Испытание проделанной работы

В данной работе для дистанционного контроля и мониторинга стендов использовалось *Android* приложение «*MQTT-Dashboard*». Приложение было настроено на отправление и получение данных от сервера.

Отправка снятых показаний на сервер

Цель: убедиться в корректной отправке токопотребления на сервер.

Вместо стенда был подключен резистор, с которого непрерывно снимаются показания и отправляются на сервер:

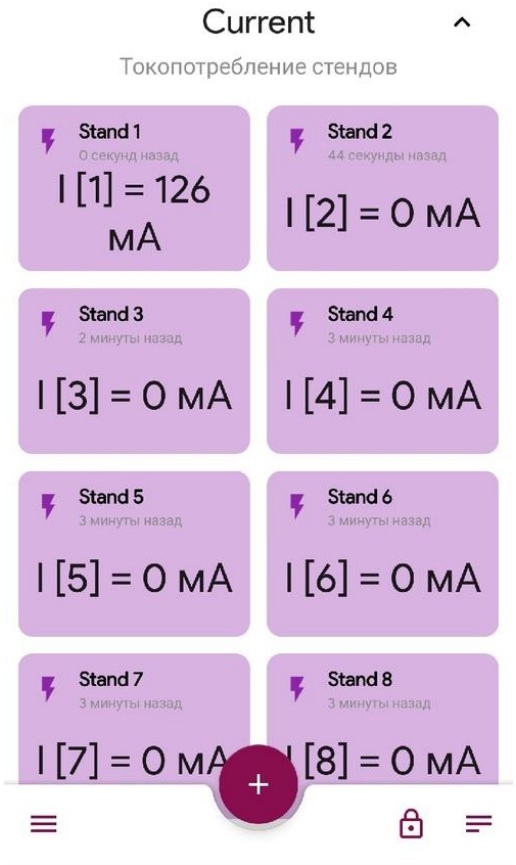

Рис. 1. Снятые показания с резистора

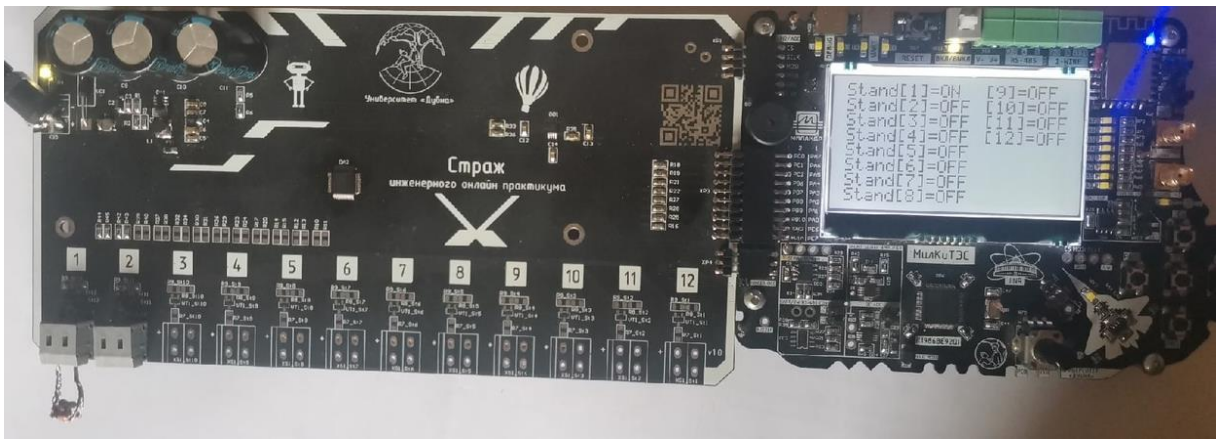

Рис. 2. Подключение резистора

Таким образом можно убедиться в том, что микроконтроллер успешно снимает показания с АЦП и отправляет их на сервер. В эксплуатации это выглядит следующим образом:

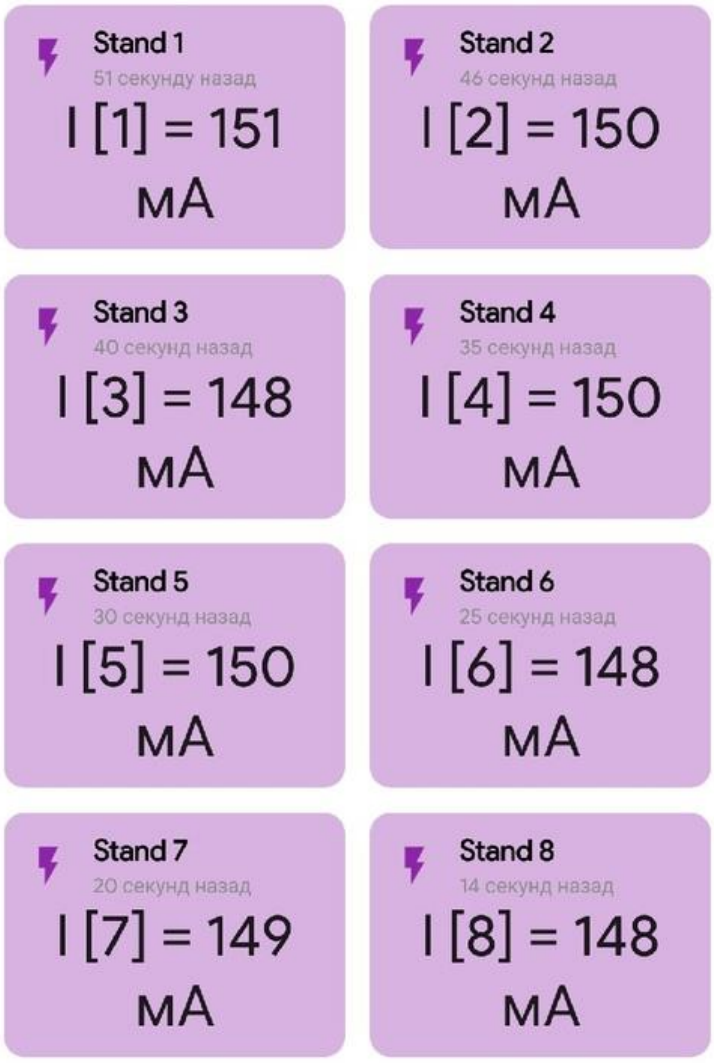

Рис. 3. Снятые показания со всех стендов

Дистанционный контроль стендов

На сервере имеются топики, управляющие питанием стендов. Всего 12 таких топиков, т.к. к комплексу «Страж» можно подключать только до 12 плат.

Цель: убедиться в корректной отправке и обработке запроса от сервера на плату, отвечающую за автоматизацию Стража.

Изначально все стенды включены и их состояние непрерывно отслеживает микроконтроллер:

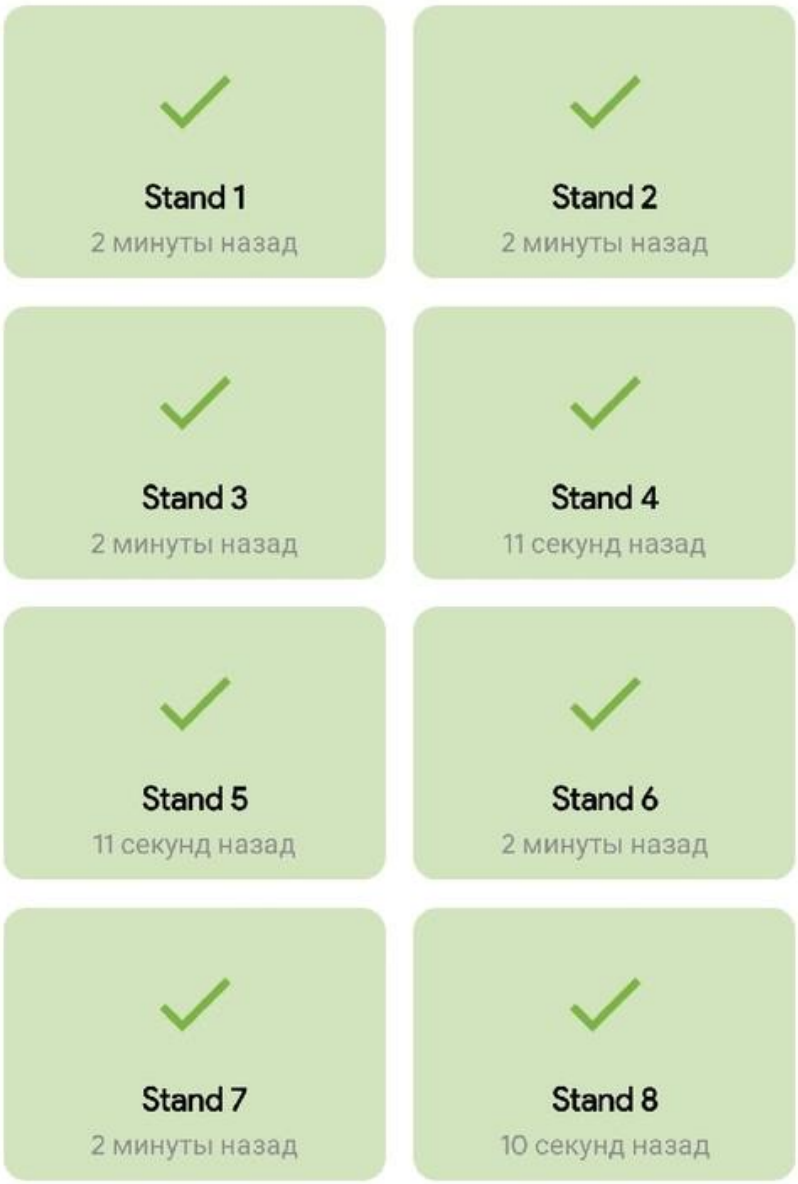

Рис. 4. Состояние стендов на сервере

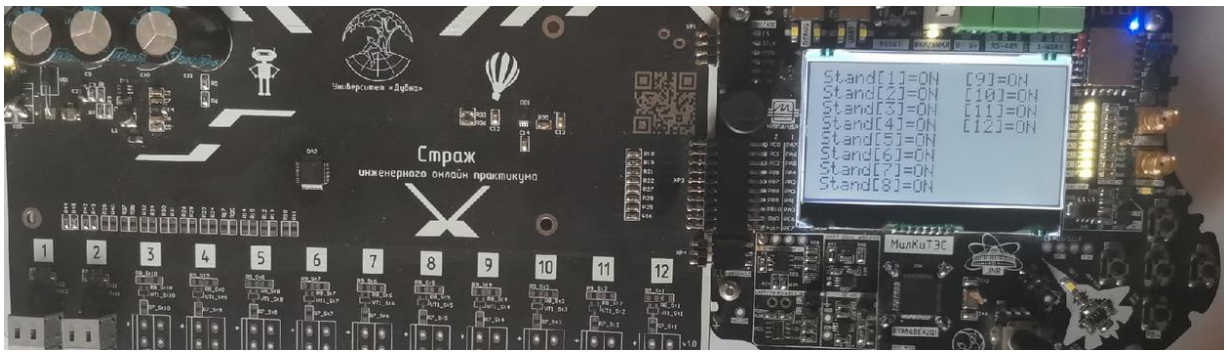

Рис. 5. Индикация дисплея

Далее через приложение был отключен 4,5 и 8 стенд:

Рис. 6. Измененное состояние стендов на сервере

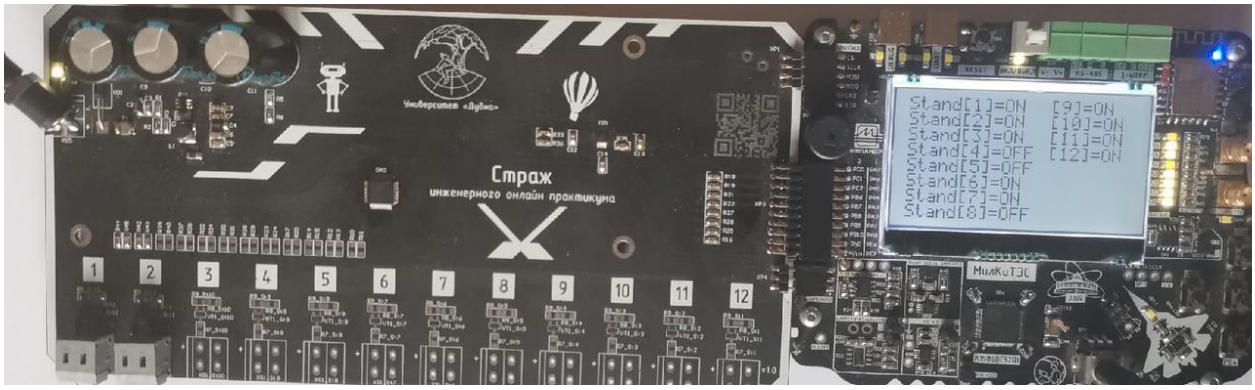

Рис. 7. Измененная индикация

После отправки запроса на микроконтроллер, состояние транзисторов, отвечающих за питание 4,5 и 8 стенда, моментально инвертировали свое состояние.

Вывод

Программное обеспечение моментально реагирует на запрос от сервера и непрерывно отправляет снятые показания.

Результат работы программы полностью удовлетворяет поставленную задачу.

Заключение

В результате выполнения бакалаврской работы разработан автоматизированный контроль печатных плат для учебно-демонстрационного комплекса «Страж». Система контроля была оснащена технологией «Интернета вещей», благодаря которой у пользователя появилась возможность дистанционного взаимодействия с учебным стендом.

В ходе работы были решены следующие задачи:

1. Автоматизированный контроль печатных плат:
    - инициализация АЦП и настройка обратной связи по напряжению;
    - установка канала мультиплексора;
    - контроль транзисторов, отвечающих за питание стендов;
    - запуск таймера и создание системы опроса печатных плат по прерыванию;
    - вывод снятых показаний на дисплей;
    - обработка показаний с АЦП.
2. Символьная система команд по UART:
    - настройка интерфейса UART на двух микроконтроллерах;
    - модель взаимодействия «Ведущий – ведомый»;
    - прерывание по поступлению байта в буфер приемника;
    - обработка алгоритма соответствующей команде.
3. Дистанционный контроль стенда:
    - настройка MQTT брокера;
    - передача показаний с АЦП на сервер;
    - прием команды от сервера (по типу – выключить стенд 1).

В процессе работы были приобретены и закреплены следующие навыки:

1. Изучены принципы работы следующих систем:
    - интерфейс UART – обмен данными двух микроконтроллеров;
    - интерфейс SPI – вывод информации на дисплей
    - оцифровка сигнала с помощью АЦП;
    - таймеры общего назначения.
2. Получены знания:
    - в программировании микроконтроллеров K1986BE92QI;

- в программировании микроконтроллеров: ESP12-F, ESP8266;
- в протоколах обмена данными технологией «Интернета вещей»;
- в организации удаленного доступа к электронному устройству.

3. Закреплены навыки работы:
    - с программатором «ST-Link»;
    - с программным кодом в среде Keil и Arduino;
    - с документациями и даташитами;
    - в отладке микроконтроллера;
    - с контрольно-измерительным оборудованием (мультиплексором, осцилогрофом);

Список литературы

1. Сахаров Ю.С. Горбунов Н.В. Люосев Д.А. Понкин Д.О. Шириков И.В. Основы программирования микроконтроллеров серии 1986BE9X в среде Keil uVision: учебное пособие. Университет Дубна, 2017. – 134с;
2. Проток MQTT [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/463669/;
3. Спецификация микроконтроллеров серии 1986BE9x, К1986BE9x, К1986BE92QI, К1986BE92QC, К1986BE91H4. ЗАО «ПКК Миландр» − Версия 3.4.3 от 22.07.2013: http://milandr.ru/uploads/Products/product_80/spec_seriya_1986BE9x.pdf;
4. *Datasheet* на TPS561208 [Электронный ресурс] – Режим доступа: https://www.ti.com/lit/ds/symlink/tps561201.pdf?ts=1623403111660&ref_url=https%253A%252F%252Fwww.google.com%252F;
5. Протоколы передачи данных *IoT* [Электронный ресурс] – Режим доступа: https://iot.ru/wiki/protokoly-peredachi-dannykh-iot;
6. Спецификация на ADG726 (мультиплексор) [Электронный ресурс] – Режим доступа: https://www.analog.com/media/en/technical-documentation/data-sheets/ADG726_732.pdf;
7. Репозиторий по программированию микроконтроллеров серии 1986BE9x, К1986BE9x, К1986BE92QI, К1986BE92QC, К1986BE91H4 [Электронный ресурс] – Режим доступа: https://github.com/owlatarms/mpb;
8. ESP8266 управление через интернет по протоколу MQTT [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/393277/;
9. *Datasheet* на *RTR*025*N*03*TL* [Электронный ресурс] – Режим доступа: https://pdf1.alldatasheet.com/datasheet-pdf/view/521031/ROHM/RTR025N03TL.html;
10. *Datasheet* на *INA225AIDGKT* [Электронный ресурс] – Режим доступа: https://pdf1.alldatasheet.com/datasheet-pdf/view/737768/TI/INA225AIDGKT.html.

Keil: main.c

```c
#include "main.h"   // подключение заголовка микроконтроллера
//#define ESP
#define Milandr

uint8_t stand[STANDS];
uint8_t In_data;  // входные данные

#define DELAY_ADC 50  // задержка между измерением показаний с АЦП

// Основная функция
int main(void)
{
  #ifdef Milandr
  Init_per();
  while (true)
  {
    Indication();
  }
  #endif

  #ifdef ESP
  while(true){}
  #endif
}
```

```c
/**
  * @brief      Прерывание UART1
  * @detailed   Прерывание возникает при поступлении 1 байта
  * @param      Функция не принимает параметры
  * @return     Функция не возвращает параметры
  *  */
void UART1_IRQHandler(void)
{
  while(MDR_UART1->FR & 1<<6)
  {
    In_data = MDR_UART1->DR;          // чтение данных

    // Обработка команды.             По UART были замечены помехи с этой целью были добавлены условия
    if (((In_data/10)<=STANDS)&&((In_data/10)>0))
    {
      if (In_data%10==1)
      {
        inquiry_stand(In_data);
        stand[(In_data/10)-1]=1;
      }
      else if (In_data%10==0)
      {
        inquiry_stand(In_data);
        stand[(In_data/10)-1]=0;
      }
    }

    //stand[(In_data/10)-1]=In_data%10; // обработка команды
  }
  MDR_UART1->ICR = 1 << 4;            // сброс прерывания
}
```

Keil: main.c

```
void Indication() // ф-я вывода состояния стэндов
{
  uint8_t page=0;
  for (uint8_t i=0;i<STANDS;i++)
  {
    if (page<8)
    {
      LCD_page_set(page++);
      LCD_column_set(0);
      LCD_print_text("Stand[");
      LCD_print_num(i+1);
      LCD_print_text("]=");
      if (stand[i]==1)
        LCD_print_text("ON ");
      else
        LCD_print_text("OFF");
    }
    else
    {
      LCD_page_set(page++-8);
      LCD_column_set(80);
      LCD_print_text("[");
      LCD_print_num(i+1);
      LCD_print_text("]=");
      if (stand[i]==1)
        LCD_print_text("ON ");
      else
        LCD_print_text("OFF");
    }
  }
}
```

```
/**
 * @brief      Обработка полученного запроса по UART
 * @detailed   Ф-я выполняет полученную команду по UART
 * @param      @data - полученные запрос
 * @return     Функция не возвращает параметры
 **/
void inquiry_stand(uint8_t data)
{
  if (((In_data/10)<=STANDS)&&((In_data/10)>0)) // если требуется вкл./выкл. стенд
  {
    switch(In_data/10)
    {
      case 1:                       // Стенд 1
        if (In_data%10)
          S1_EN;
        else
          S1_DIS;
        break;

      case 2:                       // Стенд 2
        if (In_data%10)
          S2_EN;
        else
          S2_DIS;
        break;

      case 3:                       // Стенд 3
        if (In_data%10)
          S3_EN;
        else
          S3_DIS;
        break;

      case 4:                       // Стенд 4
        if (In_data%10)
          S4_EN;
```

Keil: main.c

```
123       case 4:                          // Стенд 4
124         if (In_data%10)
125           S4_EN;
126         else
127           S4_DIS;
128         break;
129
130       case 5:                          // Стенд 5
131         if (In_data%10)
132           S5_EN;
133         else
134           S5_DIS;
135         break;
136
137       case 6:                          // Стенд 6
138         if (In_data%10)
139           S6_EN;
140         else
141           S6_DIS;
142         break;
143
144       case 7:                          // Стенд 7
145         if (In_data%10)
146           S7_EN;
147         else
148           S7_DIS;
149         break;
150
151       case 8:                          // Стенд 8
152         if (In_data%10)
153           S8_EN;
154         else
155           S8_DIS;
156         break;
```

```
157
158       case 9:                          // Стенд 9
159         if (In_data%10)
160           S9_EN;
161         else
162           S9_DIS;
163         break;
164
165       case 10:                         // Стенд 10
166         if (In_data%10)
167           S10_EN;
168         else
169           S10_DIS;
170         break;
171
172       case 11:                         // Стенд 11
173         if (In_data%10)
174           S11_EN;
175         else
176           S11_DIS;
177         break;
178
179       case 12:                         // Стенд 12
180         if (In_data%10)
181           S12_EN;
182         else
183           S12_DIS;
184         break;
185     }
186   }
187 }
```

Keil: main.h

main.c* | main.h | UART.h | Straj.h | Init.h | ADC.h | Timer.h | 1986VE9x.h

```
#include <1986VE9x.h>                    // Подключение заголовка микроконтроллера
#define   F_CPU 8000000                  // Указание тактовой частоты МК
#define STANDS  12                       // кол-во стэндов
#include  "milkites_delay.h"             // Подключение библиотеки задержек
#include  "milkites_display.h"           // Подключение библиотеки дисплея
#include  "UART.h"                       // UART
#include  "ADC.h"                        // Поключение функционала АЦП
#include  "Straj.h"                      // Подключения функционала мультиплексора
#include  "Timer.h"                      // Подлючения функционала таймеров
#include  "Init.h"                       // Инициализация МК

#define LCD_led_en  MDR_PORTE->RXTX |=  (1<<2)  // вкл. подсветки
#define LCD_led_dis  MDR_PORTE->RXTX &= ~(1<<2) // выкл. подсветки

void Indication(); // ф-я вывода состояния стэндов
void inquiry_stand(uint8_t data);// обработка полученного запроса по UART
```

Keil: UART.h

```c
/**
  * @brief      Отправка байта данных по UART1
  * @detailed
  * @param      @byte - байт данных для отправки по UART1
  * @return     Функция не возвращает параметры
  */
void uart_send_byte(uint32_t byte)
{
    // ожидание готовности UART1 для передачи байта данных
    while (MDR_UART1->FR & (1 << 5)) { }
    MDR_UART1->DR = byte; // отправка байта данных
}
```

```c
/**
  * @brief      Инициализация модуля UART1
  * @detailed
  * @param      Функция не принимает параметров
  * @return     Функция не возвращает параметры
  */
void uart_init(void)
{
  // режим работы порта PB5, PB6 - UART1 (Режим работы вывода порта - альтернативная ф-я)
  MDR_PORTB->FUNC  |= ((2 << 12) | (2 << 10));

  MDR_PORTB->ANALOG |= ((1 << 6) | (1 << 5));  // цифровой порт
  MDR_PORTB->PWR   |= ((3 << 12) | (3 << 10)); // максимально быстрый фронт

  MDR_RST_CLK->PER_CLOCK |= (1 << 6);          // вкл. тактирование UART1
  MDR_RST_CLK->UART_CLOCK = (0                 // установка делителя UART1 = 1
                           | (0 << 8)          // установка делителя UART2 = 1
                           | (1 << 24)         // разрешение тактирования UART1
                           | (0 << 25));       // запрет тактовой частоты UART2

  // Параметры делителя при частоте = 8 МГц и скорости 115200
  MDR_UART1->IBRD = 4;                         // целая часть делителя скорости
  MDR_UART1->FBRD = 22;                        // дробная часть делителя скорости
  MDR_UART1->LCR_H = ( (0 << 1)                // работа без проверки четности
                     | (0 << 2)                // бит четности откличен
                     | (0 << 3)                // кол-во стоповых бит = 1
                     | (0 << 4)                // буфер FIFO выключен
                     | (3 << 5)                // размер кадра - 8 бит
                     | (0 << 7));              // передача бита четности запрещена

  // передатчик и приемник разрешен, разрешение приемопередатчика UART1
  MDR_UART1->CR = ( (1 << 8) | (1 << 9) | 1);

  MDR_UART1->IMSC = 1 << 4;  // разрешение прерывания от приемника UARTR1XINTR
  NVIC_EnableIRQ(UART1_IRQn); // разрешение прерывания от модуля UART1
}
```

Keil: Straj.h

```
//---------------вкл./выкл. необходимого пина----------------------
#define PINA_ON(x)   MDR_PORTA->RXTX |=  (1<<x)  // вкл. пин x  PORTA
#define PINA_OFF(x)  MDR_PORTA->RXTX &= ~(1<<x) // выкл.пин x  PORTA

#define PINB_ON(x)   MDR_PORTB->RXTX |=  (1<<x)  // вкл. пин x  PORTB
#define PINB_OFF(x)  MDR_PORTB->RXTX &= ~(1<<x) // выкл.пин x  PORTB

#define PINC_ON(x)   MDR_PORTC->RXTX |=  (1<<x)  // вкл. пин x  PORTC
#define PINC_OFF(x)  MDR_PORTC->RXTX &= ~(1<<x) // выкл.пин x  PORTC

#define PIND_ON(x)   MDR_PORTA->RXTX |=  (1<<x)  // вкл. пин x  PORTD
#define PIND_OFF(x)  MDR_PORTA->RXTX &= ~(1<<x) // выкл.пин x  PORTD

#define PINF_ON(x)   MDR_PORTF->RXTX |=  (1<<x)  // вкл. пин x  PORTF
#define PINF_OFF(x)  MDR_PORTF->RXTX &= ~(1<<x) // выкл.пин x  PORTF

#define PINE_ON(x)   MDR_PORTE->RXTX |=  (1<<x)  // вкл. пин x  PORTE
#define PINE_OFF(x)  MDR_PORTE->RXTX &= ~(1<<x) // выкл.пин x  PORTE
//-----------------------------------------------------------------

/*----------------Директивы "Страж"-----------------------------------------
#define MX_EN     PIND_OFF(2)            // вкл. мультиплексор
#define MX_DIS    PIND_ON(2)             // выкл. мультиплексор

#define MX_CSa_EN   PINF_ON(7)           // 1 в MX_CSa (PE7)
#define MX_CSa_DIS  PINF_OFF(7)          // 0 в MX_CSa (PE7)

#define MX_CSb_EN   PINF_ON(0)           // 1 в MX_CSb (PE0)
#define MX_CSb_DIS  PINF_OFF(0)          // 0 в MX_CSb (PE0)
```

```
#define S12_EN     PINA_ON(0)            // ВКЛ.   стенд 1
#define S12_DIS    PINA_OFF(0)           // ВЫКЛ.  стенд 1
#define S11_EN     PINA_ON(1)            // ВКЛ.   стенд 2
#define S11_DIS    PINA_OFF(1)           // ВЫКЛ.  стенд 2
#define S10_EN     PINA_ON(2)            // ВКЛ.   стенд 3
#define S10_DIS    PINA_OFF(2)           // ВЫКЛ.  стенд 3
#define S9_EN     PINA_ON(3)             // ВКЛ.   стенд 4
#define S9_DIS    PINA_OFF(3)            // ВЫКЛ.  стенд 4
#define S8_EN     PINA_ON(4)             // ВКЛ.   стенд 5
#define S8_DIS    PINA_OFF(4)            // ВЫКЛ.  стенд 5
#define S7_EN     PINA_ON(5)             // ВКЛ.   стенд 6
#define S7_DIS    PINA_OFF(5)            // ВЫКЛ.  стенд 6
#define S6_EN     PINA_ON(6)             // ВКЛ.   стенд 7
#define S6_DIS    PINA_OFF(6)            // ВЫКЛ.  стенд 7
#define S5_EN     PINA_ON(7)             // ВКЛ.   стенд 8
#define S5_DIS    PINA_OFF(7)            // ВЫКЛ.  стенд 8
#define S4_EN     PINB_ON(8)             // ВКЛ.   стенд 9
#define S4_DIS    PINB_OFF(8)            // ВЫКЛ.  стенд 9
#define S3_EN    PINB_ON(9)              // ВКЛ.   стенд 10
#define S3_DIS   PINB_OFF(9)             // ВЫКЛ.  стенд 10
#define S2_EN    PINB_ON(10)             // ВКЛ.   стенд 11
#define S2_DIS   PINB_OFF(10)            // ВЫКЛ.  стенд 11
#define S1_EN    PINB_ON(7)              // ВКЛ.   стенд 12
#define S1_DIS   PINB_OFF(7)             // ВЫКЛ.  стенд 12
//-----------------------------------------------------------------
/*-----------------------------------------------------------------
```

Keil: Straj.h

```
/**
  * @brief      Установка канала мультиплексора
  * @detailed
  * @param      @channel - номер стенда
  * @return     Функция не возвращает параметры
  */
void MX_set_channel(uint8_t channel)
{
  switch(channel)
  {
    case 12:                //Стенд 12  0000
      PINC_OFF(0);
      PINC_OFF(1);
      PINC_OFF(2);
      PINB_OFF(6);
      break;
    case 11:                //Стенд 11  0001
      PINC_OFF(0);
      PINC_OFF(1);
      PINC_OFF(2);
      PINB_ON(6);
      break;
    case 10:                //Стенд 10  0010
      PINC_OFF(0);
      PINC_OFF(1);
      PINC_ON(2);
      PINB_OFF(6);
      break;
    case 9:                 //Стенд 9  0011
      PINC_OFF(0);
      PINC_OFF(1);
      PINC_ON(2);
      PINB_ON(6);
      break;
```

```
    case 8:                 //Стенд 8  0100
      PINC_OFF(0);
      PINC_ON(1);
      PINC_OFF(2);
      PINB_OFF(6);
      break;
    case 7:                 //Стенд 7  0101
      PINC_OFF(0);
      PINC_ON(1);
      PINC_OFF(2);
      PINB_ON(6);
      break;
    case 6:                 //Стенд 6  0110
      PINC_OFF(0);
      PINC_ON(1);
      PINC_ON(2);
      PINB_OFF(6);
      break;
    case 5:                 //Стенд 5  0111
      PINC_OFF(0);
      PINC_ON(1);
      PINC_ON(2);
      PINB_ON(6);
      break;
    case 4:                 //Стенд 4  1000
      PINC_ON(0);
      PINC_OFF(1);
      PINC_OFF(2);
      PINB_OFF(6);
      break;
    case 3:                 //Стенд 3 1001
      PINC_ON(0);
      PINC_OFF(1);
      PINC_OFF(2);
      PINB_ON(6);
      break;
    case 2:                 //Стенд 2 1010
```

Keil: Straj.h

```
153                 break;
154             case 2:                 // Стенд 2 1010
155                 PINC_ON(0);
156                 PINC_OFF(1);
157                 PINC_ON(2);
158                 PINB_OFF(6);
159                 break;
160             case 1:                 // Стенд 1 1011
161                 PINC_ON(0);
162                 PINC_OFF(1);
163                 PINC_ON(2);
164                 PINB_ON(6);
165                 break;
166         }
167     }
168
169
```

```
172 /**
173   * @brief    Инициализация "Страж"
174   * @detailed   Инициализация портов для взаимодействия
175   *             со "Стражем"
176   * @param    Функция не принимает параметров
177   * @return   Функция не возвращает параметры
178   **/
179 void Init_Straj(void)
180 {
181   /*
182   * PA0..PA7,PB8..PB10,PF5  - на выход, управление питанием с 1 по 12
183   * PC0-PC2,PB6             - на выход, установка канала мультиплексора
184   * PD2                     - на выход, вкл./выкл. мультиплексора
185   */
186
187   // подсветка дисплея управляется выводом PE2
188   MDR_PORTE->OE    = 0xff37; // биты 7,6,3 PORTE - входы, др. - выходы
189   MDR_PORTE->FUNC  = 0x0000; // функция - порт, основная функция
190   MDR_PORTE->PWR   = 0xff37; // макс. быстрый фронт
191   MDR_PORTE->ANALOG = 0xffff; // режим работы порта - цифровой ввод/вывод
192
193   /*-------------- Управление питанием ---------------------------------------*/
194   //-------------- PORTA ----------------------------------------
195   // Конфигурация линий PA0...PA7
196   MDR_PORTA->OE    |= 0x000000FF; // Вывод          (Направление передачи)
197   MDR_PORTA->FUNC  &= ~0x0000FFFF; // Ввод-вывод     (Функция)
198   MDR_PORTA->ANALOG |= 0x000000FF; // Цифровой       (Режим работы)
199   MDR_PORTA->PULL  &= ~0x00FF00FF; // Отключена      (Подтяжка)
200   MDR_PORTA->PD    &= 0x00FF00FF; // Драйвер        (Управление выводом)
201   MDR_PORTA->PWR   |= 0x0000FFFF; // Высокая        (Крутизна фронтов)
202   MDR_PORTA->GFEN  &= ~0x000000FF; // Не используется (Цифровой фильтр)
203   //------------------------------------------------------------------
204
```

```
205   //-------------- PORTB ----------------------------------------
206   // Конфигурация линий PB8..PB10
207   MDR_PORTB->OE    |= (1<<8)     // Вывод     PB8  (Направление передачи)
208                     |  (1<<9)     // Вывод     PB9
209                     |  (1<<10);   // Вывод     PB10
210   MDR_PORTB->FUNC  &= ~(3<<(2*8)); // Ввод-вывод PB8  (Функция)
211   MDR_PORTB->FUNC  &= ~(3<<(2*9)); // Ввод-вывод PB9
212   MDR_PORTB->FUNC  &= ~(3<<(2*10));// Ввод-вывод PB10
213   MDR_PORTB->ANALOG |= (1<<8)     // Цифровой   PB8  (Режим работы)
214                     |  (1<<9)     // Цифровой   PB9
215                     |  (1<<10);   // Цифровой   PB10
216   MDR_PORTB->PULL  &= ~(1<<8);    // Отключена  PB8  (Подтяжка) DOWN - подтяжка линии к потенциалу питания (3.3 В)
217   MDR_PORTB->PULL  &= ~(1<<24);   // Отключена  PB8  (Подтяжка) UP  - подтяжка линии к потенциалу земли   (0 В)
218   MDR_PORTB->PULL  &= ~(1<<9);    // Отключена  PB9  DOWN
219   MDR_PORTB->PULL  &= ~(1<<25);   // Отключена  PB9  UP
220   MDR_PORTB->PULL  &= ~(1<<10);   // Отключена  PB10 DOWN
221   MDR_PORTB->PULL  &= ~(1<<26);   // Отключена  PB10 UP
222   MDR_PORTB->PD    &= ~(1<<8);    // Драйвер    PB8  (Управление выводом)
223   MDR_PORTB->PD    &= ~(1<<24);   // Т.Г. Dis.  PB8  Триггер Шмитта(ТГ) - Отключен (Dis.)
224   MDR_PORTB->PD    &= ~(1<<9);    // Драйвер    PB9
225   MDR_PORTB->PD    &= ~(1<<25);   // Т.Г. Dis.  PB9
226   MDR_PORTB->PD    &= ~(1<<10);   // Драйвер    PB10
227   MDR_PORTB->PD    &= ~(1<<26);   // Т.Г. Dis.  PB10
228   MDR_PORTB->PWR   |= (3<<(2*8))  // Высокая    PB8  (Крутизна фронтов)
229                     |  (3<<(2*9))  // Высокая    PB9
230                     |  (3<<(2*10));// Высокая    PB10
231   MDR_PORTB->GFEN  &= ~(1<<8);    // Отключен   PB8  (Цифровой фильтр)
232   MDR_PORTB->GFEN  &= ~(1<<9);    // Отключен   PB9
233   MDR_PORTB->GFEN  &= ~(1<<10);   // Отключен   PB10
234   //------------------------------------------------------------------
235
```

```
237   //-------------- PORTB ----------------------------------------
238   MDR_PORTB->OE    |= (1<<6);     // Вывод     PB6  (Направление передачи)
239   MDR_PORTB->FUNC  &= ~(3<<(2*6)); // Ввод-вывод PB6  (Функция)
240   MDR_PORTB->ANALOG |= (1<<6);    // Цифровой   PB6  (Режим работы)
241   MDR_PORTB->PULL  &= ~(1<<6);    // Отключена  PB6  (Подтяжка) DOWN - подтяжка линии к потенциалу питания (3.3 В)
242   MDR_PORTB->PULL  &= ~(1<<22);   // Отключена  PB6  (Подтяжка) UP  - подтяжка линии к потенциалу земли   (0 В)
243   MDR_PORTB->PD    &= ~(1<<6);    // Драйвер    PB6  (Управление выводом)
244   MDR_PORTB->PD    &= ~(1<<22);   // Т.Г. Dis.  PB6  Триггер Шмитта(ТГ) - Отключен (Dis.)
245   MDR_PORTB->PWR   |= (3<<(2*6)); // Высокая    PB6  (Крутизна фронтов)
246   MDR_PORTB->GFEN  &= ~(1<<6);    // Отключен   PB6  (Цифровой фильтр)
247   //------------------------------------------------------------------
```

Keil: Straj.h

```
  //--------------PORTC-------------------------------------------
  // Конфигурация линий PC0..PC2
  MDR_PORTC->OE    |= (1<<0)      // Вывод     PC0  (Направление передачи)
                   |  (1<<1)      // Вывод     PC1
                   |  (1<<2);     // Вывод     PC2
  MDR_PORTC->FUNC  &= ~(3<<(2*0)); // Ввод-вывод PC0  (Функция)
  MDR_PORTC->FUNC  &= ~(3<<(2*1)); // Ввод-вывод PC1
  MDR_PORTC->FUNC  &= ~(3<<(2*2)); // Ввод-вывод PC2
  MDR_PORTC->ANALOG |= (1<<0)     // Цифровой  PC0  (Режим работы)
                    |  (1<<1)     // Цифровой  PC1
                    |  (1<<2);    // Цифровой  PC2
  MDR_PORTC->PULL  &= ~(1<<0);    // Отключена PC0  (Подтяжка) DOWN - подтяжка линии к потенциалу питания (3.3 В)
  MDR_PORTC->PULL  &= ~(1<<16);   // Отключена PC0  (Подтяжка) UP   - подтяжка линии к потенциалу земли   (0 В)
  MDR_PORTC->PULL  &= ~(1<<1);    // Отключена PC1  DOWN
  MDR_PORTC->PULL  &= ~(1<<17);   // Отключена PC1  UP
  MDR_PORTC->PULL  &= ~(1<<2);    // Отключена PC2  DOWN
  MDR_PORTC->PULL  &= ~(1<<18);   // Отключена PC2  UP
  MDR_PORTC->PD    &= ~(1<<0);    // Драйвер   PC0  (Управление выводом)
  MDR_PORTC->PD    &= ~(1<<16);   // Т.Г.- Dis. PC0  Триггер Шмитта(ТГ) - Отключен (Dis.)
  MDR_PORTC->PD    &= ~(1<<1);    // Драйвер   PC1
  MDR_PORTC->PD    &= ~(1<<17);   // Т.Г.- Dis. PC1
  MDR_PORTC->PD    &= ~(1<<2);    // Драйвер   PC2
  MDR_PORTC->PD    &= ~(1<<18);   // Т.Г.- Dis. PC2
  MDR_PORTC->PWR   |= (3<<(2*0))  // Высокая   PC0  (Крутизна фронтов)
                   |  (3<<(2*1))  // Высокая   PC1
                   |  (3<<(2*2)); // Высокая   PC2
  MDR_PORTC->GFEN  &= ~(1<<0);    // Отключен  PC0  (Цифровой фильтр)
  MDR_PORTC->GFEN  &= ~(1<<1);    // Отключен  PC1
  MDR_PORTC->GFEN  &= ~(1<<2);    // Отключен  PC2
  //-------------------------------------------------------------
```

```

  //--------------PORTF-------------------------------------------
  MDR_PORTF->OE    |= (1<<4);     // Вывод     PF4  (Направление передачи)
  MDR_PORTF->FUNC  &= ~(3<<(2*4)); // Ввод-вывод PF4  (Функция)
  MDR_PORTF->ANALOG |= (1<<4);    // Цифровой  PF4  (Режим работы)
  MDR_PORTF->PULL  &= ~(1<<4);    // Отключена PF4  (Подтяжка) DOWN - подтяжка линии к потенциалу питания (3.3 В)
  MDR_PORTF->PULL  &= ~(1<<20);   // Отключена PF4  (Подтяжка) UP   - подтяжка линии к потенциалу земли   (0 В)
  MDR_PORTF->PD    &= ~(1<<4);    // Драйвер   PF4  (Управление выводом)
  MDR_PORTF->PD    &= ~(1<<20);   // Т.Г.- Dis. PF4  Триггер Шмитта(ТГ) - Отключен (Dis.)
  MDR_PORTF->PWR   |= (3<<(2*4)); // Высокая   PF4  (Крутизна фронтов)
  MDR_PORTF->GFEN  &= ~(1<<4);    // Отключен  PF4  (Цифровой фильтр)
  //-------------------------------------------------------------
  /*-------------------------------------------------------------------------------*/

  /*--------------вкл./выкл. мультиплексора-------------------------------------*/
  // Конфигурация линии PD2
  MDR_PORTD->OE    |= (1<<2);     // Вывод     PD2  (Направление передачи)
  MDR_PORTD->FUNC  &= ~(3<<(2*2)); // Ввод-вывод PD2  (Функция)
  MDR_PORTD->ANALOG |= (1<<2);    // Цифровой  PD2  (Режим работы)
  MDR_PORTD->PULL  &= ~(1<<2);    // Отключена PD2  (Подтяжка) DOWN - подтяжка линии к потенциалу питания (3.3 В)
  MDR_PORTD->PULL  &= ~(1<<18);   // Отключена PD2  (Подтяжка) UP   - подтяжка линии к потенциалу земли   (0 В)
  MDR_PORTD->PD    &= ~(1<<2);    // Драйвер   PD2  (Управление выводом)
  MDR_PORTD->PD    &= ~(1<<18);   // Т.Г.- Dis. PD2  Триггер Шмитта(ТГ) - Отключен (Dis.)
  MDR_PORTD->PWR   |= (3<<(2*2)); // Высокая   PD2  (Крутизна фронтов)
  MDR_PORTD->GFEN  &= ~(1<<2);    // Отключен  PD2  (Цифровой фильтр)
  /*-------------------------------------------------------------------------------*/
}
```

Keil: Init.h

```
void Init_per()
{
  LED_Init();                    // Инициализация светодиодов

  MDR_RST_CLK->PER_CLOCK = 0xffffffff; // вкл. тактир. всей периферии МК

  MDR_PORTE->OE    = 0xff37; // биты 7,6,3 PORTE - входы, др. - выходы
  MDR_PORTE->FUNC  = 0x0000; // функция - порт, основная функция
  MDR_PORTE->PWR   = 0xff37; // макс. быстрый фронт
  MDR_PORTE->ANALOG = 0xffff; // режим работы порта - цифровой ввод/вывод

  delay_init();                  // инициализация системы задержек

  LCD_init();                    // инициализация дисплея
  LCD_clear();                   // очистка дисплея
  MDR_PORTE->RXTX |= (1<<2); // вкл. подсветки

  Init_Straj();                  // Инициализация "Страж"
  MCU_ADC1_init();               // Инициализация АЦП

  uart_init();                  // инициализация модуля UART
  MDR_UART1->IMSC = 1 << 4;   // инициализация прерывания от UART
  NVIC_EnableIRQ(UART1_IRQn);// разрешение прерывания от модуля UART1

  //-----------------Таймер1 измер. напряжения ------------------------------
  Timer1_init();                    // инициализация Таймера 1
  NVIC_EnableIRQ(Timer1_IRQn);      // разрешение прерывания от Таймера 1
  __enable_irq();                   // глобальное разрешение прерываний
  Timer1_start();                   // запуск Таймера 1
  NVIC_SetPriority (Timer1_IRQn, 3);  // Установка приоритета для таймера
  //-------------------------------------------------------------------------
}
```

Keil: ADC.h

```c
void MCU_ADC1_init()
{
  MDR_RST_CLK->PER_CLOCK |= RST_CLK_PCLK_ADC_Msk;          // вкл. тактирование АЦП
  // настройка конфигурации АЦП
  MDR_ADC->ADC1_CFG = ( 1 << ADC1_CFG_REG_ADON_Pos)        // Работа АЦП                    (включен)
                    |(0 << ADC1_CFG_REG_CLKS_Pos)          // Источник тактирования АЦП     (CPU)
                    |(0 << ADC1_CFG_REG_SAMPLE_Pos)        // Способ запуска АЦП            (однократный)
                    |(0 << ADC1_CFG_REG_CHS_Pos)           // Целевой канал преобразователя (уст. во время работы программы)
                    |(0 << ADC1_CFG_REG_CHCH_Pos)          // Режим последовательного переключения каналов (отключён)
                    |(0 << ADC1_CFG_REG_RNGC_Pos)          // Контроль границ преобразования (отключён)
                    |(0 << ADC1_CFG_M_REF_Pos)             // Источник опорного напряжения  (внутренний)
                    |(3 << ADC1_CFG_REG_DIVCLK_Pos)        // Делитель тактовой частоты АЦП (2^3 = 8)
                    |(0 << ADC1_CFG_SYNC_CONVER_Pos)       // Режим запуска двух АЦП (независимый)
                   //...Конфигурация датчика температуры и внутреннего источника напряжения 1.23 В
                    | (0 << ADC1_CFG_TS_EN_Pos)            // Работа блока датчика температуры и внутреннего источника напряжения 1.23 В (отключён)
                    | (0 << ADC1_CFG_TS_BUF_EN_Pos)        // Работа усилителя для датчика температуры и внутреннего источника напряжения 1.23 В (отключён)
                    | (0 << ADC1_CFG_SEL_TS_Pos)           // Преобразование сигнала с датчика температуры (отключено)
                    | (0 << ADC1_CFG_SEL_VREF_Pos)         // Преобразование сигнала с внутреннего источника напряжения 1.23 В (отключено)
                    | (0 << ADC1_CFG_TR_Pos)               // Подстройка напряжения внутреннего источника 1.23 В
                   //...Настройка задержек при преобразовании
                   //|(7 << ADC1_CFG_DELAY_GO_Pos)         // Дополнительная задержка при выборе канала (8 тактов ядра)
                    |(0 << ADC1_CFG_DELAY_ADC_Pos);        // Разность фаз между циклами преобразователей (не используется)

                   // ADC3 MX_OUT - напряжение на мультиплексоре
                   MDR_PORTD->OE    &=~(1 << 3); // настройка PD3 на вход
                   MDR_PORTD->ANALOG &=~(1 << 3); // перевод PD3 в аналоговый режим

                   // ADC4 ШУНТ - измерение напряжения главной платы
                   MDR_PORTD->OE    &=~(1 << 5); // настройка PD5 на вход
                   MDR_PORTD->ANALOG &=~(1 << 5); // перевод PD5 в аналоговый режим

                   // ADC5 12v/4 - общ. напряжение
                   MDR_PORTD->OE    &=~(1 << 5); // настройка PD5 на вход
                   MDR_PORTD->ANALOG &=~(1 << 5); // перевод PD5 в аналоговый режим
}
```

```c
/**
  * @brief     Установка канала АЦП
  * @detailed  3 канал - измерять напряжение на мультиплексоре
  *            4 канал - напряжение главной платы
  *            5 канал - измерять общ напряжение
  * @param     @channel - номер канала АЦП
  * @return    Функция не возвращает параметры
  */
void MCU_ADC_set_ch(uint8_t channel)
{
  // проверка номера канала и возврат в случае выхода из диапазона
  if (channel > 15) return;              // всего 16 каналов (с 0)
  MDR_ADC->ADC1_CFG  |= channel << 4;    // уст. канала АЦП
}

void MCU_ADC_start_conv(void)    // начало преобразования
{
  MDR_ADC->ADC1_CFG |= 1 << ADC1_CFG_REG_GO_Pos;  // Запись "1" начинает процесс преобразования
}
```

```c
/**
  * @brief     Чтение результатов преобразования АЦП
  * @detailed
  * @param     Не принимает параметров
  * @return    @ADC_data - результат преобразования
  */
uint32_t MCU_ADC_read(void)
{
  uint32_t ADC_data = 0;  // локальная для хранения результата преобр.

  MCU_ADC_start_conv();  // команда начала преобразования
  // пустой цикл - ожидание окончания преобразования
  while(!(MDR_ADC->ADC1_STATUS) & (1<<2)) { }
  ADC_data = MDR_ADC->ADC1_RESULT; // чтение результата преобразований
  // очистка битов содержащих номер канала преобразования - обнуление
  // старшего полубайта регистра
  ADC_data  =  ADC_data & 0x0FFF;
  return ADC_data; // возврат результата преобразования
}
```

Arduino

```
#define  F_CPU 8000000                  // Указание тактовой частоты МК
#include  "milkites_delay.h"            // Подключение библиотеки задержек
#include  "milkites_display.h"          // Подключение библиотеки дисплея
#define STANDS  12                      // кол-во стэндов
#define TIM1_INTERRUPT 5000             // Период опроса АЦП (мс.)
#define GET_CURRENT(x) ((int32_t) ((((float)x*0.0008) / (0.1*200)) * 1000))
#define DELAY_UART 5                    // задержка между отправкой байта в UART

volatile uint32_t Count_current=0;  // счетчик снятия показаний
volatile uint32_t Current[STANDS];  // токопотребление стендов
volatile uint32_t Current_general;  // значение общего тока

//------------------ Timer 1 -----------------------------------------------
void Timer1_init(void)
{ // настройка T1 на генерирование прерывания каждум секунду
  MDR_RST_CLK->TIM_CLOCK |= (1<< 24); // вкл. тактирование Таймера 1

  // режим счета - вверх,начальное значение - число из регистра CNT
  MDR_TIMER1->CNTRL = 0x00000000;
  MDR_TIMER1->PSG = 7999;  // предделитель частоты
  MDR_TIMER1->ARR = TIM1_INTERRUPT-1;  // основание счета = CNT + 1 = TIM1_INTERRUPT
  MDR_TIMER1->CNT = 0;     // начальное значение счетчика
  MDR_TIMER1->IE  = 2;     // разрешение генерир. прерывание при CNT=ARR
}
```

Arduino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>
#define STANDS 12                               // кол-во стендов

// UART
SoftwareSerial softSerial(21,22);               // объявление задействованных дискретных каналов      RX, TX
#define BYTE_UART 3                             // кол-во байт UART
uint32_t TX_Arr[BYTE_UART];                     // массив принятых байт
uint8_t CountArr=0;                             // кол-во приинятых байт
uint32_t Current_St[STANDS];                    // токопотребление на стендах

#define LED 2                                   // определяем пин светодиода
#define MQTT_client "ah76kkji"                  // произвольное название MQTT клиента, иногда требуется уникальное.

// настройки домашней сети
const char *ssid          = "xxxxx";            // название точки доступа
const char *pass          = "xxxx";    // пароль от точки доступа

// настройки для MQTT брокера
const char *mqtt_server   = "xxxx";     // адрес сервера MQTT
const int   mqtt_port     = xxx;             // порт для подключения к серверу MQTT  TLS: 3011
const char *mqtt_user     = "xxxx";       // логин от сервера MQTT
const char *mqtt_pass     = "xxxx";       // пароль от сервера MQTT

const char *led_topic     = "test/led";       // топик для светодиода
const char *data_topic    = "test/data";      // топик для данных


    const char *topic_gen_current = "current/g";  // топик публикации общ. значения тока
    // Топики для вкл./выкл. Стэндов
    const String stand[STANDS]=
    {
      "stand/1",  // Стэнд 1
      "stand/2",  // Стэнд 2
      "stand/3",  // Стэнд 3
      "stand/4",  // Стэнд 4
      "stand/5",  // Стэнд 5
      "stand/6",  // Стэнд 6
      "stand/7",  // Стэнд 7
      "stand/8",  // Стэнд 8
      "stand/9",  // Стэнд 9
      "stand/10", // Стэнд 10
      "stand/11", // Стэнд 11
      "stand/12"  // Стэнд 12
    };


      // Топики для публикации токопотребления
      const String topic_current[STANDS]=
      {
        "current/1",  // Ток стенда 1
        "current/2",  // Ток стенда 2
        "current/3",  // Ток стенда 3
        "current/4",  // Ток стенда 4
        "current/5",  // Ток стенда 5
        "current/6",  // Ток стенда 6
        "current/7",  // Ток стенда 7
        "current/8",  // Ток стенда 8
        "current/9",  // Ток стенда 9
        "current/10", // Ток стенда 10
        "current/11", // Ток стенда 11
        "current/12"  // Ток стенда 12
      };

      int pause = 300;                              // переменная для паузы между отправками данных
      long int times=0;                             // для времени

      WiFiClient wclient;
      PubSubClient client(wclient, mqtt_server, mqtt_port);
```

Arduino

```
// получение данных с сервера и отработка
void callback(const MQTT::Publish& pub)
{
    String topic = pub.topic();
    String payload = pub.payload_string();  // чтение данных из топика
    // действия над светодиодом в зависимости от данных из топика

    // Проверяем топики Стендов
    for (uint8_t i=0;i<STANDS;i++)
      if (topic==stand[i])
      {
        // Формируем байт для отправки команды
        uint8_t com = i+1;          // Номер стенда
        com*=10;                    // Смещаем номер слево
        if (payload[0]=='1')        // Состояние стенда   '1' - вкл.
          com+=1;

        Serial.write(com);          // Отправляем команду

        break;                      // Выход из цикла
      }
}
```

```
void data_UART()
{
  while(Serial.available())
  {
    digitalWrite(LED,LOW);                      // вкл. светодиод (пришли данные по UART)
    if (CountArr<BYTE_UART)                     // если еще не пришло нужное кол-во байт
        TX_Arr[CountArr++]=Serial.read();       // считывание байта
    else                                        // если набрали нужное кол-во
    {
      uint32_t data=0;                          // для формирования числа принятого по UART
      for (uint8_t i=0;i<BYTE_UART-1;i++)       // цикл, проходящий по всем эл. массива (кроме последнего)
        data|=(TX_Arr[i]<<(i*8));               // их полученных байт формируем число
      CountArr=0;                               // обнуление счетчика


      if ((char)TX_Arr[BYTE_UART-1]=='g')       // если последний байт G - общ. значение тока
        client.publish(topic_gen_current,String(data));// публикации общ. токопотребления
      else
        client.publish(topic_current[TX_Arr[BYTE_UART-1]],String(data));// ток стенд    (последний принятый байт номер стенда с 0)
      digitalWrite(LED,HIGH);                   // выкл. светодиод (данные считаны и обработаны)
    }
  }
}
```

Arduino

```
// Функция отправки показаний
void refreshData() {
  if (pause == 0) {
    times = millis(); // формируем данные для отправки
    client.publish(data_topic, String(times));
    pause = 3000; // пауза меду отправками 3 секунды
  }
  pause--;

  delay(1);
}
//-------------------------------------

void setup() {
    softSerial.begin(115200);       // Инициализация программного последовательного порта
    Serial.begin(115200);           // Инициализация порта

    pinMode(LED, OUTPUT);           // пин светика на выход
    digitalWrite(LED,HIGH);         // состояние светодиода выкл.

    // изначально токпотребление на всех стендах равно 0
    for (uint8_t i=0;i<STANDS;i++)
      Current_St[i]=0;
}


void subscription_topic() // ф-я подписки на топики
{

  for (uint8_t i=0;i<STANDS;i++)                // Подписка на топики Стэндов
    client.subscribe(stand[i]);

  for (uint8_t i=0;i<STANDS;i++)                // Подпись на топик публикации Покопотребления
    client.subscribe(topic_current[i]);

  client.subscribe(topic_gen_current);          // подписка на топик публикации общ. токопотребления

  client.subscribe(led_topic);                  // подписка на топик led
  client.subscribe(data_topic);                 // подписка на топик data
}


 void loop() {

    if (WiFi.status() != WL_CONNECTED) { // если соединения нет
        WiFi.begin(ssid, pass);// подключаемся к wi-fi
        if (WiFi.waitForConnectResult() != WL_CONNECTED) // ждем окончания подключения
            return;
    }

    // подключаемся к MQTT серверу
    if (WiFi.status() == WL_CONNECTED) { // если есть подключение к wi-fi
        if (!client.connected()) { // если нет подключения к серверу MQTTsetServer
           // Serial.println("MQTT - none");
            if (client.connect(MQTT::Connect(MQTT_client) // если соединились то делаем всякое
                               .set_auth(mqtt_user, mqtt_pass))) {
               // Serial.println("MQTT - ok");
                client.set_callback(callback);
                subscription_topic();             // подписка на топики
            }
            /*else { Вывод на дисплей
                Serial.println("MQTT - error");   // если не удалось подключиться сообщаем в порт
            }
            */
        }

        if (client.connected()){ // если есть соединение с MQTT
            client.loop();
            if (Serial.available())
                data_UART();
            //refreshData();
```