

[REDACTED]
Houston Community College

ENGR 1201

November 26th, 2019

NEW VENDING MACHINES FOR HCC CENTRAL STUDENTS AND STAFFS

Author: [REDACTED]

Email: [REDACTED]

Tel: [REDACTED]

Research Paper for ENGR 1201

Introduction to Engineering Fall 2019

I) Abstract

Earlier this fall, some vending machines on HCC Central Campus have broken down. Because of student's and staff's level of comfort, I hereby attempting to design a vending machine for everyone to use. I will mainly focus on the design of a vending machine's control system and inventory tracking software instead of its physical parts.

Keywords: *vending machines, HCC Central, control system*

II) Introduction:

A vending machine is an automated commercial tool that is used to sell products without the selling having to show up. The first vending machines are thought to be invented in England, around the 1880s. However, this is not totally untrue if we consider the vague proof about an invention of an ancient mathematician to sell holy water. Nowadays, these machines sell almost everything, from small packs of snack, coke to luxurious electronic devices like Xbox gaming consoles.

In the U.S, there are around 4.6 million vending machines supplying mostly food, drinks and cigarette. However, the anti-smoking campaign in early 1990s has drastically declined the amount of cigarette vending machines around the country. The countries with most developed automation industry, Japan, also have the most advanced vending machines. The Japanese machines can make you a bowl of hot Ramen with around 600-1200 yen (\$6-\$11) in under a minute. They also sell all types of magazines, toys and even clothes sometimes.

There are two different types of vending machines with different operating mechanics. In a mechanical machine, a dispenser is normally activated by a linkage system that is locked by a pin. The pin will be pushed over when the right amount of coins is input, then allowing a selective rod to travel the full distance and trigger the linkage. There are also electronic vending machines, which are more widely used nowadays, they can accept both coins and bills. Instead of linkage, the machines contain several switches that are activated when coins and bills are received. Unlike mechanical machines, when a product is selected, every other product would be prevented from falling down the rack and the machine starts the finalization.

One of the most significant parts of the machine is its ability to differentiate real and fake money. Coins would be elaborately examined by measuring weight, diameter, and thickness.

When the coin is put in, it will roll through a rack that captures its velocity. This method can be used to elevate the coin's alloy to see if it is counterfeit or not. The bills are scanned for marbling, printed notes and anti-counterfeit stamps.

III) Control system:

1) Requirements

When it comes to dealing with money recognition for a machine it is always hard. To me, the best part about a machine is it does exactly what I tell it to do. Also, the worst part about a machine is it does exactly what I tell it to do.

Carefully listed out the problems that might be caused in a machine's operation, the machine must be able to:

1. When the currency slot is invaded, sensors have to detect the correct value of the money and communicate with the system.
2. The system is able to receive and recognize the data.
3. The key panel is activated and keep track of the inventory, waiting for the customer to choose.
4. Displaying information about the chosen product whether if they have been sold out.
5. Read the balance, if the amount is insufficient after a certain time, refund will be generated. If the amount is filled, the chosen product will be driven to the recipient.
6. Finalizing, zero the money to the buyer and complete the transaction.

2) Design

Let assume that I am a terrible businessman who only sells 5 types of drink: Coca-Cola, Soda, Milk, Tea, and Coffee. The price tags are as follows: \$1.25, \$1, \$2, \$1 and \$1.50.

The machine can recognize five-dollar bill, one-dollar bill, dimes, nickels, and quarters (from a dime to \$5. It will not accept cents or anything higher than \$10.

3) Operation:

- The customer input money.
- The money is recognized, forming a signal.
- The signal is transmitted to the PLC.
- The money is counted and displayed at the same time.
- When the amount is more than \$1, Soda and Tea lights turn on and remain with higher values.
- When the amount is more than \$1.25, Coca-Cola light turns on and remains with higher values
- When the amount is more than \$1.50, Coffee light turns on and remains with higher values
- When the amount is more than \$2, Milk light turns on.
- Through the PLC, the shipment parts drive the products out to the receiving rack, sending out goods.
- If there is balance the machine will process a refund.

Figure 1 below illustrates how the system works

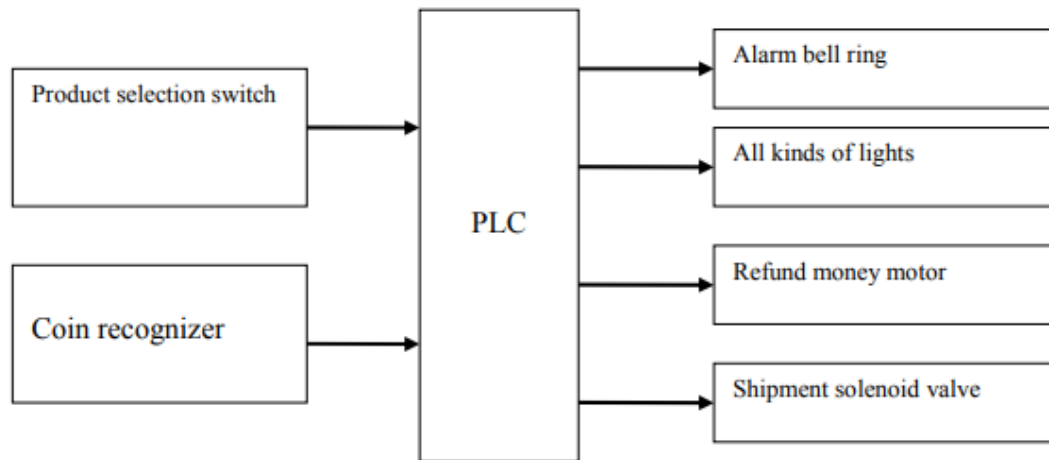


Figure 1. (Operation logistics)

4) Initial codes:

The initial codes, using Java can be scripted as following:

```
package vending Machine;
```

```
import java.util.Map;
```

```
import java.util.Set;
```

```
public class VendingMachine {
```

```
    private int collectedCash;
```

```
    private State state;
```

```
    private Map<String, Set<String>> productCodeItemMap;
```

```

private Map<String, Integer> productCodePriceMap;

public VendingMachine() {
    this.collectedCash = 0;
    this.state = State.READY;
}

public void collectCash(int cash) {
    switch (state) {
        case READY:
            handleCollectCash(cash);
            break;
        case DISPENSE_CHANGE:
            throw new RuntimeException("Dispensing change. Unable to collect cash");
        case DISPENSE_ITEM:
            throw new RuntimeException("Dispensing item. Unable to collect cash");
        case TRANSACTION_CANCELLED:
            throw new RuntimeException("Transaction cancelled. Unable to collect cash");
    }
}

public void dispenseItem(String productCode) {
    switch (state) {
        case READY:
            throw new RuntimeException("Unable to dispense Item. Cash not collected");
        case DISPENSE_CHANGE:
            handleDispenseChange(productCode);
            break;
        case DISPENSE_ITEM:
            handleDispenseItem(productCode);
            break;
        case TRANSACTION_CANCELLED:
            throw new RuntimeException("Transaction cancelled. Unable to dispense Item");
    }
}

private void handleCollectCash(int cash) {
    this.collectedCash += cash;
}

private void handleDispenseItem(String productCode) {
    // logic to dispense item
}

private void handleDispenseChange(String productCode) {
    // logic to dispense Change
}
}

```

5) PLC logistic pattern:

The PLC itself will need some logistics to work, as it is Programmable doesn't mean it is going to program itself. The pattern will simply restate the ideas from the operation, now with the actions that the system has to conduct then accumulate it into a systematic description.

We start with the very first step to start the machine: inputting the coins. At this stage, the system will recognize that the money rack is invaded and show the amount on the screen while counting the money.

When the amount of money for each item is reached, the light on the button for that very item will bright up. This is taken after the counting is conducted of course.

After the customer chooses the item, the system has to subtract any remaining balance and refund it as well as dispense the item to the customer until the balance is zero out.

The PLC working logistics can be described as *Figure 2* shows:

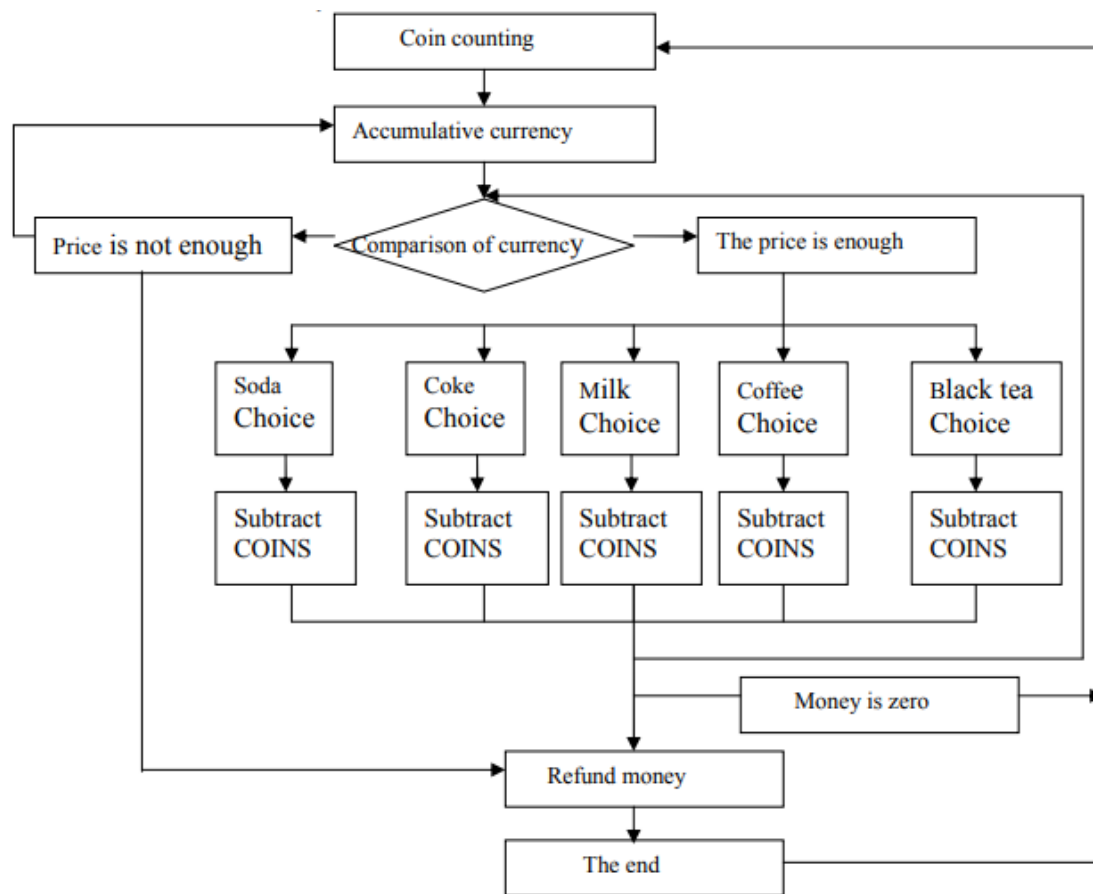


Figure 2 (PLC's operation logistics)

Adding the PLC's operation to the initial codes, the new codes will follow these four steps:

collectCash, dispenseCash, dispenseItem, and cancelTransaction

```
package vending Machine;
```

```
public interface State {
    public void collectCash(int cash);
    public void dispenseChange(String productCode);
    public void dispenseItem(String productCode);
    public void cancelTransaction();
}
```

6) Modified code:

-Dispense Change :

```
1 package vendingMachine;
2
3 public class DispenseChange implements State {
4     private VendingMachine vendingMachine;
5
6     DispenseChange(VendingMachine vendingMachine) {
7         this.vendingMachine = vendingMachine;
8     }
9
10    @Override
11    public void collectCash(int cash) {
12        throw new RuntimeException("Dispensing Change. Unable to collect cash");
13    }
14
15    @Override
16    public void dispenseChange(String productCode) {
17        int change = this.vendingMachine.calculateChange(productCode);
18        System.out.println("Change of " + change + " returned");
19        this.vendingMachine.setState(new DispenseItem(this.vendingMachine));
20        this.vendingMachine.dispenseItem(productCode);
21    }
22
23    @Override
24    public void dispenseItem(String productCode) {
25        throw new RuntimeException("Dispensing change. Unable to dispense Item");
26    }
27
28    @Override
29    public void cancelTransaction() {
30        throw new RuntimeException("Dispensing change. Unable to cancel transaction");
31    }
32 }
33
```

-Dispense Item:

```
1 package vendingMachine;
2
3 public class DispenseItem implements State {
4     private final VendingMachine vendingMachine;
5
6     DispenseItem(VendingMachine vendingMachine) {
7         this.vendingMachine = vendingMachine;
8     }
9
10    @Override
11    public void collectCash(int cash) {
12        throw new RuntimeException("Dispensing item.Unable to collect cash");
13    }
14
15    @Override
16    public void dispenseChange(String productCode) {
17        throw new RuntimeException("Dispensing item.Unable to dispense change");
18    }
19
20    @Override
21    public void dispenseItem(String productCode) {
22        vendingMachine.removeProduct(productCode);
23        System.out.println("Dispensing item " + productCode);
24        vendingMachine.setState(new Ready(this.vendingMachine));
25    }
26
27    @Override
28    public void cancelTransaction() {
29        throw new RuntimeException("Dispensing item.Unable to cancel the Transaction");
30    }
31 }
32
```

--Collect Cash

```
1 package vendingMachine;
2
3 public class Ready implements State {
4     private VendingMachine vendingMachine;
5
6     Ready(VendingMachine vendingMachine) {
7         this.vendingMachine = vendingMachine;
8     }
9
10    @Override
11    public void collectCash(int cash) {
12        this.vendingMachine.addCollectedCash(cash);
13    }
14
15    @Override
16    public void dispenseChange(String productCode) {
17        this.vendingMachine.setState(new DispenseChange(this.vendingMachine));
18        this.vendingMachine.dispenseChange(productCode);
19    }
20
21    @Override
22    public void dispenseItem(String productCode) {
23        throw new RuntimeException("Txn not initiated.Unable to dispense item");
24    }
25
26    @Override
27    public void cancelTransaction() {
28        this.vendingMachine.setState(new TransactionCancelled(vendingMachine));
29        this.vendingMachine.cancelTransaction();
30    }
31 }
32
```

-Transaction Cancelled

```
1 package vendingMachine;
2
3 public class TransactionCancelled implements State {
4     private VendingMachine vendingMachine;
5
6     TransactionCancelled(VendingMachine vendingMachine) {
7         this.vendingMachine = vendingMachine;
8     }
9
10    @Override
11    public void collectCash(int cash) {
12        throw new RuntimeException("Unable to collect cash in a cancelled transaction");
13    }
14
15    @Override
16    public void dispenseChange(String productCode) {
17        throw new RuntimeException("Unable to dispense change in a cancelled transaction");
18    }
19
20    @Override
21    public void dispenseItem(String productCode) {
22        throw new RuntimeException("Unable to dispense item in a cancelled transaction");
23    }
24
25    @Override
26    public void cancelTransaction() {
27        System.out.println("Returning collected cash " + vendingMachine.getCollectedCash());
28        vendingMachine.setCollectedCash(0);
29        vendingMachine.setState(new Ready(vendingMachine));
30    }
31 }
32
```

IV) Conclusion:

My design might have problems as each particular states have to stay aware the whole time to conduct the next states and so on, and those states must be heavily coded to avoid chain interruption during the process. However I am also confident that my design can be very flexible, if a new state want to enter the process there wouldn't have to be a new code block to change it and modification is always open.

All in all, vending machine is a quick and convenient way to buy snacks for HCC students and staffs. I do think replacing the old broken vending machines in Science Building for everyone to use is good idea. The location is also important, I notice that there is not a single machine in the San Jacinto building and there are three of them in the other one. I recommend put at least one of the vending machines somewhere in the San Jacinto building to reduce the walking for some people.

V) Reference

- [1] He Yong Yi. A Control System of Material Handling in FMS. Journal of shanghai university. Vol.1.No.1.1997.
- [2] Ren Sheng-le. Development of PLC-based Tension Control System. Chinese Journal of Aeronautics 20 (2007) 266-271.
- [3] R. Hatch, "The synergism of GPS code and carrier measurements" International Geodetic Symposium on Satellite Doppler Positioning, vol. 1, pp. 1213-1231, May 1983.
- [4] K. Bore, D. M. Akos, N. Bertelsen, A software-defined GPS, and Galileo receiver. Birkhauser, pp. 109-123. 2007.
- [5] Altera Corporation. Video and Image Processing Suite User Guide[Z].2010.7.
- [6] O. Montenbruck, T. V. Helleputte, K. Remco, "Reduced dynamic orbit determination using GPS code and carrier measurements," Aerospace Science and Technology vol. 9, pp. 261-271, April 2005. [Z] Unitrode Application Note.1999.