Студент **Михайлов Фёдор Михайлович**

(ФГБОУ ВО СПбГАУ)

Канд. техн. наук **Юрий Валентинович Иванов**

(ФГБОУ ВО СПБГАУ)

**Система многоканального автополива для умных теплиц на основе ардуино**

Многоканальная система автополива растений для установки в умную теплицу. Особенности:

* Поддержка от 1 до 15 помп (Arduino NANO / UNO)
* Индивидуальная настройка периода и времени работы
* Дисплей 1602 с отображением настроек
* Индивидуальное название каждого канала (можно по-русски!)
* Удобное управление и настройка энкодером
* Хранение настроек в энергонезависимой памяти
* Настройка уровня управляющего сигнала
* Настройка часы/минуты/секунды работы
* Параллельный режим работы / очередь



Рисунок 1 – Система многоканального полива

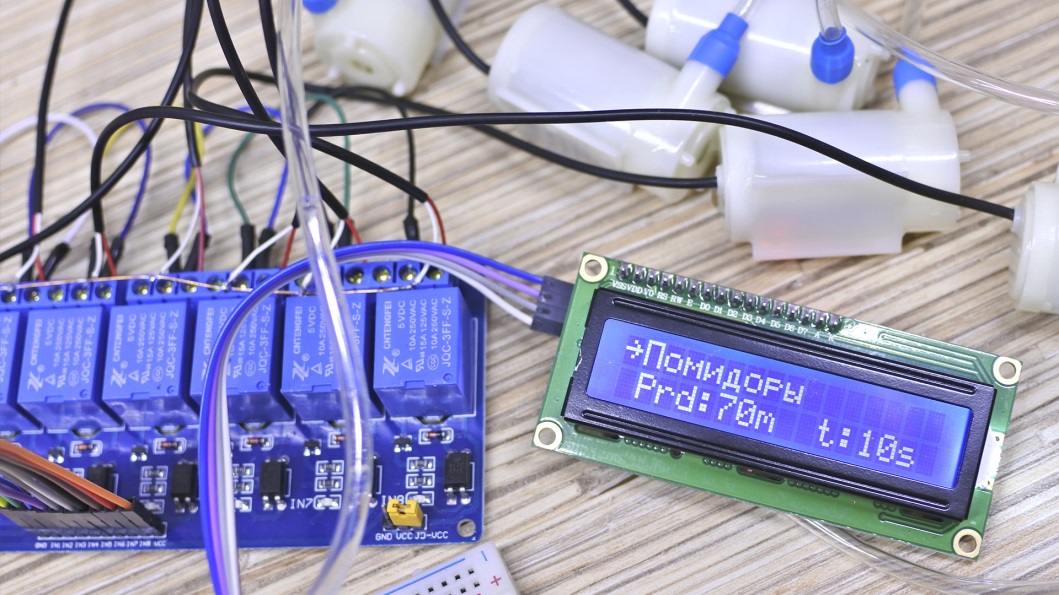


Рисунок 2 – Основные элементы многоканального полива

**Инструменты и материалы:**

* Arduino NANO 328p
* Помпа 12V
* Дисплей 1602
* Энкодер
* Реле электромагнитное и твердотельное
* Макетная плата и провода
* Блок питания 12V
* Разветвитель потока с краниками
* Шланг силиконовый
* Тройничок

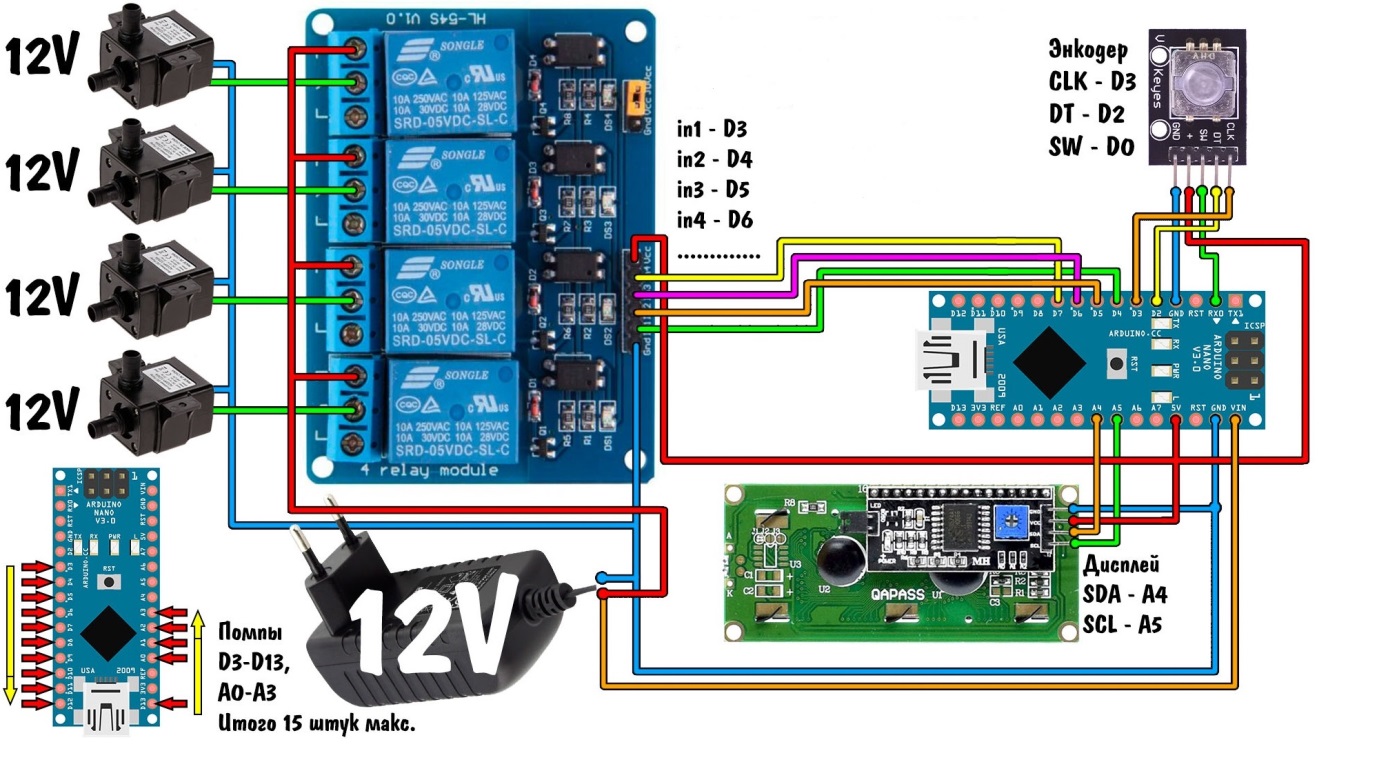


Рисунок 3 – Схема многоканального автополива

**Программа:**

static const wchar\_t \*relayNames[] = {

L"Картошка",

L"Морковка",

L"Лук",

L"Перец",

L"Огурцы",

L"Помидоры",

L"Клубника",

L"Петрушка",

};

#define CLK 3

#define DT 2

#define SW 0

#include "GyverEncoder.h"

Encoder enc1(CLK, DT, SW);

#include <EEPROMex.h>

#include <EEPROMVar.h>

#include "LCD\_1602\_RUS.h"

// -------- АВТОВЫБОР ОПРЕДЕЛЕНИЯ ДИСПЛЕЯ-------------

// Если кончается на 4Т - это 0х27. Если на 4АТ - 0х3f

#if (DRIVER\_VERSION)

LCD\_1602\_RUS lcd(0x27, 16, 2);

#else

LCD\_1602\_RUS lcd(0x3f, 16, 2);

#endif

// -------- АВТОВЫБОР ОПРЕДЕЛЕНИЯ ДИСПЛЕЯ-------------

uint32\_t pump\_timers[PUPM\_AMOUNT];

uint32\_t pumping\_time[PUPM\_AMOUNT];

uint32\_t period\_time[PUPM\_AMOUNT];

boolean pump\_state[PUPM\_AMOUNT];

byte pump\_pins[PUPM\_AMOUNT];

int8\_t current\_set;

int8\_t current\_pump;

boolean now\_pumping;

int8\_t thisH, thisM, thisS;

long thisPeriod;

boolean startFlag = true;

void setup() {

// --------------------- КОНФИГУРИРУЕМ ПИНЫ ---------------------

pinMode(PUMP\_PIN, OUTPUT);

digitalWrite(PUMP\_PIN, !SWITCH\_LEVEL); // выключаем

for (byte i = 0; i < PUPM\_AMOUNT; i++) { // пробегаем по всем помпам

pump\_pins[i] = START\_PIN + i; // настраиваем массив пинов

pinMode(START\_PIN + i, OUTPUT); // настраиваем пины

digitalWrite(START\_PIN + i, !SWITCH\_LEVEL); // выключаем

}

// --------------------- ИНИЦИАЛИЗИРУЕМ ЖЕЛЕЗО ---------------------

//Serial.begin(9600);

lcd.init();

lcd.backlight();

lcd.clear();

//enc1.setStepNorm(1);

//attachInterrupt(0, encISR, CHANGE);

enc1.setType(ENCODER\_TYPE);

if (ENC\_REVERSE) enc1.setDirection(REVERSE);

// --------------------- СБРОС НАСТРОЕК ---------------------

if (!digitalRead(SW)) { // если нажат энкодер, сбросить настройки до 1

lcd.setCursor(0, 0);

lcd.print("Reset settings");

for (byte i = 0; i < 500; i++) {

EEPROM.writeLong(i, 0);

}

}

while (!digitalRead(SW)); // ждём отпускания кнопки

lcd.clear(); // очищаем дисплей, продолжаем работу

// --------------------------- НАСТРОЙКИ ---------------------------

// в ячейке 1023 должен быть записан флажок, если его нет - делаем (ПЕРВЫЙ ЗАПУСК)

if (EEPROM.read(1023) != 5) {

EEPROM.writeByte(1023, 5);

// для порядку сделаем 1 ячейки с 0 по 500

for (byte i = 0; i < 500; i += 4) {

EEPROM.writeLong(i, 0);

}

}

for (byte i = 0; i < PUPM\_AMOUNT; i++) { // пробегаем по всем помпам

period\_time[i] = EEPROM.readLong(8 \* i); //. На чётных - период (ч)

pumping\_time[i] = EEPROM.readLong(8 \* i + 4); // на нечётных - полив (с)

if (SWITCH\_LEVEL) // вырубить все помпы

pump\_state[i] = 0;

else

pump\_state[i] = 1;

}

// ---------------------- ВЫВОД НА ДИСПЛЕЙ ------------------------

drawLabels();

changeSet();

}

void loop() {

encoderTick();

periodTick();

flowTick();

}

void periodTick() {

for (byte i = 0; i < PUPM\_AMOUNT; i++) { // пробегаем по всем помпам

if (startFlag ||

(period\_time[i] > 0

&& millis() - pump\_timers[i] >= period\_time[i] \* 1000

&& (pump\_state[i] != SWITCH\_LEVEL)

&& !(now\_pumping \* !PARALLEL))) {

pump\_state[i] = SWITCH\_LEVEL;

digitalWrite(pump\_pins[i], SWITCH\_LEVEL); // открыть КЛАПАН

pump\_timers[i] = millis();

now\_pumping = true;

digitalWrite(PUMP\_PIN, SWITCH\_LEVEL); // включить общую ПОМПУ

//Serial.println("Pump #" + String(i) + " ON");

}

}

startFlag = false;

}

void flowTick() {

for (byte i = 0; i < PUPM\_AMOUNT; i++) { // пробегаем по всем помпам

if (pumping\_time[i] > 0

&& millis() - pump\_timers[i] >= pumping\_time[i] \* 1000

&& (pump\_state[i] == SWITCH\_LEVEL) ) {

pump\_state[i] = !SWITCH\_LEVEL;

digitalWrite(pump\_pins[i], !SWITCH\_LEVEL); // закрыть КЛАПАН

if (TIMER\_START) pump\_timers[i] = millis();

now\_pumping = false;

digitalWrite(PUMP\_PIN, !SWITCH\_LEVEL); // выключить общую ПОМПУ

//Serial.println("Pump #" + String(i) + " OFF");

}

}

}

/\*

void encISR() {

enc1.tick(); // отработка энкодера

}

\*/

void encoderTick() {

enc1.tick(); // отработка энкодера

if (enc1.isTurn()) { // если был совершён поворот

if (enc1.isRight()) {

if (++current\_set >= 7) current\_set = 6;

} else if (enc1.isLeft()) {

if (--current\_set < 0) current\_set = 0;

}

if (enc1.isRightH())

changeSettings(1);

else if (enc1.isLeftH())

changeSettings(-1);

changeSet();

}

}

// тут меняем номер помпы и настройки

void changeSettings(int increment) {

if (current\_set == 0) {

current\_pump += increment;

if (current\_pump > PUPM\_AMOUNT - 1) current\_pump = PUPM\_AMOUNT - 1;

if (current\_pump < 0) current\_pump = 0;

s\_to\_hms(period\_time[current\_pump]);

drawLabels();

} else {

if (current\_set == 1 || current\_set == 4) {

thisH += increment;

} else if (current\_set == 2 || current\_set == 5) {

thisM += increment;

} else if (current\_set == 3 || current\_set == 6) {

thisS += increment;

}

if (thisS > 59) {

thisS = 0;

thisM++;

if (thisM > 59) {

thisM = 0;

thisH++;

}

}

if (thisS < 0) {

thisS = 59;

thisM--;

if (thisM < 0) {

thisM = 59;

thisH--;

if (thisH < 0) thisH = 0;

}

}

if (current\_set < 4) period\_time[current\_pump] = hms\_to\_s();

else pumping\_time[current\_pump] = hms\_to\_s();

}

}

// вывести название реле

void drawLabels() {

lcd.setCursor(1, 0);

lcd.print(" ");

lcd.setCursor(1, 0);

lcd.print(relayNames[current\_pump]);

}

// изменение позиции стрелки и вывод данных

void changeSet() {

switch (current\_set) {

case 0: drawArrow(0, 0); update\_EEPROM();

break;

case 1: drawArrow(7, 1);

break;

case 2: drawArrow(10, 1);

break;

case 3: drawArrow(13, 1);

break;

case 4: drawArrow(7, 1);

break;

case 5: drawArrow(10, 1);

break;

case 6: drawArrow(13, 1);

break;

}

lcd.setCursor(0, 1);

if (current\_set < 4) {

lcd.print(L"ПАУЗА ");

s\_to\_hms(period\_time[current\_pump]);

}

else {

lcd.print(L"РАБОТА");

s\_to\_hms(pumping\_time[current\_pump]);

}

lcd.setCursor(8, 1);

if (thisH < 10) lcd.print(0);

lcd.print(thisH);

lcd.setCursor(11, 1);

if (thisM < 10) lcd.print(0);

lcd.print(thisM);

lcd.setCursor(14, 1);

if (thisS < 10) lcd.print(0);

lcd.print(thisS);

}

// перевод секунд в ЧЧ:ММ:СС

void s\_to\_hms(uint32\_t period) {

thisH = floor((long)period / 3600); // секунды в часы

thisM = floor((period - (long)thisH \* 3600) / 60);

thisS = period - (long)thisH \* 3600 - thisM \* 60;

}

// перевод ЧЧ:ММ:СС в секунды

uint32\_t hms\_to\_s() {

return ((long)thisH \* 3600 + thisM \* 60 + thisS);

}

// отрисовка стрелки и двоеточий

void drawArrow(byte col, byte row) {

lcd.setCursor(0, 0); lcd.print(" ");

lcd.setCursor(7, 1); lcd.print(" ");

lcd.setCursor(10, 1); lcd.print(":");

lcd.setCursor(13, 1); lcd.print(":");

lcd.setCursor(col, row); lcd.write(126);

}

// обновляем данные в памяти

void update\_EEPROM() {

EEPROM.updateLong(8 \* current\_pump, period\_time[current\_pump]);

EEPROM.updateLong(8 \* current\_pump + 4, pumping\_time[current\_pump]);

}