**DESIGN OF VENDING MACHINE**

**USING LABVIEW**

Course Title: Skilling ( LabVIEW/Multisim)

Course code: 17TS401

submitted by

Y.HESHMA CHOWDARY 170040964

YALAMANCHILI SUNDEEP 170040970

IDARAPALLI SAI SOURAV 170041010

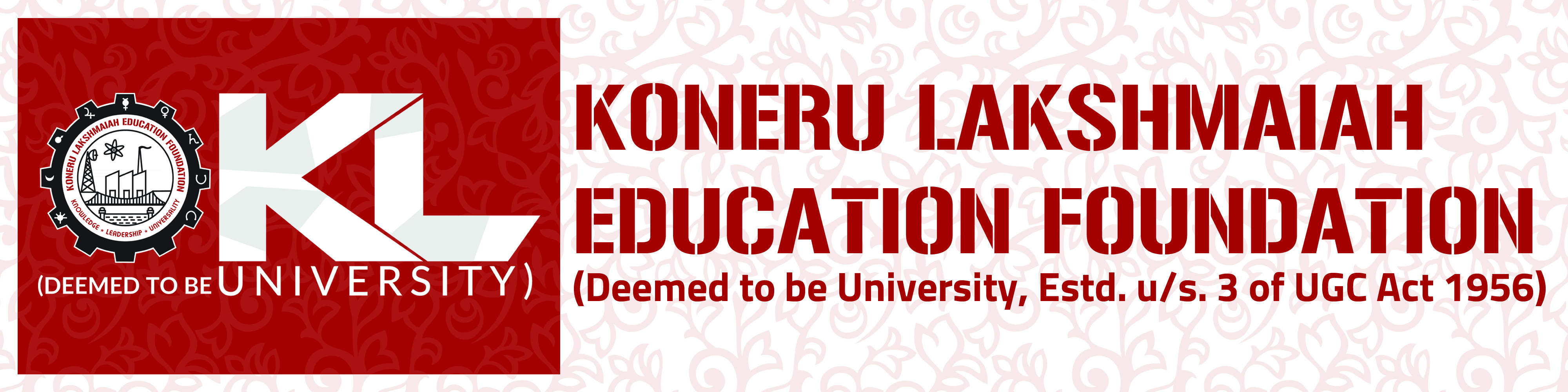
Submitted to

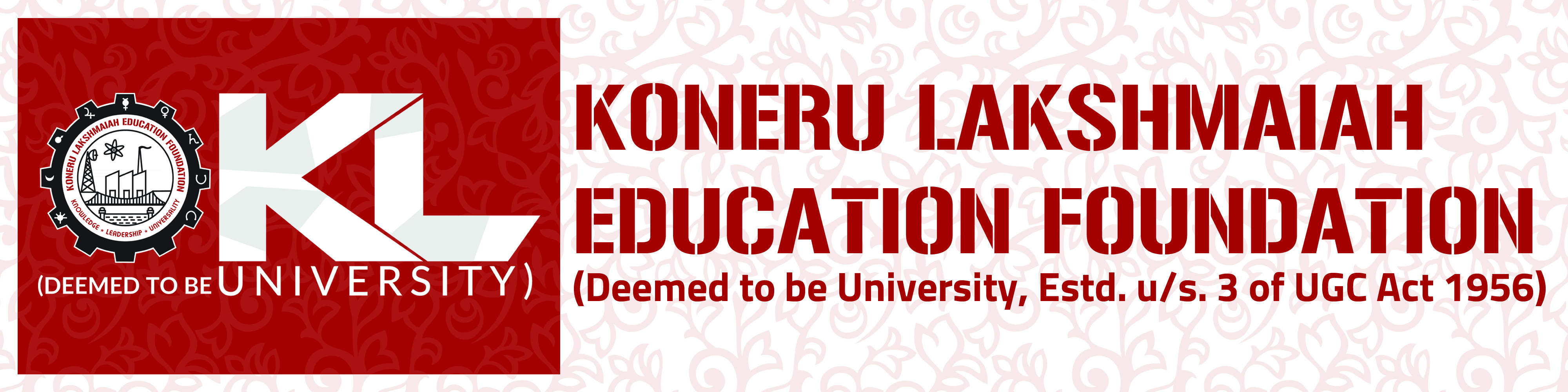
K.SIVA LATHA

S.VAMSEE KRISHNA

Asst.Professor

Department of ECE







**K L E F**

**DEPARTMENT OF ECE**

**SKILLING (17 TS 401)**

**CERTIFICATE**

This is to certify that Y.HESHMA,Y.SUNDEEP,I.SOURAV bearing Univ. Regd. No. 170040964, 170040970, 170041010 respectively of section S-02 studying II B.Tech in ECE has satisfactorily completed SKILL programme project in the semester III during the academic year 2018 – 2019.

**Signature of Course Instructor Signature of the Course Coordinator**

**Signature of HOD**

**ACKNOWLEDGEMENT**

It is great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satyanarayana**, for giving the opportunity and platform with facilities in accomplishing the project based laboratory report.

I express the sincere gratitude to our principal **Dr. A. Anand Kumar**, Dean Academics **Dr.Pranvir S Satvat** and HOD **Dr.K.CH.Sri** **Kavya** for their administration towards our academic growth.

I express sincere gratitude to our Coordinator **Mr. Madhav Prasad** for his leadership and constant motivation provided in successful completion of our academic semester.

I express my sincere thanks to our project supervisor **Ms.K.Siva Latha** for his/her novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

Y.HESHMA 170040964

Y.SUNDEEP 170040970

I.SOURAV 170041010

**ABSTRACT**

The vending machine is one of the fundamental architectures NI LabVIEW developers frequently use to build applications quickly. Developers use state machines in applications where distinguishable states exist. Each state can lead to one or multiple states and can end the process flow. A state machine relies on user input or in-state calculation to determine which state to go to next. Many applications require an “initialize” state followed by a default state, where you can perform many different actions. These actions depend on previous and current inputs as well as states. You can use a “shutdown” state to perform cleanup actions.

**TABLE OF CONTENTS**

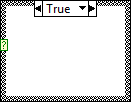
|  |  |
| --- | --- |
| CHAPTER NO TITLE | PAGE NO. |
| Abstract | I |
| List of symbols and functions | 6 |
| 1.1.1 Introduction | 11 |
| 1.1.2 Flow Diagram | 17 |
| 1.2.1 Technical Description of The  Project | 18 |
| 1.2.2 Experimental Results | 20 |
| 1.2.3 Conclusion and Futurescope | 24 |
| 1.2.4 Reference | 26 |

**LIST OF SYMBOLS AND FUNCTIONS**

While loop:

: The While Loop structure in LabView is the same as a do-while loop that is used in other languages. and requires a termination condition. The while loop is a structure that will continuously execute. the functions inside it until the conditions for termination are reached

Case structure:

 A Case structure has two or more **subdiagrams**, or cases. Only one **subdiagram** is visible at a time, and the structure executes only one case at a time. An input value determines which **subdiagram** executes.

Enum constant:

: An enum can only be a data type of integer. It consists of consecutive values from zero to n-1 (number of elements in enum). LabVIEW replaces "B","D","E" with "B...E" because it is using the integer values from your enum to select the case, not the labels. As you add cases to your long term code, add the new items to the end of the list.

Wait:

 After you create a loop, you can place a **wait** VI inside of the loop to control how long it waits before performing the next iteration. There are two basic **wait functions in LabVIEW**: **Wait** (ms) and**Wait** Until Next ms Multiple.

Stop:

 Stops the VI in which it executes, just as if you clicked the Abort Execution button on the toolbar. Before you call this function with a TRUE **input**, be sure to complete all final tasks for the VI first, such as closing files, setting safe values for devices being controlled, and so on.

select:

Returns the value wired to the **t** input or **f** input, depending on the value of **s**. If **s** is TRUE, this function returns the value wired to **t**. If **s** is FALSE, this function returns the value wired to **f**.

The connector pane displays the default data types for this polymorphic function.

Cluster:

  A **cluster** is similar to a record or a struct in text-based programming languages. Similar to arrays, a **cluster** is either a control or an indicator and cannot contain a mixture of controls and indicators.

For loop:

 It runs from i=0 to i= n times.

Add:

Computes the sum of two inputs.

Subtract:

Computes the difference of two inputs.

Event structure:

Event Structures have one or more sub diagrams, or event cases, exactly one which executes when the structures executes.



OK button : When button is presssed, exit loop and continue with code. You must have the wire from the loop to the rest of code structure. It can be any wire. This will prevent the code from running until the loop is done. Be sure to add the delay in the loop to prevent using 100% CPU time while waiting for the user to press the RUN button (OK Button).

Boolean : LabVIEW stores Boolean data as 8-bit values. A Boolean can be used in LabVIEW to represent a 0 or 1, or a TRUE or FALSE. If the 8-bit value is zero, the Boolean value is FALSE. Any nonzero value represents TRUE

And Gate:

 It performs and operation as per the truth table

String indicator

Path controls and indicators work similarly to string controls and indicators, but LabVIEW formats the path using the standard syntax for the platform you are using. Invalid Paths. If a function that returns a path fails, the function returns <Not A Path> in the indicator

. 

While loop : The While Loop structure in LabView is the same as a do-while loop that is used in other languages. and requires a termination condition. The while loop is a structure that will continuously execute. the functions inside it until the conditions for termination are reached.

String constant : A tab constant is actually a single character. LabVIEW strings more or less show exactly what it is which is occupying the space of a single ASCII character, but it is non-printable, so it looks like a space.

OR gate:

* It performs or operation as per the truth table.

Shift register: It allows looping structures, such as the For and While loop, to carry over values from one iteration into the next. This can be extremely powerful in allowing the developer to build up arrays of information or retain state information about a program running in a loop.

**INTRODUCTION**

A vending machine is an automated machine that provides items such as [snacks](https://en.wikipedia.org/wiki/Snacks), [beverages](https://en.wikipedia.org/wiki/Beverage), [cigarettes](https://en.wikipedia.org/wiki/Cigarettes) and [lottery tickets](https://en.wikipedia.org/wiki/Lottery_ticket) to consumers after money, a [credit card](https://en.wikipedia.org/wiki/Credit_card), or specially designed card is inserted into the machine.[[1]](https://en.wikipedia.org/wiki/Vending_machine#cite_note-1) The first modern vending machines were developed in England in the early 1880s and dispensed [postcards](https://en.wikipedia.org/wiki/Postcard). Vending machines exist in many countries, and in more recent times, specialized vending machines that provide less common products compared to traditional vending machine items have been created and provided to consumers.

The first modern coin-operated vending machines were introduced in [London](https://en.wikipedia.org/wiki/London), [England](https://en.wikipedia.org/wiki/England) in the early 1880s, dispensing [postcards](https://en.wikipedia.org/wiki/Postcard). The machine was invented by Percival Everitt in 1883 and soon became a widespread feature at railway stations and post offices, dispensing [envelopes](https://en.wikipedia.org/wiki/Envelopes), [postcards](https://en.wikipedia.org/wiki/Postcard), and [notepaper](https://en.wikipedia.org/wiki/Notepaper). The Sweetmeat Automatic Delivery Company was founded in 1887 in England as the first company to deal primarily with the installation and maintenance of vending machines. In 1893, [Stollwerck](https://en.wikipedia.org/wiki/Stollwerck" \o "Stollwerck), a German chocolate manufacturer, was selling its chocolate in 15,000 vending machines. It set up separate companies in various territories to manufacture vending machines to sell not just chocolate, but cigarettes, matches, chewing gum and soap products.[[6]](https://en.wikipedia.org/wiki/Vending_machine#cite_note-6)

The first vending machine in the U.S. was built in 1888 by the [Thomas Adams Gum Company](https://en.wikipedia.org/wiki/Thomas_Adams_Gum_Company),[[7]](https://en.wikipedia.org/wiki/Vending_machine#cite_note-NAMA_History-7) selling gum on New York City train platforms. The idea of adding games to these machines as a further incentive to buy came in 1897 when the Pulver Manufacturing Company added small figures, which would move around whenever somebody bought some gum from their machines. This idea spawned a whole new type of mechanical device known as the "trade stimulators".

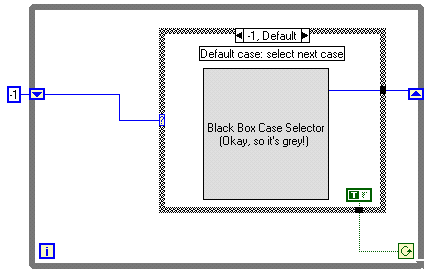
The state machine is one of the fundamental architectures NI LabVIEW developers frequently use to build applications quickly. Developers use state machines in applications where distinguishable states exist. Each state can lead to one or multiple states and can end the process flow.A finite-state machine (FSM) or finite-state automaton (FSA, plural: *automata*), finite automaton, or simply a state machine, is a mathematical [model of computation](https://en.wikipedia.org/wiki/Model_of_computation). It is an [abstract machine](https://en.wikipedia.org/wiki/Abstract_machine) that can be in exactly one of a finite number of [*states*](https://en.wikipedia.org/wiki/State_(computer_science)) at any given time. The FSM can change from one state to another in response to some external [inputs](https://en.wikipedia.org/wiki/Input_(computer_science)); the change from one state to another is called a *transition*. An FSM is defined by a list of its states, its initial state, and the conditions for each transition. Finite state machines are of two types - [deterministic finite state machines](https://en.wikipedia.org/wiki/Deterministic_finite_automaton) and [non-deterministic finite state machines](https://en.wikipedia.org/wiki/Nondeterministic_finite_automaton).[[1]](https://en.wikipedia.org/wiki/Finite-state_machine#cite_note-1) A deterministic finite-state machine can be constructed equivalent to any non-deterministic one.

The behavior of state machines can be observed in many devices in modern society that perform a predetermined sequence of actions depending on a sequence of events with which they are presented. Simple examples are [vending machines](https://en.wikipedia.org/wiki/Vending_machine), which dispense products when the proper combination of coins is deposited, [elevators](https://en.wikipedia.org/wiki/Elevator), whose sequence of stops is determined by the floors requested by riders, [traffic lights](https://en.wikipedia.org/wiki/Traffic_light), which change sequence when cars are waiting, and [combination locks](https://en.wikipedia.org/wiki/Combination_lock), which require the input of combination numbers in the proper order.

The finite state machine has less computational power than some other models of computation such as the [Turing machine](https://en.wikipedia.org/wiki/Turing_machine).[[2]](https://en.wikipedia.org/wiki/Finite-state_machine#cite_note-Belzer-2)The computational power distinction means there are computational tasks that a Turing machine can do but a FSM cannot. This is because a FSM's [memory](https://en.wikipedia.org/wiki/Computer_memory) is limited by the number of states it has. FSMs are studied in the more general field of [automata theory](https://en.wikipedia.org/wiki/Automata_theory).

A state machine relies on user input or in-state calculation to determine which state to go to next. Many applications require an “initialize” state followed by a default state, where you can perform many different actions. These actions depend on previous and current inputs as well as states. You can use a “shutdown” state to perform cleanup actions. In LabVIEW software, you can create a basic state machine with a while loop, a shift register, a case statement, and some form of case selector (case selectors are discussed in a later section). The while loop is the main program loop, which executes until the conditions for exiting the program are met.

The while loop’s main responsibility is to call the case selector and then execute the appropriate case. The shift register keeps track of which case should execute next. Finally, each case of the case statement contains the action for one specific use action. Often the default case is used as the place to check the case selector (in other words, if the user did nothing, check again to see if he has done something yet).



FRONT PANEL:

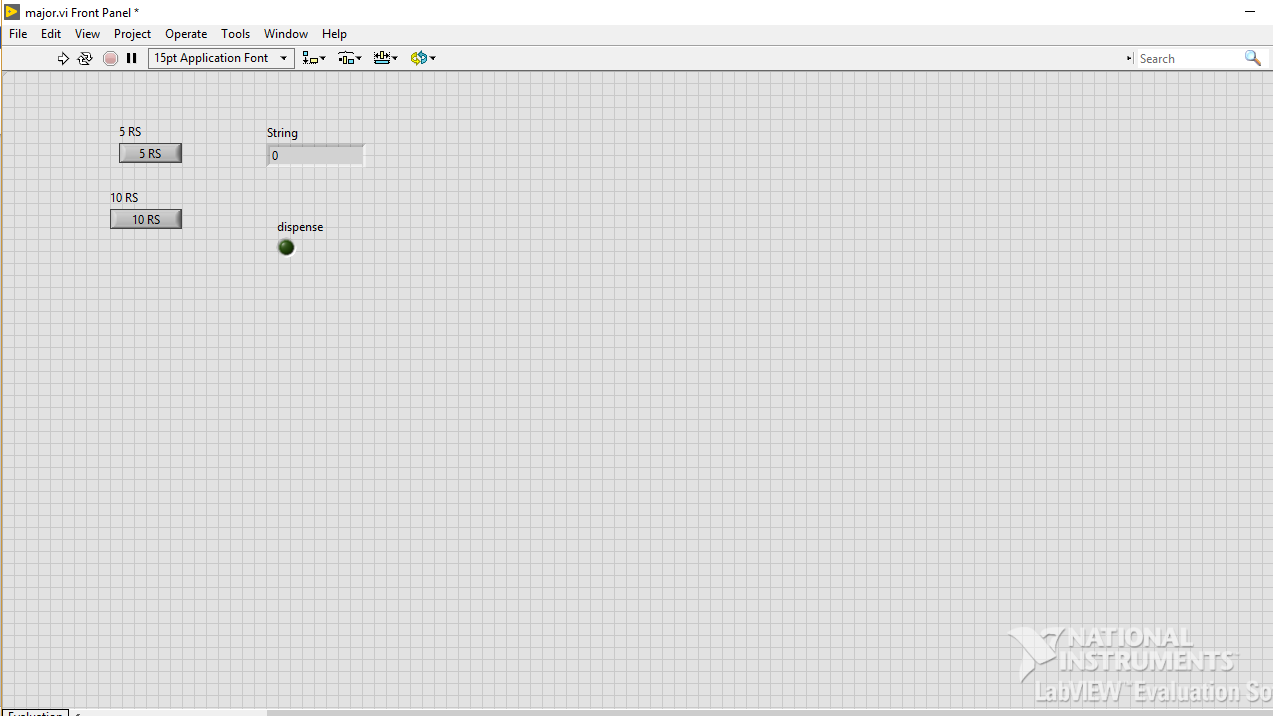


Figure 1.1 front panel

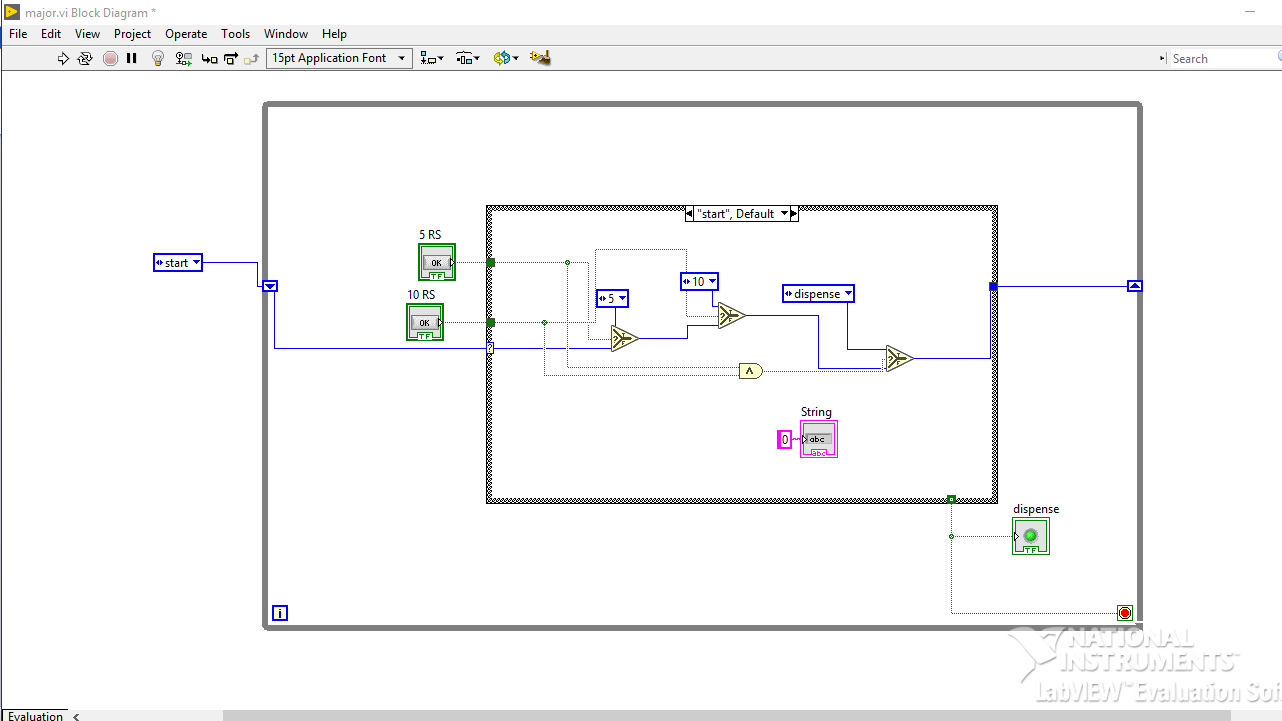
BLOCK DIAGRAM 

Figure 1.2 block diagram for default case

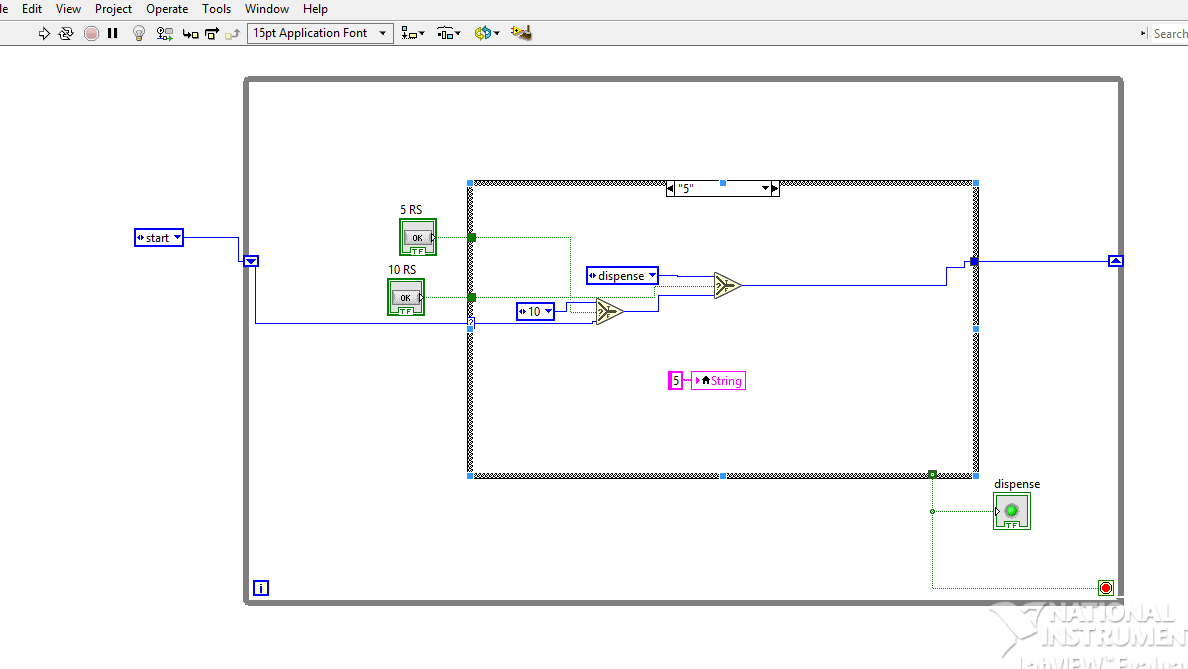


Figure 1.1 for 5 rupees case

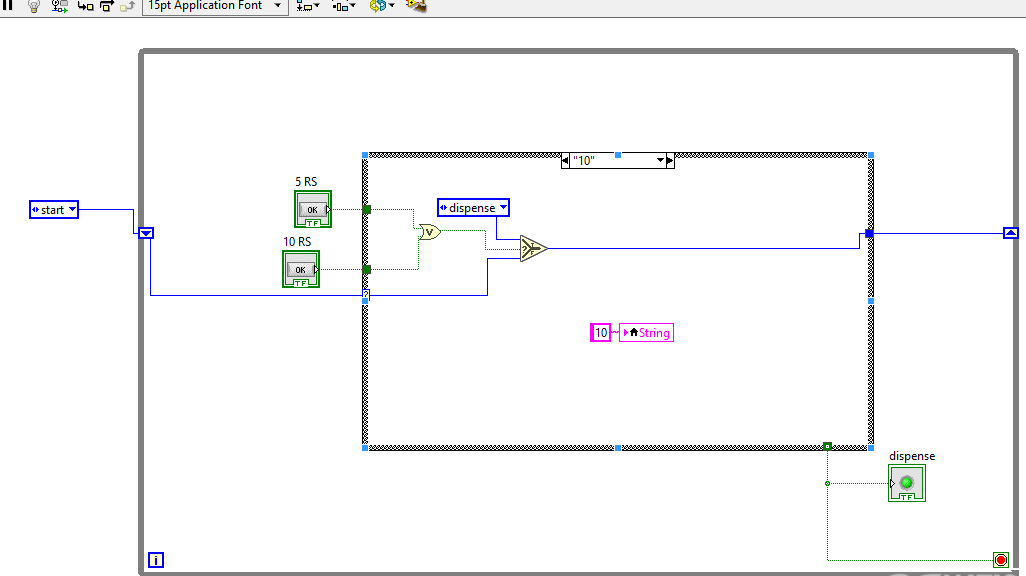


Figure 1.3 for 10 rupees case

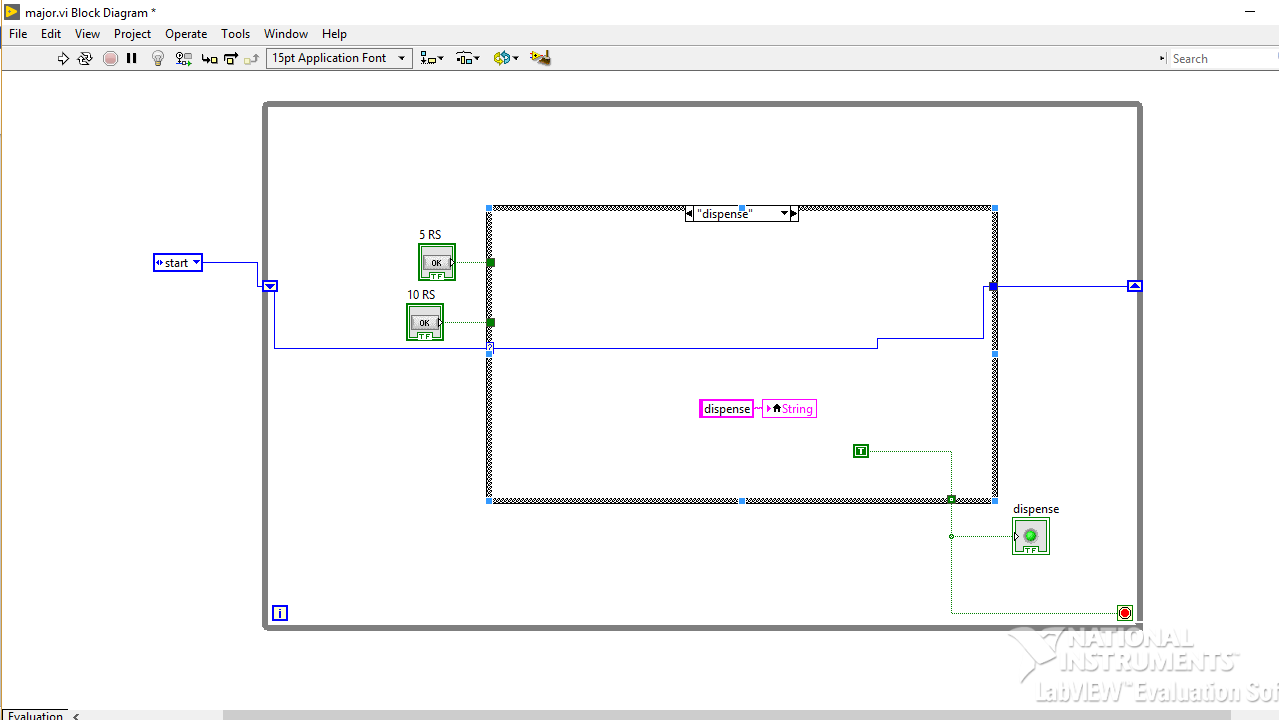


Figure 1.4 for despence case

**FLOW DIAGRAM:**

Model flow digram

**TECHNICAL DESCRIPTION OF THE PROJECT**

Using the state diagram above, create a state machine.

1. Create a new blank VI.

2. on the front panel place:

• Two text buttons named “5rs” and “10rs”

• Text indicator named “Money Deposited”

• Boolean indicator named “Dispense coke”

3. Place a while loop on the block diagram.

4. Place a case structure in the while loop.

5. Create a shift register on the while loop.

6. Create an Enum and wire it to the shift register to initialize it. Right-click the Enum, select Edit Items, and add the following “states”:

• Start

• 5 cents

• 10 cents

• Dispense

8. Wire the shift register to the conditional input of the case structure.

Imagine a vending machine that accepts combinations of nickels and dimes to get a coke. The cost of a coke is 15 cents and the machine does not return change. First, establish the states that the vending machine might be in:

Start: No money inserted

5 cents

10 cents

Done: 15 cents

From these lists, you see that there are four states and three possible paths from each state. You need to depict which states are connected by which paths. For example, when the vending machine is in the initial start state, the total change inserted is 0 cents. When a nickel is inserted, the vending machine must go to the 5 cent state. Therefore, the start state leads to

the 5 cent state by the nickel path.

9. Right-click the case box at the top of the case structure and select Add Case for Every Value.

10. Wire the different cases as depicted in the following figures.

**EXPERIMENTAL RESULT**

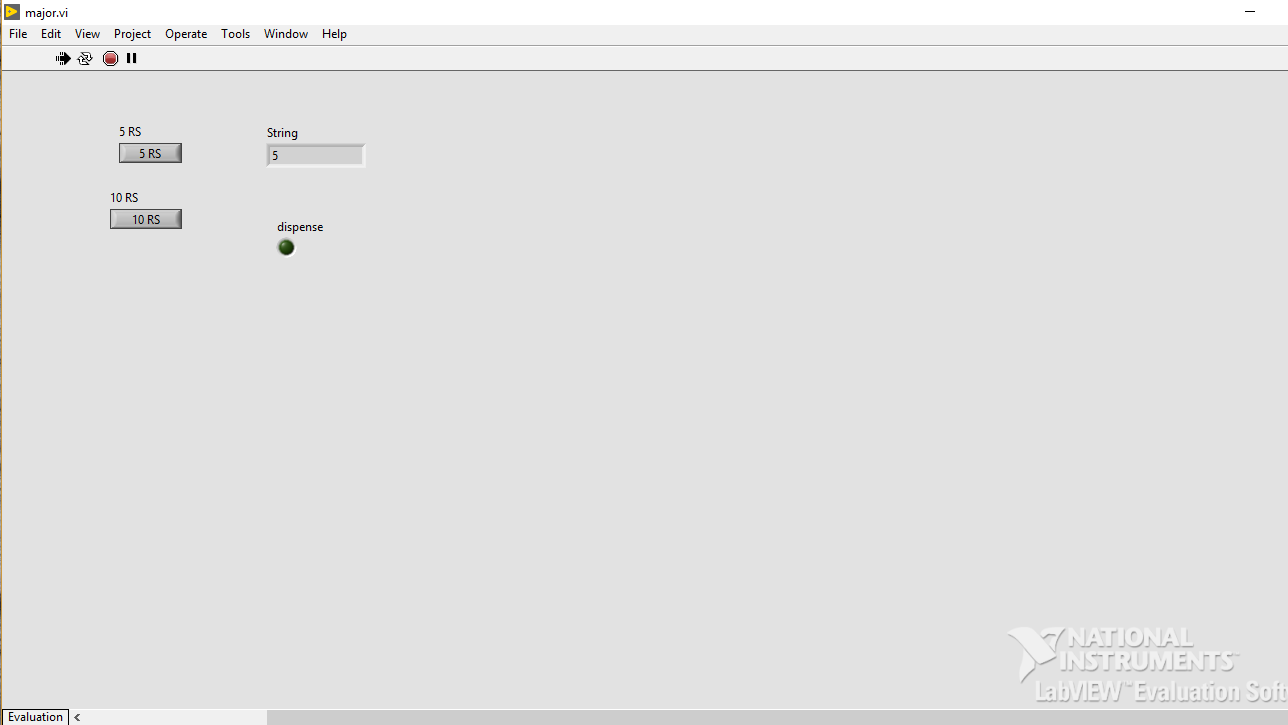
****

Figure 2.1 for 5 rupees

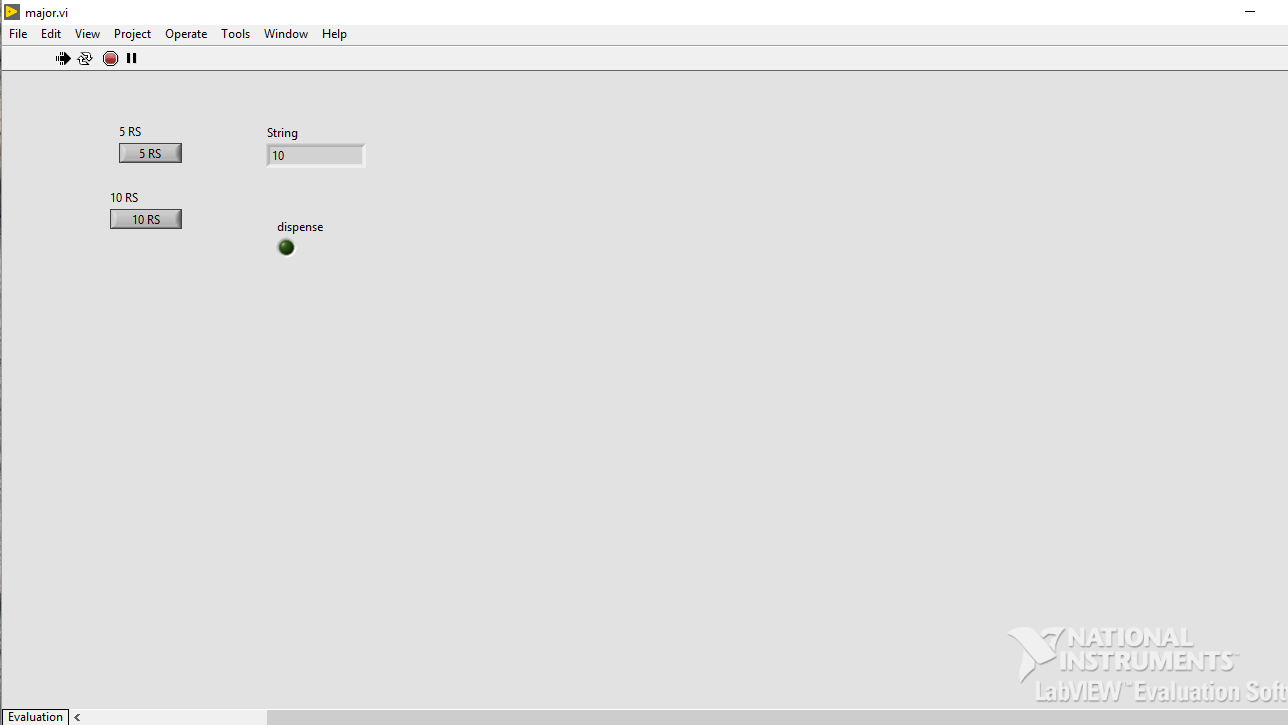
****

Figure 2.2 for 10 rupees

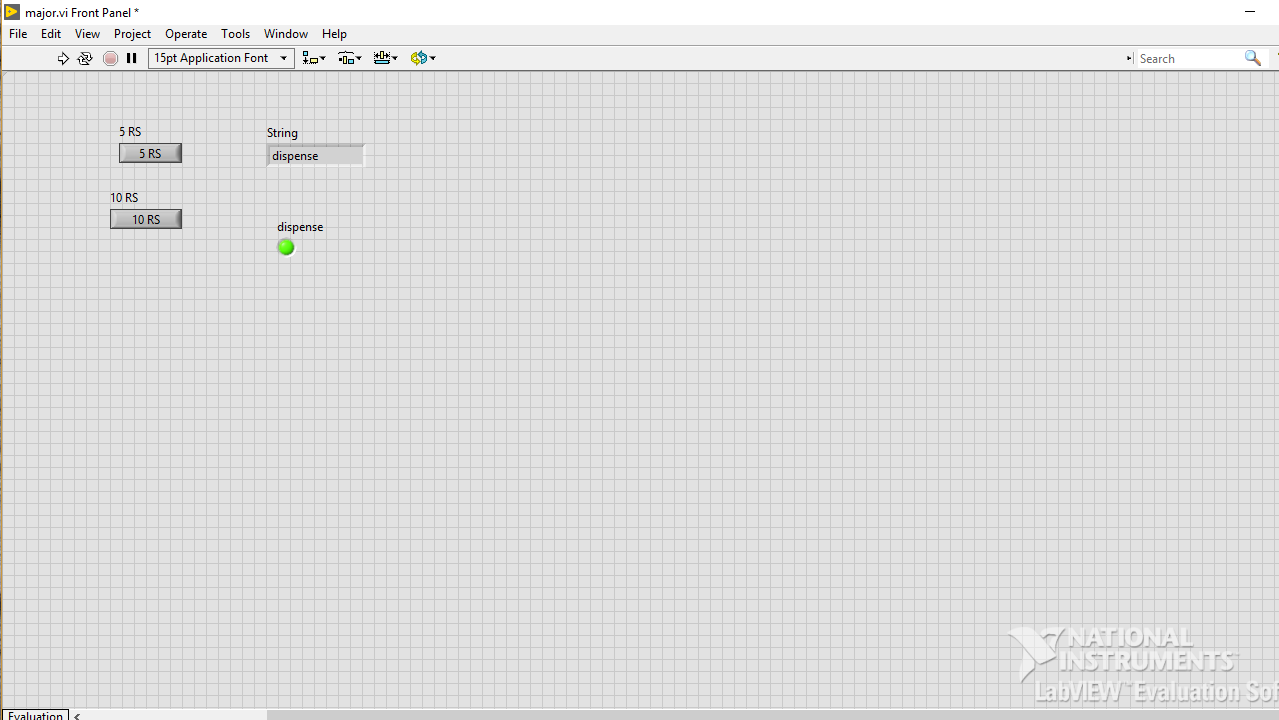
****

Figure 2.3 for 15 rupees and chocolate dispensed

**CONCLUSION AND FUTURESCOPE**

Imagine a vending machine that accepts combinations of nickels and dimes to get a coke. The cost of a coke is 15 cents and the machine does not return change. First, establish the states that the vending machine might be in:

Start: No money inserted

5 cents

10 cents

Done: 15 cents

Now think about the possible ways or paths that the vending machine can take to get into these states. Don’t forget to consider how the vending machine starts in the initial state.

• Nickel is inserted

• Dime is inserted

• Default: Nothing is inserted

From these lists, you see that there are four states and three possible paths from each state. You need to depict which states are connected by which paths. For example, when the vending machine is in the initial start state, the total change inserted is 0 cents. When a nickel is inserted, the vending machine must go to the 5 cent state. Therefore, the start state leads to

the 5 cent state by the nickel path. By considering all states and paths, you can create a state diagram for the vending machine.

In 1959, the Vendo Co. unveiled a new "million-dollar line" of Coca-Cola vending machines. Highlights included such futuristic details as a lit sign for "greater-selling impact" and a new fully pivoting hinge that let the door leading to the soda open 180 degrees. The new designs, immortalized in the short promotional film The Refreshing Look, represented the pinnacle of vending-machine technology.

Almost five decades later, Coke machines are more ubiquitous than ever. But the future of automatic merchandising, as it's known in the trade, isn't in mass-produced cans of soda. Rather, it's in products and services created on demand and custom-tailored to consumers' preferences.People feel like they're entitled to something that's made just for them," says James Bickers, editor of SelfServiceWorld magazine. Why accept a vanilla ice-cream bar when you can have fresh cake-batter ice cream with peanut-butter cups?Vending machines are always there when you need them. Whether you want to quench your thirst, satisfy your sweet tooth, or just munch on some chips, it’s hard to walk past one without counting your change.Of course, today, most [vending machines](https://www.littlethings.com/books-actually-vending-machines/?utm_medium=google) also take credit cards. However, if you think a card reader is high tech, the next generation of vending machines is going to blow your mind.

The trend of cashless payments through vending machines is growing significantly due to an increase in smartphone addiction and payments using Internet-enabled devices, debit cards, and credit cards. Technavio’s market study identifies the growing demand for cashless vending machines to be one of the primary growth factors for the intelligent vending machine market. Companies are installing vending machines that accept payments through credit cards, debit cards, and mobile payments. In addition, retailers preference is shifting towards cashless transaction facilities and machines with cashless features. Our market analysts estimate that the market will grow steadily at a CAGR of greater than 34% by 2022.

The growing ability of real-time data collection through intelligent vending machines is one of the key factors driving the growth of the intelligent vending machine market. The functionality of the machine is decided by the hardware, software, and the microprocessors and controllers used in designing the machine. Intelligent vending machines are being equipped to handle in-data collection, data transfer, and data analysis. These advanced features enable the retailers to focus on their core competencies of timely distribution and product quality. Vending machines use customized software analytics and cloud-based services to manage large real-time customer data.

## Competitive landscape and key vendors

The intelligent vending machine market is moderately concentrated owing to the strong market presence of few key vendors. Intelligent vending machine manufacturers compete intensely in terms of price, quality, regularity compliance, and innovation. Consumers differentiate between diverse product offerings based on brand, labor, technology, capex, and R&D. According to the vending machine industry analysis, the entry of start-ups, organic, and inorganic players in this vending machine market intensifies the level of competition among the players in the smart vending machine market.

**References**

**Source:** [**http://www.ni.com/white-paper/7595/en/**](http://www.ni.com/white-paper/7595/en/)

https://www.youtube.com/watch?v=c5qFBed881s

https://en.wikipedia.org/wiki/Vending\_machine

[www.viralblog.com/trends-innovations/20-interactive-vending- machines-campaigns/](http://          www.viralblog.com/trends-innovations/20-interactive-vending-      machines-campaigns/)