**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №5

з дисципліни "Верифікація цифрових систем"

на тему: "**РОБОТА З МІКРОКОНТРОЛЕРОМ ЗА ДОПОМОГОЮ ВІДДАЛЕНОЇ ЛАБОРАТОРІЇ GOLDI**"

Варіант №12

Виконав:

студент групи КНТ-227 С. М. Медведєв

Прийняв:

к.т.н., доцент Т. І. Каплієнко

2019

**Мета роботи:** ознайомитись з повним процесом використання, створення програми для мікроконтролерів за допомогою віддаленої лабораторії GOLDI та програми AtmelStudio.

**План**

1. Створити програму для виконання базових дій інструментом ProductionCell;
2. Завантажити її до віддаленої лабораторії;
3. Переконатися у її працездатності.

**Хід роботи**

**Лістинг файлу UserDesign.h**

// ######################################################################################

// # #

// # This module implements the users design #

// # #

// ######################################################################################

#ifndef \_USERDESIGN\_H

#define \_USERDESIGN\_H

#include "../Main.h"

extern void StateMachineInit(void); // This function initializes the state machine

extern void StateMachineUpdate(void); // This function updated the state machine

// ######################################################################################

// # Add a new state for state machine here #

// ######################################################################################

typedef enum

{

Init, RT2\_init, RT2\_rotate\_l, RT2\_download, RT2\_rotate\_r, RT2\_upload, RT2\_to\_wait,

TT\_init, TT\_move\_r, TT\_download, TT\_move\_l, TT\_upload, TT\_to\_wait,

RT1\_init, RT1\_rotate\_l, RT1\_download, RT1\_rotate\_r, RT1\_upload, RT1\_to\_wait,

MM\_init, MM\_move\_f, MM\_down, MM\_work, MM\_work\_time, MM\_up, MM\_move\_b

} AutomatStates\_t;

#endif

**Лістинг файлу UserDesign.c**

// ######################################################################################

// # #

// # This module implements the users design #

// # #

// ######################################################################################

#include <util/delay.h>

#include "UserDesign.h"

AutomatStates\_t State;

// ######################################################################################

// # This function initializes the finite state machine with start state #

// ######################################################################################

void StateMachineInit(void)

{

State = Init;

}

void SetY\_0(){

Actuators.TransportTableMoveToConveyorBelt3 = 0; //y0

Actuators.TransportTableMoveToConveyorBelt1 = 0; //y1

Actuators.TransportTableDriveConveyorBeltSimilarToConveyorBelt1 = 0; //y2

Actuators.TransportTableDriveConveyorBeltSimilarToConveyorBelt3 = 0; //y3

Actuators.ConveyorBelt1DriveBelt = 0; //y4

Actuators.Turntable1RotateToConveyorBelt1 = 0; //y5

Actuators.Turntable1RotateToConveyorBelt2 = 0; //y6

Actuators.Turntable1DriveBelt = 0; //y7

Actuators.ConveyorBelt2DriveBelt = 0; //y8

Actuators.Turntable2RotateToConveyorBelt2 = 0; //y9

Actuators.Turntable2RotateToConveyorBelt\_3 = 0; //y10

Actuators.Turntable2DriveBelt = 0; //y11

Actuators.ConveyorBelt3DriveBelt = 0; //y12

Actuators.MillingMachineApproachConveyorBelt\_2 = 0; //y13

Actuators.MillingMachineRetreatFromConveyorBelt2 = 0; //y14

Actuators.MillingHeadRise = 0; //y15

Actuators.MillingHeadLower = 0; //y16

Actuators.MillingHeadDriveHead = 0; //y17

}

void SetY(int z){

if (z == 0) Actuators.TransportTableMoveToConveyorBelt3 = 1; //y0

if (z == 1) Actuators.TransportTableMoveToConveyorBelt1 = 1; //y1

if (z == 2) Actuators.TransportTableDriveConveyorBeltSimilarToConveyorBelt1 = 1;

//y2

if (z == 3) Actuators.TransportTableDriveConveyorBeltSimilarToConveyorBelt3 = 1;

//y3

if (z == 4) Actuators.ConveyorBelt1DriveBelt = 1; //y4

if (z == 5) Actuators.Turntable1RotateToConveyorBelt1 = 1; //y5

if (z == 6) Actuators.Turntable1RotateToConveyorBelt2 = 1; //y6

if (z == 7) Actuators.Turntable1DriveBelt = 1; //y7

if (z == 8) Actuators.ConveyorBelt2DriveBelt = 1; //y8

if (z == 9) Actuators.Turntable2RotateToConveyorBelt2 = 1; //y9

if (z == 10) Actuators.Turntable2RotateToConveyorBelt\_3 = 1; //y10

if (z == 11) Actuators.Turntable2DriveBelt = 1; //y11

if (z == 12) Actuators.ConveyorBelt3DriveBelt = 1; //y12

if (z == 13) Actuators.MillingMachineApproachConveyorBelt\_2 = 1; //y13

if (z == 14) Actuators.MillingMachineRetreatFromConveyorBelt2 = 1; //y14

if (z == 15) Actuators.MillingHeadRise = 1; //y15

if (z == 16) Actuators.MillingHeadLower = 1; //y16

if (z == 17) Actuators.MillingHeadDriveHead = 1; //y17

}

void delay(int t){

int g = 1;

for (int k = 0; k < t; k++)

{

for (int i = 0; i < 14000; i++)

{

for (int j = 0; j < 1000; j++)

{

g \*= -1;

}

}

}

}

// ######################################################################################

// # This function updates the current state of the finite state machine #

// ######################################################################################

void StateMachineUpdate(void)

{

SetY\_0();

int s[] = {

Sensors.TransportTableInLineWithConveyorBelt3, //x0

Sensors.TransportTableInLineWithWonveyorBelt1, //x1

Sensors.TransportTableWorkpieceAvailable, //x2

Sensors.ConveyorBelt1WorkpieceAvailable, //x3

Sensors.Turntable1InLineWithConveyorBelt1, //x4

Sensors.Turntable1InLineWithConveyorBelt2, //x5

Sensors.Turntable1WorkpieceAvailable, //x6

Sensors.ConveyorBelt2WorkpieceAvailable, //x7

Sensors.Turntable2InLineWithConveyorBelt2, //x8

Sensors.Turntable2InLineWithConveyorBelt3, //x9

Sensors.Turntable2WorkpieceAvailable, //x10

Sensors.ConveyorBelt3WorkpieceAvailable, //x11

Sensors.MillingMachineAwayFromConveyorBelt2, //x12

Sensors.MillingMachineAtConveyorBelt2, //x13

Sensors.MillingHeadIsUp, //x14

Sensors.MillingHeadIsDown, //x15

Sensors.EmergencyStop //x16

};

switch (State)

{

case Init:

{

State = Init;

// переход заготовки с ленты CB2 на ленту CB3, если заготовка на ленте CB2 или на столе RT2

if (s[7] || s[10]) State = RT2\_init;

// переход заготовки с ленты CB3 на ленту CB1, если заготовка на TT или на ленте CB3

if (s[2] || s[11]) State = TT\_init;

// переход заготовки с ленты CB1 на ленту CB2, если заготовка на ленте CB1 или на столе RT1

if (s[3] || s[6]) State = RT1\_init;

//работа фрезерного станка, если заготовка на ленте CB2

if (s[7]) State = MM\_init;

break;

}

//перемещение груза с ленты CB2 на ленту CD3, при помощи стола RT2

case RT2\_init:

{

State = Init;

//вращение RT2 влево, если заготовка на ленте CB2 и стол RT2 не на одной линии с конвейерной лентой 2

if (s[7] && !s[8]) State = RT2\_rotate\_l;

//перемещение груза с ленты CB2 на стол RT2, если поворотный стол RT2 на одной линии с конвеерной лентой CB2 и заготовка не на ленте CB2

if (s[8] && !s[10]) State = RT2\_download;

//вращение RT2 вправо, если заготовка на поворотном столе RT2 и стол RT2 не на одной линии с конвейерной лентой 3

if (s[10] && !s[9]) State = RT2\_rotate\_r;

//перемещение груза со стола RT2 на ленту CB3, если поворотный стол RT2 на одной линии с конвейерной лентой CB3 и заготовка не на ленте CB3

if (s[9] && !s[11]) State = RT2\_upload;

//нахождение груза на ленте CB3, если поворотный стол RT2 на одной линии с конвейерной лентой CB3 и заготовка на ленте CB3

if (s[9] && s[11]) State = RT2\_to\_wait;

break;

}

case RT2\_rotate\_l: //вращение RT2 влево

{

SetY(9); //RT2 повернуть на конвейер CB2

if (s[7] && !s[8]) State = RT2\_rotate\_l;

else State = RT2\_init;

break;

}

case RT2\_download: //перемещение груза с ленты CB2 на стол RT2

{

SetY(8); //конвейерная лента CB2 - движение

SetY(11); //поворотный стол RT2 - движение

if (s[8] && !s[10]) State = RT2\_download;

else State = RT2\_init;

break; }

case RT2\_rotate\_r: //вращение RT2 вправо

{

SetY(10); //RT2 повернуть на конвейер CB3

if(!s[9] && s[10]) State = RT2\_rotate\_r;

else State = RT2\_init;

break;

}

case RT2\_upload: //перемещение груза со стола RT2 на ленту CB3

{

SetY(11); //поворотный стол RT2 - движение

SetY(12); //конвейерная лента CB3 - движение

if (s[9] && !s[11]) State = RT2\_upload;

else State = RT2\_init;

break;

}

//нахождение груза на ленте CB3 и возвращение стола RT2 в горизонтальное положение

case RT2\_to\_wait:

{

SetY(9); //RT2 повернуть на конвейер CB2

if (s[9] && s[11]) State = RT2\_to\_wait;

else State = RT2\_init;

break;

}

//переход заготовки с ленты CB3 на ленту CB1, при помощи транспортного стола TT

case TT\_init:

{

State = Init;

if (!s[0] && s[11]) State = TT\_move\_r;

if (s[0] && !s[2]) State = TT\_download;

if (!s[1] && s[2]) State = TT\_move\_l;

if (s[1] && !s[3]) State = TT\_upload;

if (s[1] && (s[3] || s[11])) State = TT\_to\_wait;

break;

}

case TT\_move\_r:

{

SetY(0);

if (!s[0] && s[11]) State = TT\_move\_r;

else State = TT\_init;

break;

}

case TT\_download:

{

SetY(3);

SetY(12);

if (s[0] && !s[2]) State =TT\_download;

else State = TT\_init;

break;

}

case TT\_move\_l:

{

SetY(1);

if (!s[1] && s[2]) State = TT\_move\_l;

else State = TT\_init;

break;

}

case TT\_upload:

{

SetY(2);

SetY(4);

if (s[1] && !s[3]) State = TT\_upload;

else State = TT\_init;

break;

}

case TT\_to\_wait:

{

SetY(0);

if (s[1] && (s[3] || s[11])) State = TT\_to\_wait;

else State = TT\_init;

break;

}

case RT1\_init:

{

State = Init;

if (s[3] && !s[4]) State = RT1\_rotate\_l;

if (!s[0] && s[4] && !s[6]) State = RT1\_download;

if (!s[5] && s[6]) State = RT1\_rotate\_r;

if (s[5] && !s[7]) State = RT1\_upload;

if (s[5] && (s[7] || s[3])) State = RT1\_to\_wait;

break;

}

case RT1\_rotate\_l:

{

SetY(5);

if (s[3] && !s[4]) State = RT1\_rotate\_l;

else State = RT1\_init;

break;

}

case RT1\_download:

{

SetY(4);

SetY(7);

if (!s[0] && s[4] && !s[6]) State = RT1\_download;

else State = RT1\_init;

break;

}

case RT1\_rotate\_r:

{

SetY(6);

if (!s[5] && s[6]) State = RT1\_rotate\_r;

else State = RT1\_init;

break;

}

case RT1\_upload:

{

SetY(7);

SetY(8);

if (s[5] && !s[7]) State = RT1\_upload;

else State = RT1\_init;

break;

}

case RT1\_to\_wait:

{

SetY(5);

if (s[5] && (s[7] || s[3])) State = RT1\_to\_wait;

else State = RT1\_init;

break;

}

case MM\_init:

{

State = Init;

if (s[7]) State = MM\_move\_f;

break;

}

case MM\_move\_f:

{

SetY(14);

if (s[7] && !s[13]) State = MM\_move\_f;

else State = MM\_down;

break;

}

case MM\_down:

{

SetY(16);

SetY(17);

if (s[7] && s[13] && !s[15]) State = MM\_down;

else State = MM\_work;

break;

}

case MM\_work:

{

SetY(17);

if (s[7] && s[13] && s[15]) State = MM\_work\_time;

break;

}

case MM\_work\_time:

{

delay(10000);

State = MM\_up;

break;

}

case MM\_up:

{

SetY(15);

if (!s[14]) State = MM\_up;

else State = MM\_move\_b;

break;

}

case MM\_move\_b:

{

SetY(13);

if (!s[12]) State = MM\_move\_b;

else State = RT2\_init;

break;

}

}

}

Для більш простої розробки вся система була розбита на 4 підстистеми: поворотний стіл 1 та 2, транспортний стіл та фрезерний верстат (***RT1\_init, RT2\_init, TT\_init, MM\_init***).

Функції ***SetY\_0()*** та ***SetY()*** слугують для скидання та встановлення ( нулів та одиниць) роботи відповідних пристроїв. Масив ***s[]*** відповідає за зберігання значень сенсорів.

**Висновки:** в ході виконання лабораторної роботи, було розроблено прошивку для мікроконтролеру, яка дозволяє пересувати заготовку та виконувати операції обробки станком над нею.