Программирование контроллера

В этом разделе мы будем считать, что читатель уже знаком с тем, что из себя представляет микроконтроллер ATMega32 и основами программирования МК этой серии на языке Си. Если это не так — рекомендуем перед началом работы прочитать документ «Что такое МК и основы Си для МК», доступный в печатном виде или на сайте проекта OpenRobotics в разделе «Общая документация».

Все общие вопросы создания проекта, его компиляции в прошивку и загрузки полученного .hex-файла в контроллер будут рассмотрены в следующем разделе, поэтому рекомендуется отнестись к нему особенно внимательно.

Готовим рабочее место

При программировании контроллера нам потребуется среда разработки прошивок и программатор для заливки созданных прошивок в микроконтроллер. Крайне рекомендуется для целей вывода отладочной информации иметь UART-соединение с ПК (USB<=>RoboBus конвертер или Bluetooth-соединение с ПК).

Устанавливаем и проверяем на простой программе компилятор и среду разработки

В качестве компилятора мы будем использовать gcc из пакета WinAVR, а в качестве среды разработки AVR Studio от компании ATMEL. Оба продукта являются бесплатными в том числе для коммерческого использования. Устанавливать пакеты нужно в том же порядке, в каком ниже указано, где их скачивать:

Скачать последнюю версию AVR Studio можно по ссылке: http://www.atmel.ru/Software/Software.htm. Скачать последнюю версию WinAVR можно по ссылке: http://winavr.sourceforge.net/.

После установки пакетов необходимо протестировать работоспособность среды разработки, для этого надо скомпилировать в среде разработки пустой тестовый проект:

1. Запускаем AVR Studio;
2. Выбираем в меню "Projects" пункт "New project";
3. Выбираем в появившемся окне тип проекта "gcc";
4. Вводим название проекта и путь к нему; (ВНИМАНИЕ - в пути и в названии не должно быть русских букв!);
5. Создаём проект, в окно кода вводим приведенный ниже код "main.c";
6. Настраиваем параметры проекта (пункт меню «Project», команда «Configuration options», где выбираем тип МК — ATMega32, выставляем частоту 7372800Hz);
7. Собираем проект (пункт меню "Build", команда "Build", или просто клавиша «F7»);
8. Если внизу в окне сообщений появилось "Build succeeded with 0 Warnings...", то поздравляем, всё успешно скомпилировалось, вы установили и проверили среду разработки;
9. Иначе надо искать что сделано не так, помощь в этом можно получить на страницах робофорума по адресу http://www.roboforum.ru/viewtopic.php?f=69&t=5086;

Файл «main.c»:

```
int main(void)
{
    return 0;
}
```

Далее полученный файл прошивки с расширением .hex (он будет лежать в подпапке default папки созданного проекта) можно залить в контроллер следуя инструкции описанной выше в разделе «Загрузка .hex-прошивок с помощью программатора AVR910».

Базовые функции ввода-вывода:

Мигаем светодиодом

В этом проекте мы научимся мигать встроенным в контроллер светодиодом «D6 led».

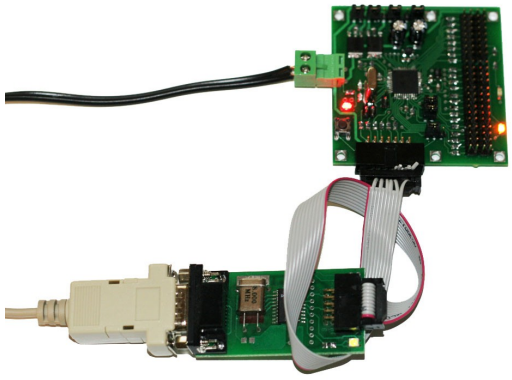
Требуемое аппаратное обеспечение:

1. Контроллер OR-AVR-M32-N;
2. Программатор и кабель ROBOBUS к нему;
3. Источник питания 5-16 В.

Файл «d6_led_flash.c»:

```
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRD = 0x40; //6-й пин порта D настроим как выход
    while(1) //Бесконечный цикл
    {
        PORTD = PORTD & 0xBF; //Сбросим 6-й пин порта D в 0
        _delay_ms(250);       //Ждем 0.25 сек
        PORTD = PORTD | 0x40; //Установим 6-й пин порта D в 1
        _delay_ms(250);       //Ждем 0.25 сек
    }
    return 0;
}
```

Внешний вид получившейся сборки
при использовании программатора AVR910



Зажигаем светодиод при нажатии кнопки

В этом проекте мы научимся включать встроенный в контроллер светодиод «D6 led» при нажатии кнопки подключенной к GPIO-порту D4.

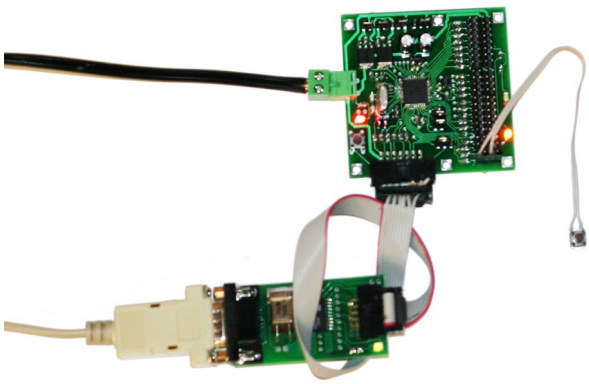
Требуемое аппаратное обеспечение:

1. Контроллер OR-AVR-M32-N;
2. Программатор и кабель ROBOBUS к нему;
3. Источник питания 5-16 В.
4. Кнопка с GPIO-интерфейсом.

Файл «d6_led_button.c»:

```
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRD  = 0x40; //6-й пин порта D настроим как выход
    PORTD = 0x10; //4-й пин порта D подтянем к +3.3 В
    while(1) //Бесконечный цикл
    {
        if((PIND & 0x10) == 0x00){ //Если нажата кнопка (на D4 напряжение 0 В)
          PORTD = PORTD & 0xBF;  //Сбросим 6-й пин порта D в 0
        }else{                   //Иначе не нажата (на D4 напряжение +3.3 В)
          PORTD = PORTD | 0x40;  //Установим 6-й пин порта D в 1
        };
    }


    return 0;
}
```

Внешний вид получившейся сборки
при использовании программатора AVR910



Читаем с помощью АЦП уровень напряжения

В этом проекте мы научимся включать встроенные в контроллер светодиоды «D6 led» и «D7 led» в зависимости от уровня напряжения подведённого к GPIO-порту A0, на котором есть функция АЦП.

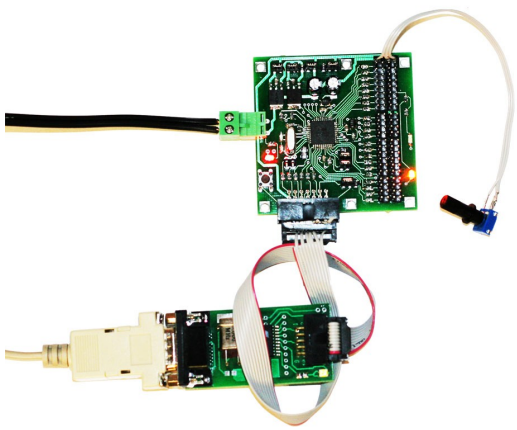
Требуемое аппаратное обеспечение:

1. Контроллер OR-AVR-M32-N;
2. Программатор и кабель ROBOBUS к нему;
3. Источник питания 5-16 В.
4. Переменный резистор с GPIO-интерфейсом.

Файл «led_adc.c»:

```
#include <avr/io.h>
#include <util/delay.h>

//Установить/сбросить бит – удобные в работе сокращения
#define sbi(port, bit) (port) |= (1 << (bit))
#define cbi(port, bit) (port) &= ~(1 << (bit))

int main(void)
{
    DDRD  = 0xC0; //6-й и 7-й пины порта D настроим как выход

    //Иниациализируем АЦП
    sbi(ADCSRA, ADEN);                      // Включим АЦП (подадим питание)
    cbi(ADCSRA, ADATE);                     // Режим работы - оцифровка отдельных значений
    ADCSRA = ((ADCSRA & ~0x07) | 0x06);     // Установим множитель частот clk/64
    ADMUX = ((ADMUX & ~0xC0) | (0x01<<6));  // Установим базис для измерений VCC
    sbi(ADMUX, ADLAR);                      // Включим выравнивание результата по левому краю

    while(1) //Бесконечный цикл
    {
        //Прочитаем значение из АЦП
        int8_t adc_id=0;                        //Будем читать АЦП с порта A0
        ADMUX=(ADMUX & ~0x1F) | (adc_id & 0x1F); //выбираем порт
        sbi(ADCSRA,ADIF);                       //сбросим флаг «конвертация завершена»
        sbi(ADCSRA, ADSC);                      //начинаем конвертацию
        while(!bit_is_set(ADCSRA, ADSC)){};     // ждём флаг окончания конвертации
        int8_t adc_value=ADCH;                  // читаем чего получилось

        //В зависимости от значения зажгём 0..2 светодиода
        if(adc_value<60){         //Если на входе меньше 60
          PORTD = PORTD & 0x3F;  //Сбросим 6-й и 7-й пин порта D в 0
        }else if(adc_value<120){ //Если на выходе 60..119
          PORTD = PORTD & 0x7F;  //Сбросим 7-й пин порта D в 0
          PORTD = PORTD | 0x40;  //Установим 6-й пин порта D в 1
        }else{                    //Если на выходе 120 и больше
          PORTD = PORTD | 0xC0;  //Установим 6-й и 7-й пин порта D в 1
        };
    };
    return 0;
}
```

Внешний вид получившейся сборки
при использовании программатора AVR910



Использование таймеров, ШИМ и прерываний:

Мигаем светодиодом с использованием таймера

В этом проекте мы научимся мигать светодиодом «D6-led» с использованием встроенного в микроконтроллер таймера №0.

Требуемое аппаратное обеспечение:

1. Контроллер OR-AVR-M32-N;
2. Программатор и кабель ROBOBUS к нему;
3. Источник питания 5-16 В.

Файл «led_timer.c»:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

//Установить/сбросить бит – удобные в работе сокращения
#define sbi(port, bit) (port) |= (1 << (bit))
#define cbi(port, bit) (port) &= ~(1 << (bit))

static int8_t state=0; //Заведём свой счетчик

//При переполнении счетчика таймера
SIGNAL(SIG_OVERFLOW0){
  TCNT0=55;          //Переполнение счетчика таймера каждые 255-55=200 тиков - ~35 раз/сек
  state++;           //Увеличим свой счетчик на 1
  if(state<5){       //Если наш счетчик меньше 5
    sbi(PORTD,6);    //тогда включим светодиод D6
  }else{             //иначе
    cbi(PORTD,6);    //выключим светодиод D6
  };
  if(state>10){ state=0; }; //Если наш счетчик >10 тогда обнулим его
};

int main(void)
{
    DDRD  = 0xC0;      //6-й и 7-й пины порта D настроим как выход

    TCCR0=0x05;        //Включим только таймер, при этом с множителем 1024 (1 тик таймера каждые 1024 такта МК)
    TCNT0=55;          //Переполнение счетчика каждые 255-55=200 тиков - ~35 раз/сек

    sbi(TIMSK,TOIE0); //Разрешим прерывание по переполнению счетчика таймера №0
    sei();             //Разрешаем прерывания вообще

    while(1){};        //Бесконечный пустой цикл (работать будут только прерывания)

    return 0;
}
```

Внешний вид сборки ничем не будет отличаться от первой.

Плавно мигаем светодиодом с использованием ШИМ

В этом проекте мы научимся плавно мигать светодиодом «D7-led» с использованием встроенного в микроконтроллер таймера №2, ШИМ-выход которого «OC2» как раз подключен к 7-й линии порта «D».

Требуемое аппаратное обеспечение:

1. Контроллер OR-AVR-M32-N;
2. Программатор и кабель ROBOBUS к нему;
3. Источник питания 5-16 В.

Файл «led_pwm.c»:

```
#include <avr/io.h>
#include <util/delay.h>

//Установить/сбросить бит – удобные в работе сокращения
#define sbi(port, bit) (port) |= (1 << (bit))
#define cbi(port, bit) (port) &= ~(1 << (bit))

int main(void)
{
    DDRD  = 0xC0;     //6-й и 7-й пины порта D настроим как выход

    TCCR2=0x69;       //Включим таймер и ШИМ с множителем 1 (1 тик таймера каждый такт МК)
                      //Режим ШИМ – нормальный, при достижении заданного значения сбрасываем в 0 порт

    TCNT2=0;          //Начинаем таймер с нуля

    //бесконечно
    while(1){

      //плавно увеличим яркость
      for(uint8_t x=0; x<200; x++){
        OCR2=x;             //установим скважность импульса
        _delay_ms(10);       //Ждем 0.01 сек
      };

      //плавно уменьшим яркость
      for(uint8_t x=200; x>0; x--){
        OCR2=x;             //установим скважность импульса
        _delay_ms(10);       //Ждем 0.01 сек
      };
    };

    return 0;
}
```

Внешний вид сборки ничем не будет отличаться от первой.

Обрабатываем внешние сигналы с помощью прерываний

В этом проекте мы научимся включать встроенный в контроллер светодиод «D6 led» при нажатии кнопки подключенной к GPIO-порту B2. Только на этот раз мы не будем в цикле опрашивать кнопку, а будем делать это только если входящий уровень на этом порту поменялся.

Требуемое аппаратное обеспечение:

1. Контроллер OR-AVR-M32-N;
2. Программатор и кабель ROBOBUS к нему;
3. Источник питания 5-16 В.
4. Кнопка с GPIO-интерфейсом.

Внимание! Из-за дребезга контактов вполне может быть, что светодиод будет переключаться не при каждом нажатии кнопки. Для избежания этого эффекта имеет смысл параллельно кнопке устанавливать конденсатор.

Файл «b2_led_int.c»:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

//Установить/сбросить бит – удобные в работе сокращения
#define sbi(port, bit) (port) |= (1 << (bit))
#define cbi(port, bit) (port) &= ~(1 << (bit))

static int8_t state=0; //Заведём свой статус

//При переполнении счетчика таймера
SIGNAL(SIG_INTERRUPT2){
  state ^= 1;        //Изменим статус
  if(state==0){       //Если наш статус=0
    sbi(PORTD,6);    //тогда включим светодиод D6
  }else{             //иначе
    cbi(PORTD,6);    //выключим светодиод D6
  };
};

int main(void)
{
    DDRD  = 0x40; //6-й пин порта D настроим как выход
    PORTB = 0x04; //2-й пин порта B подтянем к +3.3 В

    sbi(MCUCSR,ISC2); //Прерывание по переднему фронту (при увеличении напряжения на входе)

    sbi(GICR,INT2);   //Разрешим прерывание по переполнению счетчика таймера №0
    sei();            //Разрешаем прерывания вообще

    while(1){};       //Бесконечный пустой цикл (работать будут только прерывания)
    return 0;
}
```

Внешний вид сборки ничем не будет отличаться от сборки с кнопкой, только подключена она будет к порту «B2».

Использование протоколов взаимодействия с другими модулями:

Выводим информацию на ПК через UART;

В этом проекте мы научимся передавать на ПК через UART информацию о состоянии кнопки, подключенной к порту .

Требуемое аппаратное обеспечение:

1. Контроллер OR-AVR-M32-N;
2. Программатор и кабель ROBOBUS к нему;
3. Модуль для связи контроллера и ПК по UART'у (например, Bluetooth-адаптер);
4. Источник питания 5-16 В;
5. Кнопка с GPIO-интерфейсом.

Вам потребуется добавить в проект 2 уже готовых файла «uart.c» и «uart.h»:

1. Скачайте их на сайте проекта OpenRobotics в разделе «Общие файлы» (uart.zip);
2. Поместите их в папку своего проекта в файловой системе;
3. Добавьте их в Source Files и в Header Files разделы проекта соответственно;

Кроме того вам потребуется создать свой собственный файл заголовков «defines.h» и указать в нём требуемую скорость UART'а (в зависимости от используемого модуля для связи с ПК).

Файл «defines.h»:

```
#define UART_BAUD  115200
```

Файл «d4_uart.c»:

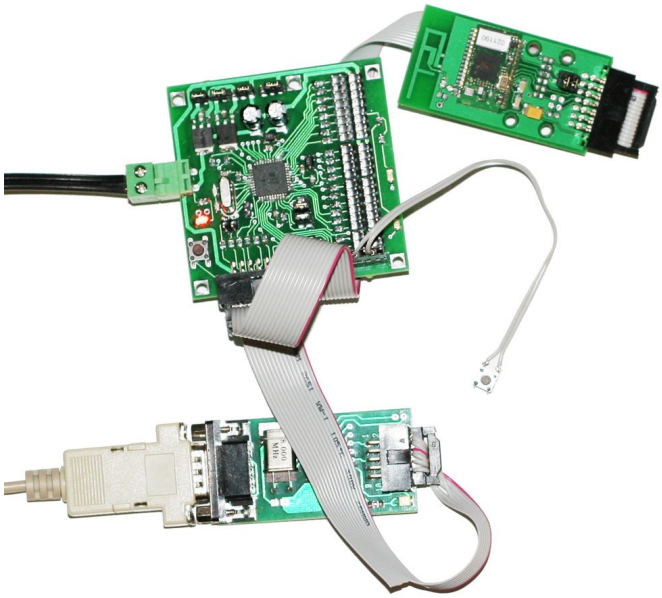
```
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>

#include <avr/io.h>
#include <util/delay.h>
#include "uart.h"

//создаём поток UART используя функции получения символа и отправки символа из библиотеки "uart.h"
FILE uart_str = FDEV_SETUP_STREAM(uart_putchar, uart_getchar, _FDEV_SETUP_RW);

int main(void)
{
    uart_init(); //Инициализируем UART
    stdout = stdin = &uart_str; //Назначим потоки стандартного ввода\вывода на UART
    uint8_t btn=1; //Заведём переменную состояния кнопки

    PORTD = 0x10; //4-й пин порта D подтянем к +3.3 В
    while(1) //Бесконечный цикл
    {
        if((PIND & 0x10) == 0x00){ //Если нажата кнопка (на D4 напряжение 0 В)
          if( btn != 1){ //Если состояние стало «нажата»
            printf("Button pressed!"); //сообщить
            btn=1; //запомнить состояние
          };
        }else{ //Иначе не нажата (на D4 напряжение +3.3 В)
          if( btn != 0){ //Если состояние стало «не нажата»
            printf("Button released!"); //сообщить
            btn=0; //запомнить состояние
          };
        };
        _delay_ms(50);       //Ждем 0.05 сек
    };
    return 0;
}
```

На этом создание прошивки завершено. Но так как мы будем работать с ПК, это еще не всё.

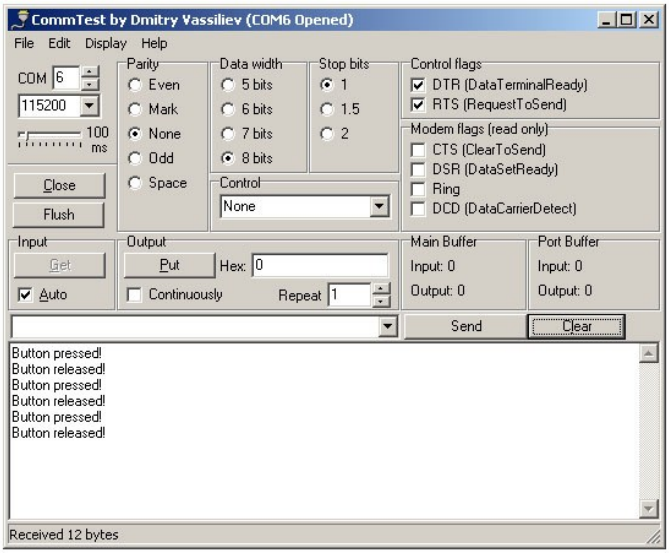
Для проверки работы полученной программы нам потребуется средство работы с COM-портом на ПК (например, CommTest.exe, который можно скачать на сайте OpenRobotics в разделе «Общие файлы»).

Если вы используете Bluetooth-адаптер — возможно вам нужно будет скачать и установить какую-либо программу-драйвер Bluetooth-порта, которая создаст для вашего устройства виртуальный COM-порт на ПК. (Например, можно скачать пробную версию BlueSoleil).

Внешний вид получившейся сборки
при использовании программатора AVR910
и модуля Bluetooth-адаптера OR-BT20-115.2



Вид окна терминальной программы
(выбран соответствующий COM-порт и правильно указана скорость)



Работаем с сонаром SRFxx через I2C

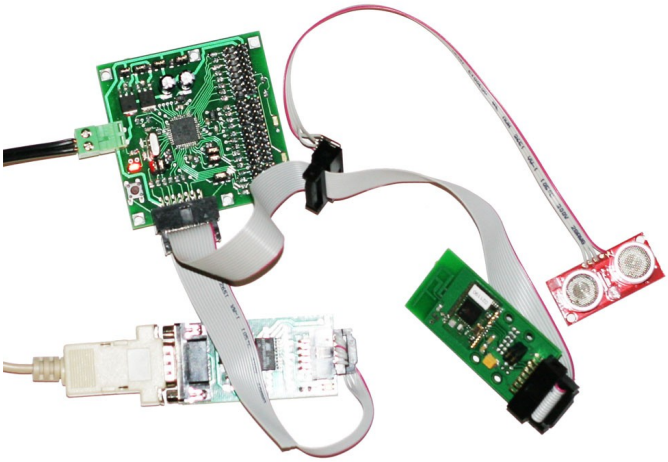
В этом проекте мы научимся работать с сонаром по i2c и передавать расстояние на ПК.

Требуемое аппаратное обеспечение:

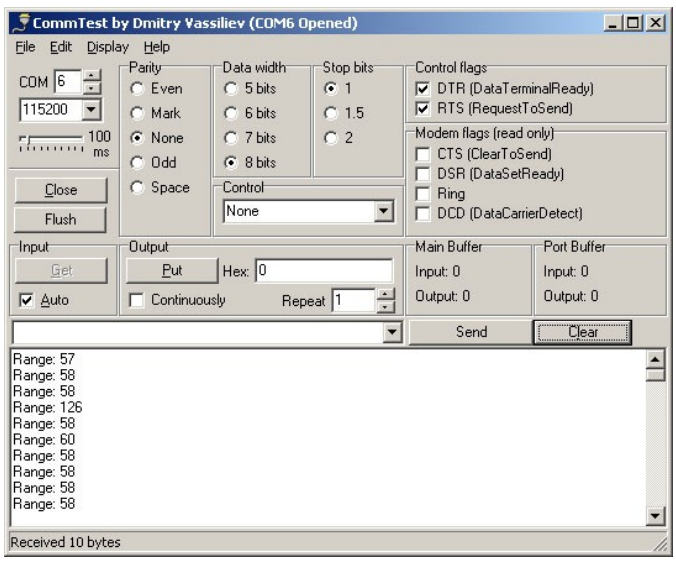
1. Контроллер OR-AVR-M32-N;
2. Программатор и кабель ROBOBUS к нему;
3. Модуль для связи контроллера и ПК по UART'у (например, Bluetooth-адаптер);
4. Сонар Devantech SRF08 (SRF02/SRF10/SRF235) с RoboBus-интерфейсом.
5. Источник питания 5-16 В;

Как и в предыдущем проекте вам потребуется добавить в проект 2 уже готовых файла «uart.c» и «uart.h» и создать свой собственный файл заголовков «defines.h».

Внешний вид получившейся сборки
при использовании программатора AVR910,
модуля Bluetooth-адаптера OR-BT20-115.2 и
сонара Devantech SRF08 с разъемом RoboBus



Вид окна терминальной программы
(выбран соответствующий COM-порт и правильно указана скорость)



Файл «sonar_uart.c»:

```
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>

#include <avr/io.h>
#include <util/delay.h>
#include "uart.h"

char i2c_read(char address, char reg)
{
   char read_data = 0;

   TWCR = 0xA4;                                 // send a start bit on i2c bus
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   TWDR = address;                              // load address of i2c device
   TWCR = 0x84;                                 // transmit
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   TWDR = reg;                                  // send register number to read from
   TWCR = 0x84;                                 // transmit
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit

   TWCR = 0xA4;                                 // send repeated start bit
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   TWDR = address+1;                            // transmit address of i2c device with readbit set
   TWCR = 0xC4;                                 // clear transmit interupt flag
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   TWCR = 0x84;                                 // transmit, nack (last byte request)
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   read_data = TWDR;                            // and grab the target data
   TWCR = 0x94;                                 // send a stop bit on i2c bus
   return read_data;

}

void i2c_transmit(char address, char reg, char data)
{
   TWCR = 0xA4;                                 // send a start bit on i2c bus
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   TWDR = address;                              // load address of i2c device
   TWCR = 0x84;                                 // transmit
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   TWDR = reg;
   TWCR = 0x84;                                 // transmit
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   TWDR = data;
   TWCR = 0x84;                                 // transmit
   while(!(TWCR & 0x80));                       // wait for confirmation of transmit
   TWCR = 0x94;                                 // stop bit
}

void i2c_init(void)
{
   TWBR = 32;                                   // Установим скорость шины i2c 100КГц
}

//создаём поток UART используя функции получения символа и отправки символа из библиотеки "uart.h"
FILE uart_str = FDEV_SETUP_STREAM(uart_putchar, uart_getchar, _FDEV_SETUP_RW);

int main(void)
{
    i2c_init();                 //Инициализируем I2C
    uart_init();                //Инициализируем UART
    stdout = stdin = &uart_str; //Назначим потоки стандартного ввода\вывода на UART

    while(1) //Бесконечный цикл
    {
        unsigned int range;

        i2c_transmit(0xE0,0,0x51);     // Даём команду на запуск сонара, указав результат вернуть в сантиметрах
        _delay_ms(70);                 // Ждем 70мс, пока сонар проведёт измерение

        range = i2c_read(0xE0,2) <<8;  // Читаем старший бит расстояния
        range += i2c_read(0xE0,3);     // Читаем младший бит расстояния

        printf("Range: %d\n",range);   //Покажем результат по уарту

        _delay_ms(930);                //Ждем 0.930 сек (мерять будем раз в секунду)
    };
    return 0;
}
```

Для работы с UART'ом нам потребуются те же инструменты, что и в предыдущей задаче.