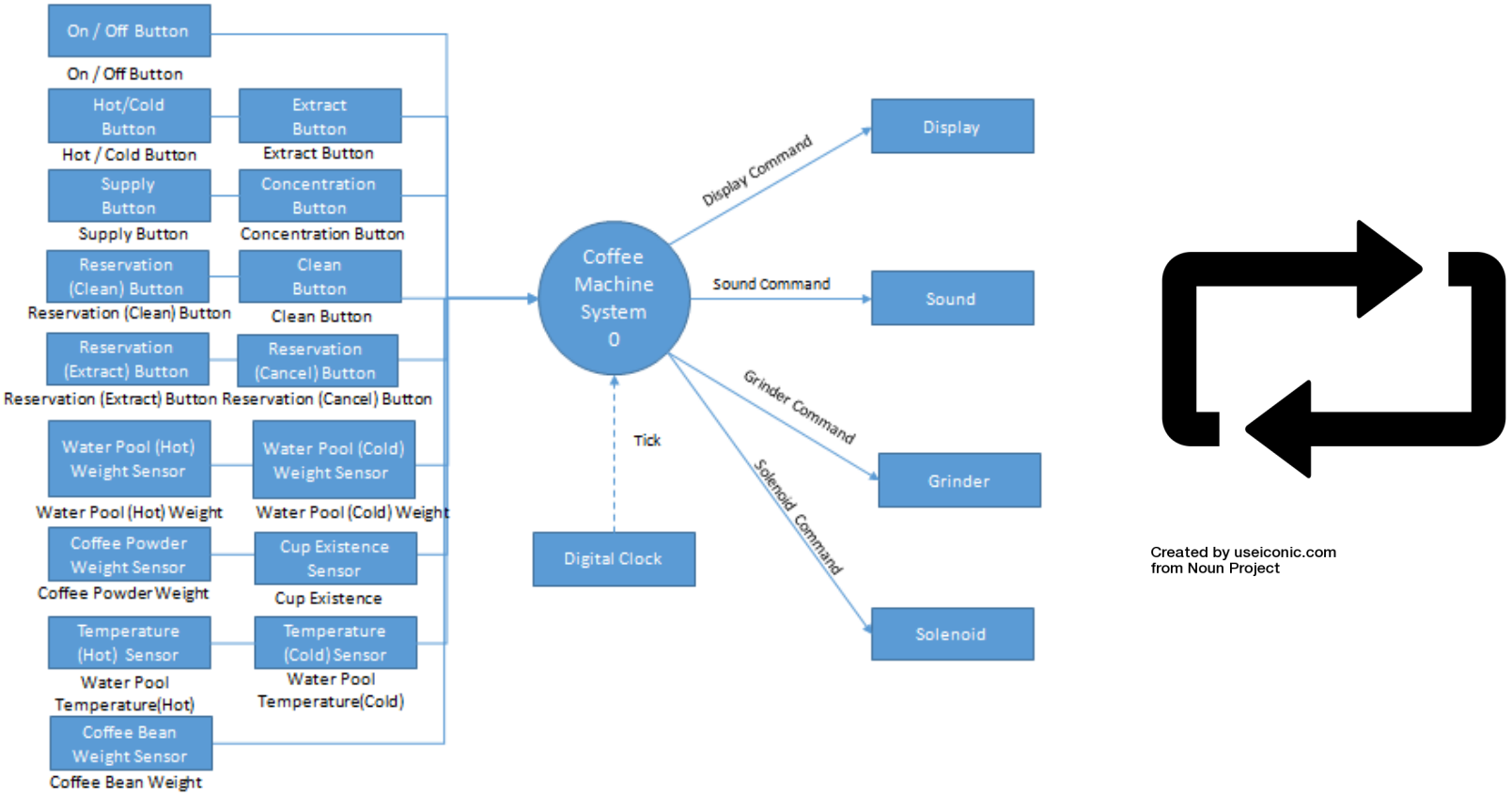
1. 코드 구현 구조

Feedback Module



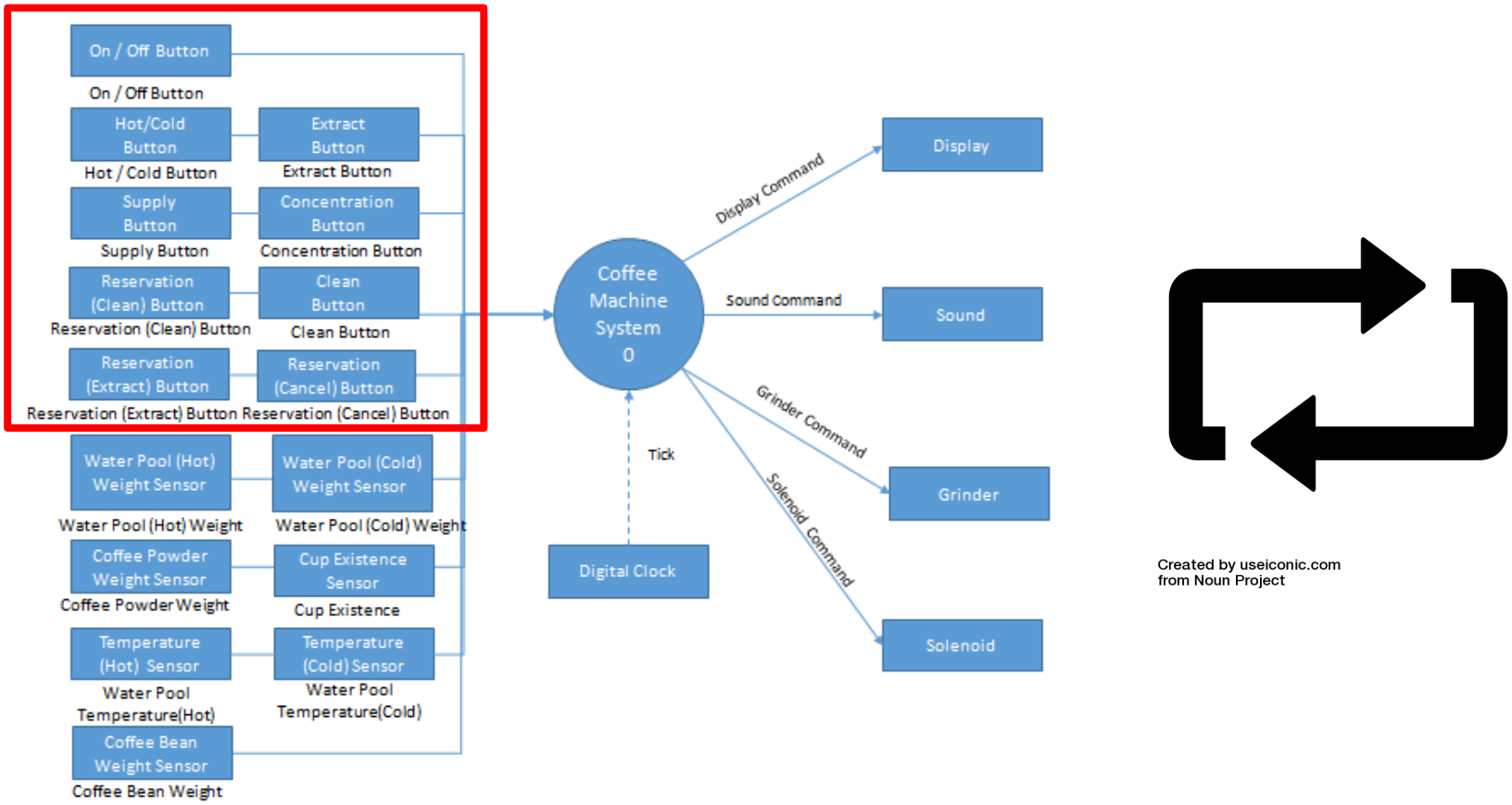
2. 코드 구현 상세

Concept



Created by useiconic.com
from Noun Project

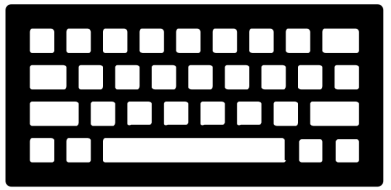
2. 코드 구현 상세 Button



Created by useiconic.com
from Noun Project

2. 코드 구현 상세 Button

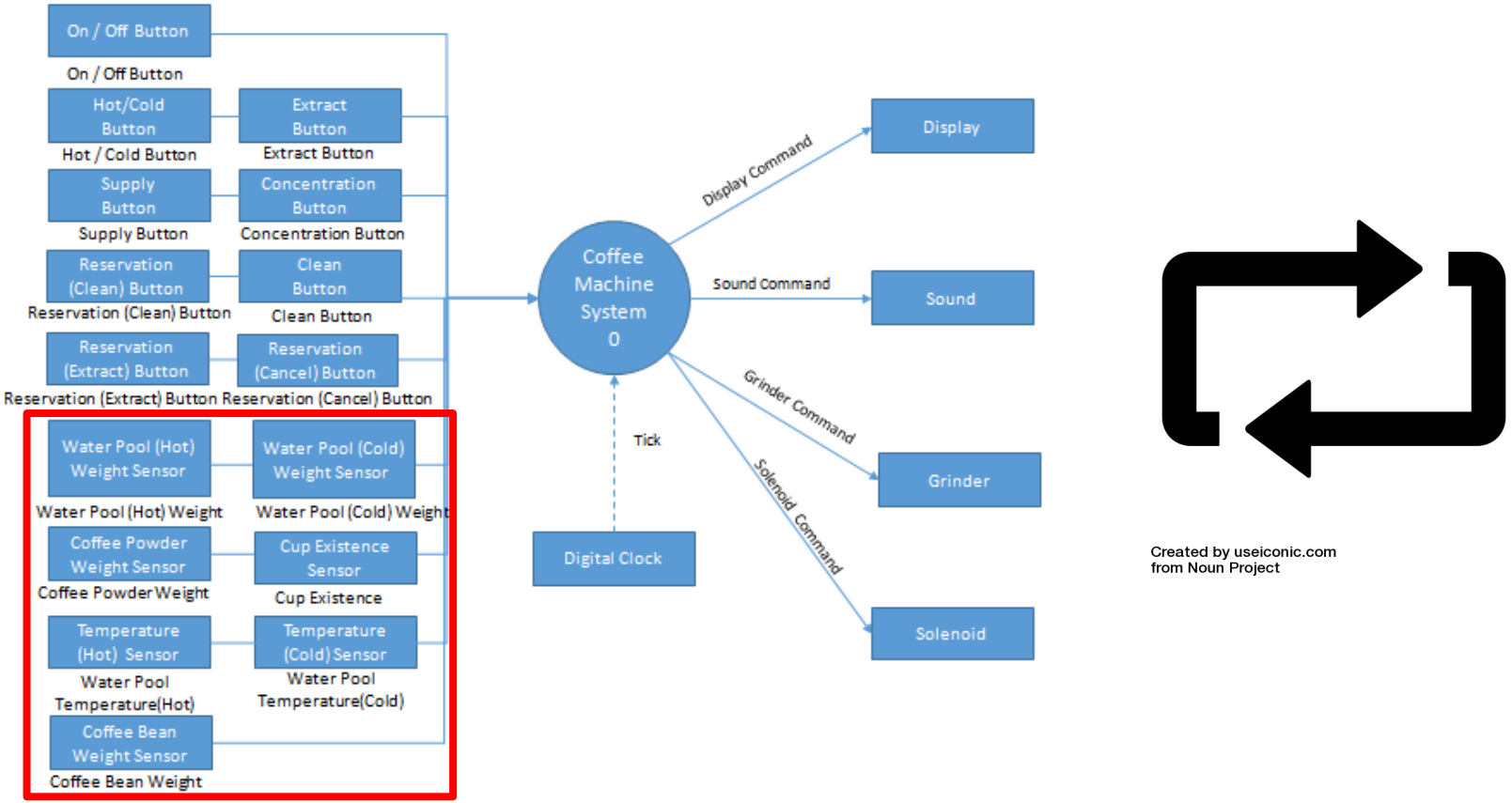
```
btn_init(&btn_temperature, 'm');
btn_init(&btn_onoff, 'o');
btn_init(&btn_extract, 'e');
btn_init(&btn_concentration, 'c');
btn_init(&btn_supply, 's');
btn_init(&btn_reservation_clean, 'l');
btn_init(&btn_reservation_extract, 't');
btn_init(&btn_reservation_cancel, 'a');
btn_init(&btn_clean, 'n');
```



Created by useiconic.com
from Noun Project

2. 코드 구현 상세

Sensor

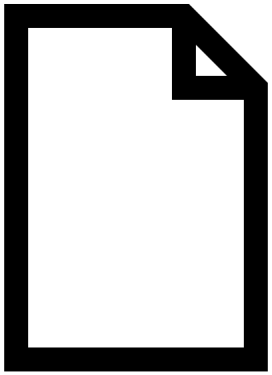


Created by useiconic.com
from Noun Project

2. 코드 구현 상세

Sensor

```
sensor_init(&sensor_hot_weight, "hot_weight.txt", 0, 500);
sensor_init(&sensor_cold_weight, "cold_weight.txt", 0, 500);
sensor_init(&sensor_cup_existence, "cup_existence.txt", 0, 1);
sensor_init(&sensor_coffee_bean_weight, "coffee_bean_weight.txt", 0, 100);
sensor_init(&sensor_coffee_powder_weight, "coffee_powder_weight.txt", 0, 100);
sensor_init(&sensor_hot_temperature, "coffee_water_hot_temperature.txt", -100,
                100);
sensor_init(&sensor_cold_temperature, "coffee_water_cold_temperature.txt", -100,
                100);
sensor_init(&sensor_use_count,"sensor_count.txt",0,10);
```



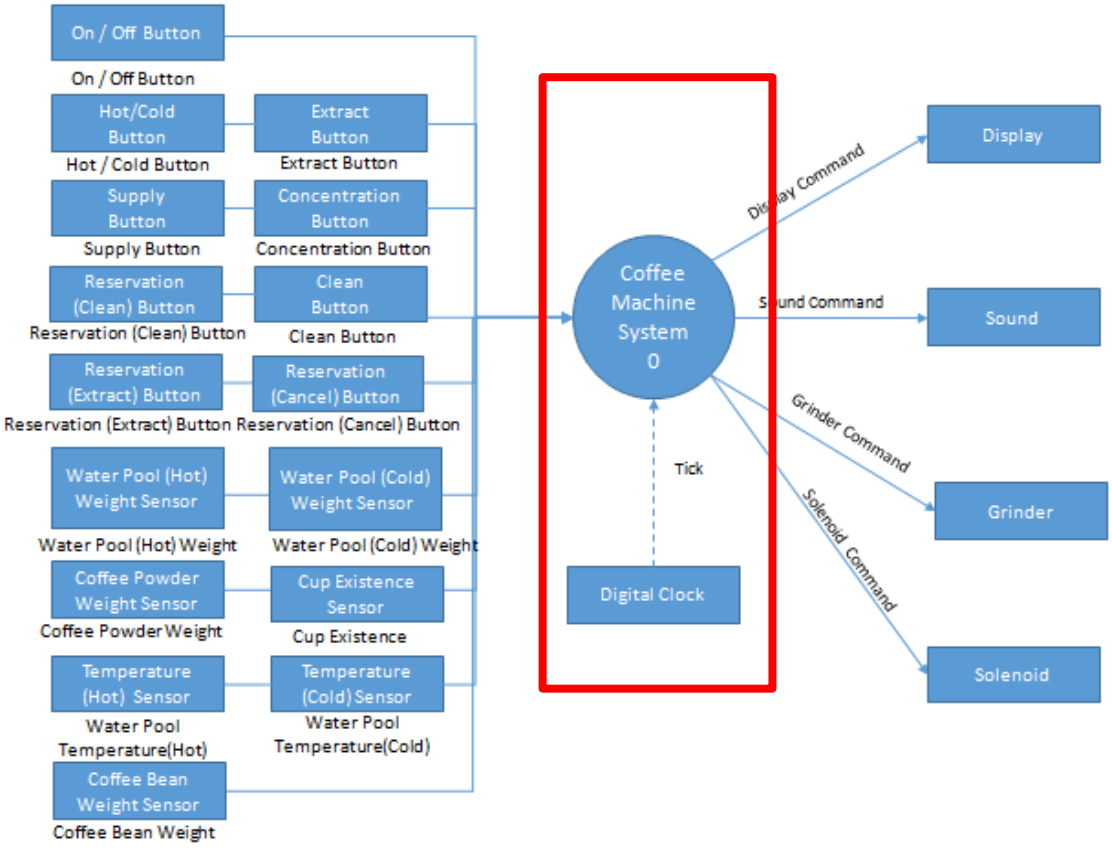
Created by Galaxicon
from Noun Project

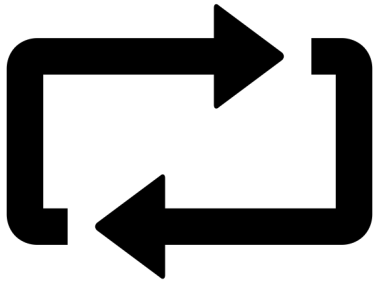
2. 코드 구현 상세 Structure

```
struct btn_ctx {
    char key;
    int pressed;
}
```

```
struct sensor_ctx {
    char * filename;
    int min;
    int max;
}
```

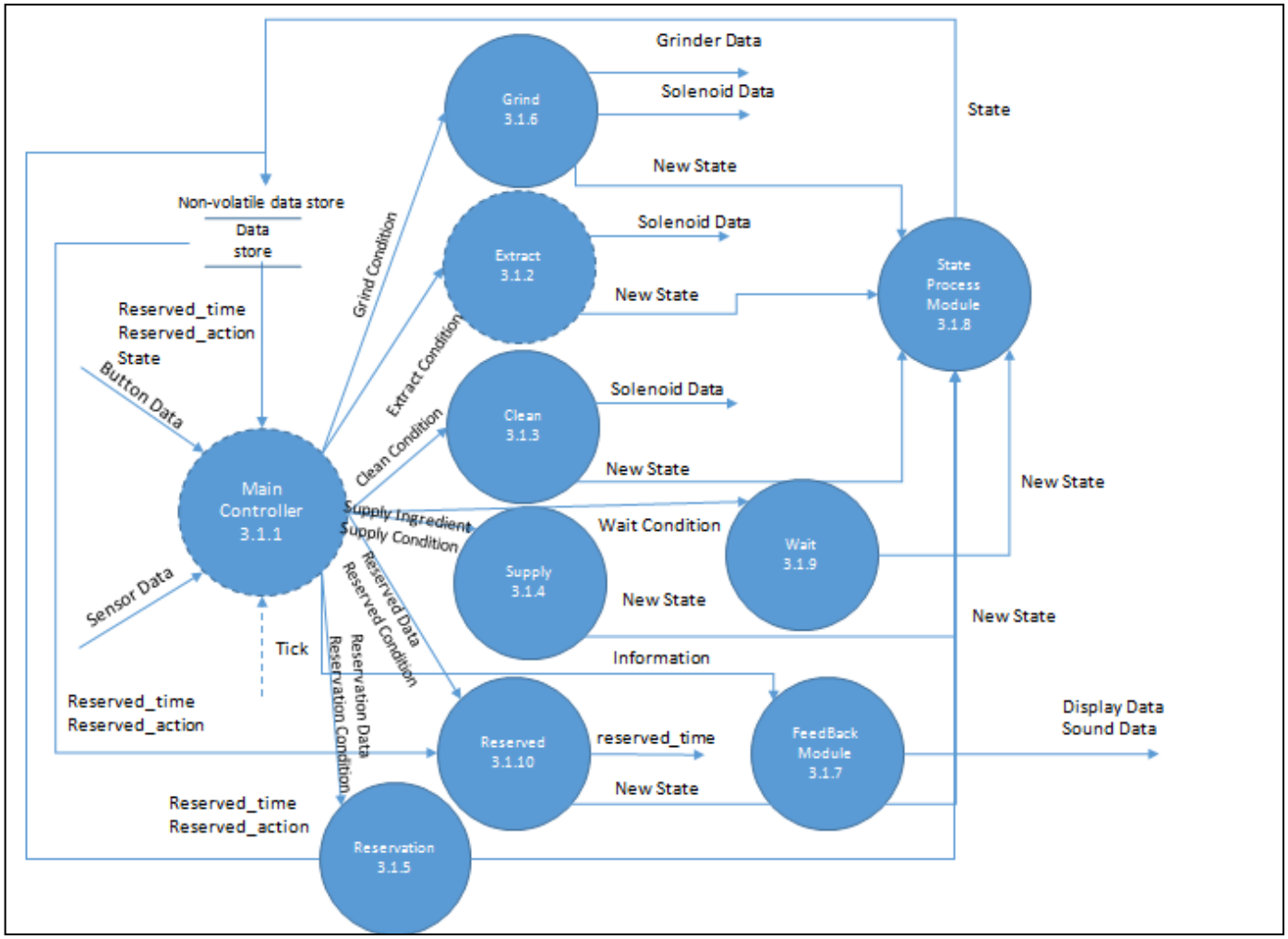
2. 코드 구현 상세 Tick





Created by useiconic.com
from Noun Project

2. 코드 구현 상세 Tick



2. 코드 구현 상세 Data Process

```
while (1) {
        print_state();
        btns_update();

        if(power_flag){
                wait_tick(state);
                grind_tick(state);
                extract_tick(state);
                reserve_tick(state);
                clean_tick(state);
                supply_tick(state);
                reserved_tick(state);
        }
}
```

2. 코드 구현 상세

State processing module

```
void btns_update() {
        char c = getch();
        if(c == btn_temperature.key){
                btn_press(&btn_temperature);
        }else if(c == btn_extract.key){
                btn_press(&btn_extract);
        }else if(c == btn_supply.key){
                btn_press(&btn_supply);
        }else if(c == btn_concentration.key){
                btn_press(&btn_concentration);
        }else if(c == btn_reservation_clean.key){
                btn_press(&btn_reservation_clean);
        }else if(c == btn_reservation_extract.key){
                btn_press(&btn_reservation_extract);
        }else if(c == btn_reservation_cancel.key){
                btn_press(&btn_reservation_cancel);
        }else if(c == btn_clean.key){
                btn_press(&btn_clean);
        }else if(c == btn_onoff.key){
                btn_press(&btn_onoff);
        } else {
                ungetch(c);
        }
        state_process();
}
```

```
void state_process(){
        if(btn_is_pressed(&btn_temperature)){
                temp_flag = !temp_flag;
                btn_release(&btn_temperature);
        }else if(btn_is_pressed(&btn_extract)){
                new_state(STATE_EXTRACT);
                btn_release(&btn_extract);
        }else if(btn_is_pressed(&btn_supply)){
                new_state(STATE_SUPPLY);
                btn_release(&btn_supply);

                supply_type = 0;
                supply_amount = 0;
        }else if(btn_is_pressed(&btn_concentration)){
                btn_release(&btn_concentration);
        }else if(btn_is_pressed(&btn_reservation_clean)){
                if(state == STATE_WAIT)
                    new_state(STATE_RESERVE);

                btn_release(&btn_reservation_clean);
        }else if(btn_is_pressed(&btn_reservation_extract)){
                if(state == STATE_WAIT)
                    new_state(STATE_RESERVE);

                btn_release(&btn_reservation_extract);
        }else if(btn_is_pressed(&btn_reservation_cancel)){
                if(state == STATE_WAIT)
                    new_state(STATE_RESERVE);

                btn_release(&btn_reservation_cancel);
        }else if(btn_is_pressed(&btn_clean)){
                new_state(STATE_CLEAN);
                btn_release(&btn_clean);
        }else if(btn_is_pressed(&btn_onoff)){
                power_flag = !power_flag;
                btn_release(&btn_onoff);
        } else {
                return;
        }

        werase(stdscr);
}
```

2. 코드 구현 상세 Grinder

```
void grind_tick(int now_state) {
        if(now_state == STATE_GRIND) {
                if (sensor_get(&sensor_coffee_bean_weight) < 10) {
                        // printf("Fail");
                        draw_warning(win, "원두가 부족합니다.", 1);
                        new_state(STATE_WAIT); // 대기 상태로 변경
                } else {
                        Solenoid_Command(On);
                        if(mysleep(&grind_timer, 3)) {
                                sensor_sub(&sensor_coffee_bean_weight,10); // 소모되는 원두의 양
                                sensor_add(&sensor_coffee_powder_weight,10);
                                Solenoid_Command(Off);
                                new_state(STATE_EXTRACT);
                        }
                }
        }
}
```

2. 코드 구현 상세 Extract

```
void extract_tick(int now_state) {
        int Count = sensor_get(&sensor_use_count);
        if (now_state == STATE_EXTRACT) {
                if (sensor_get(&sensor_coffee_powder_weight) < 10) {
                        new_state(STATE_GRIND);
                } else if (sensor_get(&sensor_coffee_powder_weight) >= 10) {
                        if(temp_flag == 0){ //Hot
                                if(sensor_get(&sensor_hot_weight) >= 300 && sensor_get(
                                        Solenoid_Command(On);

                                        if(mysleep(&timer, 3)) {
                                                Solenoid_Command(Off);
                                                new_state(STATE_WAIT);
                                                sensor_sub(&sensor_hot_weight,(concent
                                                sensor_sub(&sensor_coffee_powder_weight
                                                sensor_add(&sensor_use_count,1);
                                        }
```

3. Unit Test

CTEST

CTEST

ctest is a unit test framework for software written in C.

Features:

- adding tests with minimal hassle (no manual adding to suites or testlists!)
- supports suites of tests
- supports setup() teardown() per test
- output format not messed up when tests fail, so easy to parse.
- displays elapsed time, so you can keep your tests fast
- uses coloring for easy error recognition
- only use coloring if output goes to terminal (not file/process)
- it's small (a little over 300 lines of code!)
- it's easy to integrate (only 1 header file)
- has SKIP option to skip certain test (no commenting test out anymore)
- Linux + OS/X support

3. Unit Test

Code

```
CTEST(sensor_test, extract_test1) {
sensor_update(&sensor_hot_weight, 500);
sensor_update(&sensor_cup_existence, 1);
sensor_update(&sensor_coffee_powder_weight, 0);
new_state(STATE_EXTRACT);
extract_tick(state);
ASSERT_EQUAL(state, STATE_GRIND);
}

CTEST(sensor_test, extract_test2) {
sensor_update(&sensor_hot_weight, 200);
sensor_update(&sensor_coffee_powder_weight, 15);
new_state(STATE_EXTRACT);
extract_tick(state);
ASSERT_EQUAL(state, STATE_WAIT);
}

CTEST(sensor_test, extract_test3) {
sensor_update(&sensor_hot_weight, 500);
sensor_update(&sensor_cup_existence, 1);
sensor_update(&sensor_coffee_powder_weight, 10);
new_state(STATE_EXTRACT);
extract_tick(state);
ASSERT_EQUAL(state, STATE_EXTRACT);
}

extern int temp_flag;

CTEST(sensor_test, test4) {
sensor_update(&sensor_hot_weight, 500);
sensor_update(&sensor_cold_weight, 0);
sensor_update(&sensor_cup_existence, 1);
sensor_update(&sensor_coffee_powder_weight, 10);
temp_flag = 1;
```

• • •

```
int main(int argc, const char *argv[])
{
    sensor_init(&sensor_hot_weight, "hot_weight.test.txt", 0, 500);
    sensor_init(&sensor_cold_weight, "cold_weight.test.txt", 0, 500);
    sensor_init(&sensor_cup_existence, "cup_existence.test.txt", 0, 1);
    sensor_init(&sensor_coffee_bean_weight, "coffee_bean_weight.test.txt", 0, 100);
    sensor_init(&sensor_coffee_powder_weight, "coffee_powder_weight.test.txt", 0, 100);
    sensor_init(&sensor_hot_temperature, "coffee_water_hot_temperature.test.txt", -100, 100);
    sensor_init(&sensor_cold_temperature, "coffee_water_cold_temperature.test.txt", -100, 100);
    sensor_init(&sensor_use_count,"sensor_count.test.txt",0,10);

    int result = ctest_main(argc, argv);
    return result;
}
```

3. Unit Test

Result

```
[OK]
TEST 4/25 sensor_test:extract_test3 NORMAL
[OK]
TEST 5/25 sensor_test:test4 ERROR_COLD
[OK]
TEST 6/25 sensor_test:test5 ERROR_CUP
[OK]
TEST 7/25 sensor_test:test6 ERROR_COFFEE_BEAN
[OK]
TEST 8/25 sensor_test:test7 [OK]
TEST 9/25 sensor_test:test8 [OK]
TEST 10/25 sensor_test:test9 [OK]
TEST 11/25 sensor_test:test10 [OK]
TEST 12/25 sensor_test:test11 [OK]
TEST 13/25 sensor_test:test12 [OK]
TEST 14/25 sensor_test:test13 [OK]
TEST 15/25 sensor_test:test14 [OK]
TEST 16/25 sensor_test:test15 ERROR_SOLENOID
[OK]
TEST 17/25 sensor_test:test16 [OK]
TEST 18/25 sensor_test:test17 [OK]
TEST 19/25 sensor_test:test18 [OK]
TEST 20/25 sensor_test:test19 ERROR_GRIND
[OK]
TEST 21/25 sensor_test:test20 [OK]
TEST 22/25 sensor_test:test21 [OK]
TEST 23/25 sensor_test:test22 [OK]
TEST 24/25 sensor_test:test23 [OK]
TEST 25/25 sensor_test:test24 [OK]
RESULTS: 25 tests (25 ok, 0 failed, 0 skipped) ran in 117 ms

C:\Users\PJY\Desktop\2016se-master>
```