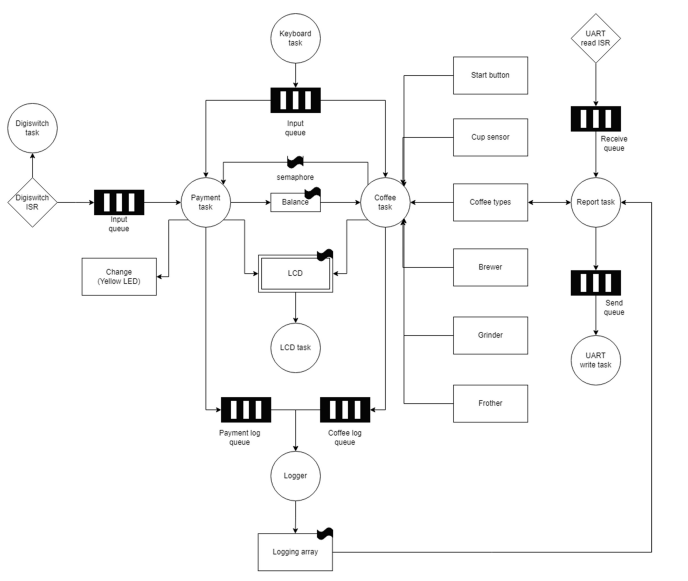
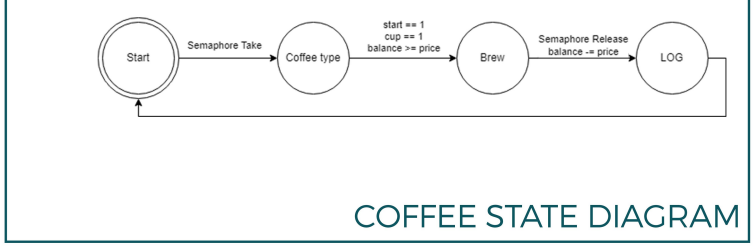
Bagas Aditya

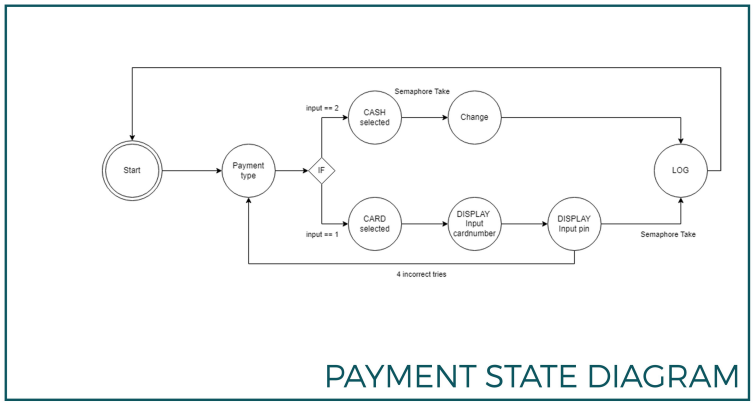
Titus Gilboa



TASK DIAGRAM



COFFEE STATE DIAGRAM



PAYMENT STATE DIAGRAM

```
/*************************  Variables  *************************/
//enum CoffeeTask_states CoffeeTask_state = Start;
COFFEE_TYPE coffee_types[COFFEE_TYPES_LENGTH];
SemaphoreHandle_t coffee_types_mutex;

COFFEE_TYPE current_coffee;

SemaphoreHandle_t active_semaphore;

extern TaskHandle_t payment_t;

extern INT8U balance;
extern SemaphoreHandle_t balance_mutex;
> #pragma region ...
void coffee_init()
{
    active_semaphore = xSemaphoreCreateBinary();
//    turn =xSemaphoreCreateMutex();
    configASSERT(active_semaphore);

    coffee_types_mutex = xSemaphoreCreateMutex();
    xSemaphoreGive(coffee_types_mutex);

    COFFEE_TYPE latte;
    latte.active = 1;
    strcpy(latte.name, "latte");
    latte.price = 20;
    latte.amount_pay = 0;
    latte.grind_time = 5.0f;
    latte.brew_time = 15.0f;
    latte.milk_time = 5.0f;
    coffee_types[1] = latte;

    COFFEE_TYPE espresso;
    espresso.active = 1;
    strcpy(espresso.name, "Espresso");
    espresso.price = 18;
    espresso.amount_pay = 0;
    espresso.grind_time = 8.0f;
    espresso.brew_time = 15.0f;
    espresso.milk_time = 0.0f;
    coffee_types[0] = espresso;

    COFFEE_TYPE filter_coffee;
    filter_coffee.active = 1;
    strcpy(filter_coffee.name, "Filter coffee");
    filter_coffee.price = 1;
    filter_coffee.amount_pay = 1;
    filter_coffee.grind_time = 0.0f;
    filter_coffee.brew_time = 0.0f;
    filter_coffee.milk_time = 0.0f;
    coffee_types[2] = filter_coffee;
}
```

```
void coffee_task(void *pvParameters)
{
    COFFEE_STATES current_state = SELECT_COFFEE;
    while (1)
    {
        #pragma region ...
        switch (current_state)
        {
        case SELECT_COFFEE:
            current_state = select_coffee_state();
            break;
        case BREW:
        #pragma region ...
            ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
            #pragma region ...
            current_state = brew_state();
            break;
        case C_LOG:
            log_coffee(&current_coffee);
            current_state = SELECT_COFFEE;
            break;

        }
    }
}
```

```
COFFEE_STATES select_coffee_state()
{
    lprintf(0, "Pick coffee");

    INT8U num = -1; // To start from 0

    while (1)
    {
        xSemaphoreTake(coffee_types_mutex, portMAX_DELAY);
        do
        {
            if (num < COFFEE_TYPES_LENGTH - 1) num++;
            else num = 0;
        }
        while (!coffee_types[num].active);

        lprintf(1, "%d-%s", num, coffee_types[num].name);

        INT8S inp = key2int(key_get(pdMS_TO_TICKS(COFFEE_CYCLE_TIME_MS)));
        if (inp != -1)
        {
            if (coffee_types[inp].active)
            {
                current_coffee = coffee_types[inp];

                lprintf(0, "price : %d", current_coffee.price);
                lprintf(1, "name  : %s", current_coffee.name);

                INT8U inp = key_get(portMAX_DELAY);
                if (key2int(inp) != -1)
                {
                    // move to payment
                    xTaskNotifyGive(payment_t);

                    xSemaphoreTake(active_semaphore, portMAX_DELAY);

                    xSemaphoreGive(coffee_types_mutex);
                    return BREW;
                }
            }
        }
        xSemaphoreGive(coffee_types_mutex);
    }
}
```

coffee.c

```
void log_task(void* pvParameters)
{
    COFFEE_TYPE coffee_type;
    INT8U payment_type[CARD_LENGTH];
//    INT8U* time[4];
    INT8U i;
    //int i;
    while (1)
    {
        xQueueReceive(log_coffee_q, &coffee_type, portMAX_DELAY);
        xQueueReceive(log_payment_q, &payment_type, portMAX_DELAY);
//        xQueueReceive(log_time_q, &time, portMAX_DELAY);

        xSemaphoreTake(log_array_semaphore, portMAX_DELAY);

        LOG_TYPE* log = log_nextlog();

        log->active = 1;
        log->price = coffee_type.price;

        // Convert payment to string
        if (payment_type[0] == CASH_ID)
        {
            strcpy(&log->payment_type, "CASH    ");
        }
        else
        {
            for (i = 0; i < CARD_LENGTH; i++)
            {
                log->payment_type[i] = payment_type[i] + '0';
            }
        }

        // Get coffee number
        xSemaphoreTake(coffee_types_mutex, portMAX_DELAY);
        INT8U number = -1;
        for (i = 0; i < COFFEE_TYPES_LENGTH; i++)
        {
            if (!strcmp(coffee_types[i].name, coffee_type.name))
            { // Names are equal
                number = i;
                break;
            }
        }
        xSemaphoreGive(coffee_types_mutex);

        configASSERT(number != -1);

        log->coffee_number = number;

        xSemaphoreGive(log_array_semaphore);
    }
}
```

```
void log_init()
{
    // Create queues
    log_coffee_q = xQueueCreate(LOG_INPUT_LENGTH, LOG_COFFEE_WIDTH);
    configASSERT(log_coffee_q);

    log_payment_q = xQueueCreate(LOG_INPUT_LENGTH, LOG_PAYMENT_WIDTH);
    configASSERT(log_payment_q);

    // Create semaphore for array
    log_array_semaphore = xSemaphoreCreateMutex();
    xSemaphoreGive(log_array_semaphore);
}
```

```
void log_coffee(COFFEE_TYPE* coffee)
{
    xQueueSendToBack(log_coffee_q, coffee, portMAX_DELAY);
}

void log_payment(INT8U* payment_type)
{
    xQueueSendToBack(log_payment_q, payment_type, portMAX_DELAY);
}

void log_time(INT8U* time)
{
    xQueueSendToBack(log_time_q, time, portMAX_DELAY);
}

LOG_TYPE* log_nextlog()
{
    int i;
    for(i = 0; i < LOG_LENGTH; i++)
    {
        if (!log_array[i].active)
        {
            return &log_array[i];
        }
    }
    return NULL;
}
```

log.c

```
/*****************************************************************
*   Input    : pvParameters
*   Output   : N/A
*   Function : Payment type machine
*****************************************************************/
void payment_task(void* pvParameters)
{
    PAYMENT_STATES current_state = START;

    configASSERT(xQueueAddToSet(ch_input_queue, cash_set_q) == pdPASS);
    configASSERT(xQueueAddToSet(active_semaphore, cash_set_q) == pdPASS);
    xSemaphoreGive(active_semaphore);

    while (1)
    {

        switch (current_state)
        {
        case START:
            ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
//            xSemaphoreGive(active_semaphore);
            current_state = PAYMENT;
            break;
        case PAYMENT:
            current_state = paymenttype_state();
            break;
        case CARD_NUMBER:
            current_state = cardnumber_check_state();
            break;
        case PIN:
            current_state = pin_check_state();
            break;
        case CASH:
            current_state = cash_state();
            break;
        case CHANGE:
            current_state = change_state();
            break;
        case LOG:
            log_payment(cardnumber);
            current_state = START;
            break;
        }
    }
}
```

```
/*****************************************************************
*   Input    : N/A
*   Output   : Payment state
*   Function : Payment type state logic
*****************************************************************/
PAYMENT_STATES paymenttype_state()
{
    lprintf(0, "Payment method");
    lprintf(1, "%c Card | %c Cash", CARD_METHOD, CASH_METHOD);
    // Blocking input or cash
    //
    // Wait for key input getkey(portMAX_DELAY)
    while (1)
    {
        switch (key_get(portMAX_DELAY))
        {
        case CARD_METHOD:
            return CARD_NUMBER;

        case CASH_METHOD:
            return CASH;
        default:
            break;
        }
    }
}
```

```
/*****************************************************************
*   Input    : N/A
*   Output   : Payment state
*   Function : Cardnumber state logic / check for 8 digit card number
*****************************************************************/
PAYMENT_STATES cardnumber_check_state()
{
    lprintf(0, "Enter cardNumber");
    lprintf(1, "");// Reset bottom ready for writing digits

    INT8U digit_counter = 0;

    while (1)
    {
        if (digit_counter < CARD_LENGTH)
        {
            INT8U inp = key_get(portMAX_DELAY);
            if (key2int(inp) != -1)
            {
                cardnumber[digit_counter] = key2int(inp);
                digit_counter++;
                wr_ch_LCD(inp);
            }
        }
        else
        {
            return PIN;
        }
    }
}
```

```
/*****************************************************************
*   Input    : N/A
*   Output   : Payment state
*   Function : Cash state logic / update payment balance
*****************************************************************/
PAYMENT_STATES cash_state()
{
    xSemaphoreTake(balance_mutex, portMAX_DELAY);
    balance = 0;
    xSemaphoreGive(balance_mutex);

    lprintf(0, "Insert coins");

    xTaskNotifyGive(Coffee_t); // Start brew

    while (1)
    {
        xSemaphoreTake(balance_mutex, portMAX_DELAY);
        INT8S dir = 0;
        do
        {
            dir = digiswitch_get(5);

//            lprintf(1, "dir : %d", dir);
            switch (dir)
            {
            case 1:
                balance += CLOCKWISE;
                break;
            case -1:
                balance += COUNTERCLOCKWISE;
                break;
            }
        }
        while (dir != 0);

        lprintf(1, "Balance : %d", balance);

        xSemaphoreGive(balance_mutex);

        xQueueSelectFromSet(cash_set_q, portMAX_DELAY);

        if (xSemaphoreTake(active_semaphore, 0) == pdPASS)
        {
            xSemaphoreGive(active_semaphore);
            break;
        }
    }

    return CHANGE;
}
```

```
PAYMENT_STATES pin_check_state()
{
    INT8U pin[PIN_LENGTH];
    INT8U pin_counter = 0;
    INT8U attempts = 1;
    while (1)
    {
        if (pin_counter == 0)
        {
            lprintf(0, "Enter PIN : (%d/", CARD_MAX_ATTEMPTS - attempts + 1);
            lprintf(1, "");
        }
        INT8U inp = key_get(portMAX_DELAY);
        if (key2int(inp) != -1)
        {
            pin[pin_counter] = key2int(inp);
            pin_counter++;
            wr_ch_LCD("*");
        }
        else
        {
            if ((pin_counter == PIN_LENGTH)
                || (pin[0] == 0 && pin_counter == 1)
                || (pin[0] == 1 && pin_counter == 2)
                && (pin[pin_counter-1] == 0 || pin[pin_counter-1] == 1))
            {
                balance = DEFAULT_CARD;
                xTaskNotifyGive(coffee_t);
                lprintf(0, "");
                lprintf(1, "");

                while(1)
                {
                    // wait for brew
                    xQueueSelectFromSet(cash_set_q, portMAX_DELAY);
                    if (xSemaphoreTake(active_semaphore, 0) == pdPASS)
                    {
                        xSemaphoreGive(active_semaphore);
                        break;
                    }
                }

                xTaskNotifyGive(coffee_t); // Restart
                return LOG;
            }
            else
            {
                if (attempts < CARD_MAX_ATTEMPTS)
                {
                    lprintf(0, "Incorrect PIN");
                    lprintf(1, "Try again");
                    key_get(portMAX_DELAY);
                    pin_counter = 0;
                    attempts++;
                }
                else
                {
                    lprintf(0, "");// Reset bottom ready for writing digits
                    lprintf(1, "Rejected");
                    INT8U inp = key_get(portMAX_DELAY);
                    if (key2int(inp) != -1)
                    {
                        return PAYMENT;
                    }
                }
            }
        }
    }
}
```

```
INT8U cash_state()
{
    int i;
    ui_clear_screen();
    uprintf(buffer, "Coffee name  | Total sales\n\r");

    xSemaphoreTake(log_array_semaphore, portMAX_DELAY);
    xSemaphoreTake(coffee_types_mutex, portMAX_DELAY);

    for (i = 0; i < COFFEE_TYPES_LENGTH; i++)
    {
        if (coffee_types[i].active)
        {
            INT8U sum = 0;
            for (j = 0; j < LOG_LENGTH; j++)
            {
                if (log_array[j].active)
                {
                    if (i == log_array[j].coffee_number)
                    {
                        sum += log_array[j].price;
                        break;
                    }
                }
                else
                {
                    break;
                }
            }
            uprintf(buffer, "%-14s| %d\n\r", coffee_types[i].name, sum);
        }
        else
        {
            break;
        }
    }

    xSemaphoreGive(log_array_semaphore);
    xSemaphoreGive(coffee_types_mutex);

    uprintf(buffer, "Press any key to return\n\r");
    INT16U inp = 0;
    xQueueReceive(uart0_rx_queue, &inp, portMAX_DELAY);
    uprintf(buffer, "%d", inp);

    return MAIN_MENU;
}
```

payment.c

```
#define PRIORITY_HIGH 3
#define PRIORITY_MEDIUM 2
#define PRIORITY_LOW 1
#define PRIORITY_IDLE 0

int main(void){
    uart0_init(19200, 8, 1, 0);
    init_systick();
    key_init();
    digiswitch_init();
    lcd_init();
    switch_init();
    lcd_init();
    led_off();
    payment_init();
    coffee_init();
    log_init();

    // Create tasks

    xTaskCreate(coffee_task, "Coffee task",
configMINIMAL_STACK_SIZE + 100,
                NULL, PRIORITY_LOW, &coffee_t);

    xTaskCreate(payment_task, "Payment task",
configMINIMAL_STACK_SIZE + 100,
                NULL, PRIORITY_LOW, &payment_t);

    xTaskCreate(uart0_write_task, "UART write task",
configMINIMAL_STACK_SIZE,
                NULL, PRIORITY_LOW, NULL);

    xTaskCreate(key_task, "Key task",
configMINIMAL_STACK_SIZE,
                NULL, PRIORITY_IDLE, NULL);

    xTaskCreate(digiswitch_task, "Digitalswitch task",
configMINIMAL_STACK_SIZE,
                NULL, PRIORITY_HIGH, NULL);

    xTaskCreate(lcd_task, "LCD task",
configMINIMAL_STACK_SIZE + 100,
                NULL, PRIORITY_MEDIUM, NULL);

    xTaskCreate(log_task, "Logger task",
configMINIMAL_STACK_SIZE,
                NULL, PRIORITY_MEDIUM, NULL);

    xTaskCreate(ui_task, "UI task",
configMINIMAL_STACK_SIZE + 100,
                NULL, PRIORITY_MEDIUM, NULL);

    vTaskStartScheduler();

    return 1;
}
```

main.c

```
MENU main_menu()
{
    ui_clear_screen();
    uprintf(buffer, "----------------------\n\r");
    uprintf(buffer, "Select menu\n\r");
    uprintf(buffer, "1 - List logs\n\r");
    uprintf(buffer, "2 - Sales report\n\r");
    uprintf(buffer, "----------------------\n\r");

    INT8U inp;
    while (1)
    {
        xQueueReceive(uart0_rx_queue, &inp, portMAX_DELAY);
        switch (inp)
        {
        case '1':
            return LOG_LIST;
        case '2':
            return SALES_REPORT;
        default:
            uprintf(buffer, "%c\n\rIncorrect input\n\r", inp);
            break;
        }
    }
}
```

```
MENU log_list_menu()
{
    int i;
    ui_clear_screen();
    uprintf(buffer,
            "Order no. | Coffee no. | Coffee name   | Price | Payment type\n\r");

    xSemaphoreTake(log_array_semaphore, portMAX_DELAY);
    xSemaphoreTake(coffee_types_mutex, portMAX_DELAY);

    for (i = 0; i < LOG_LENGTH; i++)
    {
        if (log_array[i].active)
        {
            uprintf(buffer, "%-10d| %-11d| %-14s| %-6d| %s\n\r", i,
                    log_array[i].coffee_number,
                    coffee_types[log_array[i].coffee_number].name,
                    log_array[i].price, log_array[i].payment_type);
        }
        else
        {
            break;
        }
    }

    xSemaphoreGive(log_array_semaphore);
    xSemaphoreGive(coffee_types_mutex);

    uprintf(buffer, "Press any key to return\n\r");
    INT16U inp = 0;
    xQueueReceive(uart0_rx_queue, &inp, portMAX_DELAY);
    uprintf(buffer, "%d", inp);

    return MAIN_MENU;
}
```

```
MENU sales_report_menu()
{
    int i, j;
    ui_clear_screen();
    uprintf(buffer, "Coffee name  | Total sales\n\r");

    xSemaphoreTake(log_array_semaphore, portMAX_DELAY);
    xSemaphoreTake(coffee_types_mutex, portMAX_DELAY);

    for (i = 0; i < COFFEE_TYPES_LENGTH; i++)
    {
        if (coffee_types[i].active)
        {
            INT8U sum = 0;
            for (j = 0; j < LOG_LENGTH; j++)
            {
                if (log_array[j].active)
                {
                    if (i == log_array[j].coffee_number)
                    {
                        sum += log_array[j].price;
                        break;
                    }
                }
                else
                {
                    break;
                }
            }
            uprintf(buffer, "%-14s| %d\n\r", coffee_types[i].name, sum);
        }
        else
        {
            break;
        }
    }

    xSemaphoreGive(log_array_semaphore);
    xSemaphoreGive(coffee_types_mutex);

    uprintf(buffer, "Press any key to return\n\r");
    INT16U inp = 0;
    xQueueReceive(uart0_rx_queue, &inp, portMAX_DELAY);
    uprintf(buffer, "%d", inp);

    return MAIN_MENU;
}
```

report.c

video