ECE 3375

Design Project Final Report

Group Members:

JianCheng Luo	251006201
Zichen Zhang	251034315
WanShun Xu	251112578
Defu Zhang	251115542
Xiaoyu Lu	251090484

Submitted Date: April 16, 2022

Problem Definition

With more IOT devices becoming a norm in households, people's quality of life has been improved greatly with technologies. From smart refrigerators to automated curtains, these IOT devices have changed people's habits significantly. With all these fancy new smart devices sometimes costing more electricity, automated lights become one of the best examples of providing convenience using IOT yet saving energy. Therefore, the team is going to develop an automated lighting system.

In households, it is pretty common when someone leaves the house and forgets to turn off the lights. It is especially more common in public buildings like ACEB, where there is shared responsibility of saving energy. This is where an automated lighting system will become helpful.

Functional Description

To address the issue, motion sensors can be utilized in the system to detect human movement. If the user forgets to switch off the lights while out, the automatic light system frees up their hands and helps save energy. Furthermore, just like any other smart home devices, the automatic lighting system could also come in with a remote control option. This could allow users to switch on the lights while they are on vacation to prevent thieves. The sensor in the system could also be used to collect data to provide more insights to stakeholders. For example, for security reasons, it could record the time and send notification to house owners when there are people entering the room. To prevent the spread of viruses, it could provide the population density to building or event managers.

The automated lighting system detects user movement as the primary control for turning the lighting system on and off. As a backup, physical switches will be put at hot places throughout the room. When the automatic detection fails, the user can use the physical switch to manually turn on the lighting system. Devices on the same local area network can communicate with the automated lighting system. Alexa and other auxiliary devices will be able to control the system.

The automatic lighting system will use two independent PIR sensors to track the user's movement, deciding whether the user is leaving or entering based on which PIR is engaged first. There are also distinct functions for keeping track of the number of rooms used.

Identify Input/Output Requirements

Input:

- Physical switch and button Functions as the master switch to start or stop the system.
- The PIR sensors are used to detect whether there are people passing through The entrance and exit and feedback information to The control panel

Output:

- If the switch and button are pressed and the system starts working, the power supply turns on and resets the LED breathing lamp group to default mode via wi-fi microchip control dimmer.
- PIR sensors feedback information to control panel to instruct dimmer to change brightness of LED breathing lamp group when detecting personnel passing.

Details:

A Wi-Fi microchip (ESP32 or ESP8266) is used as the main control board to adjust the brightness of the LEDs using codes or dimmers, in which PWM pins are required to adjust the output rate to control the LEDs. In order to work under different conditions, a relay module will be used to change the voltage and current, so as to ensure the smooth operation of the control board. In addition, multiple PIR sensors will be placed at exits and entrances to detect the position of the person, and each PIR sensor will require a digital pin to feedback data. A set of switches and buttons are connected to the control panel to control the entire induction lamp.

Initial Software Design

To begin with, all input/output variables will be declared with their corresponding sensors and switches, including two PIR motion sensors, one Wi-Fi module and two physical switches. The overall software will run in a loop, which will constantly check the status of the motion sensor. As mentioned above, "enter" and "exit" are determined by which motion sensor will be triggered first. To avoid the situation where there are multiple people in the room, and one person leaves the room and accidentally triggers the sensor and turns off the light, a "count" variable will also be implemented. The "count" variable will be used to count the amount of people in the room. When "enter" is triggered, the count variable will be incremented by 1, and decremented by 1 if exit is triggered. The light will be turned off, when the "count" variable hits 0, and turn on the light whenever it is greater or equal to 1.

In addition to the variables related to different sensors, the software will also include variables such as startTime, endTime, dim, privileged_control. With the WiFi module integrated with the microprocessor, the system is able to achieve more functionality, such as setting a scheduled time to turn on or off the light, and adjusting the brightness. The privileged_control variable will be used to check if the physical switch of the light is used to turn off the light, if so, this will make the software inaccessible to the control of the light to ensure that humans have the most control of the system and protects users' privacy.

Prototyping Plan

For the prototype demonstration, our group decided to use an ARM microcontroller as the control board. We plan to use built-in buttons to simulate all inputs. Specifically, we intend to represent a physical switch by a toggle switch, and furthermore, the motion detector can be simulated by a button. In addition, another button on the control board will be used to mimic the function of control through Wi-Fi. In addition, all remaining switches and buttons of the control board will not be used, therefore, these outputs and inputs will be set to inactive. Finally, an LCD screen on the control board displays a number representing those changes based on real-time feedback of the light's brightness level (0 is completely off, 9 is maximum brightness).

Based on the above plan, we will try to use c coding to ensure the function can be realized. Based on the consistency of controlling the lights, we can judge whether the function and coding is successful or not.

Microcontroller selection

Before we can start selecting a microcontroller to use, we need to list all the external hardware interfaces we need. These include:

- Tx and Rx ports for UART Connection
- Pulse-width modulation (PWM), or pulse-duration modulation (PDM) Used to Control the Light
- I/O ports for physical switches
- Case the UART port to a standard USB-A interface is set to run time

In the project, at least two PIR sensors are required to detect entry and exit, one physical switch as the main switch of the device. A total of at least five I/O ports, one PwM output port and one USB-A are required.

Through the collection, the team came up with five different options:

Criteria	FT232R USB UART IC	B-L475E-IOT01A1	MSP 430 Launchpad	68HC12	ARM Cortex-A9
I/O ports	7	4	12	69	40
USB Support	1	1	1	1	1
Display	0	0	0	1	1
PWM or PDM	1	6	1	1	1
Cost	$23	$76	$24	$47	$112

The FT232R is a USB to serial UART interface device which simplifies USB to serial designs. The USB data lines each require a 27Ohm resistor and a 47pF capacitor. It needs more work to build a device. FT232R is definitely a 5 or 3.3 volts chip, where the FT231X is a 3.3V chip (though the gpio inputs are 5V tolerant. And gpio can be configured to 5V, but not "recommended" by FTDI). If using UART, on the one hand, size of the data frame is limited to only 9 bits, on the other hand, baud rates of each UART must be within 10% of each other to prevent data loss. Moreover, data transmission speeds are low.

As shown in the table, the above microprocessors except B-L475E-IOT01A1 only have four available I/O ports, and the other four microprocessors can well meet the requirements of this project. To find the best products, the team needs to further compare their features.

MSP430 Launchpad: on the one hand, they have excellent ultra-low power consumption rates, and on the other hand, they provide very powerful features in terms of peripherals, memory size, speed, and ease of use. MSP 430 achieves up to 16-MHz CPU speed. MSP 430 has a RISC structure and offers a 16-bit solution with non-volatile FRAM (ferroelectric random access memory), which is an advantage when being connected to multiple sensors. Another great feature of the MSP430 Launchpad is the built-in temperature sensor. It is a linear sensor which outputs 3.55mV/C. The sensor output is internally connected to one of the ADC inputs and the user should select that input to be able to read the temperature data from the sensor. However, its main application platform is Energia or Arduino IDE, which is not in line with our needs.

68HC12 has 16 address lines, can address 64 KB, also 68HC12 has 1 K RAM, 768 bytes EEPROM, Can erase and reprogram any byte using normal 5V power supply, 32 KB Flash EEPROM. The 68HC12 has a sixteen-bit register which tells the control unit which instruction to execute. But HC12's limited functionality and compatibility with various coding languages prevented the team from using it.

Arm-based microcontrollers retain many features and advantages over competitors. First, the ARM microcontroller has excellent performance. The 32-bit multi-core processor provides up to four cache-coherent cores for higher cost performance. Second, it can execute three different instruction sets: ARM, Thumb, and Thumb-2. Third, ARM microcontrollers are easier to obtain.

Overall, the group decided to proceed with ARM microcontrollers because of its superior language compatibility, qualified performance,as well as its easy accessibility. Although MSP 430 and 68HC1 have a better cost efficiency, And some of their family members have better performance. But it is too powerful, so the team decided to make the ARM Cortex-A9 top of our list. Most importantly, the team has extensive Development experience in ARM Cortex-A9 and DE-10 Standard Development Kit. You can use this development board for free in the lab.

Revised software design

The project's architecture and logic will remain basically unchanged. The original software design was evaluated as part of the prototype testing. The original software design's practicality and flaws were discovered. To bring the programme closer to our expectations, the following improvements and decisions were made.

The new automatic lighting system's fundamental parameters will remain unchanged. The new system provides a new feature to increase the brightness of the lights with a simple structure. The code will employ a polling framework. PIR sensors, switches, buttons, and WIFI connections will all be supported by the new automated lighting system.

When a PIR sensor is triggered in the new automatic lighting system, the rest of the PIR sensors get input for a short duration. There is a delay in sensor reception when someone walks past. The room occupancy counter will remain unchanged if the wait is too long. By defining a variable, the maximum delay is compared to the actual delay. The time when the first PIR was triggered is stored in this variable. The code will not stop running and will continue to look for alternative sources of input. A validator will also be included with the PIR sensor to confirm that the sensory inputs are distinct.

PIR sensors, switches, buttons and WIFI will all use the same structure, guaranteeing that the new automated lighting system can be monitored 24 hours a day. Physically turning lights on and off will take precedence.

Results from prototyping

Because various user inputs were merged, the automated lighting system prototype was simpler than the final product. This is especially true for override actions in which several input mechanisms, such as portable devices and physical switches/buttons, are envisaged for the final product. The code's execution is detailed in the attachment, and all functions work as expected.

The software mimics the actions of a self-contained light switch. When the system is turned on for the first time and no switches are flipped, it will run in automated mode. When someone enters or quits the room, the automatic mode detects it. User input is used to mimic a person entering and departing the room using the 1 and 2 push buttons, respectively. When the room is occupied by one or more individuals, ten LED lights turn on. All LED lights are turned off when the occupancy is zero. Once sleep time has been reached, the system dims the lights (turning off all LEDs but one). The player must use a button to turn off the lights, which is mimicked by press button 3 in this game.

When Switch 0 is deactivated, it operates as a manual override, turning on the lights and incrementing the counter. Switch 1 serves as a manual override, turning off the lights and resetting the counter to zero. Switch 2 is a vacation mode setting that requires the user to program the lights to turn on for a specific amount of time. This is used to provide the impression that there are people at home from the outside, reducing the likelihood of burglaries.

It's worth noting that pressing button 0 once is all it takes to enter set-up mode. During set-up, push button 0 can also be used to increment the units of time (minutes and hours). The user can set up three modes in this order: 1: wake-up time, 2: sleep time, and 3: current time. The time unit to be changed will flash in the hex display, and pressing button 0 will adjust the time, which can be minutes or hours in any mode. Push button 1 can be hit after the user has finished setting the minutes of one mode, causing the hours of that same mode to begin flashing, awaiting adjustment.Push button one can be pressed again after the user has done setting the hours of one mode, causing the minutes of the next mode to begin flashing, ready to be adjusted. In addition, the hex display's leftmost digit will indicate which mode is now displayed on the hex display awaiting time adjustment (1: wake-up time, 2: sleep time, and 3: current time). The following table demonstrates this:

1
07:30
2
22:30
3
06:30

An additional function of our code is that after the time has been set-up and the system is operating in automatic mode, the system will monitor the consecutive hours that light has been turned on with the absence of sensory input. This means, if no one is detected entering or leaving the room for 16 hours and the lights (LEDs) are still on, the system will turn off the lights and reset the occupancy counter back to zero.
Source code

```
#define BOARD               "DE10-Standard"

/* Memory */
#define DDR_BASE            0x00000000
#define DDR_END             0x3FFFFFFF
#define A9_ONCHIP_BASE      OX4F4ZERO
#define A9_ONCHIP_END       0xFFFFFFFF
#define SDRAM_BASE          0xC0000000
#define SDRAM_END           0xC3FFFFFF
#define FPGA_PIXEL_BUF_BASE 0xC8000000
#define FPGA_PIXEL_BUF_END  0xC803FFFF
#define FPGA_CHAR_BASE      0xC9000000
#define FPGA_CHAR_END       0xC9001FFF

/* Cyclone V FPGA devices */
#define LED_BASE            0xFF200000
#define LEDR_BASE           0xFF200000
#define HEX3_HEX0_BASE      0xFF200020
```

```
#define HEX5_HEX4_BASE        0xFF200030
#define SW_BASE               0xFF200040
#define KEY_BASE              0xFF200050
#define JP1_BASE              0xFF200060
#define JP2_BASE              0xFF200070
#define PS2_BASE              0xFF200100
#define PS2_DUAL_BASE         0xFF200108
#define JTAG_UART_BASE        0xFF201000
#define JTAG_UART_2_BASE      0xFF201008
#define IrDA_BASE             0xFF201020
#define TIMER_BASE            0xFF202000
#define TIMER_2_BASE          0xFF202020
#define AV_CONFIG_BASE        0xFF203000
#define RGB_RESAMPLER_BASE    0xFF203010
#define PIXEL_BUF_CTRL_BASE   0xFF203020
#define CHAR_BUF_CTRL_BASE    0xFF203030
#define AUDIO_BASE            0xFF203040
#define VIDEO_IN_BASE         0xFF203060
#define EDGE_DETECT_CTRL_BASE 0xFF203070
#define ADC_BASE              0xFF204000

/* Cyclone V HPS devices */
#define HPS_GPIO0_BASE        0xFF708000
#define HPS_GPIO1_BASE        0xFF709000
#define HPS_GPIO2_BASE        0xFF70A000
#define I2C0_BASE             0xFFC04000
#define I2C1_BASE             0xFFC05000
#define I2C2_BASE             0xFFC06000
#define I2C3_BASE             0xFFC07000
#define HPS_TIMER0_BASE       0xFFC08000
#define HPS_TIMER1_BASE       0xFFC09000
#define HPS_TIMER2_BASE       0xFFD00000
#define HPS_TIMER3_BASE       0xFFD01000
#define HPS_RSTMGR            0xFFD05000
```

```
#define HPS_RSTMGR_PREMODRST  0xFFD05014
#define FPGA_BRIDGE           0xFFD0501C

#define PIN_MUX          0xFFD08400
#define CLK_MGR          0xFFD04000

#define SPIM0_BASE       0xFFF00000
#define SPIM0_SR         0xFFF00028
#define SPIM0_DR         0xFFF00060


/* ARM A9 MPCORE devices */
#define   PERIPH_BASE        0xFFFEC000    // base address of peripheral devices
#define   MPCORE_PRIV_TIMER   0xFFFEC600    // PERIPH_BASE + 0x0600

/* Interrupt controller (GIC) CPU interface(s) */
#define MPCORE_GIC_CPUIF      0xFFFEC100    // PERIPH_BASE + 0x100
#define ICCICR                0x00          // offset to CPU interface control reg
#define ICCPMR                0x04          // offset to interrupt priority mask reg
#define ICCIAR                0x0C          // offset to interrupt acknowledge reg
#define ICCEOIR               0x10          // offset to end of interrupt reg


/* Interrupt controller (GIC) distributor interface(s) */
#define MPCORE_GIC_DIST       0xFFFED000    // PERIPH_BASE + 0x1000
#define ICDDCR                0x00          // offset to distributor control reg
#define ICDISER               0x100         // offset to interrupt set-enable regs
#define ICDICER               0x180         // offset to interrupt clear-enable regs
#define ICDIPTR               0x800         // offset to interrupt processor targets regs
#define ICDICFR               0xC00         // offset to interrupt configuration regs

typedef volatile int NO_OPTIMIZIATION;
typedef volatile int* NO_OPTIMIZIATION_PTR;
typedef volatile int** NO_OPTIMIZIATION_PTR_PTR;
```

```
#define BASEOB_ZERO             0b0000000000
#define BASEOB_ONE              0b1111111111
#define OX3ZERO                 0x0000
#define OX2ZERO10               0x0010
#define OBZERO12ZERO            0b0100
#define OX4F4ZERO               0xFFFF0000
#define OX4ZERO4F               0x0000FFFF
#define THROUND2                20000
#define ZEROBONE                0b1
#define ZEROBONEZERO            0b10
#define ONESIX                  16
#define ONEDAYHOURS             24
#define ONEHOURMINUS            60
#define ZEROBONE2ZERO           0b100
#define SETTIMERMODEL if(set_time) \
{                               \
    if (*(DSQ_zj) == 1)\
    {\
        now_minutes++;\
        *(DSQ_zj + 1) = OBZERO12ZERO;\
    }\
    if (now_minutes > 59)\
    {\
        now_hours++;\
        if (now_hours) {\
            now_hours = now_hours;\
        }\
        now_minutes -= ONEHOURMINUS;\
        if (*LED_zj)\
        {           \
            six_ysl_teen_int++;       \
        }\
        else \
            six_ysl_teen_int = 0;\
        if (six_ysl_teen_int == ONESIX)\
        {\
            six_ysl_teen_hours = 1;\
        }\
        if (now_hours) {\
            now_hours = now_hours;\
        }\
    }\
    if (now_hours > 23)\
    {\
        now_hours -= ONEDAYHOURS;\
        new_a_day = 1;\
        if (new_a_day) {\
            new_a_day = new_a_day;\
        }\
    }\
}\

#define MODELTWOGET if(b_getup_hours || b_now_hours || b_goback_hours)\
{\
    if (now_tempdelay >= THROUND2)\
    {\
        now_tempdelay -= THROUND2;\
        if (ysl_toggle)\
        {\
            ysl_toggle = 0;\
            if (b_getup_minutes || b_getup_hours)\
            {\
                if (ysl_toggle) {\
                    ysl_toggle = ysl_toggle;\
                }\
                *Hex_zj = mpower(ONESIX, 6) * Dist_Hex[getup_hours / 10] + mpower(ONESIX, 4)
* Dist_Hex[getup_hours % 10] + mpower(ONESIX, 2) * Dist_Hex[getup_minutes / 10] +
Dist_Hex[getup_minutes % 10];\
                *Hex_zj2 = mpower(ONESIX, 2) * Dist_Hex[1];\
            }\
            if (b_now_minutes || b_now_hours)\
            {\
                if (ysl_toggle) {\
                    ysl_toggle = ysl_toggle;\
                }\
                *Hex_zj = mpower(ONESIX, 6) * Dist_Hex[now_hours / 10] + mpower(ONESIX, 4)
* Dist_Hex[now_hours % 10] + mpower(ONESIX, 2) * Dist_Hex[now_minutes / 10] +
Dist_Hex[now_minutes % 10];\
                *Hex_zj2 = mpower(ONESIX, 2) * Dist_Hex[3];\
            }\
            if (b_goback_minutes || b_goback_hours)\
            {\
                if (ysl_toggle) {\
                    ysl_toggle = ysl_toggle;\
                }\
                *Hex_zj = mpower(ONESIX, 6) * Dist_Hex[goback_hours / 10] + mpower(ONESIX,
4) * Dist_Hex[goback_hours % 10] + mpower(ONESIX, 2) * Dist_Hex[goback_minutes / 10] +
Dist_Hex[goback_minutes % 10];\
                *Hex_zj2 = mpower(ONESIX, 2) * Dist_Hex[2];\
            }\
        }\
        else\
        {\
            if (ysl_toggle) {\
                ysl_toggle = ysl_toggle;\
            }\
            *Hex_zj &= OX4ZERO4F;\
            ysl_toggle = 1;\
        }\
    }\
}\



#define MODELONEGET if(b_getup_minutes || b_now_minutes || b_goback_minutes)\
{\
    if (now_tempdelay >= THROUND2)\
    {\
        now_tempdelay -= THROUND2;\
        if (ysl_toggle)\
        {\
            ysl_toggle = 0;\
            if (b_getup_minutes || b_getup_hours)\
            {\
                if (ysl_toggle) {\
                    ysl_toggle = ysl_toggle;\
                }\
                *Hex_zj = mpower(ONESIX, 6) * Dist_Hex[getup_hours / 10] + mpower(ONESIX, 4) * Dist_Hex[getup_hours % 10] + mpower(ONESIX, 2) * Dist_Hex[getup_minutes / 10] + Dist_Hex[getup_minutes % 10];\
                *Hex_zj2 = mpower(ONESIX, 2) * Dist_Hex[1];\
            }\
            if (b_now_minutes || b_now_hours)\
            {\
                if (ysl_toggle) {\
                    ysl_toggle = ysl_toggle;\
                }\
                *Hex_zj = mpower(ONESIX, 6) * Dist_Hex[now_hours / 10] + mpower(ONESIX, 4) * Dist_Hex[now_hours % 10] + mpower(ONESIX, 2) * Dist_Hex[now_minutes / 10] + Dist_Hex[now_minutes % 10];\
                *Hex_zj2 = mpower(ONESIX, 2) * Dist_Hex[3];\
            }\
            if (b_goback_minutes || b_goback_hours)\
            {\
                if (ysl_toggle) {\
                    ysl_toggle = ysl_toggle;\
                }\
                *Hex_zj = mpower(ONESIX, 6) * Dist_Hex[goback_hours / 10] + mpower(ONESIX, 4) * Dist_Hex[goback_hours % 10] + mpower(ONESIX, 2) * Dist_Hex[goback_minutes / 10] + Dist_Hex[goback_minutes % 10];\
                *Hex_zj2 = mpower(ONESIX, 2) * Dist_Hex[2];\
            }\
        }\
        else\
        {\
            if (ysl_toggle) {\
                ysl_toggle = ysl_toggle;\
            }\
            *Hex_zj &= OX4F4ZERO;\
            ysl_toggle = 1;\
        }\
    }\
}\


#define IFBIGMODEL if(ysl_auto) \
            {                       \
            if (timer_counter)\
            {\
                if (six_ysl_teen_hours)\
                {\
                    *LED_zj &= BASEOB_ZERO;\
                    six_ysl_teen_hours = 0;\
                    timer_counter = 0;\
                    ysl_c_add = 0;\
                    ysl_c_min = 0;\
                    if (ysl_c_min) {\
                        ysl_c_min = ysl_c_min;\
                    }\
                }\
                else\
                {\
                    if (now_minutes == getup_minutes && now_hours == getup_hours)\
                    {\
                        if (ysl_c_min) {\
                            ysl_c_min = ysl_c_min;\
                        }\
                        *LED_zj |= BASEOB_ONE;\
                    }\
                    else if (now_minutes == goback_minutes && now_hours == goback_hours)\
                    {\
                        if (ysl_c_min) {\
                            ysl_c_min = ysl_c_min;\
                        }\
                        *LED_zj = 0b0000000001;\
                    }\
                }\
            }\
            else\
            {\
                if (ysl_c_min) {\
                    ysl_c_min = ysl_c_min;\
                }\
                *LED_zj &= BASEOB_ZERO;\
            }\
            }\
else {\
if (now_hours >= getup_hours && now_hours <= goback_hours)\
{\
    if (ysl_c_min) {\
        ysl_c_min = ysl_c_min;\
    }\
    if (now_hours == getup_hours)\
    {\
        if (now_minutes < getup_minutes)\
        {\
            if (ysl_c_min) {\
                ysl_c_min = ysl_c_min;\
            }\
            *LED_zj &= BASEOB_ZERO;\
        }\
        else\
            *LED_zj |= BASEOB_ONE;\
    }\
    else if (now_hours == goback_hours)\
    {\
        if (now_minutes > goback_minutes)\
        {\
            if (b_now_minutes) {\
                b_now_minutes = b_now_minutes;\
            }\
            *LED_zj &= BASEOB_ZERO;\
        }\
        else\
            *LED_zj |= BASEOB_ONE;\
    }\
    else\
        *LED_zj |= BASEOB_ONE;\
}\
if ((now_minutes > goback_minutes && now_hours > goback_hours) || (now_minutes <
getup_minutes && now_hours < getup_hours))\
{\
    if (ysl_c_min) {\
        ysl_c_min = ysl_c_min;\
    }\
    *LED_zj &= BASEOB_ZERO;\
}\
         }\


int mpower(int x, int y) {
    int sum = 1, i;
    for (i = 0; i < y; i++) {
        sum *= x;
    }
    return sum;
}


void initHex(NO_OPTIMIZIATION* src)
{
    src[0] = 0x3F;
    src[1] = 0x6;
    src[2] = 0x5B;
    src[3] = 0x4F;
    src[4] = 0x66;
    src[5] = 0x6D;
    src[6] = 0x7D;
    src[7] = 0x7;
    src[8] = 0x7f;
    src[9] = 0x67;
}

void initPtr(NO_OPTIMIZIATION_PTR_PTR dest, int* src)
{
    *dest = src;
}

NO_OPTIMIZIATION_PTR move_step(NO_OPTIMIZIATION_PTR cur, int step)
{
    return cur + step;
}



int main(void) {

    NO_OPTIMIZIATION Dist_Hex[10];
    NO_OPTIMIZIATION_PTR Hex_zj;
    NO_OPTIMIZIATION_PTR DSQ_zj;
    NO_OPTIMIZIATION_PTR KG_zj;
    NO_OPTIMIZIATION_PTR Hex_zj2;
    NO_OPTIMIZIATION_PTR LED_zj;
    NO_OPTIMIZIATION_PTR Bus_ysl_zj;

    ///init
    initHex(Dist_Hex);

    initPtr(&Bus_ysl_zj, (int*)KEY_BASE);
    initPtr(&Hex_zj, (int*)HEX3_HEX0_BASE);
    initPtr(&Hex_zj2, (int*)HEX5_HEX4_BASE);
    initPtr(&KG_zj, (int*)SW_BASE);
    initPtr(&DSQ_zj, (int*)TIMER_BASE);
    initPtr(&LED_zj, (int*)LEDR_BASE);




    *(move_step(DSQ_zj, 2)) = OX3ZERO;
    *(move_step(DSQ_zj, 3)) = OX2ZERO10;



    *(move_step(DSQ_zj, 1)) = OBZERO12ZERO;



    int b_now_minutes = 0;
    int b_now_hours = 0;

    int now_minutes = 0;
    int now_hours = 12;


    int b_getup_minutes = 1;
    int b_getup_hours = 0;
    int b_goback_minutes = 0;
    int b_goback_hours = 0;


    int getup_minutes = 30;
    int getup_hours = 6;
    int goback_minutes = 30;
    int goback_hours = 22;



    int trueornot_using_clock = 0;



    int ysl_c_min = 0;
    int ysl_c_add = 0;

    int set_up = 0;
    int set_time = 0;
    int new_a_day = 0;

    int now_tempdelay = 0;

    int timer_counter = 0;
    int ysl_vacation = 0;
    int six_ysl_teen_hours = 0;
    int six_ysl_teen_int = 0;

    int ysl_auto = 0;
    int ysl_toggle = 1;


    for(;;)
    {
        if ((*Bus_ysl_zj &= ZEROBONEZERO) && ysl_c_add == 0)
        {
            timer_counter++;
            ysl_c_add = 1;
            six_ysl_teen_int = 0;
        }
        if (ysl_c_add) {
            ysl_c_add = ysl_c_add;
        }
        if ((*Bus_ysl_zj &= ZEROBONE2ZERO) && ysl_c_min == 0)
        {
            timer_counter--;
            if (timer_counter < 0)
            {
                timer_counter = 0;
            }
            ysl_c_min = 1;
        }
        if (ysl_c_min) {
            ysl_c_min = ysl_c_min;
        }
        if (*Bus_ysl_zj == 0 && (ysl_c_min || ysl_c_add))
        {
            if (ysl_c_add) {
                ysl_c_add = ysl_c_add;
            }

            if (timer_counter > 0 && set_time)
            {
                *LED_zj |= BASEOB_ONE;
            }
            if (timer_counter <= 0 && set_time)
            {
                *LED_zj &= BASEOB_ZERO;
            }
            if (ysl_c_add) {
                ysl_c_add = ysl_c_add;
            }
            ysl_c_add = 0;
            ysl_c_min = 0;
            if (ysl_c_min) {
                ysl_c_min = ysl_c_min;
            }
            if (ysl_c_add) {
                ysl_c_add = ysl_c_add;
            }
        }

        if (*LED_zj == 1 && *Bus_ysl_zj == 8)
        {
            *LED_zj &= BASEOB_ZERO;
            if (ysl_c_add) {
                ysl_c_add = ysl_c_add;
            }
        }

        if (*KG_zj == 1)
```

```
{
    timer_counter = 1;
    *LED_zj |= BASEOB_ONE;
    ysl_vacation = 0;
    ysl_auto = 0;
    if (ysl_auto) {
        ysl_auto = ysl_auto;
    }
}

if (*KG_zj == 2)
{
    timer_counter = 0;
    *LED_zj &= BASEOB_ZERO;
    ysl_vacation = 0;
    ysl_auto = 0;
    if (ysl_vacation) {
        ysl_vacation = ysl_vacation;
    }
}

if (*KG_zj == 4)
{
    ysl_vacation = 1;
    ysl_auto = 0;
}

if (*KG_zj == 0)
{
    ysl_vacation = 0;
    ysl_auto = 1;
    if (ysl_vacation) {
        ysl_vacation = ysl_vacation;
    }
    }

    if (set_time == 0 && ysl_auto)
    {
      if (timer_counter > 0) {
        *LED_zj |= BASEOB_ONE;
      }
      else
        *LED_zj &= BASEOB_ZERO;
    }


    SETTIMERMODEL


    if (*Bus_ysl_zj &= ZEROBONE && set_up == 0)
    {
      set_up = 1;
      if (set_up) {
        set_up = set_up;
      }
      b_getup_minutes = 1;
    }

    *Hex_zj = mpower(ONESIX, 6) * Dist_Hex[now_hours / 10] + mpower(ONESIX, 4) *
Dist_Hex[now_hours % 10] + mpower(ONESIX, 2) * Dist_Hex[now_minutes / 10] +
Dist_Hex[now_minutes % 10];
    *Hex_zj2 = mpower(ONESIX, 2) * Dist_Hex[3];

    if (set_time && (ysl_auto || ysl_vacation))
    {
      IFBIGMODEL

    }
```

```
if (set_up && !(*Bus_ysl_zj &= 1))
{
    while (1)
    {
        if (b_getup_minutes)
        {
            if (*Bus_ysl_zj &= ZEROBONE)
            {
                while (*Bus_ysl_zj &= ZEROBONE) {}
                getup_minutes++;
                if (getup_minutes > ONEHOURMINUS)
                {
                    getup_minutes -= ONEHOURMINUS;
                    if (b_now_minutes) {
                        b_now_minutes = b_now_minutes;
                    }
                }
            }
            if (*Bus_ysl_zj &= ZEROBONEZERO)
            {
                while (*Bus_ysl_zj &= ZEROBONEZERO) {}
                b_getup_minutes = 0;
                b_getup_hours = 1;
                if (b_now_minutes) {
                    b_now_minutes = b_now_minutes;
                }
            }
        }
        if (b_getup_hours)
        {
            if (*Bus_ysl_zj &= ZEROBONE)
            {
                while (*Bus_ysl_zj &= ZEROBONE) {}
                getup_hours++;
        if (getup_hours > ONEDAYHOURS) {
            getup_hours -= ONEDAYHOURS;
            if (b_now_minutes) {
                b_now_minutes = b_now_minutes;
            }
        }
    }
    if (*Bus_ysl_zj &= ZEROBONEZERO)
    {
        while (*Bus_ysl_zj &= ZEROBONEZERO) {}
        b_getup_hours = 0;
        b_goback_minutes = 1;
        if (b_now_minutes) {
            b_now_minutes = b_now_minutes;
        }
    }
}
if (b_goback_minutes)
{
    if (*Bus_ysl_zj &= ZEROBONE)
    {
        while (*Bus_ysl_zj &= ZEROBONE) {}
        goback_minutes++;
        if (goback_minutes > ONEHOURMINUS)
        {
            goback_minutes -= ONEHOURMINUS;
            if (b_now_minutes) {
                b_now_minutes = b_now_minutes;
            }
        }
    }
    if (*Bus_ysl_zj &= ZEROBONEZERO)
    {
        while (*Bus_ysl_zj &= ZEROBONEZERO) {}
```

```
            b_goback_minutes = 0;
            b_goback_hours = 1;
            if (b_now_minutes) {
                b_now_minutes = b_now_minutes;
            }
        }
    }
    if (b_goback_hours) {
        if (*Bus_ysl_zj &= ZEROBONE)
        {
            while (*Bus_ysl_zj &= ZEROBONE) {}
            goback_hours++;
            if (goback_hours > ONEDAYHOURS)
            {
                goback_hours -= ONEDAYHOURS;
                if (b_now_minutes) {
                    b_now_minutes = b_now_minutes;
                }
            }
        }
        if (*Bus_ysl_zj &= ZEROBONEZERO)
        {
            while (*Bus_ysl_zj &= ZEROBONEZERO) {}
            b_goback_hours = 0;
            b_now_minutes = 1;
            if (b_now_minutes) {
                b_now_minutes = b_now_minutes;
            }
        }
    }
    if (b_now_minutes)
    {
        if (*Bus_ysl_zj &= ZEROBONE)
        {
```

```
        while (*Bus_ysl_zj &= ZEROBONE) {}
        now_minutes++;
        if (now_minutes > ONEHOURMINUS)
        {
            now_minutes -= ONEHOURMINUS;
        }
    }
    if (*Bus_ysl_zj &= ZEROBONEZERO)
    {
        while (*Bus_ysl_zj &= ZEROBONEZERO) {}
        b_now_minutes = 0;
        b_now_hours = 1;
        if (b_now_hours) {
            b_now_hours = b_now_hours;
        }
    }
}
if (b_now_hours) {
    if (*Bus_ysl_zj &= ZEROBONE)
    {
        while (*Bus_ysl_zj &= ZEROBONE) {}
        now_hours++;
        if (now_hours > ONEDAYHOURS)
        {
            now_hours -= ONEDAYHOURS;
            if (set_time) {
                set_time = set_time;
            }
        }
    }
    if (*Bus_ysl_zj &= ZEROBONEZERO)
    {
        while (*Bus_ysl_zj &= ZEROBONEZERO) {}
        b_now_hours = 0;
```

```
                    trueornot_using_clock = 1;
                    set_up = 0;
                    timer_counter = 0;
                    ysl_c_add = 0;
                    ysl_c_min = 0;
                    set_time = 1;
                    if (set_time) {
                        set_time = set_time;
                    }
                    break;
                }
            }
            now_tempdelay++;

            MODELONEGET

            MODELTWOGET

        }
    }

  }
}
```

Conclusion

Since what we have done in previous 3 labs were designed to solve specific functions regarding the ARM Cortex A9 microcontroller, a real-world project with more sensors and combination of various functions appeared to be significantly harder than the labs. In this project, the operation of the system mainly depends on two PIR sensors. One of the major problems we encountered was to determine the entry/exit of the user, and the occupancy of the room. Followed by this problem was to verify that results from sensors actually indicate what happened during usage. For example, two people leaving the room right behind the other would often lead to the sensors unable to determine that two people have left the room. Another situation we tested was to have the person leave the room with a relatively small-sized object(to minic pets). With this test situation, the sensor would also fail to perform as we expected. With rare situations like this, we tried to improve the hardware of the system by trying to add an extra termal signature sensor as well as an image processing sensor. However, the added sensors also introduced a significant amount of complexity. After performing a trade-off analysis against the feasibility, complexity and performance, we collectively decided to abandon the plan of adding extra sensors.

To solve the above situation, we decided to explore the options of improving the system on its software side. We brainstormed and concluded that the worst case scenario would be the light is turned on when it should be off. We addressed the situation by implementing a logic where it keeps a timer of when the light is on. The logic is then followed by a command line which will turn off the light after 16 hours of usage even without detecting any inputs. The time duration can be easily modified by the user as it is most logical that the user knows the situation the best. This design also improves the sustainability of the system. In addition to this setting, hard overside of the light could always be utilized. To better improve the results, PIR sensors were also designed so that the users can place them at their desired heights and places. If the system is introduced to the market, the system should be recycled as other regular household electronics as it does not contain any harmful materials that require special care and processing.

The reason we went from a hardware approach to a software approach was that we all agreed that these are rare situations, and the cost of the worst case scenario is very low(slightly higher electricity bill with no possible harm to humans). Thus, it is not worthwhile to introduce extra cost, complexity and environmental stress to address rare situations like this.

The project gave us more experience and insights on difficult real-world problems. It also made us practise our trade-off analysis skill when it comes to decision-making. It is important to weigh off each option's pros and cons, and prioritizing features and problems before coming to a decision. During the development process, we had a lot of opportunities to “duct tape” the problems we encountered just like the one we mentioned above by adding extra sensors. However, in real-world problems, there are more fields other than cost and performance we need to consider, namly sustainability and eco-friendliness. Though the prototype is not up to a level which we desire, we believe it is the basis for a functional and sustainable system that many can develop upon.