```c
#include <avr/io.h>
#include <stdbool.h>
#include <stdint.h>

typedef struct temperature_system_input
{
    float input_temperature_in_celsius;
    float output_temperature_in_celsius;

} temperature_system_input;

typedef struct temperature_system_output
{
    bool input_temperature_is_ok;
    bool output_temperature_is_ok;
} temperature_system_output;


typedef struct output_draying_system
{
    uint8_t heat_flow_in_watts;                    // P(t)
    uint8_t air_flow_velocity_in_feet_per_minute; // Q(t)
} output_draying_system;

typedef struct input_draying_system
{
    float humidity_in_percentage;
} input_draying_system;

float read_temperature_sensor();
void wait_to_analogic_digital_conversion_finish();

float read_input_temperature_sensor_in_celsius()
{
    /*
        ADMUX -- ADC Multiplexer Selection Register

        Segundo a referência os bits  6 e 7 representam o tipo de tensão de referência
        no caso 00, significa que a tensão de referência será proveniente de uma tensão
        externa, que será ligada ao pino AREF

        no caso 11, significa que ele usará a tensão interna de 1.1V como referência

        o bit da posição 5 representa se o resultado do da conversão estará ajustado a dire
        isso porque o resultado da conversão são 10bis e irem precisar de 2 registradores p
        ajustando a esquerda, podemos ler o resultado do primeiro registrador e ignorar os 2
```

```
        tendo em visa que na leitura pode haver ruído,então ignorar os 2 bits pode ser basta

        Os 4 bits menos significativos ou seja os bits da posição 0,1,2,3
        representam as entradas do conversor. No arduino uno elas represetam
        as entradas do sinal
        0000 - ADC[0]
        0001 - ADC[1]
        0010 - ADC[2]
        0011 - ADC[3]
        0100 - ADC[4]
        0101 - ADC[5]
        0110 - ADC[6]
        0111 - ADC[7]

        no caso a entrada escolhida foi a ADC[0]
        e a tensão de referência externa de  é 5V, uma vez que
        podemos representar a tensão da umidade de 0 2.5V e a tensão de 0 até 5V
        dos sensores de temperatura.
        Ou seja os valores de umidade varia de 0 até 512 (1024/2)
        e os valores de temperatura variam de 0 até 1024


        traduzindo a minha escolha para os bits fica
        76
        00 - referência externa

        3210
        0000 -  ADC[0]

        5
        0 - lendo os resistadores da direita para esquerda

    */
    //        76543210
    ADMUX = 0b00000000;

    return read_temperature_sensor();
}

float read_temperature_sensor()
{

    wait_to_analogic_digital_conversion_finish();

    /*
        1024  - 200 graus
```

```
         ADC  - y

          y = 200*ADC/1024
    */

    return 200.0 * (float)ADC / 1024.0;
}

void wait_to_analogic_digital_conversion_finish()
{
    // habilitando o bit 6 do ADCSRA inicializo o processo de conversão
    //          76543210
    ADCSRA |= 0b01000000;

    /*
        no bit 4 representa a flag de interrupção, quando a interrupção ocorrer esse bit va
        ou seja quando derminar o processo de de conversão então sai do while
    */

    //                  76543210
    while (!(ADCSRA & 0b00010000))
    {
    }
}

float read_output_temperature_sensor_in_celsius()
{

    /*

        no caso a entrada escolhida foi a ADC[1]

        traduzindo a minha escolha para os bits fica
        76
        00 - tensão de referência externa

        3210
        0001 -  ADC[1]

        5
        0 - lendo os resistadores da direita para esquerda

    */
    //        76543210
    ADMUX = 0b00000001;
```

```
    return read_temperature_sensor();
}

float read_humidity_sensor_in_percentage()
{
    /*

        no caso a entrada escolhida foi a ADC[2]

        traduzindo a minha escolha para os bits fica
        76
        00 - tensão de referência externa

        3210
        0010 -  ADC[2]

        5
        0 - lendo os resistadores da direita para esquerda

    */
    //        76543210
    ADMUX = 0b00000010;

    wait_to_analogic_digital_conversion_finish();

    /*
        512  - 100 %
        ADC  - y

          y = 100*ADC/512
    */


   float humidity = 100.0 * (float)ADC / 512.0;
    return humidity > 100 ? 100 : humidity;
}

void write_air_flow_in_cubic_feet_per_minute(uint8_t *velocity)
{
  /*
    OCR0A representa a saida do pino PD6 == D6 do arduino uno

    sabendo que o contador vai 0 até 255 e Q(t) vai de 0 até 100, então

    100 - 255
     x  - y
```

```
     x*255/100

    */

   OCR0A = (*velocity) * 255 / 100;

}

void write_heat_flow_in_watts(uint8_t *power)
{

     /*
    OCR0B representa a saida do pino PD5 == D5 do arduino uno

    sabendo que o contador vai 0 até 255 e o P(t) vai de 0 até 100, então:

    100 - 255
     x  - y

     x*255/100

    */

    OCR0B  = (*power) * 255 / 100;
}

void power_led(bool turn_on)
{

    if (turn_on)
        PORTB |= 0b00000001;
    else
        PORTB &= 0b11111110;
}

void low_temperature_led(bool turn_on)
{
    if (turn_on)
        PORTB |= 0b0000100; // turing on led
    else
        PORTB &= 0b11111011; // turing off led
}

void high_temperature_led(bool turn_on)
{
```

```
    if (turn_on)
        PORTB |= 0b00000010; // turing on led
    else
        PORTB &= 0b11111101; // turing off led
}

bool read_switch_button_status()
{
    return PINB >> 3 == 1;
}




void setup()
{
     /*
        Utilizado portas D8,D9,D10 como saída para os leds
        Utilizado a porta D11 para entrada do botão switch
        PINB pinos de entrada na da porta PORTB
        D8, D9 e D10, D11 representa os bits 0,1,2,3 do registrador DDRB


     */
    //       76543210
    DDRB = 0b00000111;


    /*
        PORTD

        são as portas digitais do arduino de  D0 até D7
        quero os pinos do PWM logo, os Pinos 5 e 6
        ou seja
        os bits 5 e 6 deve ser marcados como saída
    */
   //       76543210
   DDRD = 0b01100000;



    /*
        Habilitando todas as portas C do atmega 328p que equivale as portas
        A0,A1,A2,A3,A4,A5 do arduino uno respectivamente.
        As portas A0,A1,A2 serão as entradas dos sensores de temperatura e sensor de umidade
        As portas A0, A1,A2  também se equivale as portas ADC[0], ADC[1] e ADC[2], respectiv
```

```
*/
DDRC = 0b00000000;

/*
  ADCSRA -- significa  ADC Control and Status Register A

  no bit 7 habilita ou não o ADC
  no bit 6 se 1 então começa a conversão se estiver em modo simples
  no bit 5 se 1 então habilita o auto gatilho
  no bit 4 representa a flag de interrupção, quando a interrupção ocorrer esse bit vai p
  no bit 3 habilita a interrupção

  e os 3 bits menos significativos sever para escolher o Prescaler do ADC
  na prática utilizando o atmega a 16MHz a única opção é o 111

  7 - estou habilitando o ADC
  6 - estou informação que não quero começar a conversão
  5 - não quero habilitar a conversão automática ou seja eu que vou sinalizar quando co
  4 - não controlo
  3 - não quero interromper nenhuma conversão a final nem comecei uma.
  2:0 - sou obrigado a usar o o Prescaler 111 por está trabalhando a 16MHz
*/
//         76543210
ADCSRA = 0b10000111;

/*
    ADCSRB -- significa ADC Control and Status Register B
    bit 7 - nenhuma informação encontrada, pode existir só não encontrei / não foi neces
    bit 6 - nenhuma informação encontrada, pode existir só não encontrei / não foi neces
    bit 5 - nenhuma informação encontrada, pode existir só não encontrei / não foi neces
    bit 4 - nenhuma informação encontrada, pode existir só não encontrei / não foi neces
    bit 3 - nenhuma informação encontrada, pode existir só não encontrei / não foi neces


    os 3 últimos bits [2:0]  sevem para definir a fonte do auto gatilho

    000 -- significa que a conversão será contínua, onde ao final de uma conversão outr
    001 -- significa que a conversão se dará a partir de um comparador analógico, ou se
    010 -- significa que a conversão se dará a partir de uma interrupção externa
    011 -- significa que a conversão se dará a partir da comparação de A do TC0
    100 -- significa que a conversão se dará a partir do estouro de contagem do TC0
    101 -- significa que a conversão se dará a partir da comparação de B do TC1
    110 -- significa que a conversão se dará a partir do estouro de contagem do TC1
    111 -- evento de captura do TC1
```

```
    nesse caso habilitei  a fonte do gatilho a partir da comparação do comparador analóg
    dessa forma eu informo quando quero fazer a conversão para digital
*/

//         76543210
ADCSRB = 0b00000001;

/*
     DIDR0 -- Digital Input Disable Register 0

     bit 7 -- nenhuma informação encontrada, pode existir só não encontrei / não foi nec
     bit 6 -- nenhuma informação encontrada, pode existir só não encontrei / não foi nec
     bit 5 --  se 1 então DESABILITA o canal ADC[5]
     bit 4 --  se 1 então DESABILITA o canal ADC[4]
     bit 3 --  se 1 então DESABILITA o canal ADC[3]
     bit 2 --  se 1 então DESABILITA o canal ADC[2]
     bit 1 --  se 1 então DESABILITA o canal ADC[1]
     bit 0 --  se 1 então DESABILITA o canal ADC[0]

     no nosso caso iremos precisar de 3 portas do conversor digital

     ADC[0] -- sensor de temperatura de entrada
     ADC[1] -- sensor de temperatura de saída
     ADC[2] -- sensor de humidade

     portanto os bits  3,4,5 serão 1
     por via das dúvidas irei deixar os bits 6 e 7 sendo 0

*/

//        76543210
DIDR0 = 0b00111000;

/*
TCCRnA -- Timer /Counter Control n Register A

bit  7 -  COM0A1
bit  6 -  COM0A0
bit  5 -  COM0B1
bit  4 -  COM0B0
bit  3 -  ...
bit  2 -  ...
bit  1 -  WGM01
bit  0 -  WGM00

se COM0A1,COM0A0 = 00 então é a operação normal do pino, OC0A desconectado.
```

```
    se COM0A1,COM0A0 = 01 operação normal do pino, OC0A desconectado.
    se COM0A1,COM0A0 = 10 OC0A é limpo na igualdade de comparação. (modo não invertido)
    se COM0A1,COM0A0 = 11 OC0A é ativo na igualdade de comparação e limpo no valor de TC mí

    TCCRnB -- Timer /Counter Control n Register B

    bit  7 -  FOC0A
    bit  6 -  FOC0B
    bit  5 -  ...
    bit  4 -  ...
    bit  3 -  WGM02
    bit  2 -  CS02
    bit  1 -  CS01
    bit  0 -  CS00
3
    FOC0A e FOC0B -  Force Output Compare A e B, quando modo não-PWM, força uma comparação

    CS02,CS01,CS00 seleção do clock. Onde:

    CS02,CS01,CS00 = 000 , então sem Fonte de Clock (TC0 parado)
    CS02,CS01,CS00 = 001 , prescaler = 1
    CS02,CS01,CS00 = 010 , prescaler = 8
    CS02,CS01,CS00 = 011 , prescaler = 64
    CS02,CS01,CS00 = 100 , prescaler = 256
    CS02,CS01,CS00 = 101 , prescaler = 1024

    CS02,CS01,CS00 = 110, Clock externo no pino T0. Contagem na borda de descida
    CS02,CS01,CS00 = 111, Clock externo no pino T0. Contagem na borda de subida


    então para configurar o PWD se utiliza os seguintes registradores:  WGM02,WGM01,WGM00

    WGM02,WGM01,WGM00 = 000, modo de operação TC normal
    WGM02,WGM01,WGM00 = 001, modo de operação TC PWM fase corrigida
    WGM02,WGM01,WGM00 = 010, modo de operação TC CTC
    WGM02,WGM01,WGM00 = 011, modo de operação TC PWM rápido


    A equação da frequencia do PWM rápido é:

    f_pwm = 16MHz/(prescaler*256), usando o prescaler = 256, temos que:
    f_pwm = 234.9624060150376 Hz,um valor acima de 100Hz e abaixo de 100Khz

    Também vamos usar a configuração não invertida, dessa forma a tensão rms
    cresce com o valor do contatador
```

```
    favamos usar o Fast PWM logo, o 3-bit do TCCRnB (WGM02) deve ser 0
    e 1:0-bit do TCCRnB (WGM01,WGM00) = 11

    */

    //         76543210
    TCCR0A = 0b10100011;
    //         76543210
    TCCR0B = 0b00000100;

  Serial.begin(9600);
}

output_draying_system drying_system(input_draying_system *input);
temperature_system_output temperature_system(temperature_system_input *input);
```

```
void loop(){
  temperature_system_input temp_input;

    input_draying_system draying_input;

    bool switch_status = read_switch_button_status();
    temp_input.input_temperature_in_celsius = read_input_temperature_sensor_in_celsius();
    temp_input.output_temperature_in_celsius = read_output_temperature_sensor_in_celsius();

    /*
            toda a regra de negócio utilizada para decidir se a temperatura está ou não
            de acoordo está definida no módulo temperature system
            é nele que se verifica se a tempertura de entrada e saída estão de acordo com
    */
    temperature_system_output temp_output = temperature_system(&temp_input);

    power_led(switch_status);
    low_temperature_led(switch_status && !temp_output.input_temperature_is_ok);
    high_temperature_led(switch_status && !temp_output.output_temperature_is_ok);

    if (!switch_status || !temp_output.input_temperature_is_ok || !temp_output.output_temper
    {
        output_draying_system draying_output = {0, 0};
        write_air_flow_in_cubic_feet_per_minute(&draying_output.air_flow_velocity_in_feet_pe
```

```
        write_heat_flow_in_watts(&draying_output.heat_flow_in_watts);
        return;
    }

    draying_input.humidity_in_percentage = read_humidity_sensor_in_percentage();

    /*
             o modulo drying_system, é onde está localizado toda a regra de negócio
             que determina as velocidade do secador e a potência do seu aquecedor.
    */
    output_draying_system draying_output = drying_system(&draying_input);

    write_air_flow_in_cubic_feet_per_minute(&draying_output.air_flow_velocity_in_feet_per_mi

    write_heat_flow_in_watts(&draying_output.heat_flow_in_watts);

}
output_draying_system drying_system(input_draying_system *input)
{

  output_draying_system output;


    if (input->humidity_in_percentage >= 100)
    {
        output.air_flow_velocity_in_feet_per_minute = 25;
        output.heat_flow_in_watts = 100;
    }

    else if (input->humidity_in_percentage >= 50)
    {
        output.air_flow_velocity_in_feet_per_minute = 50;
        output.heat_flow_in_watts = 50;
    }

    else if (input->humidity_in_percentage >= 25)
    {
        output.air_flow_velocity_in_feet_per_minute = 100;
        output.heat_flow_in_watts = 25;
    }

    else if (input->humidity_in_percentage >= 0)
    {
        output.air_flow_velocity_in_feet_per_minute = 0;
        output.heat_flow_in_watts = 0;
    }
```

```
    return output;
}

temperature_system_output temperature_system(temperature_system_input *input)
{
    temperature_system_output output;

    if (input->input_temperature_in_celsius < 20)
        output.input_temperature_is_ok = false;
    else
        output.input_temperature_is_ok = true;

    if (input->output_temperature_in_celsius > 120)
        output.output_temperature_is_ok = false;
    else
        output.output_temperature_is_ok = true;

    return output;
}
```

Em relação as medidas de temperatura e umidade, Utilizei uma tensão de referência externa de 5V dessa forma o sensor de umidade, perde um pouco de sua resolução, mas tendo em vista que: - $2.5/1024 = 0.0024$ - $2.5/512 = 0.0049$

então não existe uma perda significativamente grande.