Федеральное государственное автономное
образовательное учреждение
высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт инженерной физики и радиоэлектроники
Кафедра «Радиотехника»

КУРСОВАЯ РАБОТА

Разработка устройства автоматического регулирования температуры

Руководитель _____ С. В. Сизасов
подпись, дата инициалы, фамилия

Студент РФ13-32Б 051201374 _____ В. И. Зуевский
номер группы номер зачетной книжки подпись, дата инициалы, фамилия

Студент РФ13-32Б 051201548 _____ М. Е. Забродин
номер группы номер зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2016

СОДЕРЖАНИЕ

Техническое задание

1. Разработать устройство автоматического регулирования температуры(термостат).
2. Реализовать индикацию текущей температуры и влажности используя семисегментный индикатор.
3. Переключение типа выводимой информации должно осуществляться с кнопки.
4. Ограничить диапазон рабочих температур контроллера.
5. Реализовать возможность задания температуры при отклонении от которой на 1.5 градуса Цельсия будет происходить включение охлаждающей или обогревающей системы.

1 Анализ технического задания

Для выполнения поставленной задачи необходимо:

1. Выбрать необходимые электрические компоненты для решения поставленной задачи.
2. Составить электрическую принципиальную схему разрабатываемого устройства
3. Проанализировать способ взаимодействия между выбранными электрическими компонентами.
4. Составить блок-схему работы устройства и реализовать программу.
5. Проверить работоспособность разрабатываемого устройства в САПР Proteus.
6. Собрать устройство на макетной плате

2 Ход работы

2.1 Выбор электрических компонентов

Проанализировав различные электронные компоненты, был сделан вывод: для решения поставленной задачи необходимо использовать микроконтроллер Atmel ATmega16a(рисунок 1), датчик влажности и температуры DHT22 (рисунок 2) позволяющий производить измерение температуры в диапазоне от -40 до +125 с точностью ± 0.5 градусов по Цельсию, драйвер MAX7219CNG (рисунок 3) для оптимизации работы с семисегментым индикатором работающий по шине данных SPI, однотактные кнопки для управления устройством, светодиоды.

Рисунок 1 — Микроконтроллер ATmega16a

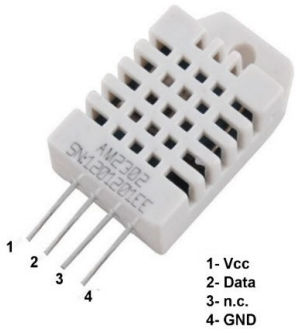

Рисунок 2 — Датчик температуры и влажности DHT22

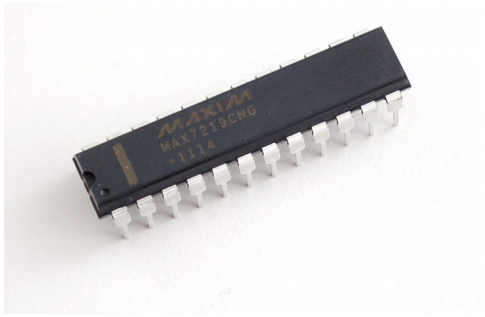

Рисунок 3 — Драйвер MAX7219

2.2 Электрическая принципиальная схема

Перед разработкой программы устройству, необходимо составить электрическую принципиальную схему, для точного понимания того, какие выводы МК потребуются для работы. На рисунке 4 представлена электрическая принципиальная схема разрабатываемого устройства.

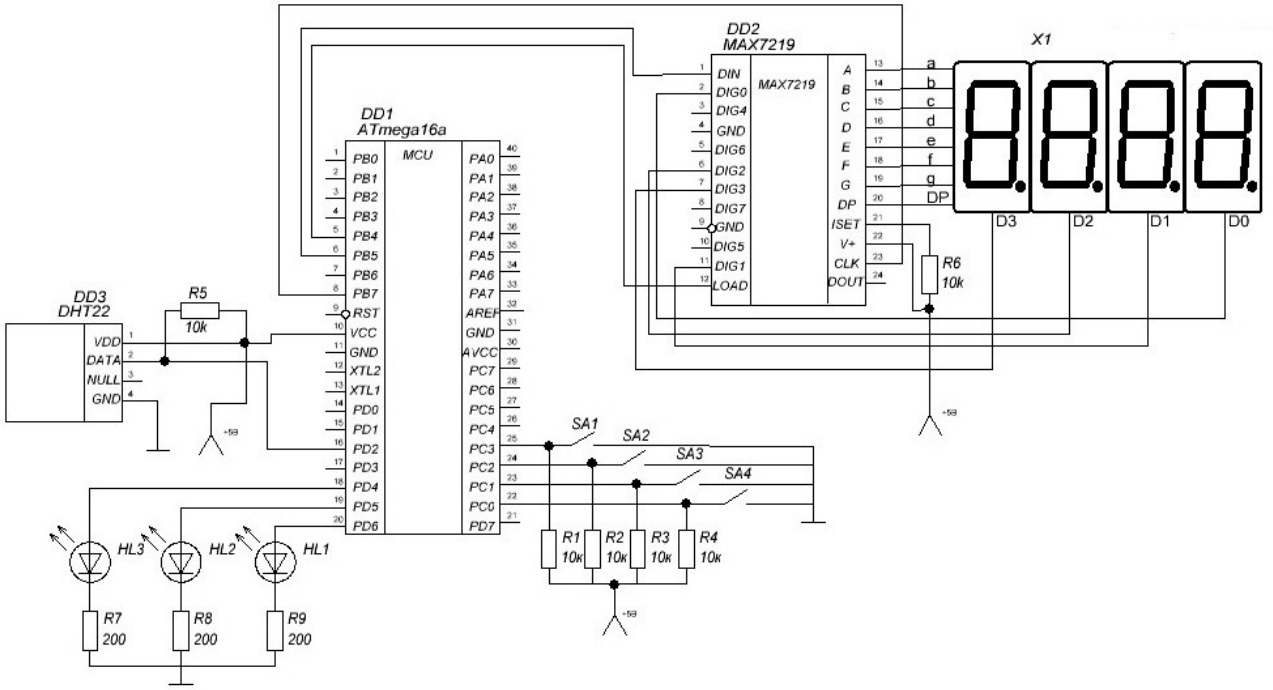

Рисунок 4 — Электрическая принципиальная схема устройства

В соответствии со схемой (рисунок 4), для подключения к МК светодиодов необходимо задействовать выводы PD4-PD6; для подключения кнопок управления выводы PC0-PC3, для подключения MAX7219CNG выводы PB4, PB5, PB7; для подключения датчика влажности и температуры DHT22 вывод PD2.

Проведя анализ составленной схемы можно сделать вывод о точном количестве необходимых электрических компонентов:

1. Atmel ATmega16a — 1шт.
2. MAX7219CNG – 1 шт.
3. Четырёх разрядный семисегментный индикатор — 1шт.
4. Датчик влажности и температуры — 1 шт.
5. Резистор 10 кОм — 6 шт.
6. Резистор 200 Ом — 3 шт.
7. Однотактная кнопка — 4 шт.

2.3 Анализ методов взаимодействия между выбранными компонентами

2.3.1 Датчик влажности и температуры DHT22

На рисунке 5 показан протокол передачи данных по шине Single-bus между микроконтроллером и датчиком температуры и влажности DHT22. На рисунке 5 показано, что данные от датчика DHT22 идут старшим битом вперед, порядок байт таков:

1. Старший байт влажности.
2. Младший байт влажности.
3. Старший байт температуры.
4. Младший байт температуры.
5. Контрольная сумма — сумма всех предшествующий байт.

Отрицательную температуру датчик представляет в виде абсолютного значения температуры с установленным старшим битом в старшем байте в единицу. Значения влажности и температуры приходят умноженными на 10, для передачи дробной части значений.

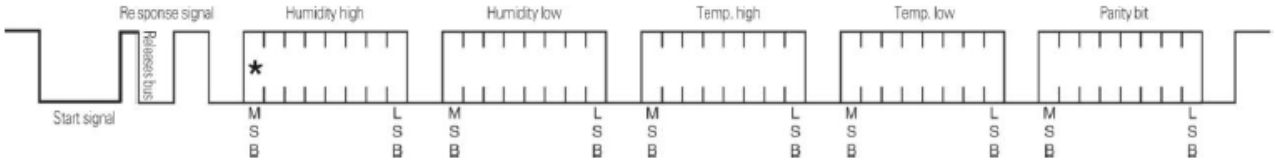

Рисунок 5 — Протокол передачи данных по шине Single-bus

Описание протокола взаимодействия между микроконтроллером и датчиком DHT22:

1. Микроконтроллер запрашивает измерения у датчика DHT22 устанавливая на шине данных 0 на 0.8-1 мс (запрос данных допустимо производить не чаще чем раз в 2 секунды).
2. Микроконтроллер устанавливает на шине данных 1 на 80мкс.
3. Датчик отвечает 0 на шине данных на 75-85мкс.
4. Датчик отвечает 1 на шине данных на 75-85мкс и начинает передавать данные с периодичностью 50мкс.
5. В зависимости от длительности импульса, каждый бит — это 0 в течение 22-30 мкс и 1 в течение 68-75 мкс.
6. После окончания передачи данных, в течении 45-55мкс датчик устанавливает 1 на шине данных.

2.3.2 Драйвер семисегментного индикатора MAX7219CNG

Драйвер MAX7219CNG управляется по последовательной шине SPI. На рисунке 6 представлен формат отправляемых данных в драйвер семисегментного индикатора MAX7219CNG. Биты D15-D12 не несут в себе полезной информации, будем отправлять в них 0; в поле ADRES указывается, что необходимо сделать:

1. Если в ADRES передается значение 1…8 (0001…1000), то это выбор знакоместа. В поле DATA в этом случае передается информация о сегментах выбранного знакоместа.
2. Если в ADRES передается значение 9…15 (1001…1111), то это указание выполнить некоторую служебную инструкцию.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADRES				DATA							

Рисунок 6 — Формат отправляемых данных в MAX7219CNG

Для нормальной работы микросхемы её необходимо инициализировать после подачи питания. Инициализация подразумевает некоторую последовательность команд, после которой микросхема переходит в рабочий режим и начинает реагировать на команды и данные. Необходимые действия для инициализации микросхемы:

1. Выйти из спящего режима.
2. Задать количество используемых сегментов.
3. Выбрать режим декодирования.
4. Задать интенсивность свечения.
5. Сбросить данные находящиеся в используемых сегментах.

2.4 Разработка программы для устройства

Для корректной работы устройства при подаче питания необходимо произвести начальную инициализацию: Задать вектора прерываний, инициализировать стек, настроить выводы МК в соответствии с назначением, произвести инициализацию MAX7219CNG (рисунок 13), задать параметры таймера который будет отмерять промежутки между измерениями, задать стандартное значение поддерживаемой температуры. На блок-схеме (рисунок 7) показан алгоритм начальной инициализации устройства. По окончании инициализации устройство переходит к выполнению основного цикла (рисунок 8).

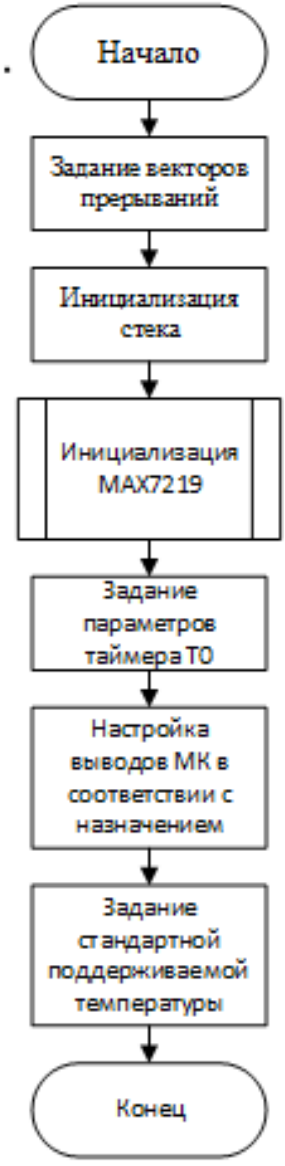

Рисунок 7 — блок-схема «инициализация устройства при включении»

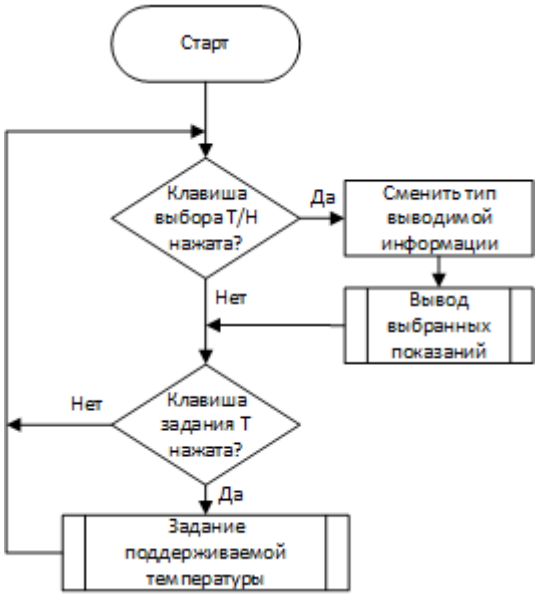

Рисунок 8 — блок-схема «основной цикл»

На рисунке 8 представлена блок-схема основного цикла программы. На блок-схеме показана логика обработки нажатия клавиш управления. При нажатии клавиши выбора Т/Н, происходит смена типа выводимой информации и последующий вызов подпрограммы (рисунок 9) её вывода на семисегментный индикатор; при нажатии клавиши SET, происходит переход в подпрограмму задания поддерживаемой температуры (рисунок 10).

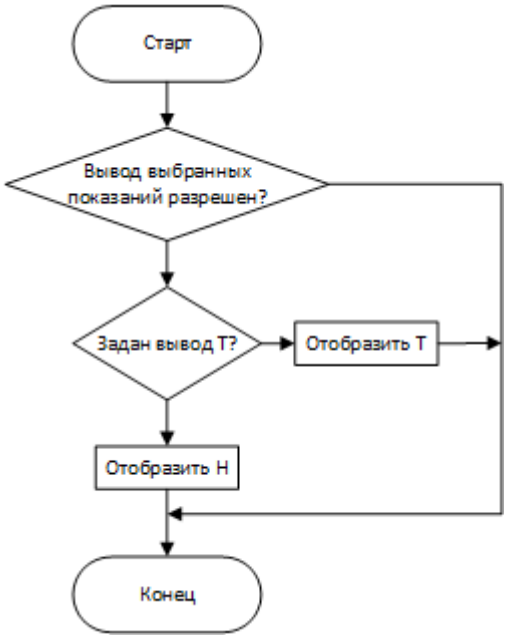

Рисунок 9 — блок-схема «вывод выбранных показаний»

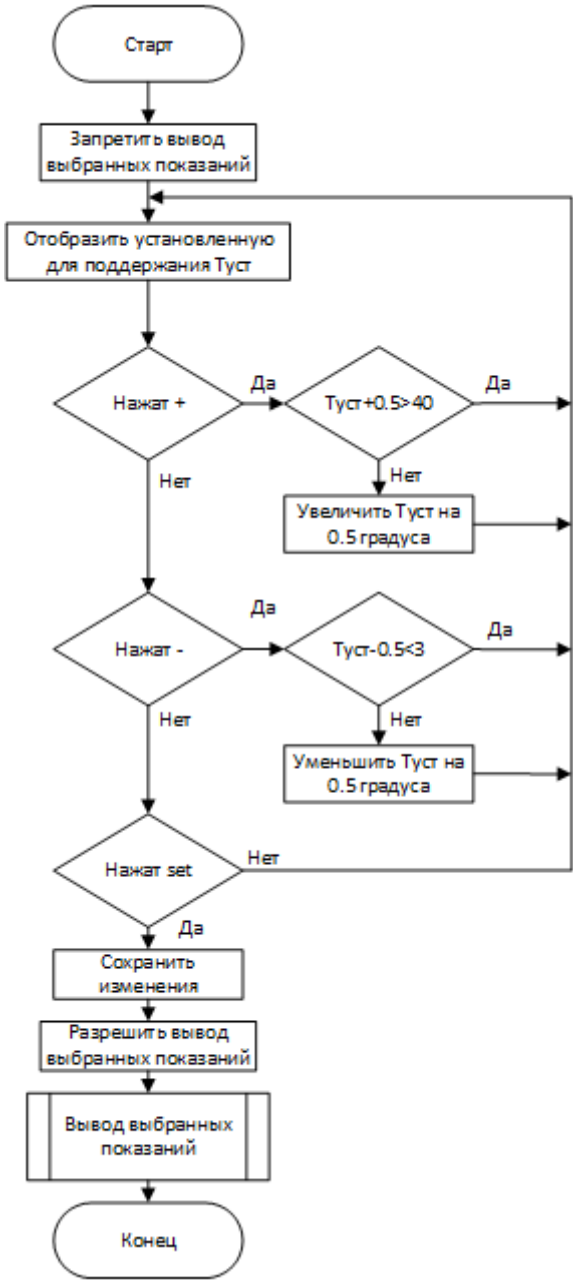

Рисунок 10 — блок-схема «задание поддерживаемой температуры»

На рисунке 10 приведена блок схема алгоритма задания поддерживаемой температуры. Переход к выполнению данного алгоритма происходит после нажатия клавиши SET. Вначале выполнения данного алгоритма происходит запрет отображения выбранных показаний T/H, после чего происходит отображение установленной для поддержания температуры ($T_{уст}$). Далее в зависимости от нажатой клавиши происходит увеличение, уменьшение, либо сохранение поддерживаемой температуры. Устанавливаемая для поддержания температура находится в диапазоне от 3 до +40 градусов по Цельсию. По завершении сохранения устанавливаемой температуры происходит разрешение вывода и сам вывод выбранных показаний.

С периодичностью в 5 секунд от прерывания по переполнению таймера T0 происходит вызов подпрограммы опроса датчика влажности и температуры DHT22. Блок-схема алгоритма опроса датчика представлена на рисунке 11. При успешном получении данных от датчика происходит сохранение новых значений влажности и температуры, вывод выбранных показаний (рисунок 9) и проверка нового значения температуры (рисунок 12). В случае получения некорректных данных производится повторный замер при следующем прерывании.

Рисунок 11 — блок-схема «опрос датчика»

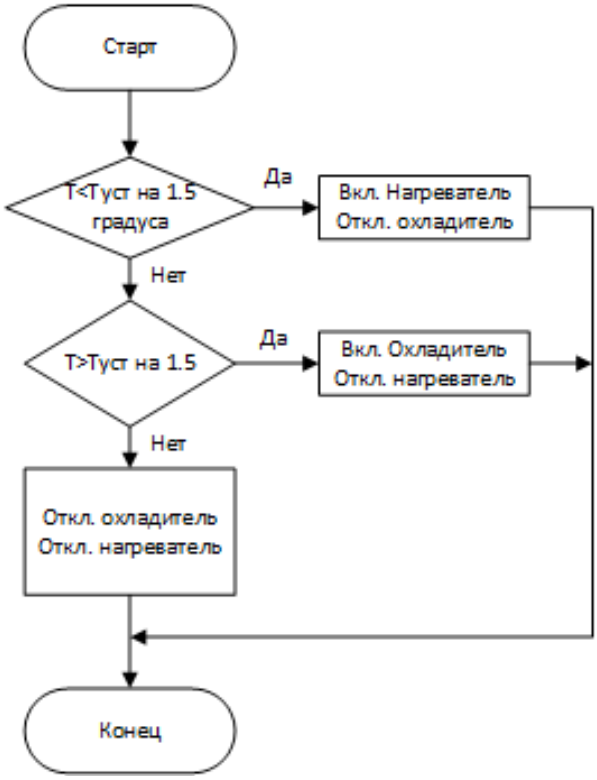

Рисунок 12 — блок-схема «проверка температуры»

На рисунке 12 показана блок-схема алгоритма проверки температуры на принадлежность диапазону от $T_{уст}$-1.5 до $T_{уст}$+1.5. В случае если температура принадлежит заданному диапазону, происходит отключение охладителя и нагревателя. В случае отклонения в меньшую, либо большую сторону происходит включение нагревателя, либо включение охладителя соответственно.

Рисунок 13 — блок-схема «инициализация MAX7219CNG»

На рисунке 13 представлена блок-схема инициализации MAX7219CNG. Для корректного взаимодействия с микросхемой модуль SPI микроконтроллера должен быть настроен на работу в режиме SPI Mode 0 (при отсутствии передачи данных, на шине SCK установлен 0; по переднему фротну на SCK происходит установка данных на шину, по заднему фиксация значения). Далее отправляется команда перевести MAX7219CNG во включенное состояние, задание интенсивности свечения сегментов, задание количества активных разрядов, выбор режима декодирования, отображение знака минус во все активные разряды.

В соответствии с приведенными выше блок-схемами разработана программа для устройства автоматического регулирования температуры. Код программы приведет в приложении А.

2.5 Сборка устройства

На рисунке 14 представлен макет спроектированного устройства. На макетной плате расположились 3 светодиода, 4 однотактных кнопки, микроконтроллер, драйвер семисегментного индикатора, семисегментный индикатор, датчик влажности и температуры, несколько резисторов. Для соединения компонентов между собой использовались специально обученные медные провода взятые из кабеля типа витая пара, а также провода для макетирования.

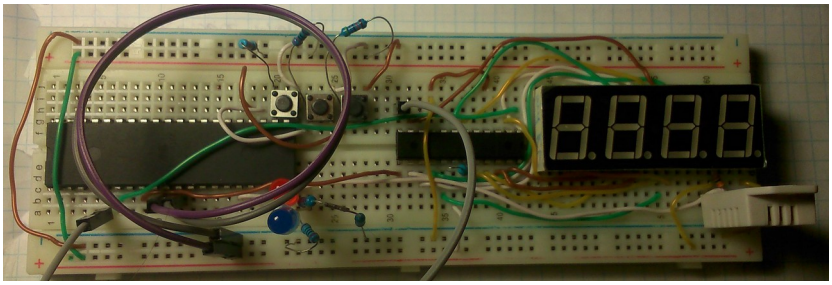

Рисунок 14 – Устройство в сборе на макетной плате

Для загрузки разработанной программы в устройство использовался программатор USBASP (рисунок 15).

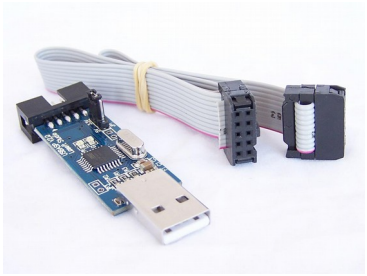

Рисунок 15 - программатор USBASP

После загрузки разработанной программы устройство перешло в рабочее состояние индицируя текущую температуру на семисегментном индикаторе, а также превышение заданной по умолчанию температуры в большую сторону красным светодиодом (рисунок 16).

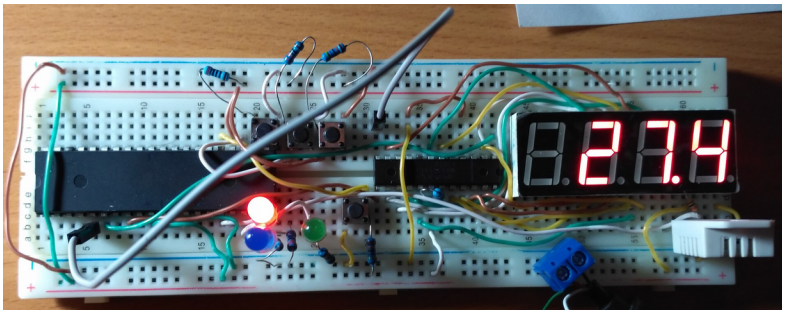

Рисунок 16 — Собранное устройство во включенном состоянии

Заключение

В ходе выполнения курсовой работы было спроектировано, собрано и протестировано устройство автоматического регулирования температуры. Диапазон температур задаваемых для поддержания лежит в пределах от +3 до +40 градусов Цельсия. Задание температуры для поддержания осуществляется с клавиш управления (SET, +, -); выбор выводимой информации на семисегментный индикатор осуществляется нажатием на клавишу выбора выводимой информации(T/H), зажигание светодиода, подключенного к выводу PD4 МК, показывает, что выбрано отображение влажности, отключенный светодиод показывает, что выбран вывод температуры. Допустимым отклонением температуры окружающей среды от заданной для поддержания, является отклонение на 1.5 градуса в большую, либо меньшую сторону. В зависимости от отклонения температуры в большую, либо меньшую сторону включается охладитель, либо нагреватель .

Спроектированное устройство допустимо использовать при температуре окружающей среды в пределах от 0 до +70 градусов Цельсия, данные ограничения вносит драйвер семисегментого индикатора MAX7219CNG. Диапазон рабочих температур можно расширить заменой MAX7219CNG на MAX7219ENG. Для питания устройства необходим источник стабилизированного напряжения с выходным напряжением в промежутке 4.8В до 5.2В.

В ходе тестирования установлено, что устройство стабильно работает при различной температуре окружающей среды, диапазон температур в которых производилось тестирование от -30 до +30 градусов по Цельсию.

Разработанная программа занимает 1026 байт программной памяти контроллера, что составляет 6.8% от общего объема программной памяти контроллера. Данное замечание позволяет при необходимости, заменить микроконтроллер ATmega16A на другой, более дешевый, микроконтроллер из той же серии с меньшим объемом памяти.

Список использованных источников

1. Ревич Ю. В. - Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. - 3-е издание., испр.- Спб.: БХВ-Петербург, 2014. - 368с. ил. - (Электроника)
2. Евстифеев А. В. - Микроконтроллеры AVR семейства Tiny и Mega фирмы ATMEL, 5-е изд., стер. - М.: Издательский дом «Додэка-XXI», 2008.-560с.
3. Atmel Corporation. / Rev.: Atmel-8154C-8-bit-AVR-ATmega16A_Datasheet-07/2014.-335c.
4. AOSONG Electronics Co, Ltd. – Temperature and humidity module. AM2302 Product Manual.-11c.
5. AVR. Учебный курс. // Easy Electronics Электроника для всех – Режим доступа: http://easyelectronics.ru/

Приложение А — Исходный код программы

```
.include "m16adef.inc"   ; Используем ATMega16A

.equ LOAD = PB4
.equ MOSI = PB5
.equ SCK = PB7
.equ DHT22 = PD2
.def toWriteSPI=r25

.equ setbtn = PC0
.equ minusbtn = PC1
.equ plusbtn = PC2
.equ select_btn = PC3

.equ hotledpin = PD6
.equ iceledpin = PD5
.equ typeledpin = PD4

.def tmp=r21
.def razr0=r22
.def razr1=r23
.def razr2=r24

.def flags = r27

.def TIM0counter = r26
.equ TIM0delay = 152

;= Start macro.inc ================================

.MACRO MAX7219_Write2
/*
 * Отправляет данные MAX7219/MAX7221
 * @0 Адрес, @1 значение (берется из памяти)
 */
       cbi PORTB, LOAD
       ldi toWriteSPI, (@0)
       call SPI_Send_byte
       lds toWriteSPI, (@1)
       call SPI_Send_byte
       sbi PORTB, LOAD
.ENDM

;= End  macro.inc =================================


; RAM =============================================
.DSEG          ; Сегмент ОЗУ
Temperature:          .byte 2
Humidity:             .byte 2
SegmentsData:         .byte 4
DHT22_counter:        .byte 1
DHT22_data:           .byte 5
Temperature_set:      .byte 2


; FLASH ===========================================
.CSEG          ; Кодовый сегмент
jmp RESET; Reset Handler
jmp readDHT22_INT0; EXT_INT1 ; IRQ0 Handler
jmp 0; IRQ1 Handler
jmp 0; TIM2_COMP ; Timer2 Compare Handler
jmp 0; TIM2_OVF ; Timer2 Overflow Handler
jmp 0; TIM1_CAPT ; Timer1 Capture Handler
```

```
jmp 0; TIM1_COMPA ; Timer1 CompareA Handler
jmp 0; TIM1_COMPB ; Timer1 CompareB Handler
jmp 0; TIM1_OVF ; Timer1 Overflow Handler
jmp TIM0_POLL; TIM0_OVF ; Timer0 Overflow Handler
jmp 0; SPI_STC ; SPI Transfer Complete Handler
jmp 0; USART_RXC ; USART RX Complete Handler
jmp 0; USART_UDRE ; UDR Empty Handler
jmp 0; USART_TXC ; USART TX Complete Handler
jmp 0; ADC ; ADC Conversion Complete Handler
jmp 0; EE_RDY ; EEPROM Ready Handler
jmp 0; ANA_COMP ; Analog Comparator Handler
jmp 0; TWSI ; Two-wire Serial Interface Handler
jmp 0; jmp EXT_INT2 ; IRQ2 Handler
jmp 0; jmp TIM0_COMP ; Timer0 Compare Handler
jmp 0; jmp SPM_RDY ; Store Program Memory Ready Handler


RESET:
ldi r16,high(RAMEND)  //
out SPH,r16           //
ldi r16,low(RAMEND)   //
out SPL,r16           // располагаем стек вконце озу

call init_MAX7219

ldi tmp, (1<<TOIE0)
out TIMSK, tmp
ldi TIM0counter, TIM0delay
ldi tmp, 0b00000101   // таймер с предделителем 1:1024
out TCCR0, tmp

SET // чтобы ничего не выводилось пока не будут получены первые замеры
sei // разрешить прерывания

ldi tmp, (1<<hotledpin)|(1<<iceledpin)|(1<<typeledpin)|(1<<DHT22)
out DDRD, tmp
sbi PORTD, DHT22

ldi tmp, 0x00                       // дефолтное значение заданной температурки
sts Temperature_set+0, tmp
ldi tmp, 0xFA
sts Temperature_set+1, tmp

loop:
        brts skip // пока установлен флаг скипаем запрос данных от датчика
        selectoutputdata:
        sbic PINC, select_btn
                rjmp skip
        release_select_btn:
        call delay100ms
        sbis PINC, select_btn
        rjmp release_select_btn
                ldi tmp, (1<<typeledpin)
                in r16, PORTD
                EOR tmp, r16          // исключающее или, вкл/выкл диодик
                out PORTD, tmp
                call viewselected
        skip:
        sbic PINC, setbtn
                rjmp selectoutputdata
        release_set:
        call delay100ms
        sbis PINC, setbtn
        rjmp release_set
        call settemp
rjmp loop
```

```
/*
Вывод влажности или температуры
на 7 сегм
*/

viewselected:
sbrc flags, 0                // пропускаем вывод если установлен бит запрета вывода
rjmp viewselected_exit
sbis PORTD, typeledpin       // если 1, то выводим влажность
rjmp viewselected_load_T
       lds r16, Humidity+0
       lds r17, Humidity+1
       rjmp viewselected_view
viewselected_load_T:
       lds r16, Temperature+0
       lds r17, Temperature+1
viewselected_view:
       call viewNumber
viewselected_exit:
ret

/*
Установка удерживаемой температуры
*/

settemp:
       sbr flags, (1<<0) // 0 бит - флаг запрета обновления информации кроме той что сейчас
задаем
       lds r16, Temperature_set+0 // H
       lds r17, Temperature_set+1 // L      T_set 3..40 градусов  //
       call viewNumber
       mov r18, r16                        //H T_tmp
       mov r19, r17                        //L T_tmp

       settemp_viewnum:
              cp r17, r19
              cpc r16, r18
              breq checkplusbtn // если равно, то не нужно повторно выводить
              mov r16, r18                        //H T_tmp
              mov r17, r19
           call viewNumber
       checkplusbtn:         // при нажатии кнопки на ногу приходит 0!
       sbic PINC, plusbtn
              rjmp checkminusbtn
       release_plus:
              call delay100ms
       sbis PINC, plusbtn
              rjmp release_plus
              rjmp plus_pressed
       checkminusbtn:
       sbic PINC, minusbtn
              rjmp checksetbtn
       release_minus:
              call delay100ms
       sbis PINC, minusbtn
              rjmp release_minus
              rjmp minus_pressed
       checksetbtn:
       sbic PINC, setbtn
              rjmp settemp_viewnum
       release_setbtn:
              call delay100ms
       sbis PINC, setbtn
              rjmp release_setbtn
              rjmp set_pressed
```

```
        plus_pressed:
                ldi tmp, 5
                add r19, tmp
                clr tmp
                adc r18, tmp // увеличиваем на 0.5

                ldi r21, 0x90 // L
                ldi r20, 0x01 // H 400 = 40.0 - верхний предел

                cp r21, r19
                cpc r20, r18
                brsh settemp_viewnum // сохранить, если не больше 40.0
                subi r19, 5
                sbci r18, 0 // не даем превысить предел!*/
        rjmp settemp_viewnum

        minus_pressed:
                ldi r21, 0x1E // L
                ldi r20, 0x00 // H -150 = -15,0 - верхний предел
                subi r19, 5
                sbci r18,   0 // увеличиваем на 0.5
                cp r19, r21
                cpc r18, r20
                brsh settemp_viewnum // сохранить, если не меньше 3
                ldi tmp, 5
                add r19, tmp
                clr tmp
                adc r18, tmp // увеличиваем на 0.5
        rjmp settemp_viewnum

        set_pressed:
                sts  Temperature_set+0, r18
                sts  Temperature_set+1, r19
                cbr flags, (1<<0)
                call viewselected
ret

/*
Проверка температуры
в перспективе сюда надо число передавать из __пока не ясно откуда__
*/

checktemp:
        lds r16, Temperature+0 //h
        lds r17, Temperature+1 //l
        lds r18, Temperature_set+0
        lds r19, Temperature_set+1

        sbrc r16, 7                    //
         rjmp lesstemp // число отрицательное точно меньше минимально допустимой

        ldi tmp, 15
         add r19, tmp
        ldi tmp, 0
         adc r18, tmp
        cp r19, r17
        cpc r18, r16
        brlo largertemp  // если установленная+1.5 меньше текущей

        subi r19, 30
        sbci r18, 0
        cp r19, r17
        cpc r18, r16
        brlo oktemp           // если установленная -1.5 меньше текущей

lesstemp:
```

```
        in tmp, PORTD
        sbr tmp, (1<<iceledpin)
        cbr tmp, (1<<hotledpin)
        out PORTD, tmp
        ret

largertemp:
        in tmp, PORTD
        cbr tmp, (1<<iceledpin)
        sbr tmp, (1<<hotledpin)
        out PORTD, tmp
        ret

oktemp:
        in tmp, PORTD
        cbr tmp, (1<<hotledpin)|(1<<iceledpin)
        out PORTD, tmp
        ret

/*
*       БЛОК КОДА ДЛЯ
*       реализации работы
*       DHT22
*/

TIM0_POLL:
dec TIM0counter
breq getMeashurments
reti

getMeashurments:
               push razr2
               push razr1
               push razr0
               push tmp

               ldi razr2, 0x00
               ldi razr1, 0x03
               ldi razr0, 0x20
               sbi DDRD, DHT22
               cbi PORTD, DHT22
        call Delay                      // Просаживаем линию на 800мкс, F=8Mhz
               sbi PORTD, DHT22
               ldi razr2, 0x00
               ldi razr1, 0x00
               ldi razr0, 0x30
        call Delay                      // Отпустили линию ~30мкс [по даташиту20-40мкС]
               cbi DDRD, DHT22
               sbi PORTD, DHT22
               ldi razr2, 0x00
               ldi razr1, 0x00
               ldi razr0, 0x50
        call Delay                      // Через 50 мкс на линии точно должен быть 0
               sbic PIND, DHT22
                      rjmp DHT22err
               ldi razr2, 0x00
               ldi razr1, 0x00
               ldi razr0, 0x80
        call Delay                      // Через 80 мкс на линии точно должен быть 0
               sbis PIND, DHT22
                      rjmp DHT22err
               set                      // ставим флаг T, запрещаем вызов лишнего
```

```
        ldi tmp, 40           // нужно получить 40 бит(5 байт)
                sts DHT22_counter, tmp //r21
                clr tmp
                sts DHT22_data+0, tmp  //r16
                sts DHT22_data+1, tmp
                sts DHT22_data+2, tmp  // -
                sts DHT22_data+3, tmp
                sts DHT22_data+4, tmp  //r20

                in tmp, MCUCR
                sbr tmp, (1<<ISC00)|(1<<ISC01) // Прерывание по восходящему фронту
                out MCUCR, tmp
                in tmp, GICR
                sbr tmp, (1<<INT0)                    // вкл int0
                out GICR, tmp
        pop tmp
        pop razr0
        pop razr1
        pop razr2
reti

DHT22err:
        ldi TIM0counter, TIM0delay
        pop tmp
        pop razr0
        pop razr1
        pop razr2
reti

readDHT22_INT0:
                push razr2
                push razr1
                push razr0
                push r16
                push r17
                push r18
                push r19
                push r20              // бэкап регистров
                push r21              //
                lds r16, DHT22_data+0
                lds r17, DHT22_data+1
                lds r18, DHT22_data+2
                lds r19, DHT22_data+3
                lds r20, DHT22_data+4 // Загружаем из ОЗУ то что уже напринимали, ну либо нули
если первого ждем
                lds r21, DHT22_counter

                ldi razr2, 0x00
                ldi razr1, 0x00
                ldi razr0, 0x50
                call Delay //  50 мкс

                sbis PIND, DHT22      // если до сих пор 1, значит пришла единица
                        rjmp readDHT22_set0

readDHT22_set1:
        sec             // 1 в флаг переноса
        rjmp readDHT22_rol

readDHT22_set0:
        clc             // 0 в флаг переноса
```

```
readDHT22_rol:
        rol r16                 // parity
                rol r17         // t low
                rol r18         // t high
                rol r19         // h low
        rol r20                 // h high
        dec r21
        cpi r21, 0 // r21=0 => прошли все 40 бит!
                breq readDHT22_endReceive
        sts DHT22_counter, r21
        sts DHT22_data+0, r16
                sts DHT22_data+1, r17
                sts DHT22_data+2, r18
                sts DHT22_data+3, r19
        sts DHT22_data+4, r20 // вернули все обратно в ОЗУ
        rjmp readDHT22_POP
reti

readDHT22_endReceive:
        in       tmp, MCUCR
        cbr             tmp, (1<<ISC00)|(1<<ISC01)
         out    MCUCR, tmp              ;отключаем прерывание INT0
        in              tmp, GICR                   ;
        cbr             tmp, (1<<INT0)              ;
     out        GICR, tmp

    sub       r16, r17
    sub       r16, r18
    sub       r16, r19
        sub       r16, r20
        cpi r16,        0 // Если 0, то все ок, загружаем результаты в озу
                breq    readDHT22_validation_OK
        rjmp    readDHT22_finished

readDHT22_validation_OK:
        sbrs r18, 7
                rjmp t_notneg  // датчик хранит отрицательные числа в сомнительном виде
                cbr r18, (1<<7) // абсолютное значение + знак минуса установкой 7 бита старшего
байте
                com r18         // приводим к человеческому виду - доп.код
                com r17
                ldi tmp, 1
                 add r17, tmp
                ldi tmp, 0
                 adc r18, tmp
        t_notneg:
        sts       Humidity+0,r20              // H
        sts       Humidity+1,r19              // L
        sts       Temperature+0,r18    // H
        sts       Temperature+1,r17    // L
        call viewselected
        call checktemp

readDHT22_finished:
        clt
        ldi TIM0counter, TIM0delay

readDHT22_POP:
    pop r21
        pop r20
        pop r19
        pop r18
        pop r17
        pop r16 // возвращаем регистрам то что было в них до прерывания
        pop razr0
        pop razr1
```

```
        pop razr2
reti

/*
*       БЛОК КОДА ДЛЯ
*       реализации работы
*       Семисегментника
*/

init_MAX7219:
        ldi tmp, (1<<SPE)|(0<<DORD)|(1<<MSTR)|(0<<CPOL)|(0<<CPHA)|(0<<SPR1)|(0<<SPR0)
        out SPCR, tmp
        ldi tmp, (0<<SPI2X) // SPI mode 0
        out SPSR, tmp
        ldi tmp, (1<<SCK)|(1<<MOSI)|(1<<LOAD)
        out DDRB, tmp

init_MAX7219_values:
.db     0x0C, 0x01      // shutdown
.db     0x0A, 0x0A      // intensity
.db     0x0B, 0x03      // Scan Limit
.db     0x09, 0x0F      // Decode Mode
.db     0x04, 0x0A      // -
.db     0x03, 0x0A      // -
.db     0x02, 0x0A      // -
.db     0x01, 0x0A      // -

ldi ZH, High(init_MAX7219_values*2)
ldi ZL, Low(init_MAX7219_values*2)
ldi tmp, 8      // счетчик итераций

initloop:
        cbi PORTB, LOAD // нулем на линии говорим что ща бум передавать
        lpm toWriteSPI, Z
         call SPI_Send_byte
         adiw Z, 1
        lpm toWriteSPI, Z
         call SPI_Send_byte
         adiw Z, 1
        sbi PORTB, LOAD

        subi tmp,1
        cpi tmp, 0
brne initloop // пока не передадим все 16 байт

ret

SPI_Send_byte:
        out SPDR,toWriteSPI   ;запись данных в регистр данных
        WaitTransmit:
        sbis SPSR,SPIF ;ждем окончания передачи байта
        rjmp WaitTransmit
ret
```

```
 SplitNumber:
/*
        Разбивает число по разрядам, приходящее через регистры r16-H, r17-L
        r18 - счетчик

        результат скидывает в ОЗУ:
        SegmentsData+0 Знак
        SegmentsData+1 3p
        SegmentsData+2 2p
        SegmentsData+3 1p
*/

        push r16
        push r17
        push r18
        push tmp

        clr r18
        ldi tmp, 0x0F  //выпиливаем с индикатора минус если был
        sts SegmentsData+0, tmp
        sbrs r16, 7           // число отрицательное если 7 бит установлен
                rjmp SplitNumber_div100
        ldi tmp, 0x0A  // 0x0A есть знак минус в BCD Code B
        sts SegmentsData+0, tmp

        com r16                                 //
        com r17                                 //
        ldi tmp, 1                              //
        add r17, tmp                    //
        ldi tmp, 0                              //
        adc r16, tmp                    // из доп. кода в абсолютное значение

SplitNumber_div100:
        subi r17, 100                           //
        sbci r16, 0                                     //
        brcs SplitNumber_end_div100  //
        inc r18                                         //
        rjmp SplitNumber_div100             // считаем кол-во сотен

SplitNumber_end_div100:
        ldi tmp, 100 //
        add r17, tmp // делаем нормальный остаток, а то там отрицательное число
        sts SegmentsData+1, r18
        tst r18
                brne end_div100_notnull // если соотен нет, то на их место впихуем минус
                lds tmp, SegmentsData+0
                sts SegmentsData+1, tmp      // 4 разряд -> 3 разряд
                ldi tmp, 0x0F
                sts SegmentsData+0, tmp // выключаем 4 разряд
        end_div100_notnull:
        clr r18

SplitNumber_div10:
        subi r17, 10
        brcs SplitNumber_end_div10
        inc r18
        rjmp SplitNumber_div10
```

```
SplitNumber_end_div10:
        sbr r18, (1<<7) // прикручиваем точку, единица в 7 бит
        sts SegmentsData+2, r18
        ldi tmp, 10
        add r17, tmp
        sts SegmentsData+3, r17

        pop tmp
        pop r18
        pop r17
        pop r16
ret

viewNumber:
        /*
        Выводим число на семисегментник
        Пример: 153 будет выведено как 15.3
        ldi r16, 0x00  H
        ldi r17, 0x00  L
        call viewNumber
        */
        call SplitNumber
        MAX7219_Write2 0x04, SegmentsData
        MAX7219_Write2 0x03, SegmentsData+1
        MAX7219_Write2 0x02, SegmentsData+2
        MAX7219_Write2 0x01, SegmentsData+3
ret

Delay:
//N=TF/5
subi razr0,1
sbci razr1,0
sbci razr2,0
brcc Delay
reti

delay100ms:
        ldi razr2, 0x02
        ldi razr1, 0x71
        ldi razr0, 0x00
        call Delay // 0.2 секунды задержечка
ret

; EEPROM =========================================
.ESEG          ; Сегмент EEPROM
// Не использовался, хотя, можно было бы организовать сохранение T заданной для поддержания
```