STEMSEL Automotive Project 6 – RPM Tachometer

Problem

Imagine if you never knew how fast your vehicle is travelling. In all or most of our cars today there is a dial on the dashboard that displays various information regarding the vehicle's movement. The tachometer is a dial that calculates and displays the amount of revs (revolutions per second) that the vehicle is doing. Now we will investigate how the tachometer shows how fast the car is going and how many revs it is doing.

Background

There is much more to a tachometer than just connecting the red and black wires. For a tachometer to work, there are some sensors attached next to the engine's drive shaft on an iron wheel, where some magnets are also placed on the actual shaft, once the magnet passes these sensors, a voltage is generated by having charged the magnet. This voltage is directly proportional to the shaft speed which is sent back to the tachometer to show the information.
In newer cars today, there is less use of this method with the magnets as there could be some interference with outside sources that the sensors pick up. A new method, which consists of an infra-red laser where it shines a beam towards the shaft. The shaft itself has one spot which is reflective and when the beam is reflected back a signal is achieved, by measuring the rate at which the beam is reflected back.

Ideas

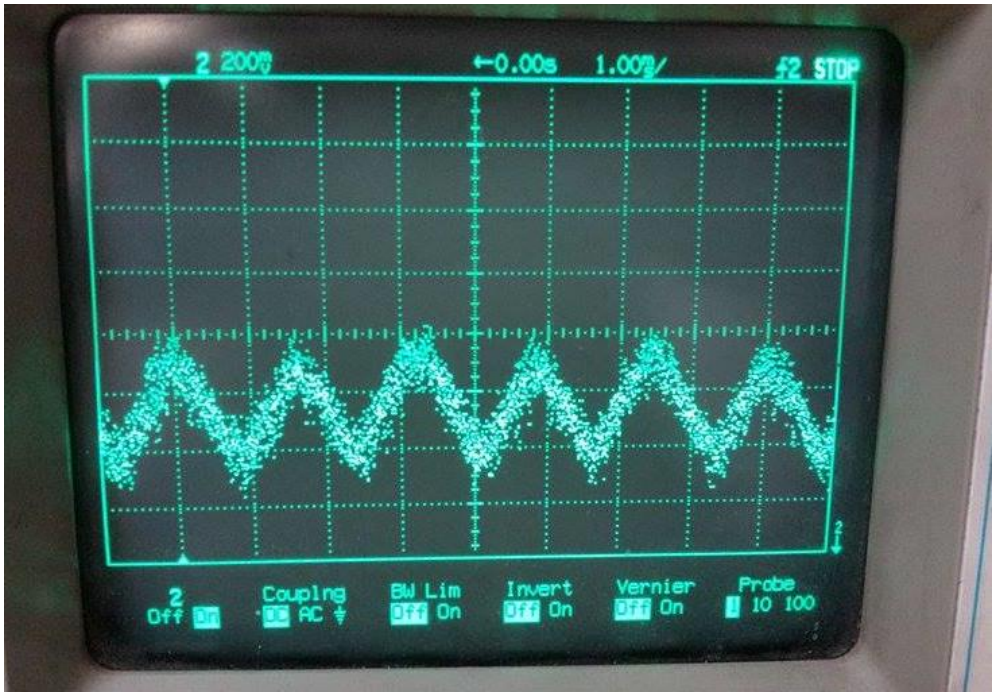
We normally see the tachometer in the car displaying the revs. How can we simulate this using the available components in the kit box? A fan could be used to simulate a wheel and an LCD could be used as a display.

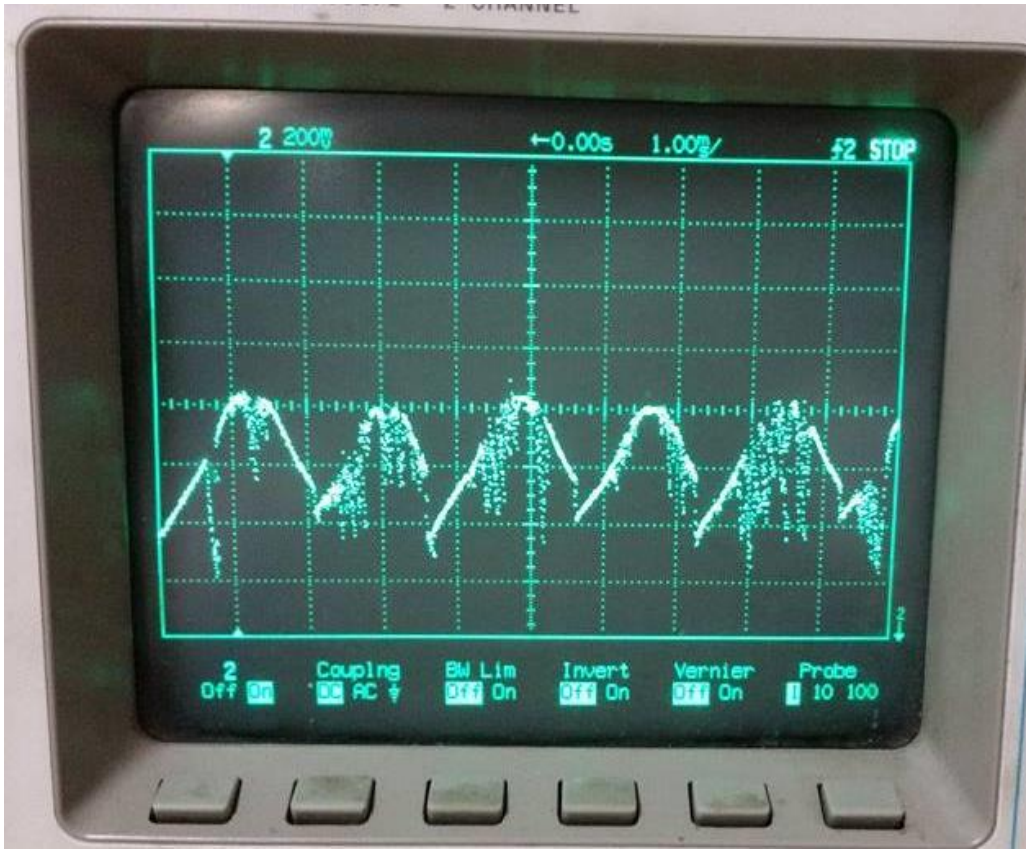
Plan

We now want to simulate a tachometer by using the motor and LCD. The user will blow onto the fan to allow rotation, then the board will calculate the voltage that is being generated and will display on an LCD screen. By measuring the period of the wave that is generated, it is possible to calculate the rotation of the motor.

Voltage signal generated by motor

When the motor is rotated a voltage is generated. When the motor is connected to an oscilloscope and then rotated at a constant speed the voltage that is generated can be seen as a signal. In the two pictures below we can see that there are noticeable maxima and minima for the signal. The first picture is the signal with significantly less noise than the second.





Design

All that is required is to click on the LCD icon from the circuit groups and a Analouge input for the motor to C7. The LCD will be automatically attached to the ezCircuit.

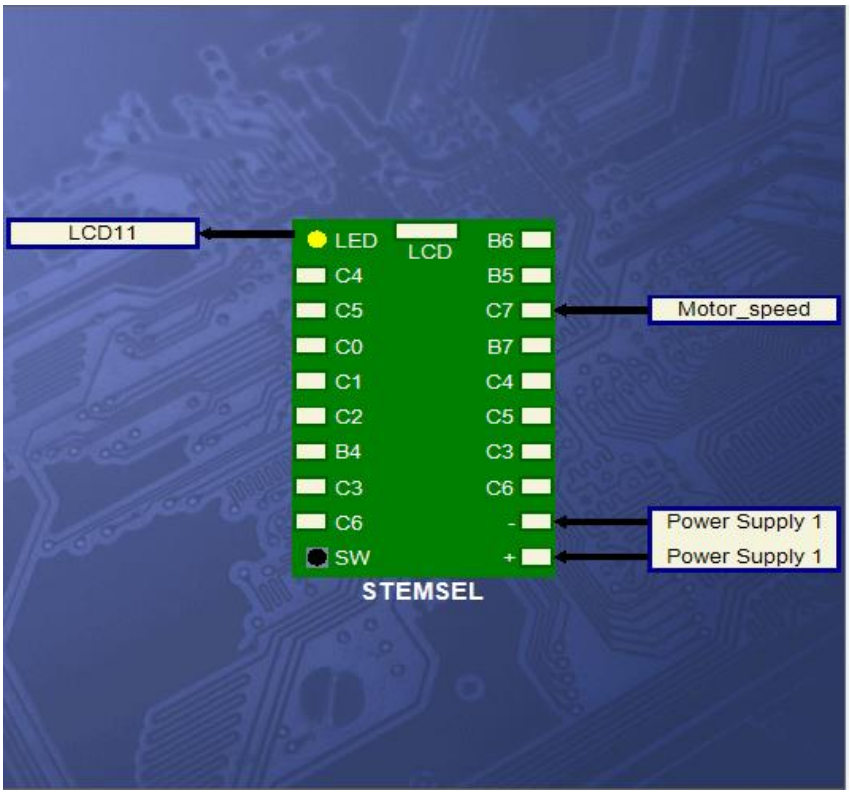


Figure 1 - Circuit Design

Build the circuit

Use the ezCircuit Designer I/O diagram to connect the hardware. Remember that black wires connect to the negative pin (-), red wires connect to the positive pin (+), and white wires connect to the pin we designated in our design. Once you have built your circuit, send the design to CoreChart.

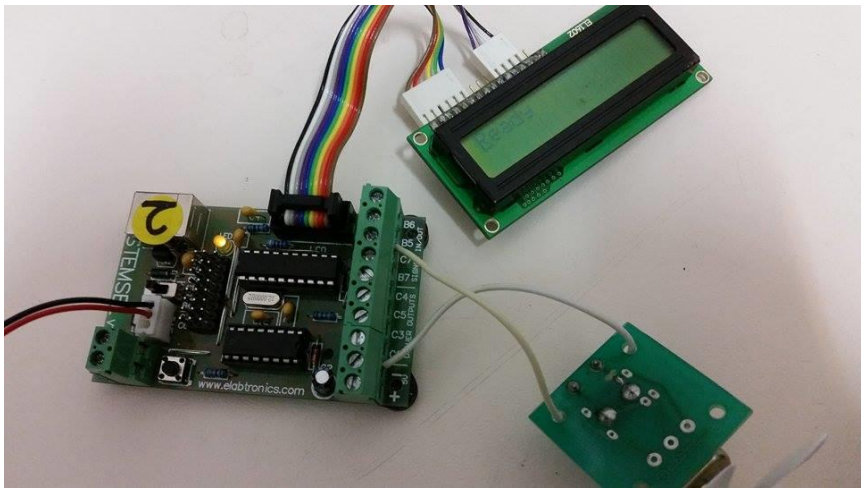


Figure 2 - The Circuit

Programming

Click the 'Send To Corechart' button. After you have used the test routines to check the outputs, delete them so that we can start writing our own program. We will be investigating the different steps one by one, then putting them all together at the end to make our game.

Part A: This will step you through spinning the fan and receiving a message on the LCD to let you know that the fan has been turned.

1. First we need to have an initial message for the LCD to let the user know that the board is ready. Click on LCD and insert an LCDformat to set it to home with 0 delay.

2. Now insert an LCD message that says "Ready".

3. Next we need an AnalogIn for the actual motor itself, so let's have an input of Motor_Speed and save it as "Motor_Volts". Next we need to make sure that when the fan is not moving, that the LCD still displays ready. To do this we add a compare block that compares if the voltage is below 0.05V.

4. After this add a GoTo icon to make the program loop if the fan is not moving. Next, add a time delay of 50 milliseconds to pick up the fan that is spinning. We now want to clear what was displayed on the LCD beforehand and enter a message to let the user know that the board has reacted to the rotating motor. Enter a message saying "It Spins!"

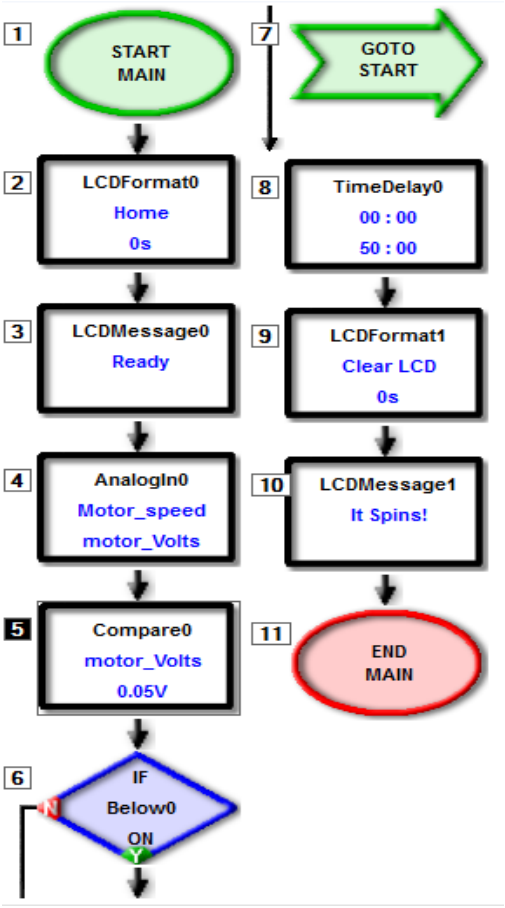
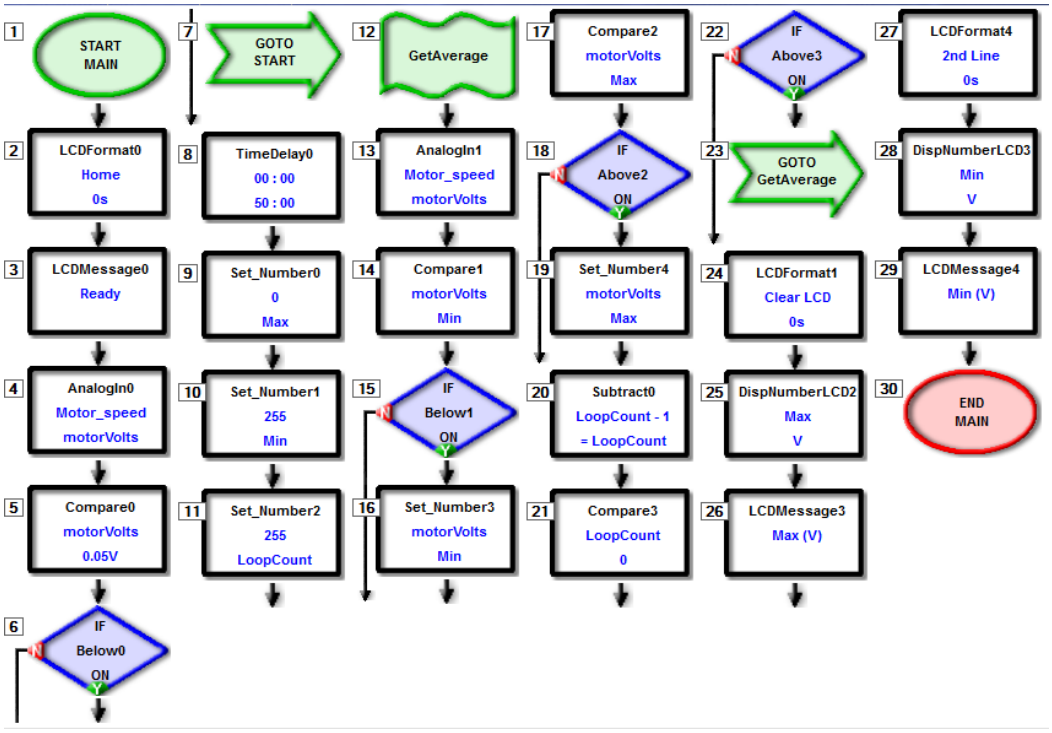


Figure 4 – Part A

Part B: This part will step you through being able to display the max and min voltage values for the signal that is generated when the motor is rotated.

1. Firstly, let's delete steps 9 and 10 because we won't be needing the LCD to display this text anymore.
2. We now need to set 3 different numbers, so now we will set 0 as Max, 255 as Min and 255 as LoopCount.
3. We need to create a loop now so that the Max and Min value of the signal's voltage can be calculated. We can start this by making a reference point, an address called GetAverage.
4. Now let's include the motor, use an AnalogIn, and select Motor_Speed and save it as Motor_Volts. Now we want to compare the MotorVolts with Min and Max, so we add a compare and compare if it's below min. If it is, set MotorVolts to Min and compare to see if MotorVolts is higher than max. If it is then we want to set MotorVolts to Max.
5. Now for the looping section we can **subtract** 1 from LoopCount and save it as LoopCount. Compare if LoopCount is above 0, if it is then put a GoTo GetAverage. This will allow the code to keep looping until the count is below 0.
6. We now want to display this to the LCD, so now clear the LCD with 0 time delay. Now we want to use DisplayNumber using the LCD and we can display, Max is what we want to display first, make sure Voltage box is ticked. After display an LCD message that says " Max" make sure that you leave a space in front of it.
7. On the second line with 0 time delay, we want to display Min, again, make sure the Voltage box is ticked. After, display an LCD message that says " Min" make sure that you leave a space in front of it to leave a space in the LCD. *Figure 5 – Part B*



Part C: In this section we are going to implement a reset, so that the user does not have to power off then back on the board to get another reading.

1. First we shall create a new address called CheckDone. This is to enable a constant loop that will check to see if the button is pressed or not. Now we will add an 'If' statement to check if A3PushButton is 'on' (to check if the button is not pressed as it is an active low). Now put a GoTo icon and direct it to CheckDone. This creates a constant loop that checks if the button is pressed.
2. Now if the button is 'off', we will clear the LCD with 0 time delay and add a GoTo icon that is directed to start. This is now essentially a reset button that will go back to the start if the board button is pressed.

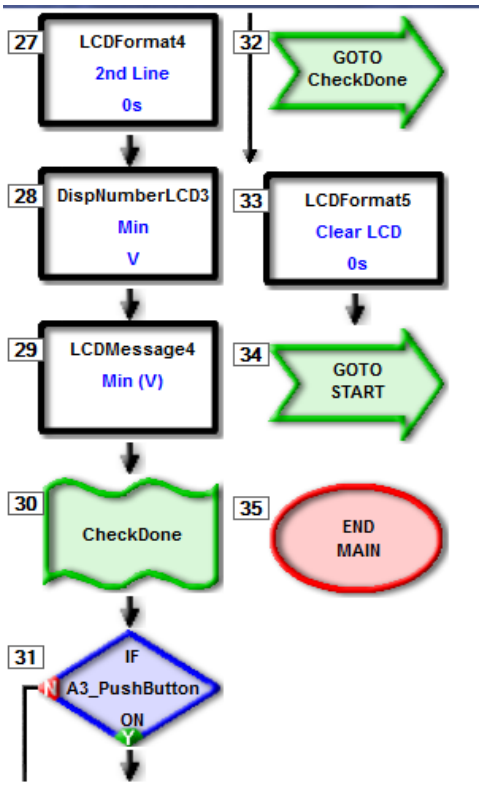
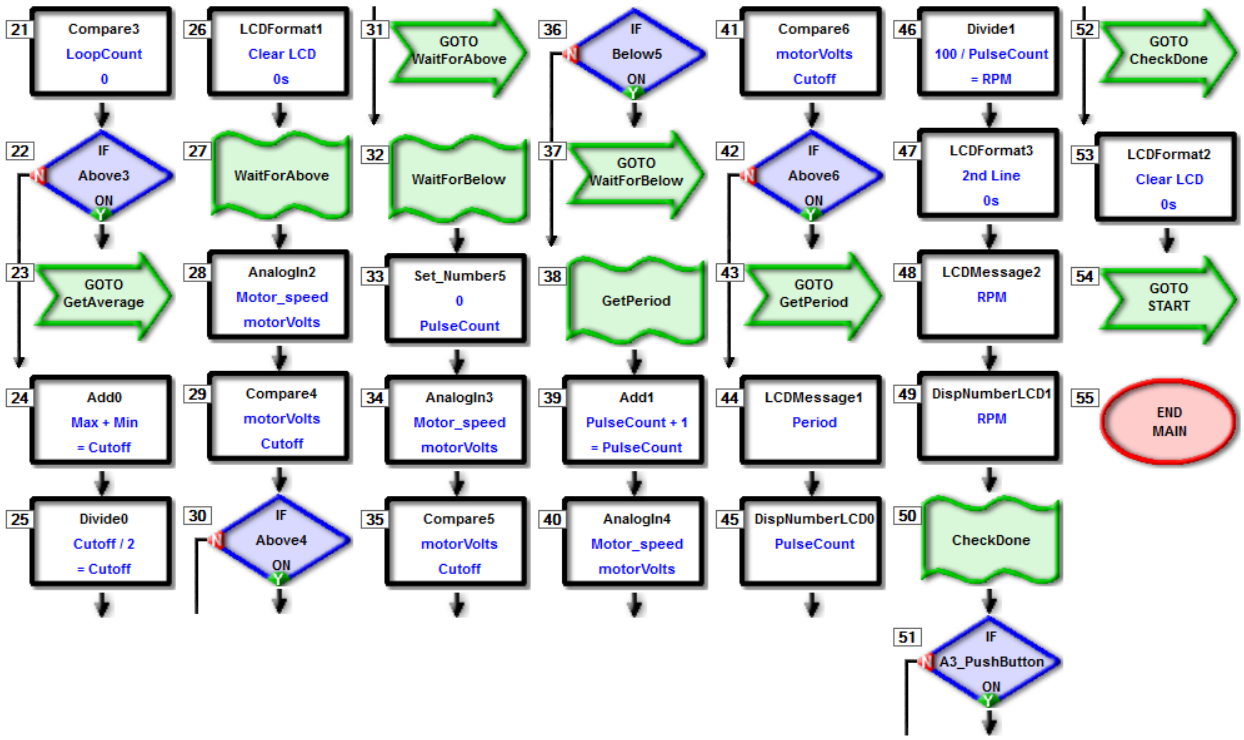


Figure 6 – Part C

Part D: In this section we will be able to calculate the period of the signal generated and the RPM of the motor.

1. First delete sections 24-29 as we don't need to display the Max and Min anymore as we will be using the LCD to display the period and RPM.
2. Under step 24, add an 'add' to add Max + Min and save it as "Cutoff". Now divide Cutoff by 2 and save it as Cutoff. Next clear the LCD with 0 second delay.
3. We now want to create a loop so we will add in a new address called "WaitForAbove". Next use an AnalogIn to use the Motor_Speed and save it as motorVolts. Now we want to compare if motorVolts is above the Cutoff. If it is add a goto 'WaitForAbove'.
4. Now we will create another loop called "WaitForBelow", first we will have to set a new number to count the pulse. So we set a number called PulseCount and start it at 0. Next we bring the motor in by using another AnalogIn block. Now we need to compare the MotorVolts with the Cutoff. If it is below make the program GoTo 'WaitForBelow'.
5. Now we will create the last main loop that gets the period. Create an address called "GetPeriod". Next Add 1 to PulseCount. Now we include the Motor again so create another AnalogIn with Motor_Speed and save it as motorVolts, compare motorVolts to CutOff. If it is above add a GoTo to make it go to GetPeriod.
6. Now we have all of the information we need to display on the LCD. As similar as in parts A and B, we need to display message "Period" along with PulseCount.
7. Then divide PulseCount by 100 and save that as RPM to calculate the RPM. Now for the second line with 0 second delay, display a message "RPM" and DispNumber 'RPM' *Figure 7 – Part - D*



Extension

For a small extension, once the motor is actually rotating, a red LED could be triggered to show that the fan is in motion. Now if the motor is again, turning, if it generates a higher voltage signal than the previous attempt then the green LED will light up. If not then the red LED will stay on. This will require a few extra loops to get working.

Summary

Now we can see that there is much more to tachometer than just a few magnets and sensors. It is good to know that the board is capable of simulating such a thing. Also the concept of voltage signals when a motor is rotating was grasped.