```
000 // Черноок Анастасия Юрьевна
001 // БГУиР. ЭВМ. 150501
002 // Микропроцессорное устройство обработки сигналов управления
003
004 #include "nRF24L01.h" // библиотека для nRF24L01+
005 #include "RF24.h"     // библиотека для радио модуля
006
007 #include <Adafruit_PWMServoDriver.h> // библиотека для PWM модуля
008 #include <LiquidCrystal_I2C.h>       // библиотека для дисплея
009
010 #define PIPE_ID 0xF0F1F2F3F4LL  // идентификатор передачи
011 #define PIN_CE  9               // номер вывода, к которому подключен вывод CE радиомодуля
012 #define PIN_CSN 10              // номер вывода, к которому подключен вывод CSN радиомодуля
013 RF24 radio(PIN_CE, PIN_CSN);    // создание объекта radio с указанием выводов CE и CSN
014 #define RADIO_CHANNEL 0         // используемый канал (м.б. установлен от 0 до 127) - 2,4 ГГц
015
016 #define PIPE_ID_STR "0xF0F1F2F3F4"
017 #define RADIO_FREQ_STR "2.4GHz"
018 #define RADIO_SPEED_STR "1MBPS"
019
020 #define AXIS_NUM 6       // количество осей вращения
021 uint8_t data[AXIS_NUM];  // массив для хранения полученных данных
022
023 // создание объекта pwm1 для управления сервоприводами с помощью pca9685
024 Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver();
025 #define START_SERVO_045 310    // значения для установления начальных позиций сервоприводов
026 #define START_SERVO_12 340
027 #define START_SERVO_3 390
028
029 // частота работы модуля
030 #define PCA_FREQ 50
031 // момент цикла из 4096 частей для установления низкого уровня ШИМ, соответствующий
032 // установке сервопривода на угол поворота 0°
033 #define PWM_MIN_BOUND 100
034 // момент цикла из 4096 частей для установления низкого уровня ШИМ, соответствующий
035 // установке сервопривода на угол поворота 180°
036 #define PWM_MAX_BOUND 520
037
038 #define MIN_DEGREE 0         // минимальный возможный угол поворота сервопривода
039 #define MAX_DEGREE 180       // максимальный возможный угол поворота сервопривода
040
041 #define AXIS_1_MIN_BOUND 50   // минимальный угол поворота на оси 1
042 #define AXIS_2_MAX_BOUND 140  // максимальный угол поворота на оси 2
043 #define AXIS_5_MIN_BOUND 50   // минимальный угол поворота на оси 5
044 #define AXIS_5_MAX_BOUND 140  // максимальный угол поворота на оси 5
045
046
047 #define LCD_I2C_ADDRESS 0x27  // I2C адрес дисплея
048 #define LCD_COLUMS_NUM 16     // количество столбцов в дисплее
049 #define LCD_ROWS_NUM 2        // количество строк в дисплее
050 // создание объекта для управления дисплеем с помощью I2C интерфейса
051 LiquidCrystal_I2C lcd(0x27,16,2);
052
053 #define SWITCH1_PIN 3         // номер вывода, к которому подключена первая кнопка
054 #define SWITCH2_PIN 4         // номер вывода, к которому подключена вторая кнопка
055 boolean lastButton_1 = LOW;   // предыдущее состояние первой кнопки
056 boolean lastButton_2 = LOW;   // предыдущее состояние второй кнопки
057 boolean currButt_1 = LOW;     // текущее состояние первой кнопки
058 boolean currButt_2 = LOW;     // текущее состояние второй кнопки
059
060 #define LED_PIN_GREEN 5       // номер вывода, к которому подключен зеленый светодиод
061 #define LED_PIN_YELLOW 6      // номер вывода, к которому подключен желтый светодиод
062 #define LED_PIN_BLUE 7        // номер вывода, к которому подключен синий светодиод
063 boolean ledOn_green = false;  // состояние зеленого светодиода
064 boolean ledOn_blue = false;   // состояние синего светодиода
065
066 #define BLINKS_NUM 2
```

```
067 #define BLINK_ON 250
068 #define BLINK_OFF 380
069
070 // прием данных разрешен/запрещен
071 boolean data_receiving_is_on = true;
072 // на дисплей выводятся углы установки сервоприводов
073 boolean lcd_vals_mode_is_on = true;
074
075 // Функция установки значений и параметров
076 void setup() {
077
078   // РАДИОМОДУЛЬ
079   radio.begin();                      // инициализация модуля NRF24L01
080   radio.setChannel(RADIO_CHANNEL);    // выбираем канал
081   radio.setDataRate(RF24_1MBPS);      // скорость передачи данных 1 Мбит/сек
082   radio.setPALevel(RF24_PA_HIGH);
083   // устанавливаем высокую мощность передатчика (-6dBm)
084   radio.openReadingPipe(1, PIPE_ID);
085   // открываем одну трубу с уникальным ID
086   radio.startListening();
087   // начинаем прослушивать трубу
088
089   // ШИМ МОДУЛЬ
090   pwm1.begin();
091   // инициализация I2C интерфейса для 16-тиканального ШИМ модуля
092   pwm1.setPWMFreq(PCA_FREQ);          // устанавливаем частоту
093
094   pwm1.setPWM(0, 0, START_SERVO_045);
095   // установка начальных положений сервоприводов
096   pwm1.setPWM(1, 0, START_SERVO_12);
097   // 1-й параметр: выбор сервопривода
098   pwm1.setPWM(2, 0, START_SERVO_12);
099   // 2-й параметр: на какой момент цикла из 4096 частей задать активный уровень ШИМ
100   // 3-й параметр: на какой момент цикла из 4096 частей задать низкий уровень ШИМ
101   pwm1.setPWM(3, 0, START_SERVO_3);
102   pwm1.setPWM(4, 0, START_SERVO_045);
103   pwm1.setPWM(5, 0, START_SERVO_045);
104
105   // ДИСПЛЕЙ
106   lcd.init();                      // инициализация I2C интерфейса для дисплея
107   lcd.backlight();                 // включение подсветки дисплея
108   lcd.print("Hello =)");
109
110   // ВЫВОДЫ
111   pinMode(SWITCH1_PIN, INPUT);        // установка режима ввода для кнопок
112   pinMode(SWITCH2_PIN, INPUT);
113   pinMode(LED_PIN_GREEN, OUTPUT);     // установка режима вывода для светодиодов
114   pinMode(LED_PIN_YELLOW, OUTPUT);
115   pinMode(LED_PIN_BLUE, OUTPUT);
116
117   // Мигание желтым светодиодом в начале работы устройства
118   int i = 0;
119   for (; i < BLINKS_NUM; i++) {
120     digitalWrite(LED_PIN_YELLOW, HIGH);
121     delay(BLINKS_ON);
122     digitalWrite(LED_PIN_YELLOW, LOW);
123     delay(BLINKS_OFF);
124   }
125   digitalWrite(LED_PIN_YELLOW, HIGH);
126 } // void setup()
127
128 // Функция вывода на дисплей полученных по радиоканалу чисел
129 void print_degrees() {
130   int i = 0;
131   int j = 0;
132   for (; i < AXIS_NUM; i++) {
133     if (i % 2 == 0)
```

```
      lcd.setCursor(j, 0);
    else {
     lcd.setCursor(j, 1);
     j += 5;
    }
    lcd.print(i + 1);
    lcd.print(":");
    lcd.print(data[i]);
    lcd.print(" ");
  } // for (; i < AXIS_NUM; i++)
}

// Функция для обработки нажатия кнопки (с учетом дребезжания)
boolean debounce(boolean last, int swPin) {
  boolean current = digitalRead(swPin);
  if (last != current) {
    delay(5);
    current = digitalRead(swPin);
  }
  return current;
}

// Функция вывода настроек радиомодуля
void print_radio_info() {
  lcd.setCursor(0, 0);
  lcd.print("ID:");
  lcd.print(PIPE_ID_STR);

  lcd.setCursor(0, 1);
  lcd.print("F-");
  lcd.print(RADIO_FREQ_STR);

  lcd.setCursor(9, 1);
  lcd.print("v-");
  lcd.print(RADIO_SPEED_STR);
}


// Главная фунция, зацикленно выполняющаяся устройством
void loop() {

  // Проверка разрешения на прием данных
  if (data_receiving_is_on == true) {

    // Проверяем буфер обмена
    if (radio.available()) {

      radio.read(&data, sizeof(data)); // читаем данные из канала

      digitalWrite(LED_PIN_BLUE, HIGH);

      if (lcd_vals_mode_is_on == true)
        print_degrees();
      else
        print_radio_info();

      // Установка углов поворотов сервоприводов с учетом заданных углов ограничений
      pwm1.setPWM(0, 0, map(data[0],
      MIN_DEGREE, MAX_DEGREE, PWM_MIN_BOUND, PWM_MAX_BOUND));

      if (data[1] > AXIS_1_MIN_BOUND) {
        pwm1.setPWM(1, 0, map(data[1],
        MIN_DEGREE, MAX_DEGREE, PWM_MIN_BOUND, PWM_MAX_BOUND));
        pwm1.setPWM(2, 0, map(data[1],
        MIN_DEGREE, MAX_DEGREE, PWM_MIN_BOUND, PWM_MAX_BOUND));
      }

```

```
      if (data[2] < AXIS_2_MAX_BOUND)
        pwm1.setPWM(3, 0, map(data[2],
        MIN_DEGREE, MAX_DEGREE, PWM_MIN_BOUND, PWM_MAX_BOUND));

      pwm1.setPWM(4, 0, map(data[3],
      MIN_DEGREE, MAX_DEGREE, PWM_MIN_BOUND, PWM_MAX_BOUND));

      pwm1.setPWM(5, 0, map(data[4],
      MIN_DEGREE, MAX_DEGREE, PWM_MIN_BOUND, PWM_MAX_BOUND));

      if (data[5] > AXIS_5_MIN_BOUND && data[5] < AXIS_5_MAX_BOUND)
        pwm1.setPWM(6, 0, map(data[5],
        MIN_DEGREE, MAX_DEGREE, PWM_MIN_BOUND, PWM_MAX_BOUND));


      delay(20);
    } //if (radio.available())
    else
      digitalWrite(LED_PIN_BLUE, LOW);

  } // if (data_receiving_is_on == true)
  else
    digitalWrite(LED_PIN_BLUE, LOW);


  // Обрабатываем нажатие кнопки смены выводимой на экран информации
  currButt_1 = debounce(lastButton_1, SWITCH1_PIN);
  if (lastButton_1 == LOW && currButt_1 == HIGH){

    lcd_vals_mode_is_on = !lcd_vals_mode_is_on;
    if (lcd_vals_mode_is_on == false)
      print_radio_info();
  }
  lastButton_1 = currButt_1;

  // Обрабатываем нажатие кнопки разрешения приема данных
  currButt_2 = debounce(lastButton_2, SWITCH2_PIN);
  if (lastButton_2 == LOW && currButt_2 == HIGH){
    data_receiving_is_on = !data_receiving_is_on;
  }
  lastButton_2 = currButt_2;
  digitalWrite(LED_PIN_GREEN, data_receiving_is_on);

}
```