

**Федеральное государственное бюджетное образовательное
учреждение высшего образования "Белгородский государственный
технологический университет им. В.Г. Шухова"**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем.

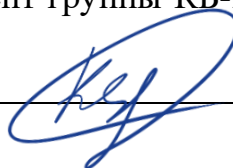
Лабораторная работа работа № 4

Изучение принципов управления ЖК дисплеем типа 1602 с использованием
последовательного интерфейса I2C.

Вариант 13

Выполнил:

Студент группы КБ-211



Коренев Д.Н.

Принял:

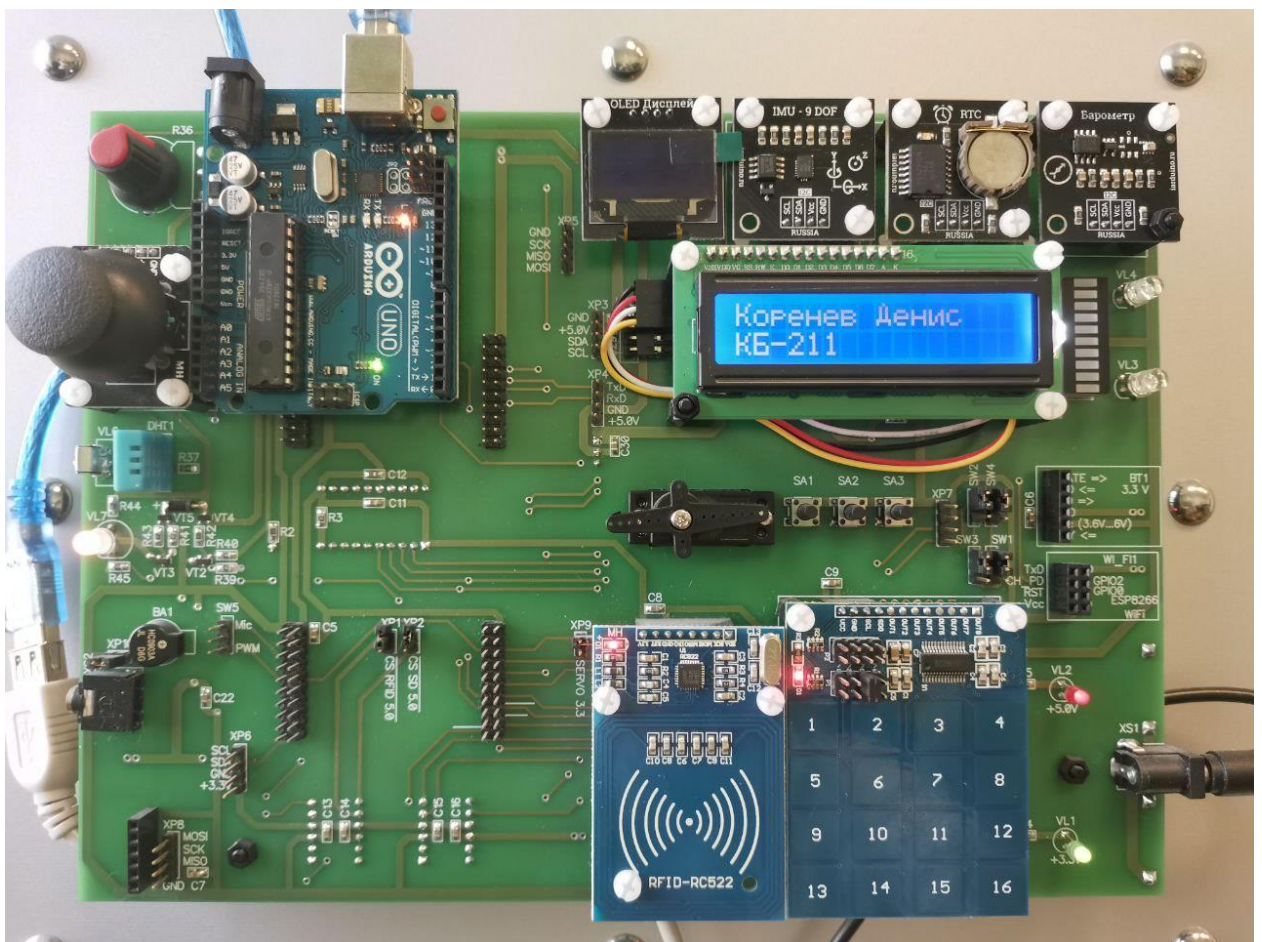
Шамраев А.А.

Цель работы: изучить принципы функционирования и возможности программного управления ЖК дисплеем типа 1602 на базе контроллера HD44780 и его аналогах, разработать алгоритм и программу для вывода информации на экран.

Задание

Разработать в среде программирования Arduino IDE программу для микроконтроллера ATmega 328P:

1. На “3” вывести имя, фамилию и группу студента на ЖКИ. Группу вывести русскими буквами, для создания русских символов использовать функцию createChar.



Код программы:

```
1 #include <Wire.h>
2 #include <LiquidCrystal_I2C.h>
3 LiquidCrystal_I2C lcd(0x27, 16, 2);
4
5 byte cChars[][8] = {
6     {0b00010001,
7     0b00010010,
8     0b00010010,
9     0b00011100,
10    0b00010010,
11    0b00010010,
```

```

12     0b00010001,
13     0b00000000}, // К
14     {0b00000000,
15     0b00000000,
16     0b00011110,
17     0b00010001,
18     0b00011110,
19     0b00010001,
20     0b00011110,
21     0b00000000}, // В
22     {0b00000000,
23     0b00000000,
24     0b00010001,
25     0b00010001,
26     0b00011111,
27     0b00010001,
28     0b00010001,
29     0b00000000}, // Н
30     {0b00011111,
31     0b00010000,
32     0b00010000,
33     0b00011110,
34     0b00010001,
35     0b00010001,
36     0b00011111,
37     0b00000000}, // Б
38     {0b00000100,
39     0b00001010,
40     0b00001010,
41     0b00001010,
42     0b00001010,
43     0b00011111,
44     0b00010001,
45     0b00000000}, // Д
46     {0b00000000,
47     0b00000000,
48     0b00010001,
49     0b00010011,
50     0b00010101,
51     0b00010101,
52     0b00011001,
53     0b00000000} // И
54 };
55
56
57 // Корнев
58 // КБ-211
59 void setup()
60 {
61
62     lcd.init();
63
64     lcd.createChar(0, cChars[0]); // К
65     lcd.createChar(1, cChars[1]); // В
66     lcd.createChar(2, cChars[2]); // Н
67     lcd.createChar(3, cChars[3]); // Б
68     lcd.createChar(4, cChars[4]); // Д
69     lcd.createChar(5, cChars[5]); // И
70
71     lcd.backlight();
72
73     lcd.setCursor(0,0);
74     lcd.write((uint8_t)0);
75     lcd.print("ope");
76     lcd.write((uint8_t)2);
77     lcd.print("e");
78     lcd.write((uint8_t)1);
79     lcd.print(" ");
80     lcd.write((uint8_t)4);

```

```

81  lcd.print("e");
82  lcd.write((uint8_t)2);
83  lcd.write((uint8_t)5);
84  lcd.print("c");
85
86  lcd.setCursor(0,1);
87  lcd.write((uint8_t)0);
88  lcd.write((uint8_t)3);
89  lcd.print("-211");
90
91  }
92
93  void loop()
94  {
95  }

```

2. На “4” записать новый символ в память знакогенератора при помощи команд (функция `command`). Вывести какую-либо строку, используя свой символ, в соответствии с вариантом задания (номер в журнале $\% 5 + 1$):

1) Установив флаг `SH=1`, установить положение указателя в конец строки и выводить символы один за другим с интервалом 1 сек. Новые символы должны появляться на месте курсора, а уже имеющийся текст сдвигаться влево.

2) Выводить строку справа налево посимвольно с интервалом 1 сек.

3) С помощью команды сдвига экрана создать эффект бегущей строки влево.

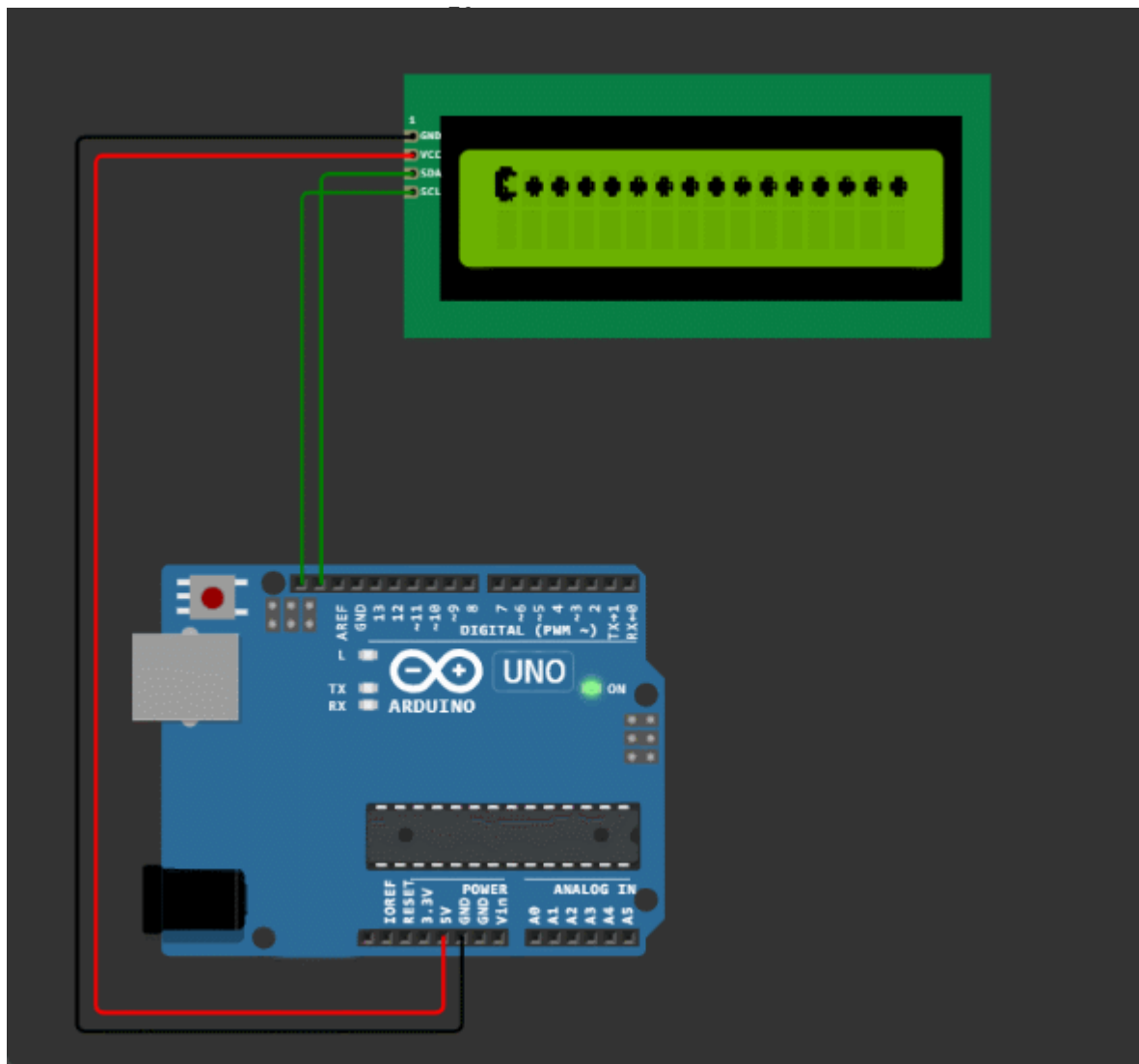
4) С помощью команды сдвига экрана создать эффект бегущей строки вправо.

5) Вывести строку посимвольно, а затем сменить направление сдвига указателя и перезаписать её другой строкой.

3. На “5” взаимодействовать с дисплеем в обход библиотеки. Выполнить вариант задания из 2 пункта. Создать анимированный символ, создав несколько символов-кадров.

Ссылка на проект Wokwi:

<https://wokwi.com/projects/379373289957283841>



```

1  #ifndef LC_I2C_H__
2  #define LC_I2C_H__
3
4  #include <Wire.h>
5
6  // Основные команды
7  #define LCD_CLEARDISPLAY 0x01 // Очистить дисплей и установить
8                                // курсор в начальное положение
9  #define LCD_RETURNHOME 0x02  // Установить курсор в начальное
10                               // положение
11 #define LCD_ENTRYMODESET 0x04 // Установить режим ввода данных
12 #define LCD_DISPLAYCONTROL 0x08 // Режимы дисплея
13 #define LCD_CURSORSHIFT 0x10 // Сдвиг курсора
14 #define LCD_FUNCTIONSET 0x20 // Установка размера символа,
15                               // кол-во строк дисплея,
16                               // интерфейс данных
17 #define LCD_SETCGRAMADDR 0x40 // Установка адреса CGRAM
18 #define LCD_SETDDRAMADDR 0x80 // Установка адреса DDRAM
19
20 // Флаги для всякого разного
21 #define LCD_ENTRYRIGHT 0x00 // Напр. движ. курсора <--
22 #define LCD_ENTRYLEFT 0x02 // Напр. движ. курсора -->
23 #define LCD_ENTRYSHIFTINCREMENT 0x01 // Прокрутка текста <--
24                                     // (справа-налево)
25 #define LCD_ENTRYSHIFTDECREMENT 0x00 // Отключить прокрутку
26

```

```

27 // Флаги вкл/выкл
28 #define LCD_DISPLAYON 0x04 // Дисплей ON
29 #define LCD_DISPLAYOFF 0x00 // Дисплей OFF
30 #define LCD_CURSORON 0x02 // Курсор ON
31 #define LCD_CURSOROFF 0x00 // Курсор OFF
32 #define LCD_BLINKON 0x01 // Мигание ON
33 #define LCD_BLINKOFF 0x00 // Мигание OFF
34
35 // Флаги сдвигов
36 #define LCD_DISPLAYMOVE 0x08 // Движение всего дисплея,
37 // вместо курсора
38 #define LCD_CURSORMOVE 0x00 // Движение курсора,
39 // вместо всего дисплея
40 #define LCD_MOVERIGHT 0x04 // Движение вправо
41 #define LCD_MOVELEFT 0x00 // Движение влево
42
43 // Флаги наборов функций
44 #define LCD_8BITMODE 0x10 // 8-битный режим
45 #define LCD_4BITMODE 0x00 // 4-битный режим
46 #define LCD_2LINE 0x08 // 1 строчка
47 #define LCD_1LINE 0x00 // 2 строчки
48 #define LCD_5x10DOTS 0x04 // Режим символов 5x10
49 #define LCD_5x8DOTS 0x00 // Режим символов 5x8
50
51 // Флаги подсветки
52 #define LCD_BACKLIGHT 0x08 // Подсветка ON
53 #define LCD_NOBACKLIGHT 0x00 // Подсветка OFF
54
55 #define En B00000100 // Бит включения
56 #define Rw B00000010 // Бит Read/Write
57 #define Rs B00000001 // Бит выбора регистра
58
59 class LC_I2C {
60 public:
61 // Конструктор
62 //
63 // Parameters
64 // -----
65 // lcd_addr(uint8_t): I2C адрес LCD дисплея
66 // lcd_cols(uint8_t): Кол-во столбцов на дисплее
67 // lcd_rows(uint8_t): Кол-во строк на дисплее
68 // charsize(uint8_t): Размер символов, LCD_5x10DOTS
69 // или LCD_5x8DOTS
70 LC_I2C(uint8_t lcd_addr, uint8_t lcd_cols, uint8_t lcd_rows,
71 uint8_t charsize = LCD_5x8DOTS);
72
73 // Обязательно вызывать после создания объекта дисплея
74 void begin();
75
76 // Очистить дисплей и сбросить положение курсора
77 void clear();
78
79 // Возврат курсора
80 void home();
81
82 // Выключить вывод символов
83 void noDisplay();
84
85 // Показывать символы на дисплее
86 void display();
87
88 // Не моргать курсором
89 void noBlink();
90
91 // Моргать курсором
92 void blink();
93
94 // Выключить курсор
95 void noCursor();

```

```

96
97     // Включить курсор
98     void cursor();
99
100    // Вывод строки
101    //
102    // Parameters
103    // -----
104    //     str(char*): Указатель на начало строки
105    void print(char* str);
106
107    // Сдвиг дисплея влево
108    void scrollDisplayLeft();
109
110    // Сдвиг дисплея вправо
111    void scrollDisplayRight();
112
113    //
114    void printLeft();
115
116    //
117    void printRight();
118
119    // Порядок текста слева-направо
120    void leftToRight();
121
122    // Порядок текста справа-налево
123    void rightToLeft();
124
125    // Инкремент сдвига
126    void shiftIncrement();
127
128    // Декремент сдвига
129    void shiftDecrement();
130
131    // Выключить подсветку
132    void noBacklight();
133
134    // Включить подсветку
135    void backlight();
136
137    // Возвращает состояние подсветки
138    //
139    // Returns
140    // -----
141    //     bool: Состояние подсветки
142    bool getBacklight();
143
144    // Выводить текст по правому краю курсора
145    void autoscroll();
146
147    // Выводить текст по левому краю курсора
148    void noAutoscroll();
149
150    // Позволяет заполнить первые 8 байт CGRAM
151    // собственными символами
152    void createChar(uint8_t, uint8_t[]);
153
154    // Установить положение курсора
155    void setCursor(uint8_t, uint8_t);
156
157    // Написать символ на дисплее
158    virtual size_t write(uint8_t);
159
160    // Отправить команду
161    void command(uint8_t);
162
163    // Включить мигание курсора
164    inline void blink_on() { blink(); }

```



```

165
166 // Выключить мигание курсора
167 inline void blink_off() { noBlink(); }
168
169 // Включить курсор
170 inline void cursor_on() { cursor(); }
171
172 // Выключить курсор
173 inline void cursor_off() { noCursor(); }
174
175 // ----==== Compatibility API function aliases =====
176
177 // Штука для backlight() и nobacklight()
178 void setBacklight(uint8_t new_val);
179
180 // Штука для createChar()
181 void load_custom_character(uint8_t char_num, uint8_t *rows);
182
183 private:
184 // Отправить команду или данные
185 void send(uint8_t, uint8_t);
186 void write4bits(uint8_t);
187 void expanderWrite(uint8_t);
188 void pulseEnable(uint8_t);
189 uint8_t _addr;
190 uint8_t _displayfunction;
191 uint8_t _displaycontrol;
192 uint8_t _displaymode;
193 uint8_t _cols;
194 uint8_t _rows;
195 uint8_t _charsize;
196 uint8_t _backlightval;
197 };
198
199 #endif // LC_I2C_H__
200
201
202
203 // #include "LC_I2C.h"
204
205 LC_I2C::LC_I2C(uint8_t lcd_addr, uint8_t lcd_cols,
206               uint8_t lcd_rows, uint8_t charsize)
207 {
208     _addr = lcd_addr;
209     _cols = lcd_cols;
210     _rows = lcd_rows;
211     _charsize = charsize;
212     _backlightval = LCD_BACKLIGHT;
213 }
214
215 void LC_I2C::begin() {
216     Wire.begin();
217     _displayfunction = LCD_4BITMODE | LCD_1LINE | LCD_5x8DOTS;
218     if (_rows > 1) {
219         _displayfunction |= LCD_2LINE;
220     }
221     if ((_charsize != 0) && (_rows == 1)) {
222         _displayfunction |= LCD_5x10DOTS;
223     }
224     delay(50); // импульс 50мс
225     // Устанавливаем RS и R/W на 0, чтобы начать команды
226     expanderWrite(_backlightval); // выключить подсветку
227     delay(1000);
228     // Переключаем дисплей в 4-битовый режим
229     write4bits(0x03 << 4);
230     delayMicroseconds(4500); // ждем 4.5мс
231     // Вторая команда
232     write4bits(0x03 << 4);
233     delayMicroseconds(4500); // ждем 4.5мс

```



```

234 // Третья команда
235 write4bits(0x03 << 4);
236 delayMicroseconds(150);
237 // Окончательно устанавливаем дисплей в 4-битовый режим
238 write4bits(0x02 << 4);
239 // Устанавливаем кол-во строк, шрифт и т.д.
240 command(LCD_FUNCTIONSET | _displayfunction);
241 // Включаем дисплей без курсора и без мигания по умолчанию
242 _displaycontrol = LCD_DISPLAYON | LCD_CURSOROFF | LCD_BLINKOFF;
243 display();
244 // Очищаем дисплей
245 clear();
246 // Устанавливаем направление текста (для нормальных языков)
247 _displaymode = LCD_ENTRYLEFT | LCD_ENTRYSHIFTDECREMENT;
248 // Ждем дальнейших команд
249 command(LCD_ENTRYMODESET | _displaymode);
250 home();
251 }
252
253 // ----- Высокоуровневые команды :3 -----
254
255 void LC_I2C::clear(){
256     command(LCD_CLEARDISPLAY);
257     delayMicroseconds(2000);
258 }
259
260 void LC_I2C::home(){
261     command(LCD_RETURNHOME);
262     delayMicroseconds(2000);
263 }
264
265 void LC_I2C::setCursor(uint8_t col, uint8_t row){
266     int row_offsets[] = { 0x00, 0x40, 0x14, 0x54 };
267     if (row > _rows) {
268         row = _rows-1; // считаем с 0
269     }
270     command(LCD_SETDDRAMADDR | (col + row_offsets[row]));
271 }
272
273 void LC_I2C::noDisplay() {
274     _displaycontrol &= ~LCD_DISPLAYON;
275     command(LCD_DISPLAYCONTROL | _displaycontrol);
276 }
277
278 void LC_I2C::display() {
279     _displaycontrol |= LCD_DISPLAYON;
280     command(LCD_DISPLAYCONTROL | _displaycontrol);
281 }
282
283 void LC_I2C::noCursor() {
284     _displaycontrol &= ~LCD_CURSORON;
285     command(LCD_DISPLAYCONTROL | _displaycontrol);
286 }
287 void LC_I2C::cursor() {
288     _displaycontrol |= LCD_CURSORON;
289     command(LCD_DISPLAYCONTROL | _displaycontrol);
290 }
291
292 void LC_I2C::noBlink() {
293     _displaycontrol &= ~LCD_BLINKON;
294     command(LCD_DISPLAYCONTROL | _displaycontrol);
295 }
296 void LC_I2C::blink() {
297     _displaycontrol |= LCD_BLINKON;
298     command(LCD_DISPLAYCONTROL | _displaycontrol);
299 }
300
301 void LC_I2C::scrollDisplayLeft(void) {
302     command(LCD_CURSORSHIFT | LCD_DISPLAYMOVE | LCD_MOVELEFT);

```

```

303 }
304 void LC_I2C::scrollDisplayRight(void) {
305     command(LCD_CURSORSHIFT | LCD_DISPLAYMOVE | LCD_MOVERIGHT);
306 }
307
308 void LC_I2C::leftToRight(void) {
309     _displaymode |= LCD_ENTRYLEFT;
310     command(LCD_ENTRYMODESET | _displaymode);
311 }
312
313 void LC_I2C::rightToLeft(void) {
314     _displaymode &= ~LCD_ENTRYLEFT;
315     command(LCD_ENTRYMODESET | _displaymode);
316 }
317
318 void LC_I2C::autoscroll(void) {
319     _displaymode |= LCD_ENTRYSHIFTINCREMENT;
320     command(LCD_ENTRYMODESET | _displaymode);
321 }
322
323 void LC_I2C::noAutoscroll(void) {
324     _displaymode &= ~LCD_ENTRYSHIFTINCREMENT;
325     command(LCD_ENTRYMODESET | _displaymode);
326 }
327
328 void LC_I2C::createChar(uint8_t location,
329     uint8_t charmap[]) {
330     location &= 0x7;
331     command(LCD_SETCGRAMADDR | (location << 3));
332     for (int i=0; i<8; i++) {
333         write(charmap[i]);
334     }
335 }
336
337 void LC_I2C::noBacklight(void) {
338     _backlightval=LCD_NOBACKLIGHT;
339     expanderWrite(0);
340 }
341
342 void LC_I2C::backlight(void) {
343     _backlightval=LCD_BACKLIGHT;
344     expanderWrite(0);
345 }
346
347 bool LC_I2C::getBacklight() {
348     return _backlightval == LCD_BACKLIGHT;
349 }
350
351 void LC_I2C::print(char* str){
352     while(*str){
353         write((int) *str++);
354     }
355 }
356
357 // ----- Среднеуровневые команды -----
358
359 inline void LC_I2C::command(uint8_t value) {
360     send(value, 0);
361 }
362
363 inline size_t LC_I2C::write(uint8_t value) {
364     send(value, Rs);
365     return 1;
366 }
367
368 // ----- Низкоуровневые команды -----
369
370 void LC_I2C::send(uint8_t value, uint8_t mode) {
371     uint8_t highnib=value & 0xf0;

```

```

372     uint8_t lownib=(value << 4) & 0xf0;
373     write4bits((highnib) | mode);
374     write4bits((lownib) | mode);
375 }
376
377 void LC_I2C::write4bits(uint8_t value) {
378     expanderWrite(value);
379     pulseEnable(value);
380 }
381
382 void LC_I2C::expanderWrite(uint8_t _data){
383     Wire.beginTransaction(_addr);
384     Wire.write((int)(_data) | _backlightval);
385     Wire.endTransmission();
386 }
387
388 void LC_I2C::pulseEnable(uint8_t _data){
389     expanderWrite(_data | En); // En high
390     delayMicroseconds(1);      // импульс >450нс
391
392     expanderWrite(_data & ~En); // En low
393     delayMicroseconds(50);      // импульс > 37мкс
394 }
395
396 void LC_I2C::load_custom_character(uint8_t char_num, uint8_t *rows){
397     createChar(char_num, rows);
398 }
399
400 void LC_I2C::setBacklight(uint8_t new_val){
401     if (new_val) {
402         backlight();           // включить подсветку on
403     } else {
404         noBacklight();         // выключить подсветку off
405     }
406 }
407
408
409 uint8_t strlen(char* str){
410     uint8_t size = 0;
411     while(*str){
412         str++;
413         size++;
414     }
415     return size;
416 }
417
418
419 // #include "LC_I2C.h"
420
421 LC_I2C lcd(0x27, 16, 2);
422
423 byte cChars[][8] = {
424     {0b00010001,
425     0b00010010,
426     0b00010010,
427     0b00011100,
428     0b00010010,
429     0b00010010,
430     0b00010001,
431     0b00000000}, // К
432     {0b00000000,
433     0b00000000,
434     0b00011110,
435     0b00010001,
436     0b00011110,
437     0b00010001,
438     0b00011110,
439     0b00000000}, // В
440     {0b00000000,

```

```

441 0b00000000,
442 0b00010001,
443 0b00010001,
444 0b00011111,
445 0b00010001,
446 0b00010001,
447 0b00000000}, // н
448 {0b00011111,
449 0b00010000,
450 0b00010000,
451 0b00011110,
452 0b00010001,
453 0b00010001,
454 0b00011111,
455 0b00000000}, // Б
456 {0b00000100,
457 0b00001010,
458 0b00001010,
459 0b00001010,
460 0b00001010,
461 0b00011111,
462 0b00010001,
463 0b00000000}, // Д
464 {0b00000000,
465 0b00000000,
466 0b00001010,
467 0b00011111,
468 0b00011111,
469 0b00001110,
470 0b00000100,
471 0b00000000}, // heart_big
472 {0b00000000,
473 0b00000000,
474 0b00000000,
475 0b00001010,
476 0b00011111,
477 0b00001110,
478 0b00000100,
479 0b00000000}, // heart_small
480 {0b00000000,
481 0b00000000,
482 0b00010001,
483 0b00010011,
484 0b00010101,
485 0b00010101,
486 0b00011001,
487 0b00000000}, // и
488 {0b00001110,
489 0b00011111,
490 0b00011100,
491 0b00011000,
492 0b00011100,
493 0b00011111,
494 0b00001110,
495 0b00000000}, // pacman_open
496 {0b00001110,
497 0b00011111,
498 0b00011111,
499 0b00011111,
500 0b00011111,
501 0b00011111,
502 0b00001110,
503 0b00000000}, // pacman_closed
504 {0b00000000,
505 0b00000000,
506 0b00000110,
507 0b00001111,
508 0b00001111,
509 0b00000110,

```

```

510 0b00000000,
511 0b00000000}, // pacman_dot
512 };
513
514 void setup(){
515   lcd.begin();
516   lcd.home();
517   lcd.backlight();
518
519   lcd.createChar(0, cChars[5]);
520   lcd.createChar(1, cChars[6]);
521   lcd.createChar(3, cChars[8]);
522   lcd.createChar(4, cChars[9]);
523   lcd.createChar(5, cChars[10]);
524 }
525
526 void loop(){
527
528   int speed_div = 500;
529   lcd.home();
530
531   lcd.write(3);
532   lcd.write(5);
533   lcd.write(5);
534   lcd.write(5);
535   lcd.write(5);
536   lcd.write(5);
537   lcd.write(5);
538   lcd.write(5);
539   lcd.write(5);
540   lcd.write(5);
541   lcd.write(5);
542   lcd.write(5);
543   lcd.write(5);
544   lcd.write(5);
545   lcd.write(5);
546   lcd.write(5);
547   lcd.write(5);
548   // Задержка перед началом сдвига
549   delay(speed_div);
550
551   // Цикл для сдвига текста вправо
552   for (int positionCounter = 0; positionCounter < 16; positionCounter++) {
553     // Сдвиг вправо
554     lcd.scrollDisplayRight();
555     if (positionCounter % 2 == 0){
556       lcd.createChar(3, cChars[9]);
557     } else {
558       lcd.createChar(3, cChars[8]);
559     }
560     // Задержка между каждым сдвигом
561     delay(speed_div);
562   }
563
564   // Очистка дисплея перед следующим циклом
565   lcd.clear();
566 }

```