Timing Toy Cars using Pneumatics and Sensors

## By

## Klint Mahne

## The Project Synopsis

This project was born out of an assignment to utilize a Micro800 series PLC, interface it with an HMI that allowed cycling between two screens, include two different sensor types, and control a dual action cylinder in ladder logic. The system had to be set up with safety mechanisms consisting of either 2-hand control or light curtains. The HMI would be required to control at least one function via a button of the touchscreen or a physical “F key” button located on the HMI. Using these parameters, I decided a fun way to demonstrate the system would be to have a pneumatic cylinder, operating off a directional control valve, push a small Hot Wheels car from one end of a Hot Wheels track to the other. Each side of the track would contain two proximity sensors to mark the time it left the starting line and to mark the time it arrived at the finish line. The HMI would contain a launch button on one screen and a statistics screen on the other.

## Items Used

For this project the following parts were used in the construction. As you can imagine, the design was very crude and could be improved upon in many ways.

|  |  |
| --- | --- |
| Micro830 PLC Controller | 1. Double Acting Cylinder with magnets
 |
| Connected Components Workbench Program | 1. 5/2 Solenoid Dual Control Valve
 |
| (3ft) Hot Wheel’s Track | (1) Hot Wheels Die Cast Car |
| (2) Proximity Sensors | (2) Magnetic Sensors |
| (1) 2711R-T4T HMI | (1) Momentary Key switch |
| Zip Ties  | 1. Pneumatic Speed Control Restrictor
 |
| Pneumatic Tubing | Electrical Wire  | Compressed Air System |

## Assembling the Physical Parts

The first place to start for the project is the assemble the actual moving parts. It is important to ensure that all the parts will be in the correct place and in working order for the sensors to properly pick up on them. You can technically assemble them in a modular fashion; however, I feel that its most appropriate to start with the basic parts and keep it as simple as possible before integrating more components. In this case, I suggest starting with the assembly of the Hot Wheels track. While assembling the track take note to ensure the sections of the track snap together smoothly and there are no edges that could potentially knock the toy car off. It is a good idea to have something stable at the end of the track in which to catch the car.

Once the track has been assembled, place the proximity sensors at appropriate locations of the track making sure to keep note of which sensor is the starting line and which is the finish line. From here, manually push the toy car along the track to test there are no points of instability and the sensors are safe from impact. Retrieve the pneumatic speed control restrictors and integrate them into the dual acting cylinder. Test the cylinder by pulling the shaft out to its extracted state and pushing it back into its retracted state. Using the Zip Ties, secure the cylinder to the end of the Hot Wheels track associated with the starting line. With the air compressor disconnected, use the pneumatic tubing to connect the cylinder to the dual control valve, ensuring to put in the speed control restrictors. Connect all relevant inputs and outputs of solenoid control lines into the output side of the PLC. Wire the magnetic sensors into two inputs of the PLC. Power up the PLC and put it into ‘RUN’ mode by downloading some empty code to the controller.

While the cylinder is in the retracted state, move one of the magnetic sensors on the cylinder’s outside housing until it detects the magnet and lights up. You should also see the corresponding input on the PLC light up. Secure the sensor to this spot on the cylinder using the zip ties. Pull the cylinder out to it’s extended state and place the second magnetic sensors following the same procedure as the first, making sure to note the PLC input lighting up as the sensor activates. Once completed, connect the air compressor up to the dual control valve and test the extending and retracting states of the cylinder by pressing the override buttons on the dual control valve. Note that the sensors should be activating and deactivating when the cylinder is changing states; if not then make any adjustment as needed. Disconnect the air compressor and power off the PLC.

## Programming the PLC

 Prior to hooking up or designing screens on the HMI, it is important to have the proper programming in the PLC. The HMI will need to be able to communicate with the PLC controller to know what types of variables it needs to control. Therefore, the variables must already be declared in the program for the HMI to be able to recognize them. Firstly, we will want to start by making the in/out map so we know where our inputs and outputs will be going. I combined my inputs and outputs into one program to help centralize it, but you can have separate input programs and output programs if you’re more comfortable with that. As you can see, I have one input for the 4 sensors and one for the key switch that will be activating the launch program. For my outputs, there is only the extender and retractor for the solenoid on the dual control valves.


I also have an additional line 8 not pictured; it’s not necessarily an input nor output, but rather a variable called gxKeyOff. This is simply a coil that is activated for by the gxKeySwitch being normally closed. Normally this would be unnecessary, however for the purposes of the HMI it will be a nice shortcut.

I used 3 UDFBs to develop the main function, allowing it to work as a simple state machine. The first UDFB takes the role of controlling the cylinder to launch the car. It works by using the sensors to determine where the cylinder is. Line 4 displays how the coil will extend when the enable line is active; essentially turning off the retraction sensor. As this is happening, a ‘charged’ variable is set to indicate that the process has started. The enable stays active until the cylinder is extended all the way and trips the extended sensor. Once this happens, a Valid Extension variable causes line 5 to activate, sending a pulse to an output called xValid. The output is read by the overall main program and moves the program to the next state, turning off the enable. When the enable is turned off, but the charged variable was set, the coil starts retracting. Once it is fully retracted, the sensor closes on line 2, the charged variable is reset, and the block goes into the valid retract state. This too sends a pulse to the valid output, allowing the main program to move to the next state.



The second UDFB is designed for scorekeeping and is extremely simple in it’s design. When enabled has a counter to note the number of races it has completed. The second line resets the values, and the third line keeps track of the Top Score.



The third UDFB has a few more variables but is also very simple in design. The first line activates a busy status when the enable has started the program and the valid (finished) has not been completed. This is really for any external piece of the program that might needed to have used the busy and known whether it could continue. The second line resets the time to 0. Lines 3 and for 4 take down the time at each start and finish event, and the final line subtracts the finish time from the start time to display how long the race took.


The main program is set up in the fashion of a state machine with two categories. The first 5 lines are focused on the movement between states, while lines 7 & 8 are the enable lines for the UDFBs, or rather the actions of each state. Line 6 is designated as a jumping location which is to prevent the state machine from immediately moving from state to state prior to the implementation of the action of the certain state.

Line 1 is set so that anytime the key is switched to the off position, the machine will be in an ‘idle’ state where it stays until certain criteria are met. Line 2 addresses how to get out of a fault state by pressing the pushbutton for 2 seconds. Line 3 shows the first movement in the state machine; when in idle once the key is turned on, the launch button is pressed, it will move to state 1. Lines 4 & 5 act as states 01 and 02 to advance the actions of the state machine, where once it has gone through it’s full cycle will end up back at the idle state.


## Setting up the HMI

 The final piece in this is the HMI integration into the project. Note that one could do the HMI before the physical setting up or the programming of the PLC, however I felt it ideal to have it as the last part of the tutorial based upon the pre-requisites it need to set up. The idea of the HMI is both to interact with it to have the machine react as well as display the statuses of the machine. Being that the assignment required two screens I figured I could make one that launches the car and another that displays the number of races, the time of the current race, and the time of the best race. The first screen, which included one of the buttons that would be required to launch the car, would have a message displayed if the key switch was not activated. This message would then disappear and be replaced by a “Launch” button once the key switch was activated. Pressing this button would then launch the car, however as soon as the key switch was no longer activated the launch button would disappear and would be replaced with the message – this was designed as part of the two-hand safety.

The other screen simply pulled the global variables and displayed them, as well as had a Reset button that would erase the values in the variables. Both screens also have buttons to transition from one screen to the other. The idea of the HMI was to allow all functions of the PLC to work without making any changes in the CCW program.

