## Lab 1 - The Hot Dog Counter

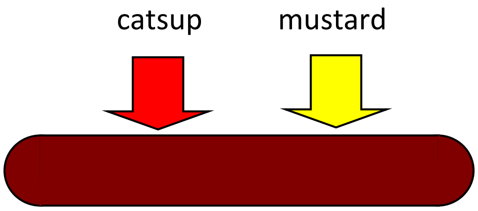
Project Description:

Fred and Rudy are making hot dogs at the ballpark. Fred dispenses mustard and Rudy dispenses catsup. A hot dog is not sold without each Fred and Rudy putting both mustard and catsup on the dog. As each pushes the button for their ingredient, a signal is fed to the PLC for the action. Either button may be pushed first. Design a program to count the total number of hot dogs made. Inputs should be wired to contacts and labeled as mustard and catsup. A display is kept in the PLC showing up-to-date counts of hot dogs made by Fred and Rudy.

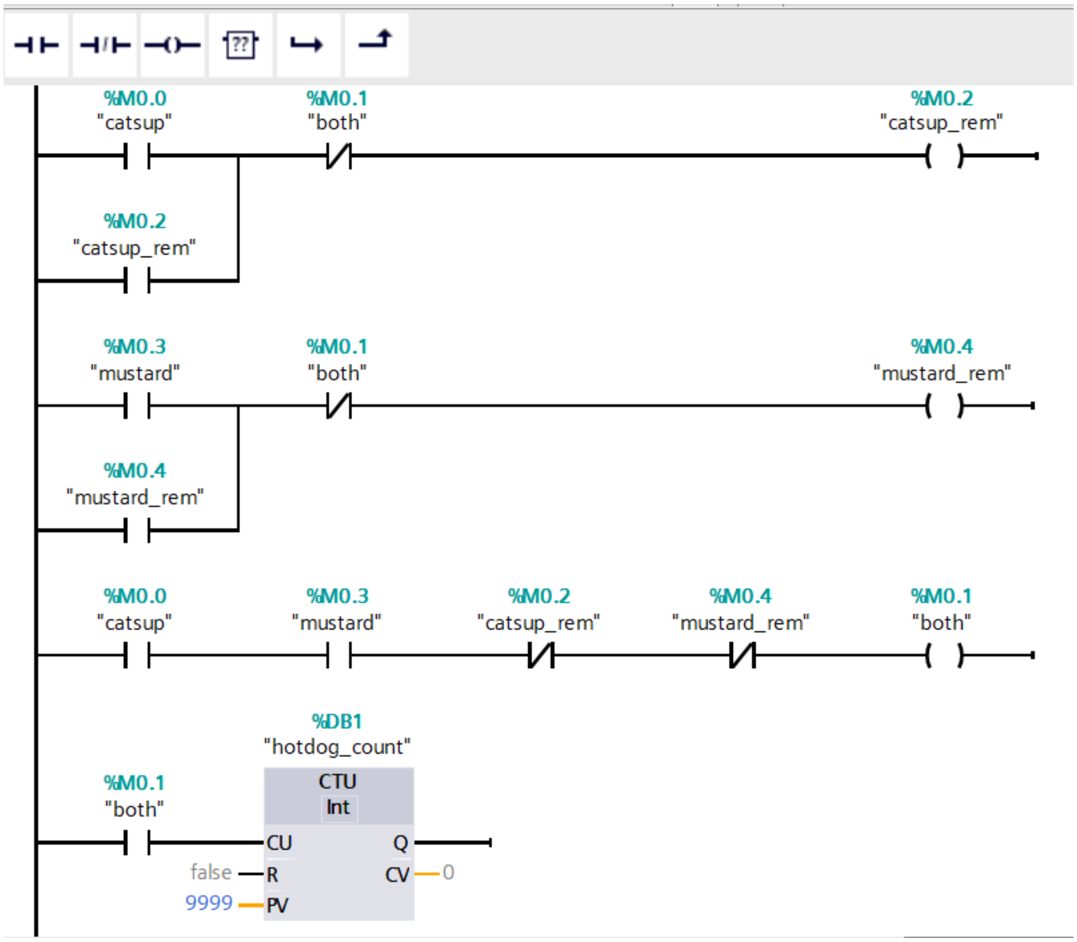
To complete the lab, enter the program shown later in the lab into the PLC and wire the two inputs.

Watch the count accumulate in the counter as the two buttons are pressed in any order.
Get a listing from the listing software on the programming software package.
The documented listing of the program may be used as the final lab report.

Wire the PLC to the inputs for this lab and to inputs or outputs for other labs per the diagram on the next page.
The next page shows the layout of the PLC on the trainer and the PLC wiring schematic. To wire the two inputs, wire through the two pushbuttons selected so that 24 volts is at the terminals of I/0 and I/1 when the two buttons are pushed.



Enter the following 4 rung program in Siemens TIA Portal.

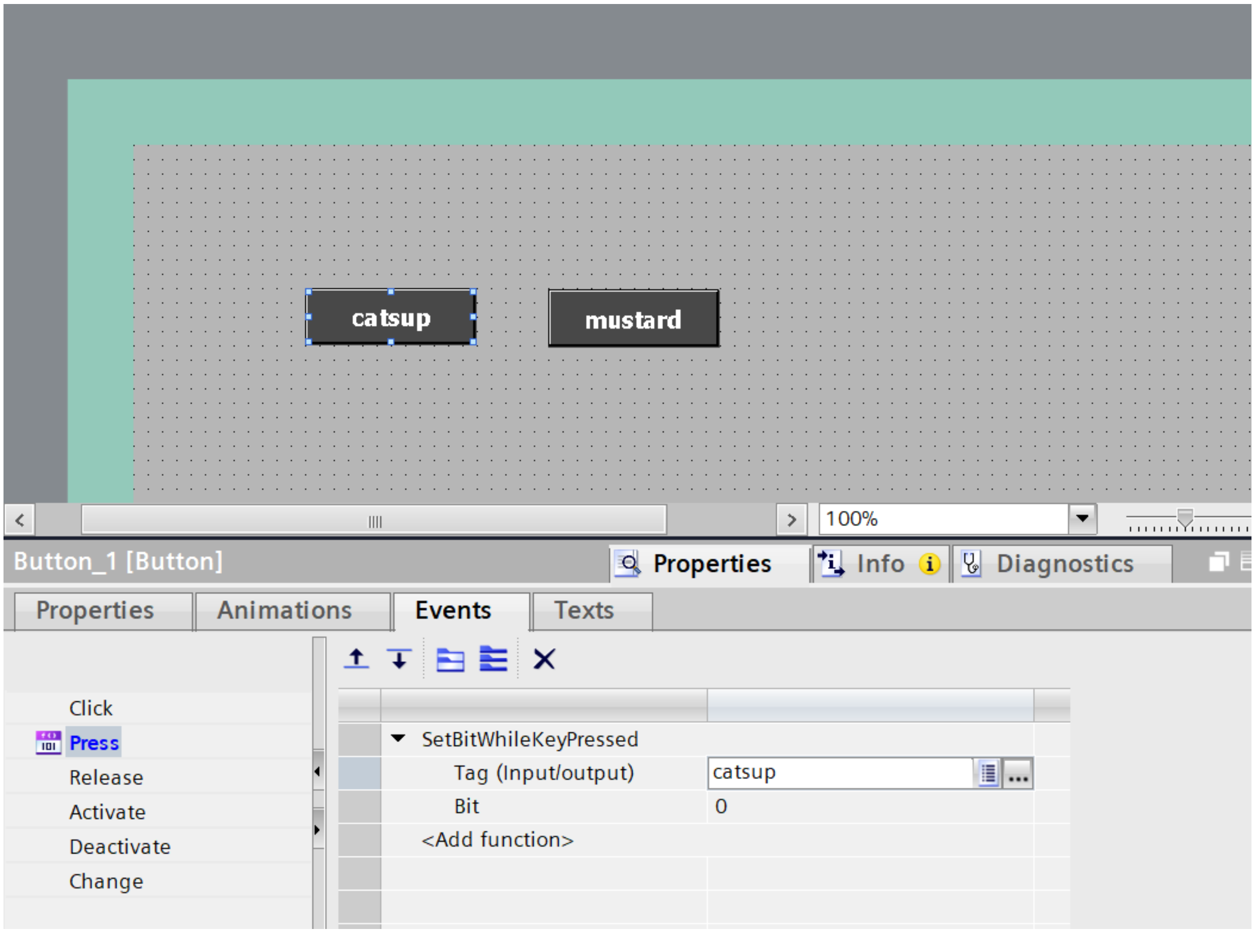


The count of hot dogs made is found in the accumulated value of the counter.

In the example above, we show the type of inputs available for PV and CV. In general, PV is short for process variable and CV is short for the controlled variable. For the up-counter, PV is the count preset and CV is the active count. The PV may hold a constant or a variable.

The HMI screen is shown below. Remember that the pushbutton may be programmed as a setbitwhilekeypressed bit. The first time the screens are set up in the computer, a change to the pgpc interface must occur. This is outlined in the video as well as at the end of Ch. 15 of the Hybrid Text.

When the pushbutton is pushed, watch the program change to reflect the state of the remembered bit. It stays on until the other button is pushed and then removed. Then the bit 'both' turns on, counts up one and resets the memory or seal bits to the two memory circuits. Watch as this takes place.



SetBitWhileKeyPressed

Tag (Input/output) catsup

Bit 0

<Add function>

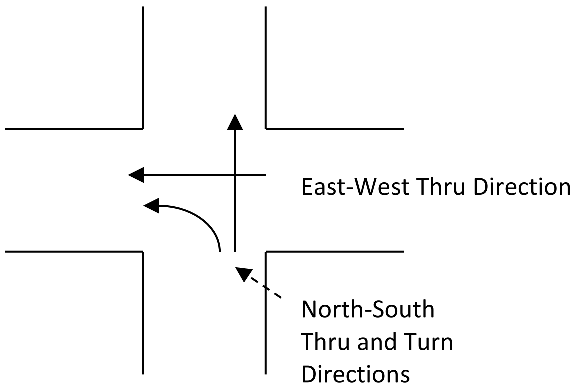
Setbitwhilekeypressed.  Remember.

To get credit for this lab, take a screen shot and email to the instructor or use your phone's camera to take a picture of the circuit on the screen with the count above zero.  This proves the program is working correctly.

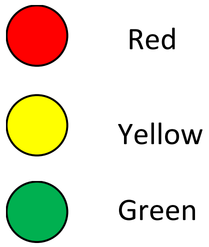
## Lab 2 - The Traffic Light Program

Project Description:

A traffic intersection has the following three lane assignments:
East-West Thru
North-South Turn
North-South Thru



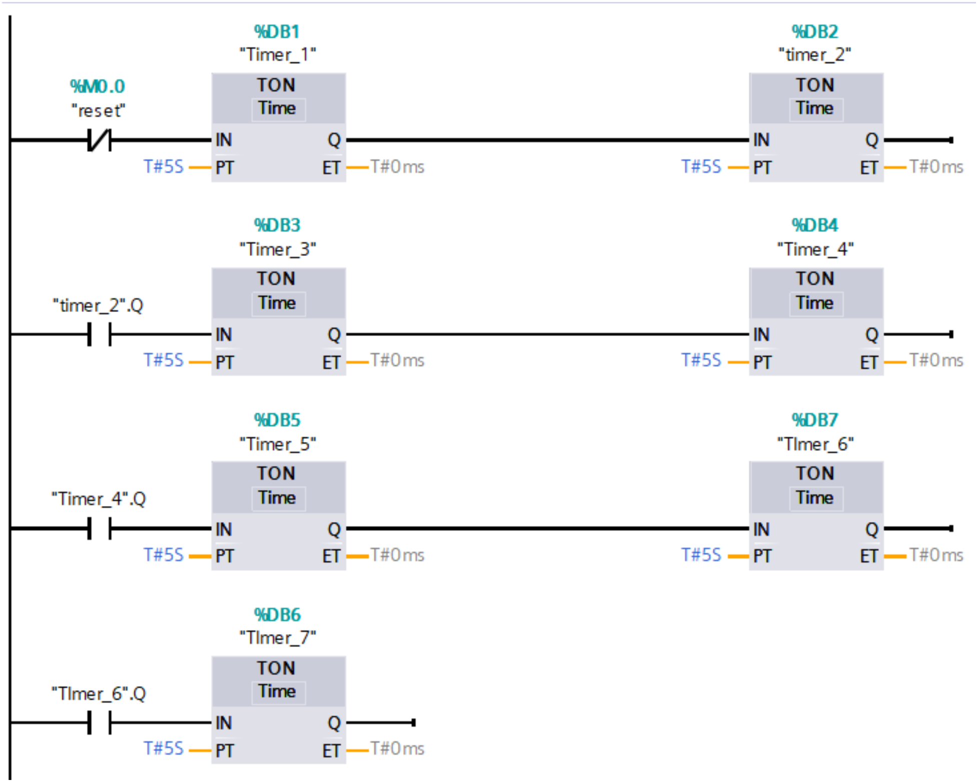
Two sets of traffic lights are found for each turn direction although the lab uses only one set. Each turn direction has a set of three lights as follows:

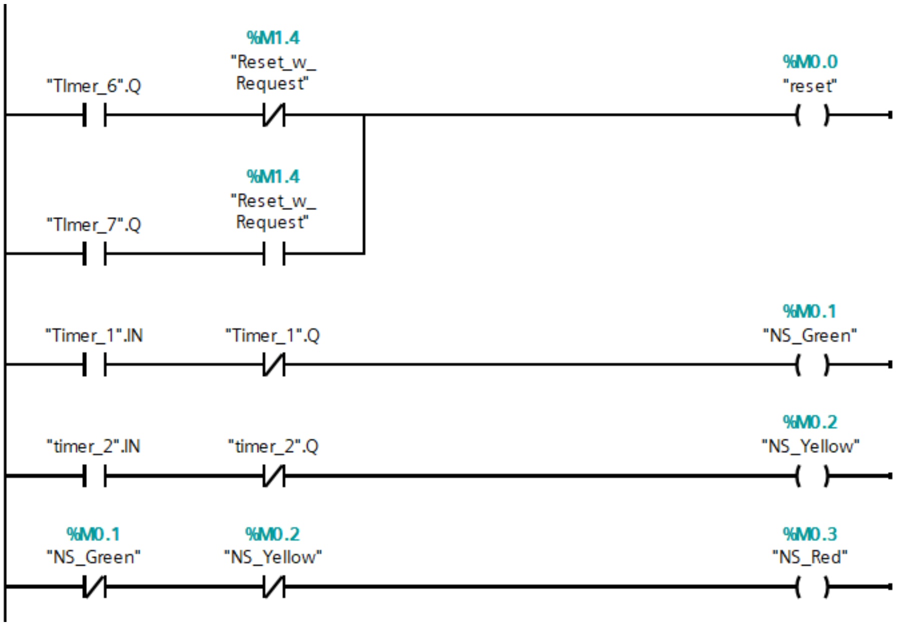


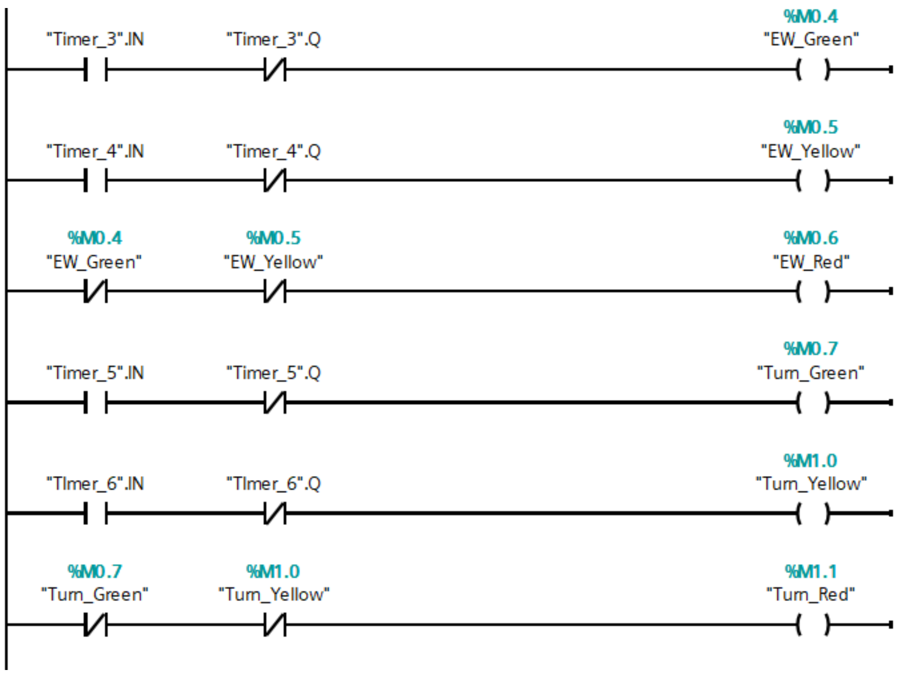
Although traffic intersection logic tends to be very complicated in order to provide fool-proof operation of the traffic intersection, a simplified chart of the operation of the lights can be used to program the lights and operate the intersection.  Each interval is an interval of time and after the last interval, the process repeats from the top.  The intersection's operational chart:

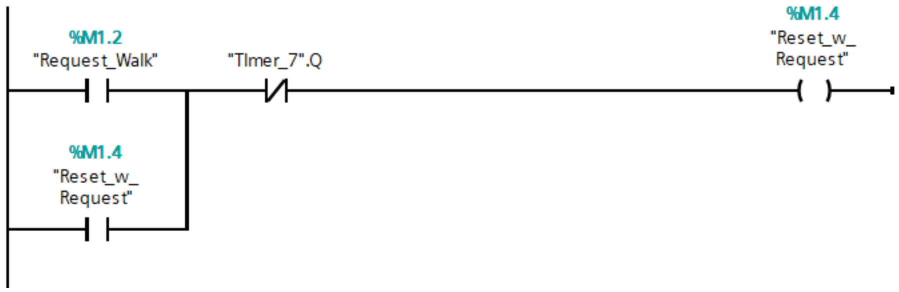
| Interval | N-S Thru Lane | N-S Turn Lane | E-W Thru Lane |
|---|---|---|---|
| 1 | Green | Red | Red |
| 2 | Yellow | Red | Red |
| 3 | Red | Green | Red |
| 4 | Red | Yellow | Red |
| 5 | Red | Red | Green |
| 6 | Red | Red | Yellow |

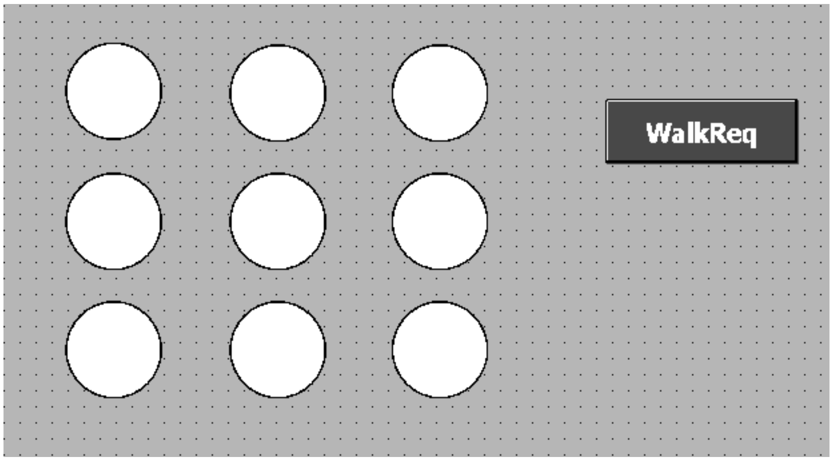
This lab consists of programming the nine lights to cycle through the proper sequence to control traffic flow at the intersection described above.

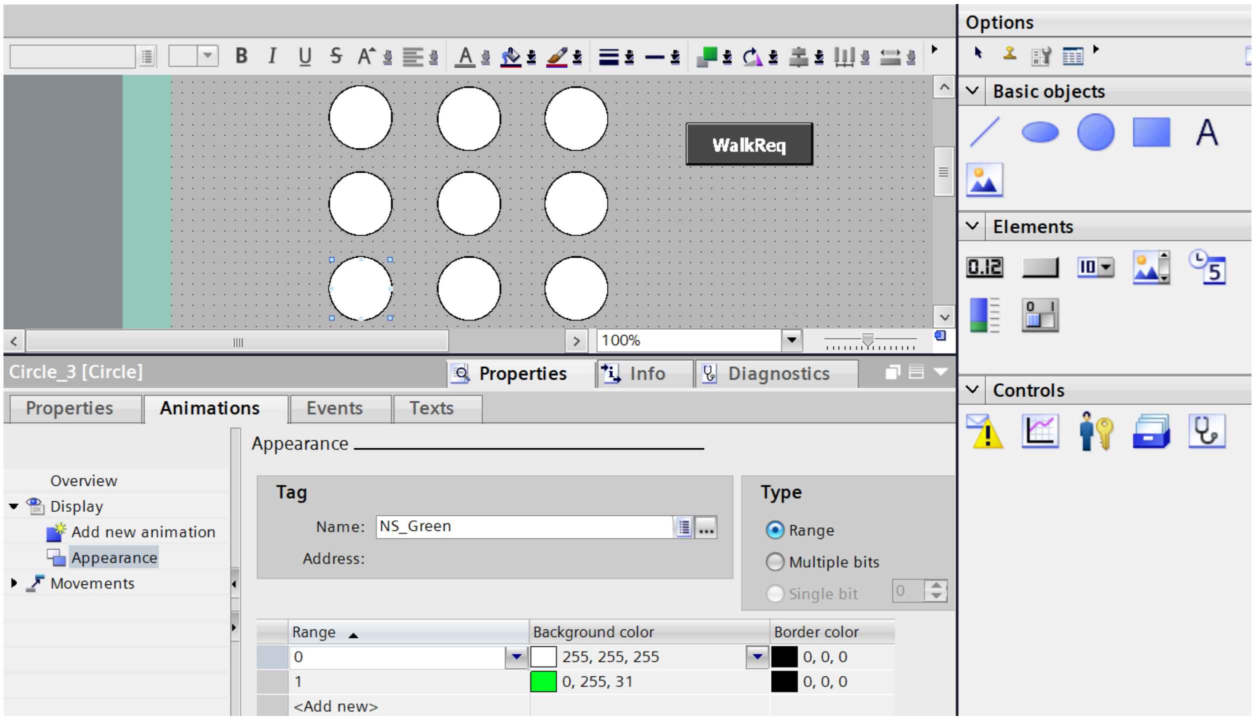





"Timer_3".IN
"Timer_3".Q
%M0.4
"EW_Green"
"Timer_4".IN
"Timer_4".Q
%M0.5
"EW_Yellow"
%M0.4
"EW_Green"
%M0.5
"EW_Yellow"
%M0.6
"EW_Red"
"Timer_5".IN
"Timer_5".Q
%M0.7
"Turn_Green"
"TImer_6".IN
"TImer_6".Q
%M1.0
"Turn_Yellow"
%M0.7
"Turn_Green"
%M1.0
"Turn_Yellow"
%M1.1
"Turn_Red"


%M1.2
"Request_Walk"
"TImer_7".Q
%M1.4
"Reset_w_
Request"
%M1.4
"Reset_w_
Request"


WalkReq



Circle_3 [Circle]

Properties | **Animations** | Events | Texts

- Overview
- Display
  - Add new animation
  - Appearance
- Movements

Appearance

**Tag**

Name: NS_Green

Address:

**Type**

● Range

○ Multiple bits

○ Single bit 0

| Range | Background color | Border color |
|---|---|---|
| 0 | ⬜ 255, 255, 255 | ⬛ 0, 0, 0 |
| 1 | 🟩 0, 255, 31 | ⬛ 0, 0, 0 |
| <Add new> | | |

## Lab 3 - The Cash Register Program

Project Description:

Design a simple cash register similar to one found at McDonald's or Burger King.  To do this, determine a menu of five or six items from the restaurant.  Also, include a Total button or a clear button or possibly both.  Also, include a means for backing out of a mistake without starting over from zero.  Display the cost of the total order in the PLC at an address in the data table.  Use Floating Point Math with two decimal places.

| | | |
|---|---|---|
| Whopper Combo | Whopper | Cancel Last |
| Whopper Dbl Combo | Fries | New Order |
| Whopper Jr Combo | Drink | Total/Tax/Optional |

Find the approximate prices from a McDonald's or Burger King for the items you choose.  When an item is entered, its count is incremented automatically by one.  If a button is entered multiple times, the count is incremented to display the total count.  If a mistake is made, the attendant must be able to back up at least one entry and erase the last item or decrement that item by one.

Display the final total in the PLC (not on the display of the trainer).

Automatically recognize that the entry of the individual items such as Whopper, Fries, and Drink will be given the price of the Whopper Combo instead of the individual prices.

## Lab 4 - The Hotdog Program - Wired

Project Description:

## Lab 5 - The Traffic Light Program with Input and Output Wiring

Project Description:

## Lab 6 - The Simon Says Program – two weeks

Project Description:

## Lab 8 - The Function Block Program

Project Description:

## Lab 9 - The Stepper Motor Program - Wired

Project Description:

## Lab 10 - Combine a Stepper Motor with a Function Block Program – two weeks

## Lab 12 - First Level of the Rubiks Block Program