**Міністерство освіти і науки України**

**Запорізький національний технічний університет**

кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи № 4

## з дисципліни «Верифікація цифрових систем»

на тему: **«Розробка прошивки для мікроконтролеру»**

Виконала:

студентка групи КНТ-715 А.B. Кущ

Прийняв :

к.н.т доцент Т. І. Каплієнко

2018

1. **Завдання на лабораторну роботу**

До моделі виробничої комірки розробити прошивку для мікроконтролеру, яка дозволяє пересувати заготовку, виконувати операції обробки станком над нею.

1. **Основні теоретичні відомості**

**Виробнича секція**

Модельна технологічна секція моделює технологічну секцію з інтегрованим круговим складом, яка , наприклад, використовується в промислових процесах різання металу.

Технологічна секція складається з рейкового візку з конвеєрною стрічкою, двох поворотних столів з конвеєрними ланцюгами, вертикального фрезерного верстата і трьох конвеєрних стрічок.

Різні транспортні засоби утворюють замкнутий цикл. Механічні перемикачі контролюють кінцеві положення рухомих конвеєрних блоків, а індуктивні безконтактні перемикачі перевіряють позиції деталей.

Модельований процес показує, як деталь потрапляє на круговий склад, потім за годинниковою стрілкою транспортується на вертикальний фрезерний верстат, де вона обробляється, і, нарешті, її передають на розвантажувальну станцію. Необроблена деталь укладається на конвеерну стрічку і переноситься на рейковий візок. Візок перевозить її до наступної конвеєрної стрічки. Там деталь транспортується до поворотного столу, який виконує поворот на 90° (+C - напрям), щоб перенести деталь на наступну конвеєрну стрічку. Ця конвеєрна стрічка транспортує її на фрезерний верстат. Вертикальна стійка виконує рухи уздовж своєї Y-осі, щоб наблизитися до деталі. Шпиндель починає обертатися і рухається в Z - напрямку, щоб здійснити послідовність обробки. Після завершення цієї послідовності, вертикальна стійка і шпиндель повертаються у своє стартове положення і оброблена деталь передається по конвееру на другий поворотний стіл,який повертає її готову знову на перший конвеєр.

|  |  |
| --- | --- |
| **Inputs / Sensors** | |
| **Х** | **Name** |
| x0 | Транспортний стіл в рядку з конвеєрною стрічкою 3 |
| x1 | Транспортний стіл в рядку з конвеєрною стрічкою 1 |
| x2 | Транспортний стіл - заготовка доступна |
| x3 | Конвеєрна стрічка 1 - заготовка доступна |
| x4 | Поворотний стіл 1 в лінії з конвеєрною стрічкою 1 |
| x5 | Поворотний стіл 1 в лінії з конвеєрною стрічкою 2 |
| x6 | Поворотний стіл 1 - заготовка в доступі |
| x7 | Конвеєрна стрічка 2 - заготовка доступна |
| x8 | Поворотний стіл 2 в лінії з конвеєрною стрічкою 2 |
| x9 | Поворотний стіл 2 в лінії з конвеєрною стрічкою 3 |
| x10 | Поворотний стіл 2 - заготовка в доступі |
| x11 | Конвеєрна стрічка 3 - заготовка доступна |
| x12 | Фрезерний верстат далеко від конвеєрної стрічки 2 |
| x13 | Фрезерний верстат біля конвеєрної стрічки 2 |
| x14 | Фрезерна головка вгорі |
| x15 | Фрезерна головка внизу |
| x16 | Екстрена зупинка |

|  |  |
| --- | --- |
| **Outputs / Actuators** | |
| **У** | **Name** |
| y0 | Транспортний стіл - рухатись до конвеєрної стрічки 3 |
| y1 | Транспортний стіл - рухатись до конвеєрної стрічки 1 |
| y2 | Транспортний стіл - рухати конвеєрну стрічку подібно конвеєрній стрічці 1 |
| y3 | Транспортний стіл - рухати конвеєрну стрічку подібно конвеєрній стрічці 3 |
| y4 | Конвеєрна стрічка 1 - рухати стрічку |
| y5 | Поворотний стіл 1 - повернути до конвеєрної стрічки 1 |
| y6 | Поворотний стіл 1 - повернути до конвеєрної стрічки 2 |
| y7 | Поворотний стіл 1 - рухати стрічку |
| y8 | Конвеєрна стрічка 2 - рухати стрічку |
| y9 | Поворотний стіл 2 - повернути до конвеєрної стрічки 2 |
| y10 | Поворотний стіл 2 - повернути до конвеєрної стрічки 3 |
| y11 | Поворотний стіл 2 - рухати стрічку |
| y12 | Конвеєрна стрічка 3 - рухати стрічку |
| y13 | Фрезерний верстат - віддалити від конвеєрної стрічки 2 |
| y14 | Фрезерний верстат - наблизитися до конвеєрної стрічки 2 |
| y15 | Фрезерна головка - підняти |
| y16 | Фрезерна головка - опускати |
| y17 | Фрезерна головка - керувати головкою |

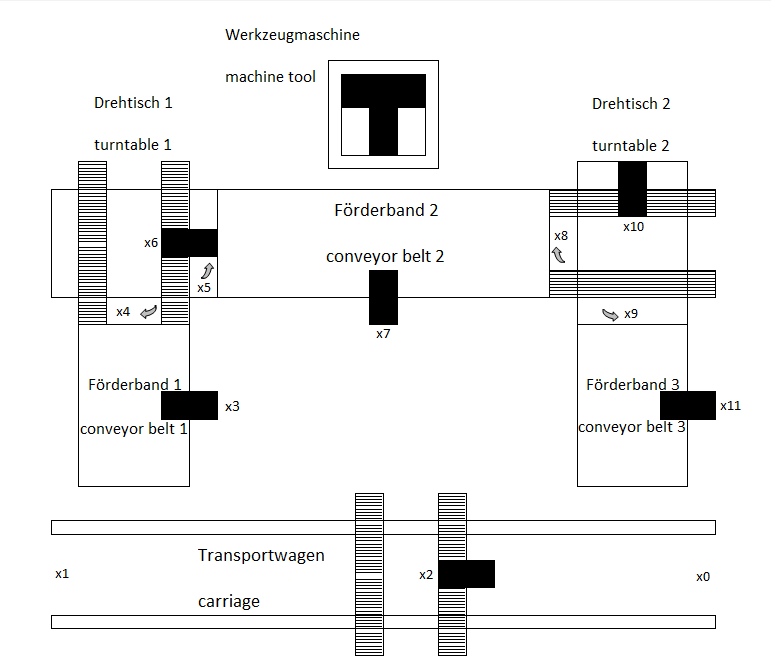


Рисунок 3.1 – Схематичне зображення виробничої секції

***Скінчений автомат (finite state machine – FSM),*** є особливим видом автомату – абстракції, що використовується для описання шляху зміни стану об’єкта в залежності від досягнутого стану та інформації отриманої ззовні. Його особливістю є скінченність множини станів автомату.

Для розробки FSM використовується графічний інтерактивний інструмент FSM (GIFT). Цей інструмент може створити FSM у вигляді машини станів (state machine), таблиці переходу (transition table), таблиці машин (machine table) або Z-рівнянь.

1. **Хід роботи**

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;

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.Turnable1DriveBelt = 0; // y7

Actuators.ConveyorBelt2DriveBelt = 0; // y8

Actuators.Turnable2RotateToConveyorBelt2 = 0; // y9

Actuators.Turnable2RotateToConveyorBelt\_3 = 0; // y10

Actuators.Turnable2DriveBelt = 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.Turnable1DriveBelt = 1; // y7

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

if (z == 9) Actuators.Turnable2RotateToConveyorBelt2 = 1; // y9

if (z == 10) Actuators.Turnable2RotateToConveyorBelt\_3 = 1; // y10

if (z == 11) Actuators.Turnable2DriveBelt = 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;

}

}

}

}

void StateMachineUpdate(void){

SetY\_0();

int s[] = {

Sensors.TransportTableInLineWithConveyorBelt3, // x0

Sensors.TransportTableInLineWithWonveyorBelt1, // x1

Sensors.TransportTableWorkpieceAvailable, // x2

Sensors.ConveoyrBelt1WorkpieceAvailable, // x3

Sensors.Turnable1InLineWithConveyorBelt1, // x4

Sensors.Turnable1InLineWithConveyorBelt2, // x5

Sensors.Turnable1WorkpieceAvailable, // x6

Sensors.ConveoyrBelt2WorkpieceAvailable, // x7

Sensors.Turnable2InLineWithConveyorBelt2, // x8

Sensors.Turnable2InLineWithConveyorBelt3, // x9

Sensors.Turnable2WorkpieceAvailable, // x10

Sensors.ConveoyrBelt3WorkpieceAvailable, // x11

Sensors.MillingMachineAwayFromConveyorBelt2, // x12

Sensors.MillingMachineAtConveyorBelt2, // x13

Sensors.MillingHeadIsUp, // x14

Sensors.MillingHeadIsDown, // x15

Sensors.EmergencyStop // x16

};

switch(State) {

case Init: {

State = Init;

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

if (s[2] || s[11]) State = TT2\_init;

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

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

break;

}

case RT2\_init: {

State = Init;

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

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

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

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

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

break;

}

case RT2\_rotate\_l: {

SetY(9);

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

else State = RT2\_init;

break;

}

case RT2\_download: {

SetY(8);

SetY(11);

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

else State = RT2\_init;

break;

}

case RT2\_rotate\_r: {

SetY(10);

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

else State = RT2\_init;

break;

}

case RT2\_upload: {

SetY(11);

SetY(12);

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

else State = RT2\_init;

break;

}

case RT2\_to\_wait: {

SetY(9);

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

else State = RT2\_init;

break;

}

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\_download;

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\_l: {

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(100000);

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;

}

}

}

**Результат виконання роботи**

Для більш простої розробки вся система була розбита на 4 підсистеми: поворотний стіл 1 та 2, транспортний стіл та фрезерний верстат.

Функції SetY\_0() та SetY() слугують для скидання та встановлення роботи відповідних пристроїв. Масив s[] слугує для зберігання значень сенсорів.

Стан Init слугує для визначення якій з підсистем зараз працювати (поворотному столу 1 чи 2, пересувному столу, чи верстату), далі система переходить в один з станів для початку роботи з підсистеми з заготовкою (RT1\_init, RT2\_init, TT\_init, MM\_init).

Всі підсистеми крім верстата можуть починати роботу з будь-якого стану (в залежності від стану всієї системи), верстат починає свою роботу з того, що заготовка приходить на місце обробки та закінчивши свою роботу виставляє стан системи в такий щоб заготовка продовжувала свій рух далі.

Кожна підсистема має стани:

* поворотні та транспортні столи мають такі стани:
  + визначення стану: RT1\_init, RT2\_init, TT\_init;
  + займання позиції для завантаження: RT1\_rotate, RT2\_ rotate, TT\_move\_r;
  + завантаження: RT1\_download, RT2\_ download, TT\_ download;
  + займання позиції для вивантаження: RT1\_rotate\_r, RT2\_ rotate\_r, TT\_move\_l;
  + вивантаження: RT1\_ upload, RT2\_ upload, TT\_ upload;
  + займання позиції очікування: RT1\_to\_wait, RT2\_to\_wait, TT\_to\_wait;
* Фрезерний верстат має такі стани :
  + початок роботи: MM\_init;
  + наблизитись до конвеєрної стрічки: MM\_ move\_f;
  + опустити фрезерну головку (вимкнувши її): MM\_down;
  + робота фрезерної головки: MM\_work;
  + затримка роботи: MM\_ work\_time;
  + підняти фрезерну головку MM\_ up;
  + очікування заготовки: MM\_ move\_b;

1. **Висновки**

На цій лабораторній роботі для моделі виробнича комірка було розроблено прошивку для мікроконтролеру, яка дозволяє пересувати заготовку та виконувати операції обробки станком над нею.