**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:

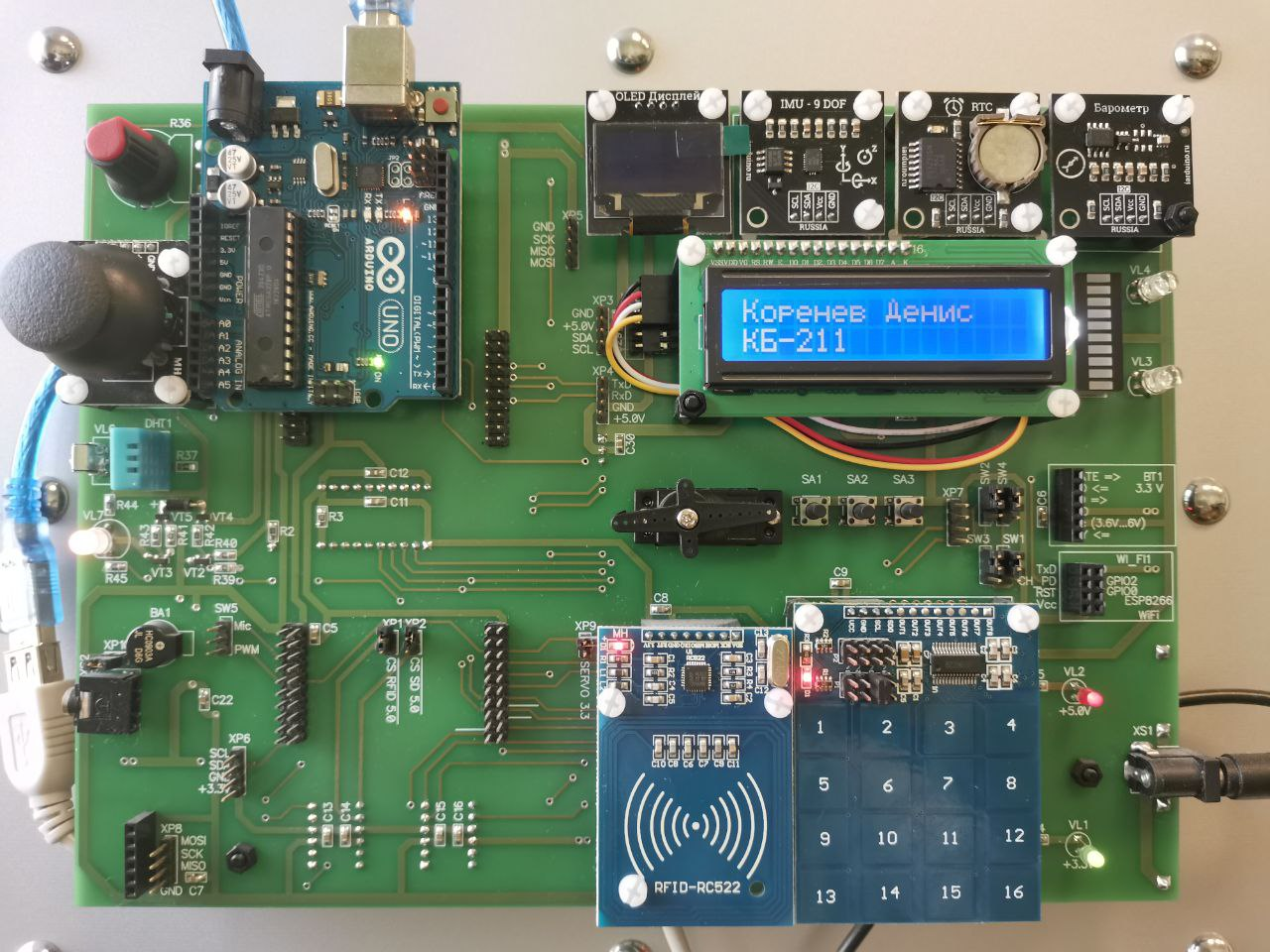
Шамраев А.А.

*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. byte cChars[][8] = {
5. {0b00010001,
6. 0b00010010,
7. 0b00010010,
8. 0b00011100,
9. 0b00010010,
10. 0b00010010,
11. 0b00010001,
12. 0b00000000}, // К
13. {0b00000000,
14. 0b00000000,
15. 0b00011110,
16. 0b00010001,
17. 0b00011110,
18. 0b00010001,
19. 0b00011110,
20. 0b00000000}, // в
21. {0b00000000,
22. 0b00000000,
23. 0b00010001,
24. 0b00010001,
25. 0b00011111,
26. 0b00010001,
27. 0b00010001,
28. 0b00000000}, // н
29. {0b00011111,
30. 0b00010000,
31. 0b00010000,
32. 0b00011110,
33. 0b00010001,
34. 0b00010001,
35. 0b00011111,
36. 0b00000000}, // Б
37. {0b00000100,
38. 0b00001010,
39. 0b00001010,
40. 0b00001010,
41. 0b00001010,
42. 0b00011111,
43. 0b00010001,
44. 0b00000000}, // Д
45. {0b00000000,
46. 0b00000000,
47. 0b00010001,
48. 0b00010011,
49. 0b00010101,
50. 0b00010101,
51. 0b00011001,
52. 0b00000000} // и
53. };
54. // Коренев
55. // КБ-211
56. void setup()
57. {
59. lcd.init();
60. lcd.createChar(0, cChars[0]); // К
61. lcd.createChar(1, cChars[1]); // в
62. lcd.createChar(2, cChars[2]); // н
63. lcd.createChar(3, cChars[3]); // Б
64. lcd.createChar(4, cChars[4]); // Д
65. lcd.createChar(5, cChars[5]); // и
67. lcd.backlight();
69. lcd.setCursor(0,0);
70. lcd.write((uint8\_t)0);
71. lcd.print("ope");
72. lcd.write((uint8\_t)2);
73. lcd.print("e");
74. lcd.write((uint8\_t)1);
75. lcd.print(" ");
76. lcd.write((uint8\_t)4);
77. lcd.print("e");
78. lcd.write((uint8\_t)2);
79. lcd.write((uint8\_t)5);
80. lcd.print("c");
82. lcd.setCursor(0,1);
83. lcd.write((uint8\_t)0);
84. lcd.write((uint8\_t)3);
85. lcd.print("-211");
87. }
88. void loop()
89. {
90. }

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.

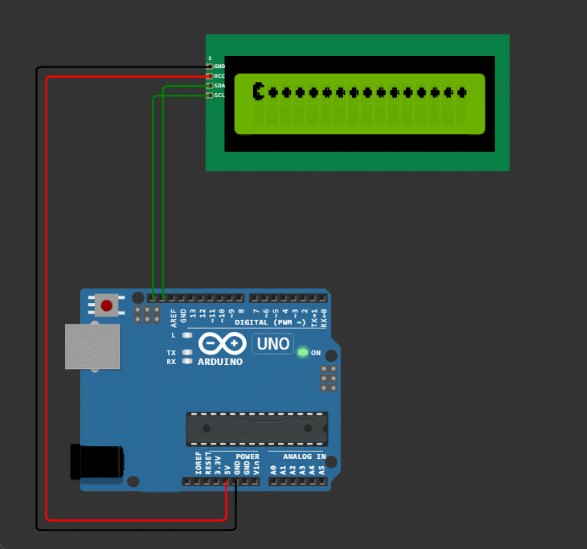
3) С помощью команды сдвига экрана создать эффект бегущей строки влево.

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>

Код программы:

1. #ifndef LC\_I2C\_H\_\_
2. #define LC\_I2C\_H\_\_
3. #include <Wire.h>
4. // Основные команды
5. #define LCD\_CLEARDISPLAY 0x01 // Очистить дисплей и установить
6. // курсор в начальное положение
7. #define LCD\_RETURNHOME 0x02 // Установить курсор в начальное
8. // положение
9. #define LCD\_ENTRYMODESET 0x04 // Установать режим ввода данных
10. #define LCD\_DISPLAYCONTROL 0x08 // Режимы дисплея
11. #define LCD\_CURSORSHIFT 0x10 // Сдвиг курсора
12. #define LCD\_FUNCTIONSET 0x20 // Установка размера символа,
13. // кол-во строк дисплея,
14. // инфтерфейс данных
15. #define LCD\_SETCGRAMADDR 0x40 // Установка адреса CGRAM
16. #define LCD\_SETDDRAMADDR 0x80 // Установка адреса DDRAM
17. // Флаги для всякого разного
18. #define LCD\_ENTRYRIGHT 0x00 // Напр. движ. курсора <--
19. #define LCD\_ENTRYLEFT 0x02 // Напр. движ. курсора -->
20. #define LCD\_ENTRYSHIFTINCREMENT 0x01 // Прокрутка текста <--
21. // (справа-налево)
22. #define LCD\_ENTRYSHIFTDECREMENT 0x00 // Отключить прокрутку
23. // Флаги вкл/выкл
24. #define LCD\_DISPLAYON 0x04 // Дисплей ON
25. #define LCD\_DISPLAYOFF 0x00 // Дисплей OFF
26. #define LCD\_CURSORON 0x02 // Курсор ON
27. #define LCD\_CURSOROFF 0x00 // Курсор OFF
28. #define LCD\_BLINKON 0x01 // Мигание ON
29. #define LCD\_BLINKOFF 0x00 // Мигание OFF
30. // Флаги сдвигов
31. #define LCD\_DISPLAYMOVE 0x08 // Движение всего дисплея,
32. // вместо курсора
33. #define LCD\_CURSORMOVE 0x00 // Движение курсора,
34. // вместо всего дисплея
35. #define LCD\_MOVERIGHT 0x04 // Движение вправо
36. #define LCD\_MOVELEFT 0x00 // Движение влево
37. // Флаги наборов функций
38. #define LCD\_8BITMODE 0x10 // 8-битный режим
39. #define LCD\_4BITMODE 0x00 // 4-битный режим
40. #define LCD\_2LINE 0x08 // 1 строчка
41. #define LCD\_1LINE 0x00 // 2 строчки
42. #define LCD\_5x10DOTS 0x04 // Режим символов 5x10
43. #define LCD\_5x8DOTS 0x00 // Режим символов 5x8
44. // Флаги подсветки
45. #define LCD\_BACKLIGHT 0x08 // Подсветка ON
46. #define LCD\_NOBACKLIGHT 0x00 // Подсветка OFF
47. #define En B00000100 // Бит включения
48. #define Rw B00000010 // Бит Read/Write
49. #define Rs B00000001 // Бит выбора регистра
50. class LC\_I2C {
51. public:
52. // Конструктор
53. //
54. // Parameters
55. // ----------
56. // lcd\_addr(uint8\_t): I2C адрес LCD дисплея
57. // lcd\_cols(uint8\_t): Кол-во столбцов на дисплее
58. // lcd\_rows(uint8\_t): Кол-во строк на дисплее
59. // charsize(uint8\_t): Размер симвлов, LCD\_5x10DOTS
60. // или LCD\_5x8DOTS
61. LC\_I2C(uint8\_t lcd\_addr, uint8\_t lcd\_cols, uint8\_t lcd\_rows,
62. uint8\_t charsize = LCD\_5x8DOTS);
63. // Обязательно вызывать после создания объекта дисплея
64. void begin();
65. // Очистить дисплей и сбросить положение курсора
66. void clear();
67. // Возврат курсора
68. void home();
69. // Выключить вывод символов
70. void noDisplay();
71. // Показывать символы на дисплее
72. void display();
73. // Не моргать курсором
74. void noBlink();
75. // Моргать курсором
76. void blink();
77. // Выключить курсор
78. void noCursor();
79. // Включить курсор
80. void cursor();
81. // Вывод строки
82. //
83. // Parameters
84. // ----------
85. // str(char\*): Указатель на начало строки
86. void print(char\* str);
87. // Сдвиг дисплея влево
88. void scrollDisplayLeft();
89. // Сдвиг дисплея вправо
90. void scrollDisplayRight();
92. //
93. void printLeft();
95. //
96. void printRight();
98. // Порядок текста слева-направо
99. void leftToRight();
101. // Порядок текста справа-налево
102. void rightToLeft();
104. // Инкремент сдвига
105. void shiftIncrement();
107. // Декремент сдвига
108. void shiftDecrement();
110. // Выключить подсветку
111. void noBacklight();
113. // Включить подсветку
114. void backlight();
116. // Возвращает состояние подсветки
117. //
118. // Returns
119. // -------
120. // bool: Состояние подсветки
121. bool getBacklight();
123. // Выровнять текст по правому краю курсора
124. void autoscroll();
126. // Выровнять текст по левому краю курсора
127. void noAutoscroll();
129. // Позволяет заполнить первые 8 байт CGRAM
130. // собственными символами
131. void createChar(uint8\_t, uint8\_t[]);
133. // Установаить положение курсора
134. void setCursor(uint8\_t, uint8\_t);
136. // Написать символ на дисплее
137. virtual size\_t write(uint8\_t);
139. // Отправить комманду
140. void command(uint8\_t);
141. // Включить мигание курсора
142. inline void blink\_on() { blink(); }
144. // Выключить мигание курсора
145. inline void blink\_off() { noBlink(); }
147. // Включить курсор
148. inline void cursor\_on() { cursor(); }
150. // Выключить курсор
151. inline void cursor\_off() { noCursor(); }
152. // ---===== Compatibility API function aliases =====---
153. // Штука для backlight() и nobacklight()
154. void setBacklight(uint8\_t new\_val);
155. // Штука для createChar()
156. void load\_custom\_character(uint8\_t char\_num, uint8\_t \*rows);
157. private:
158. // Отправить команду или данные
159. void send(uint8\_t, uint8\_t);
160. void write4bits(uint8\_t);
161. void expanderWrite(uint8\_t);
162. void pulseEnable(uint8\_t);
163. uint8\_t \_addr;
164. uint8\_t \_displayfunction;
165. uint8\_t \_displaycontrol;
166. uint8\_t \_displaymode;
167. uint8\_t \_cols;
168. uint8\_t \_rows;
169. uint8\_t \_charsize;
170. uint8\_t \_backlightval;
171. };
172. #endif // LC\_I2C\_H\_\_
173. // #include "LC\_I2C.h"
174. LC\_I2C::LC\_I2C(uint8\_t lcd\_addr, uint8\_t lcd\_cols,
175. uint8\_t lcd\_rows, uint8\_t charsize)
176. {
177. \_addr = lcd\_addr;
178. \_cols = lcd\_cols;
179. \_rows = lcd\_rows;
180. \_charsize = charsize;
181. \_backlightval = LCD\_BACKLIGHT;
182. }
183. void LC\_I2C::begin() {
184. Wire.begin();
185. \_displayfunction = LCD\_4BITMODE | LCD\_1LINE | LCD\_5x8DOTS;
186. if (\_rows > 1) {
187. \_displayfunction |= LCD\_2LINE;
188. }
189. if ((\_charsize != 0) && (\_rows == 1)) {
190. \_displayfunction |= LCD\_5x10DOTS;
191. }
192. delay(50); // импульс 50мс
193. // Устанавливаем RS и R/W на 0, чтобы начать команды
194. expanderWrite(\_backlightval); // выключить подсветку
195. delay(1000);
196. // Переключаем дисплей в 4-битовый режим
197. write4bits(0x03 << 4);
198. delayMicroseconds(4500); // ждем 4.5мс
199. // Вторая команда
200. write4bits(0x03 << 4);
201. delayMicroseconds(4500); // ждем 4.5мс
202. // Третья команда
203. write4bits(0x03 << 4);
204. delayMicroseconds(150);
205. // Окончательно устанавливаем дисплей в 4-битовый режим
206. write4bits(0x02 << 4);
207. // Устанавливаем кол-во строк, шрифт и т.д.
208. command(LCD\_FUNCTIONSET | \_displayfunction);
209. // Включаем дисплей без курсора и без мигания по умолчанию
210. \_displaycontrol = LCD\_DISPLAYON | LCD\_CURSOROFF | LCD\_BLINKOFF;
211. display();
212. // Очищаем дисплей
213. clear();
214. // Устанавлием направление текста (для нормальных языков)
215. \_displaymode = LCD\_ENTRYLEFT | LCD\_ENTRYSHIFTDECREMENT;
216. // Ждем дальнейших команд
217. command(LCD\_ENTRYMODESET | \_displaymode);
218. home();
219. }
220. // ---===== Высокоуровневые команды :3 =====---
221. void LC\_I2C::clear(){
222. command(LCD\_CLEARDISPLAY);
223. delayMicroseconds(2000);
224. }
225. void LC\_I2C::home(){
226. command(LCD\_RETURNHOME);
227. delayMicroseconds(2000);
228. }
229. void LC\_I2C::setCursor(uint8\_t col, uint8\_t row){
230. int row\_offsets[] = { 0x00, 0x40, 0x14, 0x54 };
231. if (row > \_rows) {
232. row = \_rows-1; // считаем с 0
233. }
234. command(LCD\_SETDDRAMADDR | (col + row\_offsets[row]));
235. }
236. void LC\_I2C::noDisplay() {
237. \_displaycontrol &= ~LCD\_DISPLAYON;
238. command(LCD\_DISPLAYCONTROL | \_displaycontrol);
239. }
240. void LC\_I2C::display() {
241. \_displaycontrol |= LCD\_DISPLAYON;
242. command(LCD\_DISPLAYCONTROL | \_displaycontrol);
243. }
244. void LC\_I2C::noCursor() {
245. \_displaycontrol &= ~LCD\_CURSORON;
246. command(LCD\_DISPLAYCONTROL | \_displaycontrol);
247. }
248. void LC\_I2C::cursor() {
249. \_displaycontrol |= LCD\_CURSORON;
250. command(LCD\_DISPLAYCONTROL | \_displaycontrol);
251. }
252. void LC\_I2C::noBlink() {
253. \_displaycontrol &= ~LCD\_BLINKON;
254. command(LCD\_DISPLAYCONTROL | \_displaycontrol);
255. }
256. void LC\_I2C::blink() {
257. \_displaycontrol |= LCD\_BLINKON;
258. command(LCD\_DISPLAYCONTROL | \_displaycontrol);
259. }
260. void LC\_I2C::scrollDisplayLeft(void) {
261. command(LCD\_CURSORSHIFT | LCD\_DISPLAYMOVE | LCD\_MOVELEFT);
262. }
263. void LC\_I2C::scrollDisplayRight(void) {
264. command(LCD\_CURSORSHIFT | LCD\_DISPLAYMOVE | LCD\_MOVERIGHT);
265. }
266. void LC\_I2C::leftToRight(void) {
267. \_displaymode |= LCD\_ENTRYLEFT;
268. command(LCD\_ENTRYMODESET | \_displaymode);
269. }
270. void LC\_I2C::rightToLeft(void) {
271. \_displaymode &= ~LCD\_ENTRYLEFT;
272. command(LCD\_ENTRYMODESET | \_displaymode);
273. }
274. void LC\_I2C::autoscroll(void) {
275. \_displaymode |= LCD\_ENTRYSHIFTINCREMENT;
276. command(LCD\_ENTRYMODESET | \_displaymode);
277. }
278. void LC\_I2C::noAutoscroll(void) {
279. \_displaymode &= ~LCD\_ENTRYSHIFTINCREMENT;
280. command(LCD\_ENTRYMODESET | \_displaymode);
281. }
282. void LC\_I2C::createChar(uint8\_t location,
283. uint8\_t charmap[]) {
284. location &= 0x7;
285. command(LCD\_SETCGRAMADDR | (location << 3));
286. for (int i=0; i<8; i++) {
287. write(charmap[i]);
288. }
289. }
290. void LC\_I2C::noBacklight(void) {
291. \_backlightval=LCD\_NOBACKLIGHT;
292. expanderWrite(0);
293. }
294. void LC\_I2C::backlight(void) {
295. \_backlightval=LCD\_BACKLIGHT;
296. expanderWrite(0);
297. }
298. bool LC\_I2C::getBacklight() {
299. return \_backlightval == LCD\_BACKLIGHT;
300. }
301. void LC\_I2C::print(char\* str){
302. while(\*str){
303. write((int) \*str++);
304. }
305. }
306. // ---===== Среднеуровневые команды =====---
307. inline void LC\_I2C::command(uint8\_t value) {
308. send(value, 0);
309. }
310. inline size\_t LC\_I2C::write(uint8\_t value) {
311. send(value, Rs);
312. return 1;
313. }
314. // ---===== Низкоуровневые команды =====---
315. void LC\_I2C::send(uint8\_t value, uint8\_t mode) {
316. uint8\_t highnib=value & 0xf0;
317. uint8\_t lownib=(value << 4) & 0xf0;
318. write4bits((highnib) | mode);
319. write4bits((lownib) | mode);
320. }
321. void LC\_I2C::write4bits(uint8\_t value) {
322. expanderWrite(value);
323. pulseEnable(value);
324. }
325. void LC\_I2C::expanderWrite(uint8\_t \_data){
326. Wire.beginTransmission(\_addr);
327. Wire.write((int)(\_data) | \_backlightval);
328. Wire.endTransmission();
329. }
330. void LC\_I2C::pulseEnable(uint8\_t \_data){
331. expanderWrite(\_data | En); // En high
332. delayMicroseconds(1); // импульс >450нс
333. expanderWrite(\_data & ~En); // En low
334. delayMicroseconds(50); // импульс > 37мкс
335. }
336. void LC\_I2C::load\_custom\_character(uint8\_t char\_num, uint8\_t \*rows){
337. createChar(char\_num, rows);
338. }
339. void LC\_I2C::setBacklight(uint8\_t new\_val){
340. if (new\_val) {
341. backlight(); // включить подсветку on
342. } else {
343. noBacklight(); // выключить подсветку off
344. }
345. }
346. uint8\_t strlen(char\* str){
347. uint8\_t size = 0;
348. while(\*str){
349. str++;
350. size++;
351. }
352. return size;
353. }
354. // #include "LC\_I2C.h"
355. LC\_I2C lcd(0x27, 16, 2);
356. byte cChars[][8] = {
357. {0b00010001,
358. 0b00010010,
359. 0b00010010,
360. 0b00011100,
361. 0b00010010,
362. 0b00010010,
363. 0b00010001,
364. 0b00000000}, // К
365. {0b00000000,
366. 0b00000000,
367. 0b00011110,
368. 0b00010001,
369. 0b00011110,
370. 0b00010001,
371. 0b00011110,
372. 0b00000000}, // в
373. {0b00000000,
374. 0b00000000,
375. 0b00010001,
376. 0b00010001,
377. 0b00011111,
378. 0b00010001,
379. 0b00010001,
380. 0b00000000}, // н
381. {0b00011111,
382. 0b00010000,
383. 0b00010000,
384. 0b00011110,
385. 0b00010001,
386. 0b00010001,
387. 0b00011111,
388. 0b00000000}, // Б
389. {0b00000100,
390. 0b00001010,
391. 0b00001010,
392. 0b00001010,
393. 0b00001010,
394. 0b00011111,
395. 0b00010001,
396. 0b00000000}, // Д
397. {0b00000000,
398. 0b00000000,
399. 0b00001010,
400. 0b00011111,
401. 0b00011111,
402. 0b00001110,
403. 0b00000100,
404. 0b00000000}, // heart\_big
405. {0b00000000,
406. 0b00000000,
407. 0b00000000,
408. 0b00001010,
409. 0b00011111,
410. 0b00001110,
411. 0b00000100,
412. 0b00000000}, // heart\_small
413. {0b00000000,
414. 0b00000000,
415. 0b00010001,
416. 0b00010011,
417. 0b00010101,
418. 0b00010101,
419. 0b00011001,
420. 0b00000000}, // и
421. {0b00001110,
422. 0b00011111,
423. 0b00011100,
424. 0b00011000,
425. 0b00011100,
426. 0b00011111,
427. 0b00001110,
428. 0b00000000}, // pacman\_open
429. {0b00001110,
430. 0b00011111,
431. 0b00011111,
432. 0b00011111,
433. 0b00011111,
434. 0b00011111,
435. 0b00001110,
436. 0b00000000}, // pacman\_closed
437. {0b00000000,
438. 0b00000000,
439. 0b00000110,
440. 0b00001111,
441. 0b00001111,
442. 0b00000110,
443. 0b00000000,
444. 0b00000000}, // pacman\_dot
445. };
446. void setup(){
447. lcd.begin();
448. lcd.home();
449. lcd.backlight();
450. lcd.createChar(0, cChars[5]);
451. lcd.createChar(1, cChars[6]);
452. lcd.createChar(3, cChars[8]);
453. lcd.createChar(4, cChars[9]);
454. lcd.createChar(5, cChars[10]);
455. }
456. void loop(){
457. int speed\_div = 500;
458. lcd.home();
460. lcd.write(3);
461. lcd.write(5);
462. lcd.write(5);
463. lcd.write(5);
464. lcd.write(5);
465. lcd.write(5);
466. lcd.write(5);
467. lcd.write(5);
468. lcd.write(5);
469. lcd.write(5);
470. lcd.write(5);
471. lcd.write(5);
472. lcd.write(5);
473. lcd.write(5);
474. lcd.write(5);
475. lcd.write(5);
476. lcd.write(5);
477. // Задержка перед началом сдвига
478. delay(speed\_div);
479. // Цикл для сдвига текста вправо
480. for (int positionCounter = 0; positionCounter < 16; positionCounter++) {
481. // Сдвиг вправо
482. lcd.scrollDisplayRight();
483. if (positionCounter % 2 == 0){
484. lcd.createChar(3, cChars[9]);
485. } else {
486. lcd.createChar(3, cChars[8]);
487. }
488. // Задержка между каждым сдвигом
489. delay(speed\_div);
490. }
491. // Очистка дисплея перед следующим циклом
492. lcd.clear();
493. }