Запрос от клиента-оператора выполняется в формате HTTP и, поскольку несёт в себе достаточно большое количество данных, которые требует определённой конфиденциальности, имеет тип POST. Поскольку в рамках лекции стоит задача показать принципиальную схему организации системы, то для быстрого формирования запросов прямо в строке браузера будет использоваться запрос типа GET. Например, такой запрос может иметь форму:

Исходный код 1. Содержание GET запроса

|  |
| --- |
| 192.168.250/?operation=get&value=expander-efficiency&target=expander&id=1 |

где: operation – указатель на тип выполняемой операции (get/set);

value – указатель на соответствующие точки цикла;

target – указатель на тип элемент цикла;

id – идентификатор конкретного элемента цикла типа target.

Исходный код 2. Модифицированный файл index.php

|  |
| --- |
| <?php  // включение отображения всех ошибок  **error\_reporting** (**E\_ALL**);  **ini\_set** ('display\_errors', **true**);  // подключение библиотеки CoolProp  **require** "CoolProp.php";  // получение данных из GET запроса  $operation = $\_GET['operation'] ?? '';  $value = $\_GET['value'] ?? '';  $target = $\_GET['target'] ?? '';  $id = $\_GET['id'] ?? '';  // упрощенная логика валидации  **if**(  $operation === ''  OR $value === ''  OR $target === ''  OR $id === '')  {  **echo** "Некорректный запрос";  } **else** {  // упрощенная логика обработки запроса  **if**($operation === 'get'){  **if**($value === 'expander-efficiency'){  **if**($target === 'expander'){  **if**($id === '1'){  // формирование запроса на сервер микроконтроллера  $request = **array**(  "operation" => $operation,  "value" => $value,  "target" => $target,  "id" => $id  );  // представление в формате JSON  $requestJSON = **json\_encode**($request);  // получение данных с микроконтроллера блока ожижения  $getResponse = **file\_get\_contents**('http://192.168.1.251/?'  .$requestJSON."!");  $response = **json\_decode**($getResponse, **JSON\_UNESCAPED\_UNICODE**);  // // температура в точке 2  $T2 = $response["T2"];  // // температура в точке 4  $T4 = $response["T4"];  // // давление в точке 2  $p2 = $response["p2"];  // // давление в точке 4  $p4 = $response["p4"];  // расчет термодинамических параметров  $fluid = "HYDROGEN";  $h2 = **round**(  PropsSI("H", "T", $T2, "P", $p2 \* 100000, $fluid)  / 1000, 2);  $h4 = **round**(  PropsSI("H", "T", $T4, "P", $p4 \* 100000, $fluid)  / 1000, 2);  $s2 = **round**(  PropsSI("S", "T", $T2, "P", $p2 \* 100000, $fluid)  , 2);  $s2s = $s2;  $h2s = **round**(  PropsSI("H", "S", $s2s, "P", $p4 \* 100000, $fluid)  / 1000, 2);  $eff = **round**(($h2 - $h4)/($h2 - $h2s) \* 100, 2);  **echo**("Температура на входе: $T2 K<br>");  **echo**("Давление на входе: $p2 бар<br>");  echo("Температура на выходе: $T4 K<br>");  echo("Давление на выходе: $p4 бар<br>");  **echo**("Изоэнтропный КПД детандера: $id:  η<sub>s</sub>=(h2 - h4)/(h2 - h2s)  =($h2 - $h4)/($h2 - $h2s)  = $eff %");  die;  }  }  }  }  echo "Некорректный запрос";  } |

Программный код прошивки микроконтроллера имеет следующий вид:

Исходный код 3. Прошивка микроконтроллера

|  |
| --- |
| #include <SPI.h>  #include <Ethernet.h>  #include <ArduinoJson.h>  #include <Regexp.h>  //Сервер  byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED // DE:AD:BE:EF:FE:ED  };  EthernetServer server(80);  **void** setup() {  //Ожидание прослушивания com-порта  Serial.begin(9600);  **while** (!Serial) {;}  //Инициализация сервера  Ethernet.begin(mac);  server.begin();  Serial.print("Server start at ");  Serial.println(Ethernet.localIP());  }  **void** loop() {  EthernetClient client = server.available();  **if** (client) {  // Выделение памяти для получения Get параметров  boolean getReqFirst = true;  boolean getReqStart = false;  boolean getReqEnd = false;  **String** getData = "";  boolean currentLineIsBlank = true;  // Выделение памяти для работы с JSON и регулярными выражениями  StaticJsonDocument<200> data;  MatchState ms;  Serial.println(">>> Новый запрос");  // Получение Get запроса  **while** (client.connected()) {  **if** (client.available()) {  // текущий принятый символ запроса  **char** c = client.read();  // останавливается прием get параметров  // если встретился "!"  **if** (c == '!' && getReqFirst) {  getReqFirst = false;  getReqEnd = true;  Serial.println(">>> Get end");  }  // идет прием get параметров  // если идет первый get запрос  **if** (getReqStart && getReqFirst) {  getData += c;  }  // запускается прием get параметров  // если это первая подстрока, начинающаяся с "?"  **if** (c == '?' && getReqFirst) {  getReqStart = true;  Serial.println(">>> Get start");  }  **if** (c == '\n' && currentLineIsBlank) {  **if** (getReqEnd) {  client.println("HTTP/1.1 200 OK");  client.println("Content-Type: application/json");  client.println("Connection: close");  client.println();  // Выделение памяти для ответа  StaticJsonDocument<300> doc;  // Подготовка JSON  getData.replace("%22","\"");  DeserializationError error =  deserializeJson(doc, getData.c\_str());  // Проверка JSON на наличие ошибок  **if** (error) {  Serial.print(F("Ошибка инициализации JSON: "));  Serial.println(error.f\_str());  **return**;  }  **const** **char**\* operation = doc["operation"];  Serial.print("Сервер отправил команду: ");  Serial.println(operation);    **const** **char**\* value = doc["value"];  Serial.print("Объект обработки: ");  Serial.println(value);    **const** **char**\* target = doc["target"];  Serial.print("Элемент цикла: ");  Serial.println(target);    **const** **char**\* id = doc["id"];  Serial.print("Id элементв цикла: ");  Serial.println(id);  // Формирование ответа  data["status"] = "OK";  data["T2"] = random(65, 72);  data["T4"] = **String**(random(21, 23))  + "." + **String**(random(4, 9));  data["p2"] = random(100, 105);  data["p4"] = "1."+ **String**(random(20, 35));  // Отправка ответа  serializeJsonPretty(data, client);    } **else** {  client.println("HTTP/1.1 200 OK");  client.println("Content-Type: application/json");  client.println("Connection: close");  client.println();  // Формирование ответа  data["status"] = "ERROR";  data["eror"] = "Некорректный запрос";  serializeJsonPretty(data, client);  }  **break**;  }  // Указатели на завершение запроса Get  **if** (c == '\n')  {  currentLineIsBlank = true;  }  **else** **if** (c != '\r')  {  currentLineIsBlank = false;  }  }  }  }  delay(1);  client.stop();  } |