Lab 1: Introduction to LabVIEW

ECSE 421: Embedded Systems
Department of Electrical and Computer Engineering
McGill University
Version 1.2

Adam Cavatassi, Jeremy Cooperstock, and Brett Meyer

Winter 2018

Introduction

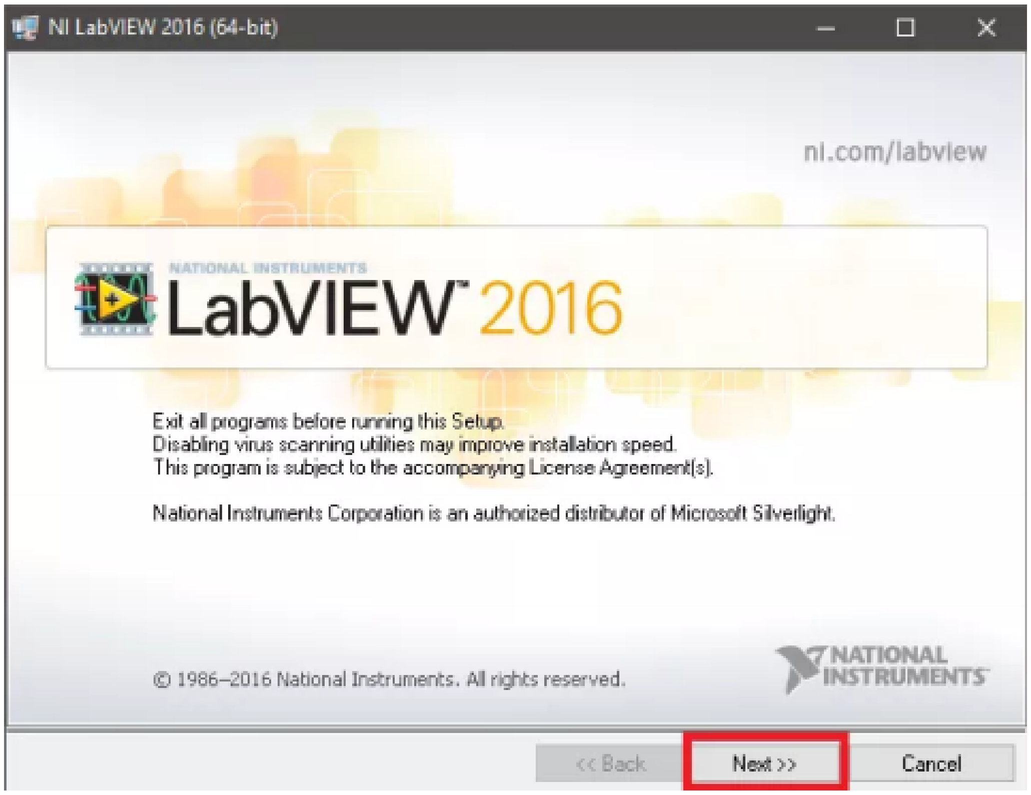
This assignment will serve as your introduction to LabVIEW. Later in the course, you will be using LabVIEW to program specialized hardware, but for this first assignment you only need the desktop computers in the lab. In many embedded systems, finite state machines are useful for controlling a program. Implementing an FSM in LabVIEW is just as easy as in C. In this problem, you will demonstrate this by writing LabVIEW code to operate a soda machine via FSM. Please read http://www.ni.com/tutorial/7595/en/ before starting this problem to get an idea about the implementation.

Problem

Suppose you have just invented a new soft drink. You want to sell as much of your product as possible, but don't have time to sell it in person because you are too busy working on your embedded systems project. Since you are an electrical engineer, you decide to build a soda machine to do the dirty work for you. To compete with Coke and Pepsi, you decide to sell your sodas for \$0.25 per can. The machine can take nickels, dimes, and quarters. The machine should dispense a soda as soon as the correct amount is entered. Also, the machine should be able to make change if someone enters more than \$0.25.

Designing the FSM

Design an FSM with 10 states corresponding to each possible value of change that a costumer enters. Each state has two outputs: the amount of change to be returned to the customer, and whether or not a soda is dispensed. The state transitions every time the costumer enters a new coin. For example, if the user enters a nickel, the FSM will enter the "5 Cents" state. In the "5 Cents" state, no soda is dispensed, and no change is returned because the total amount entered has not yet reached above \$0.25. From the "5 Cents" state, if the user enters a nickel, the next state will be "10 Cents"; if the user enters a dime, the next state will be "15 Cents"; if the user enters a quarter, the next state will be the "30 Cents" state.



Exit all programs before running this Setup.
Disabling virus scanning utilities may improve installation speed.
This program is subject to the accompanying License Agreement(s).

National Instruments Corporation is an authorized distributor of Microsoft Silverlight.

© 1986–2016 National Instruments. All rights reserved.

Figure 1: LabVIEW launch window.

Implementing the FSM in LabVIEW

Now you will implement the FSM you designed in LabVIEW.

1. Open LabVIEW and click on the **Create Project** button in the left pane.
2. Select **Blank Project** and press **Finish**.
3. In the Project Explorer, right click on **My Computer**, then **New**, then **VI**, to create a new Virtual Instrument (VI).
4. The front panel and block diagram should then open up. Make a front panel with buttons representing the user entering a nickel, dime, and quarter. There should be a string output for the total change deposited and also the amount of change to be returned. You can use an LED to indicate weather soda has been dispensed or not. Also, use a **Grab Soda** button to reset the machine after the user has taken his/her soda.
5. Just like in C, an FSM can be implemented as a case structure inside a while loop. The case structure can be found under **Programming** → **Structures** → **Case Structure**. Place the case structure inside of a while loop. The state can be kept track of using a shift-register (the little up and down arrows at the edge of the while-loop). To get these, right click on the loop boundary and select **Add Shift Register**. The initial state is piped into the shift register from outside the while-loop. In the code below, an enumeration constant is used so that each state has a unique name. This constant is available on the **Numeric** palette.

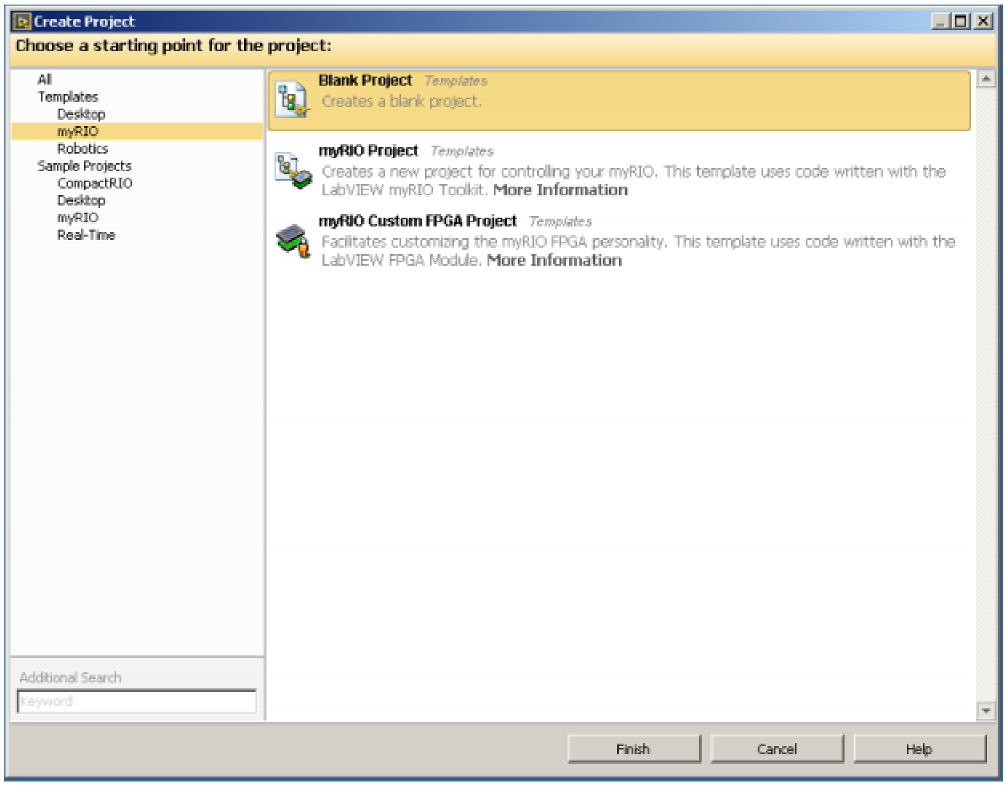

Create Project
Choose a starting point for the project:
All
Templates
Desktop
myRIO
Robotics
Sample Projects
CompactRIO
Desktop
myRIO
Real-Time
Blank Project Templates
Creates a blank project.
myRIO Project Templates
Creates a new project for controlling your myRIO. This template uses code written with the LabVIEW myRIO Toolkit. More Information
myRIO Custom FPGA Project Templates
Facilitates customizing the myRIO FPGA personality. This template uses code written with the LabVIEW FPGA Module. More Information
Additional Search
Keyword
Finish
Cancel
Help

Figure 2: Select a blank project.

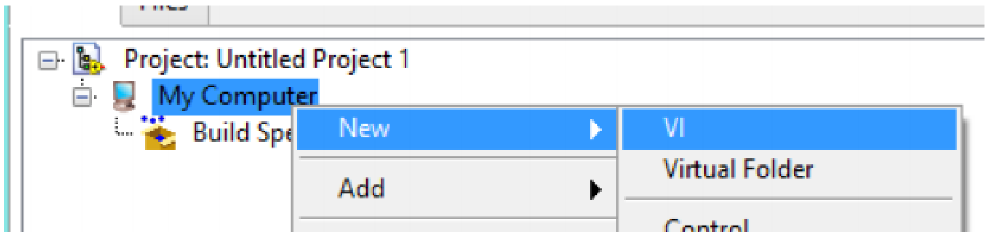


Figure 3: Build a new VI.

6. Connect the shift register to the little question mark tab on the outside of the case structure. You can add additional cases by right clicking at the case selector on top of the structure and selecting **Add Case After**.
7. Fill in the logic for each state. It should calculate the next state based on inputs and should calculate the correct outputs as well. You may find the select block useful (**Programming** → **Comparison** → **Select**). The select block is essentially a multiplexer: the boolean input decides whether to pass the input wired to the true or false terminal.

Submission requirements

For this lab, you will be required to submit a .zip file containing a screen capture of any one state of the case structure, all your LabVIEW project files, and a README which outlines how to run your code. The

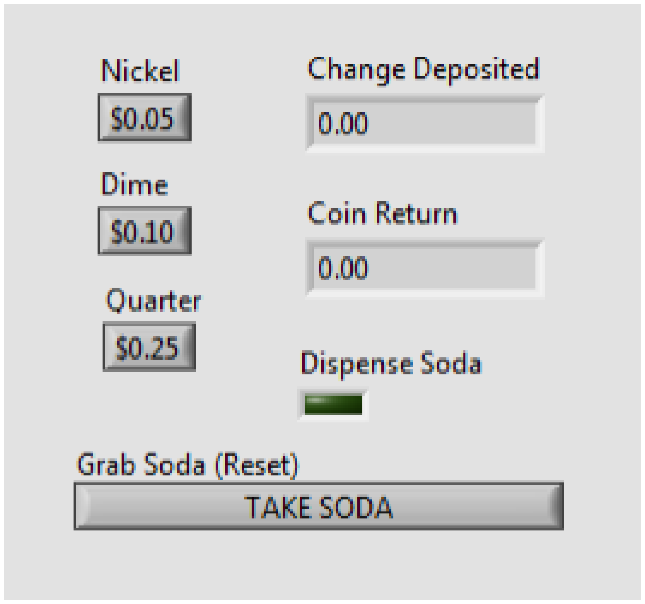


Figure 4: Example front panel of the vending machine VI.

README should also include a URL to a private YouTube video with a maximum length of 30 seconds, which illustrates the functionality of your vending machine. Specifically, demonstrate the specific case of inserting a nickel, followed by a dime, followed by a quarter.

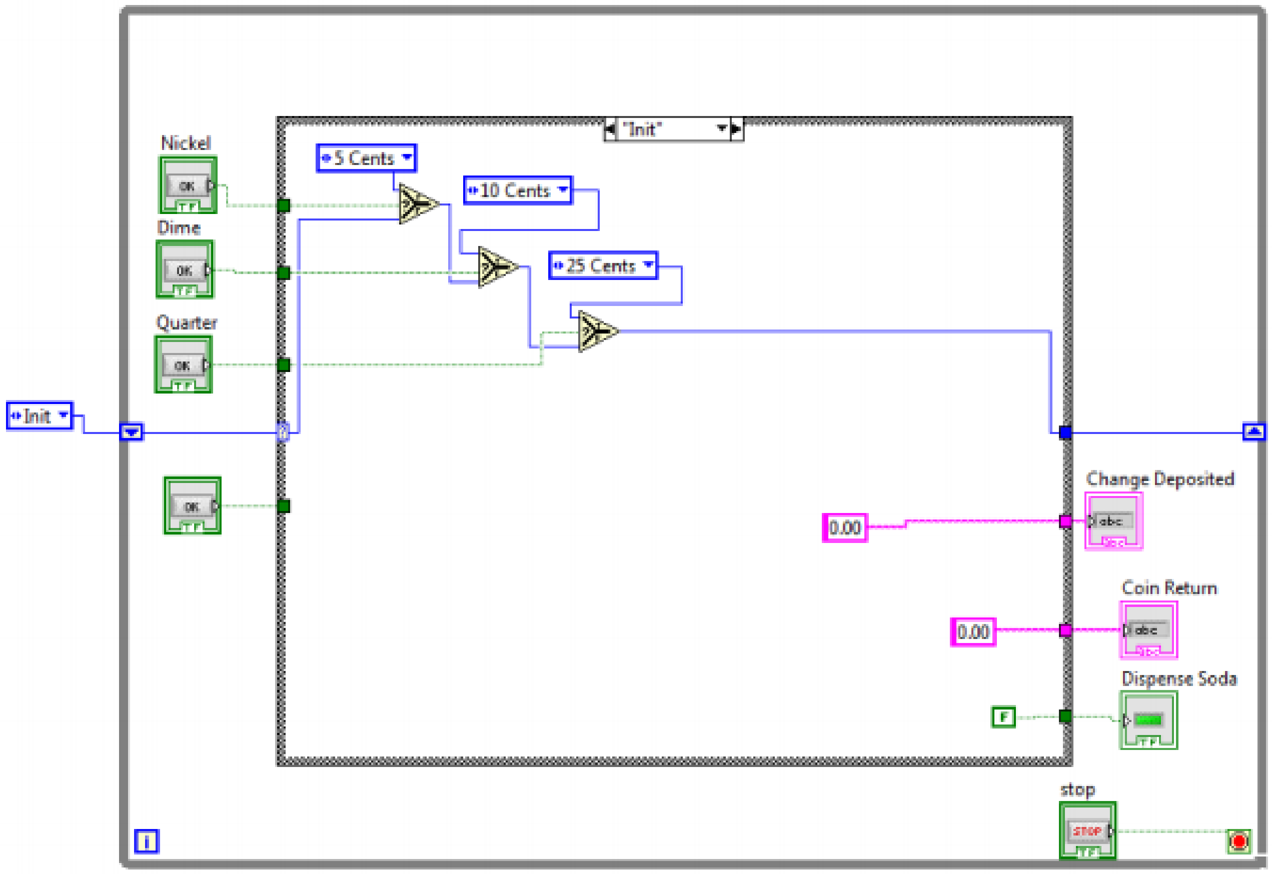


Figure 5: The VI block diagram.

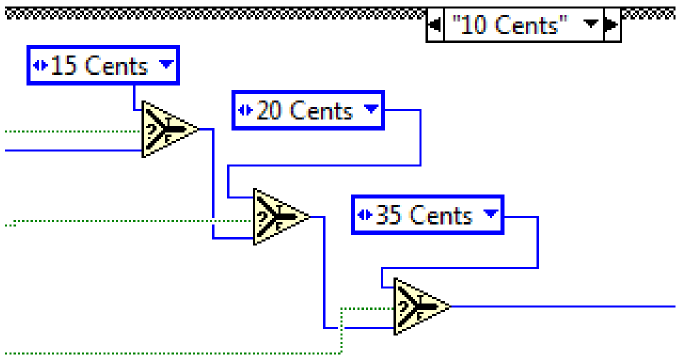


Figure 6: Usage of the select block module.