By Dende Te (s3839316) and Rish Rao (s3843246)

SCHOOL OF ENGINEERING

EEET2096 – EMBEDDED SYSTEM DESIGN AND IMPLEMENTATION

LABORATORY PROJECT REPORT

HOME MANAGEMENT SYSTEM

Executive summary

Through Keil uVision, code in C was written to develop a home management system that involved configuring peripherals such as GPIO, UART, ADC and TIM. GPIO was configured in a way to allow for switches such as the light, light sensor and fan and the LEDS for output representation for the fan, light, cooling and heating. This allows for the user to turn on house system devices through toggling and monitor which devices are currently turned on or off. ADC was configured to obtain a conversion between values of 0 to 4095 ADC which converted to temperatures between 8 degrees celsius to 45 degrees celsius. This provided a suitable basis for the user to increase or decrease the temperature and for the program to determine which system device to turn to achieve the desired temperature. The TIM peripheral was used to create delays of 1 second, 15 seconds and 0.5 seconds through the prescaler 21000 and each of their individual count values. It is to be noted that the polling method was utilised to achieve these delays. UART was configured for the required baud rate of 57600 bps to obtain feedback in terms of the status of HMS.

Table of contents

Introduction

The project is to develop a Home Management System (HMS) that is built from writing a large-scale C code and is implemented on the RMIT STM32F439 Development platform. The HMS includes an emulated temperature sensor controlled by the ADC component of the platform which will transmit the status of the outputs according to the temperature sensed. Alternatively, the switches and UART are used to communicate with the system to operate the light and fan functions. The system requires specific delay durations which are implemented through the Timers in the platform using interrupts. The buttons simulated through the switches need to be pressed for at least 500ms to be registered and continue the switch operation.

Essential background information

Peripherals

Switches and LEDS

Switches and LEDS were all configured through the GPIO ports. The project consists of seven inputs and outputs that consist of switches or LEDS. The required switches are UP switch (Fan switch) , DOWN switch (Light intensity sensor) and LEFT switch (Light switch) which are configured in ports A at pins 3, 8 and 9 respectively. The required LEDS are 3 (Fan Control Output) , 4 (Light Control Output) , 6 (Cooling Output) and 7 (Heater Output). Majority of the LEDS had different configuration ports, LED 3 is to be configured in port A at pin 10, LEDS 4 and 6 to be configured in port B at pins 0 and 8 respectively and LED 7 to be configured in port F at pin 8.

UART3 and Timers

The UART3 is configured to operate at 57,600bps, 8 data-bits, No Parity and 1 Stop bit. UART3 is enabled from setting port B at pins 10 and 11 alternative function (0x02). The UART3 is able to connect from USB to UART with the terminal emulator ‘TeraTerm’ to transmit and receive data between the development board and PC. The Timer6 is used in the system to execute delays of 1 second for monitoring rate, half a second for switch key debounce and 15 seconds for fan off period.

ADC3

The ADC3 is configured by setting port F at pin 10 to be an analogue input with a prescaler of 8. Since ADC3 is required, the channel is set to Channel 8 with a sampling rate of 56 cycles. The ADC3 has values from 0 to 4095 and can be received from the DR (data register) to convert to a temperature functioning as a temperature sensor.

Functional Block Diagram

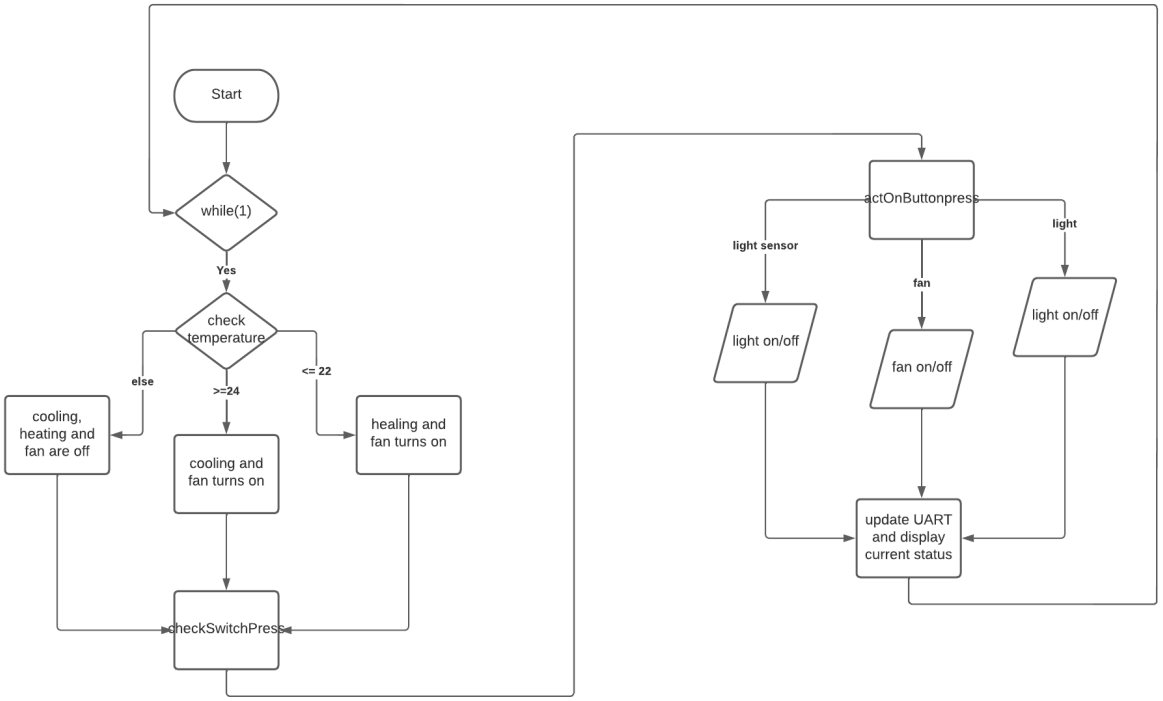

Figure 1: Flow diagram of the HMS program

Technical work and Results

Temperature Sensor (ADC) Calculation

At ADC = 0 → Temperature = 8 (degrees C)

ADC = 4095 → Temperature = 45 (degrees C)

With the given values, the linear relationship can be formed to find the temperature in terms of the ADC value.

Gradient = (45-8) / (4095-0)

= 37/4095

Temperature = (37/4095) * (ADC value) + 8

UART3 Calculation

USARTDIV = (42 x 10^6) / (16 x BAUDRATE)

= (42 x 10^6) / (16 x 57600)

= 45.5729

USARTDIV = ~45d (0x2D)

DIV_Fraction = 16 x 0.5729

DIV_Fraction = 9d (0x9)

TIM6 Calculation

15 seconds:

PSC = 21000

Onetick = 84000000 Hz/21000 = 4000 Hz

ARR/Onetick(Hz) = RT (Required Time)

ARR = 15*4000

ARR = 60000

1 second:

PSC = 21000

Onetick = 84000000 Hz/21000 = 4000 Hz

ARR/Onetick(Hz) = RT (Required Time)

ARR = 1*4000

ARR = 4000

500ms:

PSC = 21000

ARR = 0.5 * (ARR of 1 second)

ARR = 2000

Simulation Result

(a) and (c): Monitoring system parameters with changes to temperature.
The monitoring parameters displayed to the PC via UART are described in the UART3 function. The monitoring is executed through a user-defined function that handles the transmission to UART as shown in **Figure 2**.

```
// ************************ START HMS to PC *******************************
// Send monitoring parameters via UART to PC
transmit_UART(header_text, 1);                  // Send '!' header.
transmit_UART(temp_array, 4);                   // Send Temperature Value.
transmit_UART(space_text, 1);                   // Send 0b0011 == 0x20 for space.
transmit_UART(output_status, 4);                // Send Output statuses
transmit_UART(carriage_ret,1);
// Set 1 second Timer
startTIMER(countlsecond);
while((TIM6->SR & TIM_SR_UIF) == 0);
```

Figure 2: Monitoring portion of 'C' code that displays the parameters to satisfy requirements.

To test if the monitoring system is functioning correctly, 3 different ADC inputs are used and expected to be the results shown in **Table 1**.

Input (ADC)	Temperature (℃)	Expected output	Simulation output
950	16.5	!16.5 1001	!16.5 1001
2048	26.5	!26.5 1010	!26.5 1010
1700	23.3	!23.3 1011	!23.3 1011

Table 1: Expected and simulation result of the monitoring system.

In Figure 3 below, the simulation run tests the first input of ADC value 950 (0x3B6) by entering the value into the DR register of the ADC3.

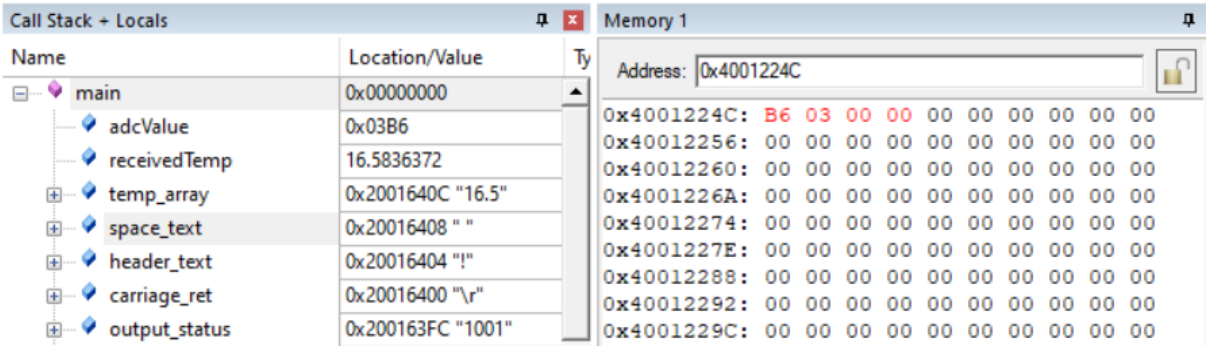

Figure 3: ADC value input of 950 (right) with its corresponding outputs (left).
The next two input values to be tested follow the same procedure as the first input. The input and outputs will be shown in Figure 4 - 5.

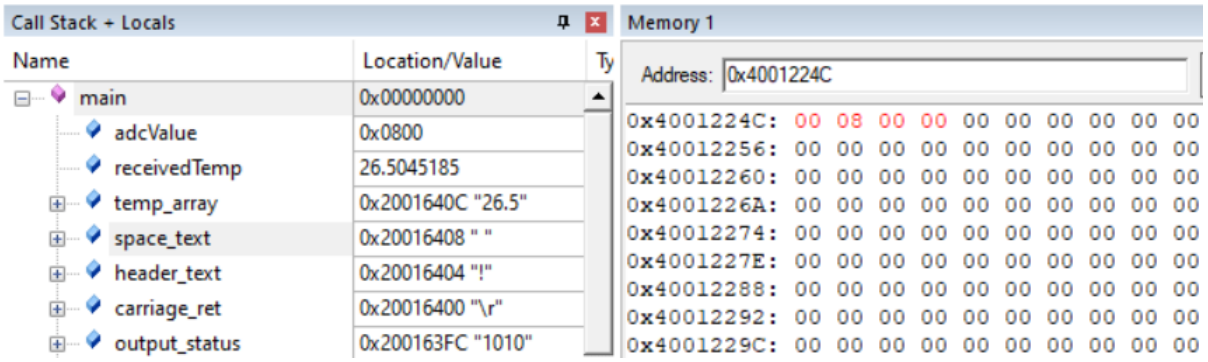

Figure 4: ADC value input of 2048 (right) with its corresponding outputs (left).

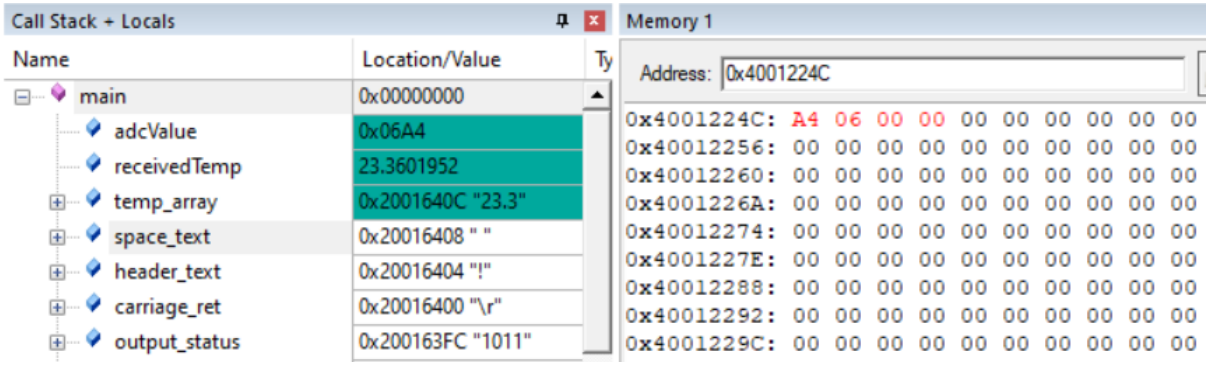

Figure 5: ADC value input of 1700 (right) with its corresponding outputs (left).

Comparing the simulation results to the expected results proves that the monitoring feature of the system functions as expected.

(b): Fan and Light switches operate as ‘toggle’ mode.

To simulate the toggle mode of the switches we test the Fan switch being pressed. The switch input is entered into the IDR register of GPIOA simulating the Fan switch being pressed, we check that the ‘readSWs’ variable matches the ‘fanSW’ variable and will then operate the ‘toggle’. Figure 6 shows the switch input manually entered and the output status displayed through UART.

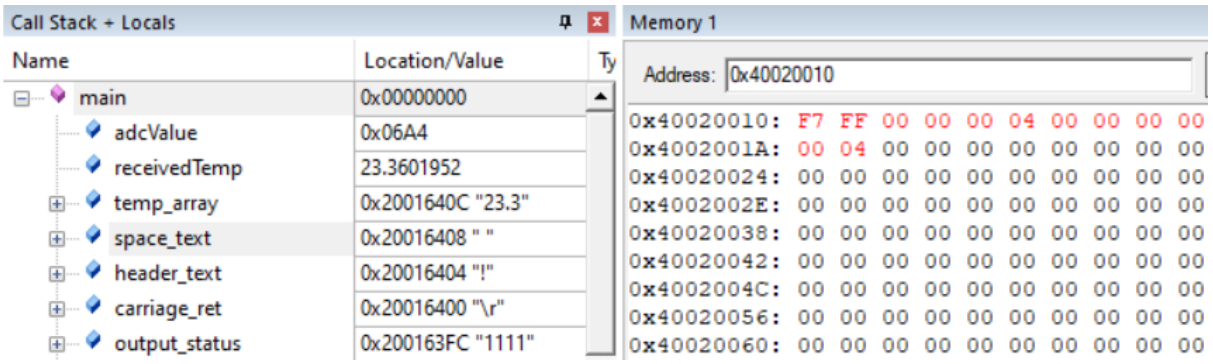

Figure 6: Fan switch value entered in IDR (right) with its corresponding outputs (left).

As the fan switch is pressed it can be seen in the ‘output_status’ array that the fan output is off after pressed. The expected output should now be “1011” after the fan switch is pressed again to toggle on.

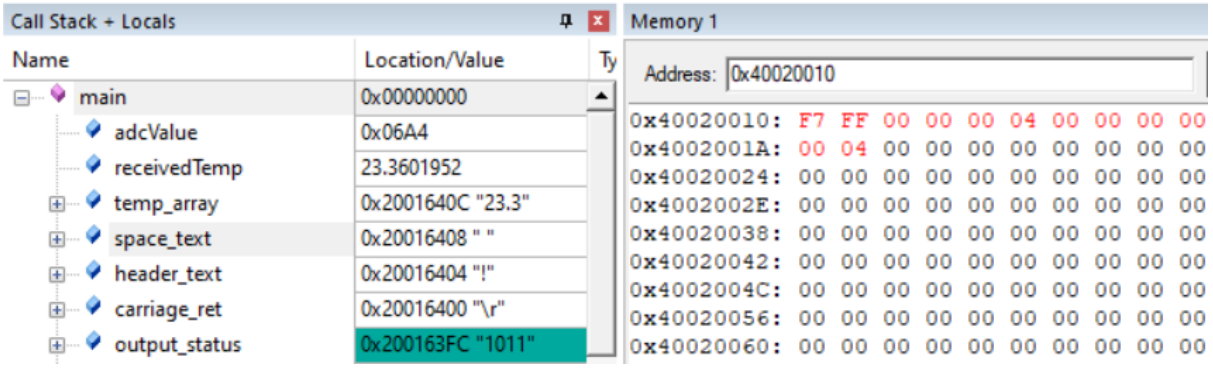

Figure 7: Fan switch toggling ‘on’ after it was toggled ‘off’ beforehand.

Figure 6 displays the output statuses as expected with only the fan parameter being toggled.

(d): Light intensity sensor operation.

The light intensity sensor operates similarly to the switches from part **(b)** with an addition of the light switch pressed at the same time. The following code in Figure 8 demonstrates the operation of the sensor with the different combinations considered to complete the task.

```
if (toggleLight == 0 && (readSWs == lightSensor)) {
  check = 8;
} else if (toggleLight == 1 && (readSWs == lightSensor)) {
  check = 17;
} else if (toggleLight == 0 && (readSWs != lightSensor)){
  check = 0;
} else if (toggleLight == 1 && (readSWs != lightSensor)) {
  check = 9;
}
```

Figure 8: Different operations of the light sensor.

(e): UART light operation.

The light output is to be operated through the UART via entering through entering two bytes in the terminal emulated by Teraterm. This function however was not fully implemented in the project.

(f): UART heating, cooling and fan operation.

The heating, cooling and fan is also to be operated through the UART via entering bytes in the terminal of Teraterm, however this function was not fully implemented.

Discussion and Conclusion

The monitoring for the UART was also observed through the terminal emulated by Teraterm. It provided the user with the current status of the HMS and the current temperature of the set through potentiometer at every 1 second via a delay. The changing of the temperature set and the checking of current devices status is also updated on the terminal. This can be shown in the table further above in the results section. When the temperatures are set to below 22 or above 24, heating and cooling are turned on respectively with the fan. The output status is also updated for the LEDS to luminate. The changing of temperature hence the changing output status can be also shown in the results section. Possible further improvements include not utilising the polling method for creating the delay. Interrupts and multiple TIM peripherals can be used to generate a more accurate response delay.

The light, fan and light intensity sensor were all able to be toggled on and off via pressing of the switch. Due to using the polling method, a proper delay of 0.5 seconds was not created, however the nature of the code required the user to hold the switch sometimes for a short uncertain time to either toggle on or off the switches. This functionality is verified through the simulator by updating the IDR and then checking the output_status linked to the UART which is provided in the results. In the future the possible improvements are to utilise interrupts and use multiple TIM peripherals to provide a more optimised delay for the switches to be more responsive and not use the current polling method. The intensity sensor is also currently a hold press and not a toggle switch which makes for another possible improvement for the future.

One major improvement that could be adhered to in the future is the addition of the operations of light, fan, heating and cooling via an input through the terminal of Teraterm.

Through the correct configuration of all peripherals and the setup and implementation of the functional block diagram, the majority of the required operations of the HMS were fulfilled and verified. These include the monitoring of the UART, changing temperature through the pentionmenter hence certain system devices turning on and off and the toggling and pressing of the switches to toggle on and off system devices. Certain improvements can be made such as incorporating other peripherals such as interrupts and using multiple of the same interrupts and fully implementing all functionality to overall provide a better Home Management System.

Appendices

Code is attached in the submission of the zip file.