LessBag

By

MoreMoney

PROGRESS REPORT III FOR EE STUDENTS

Version 1.0

TEAM 7

January 6, 2012

Görkem Çakırhan	(HW)
Ceren Canpolat	(SW)
Alper Günçal	(HW)
M.Ali Yıldırım	(MAN)
Fatih Şahin	(IE)

Student submitting the report: Ceren Canpolat

Table of Contents

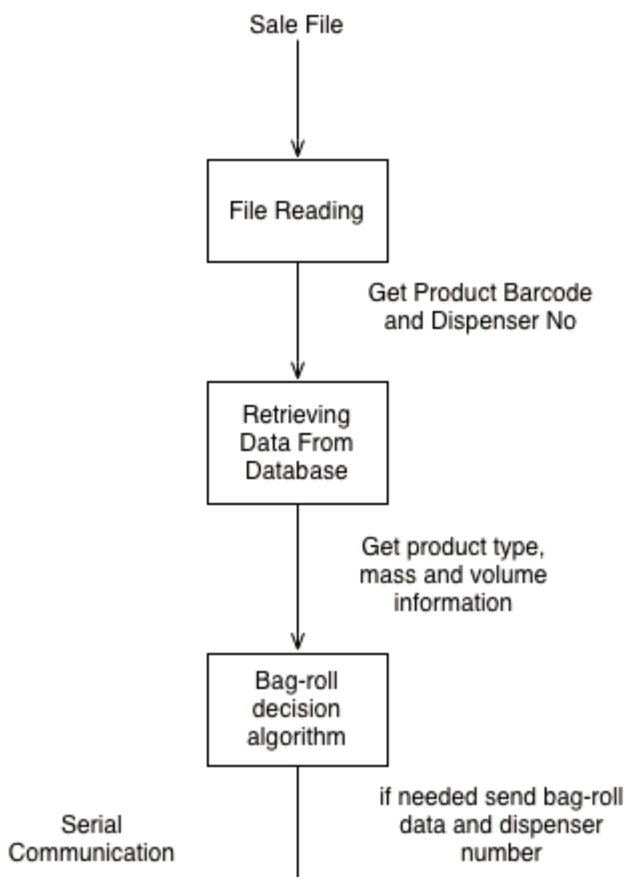
1. INTRODUCTION

1.1 About this Document and to Its Readers

Progress Report III includes the information about prototype and its test results. The reader will find the detailed design of the project and the codes written for it. Also the differences between the designed prototype and promised prototype are compared. The problems and difficulties through designing the prototype, and the solutions (if any) are given in Progress Report III. The report ends with a discussion of what have been done throughout the first semester, the unsolved problems and possible solutions to those problems that would be implemented in the second term.

2. SYSTEM

2.1 Software



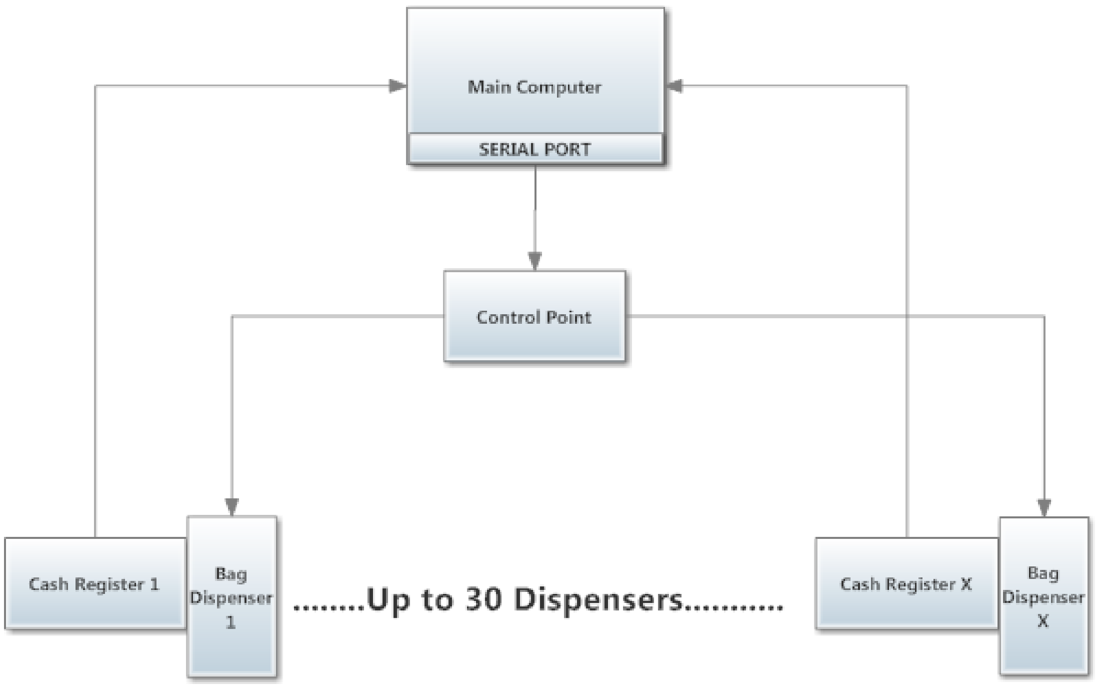
Sale File
File Reading
Get Product Barcode and Dispenser No
Retrieving Data From Database
Get product type, mass and volume information
Bag-roll decision algorithm
Serial Communication
if needed send bag-roll data and dispenser number

Figure 1: Software system overview

File reading part has file’s location at the main computer as an input, when that file’s size changes program will go and read the file in order to parse the 13 digit barcode number from it. After it gets the barcode number, this part sends that barcode value to the database in order to get the integer type, double volume (in cm^3) and double mass (in kgs) information of the product. After learning type, mass and volume information according to maximum mass and volume values of bags bag-type array will be filled and when a bag becomes necessary this algorithm will send the cash register number to the next task (serial communication) in order to deliver bag-roll request. When serial communication part of the software takes the cash register number it sends it to the serial port of the computer which has RS232 interface connected to the control point. Figure 1 summarizes the whole system flow.

2.2 Hardware

The general system is shown in Figure 2. Bag dispensers are connected to the control point, and the control point is connected to the main computer. The control point is capable of



handling up to 30 bag dispensers at the same time.

Figure 2: General system

Main computer calculates required number of bags and sends two byte of data which is the dispenser id. First byte is the ASCII value of the first digit and the second byte is the ASCII value of the second digit of the dispenser id. Control point processes the data and sends a bag giving request to the correct dispenser. Figure 3 shows the design of the control point. The Assembly code of the 8951 that used in control point is included in Appendix A.

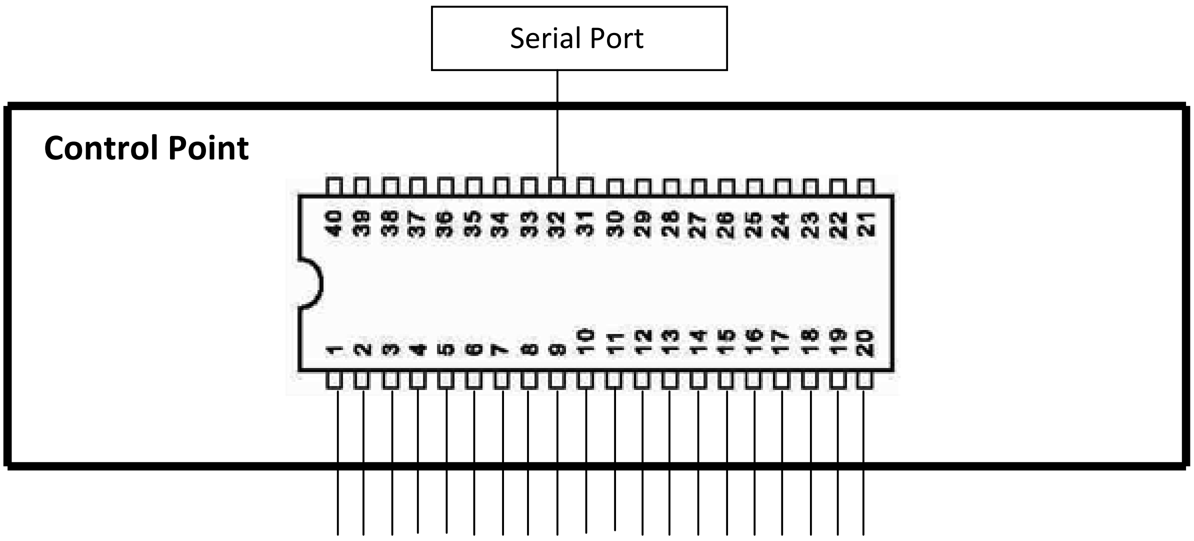


Figure 3: Detailed design of control point

Dispenser is responsible of giving a bag when a bag giving request comes. It uses a DC motor (90 rpm) to roll the bag, and uses a retro-reflective photoelectric sensor to ensure that exactly one bag is given. The sensor detects the spaces between rolled bags. A button, called 1-bag-give button, is included in the dispenser to let the cashier to give one bag when it's required without need of request that would come from control point. There is a motor driver (L293D) in the dispenser to drive the DC motor. 8951 microcontroller is the brain of the dispenser. It checks the request that would come from control point and also checks the 1-bag-give button, controls the driver, reads the sensor. Microcontroller uses 5V, driver uses both 5V and 12V, so we need a regulator that would supply both 12V and 5V. There is a two output regulator is included in the dispenser, so that the markets will directly connect the dispenser to the 220V AC standard voltage source, without need of any adaptor. The detailed design of the dispenser is shown in Figure 4. The Assembly code of the 8951 that is used in the dispenser is included in Appendix B.

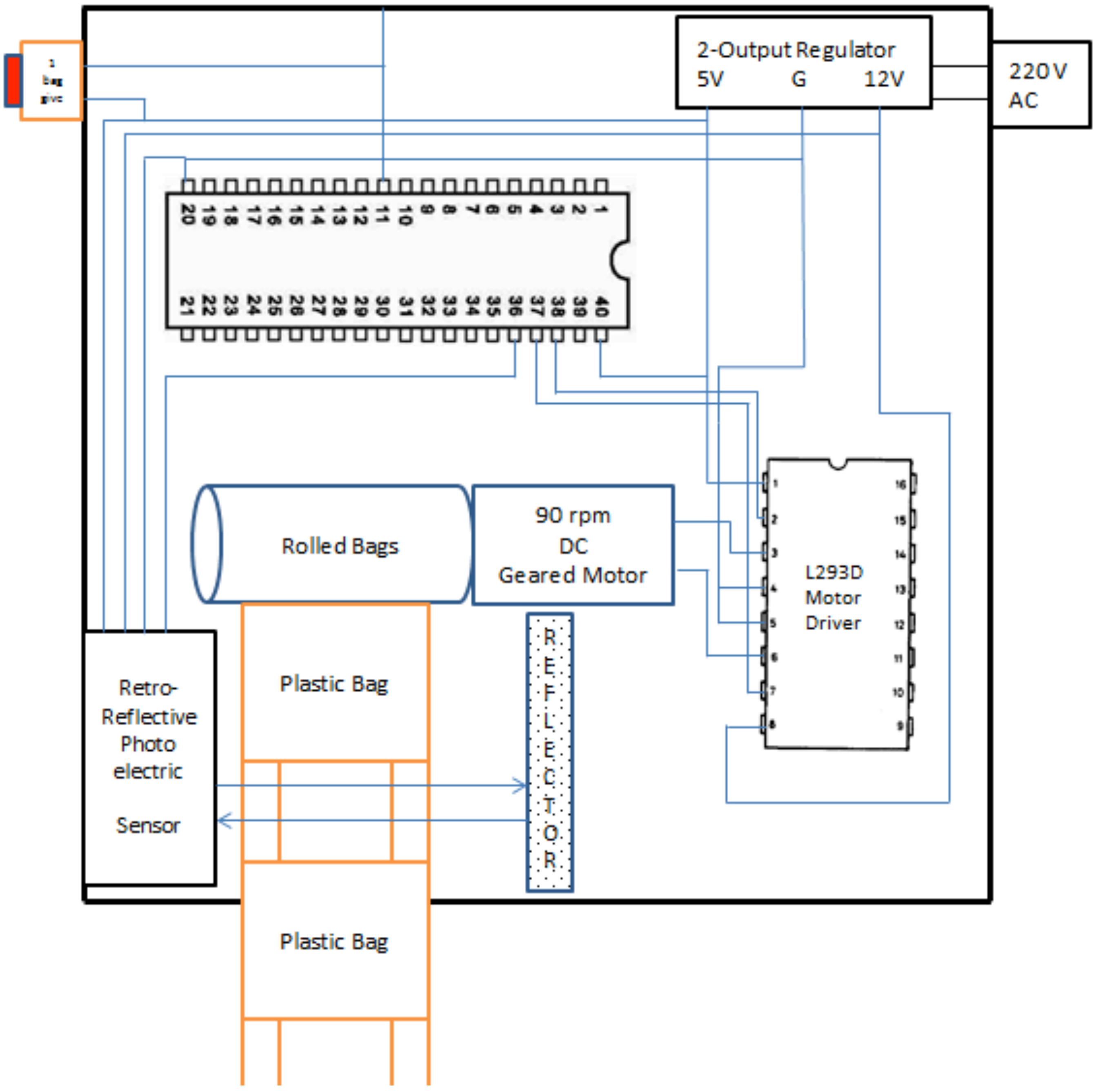


Figure 4: Detailed design of bag dispenser

3. PROTOTYPE

3.1 Comparison

3.1.1 Hardware Part Comparison with Technology Demonstration Test Plan

In the prototype test plan, we planned to take the correct data which contains the number of the bag dispenser via the serial port to control point and give the bag from the correct dispenser. We planned to give exactly one bag to the customer by using the sensor when one bag request comes from main computer. 1 bag button would be used to give a bag by the cashier whenever customer needs a bag. We planned to give the bag less than 1.3 seconds and avoid the bag to roll down when the customer pulls the bag from the dispenser. The system would not be locked when the customer pulls down the bag but there would be an opposing force to stop the bag from rolling.

We achieve to get the data which contains the cash register's number from the main computer to the control point by using serial communication. The control point sends this data to the correct dispenser's microcontroller and this dispenser's microcontroller starts the motor to give exactly one bag. We achieve to give exactly one bag by using Retro Reflective sensor. We give the bag in approximately 1.4 seconds and it is an acceptable result when compared to our expected value which was 1.3 seconds. Although the dispenser does not have locking capability to avoid the bags from rolling down while pulling the bags from the dispenser, the roll of the bag does not roll down when the customer takes the given bag from the dispenser as we planned in the test plan.

3.1.2 Software Part Comparison with Technology Demonstration Test Plan

At technology demonstration test plan, it was planned to show that all the tasks we have are doing their jobs expectedly. The duties of tasks were:

- Task 1 should be able to reach and read the sale file and give the barcode number and cash register number to Task 2.
  This task is doing its job expectedly. It reads the file where all the barcode numbers are written, and it parses the file to take the last barcode which is written to it and sends that number to Task2. However for this semester I have made the code for one cash register only, therefore the cash register number is fixed. In second semester we are planning to make the program with a multi-thread approach, therefore when a file

which belongs to one of the cash registers changes, program will automatically start a new thread so that it can calculate required number of bags and give the roll-bag request to that cash register parallel with other cash registers in case their files change too. In this approach in order to prevent any deadlock there should be a semaphore or mutex in order to connect database and serial port, since when threads want to reach to the database or serial port at the same time, this will not be possible.

- Task2 should be able to get the product information (type, mass, volume) that corresponds to the barcode number's row at the database and give that information to Task3.
  This task is working as expected. It establishes a connection with database and then takes the type, mass and volume information which corresponds to the information of the product whose barcode number came from Task1.

- Task3 should be able to determine if a bag-roll request need to be sent to Task4 in order to start serial communication with control point.
  This task is the smart algorithm where the decision whether a bag is required or not is given.
  This task works as expectedly. This task has a bag-type array whose rows are type numbers and columns are (odd ones) volume and (even ones) mass information of the bags. Therefore as expected when a new product comes, task 3 puts its volume and mass to the correct place in the bag-type array and sends a bag-roll request to Task4 if needed. Whenever bag's volume or mass becomes greater than or equal to the max possible volume and mass for a bag, it stops adding new products in that bag. This bag roll request is simply have the cash registers number in it. In Progress Report II, I have planned to add a checksum to this request, however since I am only considering one cash register here this checksum becomes unnecessary for this semester; however I will add it in multi-thread version of the program.

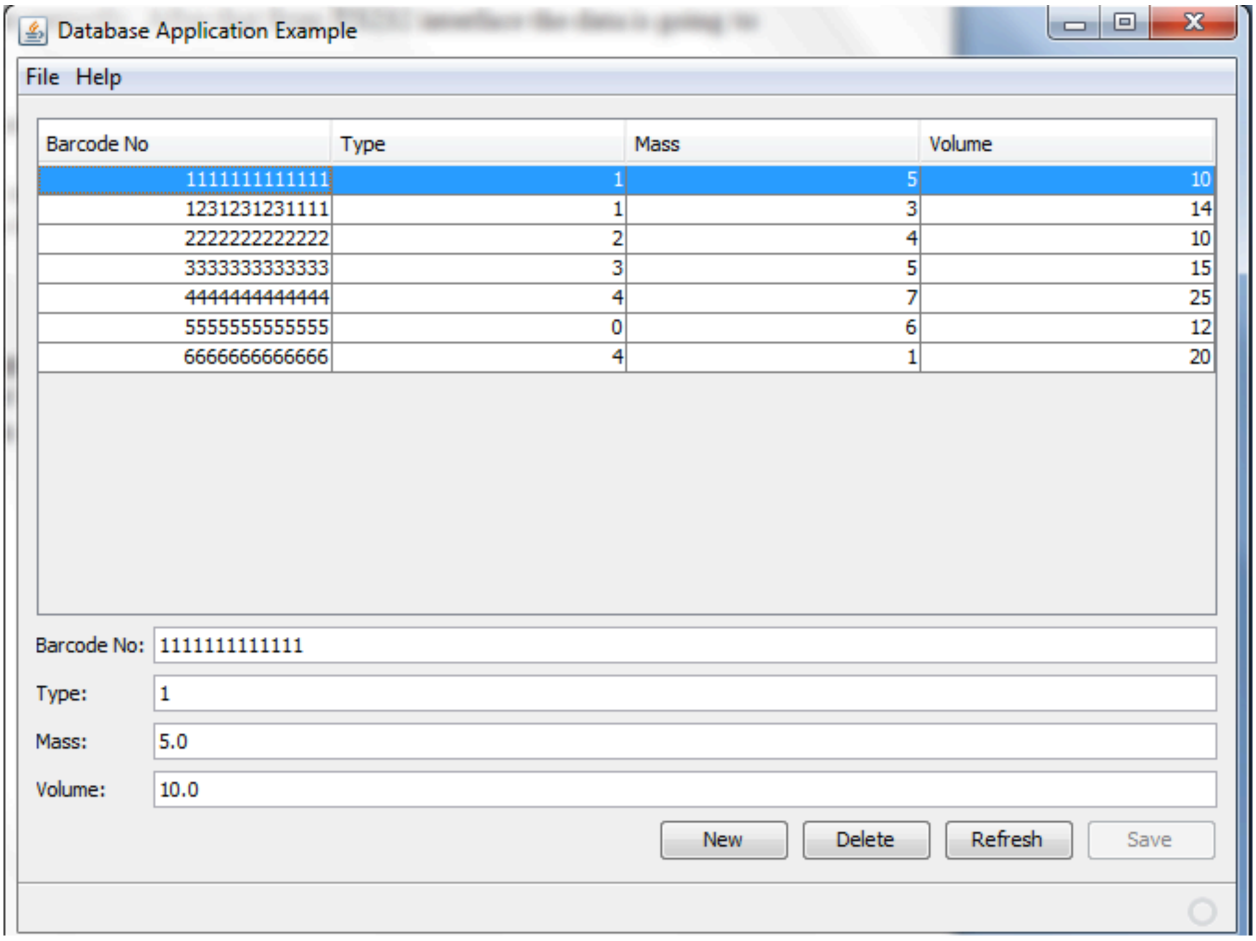
- Task4 should be able to get the bag roll request of dispenserX from Task3 and send it to control point's microcontroller with serial communication.
  This task is working expectedly; it takes the cash register number and send to the serial port of the main computer correctly. After that from RS232 interface the data is going to the control point serially.

- Task5 should be able to send an interrupt signal when control point sends a bag roll request to dispenser X.

  This task is doing its job as well, it takes the cash register number and from its specific pin, it sends a 0 to 1 transition to the specific cash register.

3.2 Test Results

To be able to test the program we added dummy product to the database and write their barcode numbers in order to the file to see if bag request is being sent correctly.
I made maximum mass in each type 8kgs and max volume is 50 cm^3. Then in order I have written 7 products.



Barcode No	Type	Mass	Volume
1111111111111	1	5	10
1231231231111	1	3	14
2222222222222	2	4	10
3333333333333	3	5	15
4444444444444	4	7	25
5555555555555	0	6	12
6666666666666	4	1	20

Barcode No: 1111111111111
Type: 1
Mass: 5.0
Volume: 10.0

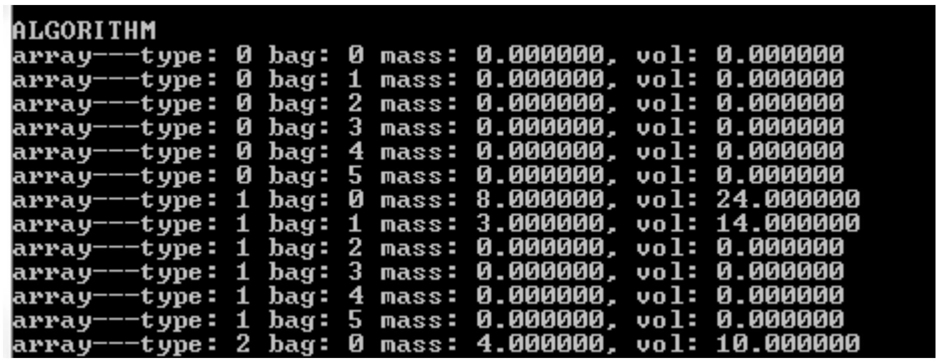
Figure 5: Database interface with dummy products

Then in order we have added them to the file. Then we simulated a shop with the combinations of the products, and for each required bag request program sent the necessary data to serial port. For example:

For product series:

1111111111111 1231231231111 2222222222222 1231231231111

3 bag requests have been sent in order: two from type 1 and one from type 2.
From bag-type array we can see that:



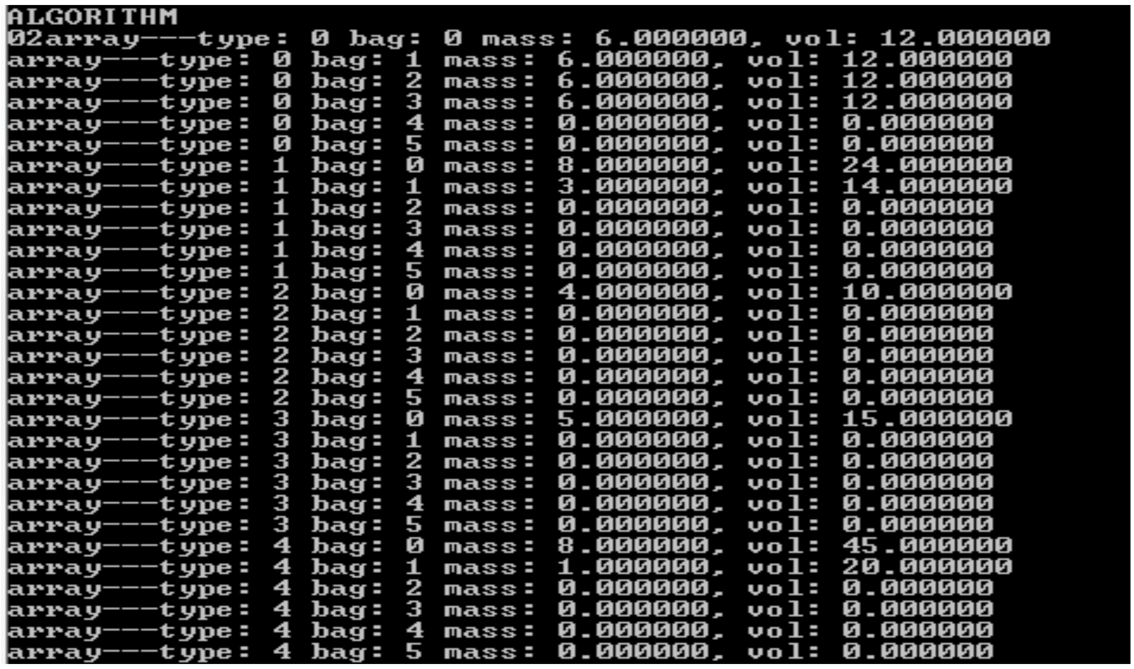
```
ALGORITHM
array---type: 0 bag: 0 mass: 0.000000, vol: 0.000000
array---type: 0 bag: 1 mass: 0.000000, vol: 0.000000
array---type: 0 bag: 2 mass: 0.000000, vol: 0.000000
array---type: 0 bag: 3 mass: 0.000000, vol: 0.000000
array---type: 0 bag: 4 mass: 0.000000, vol: 0.000000
array---type: 0 bag: 5 mass: 0.000000, vol: 0.000000
array---type: 1 bag: 0 mass: 8.000000, vol: 24.000000
array---type: 1 bag: 1 mass: 3.000000, vol: 14.000000
array---type: 1 bag: 2 mass: 0.000000, vol: 0.000000
array---type: 1 bag: 3 mass: 0.000000, vol: 0.000000
array---type: 1 bag: 4 mass: 0.000000, vol: 0.000000
array---type: 1 bag: 5 mass: 0.000000, vol: 0.000000
array---type: 2 bag: 0 mass: 4.000000, vol: 10.000000
```

Figure 6: Software output for 3 bag requests

For a larger shopping:
Product series:

5555555555555 5555555555555 5555555555555 5555555555555 1111111111111 1231231231111 1231231231111 2222222222222 3333333333333 4444444444444 6666666666666 6666666666666

We expect that for four type 0 products: 5555555555555since their mass values are large there should be four different bag requests, for the first two Type 1 products: 1111111111111 1231231231111 there should be one bag request, for the next type 1 product 1231231231111 there should be another bag request, for the type 2 and type 3 products 2222222222222 3333333333333 there should be separate two bag requests, and finally for the first two type 4 products: 4444444444444 6666666666666 there should be one bag request and for the next type 4 product 6666666666666 there should be another bag request. This adds up to 4 type 0 bags, 2 type 1 bags, 1 type 2, 1 type 3 bags and 2 type 4 bags. Now let's look at the content of bag-type array to confirm this result:
The last content of the type-bag array is:



```
ALGORITHM
02array---type: 0 bag: 0 mass: 6.000000, vol: 12.000000
array---type: 0 bag: 1 mass: 6.000000, vol: 12.000000
array---type: 0 bag: 2 mass: 6.000000, vol: 12.000000
array---type: 0 bag: 3 mass: 6.000000, vol: 12.000000
array---type: 0 bag: 4 mass: 0.000000, vol: 0.000000
array---type: 0 bag: 5 mass: 0.000000, vol: 0.000000
array---type: 1 bag: 0 mass: 8.000000, vol: 24.000000
array---type: 1 bag: 1 mass: 3.000000, vol: 14.000000
array---type: 1 bag: 2 mass: 0.000000, vol: 0.000000
array---type: 1 bag: 3 mass: 0.000000, vol: 0.000000
array---type: 1 bag: 4 mass: 0.000000, vol: 0.000000
array---type: 1 bag: 5 mass: 0.000000, vol: 0.000000
array---type: 2 bag: 0 mass: 4.000000, vol: 10.000000
array---type: 2 bag: 1 mass: 0.000000, vol: 0.000000
array---type: 2 bag: 2 mass: 0.000000, vol: 0.000000
array---type: 2 bag: 3 mass: 0.000000, vol: 0.000000
array---type: 2 bag: 4 mass: 0.000000, vol: 0.000000
array---type: 2 bag: 5 mass: 0.000000, vol: 0.000000
array---type: 3 bag: 0 mass: 5.000000, vol: 15.000000
array---type: 3 bag: 1 mass: 0.000000, vol: 0.000000
array---type: 3 bag: 2 mass: 0.000000, vol: 0.000000
array---type: 3 bag: 3 mass: 0.000000, vol: 0.000000
array---type: 3 bag: 4 mass: 0.000000, vol: 0.000000
array---type: 3 bag: 5 mass: 0.000000, vol: 0.000000
array---type: 4 bag: 0 mass: 8.000000, vol: 45.000000
array---type: 4 bag: 1 mass: 1.000000, vol: 20.000000
array---type: 4 bag: 2 mass: 0.000000, vol: 0.000000
array---type: 4 bag: 3 mass: 0.000000, vol: 0.000000
array---type: 4 bag: 4 mass: 0.000000, vol: 0.000000
array---type: 4 bag: 5 mass: 0.000000, vol: 0.000000
```

Figure 7: Software output for 10 bag requests

As we can see from the array type 0 has 4 bags open, type 1 has 2 bags open, type 2 has 1 bag open, type 3 has 1 bag open and type 4 has 2 bags open. So algorithm works correctly.

This was the test of Task 1, Task 2 and Task3. For Task4 and Task5 we connected main computer to the hardware part of the product and with the bag requests we were able to roll the bags from dispenser.

3.3 Problems Report

3.3.1 Hardware Problems

-We faced with some problems while programming the Arduino Uno. We were getting an error message that many users do and cannot solve.

-There was a problem in sending data via serial communication from the main computer to control point.

- We had a difficulty to find a sensor which can detect transparent objects. It was very difficult to find a sensor that will detect line (not area), and also will work properly for transparent objects.

- We faced a problem when we connect the ground of the microcontrollers, sensor and the motor differently and we cannot get 5V from the microcontroller for the request pin which starts the bag giving process.

3.3.2 Software Problems

-The main problem in this version of the software part of the project is its cash register number's being fixed. Since there are lots of cash registers in the supermarkets, this program which depends on only one cash register makes the product inefficient. Except this problem software part is working correctly as it was planned to be.

-Another problem is since serial communication needs some delay about 100 miliseconds between sending two bytes, these 100 milliseconds slows system a little bit, even though it is not a big number it was unexpected.

3.4 Corrections Proposed or Made

3.4.1 Hardware Corrections Proposed or Made

-When we realized that Arduino Uno sometimes does not work properly, we decided to use Atmet 8951 microcontroller instead of using Arduino.

-We solve serial communication problem by sending the ASCII equivalent of the characters rather sending character itself.

-First of all, we used an infrared sensor which transmitter and receiver placed abreast but this sensor mostly could not detect our transparent plastic bag. After we searched about the sensors, we found retro-reflective photoelectric sensor which can detect the transparent object as our plastic bag.

- We solve grounding problem by making a common ground for all of the grounds of the components.

3.4.2 Software Corrections Proposed or Made

To the problem of fixed cash register number, I came up with the solution of multithreading programming. When there are customers at each cash register at the same time, program will check each sale file of them then when changes occur it will create a thread to be able to do multitasking. Each thread will parse its own sale file and get the barcode number from it, then it will pass it to task 2 in order to make a connection with database, however if threads tries to connect the database at the same time this causes a problem therefore database will be protected with a semaphore or a mutex. After that, each tread will have its own bag-type array which is controlled by the algorithm that decides whether a bag should be rolled down or not. After that if a bag request needed threads will send their cash register numbers to the serial port which will deliver to the control point by RS232 interface. However since serial port cannot deliver the each thread's requests at the same time, it should be protected with a semaphore or a mutex too.

4. CONCLUSION

In the software, main part of the project has been completed in order to see how accurate and efficient the LessBag will be. To finish the software part completely in the second semester all the necessary corrections which has been explained in the problems and corrections proposed parts will be done in order to have a perfect product. The work that has been completed so far has four main parts which are file reading, database connection, calculation of number of bags algorithm and serial communication. The file reading part was all about parsing the cash register's sale file in order to retrieve the barcode numbers of the products which was trivial, in the database connection part there need to be an interface supplied to customer in order to create comfort while using LessBag. At the calculation of required bag numbers algorithm the values of maximum mass and volume of the bag needed to be chosen carefully in order to have accurate results in bag numbers which is the most important part of the project. While implementing this part there has been discussions in order to decide the best values, however we could not determine that numbers because we needed more statistical data, so at this point the maximum values may not be accurate. In the second semester we will extend our surveys to be able to get those accurate values. In the serial communication part the sending data process required some critical decisions such as baud rate, timeout constants and timeout multipliers, these decisions have been made after some process to find out the best numbers. In order to have the best working version of the product each decision means a lot so they should be decided carefully, after examining each situation and realizing the possible problems all the decisions have been made accordingly. As a conclusion, at software part of the project some problems have been faced with however majority of them have been handled after a testing process.

In the hardware part, we achieve the parts that we specified in the demonstration test plan. We chose the correct DC motor to give the bag in the required time. We had some difficulties to choose the sensor which can detect transparent objects. We tried many different sensors and decides to use retro-reflective photoelectric sensor which detect transparent objects. In taking the data from the serial communication part, we firstly used Arduino Uno but after we faced with some programming problems with it, we chose to use Atmel 8951 microcontroller. We complete taking the correct data which contains the cash register's id number via the serial communication and evaluate this data in the control point to send the data to the correct cash register's microcontroller. In the bag giving process, we achieve to give the bag from the correct cash register by using the cash register's id which comes from the main computer. We accomplish to give exactly one bag at each bag giving process by using our sensor and the bag is given in approximately 1.4 seconds. We also achieve to avoid the rolling of the bags when the customer rips it from the dispenser. We are planning to put a locking capability to the product to avoid the bags from rolling while the customer rips it.

APPENDICES

APPENDIX A

The assembly code of control point is given below.

```
ORG 0
        MOV P0, #0
        MOV P1, #0
        MOV P2, #0

        MOV TMOD,#20H;
        MOV TH1,#0FDH;

        MOV SCON,#50H;
        SETB TR1;
        CLR RI;
        HERE_1:JNB RI,HERE_1;
        MOV A,SBUF;        First Digit
        CLR RI;
        HERE_2:JNB RI,HERE_2;
        MOV R1,SBUF;       Second Digit
        CLR RI;



        CJNE A,#30H,NOT_ZERO
                CJNE R1,#31H,ZERO_NOT_1
                SETB P0.0
                LCALL QSDELAY
                CLR P0.0
                 LJMP HERE_1;
                ZERO_NOT_1:
                CJNE R1,#32H,ZERO_NOT_2
                SETB P0.1
                LCALL QSDELAY
                CLR P0.1
                 LJMP HERE_1;
                ZERO_NOT_2:
                CJNE R1,#33H,ZERO_NOT_3
                SETB P0.2
                LCALL QSDELAY
                CLR P0.2
                 LJMP HERE_1;
                ZERO_NOT_3:
                CJNE R1,#34H,ZERO_NOT_4
                SETB P0.3
                LCALL QSDELAY
                CLR P0.3
                 LJMP HERE_1;
                ZERO_NOT_4:
                CJNE R1,#35H,ZERO_NOT_5
                SETB P0.4
                LCALL QSDELAY
                CLR P0.4
```

```
                     LJMP HERE_1;
                    ZERO_NOT_5:
                    CJNE R1,#36H,ZERO_NOT_6
                    SETB P0.5
                    LCALL QSDELAY
                    CLR P0.5
                     LJMP HERE_1;
                    ZERO_NOT_6:
                    CJNE R1,#37H,ZERO_NOT_7
                    SETB P0.6
                    LCALL QSDELAY
                    CLR P0.6
                     LJMP HERE_1;
                    ZERO_NOT_7:
                    CJNE R1,#38H,ZERO_NOT_8
                    SETB P0.7
                    LCALL QSDELAY
                    CLR P0.7
                     LJMP HERE_1;
                    ZERO_NOT_8:
                    SETB P1.0
                    LCALL QSDELAY
                    CLR P1.0
                     LJMP HERE_1;

NOT_ZERO:CJNE A,#31H,NOT_ONE
                    CJNE R1,#30H,ONE_NOT_ZERO
                    SETB P1.1
                    LCALL QSDELAY
                    CLR P1.1
                     LJMP HERE_1;
                    ONE_NOT_ZERO:
                    CJNE R1,#31H,ONE_NOT_1
                    SETB P1.2
                    LCALL QSDELAY
                    CLR P1.2
                     LJMP HERE_1;
                    ONE_NOT_1:
                    CJNE R1,#32H,ONE_NOT_2
                    SETB P1.3
                    LCALL QSDELAY
                    CLR P1.3
                     LJMP HERE_1;
                    ONE_NOT_2:
                    CJNE R1,#33H,ONE_NOT_3
                    SETB P1.4
                    LCALL QSDELAY
                    CLR P1.4
                     LJMP HERE_1;
                    ONE_NOT_3:
                    CJNE R1,#34H,ONE_NOT_4
                    SETB P1.5
                    LCALL QSDELAY
                    CLR P1.5
                     LJMP HERE_1;
                    ONE_NOT_4:
```

```
                CJNE R1,#35H,ONE_NOT_5
                SETB P1.6
                LCALL QSDELAY
                CLR P1.6
                 LJMP HERE_1;
                ONE_NOT_5:
                CJNE R1,#36H,ONE_NOT_6
                SETB P1.7
                LCALL QSDELAY
                CLR P1.7
                 LJMP HERE_1;
                ONE_NOT_6:
                CJNE R1,#37H,ONE_NOT_7
                SETB P2.0
                LCALL QSDELAY
                CLR P2.0
                 LJMP HERE_1;
                ONE_NOT_7:
                CJNE R1,#38H,ONE_NOT_8
                SETB P2.1
                LCALL QSDELAY
                CLR P2.1
                 LJMP HERE_1;
                ONE_NOT_8:
                SETB P2.2
                LCALL QSDELAY
                CLR P2.2
                 LJMP HERE_1;



NOT_ONE:CJNE A,#32H,NOT_TWO
                CJNE R1,#30H,TWO_NOT_ZERO
                SETB P2.3
                LCALL QSDELAY
                CLR P2.3
                 LJMP HERE_1;
                TWO_NOT_ZERO:
                CJNE R1,#31H,TWO_NOT_1
                SETB P2.4
                LCALL QSDELAY
                CLR P2.4
                 LJMP HERE_1;
                TWO_NOT_1:
                CJNE R1,#32H,TWO_NOT_2
                SETB P2.5
                LCALL QSDELAY
                CLR P2.5
                 LJMP HERE_1;
                TWO_NOT_2:
                CJNE R1,#33H,TWO_NOT_3
                SETB P2.6
                LCALL QSDELAY
                CLR P2.6
                 LJMP HERE_1;
                TWO_NOT_3:
```

```
                SETB P2.7
                LCALL QSDELAY
                CLR P2.7
                 LJMP HERE_1;

        NOT_TWO:



        LJMP HERE_1;


QSDELAY:
MOV R5, #11
H3: MOV R4,#248
H2: MOV R3, #255
H1: DJNZ R3,H1
DJNZ R4, H2
DJNZ R5,H3
RET
END
```

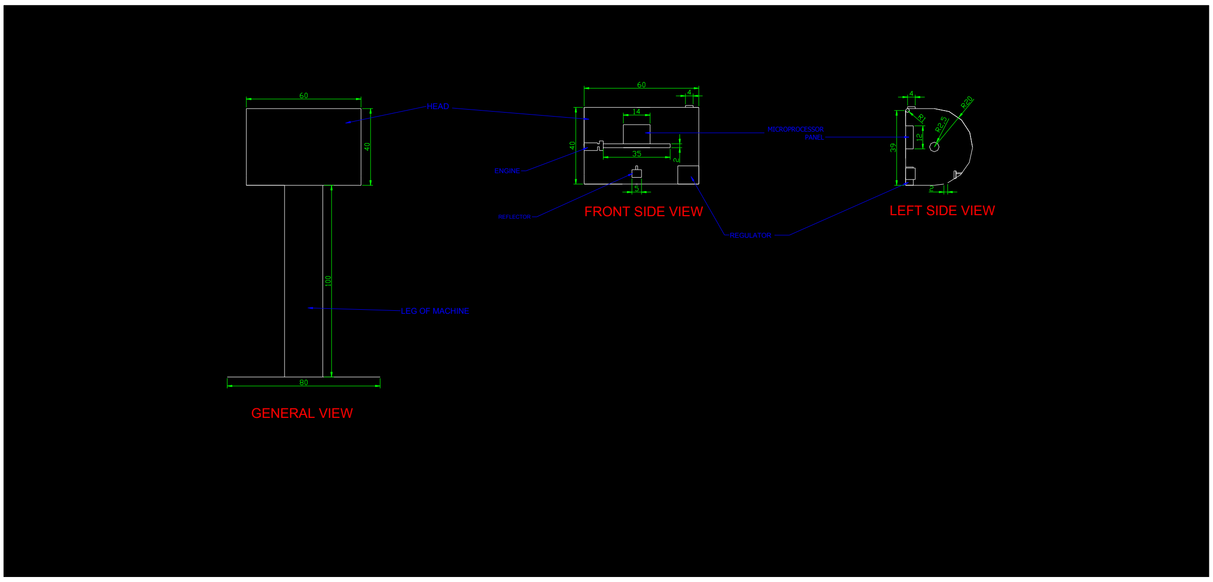
APPENDIX B

The assembly code of dispenser is given below.

```
CLR P1.1
CLR P1.2
SETB P1.0 ; MAKE P1.0 INPUT - ISTEK PINI
;CLR P1.0
SETB P1.5 ; MAKE P1.5 INPUT - SENSOR DATA
CLR P1.5

;MOTOR + P1.1
;MOTOR - P1.2


HERE: JNB P1.0, HERE
SETB P1.1
CLR P1.2        ;MOTOR RUNS
LCALL QSDELAY
HERE2: JB P1.5, HERE2
LCALL QSDELAY
HERE3: JNB P1.5, HERE3
LCALL QSDELAY

CLR P1.1
CLR P1.2        ;MOTOR STOPS
LJMP HERE

QSDELAY:
MOV R5, #4
H3: MOV R4,#248
H2: MOV R3, #255
H1: DJNZ R3,H1
DJNZ R4, H2
DJNZ R5,H3
RET
```

APPENDIX C

Mechanical drawing of LessBag is given below. Note that, this is a predesign and it will be improved in the second term.



APPENDIX D

Main computer software program code is given below.

```
/*

================================================================================
=
 Name        : heyyo.c
 Author      : Ceren Canpolat
 Version     : 1.0
 Copyright   : Your copyright notice
 Description : LessBag

================================================================================
=
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include <mysql/mysql.h>
#include <limits.h>
#include <stdint.h>
#include <unistd.h>  /* UNIX standard function definitions */
#include <fcntl.h>   /* File control definitions */
#include <errno.h>   /* Error number definitions */

int main(void) {

int FS=0;
int FNS;
char *AllData;
char *NewData;
char *Barcode;
char *Data;
double TypeBag[5][100]; //Even indexes of rows are current mass information
//and odd ones are current volume information of the bags rolled down.
//Rows are types 0:Hygiene, 1:Fragile, 2:Crushable, 3:1-bag-required, 4:Other
double MAXMASS[5];          //max mass values determined for a bag for types 0,1,2,3,4
double MAXVOLUME[5];       //max volume values determined for a bag for types 0,1,2,3,4
double ProductMASS;
double ProductVOL;
int type = 0;
int count[5];
int incre=0;
int b=13;
int ia = 0;
int ib = 0;                  // number of rolled bags for types 0,1,2,3,4
   for(ia = 0 ; ia < 5 ; ia++)
   {
      MAXMASS[ia] = 8;
      MAXVOLUME[ia] =50 ;
      count[ia] = 0;
      for(ib = 0 ; ib < 100 ; ib++)
      {
         TypeBag[ia][ib] = 0;
      }
   }
         int RollABag=0;
         int state=0;
         double mass;
         double volume;
         int n=0;
         int number=0;
         char Roll;
         while(1)
         {

         //TASK1++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
         FILE *fp;
         const char* const filename = "C:/Users/Huseyin/Desktop/CASHREG/SATISANI.txt";
                  fp = fopen ( filename, "r" ) ;
                  if( fp == NULL )
                  {
                           puts ( " cannot open file" ) ;
                           exit(0) ;
                  }
                  if (fp!=NULL)
                  {
                    fseek(fp, 0 ,SEEK_END);
                    FNS = ftell(fp);
                    fclose(fp);
                  }
                  if (FNS!=FS)
                  {
           AllData = (char *) calloc(FNS+1,1); // + null terminator
           NewData = (char *) calloc(FNS+1,1);
           Barcode = (char *) calloc(FNS+1,1);
           fp = fopen ( filename, "r" ) ;
           if (!fread( AllData,1, FNS, fp))
           {
                printf("file read error\n");
            }
            fclose(fp);
            int i=0;
            int a=0;
            while(i<=FNS)
            {
                if(*(AllData+i) >= '0' && *(AllData+i) <='9')
                {
                    *(NewData+a)=*(AllData+i);
                   // printf("newdata:%s\n",NewData);
                    a++;
                }
                i++;
               // printf("a:%d\n",a);
            }
            //printf("DENEME\n");
           // printf("newdata:%s\n",NewData);
           // printf("DENEME BITTI\n");

            int k;
        for(k=0;k<13;k++)
        {
                *(Barcode+k)=*(NewData+(b*incre)+k);
         }
               incre++;
            printf("barcode:%s\n",Barcode);

//TASK2+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
            printf("DATABASE\n");
         MYSQL *conn;
         MYSQL_RES *result;
         MYSQL_ROW row;
         MYSQL_FIELD *field;
         int num_fields;
         char query_buf[100];
         sprintf(query_buf, "SELECT type, mass, volume FROM lessbag where barcodeNo = '%s'
", Barcode);
         conn = mysql_init(NULL);
         mysql_real_connect(conn, "localhost", "root", "root", "lessbag", 0, NULL, 0);
         mysql_query(conn, query_buf);
         result = mysql_store_result(conn);
         num_fields = mysql_num_fields(result);
         while ((row = mysql_fetch_row(result)))
         {
          for(i = 0; i < num_fields; i++)
          {
            printf("\n");
            double temp;
            if(i==0)
            {
              type=  atoi(row[i]);
              printf("type: %d\n",type);
            }
            if(i==1)
            {
              ProductMASS=  atof(row[i]);
              printf("mass: %f\n",ProductMASS);
            }
            if(i==2)
             {
               ProductVOL=  atof(row[i]);
               printf("volume: %f\n",ProductVOL);
             }
          }
        }
        printf("\n");

        mysql_free_result(result);
        mysql_close(conn);

//TASK3++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
            printf("ALGORITHM\n");
            i=0;

        while(state==0)
            {
                    if(i==2*count[type])
                    {
                            RollABag=1;
                            TypeBag[type][i]=TypeBag[type][i]+ProductMASS;
                            TypeBag[type][i+1]=TypeBag[type][i+1]+ProductVOL;
                            count[type]++;
                            state=1;
                    }
            //printf("%d",i);
                    if((i==0   ||   i%2==0)   &&(MAXMASS[type]-TypeBag[type][i]>=ProductMASS)
&& (MAXVOLUME[type]-TypeBag[type][i+1]>=ProductVOL) && state==0)
                    {
                            TypeBag[type][i]=TypeBag[type][i]+ProductMASS;
                            TypeBag[type][i+1]=TypeBag[type][i+1]+ProductVOL;
                            state=1;
                    }
                    i=i+2;
            }
            state=0;
             for(ia = 0 ; ia < 5 ; ia++)
        {
            for(ib = 0 ; ib <= 10 ; ib++)
            {
 printf("array---type:      %d      bag:      %d      mass:      %f,      vol:      %f\n",ia,ib/2,
TypeBag[ia][ib],TypeBag[ia][ib+1]) ;
               ib++;
            }
        }
        ProductMASS=0;
        ProductVOL=0;
        printf("RollABag %d for type %d bagnumber %d \n",RollABag,type,count[type]);


//TASK4+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
            //printf("SERIAL\n");
            if (RollABag==1)
            {
                    RollABag=0;
//SERIAL COMMUNICATION
                   HANDLE hSerial;
   hSerial=
CreateFile("COM5",GENERIC_WRITE,0,0,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0);
                    if(hSerial==INVALID_HANDLE_VALUE)
                    {
                    if(GetLastError()==ERROR_FILE_NOT_FOUND)
                   {
                            printf("serial port does not exist\n!");
                   }
                             printf("an error occured\n");
                   }
                   DCB dcbSerialParams={0};
                   DCB dcbSerial;
                   dcbSerial.DCBlength=sizeof(dcbSerialParams);
                   if(!GetCommState(hSerial, &dcbSerialParams))
                   {
                             printf("error getting state\n");
                    }
                   dcbSerialParams.BaudRate=CBR_9600;
                   dcbSerialParams.ByteSize=8;
                   dcbSerialParams.StopBits=ONESTOPBIT;
                   dcbSerialParams.Parity=NOPARITY;
```

```
                 if(!SetCommState(hSerial, &dcbSerialParams))
                {
                            printf("error setting state\n");
                }
                 COMMTIMEOUTS timeouts={0};
                 timeouts.WriteTotalTimeoutConstant=50;
                 timeouts.WriteTotalTimeoutMultiplier=10;
                if(!SetCommTimeouts(hSerial, &timeouts))
                 {
                            printf("error setting state with timeout\n");
                }
                 char azBuff [1]={48};
                char azBuff2 [1]={ 57};
                DWORD dwBytesRead;
                DWORD dwBytesRead2;
                 if(!WriteFile(hSerial, azBuff,1, &dwBytesRead,NULL))
                 {
                                        printf("some error in writing1\n");
                }
                Sleep(100);
                 if(!WriteFile(hSerial, azBuff2,1, &dwBytesRead2,NULL))
                {
                             printf("some error in writing2\n");
                }
                 printf("seriala gondermis olmasi lazm\n");
                CloseHandle(hSerial);
         }
         printf("RollABag %d\n",RollABag);
         printf("FNS: %d Bytes \n",FNS);
          printf("FS: %d Bytes \n",FS);
          FS=FNS;
                }
       }
       puts("MOREMONEY");
       return EXIT_SUCCESS;
}
```

APPENDIX E

Database interface code is given below.

LessBag.java

```
package moremoney;

import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;
import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.Transient;

/**
 *
 * @author Ceren
 */
@Entity
@Table(name = "lessbag", catalog = "lessbag", schema = "")
@NamedQueries({
   @NamedQuery(name = "Lessbag.findAll", query = "SELECT l FROM Lessbag l"),
   @NamedQuery(name = "Lessbag.findByBarcodeNo", query = "SELECT l FROM
Lessbag l WHERE l.barcodeNo = :barcodeNo"),
   @NamedQuery(name = "Lessbag.findByType", query = "SELECT l FROM
Lessbag l WHERE l.type = :type"),
   @NamedQuery(name = "Lessbag.findByMass", query = "SELECT l FROM
Lessbag l WHERE l.mass = :mass"),
   @NamedQuery(name = "Lessbag.findByVolume", query = "SELECT l FROM
Lessbag l WHERE l.volume = :volume")})
public class Lessbag implements Serializable {
   @Transient
   private PropertyChangeSupport changeSupport = new
PropertyChangeSupport(this);
   private static final long serialVersionUID = 1L;
   @Id
   @Basic(optional = false)
   @Column(name = "barcodeNo")
   private Long barcodeNo;
   @Basic(optional = false)
   @Column(name = "type")
   private int type;
   @Basic(optional = false)
   @Column(name = "mass")
   private double mass;
   @Basic(optional = false)
```

```
@Column(name = "volume")
private double volume;

public Lessbag() {
}

public Lessbag(Long barcodeNo) {
   this.barcodeNo = barcodeNo;
}

public Lessbag(Long barcodeNo, int type, double mass, double volume) {
   this.barcodeNo = barcodeNo;
   this.type = type;
   this.mass = mass;
   this.volume = volume;
}

public Long getBarcodeNo() {
   return barcodeNo;
}

public void setBarcodeNo(Long barcodeNo) {
   Long oldBarcodeNo = this.barcodeNo;
   this.barcodeNo = barcodeNo;
   changeSupport.firePropertyChange("barcodeNo", oldBarcodeNo, barcodeNo);
}

public int getType() {
   return type;
}

public void setType(int type) {
   int oldType = this.type;
   this.type = type;
   changeSupport.firePropertyChange("type", oldType, type);
}

public double getMass() {
   return mass;
}

public void setMass(double mass) {
   double oldMass = this.mass;
   this.mass = mass;
   changeSupport.firePropertyChange("mass", oldMass, mass);
}

public double getVolume() {
   return volume;
}
```

```
    public void setVolume(double volume) {
        double oldVolume = this.volume;
        this.volume = volume;
        changeSupport.firePropertyChange("volume", oldVolume, volume);
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (barcodeNo != null ? barcodeNo.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Lessbag)) {
            return false;
        }
        Lessbag other = (Lessbag) object;
        if ((this.barcodeNo == null && other.barcodeNo != null) || (this.barcodeNo !=
null && !this.barcodeNo.equals(other.barcodeNo))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "moremoney.Lessbag[ barcodeNo=" + barcodeNo + " ]";
    }

    public void addPropertyChangeListener(PropertyChangeListener listener) {
        changeSupport.addPropertyChangeListener(listener);
    }

    public void removePropertyChangeListener(PropertyChangeListener listener) {
        changeSupport.removePropertyChangeListener(listener);
    }

}


/*
 * MoreMoneyApp.java
 */

package moremoney;
```

```
import org.jdesktop.application.Application;
import org.jdesktop.application.SingleFrameApplication;

/**
 * The main class of the application.
 */
public class MoreMoneyApp extends SingleFrameApplication {

    /**
     * At startup create and show the main frame of the application.
     */
    @Override protected void startup() {
        show(new MoreMoneyView(this));
    }

    /**
     * This method is to initialize the specified window by injecting resources.
     * Windows shown in our application come fully initialized from the GUI
     * builder, so this additional configuration is not needed.
     */
    @Override protected void configureWindow(java.awt.Window root) {
    }

    /**
     * A convenient static getter for the application instance.
     * @return the instance of MoreMoneyApp
     */
    public static MoreMoneyApp getApplication() {
        return Application.getInstance(MoreMoneyApp.class);
    }

    /**
     * Main method launching the application.
     */
    public static void main(String[] args) {
        launch(MoreMoneyApp.class, args);
    }
}


/*
 * MoreMoneyView.java
 */

package moremoney;

import org.jdesktop.application.Action;
import org.jdesktop.application.ResourceMap;
import org.jdesktop.application.SingleFrameApplication;
import org.jdesktop.application.FrameView;
import org.jdesktop.application.TaskMonitor;
import org.jdesktop.application.Task;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;
import javax.persistence.RollbackException;
import javax.swing.Timer;
import javax.swing.Icon;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import org.jdesktop.beansbinding.AbstractBindingListener;
import org.jdesktop.beansbinding.Binding;
import org.jdesktop.beansbinding.PropertyStateEvent;

/**
 * The application's main frame.
 */
public class MoreMoneyView extends FrameView {

    public MoreMoneyView(SingleFrameApplication app) {
        super(app);

        initComponents();

        // status bar initialization - message timeout, idle icon and busy animation, etc
        ResourceMap resourceMap = getResourceMap();
        int messageTimeout = resourceMap.getInteger("StatusBar.messageTimeout");
          messageTimer = new Timer(messageTimeout, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                statusMessageLabel.setText("");
            }
        });
          messageTimer.setRepeats(false);
        int                              busyAnimationRate                                  =
resourceMap.getInteger("StatusBar.busyAnimationRate");
        for (int i = 0; i < busyIcons.length; i++) {
            busyIcons[i] = resourceMap.getIcon("StatusBar.busyIcons[" + i + "]");
        }
        busyIconTimer = new Timer(busyAnimationRate, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                busyIconIndex = (busyIconIndex + 1) % busyIcons.length;
                statusAnimationLabel.setIcon(busyIcons[busyIconIndex]);
            }
        });
        idleIcon = resourceMap.getIcon("StatusBar.idleIcon");
        statusAnimationLabel.setIcon(idleIcon);
        progressBar.setVisible(false);
        // connecting action tasks to status bar via TaskMonitor
        TaskMonitor taskMonitor = new TaskMonitor(getApplication().getContext());
        taskMonitor.addPropertyChangeListener(new
java.beans.PropertyChangeListener() {
            public void propertyChange(java.beans.PropertyChangeEvent evt) {
                String propertyName = evt.getPropertyName();
                if ("started".equals(propertyName)) {
                    if (!busyIconTimer.isRunning()) {
                        statusAnimationLabel.setIcon(busyIcons[0]);
                        busyIconIndex = 0;
                        busyIconTimer.start();
                    }
                    progressBar.setVisible(true);
                    progressBar.setIndeterminate(true);
                } else if ("done".equals(propertyName)) {
                    busyIconTimer.stop();
                    statusAnimationLabel.setIcon(idleIcon);
                    progressBar.setVisible(false);
                    progressBar.setValue(0);
                } else if ("message".equals(propertyName)) {
                    String text = (String)(evt.getNewValue());
                    statusMessageLabel.setText((text == null) ? "" : text);
                    messageTimer.restart();
                } else if ("progress".equals(propertyName)) {
                    int value = (Integer)(evt.getNewValue());
                    progressBar.setVisible(true);
                    progressBar.setIndeterminate(false);
                    progressBar.setValue(value);
                }
            }
        });

        // tracking table selection
        masterTable.getSelectionModel().addListSelectionListener(
            new ListSelectionListener() {
                public void valueChanged(ListSelectionEvent e) {
                    firePropertyChange("recordSelected",                    !isRecordSelected(),
isRecordSelected());
                }
            });

        // tracking changes to save
        bindingGroup.addBindingListener(new AbstractBindingListener() {
            @Override
            public void targetChanged(Binding binding, PropertyStateEvent event) {
                // save action observes saveNeeded property
                setSaveNeeded(true);
            }
        });
        // have a transaction started
        entityManager.getTransaction().begin();
    }


    public boolean isSaveNeeded() {
        return saveNeeded;
    }

    private void setSaveNeeded(boolean saveNeeded) {
        if (saveNeeded != this.saveNeeded) {
            this.saveNeeded = saveNeeded;
            firePropertyChange("saveNeeded", !saveNeeded, saveNeeded);
        }
    }

    public boolean isRecordSelected() {
        return masterTable.getSelectedRow() != -1;
    }


    @Action
    public void newRecord() {
        moremoney.Lessbag l = new moremoney.Lessbag();
        entityManager.persist(l);
        list.add(l);
        int row = list.size()-1;
        masterTable.setRowSelectionInterval(row, row);
        masterTable.scrollRectToVisible(masterTable.getCellRect(row, 0, true));
        setSaveNeeded(true);
    }

    @Action(enabledProperty = "recordSelected")
    public void deleteRecord() {
        int[] selected = masterTable.getSelectedRows();
        List<moremoney.Lessbag>            toRemove            =            new
ArrayList<moremoney.Lessbag>(selected.length);
        for (int idx=0; idx<selected.length; idx++) {
            moremoney.Lessbag                      l                      =
list.get(masterTable.convertRowIndexToModel(selected[idx]));
            toRemove.add(l);
            entityManager.remove(l);
        }
        list.removeAll(toRemove);
        setSaveNeeded(true);
    }


    @Action(enabledProperty = "saveNeeded")
    public Task save() {
        return new SaveTask(getApplication());
    }

    private class SaveTask extends Task {
        SaveTask(org.jdesktop.application.Application app) {
            super(app);
        }
        @Override protected Void doInBackground() {
            try {
                entityManager.getTransaction().commit();
                entityManager.getTransaction().begin();
            } catch (RollbackException rex) {
                rex.printStackTrace();
                entityManager.getTransaction().begin();
                List<moremoney.Lessbag>            merged            =            new
ArrayList<moremoney.Lessbag>(list.size());
                for (moremoney.Lessbag l : list) {
                    merged.add(entityManager.merge(l));
                }
                list.clear();
                list.addAll(merged);
            }
            return null;
        }
        @Override protected void finished() {
            setSaveNeeded(false);
        }
    }

    /**
     * An example action method showing how to create asynchronous tasks
     * (running on background) and how to show their progress. Note the
     * artificial 'Thread.sleep' calls making the task long enough to see the
     * progress visualization - remove the sleeps for real application.
     */
    @Action
    public Task refresh() {
       return new RefreshTask(getApplication());
    }

    private class RefreshTask extends Task {
        RefreshTask(org.jdesktop.application.Application app) {
            super(app);
        }
        @SuppressWarnings("unchecked")
        @Override protected Void doInBackground() {
            try {
                setProgress(0, 0, 4);
                setMessage("Rolling back the current changes...");
                setProgress(1, 0, 4);
                entityManager.getTransaction().rollback();
                Thread.sleep(1000L); // remove for real app
                setProgress(2, 0, 4);

                setMessage("Starting a new transaction...");
                entityManager.getTransaction().begin();
                Thread.sleep(500L); // remove for real app
                setProgress(3, 0, 4);

                setMessage("Fetching new data...");
                java.util.Collection data = query.getResultList();
                for (Object entity : data) {
                    entityManager.refresh(entity);
                }
                Thread.sleep(1300L); // remove for real app
                setProgress(4, 0, 4);

                Thread.sleep(150L); // remove for real app
                list.clear();
                list.addAll(data);
            } catch(InterruptedException ignore) { }
            return null;
        }
        @Override protected void finished() {
            setMessage("Done.");
            setSaveNeeded(false);
        }
    }

    @Action
    public void showAboutBox() {
        if (aboutBox == null) {
            JFrame mainFrame = MoreMoneyApp.getApplication().getMainFrame();
            aboutBox = new MoreMoneyAboutBox(mainFrame);
            aboutBox.setLocationRelativeTo(mainFrame);
        }
        MoreMoneyApp.getApplication().show(aboutBox);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        bindingGroup = new org.jdesktop.beansbinding.BindingGroup();
        mainPanel = new javax.swing.JPanel();
        masterScrollPane = new javax.swing.JScrollPane();
        masterTable = new javax.swing.JTable();
        barcodeNoLabel = new javax.swing.JLabel();
        typeLabel = new javax.swing.JLabel();
        massLabel = new javax.swing.JLabel();
        volumeLabel = new javax.swing.JLabel();
        barcodeNoField = new javax.swing.JTextField();
        typeField = new javax.swing.JTextField();
        massField = new javax.swing.JTextField();
        volumeField = new javax.swing.JTextField();
        saveButton = new javax.swing.JButton();
        refreshButton = new javax.swing.JButton();
        newButton = new javax.swing.JButton();
        deleteButton = new javax.swing.JButton();
        menuBar = new javax.swing.JMenuBar();
        javax.swing.JMenu fileMenu = new javax.swing.JMenu();
        javax.swing.JMenuItem newRecordMenuItem = new javax.swing.JMenuItem();
        javax.swing.JMenuItem deleteRecordMenuItem = new
    javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        javax.swing.JMenuItem saveMenuItem = new javax.swing.JMenuItem();
        javax.swing.JMenuItem refreshMenuItem = new javax.swing.JMenuItem();
        jSeparator2 = new javax.swing.JSeparator();
        javax.swing.JMenuItem exitMenuItem = new javax.swing.JMenuItem();
        javax.swing.JMenu helpMenu = new javax.swing.JMenu();
        javax.swing.JMenuItem aboutMenuItem = new javax.swing.JMenuItem();
        statusPanel = new javax.swing.JPanel();
        javax.swing.JSeparator statusPanelSeparator = new javax.swing.JSeparator();
        statusMessageLabel = new javax.swing.JLabel();
        statusAnimationLabel = new javax.swing.JLabel();
        progressBar = new javax.swing.JProgressBar();
        org.jdesktop.application.ResourceMap resourceMap =
    org.jdesktop.application.Application.getInstance(moremoney.MoreMoneyApp.class).g
    etContext().getResourceMap(MoreMoneyView.class);
        entityManager = java.beans.Beans.isDesignTime() ? null :
    javax.persistence.Persistence.createEntityManagerFactory(resourceMap.getString("ent
    ityManager.persistenceUnit")).createEntityManager(); // NOI18N
        query = java.beans.Beans.isDesignTime() ? null :
    entityManager.createQuery(resourceMap.getString("query.query")); // NOI18N
        list = java.beans.Beans.isDesignTime() ? java.util.Collections.emptyList() :
    org.jdesktop.observablecollections.ObservableCollections.observableList(query.getRe
    sultList());

        mainPanel.setName("mainPanel"); // NOI18N

        masterScrollPane.setName("masterScrollPane"); // NOI18N

        masterTable.setName("masterTable"); // NOI18N
```

```
        org.jdesktop.swingbinding.JTableBinding                jTableBinding               =
org.jdesktop.swingbinding.SwingBindings.createJTableBinding(org.jdesktop.beansbin
ding.AutoBinding.UpdateStrategy.READ_WRITE, list, masterTable);
        org.jdesktop.swingbinding.JTableBinding.ColumnBinding      columnBinding      =
jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${b
arcodeNo}"));
        columnBinding.setColumnName("Barcode No");
        columnBinding.setColumnClass(Long.class);
        columnBinding                                                                    =
jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${t
ype}"));
        columnBinding.setColumnName("Type");
        columnBinding.setColumnClass(Integer.class);
        columnBinding                                                                    =
jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${
mass}"));
        columnBinding.setColumnName("Mass");
        columnBinding.setColumnClass(Double.class);
        columnBinding                                                                    =
jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${v
olume}"));
        columnBinding.setColumnName("Volume");
        columnBinding.setColumnClass(Double.class);
        bindingGroup.addBinding(jTableBinding);

        masterScrollPane.setViewportView(masterTable);

        barcodeNoLabel.setText(resourceMap.getString("barcodeNoLabel.text"));         //
NOI18N
        barcodeNoLabel.setName("barcodeNoLabel"); // NOI18N

        typeLabel.setText(resourceMap.getString("typeLabel.text")); // NOI18N
        typeLabel.setName("typeLabel"); // NOI18N

        massLabel.setText(resourceMap.getString("massLabel.text")); // NOI18N
        massLabel.setName("massLabel"); // NOI18N

        volumeLabel.setText(resourceMap.getString("volumeLabel.text")); // NOI18N
        volumeLabel.setName("volumeLabel"); // NOI18N

        barcodeNoField.setName("barcodeNoField"); // NOI18N

        org.jdesktop.beansbinding.Binding                  binding                   =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.Au
toBinding.UpdateStrategy.READ_WRITE,                                masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.barcodeNo}"),
barcodeNoField, org.jdesktop.beansbinding.BeanProperty.create("text"));
        binding.setSourceUnreadableValue(null);
        bindingGroup.addBinding(binding);
        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.Au
toBinding.UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"),
barcodeNoField, org.jdesktop.beansbinding.BeanProperty.create("enabled"));
        bindingGroup.addBinding(binding);

        typeField.setName("typeField"); // NOI18N

        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.Au
toBinding.UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.type}"), typeField,
org.jdesktop.beansbinding.BeanProperty.create("text"));
        binding.setSourceUnreadableValue(null);
        bindingGroup.addBinding(binding);
        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.Au
toBinding.UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"),
typeField, org.jdesktop.beansbinding.BeanProperty.create("enabled"));
        bindingGroup.addBinding(binding);

        massField.setName("massField"); // NOI18N

        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.Au
toBinding.UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.mass}"), massField,
org.jdesktop.beansbinding.BeanProperty.create("text"));
        binding.setSourceUnreadableValue(null);
        bindingGroup.addBinding(binding);
        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.Au
toBinding.UpdateStrategy.READ, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement != null}"),
massField, org.jdesktop.beansbinding.BeanProperty.create("enabled"));
        bindingGroup.addBinding(binding);

        volumeField.setName("volumeField"); // NOI18N

        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.Au
toBinding.UpdateStrategy.READ_WRITE, masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement.volume}"),
volumeField, org.jdesktop.beansbinding.BeanProperty.create("text"));
        binding.setSourceUnreadableValue(null);
        bindingGroup.addBinding(binding);
        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.Au
toBinding.UpdateStrategy.READ,                                          masterTable,
org.jdesktop.beansbinding.ELProperty.create("${selectedElement        !=       null}"),
volumeField, org.jdesktop.beansbinding.BeanProperty.create("enabled"));
        bindingGroup.addBinding(binding);

        javax.swing.ActionMap                        actionMap                              =
org.jdesktop.application.Application.getInstance(moremoney.MoreMoneyApp.class).g
etContext().getActionMap(MoreMoneyView.class, this);
        saveButton.setAction(actionMap.get("save")); // NOI18N
        saveButton.setName("saveButton"); // NOI18N

        refreshButton.setAction(actionMap.get("refresh")); // NOI18N
        refreshButton.setName("refreshButton"); // NOI18N

        newButton.setAction(actionMap.get("newRecord")); // NOI18N
        newButton.setName("newButton"); // NOI18N

        deleteButton.setAction(actionMap.get("deleteRecord")); // NOI18N
        deleteButton.setName("deleteButton"); // NOI18N

        javax.swing.GroupLayout              mainPanelLayout              =              new
javax.swing.GroupLayout(mainPanel);
        mainPanel.setLayout(mainPanelLayout);
        mainPanelLayout.setHorizontalGroup(

mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
mainPanelLayout.createSequentialGroup()
                .addComponent(newButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(deleteButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(refreshButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(saveButton)
                .addContainerGap())
            .addGroup(mainPanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
                    .addComponent(barcodeNoLabel)
                    .addComponent(typeLabel)
                    .addComponent(massLabel)
                    .addComponent(volumeLabel))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
            .addComponent(barcodeNoField,
javax.swing.GroupLayout.DEFAULT_SIZE, 315, Short.MAX_VALUE)
            .addComponent(typeField,   javax.swing.GroupLayout.DEFAULT_SIZE,
315, Short.MAX_VALUE)
            .addComponent(massField,  javax.swing.GroupLayout.DEFAULT_SIZE,
315, Short.MAX_VALUE)
            .addComponent(volumeField,
javax.swing.GroupLayout.DEFAULT_SIZE, 315, Short.MAX_VALUE))
          .addContainerGap())
        .addGroup(mainPanelLayout.createSequentialGroup()
          .addContainerGap()
          .addComponent(masterScrollPane,
javax.swing.GroupLayout.DEFAULT_SIZE, 380, Short.MAX_VALUE)
          .addContainerGap())
    );

    mainPanelLayout.linkSize(javax.swing.SwingConstants.HORIZONTAL,      new
java.awt.Component[] {deleteButton, newButton, refreshButton, saveButton});

    mainPanelLayout.setVerticalGroup(

mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
        .addGroup(mainPanelLayout.createSequentialGroup()
          .addContainerGap()
          .addComponent(masterScrollPane,
javax.swing.GroupLayout.DEFAULT_SIZE, 130, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
            .addComponent(barcodeNoLabel)
            .addComponent(barcodeNoField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
            .addComponent(typeLabel)
            .addComponent(typeField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                    .addComponent(massLabel)
                    .addComponent(massField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                    .addComponent(volumeLabel)
                    .addComponent(volumeField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                    .addComponent(saveButton)
                    .addComponent(refreshButton)
                    .addComponent(deleteButton)
                    .addComponent(newButton))
                .addContainerGap())
        );

        menuBar.setName("menuBar"); // NOI18N

        fileMenu.setText(resourceMap.getString("fileMenu.text")); // NOI18N
        fileMenu.setName("fileMenu"); // NOI18N

        newRecordMenuItem.setAction(actionMap.get("newRecord")); // NOI18N
        newRecordMenuItem.setName("newRecordMenuItem"); // NOI18N
        fileMenu.add(newRecordMenuItem);

        deleteRecordMenuItem.setAction(actionMap.get("deleteRecord")); // NOI18N
        deleteRecordMenuItem.setName("deleteRecordMenuItem"); // NOI18N
        fileMenu.add(deleteRecordMenuItem);

        jSeparator1.setName("jSeparator1"); // NOI18N
        fileMenu.add(jSeparator1);
        saveMenuItem.setAction(actionMap.get("save")); // NOI18N
        saveMenuItem.setName("saveMenuItem"); // NOI18N
        fileMenu.add(saveMenuItem);

        refreshMenuItem.setAction(actionMap.get("refresh")); // NOI18N
        refreshMenuItem.setName("refreshMenuItem"); // NOI18N
        fileMenu.add(refreshMenuItem);

        jSeparator2.setName("jSeparator2"); // NOI18N
        fileMenu.add(jSeparator2);

        exitMenuItem.setAction(actionMap.get("quit")); // NOI18N
        exitMenuItem.setName("exitMenuItem"); // NOI18N
        fileMenu.add(exitMenuItem);

        menuBar.add(fileMenu);

        helpMenu.setText(resourceMap.getString("helpMenu.text")); // NOI18N
        helpMenu.setName("helpMenu"); // NOI18N

        aboutMenuItem.setAction(actionMap.get("showAboutBox")); // NOI18N
        aboutMenuItem.setName("aboutMenuItem"); // NOI18N
        helpMenu.add(aboutMenuItem);

        menuBar.add(helpMenu);

        statusPanel.setName("statusPanel"); // NOI18N

        statusPanelSeparator.setName("statusPanelSeparator"); // NOI18N

        statusMessageLabel.setName("statusMessageLabel"); // NOI18N


statusAnimationLabel.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        statusAnimationLabel.setName("statusAnimationLabel"); // NOI18N

        progressBar.setName("progressBar"); // NOI18N

        javax.swing.GroupLayout          statusPanelLayout          =          new
javax.swing.GroupLayout(statusPanel);
        statusPanel.setLayout(statusPanelLayout);
        statusPanelLayout.setHorizontalGroup(

statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addComponent(statusPanelSeparator,
javax.swing.GroupLayout.DEFAULT_SIZE, 400, Short.MAX_VALUE)
            .addGroup(statusPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(statusMessageLabel)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,      226,
Short.MAX_VALUE)
            .addComponent(progressBar,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(statusAnimationLabel)
            .addContainerGap())
      );
      statusPanelLayout.setVerticalGroup(

statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
         .addGroup(statusPanelLayout.createSequentialGroup()
            .addComponent(statusPanelSeparator,
javax.swing.GroupLayout.PREFERRED_SIZE,                                     2,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
               .addComponent(statusMessageLabel)
               .addComponent(statusAnimationLabel)
               .addComponent(progressBar,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(3, 3, 3))
      );

      setComponent(mainPanel);
      setMenuBar(menuBar);
      setStatusBar(statusPanel);

      bindingGroup.bind();
   }// </editor-fold>

   // Variables declaration - do not modify
   private javax.swing.JTextField barcodeNoField;
   private javax.swing.JLabel barcodeNoLabel;
   private javax.swing.JButton deleteButton;
   private javax.persistence.EntityManager entityManager;
   private javax.swing.JSeparator jSeparator1;
   private javax.swing.JSeparator jSeparator2;
   private java.util.List<moremoney.Lessbag> list;
```

```
    private javax.swing.JPanel mainPanel;
    private javax.swing.JTextField massField;
    private javax.swing.JLabel massLabel;
    private javax.swing.JScrollPane masterScrollPane;
    private javax.swing.JTable masterTable;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JButton newButton;
    private javax.swing.JProgressBar progressBar;
    private javax.persistence.Query query;
    private javax.swing.JButton refreshButton;
    private javax.swing.JButton saveButton;
    private javax.swing.JLabel statusAnimationLabel;
    private javax.swing.JLabel statusMessageLabel;
    private javax.swing.JPanel statusPanel;
    private javax.swing.JTextField typeField;
    private javax.swing.JLabel typeLabel;
    private javax.swing.JTextField volumeField;
    private javax.swing.JLabel volumeLabel;
    private org.jdesktop.beansbinding.BindingGroup bindingGroup;
    // End of variables declaration

    private final Timer messageTimer;
    private final Timer busyIconTimer;
    private final Icon idleIcon;
    private final Icon[] busyIcons = new Icon[15];
    private int busyIconIndex = 0;

    private JDialog aboutBox;

    private boolean saveNeeded;
}
```