////////////////////////////////////////////////////////////////////////////////////

//////// ////////

//// ////

// БИБЛИОТЕКИ //

//// ////

//////// ////////

////////////////////////////////////////////////////////////////////////////////////

// БИБЛИОТЕКИ ЭНКОДЕРОВ //

//// ////

#include <ACE128.h>

#include <ACE128map87654321.h> // Карта пинов 87654321

////////////////////////////////////////////////////////////////////////////////////

//////// ////////

////

////

// ПЕРЕМЕННЫЕ //

//// ////

//////// ////////

////////////////////////////////////////////////////////////////////////////////////

//// ////

// ПЕРЕМЕННЫЕ КНОПОК //

//// ////

#define BUTTDELAY 50 // Время между двумя изменениями

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО КНОПОК!

#define BUTTNUM 7 // Количество кнопок

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО КНОПОК!

const byte buttPin[BUTTNUM] = {22, 24, 26, 36, 38, 28, 30}; // Пины кнопок

boolean buttPos[BUTTNUM] = {false, false, false, false, false, false, false}; // Положение кнопки

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО КНОПОК!

boolean buttRev[BUTTNUM] = {false, false, false, false, false, false, false}; // Реверс кнопки

unsigned long buttTime[BUTTNUM] = {0, 0, 0, 0, 0, 0, 0}; // Момент времени изменения положения кнопки

//// ////

// ПЕРЕМЕННЫЕ ДВУХПОЗИЦИОННЫХ ТУМБЛЕРОВ //

//// ////

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ТУМБЛЕРОВ!

#define TOGGNUM 1

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ТУМБЛЕРОВ!

const byte toggIn1[] = {34}; // Пины тумблеров поз.1

bool toggLog1[] = {false}; // Переменная для определения изменений в положении тумблера

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ТУМБЛЕРОВ!

const byte toggIn2[] = {32}; // Пины тумблеров поз.2

bool toggLog2[] = {false}; // Переменная для определения изменений в положении тумблера

//// ////

// ПЕРЕМЕННЫЕ ПРЕДОХРАНИТЕЛЕЙ //

//// ////

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ПРЕДОХРАНИТЕЛЕЙ!

#define FUSENUM 2

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ПРЕДОХРАНИТЕЛЕЙ!

const byte fuseIn[] = {40, 42}; // Пины предохранителей

bool fuseLog[] = {false, false}; // Переменная для определения изменений в положении предохранителей

//// ////

// ПЕРЕМЕННЫЕ ПОТЕНЦИОМЕТРОВ ШР - new //

//// ////

//#define razSetup

#define razERROR 20 // Шум в значении потенциометра

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ПОТЕНЦИОМЕТРОВ!

#define razNUM 6

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ПОТЕНЦИОМЕТРОВ!

const byte razPin[razNUM] = {0, 1, 2, 5, 3, 4}; // Пины потенциометров {0, 1, 2}

int razPos[razNUM] = {0, 0, 0, 0, 0, 0}; // Положение потенциометра

int oldrazPos[razNUM] = {0, 0, 0, 0, 0, 0}; // Предыдущее положение потенциометра

int razType[razNUM] = {0, 0, 0, 0, 0, 0}; // Тип потенциометра

int razBorder[razNUM][2] = {{0, 1000}, {0, 1000}, {0, 1000}, {0, 1000}, {0, 1000}, {0, 1000}};

int razAnswer[razNUM] = {1000, 1000, 1000, 1000, 1000, 1000};

#define razERRORAnswer 15

// Границы значений потенциометра

double truerazPos[razNUM] = {0, 0, 0, 0, 0, 0}; // Положение потенциометра по шкале

double oldTruerazPos[razNUM] = {0, 0, 0, 0, 0, 0}; // Старое положение потенциометра по шкале

int truerazAnswer[razNUM] = {0, 0, 0, 0, 0, 0}; // Положение потенциометра по шкале

int oldTruerazAnswer[razNUM] = {2, 2, 2, 2, 2, 2}; // Старое положение потенциометра по шкале

unsigned long razTime[razNUM] = {0, 0, 0, 0, 0, 0}; // Момент времени изменения положения кнопки

#define razDELAY 1500 // Время между двумя изменениями

#define razDELAYsetup 20 // Время между двумя изменениями

unsigned long traz = 0;

//// ////

// ПЕРЕМЕННЫЕ РАЗЪЕМОВ ШР //

//// ////

#define CONNDELAY0 1000 // Время между двумя изменениями

#define CONNDELAY1 4000 // Время между двумя изменениями (для закручивающихся)

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО РАЗЪЕМОВ!

#define CONNNUM 5 // Количество разъемов

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО РАЗЪЕМОВ!

int connType[CONNNUM] = {1, 1, 1, 1, 1}; // Тип разъема

int connDelay[CONNNUM] = {0, 0, 0, 0, 0}; // Время между двумя изменениями

const byte connPin[CONNNUM] = {18, 17, 14, 15, 16}; // Пины разъемов

boolean connPos[CONNNUM] = {false, false, false, false, false}; // Положение разъемов

unsigned long connTime[CONNNUM] = {0, 0, 0, 0, 0}; // Момент времени изменения положения разъемов

///LED//tut

#include <FastLED.h>

#define LED\_NUM 32 // Количество светодиодов

#define LED\_PIN 13 // Пин светодиодов

boolean ledPos[LED\_NUM]; // Положение /\* определено в setup(){} \*/

int i2, n2;

CRGB led[LED\_NUM]; // Определение

//// ////

// ПЕРЕМЕННЫЕ ЭНКОДЕРОВ //

//// ////

//#define ENCSETUP // Раскомментировать для настройки

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ЭНКОДЕРОВ!

#define ENCNUM 4 // Количество энкодеров

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ЭНКОДЕРОВ!

const byte encoderAddress[ENCNUM]= {32, 33, 34, 35}; // Адреса энкодеров

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ЭНКОДЕРОВ!

ACE128 enc0((uint8\_t)encoderAddress[0], (uint8\_t\*)encoderMap\_87654321); // Адрес и карта энкодера

ACE128 enc1((uint8\_t)encoderAddress[1], (uint8\_t\*)encoderMap\_87654321);

ACE128 enc2((uint8\_t)encoderAddress[2], (uint8\_t\*)encoderMap\_87654321);

ACE128 enc3((uint8\_t)encoderAddress[3], (uint8\_t\*)encoderMap\_87654321);

uint8\_t rawPos[ENCNUM] = {0, 0, 0, 0}; // Текущее положение энкодера

uint8\_t oldRawPos[ENCNUM] = {127, 127, 127, 127}; // Предыдущее положение энкодера

#ifndef ENCSETUP

int encType[ENCNUM] = {0, 0, 0, 0}; // Тип энкодера

int rawBorder[ENCNUM][2] = {{64, 1}, {116, 63}, {40, 126}, {21, 125}}; // Границы положений энкодера

int encBorder[ENCNUM][2] = {{0, 6}, {0, 7}, {0, 16}, {0, 20}}; // Границы положений энкодера по шкале

double encPoint[ENCNUM] = {0, 0, 0, 0}; //

double encPos[ENCNUM] = {0, 0, 0, 0}; // Положение энкодера по шкале

double oldEncPos[ENCNUM] = {0, 0, 0, 0}; // Предыдущее положение энкодера по шкале

double rawNorm[ENCNUM] = {0, 0, 0, 0};

#endif

//// ////

// ОБЩИЕ ПЕРЕМЕННЫЕ //

//// ////

//#define DEBUG // Раскомментировать для отладки

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ЭЛЕМЕНТОВ!

#define ELEMNUM 58

String str = "";

int i;

int n = 0;

unsigned long t = 0;

boolean checkElem = false;

void setup() {

////////////////////////////////////////////////////////////////////////////////////

//////// ////////

//// ////

// НАСТРОЙКИ //

//// ////

//////// ////////

////////////////////////////////////////////////////////////////////////////////////

Serial.begin(115200); // Инициализация порта для работы с ПК

#ifdef DEBUG

//Serial.println("DEBUG DEFINED");

#endif

Serial.println("kpa35");

Serial.println("complite");

//// ////

// НАСТРОЙКИ КНОПОК //

//// ////

for(i = 0; i < BUTTNUM; i++){

pinMode(buttPin[i], INPUT); // Настройка пина на вход

digitalWrite(buttPin[i], HIGH); // Подтяжка пина к питанию

}

for(i = 0; i < BUTTNUM; i++){ // Опрос кнопок

if(buttRev[i] == false){

if(digitalRead(buttPin[i]) == HIGH){

Serial.print("b");

Serial.print(i);

Serial.print(" off");

Serial.println("z");

}

else if(digitalRead(buttPin[i]) == LOW){

Serial.print("b");

Serial.print(i);

Serial.print(" on");

Serial.println("z");

buttPos[i] = true;

}

}

else{

if(digitalRead(buttPin[i]) == HIGH){

Serial.print("b");

Serial.print(i);

Serial.print(" on");

Serial.println("z");

buttPos[i] = true;

}

else if(digitalRead(buttPin[i]) == LOW){

Serial.print("b");

Serial.print(i);

Serial.print(" off");

Serial.println("z");

}

}

n++;

}

//// ////

// НАСТРОЙКИ ДВУХПОЗИЦИОННЫХ ТУМБЛЕРОВ //

//// ////

for(int i = 0; i < TOGGNUM; i++){

pinMode(toggIn1[i], INPUT); // Настройка пина на вход

digitalWrite(toggIn1[i], HIGH); // Подтяжка пина к питанию

pinMode(toggIn2[i], INPUT);

digitalWrite(toggIn2[i], HIGH);

}

for(int i = 0; i < TOGGNUM; i++){ // Опрос тумблеров

if(digitalRead(toggIn1[i]) == HIGH && digitalRead(toggIn2[i]) == HIGH){

Serial.print("t");

Serial.print(i);

Serial.print(" off");

Serial.println("z");

}

else if(digitalRead(toggIn1[i]) == LOW){

Serial.print("t");

Serial.print(i);

Serial.print(" H");

Serial.println("z");

toggLog1[i] = true;

}

else if(digitalRead(toggIn2[i]) == LOW){

Serial.print("t");

Serial.print(i);

Serial.print(" L");

Serial.println("z");

toggLog2[i] = true;

}

}

//// ////

// НАСТРОЙКИ ПРЕДОХРАНИТЕЛЕЙ //

//// ////

for (int i = 0; i < FUSENUM; i++) {

pinMode(fuseIn[i], INPUT); // Настройка пина на вход

digitalWrite(fuseIn[i], HIGH); // Подтяжка пина к питанию

}

for (int i = 0; i < FUSENUM; i++) { // Опрос предохранителей

if (digitalRead(fuseIn[i]) == HIGH) {

Serial.print("f");

Serial.print(i);

Serial.print(" off");

Serial.println("z");

}

else if (digitalRead(fuseIn[i]) == LOW) {

Serial.print("f");

Serial.print(i);

Serial.print(" on");

Serial.println("z");

fuseLog[i] = true;

}

}

/////////////////////////НАСТРОЙКИ РАЗЪЕМОВ ПОТ/////////////////////////////////////////////

for (i = 0; i < razNUM; i++) { // Опрос потенциометров

razPos[i] = analogRead(razPin[i]);

oldrazPos[i] = razPos[i];

truerazPos[i] = razPos[i];

if (truerazPos[i] > razBorder[i][1]) {

truerazPos[i] = razBorder[i][1];

}

else if (truerazPos[i] < razBorder[i][0]) {

truerazPos[i] = razBorder[i][0];

}

truerazPos[i] = (truerazPos[i] - razBorder[i][0]) / (razBorder[i][1] - razBorder[i][0]) \* 1000;

truerazPos[i] = (int)round(truerazPos[i]) / razERROR \* razERROR;

oldTruerazPos[i] = truerazPos[i];

#ifndef razSetup

if (razAnswer[i] >= truerazPos[i] - razERRORAnswer && razAnswer[i] <= truerazPos[i] + razERRORAnswer)

{

//if (razAnswer[i] == truerazPos[i]) {

Serial.print("c");

Serial.print(i);

Serial.print(" ");

Serial.println("offz");

truerazAnswer[i] = 1;

} else {

Serial.print("c");

Serial.print(i);

Serial.print(" ");

Serial.println("onz");

}

oldTruerazAnswer[i] = truerazAnswer[i];

#else

Serial.print("c");

Serial.print(i);

Serial.print(" ");

Serial.println(truerazPos[i]);

#endif

}

//// ////

// НАСТРОЙКИ РАЗЪЕМОВ //

//// ////

for(i = 0; i < CONNNUM; i++){

pinMode(connPin[i], INPUT); // Настройка пина на вход

digitalWrite(connPin[i], HIGH); // Подтяжка пина к питанию

if(connType[i] == 0){

connDelay[i] = CONNDELAY0;

}

else if(connType[i] == 1){

connDelay[i] = CONNDELAY1;

}

}

for(i = 0; i < CONNNUM; i++){ // Опрос разъемов

if(digitalRead(connPin[i]) == HIGH){

Serial.print("c");

Serial.print(i + 6);

Serial.print(" off");

Serial.println("z");

}

else if(digitalRead(connPin[i]) == LOW){

Serial.print("c");

Serial.print(i + 6);

Serial.print(" on");

Serial.println("z");

connPos[i] = true;

}

n++;

}

//// ////

// НАСТРОЙКИ СВЕТОДИОДОВ tyt //

//// ////

FastLED.addLeds<WS2811, LED\_PIN, RGB>(led, LED\_NUM); // Определение пина и количества

FastLED.setBrightness(255); // Задание яркости

for (i2 = 0; i2 < LED\_NUM; i2++) { // Выключение светодиодов

led[i2] = CRGB(0, 0, 0);

ledPos[i2] = false;

n2++;

FastLED.show();

}

//// ////

// НАСТРОЙКА ЭНКОДЕРОВ //

//// ////

#ifdef ENCODERSETUP

Serial.println("ENCODERSETUP DEFINED");

#endif

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ЭНКОДЕРОВ!

enc0.begin(); // Инициализация работы энкодера

enc1.begin();

enc2.begin();

enc3.begin();

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ЭНКОДЕРОВ!

rawPos[0] = enc0.rawPos(); // Опрос энкодера

rawPos[1] = enc1.rawPos();

rawPos[2] = enc2.rawPos();

rawPos[3] = enc3.rawPos();

#ifdef ENCSETUP

for(i = 0; i < ENCNUM; i++){ // Вывод текущего положения энкодера

oldRawPos[i] = rawPos[i];

Serial.print("e");

Serial.print(i);

Serial.print(" ");

Serial.println(rawPos[i]);

}

#endif

#ifndef ENCSETUP

for(i = 0; i < ENCNUM; i++){ // Вывод в порт ПК текущего состояния энкодера

oldRawPos[i] = rawPos[i];

if(rawPos[i] == 255){

#ifdef DEBUG

Serial.print("e");

Serial.print(i);

Serial.print(" ");

Serial.println("ERROR");

#endif

encPos[i] = encBorder[i][0];

}

else{

if (rawBorder[i][0] < rawBorder[i][1]){

rawNorm[i] = rawBorder[i][1] - rawBorder[i][0];

encPoint[i] = (rawBorder[i][0] + 128 - rawBorder[i][1] - 1) / 2.0;

if(rawBorder[i][0] < encPoint[i]){

encPoint[i] += rawBorder[i][1];

}

else{

encPoint[i] -= rawBorder[i][0];

}

if(encPoint[i] > 0){

if((rawPos[i] >= rawBorder[i][0]) && (rawPos[i] <= rawBorder[i][1])){

encPos[i] = (rawPos[i] - rawBorder[i][0]) / rawNorm[i];

}

else if((rawPos[i] < encPoint[i]) && (rawPos[i] >= rawBorder[i][1])){

encPos[i] = 1.00;

}

else{

encPos[i] = 0.00;

}

}

else{

if((rawPos[i] >= rawBorder[i][0]) && (rawPos[i] <= rawBorder[i][1])){

encPos[i] = (rawPos[i] - rawBorder[i][0]) / rawNorm[i];

}

else if((rawPos[i] > -encPoint[i]) && (rawPos[i] <= rawBorder[i][0])){

encPos[i] = 0.00;

}

else{

encPos[i] = 1.00;

}

}

}

else{

rawNorm[i] = 128 - rawBorder[i][0] + rawBorder[i][1] + 1;

encPoint[i] = (rawBorder[i][0] - rawBorder[i][1] - 1) / 2.0;

encPoint[i] += rawBorder[i][1];

if(rawPos[i] >= rawBorder[i][0]){

encPos[i] = (rawPos[i] - rawBorder[i][0]) / rawNorm[i];

}

else if(rawPos[i] <= rawBorder[i][1]){

encPos[i] = (rawPos[i] + 128 - rawBorder[i][0]) / rawNorm[i];

}

else if(rawPos[i] < encPoint[i]){

encPos[i] = 1.00;

}

else{

encPos[i] = 0.00;

}

}

if(encType[i] == 0){

encPos[i] = round(encPos[i] \* (encBorder[i][1] - encBorder[i][0]));

}

}

oldEncPos[i] = encPos[i];

Serial.print("e");

Serial.print(i);

Serial.print(" ");

Serial.print(encPos[i]);

Serial.println("z");

n++;

}

#endif

} // КОНЕЦ НАСТРОЕК

void loop() {

////////////////////////////////////////////////////////////////////////////////////

//////// ////////

//// ////

// РАБОТА //

//// ////

//////// ////////

////////////////////////////////////////////////////////////////////////////////////

if(checkElem == false)// Проверка окончания всех настроек

/\*if (n == ELEMNUM)\*/{

//Serial.print("SETUP DONE");

Serial.println("stop");

checkElem = true;

}

//// ////

// РАБОТА КНОПОК //

//// ////

for(int i = 0; i < BUTTNUM; i++){ // Опрос кнопок

if(buttRev[i] == false){

if((digitalRead(buttPin[i]) == HIGH) && (buttPos[i] == true)){

t = millis();

if((t - buttTime[i]) > BUTTDELAY){

Serial.print("b");

Serial.print(i);

Serial.print(" off");

Serial.println("z");

buttTime[i] = t;

buttPos[i] = false;

}

}

else if((digitalRead(buttPin[i]) == LOW) && (buttPos[i] == false)){

t = millis();

if((t - buttTime[i]) > BUTTDELAY){

Serial.print("b");

Serial.print(i);

Serial.print(" on");

Serial.println("z");

buttTime[i] = t;

buttPos[i] = true;

}

}

}

else{

if((digitalRead(buttPin[i]) == HIGH) && (buttPos[i] == false)){

t = millis();

if((t - buttTime[i]) > BUTTDELAY){

Serial.print("b");

Serial.print(i);

Serial.print(" on");

Serial.println("z");

buttTime[i] = t;

buttPos[i] = true;

}

}

else if((digitalRead(buttPin[i]) == LOW) && (buttPos[i] == true)){

t = millis();

if((t - buttTime[i]) > BUTTDELAY){

Serial.print("b");

Serial.print(i);

Serial.print(" off");

Serial.println("z");

buttTime[i] = t;

buttPos[i] = false;

}

}

}

}

//// ////

// РАБОТА ДВУХПОЗИЦИОННЫХ ТУМБЛЕРОВ //

//// ////

for(int i = 0; i < TOGGNUM; i++){ // Опрос тумблеров

if(digitalRead(toggIn1[i]) == LOW && toggLog1[i] == false){

delay(50);

Serial.print("t");

Serial.print(i);

Serial.print(" H");

Serial.println("z");

toggLog1[i] = true;

}

else if(digitalRead(toggIn2[i]) == LOW && toggLog2[i] == false){

delay(50);

Serial.print("t");

Serial.print(i);

Serial.print(" L");

Serial.println("z");

toggLog2[i] = true;

}

else if((digitalRead(toggIn1[i]) == HIGH && toggLog1[i] == true) || (digitalRead(toggIn2[i]) == HIGH && toggLog2[i] == true)){

delay(50);

Serial.print("t");

Serial.print(i);

Serial.print(" off");

Serial.println("z");

toggLog1[i] = false;

toggLog2[i] = false;

}

}

//// ////

// РАБОТА ПРЕДОХРАНИТЕЛЕЙ //

//// ////

for (int i = 0; i < FUSENUM; i++) { // Опрос предохранителей

if (digitalRead(fuseIn[i]) == LOW && fuseLog[i] == false) {

delay(50);

Serial.print("f");

Serial.print(i);

Serial.print(" on");

Serial.println("z");

fuseLog[i] = true;

}

else if (digitalRead(fuseIn[i]) == HIGH && fuseLog[i] == true) {

delay(50);

Serial.print("f");

Serial.print(i);

Serial.print(" off");

Serial.println("z");

fuseLog[i] = false;

}

}

///////////////////////////////РАБОТА РАЗЪЕМОВ ПОТ/////////////////////////////////

for (i = 0; i < razNUM; i++) { // Опрос потенциометров

traz = millis();

if ((traz - razTime[i]) > razDELAY) {

razPos[i] = analogRead(razPin[i]);

if (abs(razPos[i] - oldrazPos[i]) > razERROR) {

oldrazPos[i] = razPos[i];

truerazPos[i] = razPos[i];

if (truerazPos[i] > razBorder[i][1]) {

truerazPos[i] = razBorder[i][1];

}

else if (truerazPos[i] < razBorder[i][0]) {

truerazPos[i] = razBorder[i][0];

}

truerazPos[i] = (truerazPos[i] - razBorder[i][0]) / (razBorder[i][1] - razBorder[i][0]) \* 1000;

truerazPos[i] = (int)round(truerazPos[i]) / razERROR \* razERROR;

if (razType[i] == 0) {

if (oldTruerazPos[i] != truerazPos[i]) {

oldTruerazPos[i] = truerazPos[i];

//if (razAnswer[i] == truerazPos[i]) {

#ifndef razSetup

if (razAnswer[i] >= truerazPos[i] - razERRORAnswer && razAnswer[i] <= truerazPos[i] + razERRORAnswer)

{

truerazAnswer[i] = 1;

} else {

truerazAnswer[i] = 0;

}

if (truerazAnswer[i] != oldTruerazAnswer[i]) {

Serial.print("c");

Serial.print(i);

Serial.print(" ");

if (truerazAnswer[i] == 0) {

Serial.println("onz");

} else {

Serial.println("offz");

}

oldTruerazAnswer[i] = truerazAnswer[i];

}

#else

Serial.print("c");

Serial.print(i);

Serial.print(" ");

Serial.println(truerazPos[i]);

#endif

}

}

}

razTime[i] = traz;

}

}

//// ////

// РАБОТА РАЗЪЕМОВ ШР //

//// ////

for(int i = 0; i < CONNNUM; i++){ // Опрос разъемов

if((digitalRead(connPin[i]) == HIGH) && (connPos[i] == true)){

t = millis();

if((t - connTime[i]) > connDelay[i]){

Serial.print("c");

Serial.print(i + 6);

Serial.print(" off");

Serial.println("z");

connTime[i] = t;

connPos[i] = false;

}

}

else if((digitalRead(connPin[i]) == LOW) && (connPos[i] == false)){

t = millis();

if((t - connTime[i]) > connDelay[i]){

Serial.print("c");

Serial.print(i + 6);

Serial.print(" on");

Serial.println("z");

connTime[i] = t;

connPos[i] = true;

}

}

}

//// ////

// РАБОТА ЭНКОДЕРОВ //

//// ////

// ИЗМЕНИТЬ, ЕСЛИ ИЗМЕНИЛОСЬ КОЛИЧЕСТВО ЭНКОДЕРОВ!

rawPos[0] = enc0.rawPos(); // Опрос энкодера

rawPos[1] = enc1.rawPos();

rawPos[2] = enc2.rawPos();

rawPos[3] = enc3.rawPos();

#ifdef ENCSETUP

for(i = 0; i < ENCNUM; i++){

if (rawPos[i] != oldRawPos[i]){ // Вывод текущего положения энкодера при наличии изменений

oldRawPos[i] = rawPos[i];

Serial.print("e");

Serial.print(i);

Serial.print(" ");

Serial.println(rawPos[i]);

}

}

#endif

#ifndef ENCSETUP

for(i = 0; i < ENCNUM; i++){ // Вывод в порт ПК текущего состояния энкодера

if (rawPos[i] != oldRawPos[i]){

oldRawPos[i] = rawPos[i];

if(rawPos[i] == 255){

#ifdef DEBUG

Serial.print("e");

Serial.print(i);

Serial.print(" ");

Serial.println("ERROR");

#endif

}

else{

if (rawBorder[i][0] < rawBorder[i][1]){

if(encPoint[i] > 0){

if((rawPos[i] >= rawBorder[i][0]) && (rawPos[i] <= rawBorder[i][1])){

encPos[i] = (rawPos[i] - rawBorder[i][0]) / rawNorm[i];

}

else if((rawPos[i] < encPoint[i]) && (rawPos[i] >= rawBorder[i][1])){

encPos[i] = 1.00;

}

else{

encPos[i] = 0.00;

}

}

else{

if((rawPos[i] >= rawBorder[i][0]) && (rawPos[i] <= rawBorder[i][1])){

encPos[i] = (rawPos[i] - rawBorder[i][0]) / rawNorm[i];

}

else if((rawPos[i] > -encPoint[i]) && (rawPos[i] <= rawBorder[i][0])){

encPos[i] = 0.00;

}

else{

encPos[i] = 1.00;

}

}

}

else{

if(rawPos[i] >= rawBorder[i][0]){

encPos[i] = (rawPos[i] - rawBorder[i][0]) / rawNorm[i];

}

else if(rawPos[i] <= rawBorder[i][1]){

encPos[i] = (rawPos[i] + 128 - rawBorder[i][0]) / rawNorm[i];

}

else if(rawPos[i] < encPoint[i]){

encPos[i] = 1.00;

}

else{

encPos[i] = 0.00;

}

}

if(encType[i] == 0){

encPos[i] = round(encPos[i] \* (encBorder[i][1] - encBorder[i][0]));

}

if(encPos[i] != oldEncPos[i]){

oldEncPos[i] = encPos[i];

Serial.print("e");

Serial.print(i);

Serial.print(" ");

Serial.print(encPos[i]);

Serial.println("z");

}

}

}

}

#endif

////////////////////////////////////////////////////////////////////////////////////

//////// ////////

//// ////

// СВЯЗЬ С ПК //

//// ////

//////// ////////

////////////////////////////////////////////////////////////////////////////////////

if (Serial.available() > 0){ // Проверка, что есть данные от ПК

str = Serial.readStringUntil('\n');

#ifdef DEBUG

/\*Serial.print("USER: ");

Serial.println(str); \*/

#endif

//// ////

// СВЯЗЬ С ПК ДЛЯ КНОПОК //

//// ////

if (str.substring(0, 4).equals("butt")){// Принудительный опрос кнопки

i = str.substring(4, 6).toInt();

if(i < BUTTNUM){

if(buttPos[i] == false){

/\*Serial.print("b");

Serial.print(i);

Serial.print(" off");

Serial.println("z"); \*/

}

else if(buttPos[i] == true){

/\* Serial.print("b");

Serial.print(i);

Serial.print(" on");

Serial.println("z");\*/

}

}

}

//// ////

// СВЯЗЬ С ПК ДЛЯ ДВУХПОЗИЦИОННЫХ ТУМБЛЕРОВ //

//// ////

else if (str.substring(0, 4).equals("togg")){ // Принудительный опрос тумблеров

i = str.substring(4, 7).toInt();

if(digitalRead(toggIn1[i]) == HIGH && digitalRead(toggIn2[i]) == HIGH){

/\* Serial.print("t");

Serial.print(i);

Serial.println(" off"); \*/

}

else if(digitalRead(toggIn1[i]) == LOW){

/\* Serial.print("t");

Serial.print(i);

Serial.println(" H"); \*/

}

else if(digitalRead(toggIn2[i]) == LOW){

/\* Serial.print("t");

Serial.print(i);

Serial.println(" L"); \*/

}

}

//// ////

// СВЯЗЬ С ПК ДЛЯ ПРЕДОХРАНИТЕЛЕЙ //

//// ////

else if (str.substring(0, 4).equals("fuse")) { // Принудительный опрос предохранителя

i = str.substring(4, 5).toInt();

if (digitalRead(fuseIn[i]) == HIGH) {

/\* Serial.print("f");

Serial.print(i);

Serial.println(" off"); \*/

}

else if (digitalRead(fuseIn[i]) == LOW) {

/\* Serial.print("f");

Serial.print(i);

Serial.println(" on");\*/

}

}

//// ////

// СВЯЗЬ С ПК ДЛЯ РАЗЪЕМОВ //

//// ////

if (str.substring(0, 4).equals("conn")){// Принудительный опрос разъема

i = str.substring(4, 6).toInt();

if(i < CONNNUM){

if(connPos[i] == false){

/\* Serial.print("c");

Serial.print(i);

Serial.print(" off");

Serial.println("z"); \*/

}

else if(connPos[i] == true){

/\* Serial.print("c");

Serial.print(i);

Serial.print(" on");

Serial.println("z");\*/

}

}

}

//// ////

// СВЯЗЬ С ПК ДЛЯ СВЕТОДИОДОВ //

//// ////

else if (str.substring(0, 2).equals("on")) { // Включить светодиод

i = str.substring(2, 4).toInt();

if (i < LED\_NUM) {

led[i] = CRGB(0, 255, 0);

ledPos[i] = true;

FastLED.show();

}

}

else if (str.substring(0, 3).equals("off")) { // Выключить светодиод

i = str.substring(3, 5).toInt();

if (i < LED\_NUM) {

led[i] = CRGB(0, 0, 0);

ledPos[i] = false;

FastLED.show();

}

}

//// ////

// СВЯЗЬ С ПК ДЛЯ ЭНКОДЕРОВ //

//// ////

if(str.substring(0, 3).equals("enc")){ // Запрос состояния энкодера от ПК и вывод в порт ПК

i = str.substring(3, 4).toInt();

if(i < ENCNUM){

Serial.print("e");

Serial.print(i);

Serial.print(" ");

#ifdef ENCSETUP

Serial.println(rawPos[i]);

#endif

#ifndef ENCSETUP

Serial.print(encPos[i]);

Serial.println("z");

#endif

}

}

} // КОНЕЦ СВЯЗИ С ПК

} // КОНЕЦ РАБОТЫ