**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №5

з дисципліни "Верифікація цифрових систем"

на тему:

"Функціональне тестування для відаленної лабораторії GOLDi"

Варіант №7

Виконав:

студент групи КНТ-227 Ю. Є. Горіченко

Прийняв:

к.т.н., доцент Т. І. Каплієнко

2019

**Тема:** функціональне тестування для відаленної лабораторії GOLDi.

**Мета:** навчитись використовувати кінцеві автомати для функціонального тестування.

**Завдання на лабораторну роботу:**

1. Вивчити основні теоретичні відомості щодо функціонального тестування.
2. Розробити моделі кінцевих автоматів для функціонального тестування апаратних моделей віддаленої лабораторії GOLDi.

Розробити відповідні test cases.

**Виконання роботи**

Для 3D порталу розробити модель що обходить коло задану кількість разів у відповідному напряму та переносить вантаж з початкової точки у кінцеву.

|  |  |
| --- | --- |
| Код програми (С): |  |
| **UserDesign.h** | |
| // ######################################################################################  // # #  // # This module implements the users design #  // # #  // ######################################################################################  #ifndef \_USERDESIGN\_H  #define \_USERDESIGN\_H  #include "../Main.h"  extern void StateMachineInit(void); // This function inits the state machine  extern void StateMachineUpdate(void); // This function updated the state machine  // ######################################################################################  // # Add a new state for state maschine here #  // ######################################################################################  typedef enum {  WaitForFallingEdge,  GOTO\_Init,    GOTO\_Right\_Bottom,  GOTO\_Right\_Top,  GOTO\_Left\_Top,  GOTO\_Left\_Bottom,    GOTO\_Part\_X,  GOTO\_Part\_Y,  GOTO\_Dest\_X,  GOTO\_Dest\_Y,    DOWN\_Magnet,  UP\_Magnet,    ACTION\_Magnet  } AutomatStates\_t;    #endif | |

|  |  |
| --- | --- |
| Код програми (С): |  |
| **UserDesign.c** | |
| // ######################################################################################  // # #  // # This module implements the users design #  // # #  // ######################################################################################  #include <util/delay.h>  #include "UserDesign.h"  AutomatStates\_t State;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* SETTINGS \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  int CIRCLES = 1;  const int X\_Part = 32;  const int Y\_Part = 50;  const int X\_Dest = 163;  const int Y\_Dest = 9;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* DO NOT TOUCH \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  int MAGNET = 0;  int DONE = 0;  // ######################################################################################  // # This function initializes the finite state machine with start state #  // ######################################################################################  void StateMachineInit(void) {  State = WaitForFallingEdge;  }  // ######################################################################################  // # This function updates the current state of the finite state machine #  // ######################################################################################  void StateMachineUpdate(void) {  switch (State) {  case WaitForFallingEdge: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = 0;    if (!Sensors.UserSwitch) {  State = GOTO\_Right\_Bottom;  //State = GOTO\_Part\_X;  } else {  State = WaitForFallingEdge;  }    break;  }    case GOTO\_Init: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = !Sensors.XAxisAtPositionXMinus;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = !Sensors.YAxisAtPositionYMinus;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = 0;    break;  }    case GOTO\_Right\_Bottom: {  Actuators.XAxisDriveToXPlus = !Sensors.XAxisAtPositionXPlus;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = 0;    if (Sensors.XAxisAtPositionXPlus) {  State = GOTO\_Right\_Top;  }    break;  }    case GOTO\_Right\_Top: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = !Sensors.YAxisAtPositionYPlus;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = 0;    if (Sensors.YAxisAtPositionYPlus) {  State = GOTO\_Left\_Top;  }    break;  }    case GOTO\_Left\_Top: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = !Sensors.XAxisAtPositionXMinus;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = 0;    if (Sensors.XAxisAtPositionXMinus) {  State = GOTO\_Left\_Bottom;  }    break;  }    case GOTO\_Left\_Bottom: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = !Sensors.YAxisAtPositionYMinus;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = 0;    if (Sensors.YAxisAtPositionYMinus) {  CIRCLES--;    if (CIRCLES == 0) {  State = GOTO\_Right\_Bottom;  } else {  State = GOTO\_Part\_X;  }  }    break;  }    case GOTO\_Part\_X: {  Actuators.XAxisDriveToXPlus = 1;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = 0;    if (Sensors.PositionX == X\_Part) {  State = GOTO\_Part\_Y;  }    break;  }    case GOTO\_Part\_Y: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 1;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = 0;    if (Sensors.PositionY == Y\_Part) {  State = DOWN\_Magnet;  }    break;  }    case DOWN\_Magnet: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = !Sensors.ZAxisAtPositionZMinus;  Actuators.Magnet = MAGNET;    if (Sensors.ZAxisAtPositionZMinus) {  State = ACTION\_Magnet;  }    break;  }    case ACTION\_Magnet: {  if (MAGNET == 1) {  DONE = 1;  }  MAGNET = !MAGNET;    Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = MAGNET;    State = UP\_Magnet;  break;  }    case UP\_Magnet: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = !Sensors.ZAxisAtPositionZPlus;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = MAGNET;    if (Sensors.ZAxisAtPositionZPlus) {  if (!DONE) {  State = GOTO\_Dest\_X;  } else {  State = GOTO\_Init;  }  }    break;  }    case GOTO\_Dest\_X: {  Actuators.XAxisDriveToXPlus = 1;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 0;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = MAGNET;    if (Sensors.PositionX == X\_Dest) {  State = GOTO\_Dest\_Y;  }    break;  }    case GOTO\_Dest\_Y: {  Actuators.XAxisDriveToXPlus = 0;  Actuators.XAxisDriveToXMinus = 0;  Actuators.YAxisDriveToYPlus = 0;  Actuators.YAxisDriveToYMinus = 1;  Actuators.ZAxisDriveToZPlus = 0;  Actuators.ZAxisDriveToZMinus = 0;  Actuators.Magnet = MAGNET;    if (Sensors.PositionY == Y\_Dest) {  State = DOWN\_Magnet;  }    break;  }  }  } | |

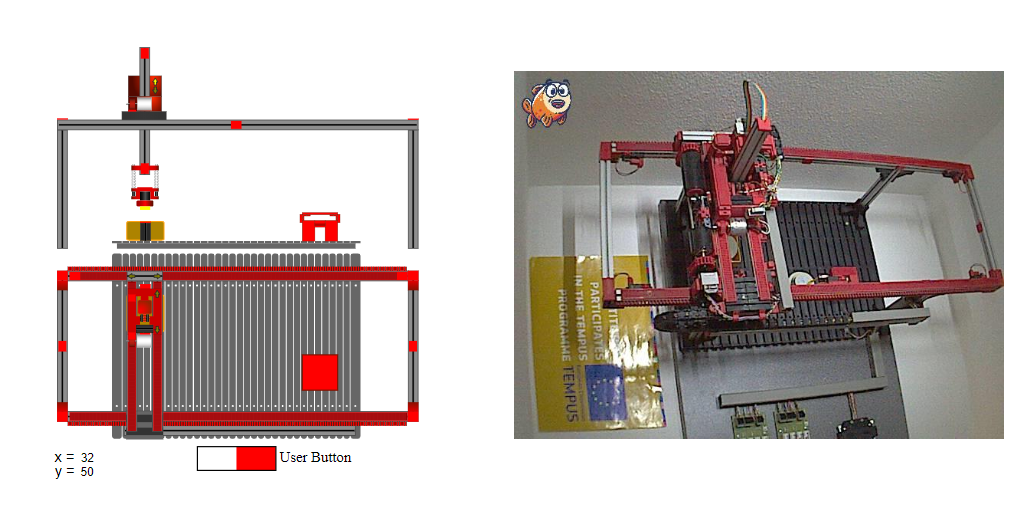
****

Рисунок 1.1 – Результат роботи 3-Axis-Portal

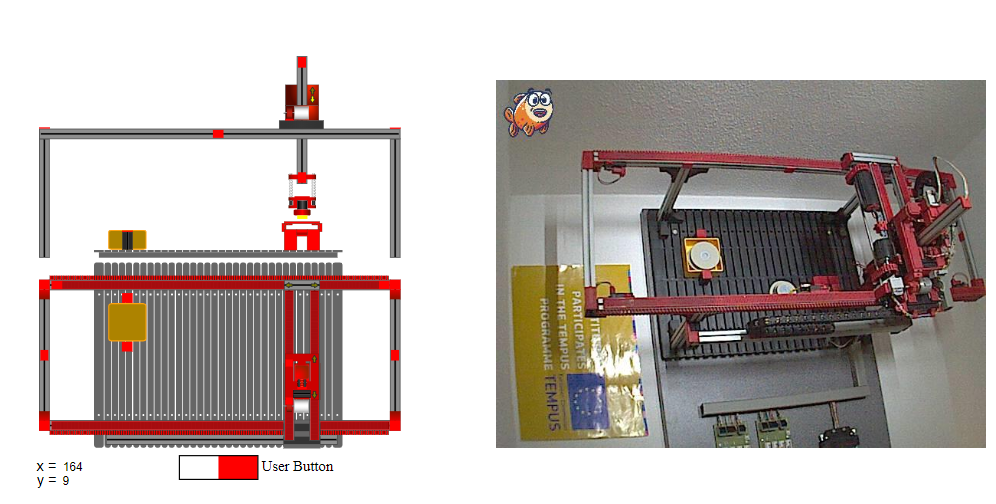


Рисунок 1.2 – Результат роботи 3-Axis-Portal

**Висновок:** в ході виконання лабораторної роботи навчився використовувати кінцеві автомати для функціонального тестування, а також, розробив модель кінцевого автомату для функціонального тестування апаратних моделей віддаленої лабораторії GOLDi.