STEMSEL Automotive Project 3 – Trip Computer

Problem

Have you ever been driving and the low fuel alarm comes on? In Australia an average of 43 people a day run out fuel whilst driving. These cars often break down in dangerous or disruptive places impacting many other drivers.

Background

Due to a myriad of different reasons, a driver will sometimes be forced to make a trip whilst being extremely low on fuel. In these situations it would extremely valuable for a driver to know exactly how far they can travel. A driver who knew that he could not reach the petrol station before running out of petrol could then avoid making the trip and causing a potentially serious traffic incident.
Additionally, even if the car is not dangerously low on petrol, this trip computer will allow the driver to properly plan their fuel management.

Ideas

So how can we determine how far the car can travel? This would be completely dependent on the amount of fuel left. Using the amount of fuel left in the car as well as knowing the economic statistics of the car (distance travelled per litre of fuel) would allow us to quite accurately predict the potential distance travelled.

Plan

We have a potentiometer in our toolkits which can be used to input the varying fuel level.
The reading from the potentiometer can be transformed into a meaningful value within our project and displayed on the LCD output.
Three LEDs (red, yellow and green) can also provide a constant visual signal to the driver as to the status of the fuel level. That is: a green LED represents high fuel value, a yellow LED represents medium fuel value and finally the red LED represents a low fuel value.

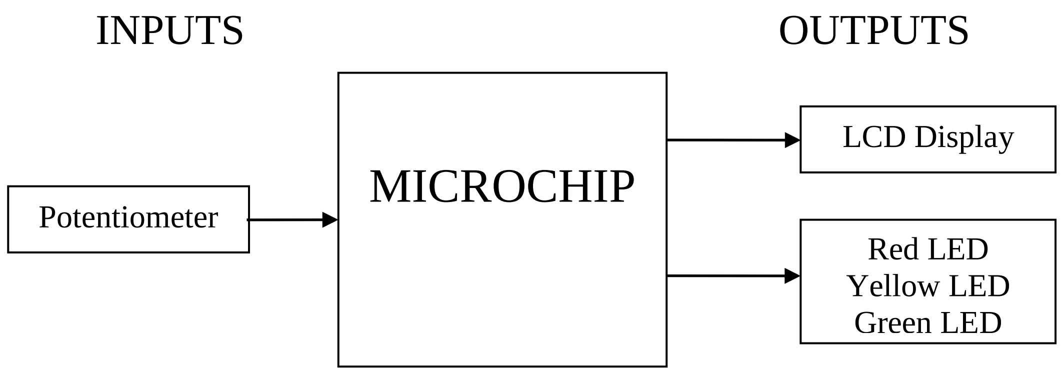
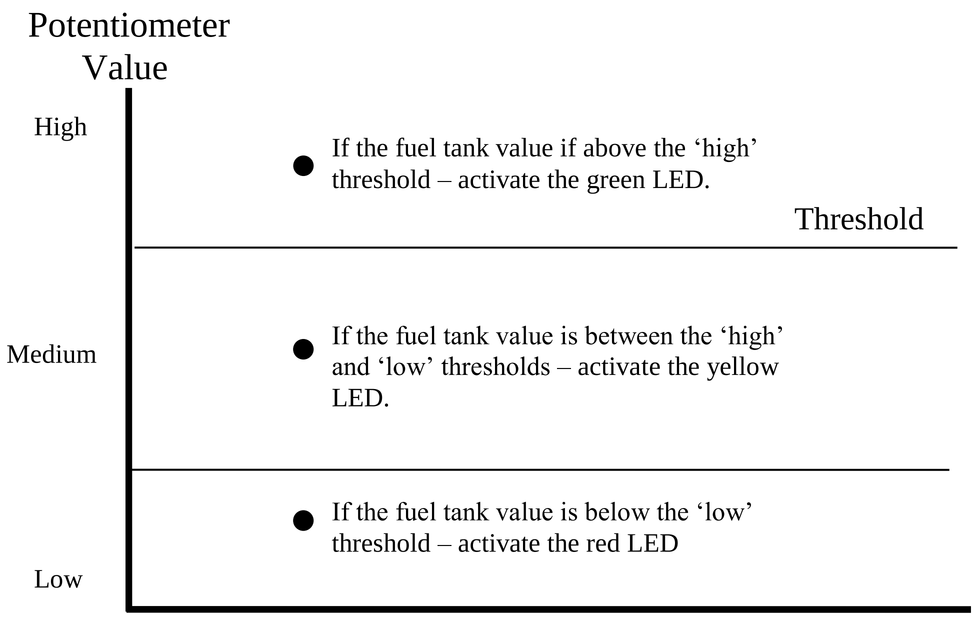


Figure 1: circuit plan

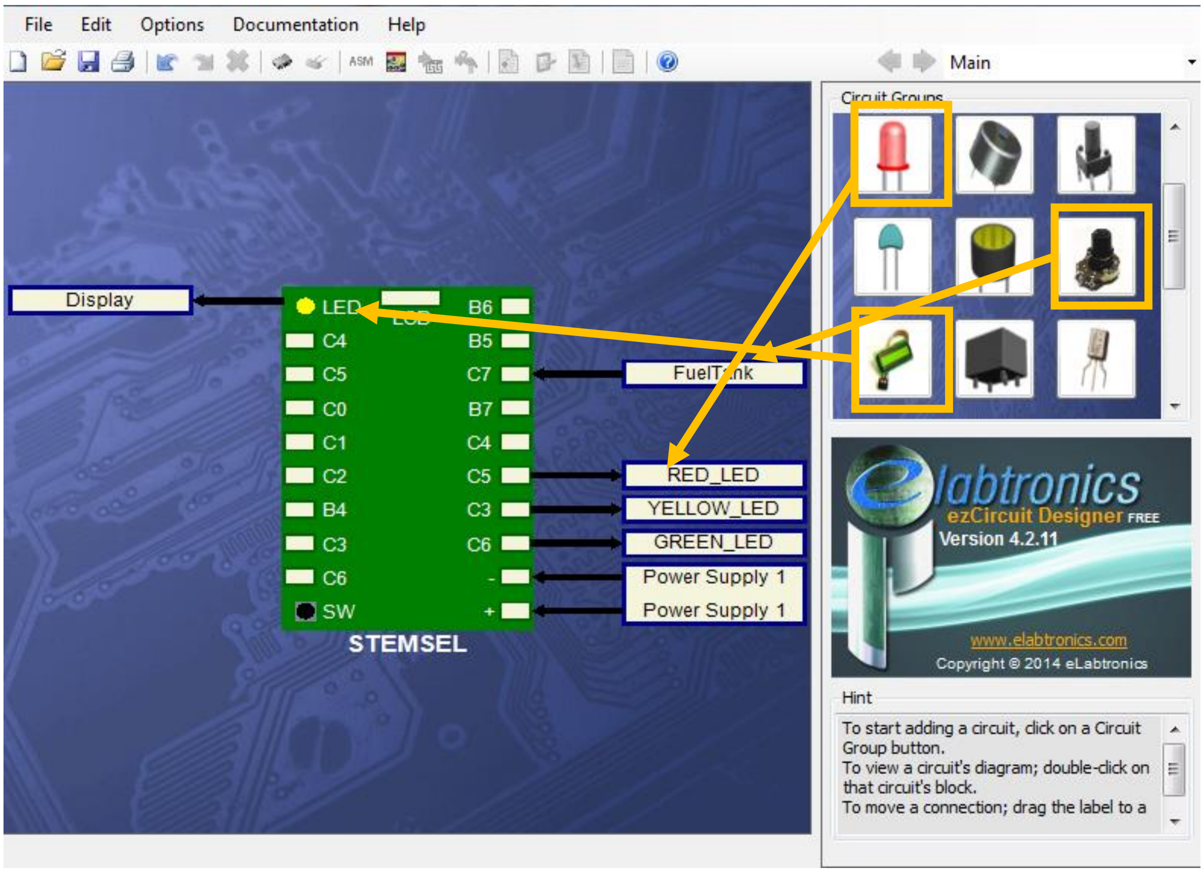


- If the fuel tank value if above the ‘high’ threshold – activate the green LED.
- If the fuel tank value is between the ‘high’ and ‘low’ thresholds – activate the yellow LED.
- If the fuel tank value is below the ‘low’ threshold – activate the red LED

Figure 2: Threshold graph

Design

Use ezCircuit Designer to construct an input/output (I/O) diagram. Add the potentiometer and LED to the recommended pins. This is indicated by pins turning light green when an icon has been clicked. Click on the LCD icon to add an LCD. Remember to right click on the names to rename the components.



To start adding a circuit, click on a Circuit Group button.
To view a circuit's diagram; double-click on that circuit's block.
To move a connection; drag the label to a

Figure 3: 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.

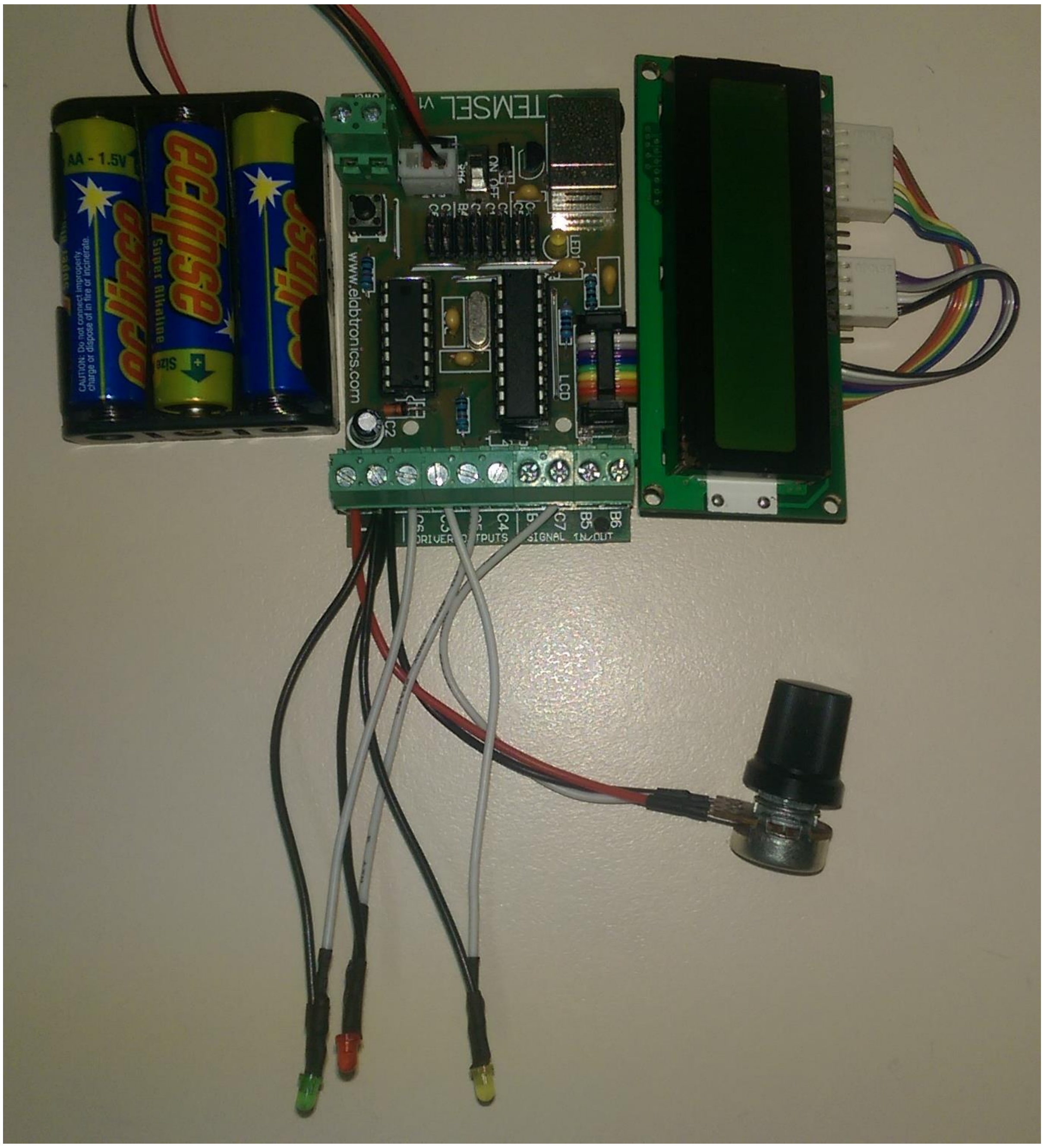


Figure 4: circuit

Programming

Click the 'Send to CoreChart' button so we can begin programming the board.

1. First there will be a test routine which checks all of the outputs are connected and working correctly. Click the 'Send program to chip' button after connecting the board to the computer using the USB cable. (Remember to be holding down the on-board push button while connecting!)

2. After the chip has been programmed remove the programming cable and turn the STEMSEL board on. The three LED's should briefly light up.

3. Delete the test routine so we can write our own program.

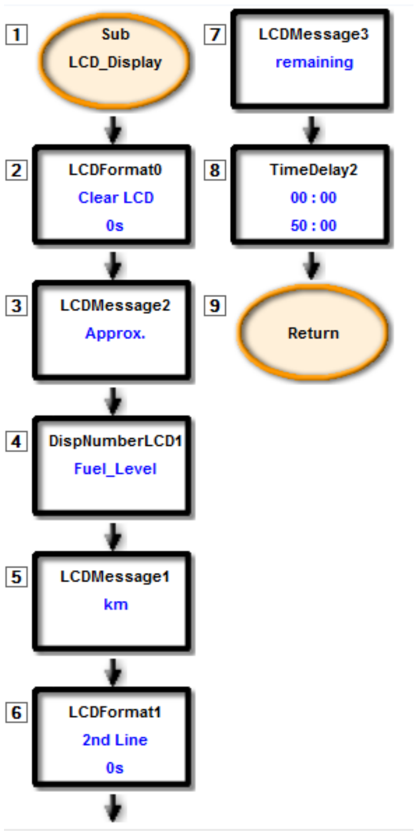
4. According to our plan, we need to take input from the potentiometer to determine the current fuel level. But first, our program needs to know what a low, medium and high fuel level actually is. Click on Numbers, and then Set_Number and create three numbers named: FUEL_HIGH, FUEL_MEDIUM and FUEL_LOW. Set these values to 255, 150 and 50 respectively.

5. Create a new address called Begin and place it after the Set_Numbers.

6. Next we need to actually take input from our potentiometer. Click on Inputs and then Analog_In. Once you have placed the button, double click on it and select FuelTank as our input pin, and save as FUEL_LEVEL

7. Now that we know the fuel level, we want to display this on our LCD. The first step is to clear the LCD. Do this by clicking on the LCD icon, and then placing a LCD format button. Double click on the LCD format button and under the mode section, select 'Clear LCD'. Set speed to 'no delay'.

8. Now we want to actually display the value as well as a meaningful message to the driver. This can be done by using a combination of LCDMessage and DispLCDNumber buttons. Try creating your own message!
(Hint: the LCD is small and can only display a relatively short message, across two lines. Try to display smaller words one button at a time. An example of a working display is shown alongside)

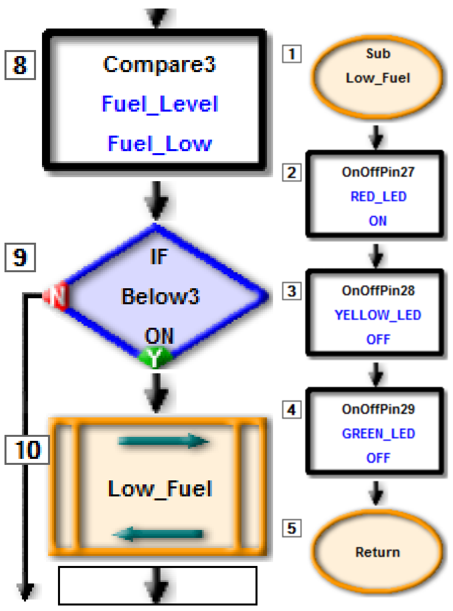


9. Once you have finished coding your LCD display, select all of the icons and group them into a group called LCD_Display

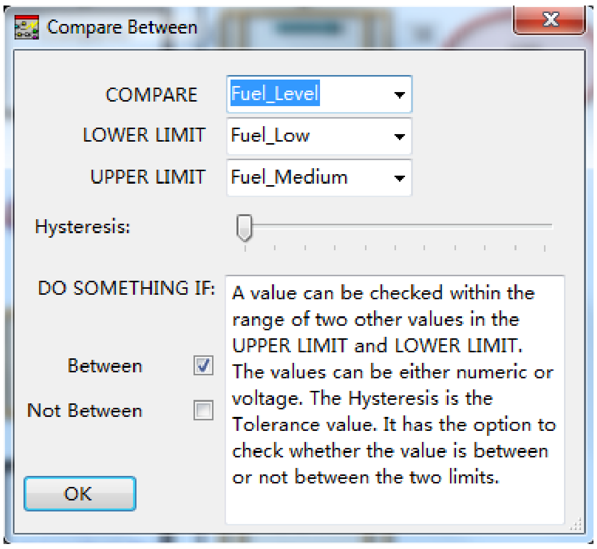
10. So now the LCD display should be working, but we also need a simpler and obvious visual indicator for the driver to know how much fuel the driver has left. This can be done using the LED's we attached to our board. Firstly, we need to know how much fuel there is left so we light up the correct LED.

11. Click on numbers and add a Compare button to our code. Double click the compare button and from Compare dropdown select 'FUEL_LEVEL' while from the Compare with dropdown select 'FUEL_LOW'. Tick the 'Below' box in the 'Do something if' section. This means that if the fuel level is below the low fuel threshold we will do something. But what do we want to do? The answer is to turn on an LED. In the case of low fuel we want to turn on the red LED.

12. Directly underneath the IF BELOW button, click on Outputs -> OnOffPin and add three on off pins. One of these pins will set RED_LED on, and the other two will turn YELLOW_LED and GREEN_LED off. Once you have added these buttons add them to a group called Low_Fuel. Your code should look something like the code shown alongside.



13. The next step in the program is to check the next threshold. To do this we will use a Compare between button. To do this click on Numbers, and add a CompBetween button. Double click on the button and select the following values from the dropdowns.


Compare Between

COMPARE: Fuel_Level
LOWER LIMIT: Fuel_Low
UPPER LIMIT: Fuel_Medium
Hysteresis:

DO SOMETHING IF:
Between ✓
Not Between ☐

A value can be checked within the range of two other values in the UPPER LIMIT and LOWER LIMIT. The values can be either numeric or voltage. The Hysteresis is the Tolerance value. It has the option to check whether the value is between or not between the two limits.

OK

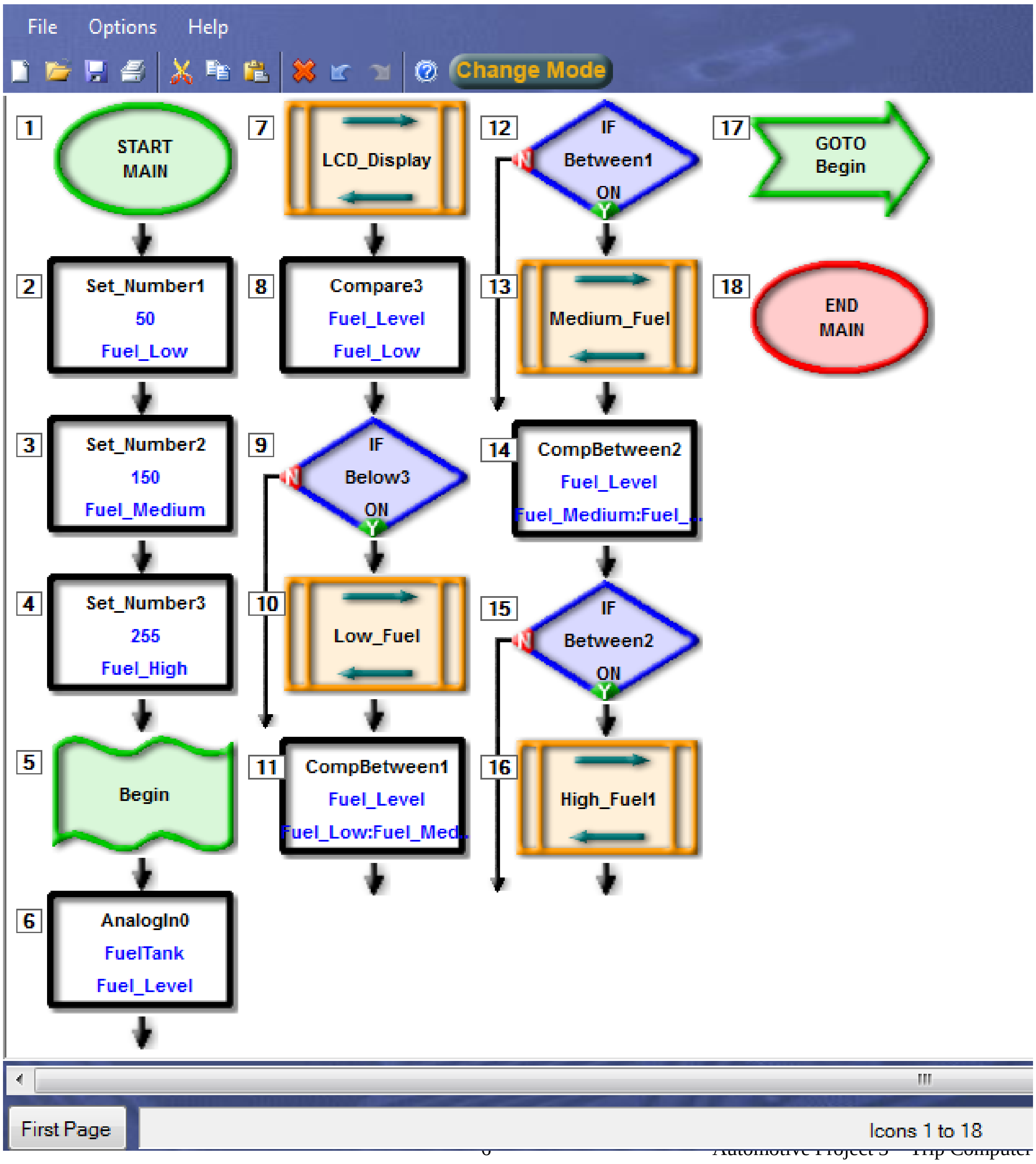
14. This means that if the fuel is between the low fuel and medium fuel thresholds, meaning we have a medium amount of

fuel left we will turn on the yellow LED. Insert three OnOffPins and use them to turn the yellow LED on, and all the other LED's off. Group them in a group called Medium_Fuel.

15. The final step is to make sure that the fuel level is at high, and then turn on the green LED. This can be done by using another CompBetween statement. This time we are comparing Fuel_Level and making sure it is between Fuel_Medium and Fuel_High.

16. If the Fuel_Level is within this range, similar to before we add three OnOff pins to turn on the green LED but turn off the other LEDs, and then group these in a group called High_Fuel.

The completed code should resemble the screenshot provided below. Screenshots of the subroutines are also provided.


File
Options
Help
Change Mode
1 START MAIN
2 Set_Number1 50 Fuel_Low
3 Set_Number2 150 Fuel_Medium
4 Set_Number3 255 Fuel_High
5 Begin
6 AnalogIn0 FuelTank Fuel_Level
7 LCD_Display
8 Compare3 Fuel_Level Fuel_Low
9 IF Below3 ON
N
Y
10 Low_Fuel
11 CompBetween1 Fuel_Level Fuel_Low:Fuel_Med..
12 IF Between1 ON
N
Y
13 Medium_Fuel
14 CompBetween2 Fuel_Level Fuel_Medium:Fuel_..
15 IF Between2 ON
N
Y
16 High_Fuel1
17 GOTO Begin
18 END MAIN
First Page
Icons 1 to 18

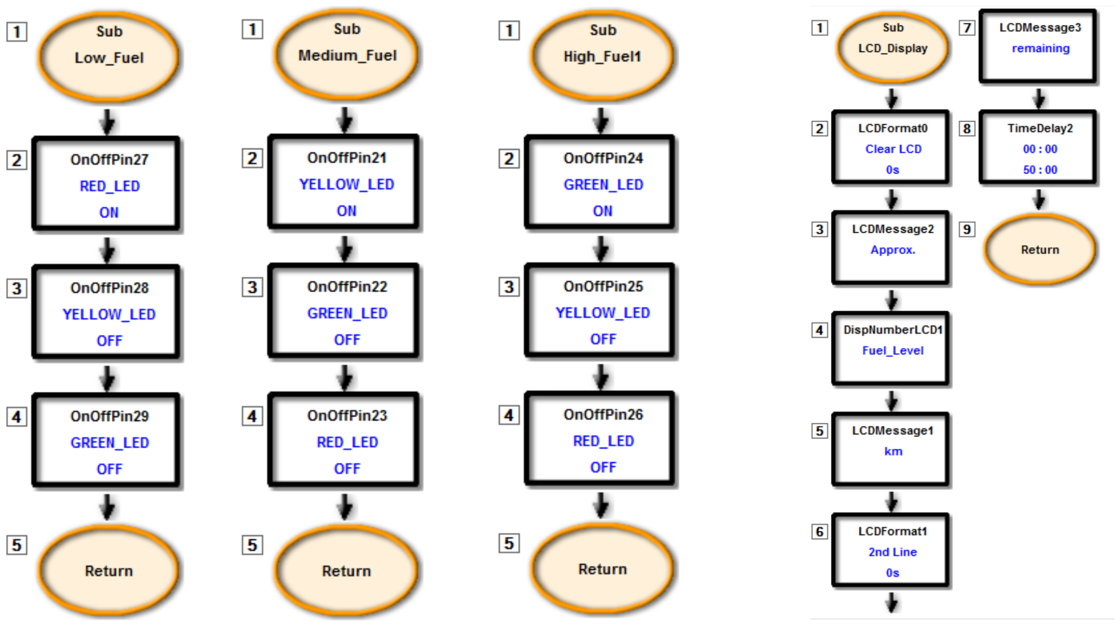


Figure 5: completed program

Activity/extension

1. Currently our Trip Computer can display a distance between zero kilometres and 255 kilometres. This is an unrefined number, limited by the range of the potentiometer. We want to polish this number make it a more accurate representation of how far a real car can actually travel? Is there a way to do this?

Summary

This project utilised microchips on the STEMSEL board to represent a 'trip computer' in a car. This trip computer makes life for the driver of the car a lot easier, as well as potentially other drivers as the trip computer has the capability of causing a driver to avoid a potentially dangerous situation by running out of fuel. The road is already a dangerous place and any steps which can be taken to reduce the risk of driving need to be taken.