STEMSEL Automotive Project 9 –Networking in Cars

Problem

When your parents are driving their car and need to turn their headlights on, all they have to do is press a button. How does the input from the button get all the way down to the headlights and tell them what to do? What if there are many different settings for the lights? How do the lights know which button has been pressed and how to react?

Background

The electronic systems in cars continues to become more and more complex, with modern cars now having a network of about 200 microchips all talking to each other. Modern cars are capable of performing a range of tasks from turning on and off lights and windshield wipers, to telling the driver when the car is about to reverse into an object. These systems are crucial for the safety of the car's passengers, so it's important that all of the microchips are working correctly together at all times.

In this project we will build a network by connecting two of the STEMSEL boards together and see if one can control the others board. We will be designing and programming a network consisting of two STEMSEL boards which can be used for a cars headlight system.

Ideas

Do we have anything in our kit that can represent a light, or something which can turn a light on or off? How can we make two STEMSEL boards work together and talk to each other?

Plan

To represent the car headlight we can use the yellow LED in our kits. We have chosen to use an LED instead of the lightbulb because modern cars use LEDs for their headlights as they last longer and are more power efficient. We will also need a way to switch the lights on and off. The push button from the kits will work well for this.

You have made a circuit like this in previous tutorials, however now we need the switch to be on one STEMSEL board, representing the driver turning the lights on, and the LED to be on another, representing the headlights of the car. To allow this to happen, the two boards need to be able to talk to each other. This can be done by connecting an output of one board, lets call it the Master board, with the input of the other board, lets call it the Slave board. This will allow the Master to talk to the Slave and tell it to turn the LED on. However the Slave will also need to talk to the Master to tell it that the LED is on, so the output of the Slave board has to be connected to the input of the Master board. A piece of wire is capable of making each of these connections. To make it easy to distinguish between the wires, perhaps use two different colours. Do not use red or black wires as they are reserved for positive and negative respectively.

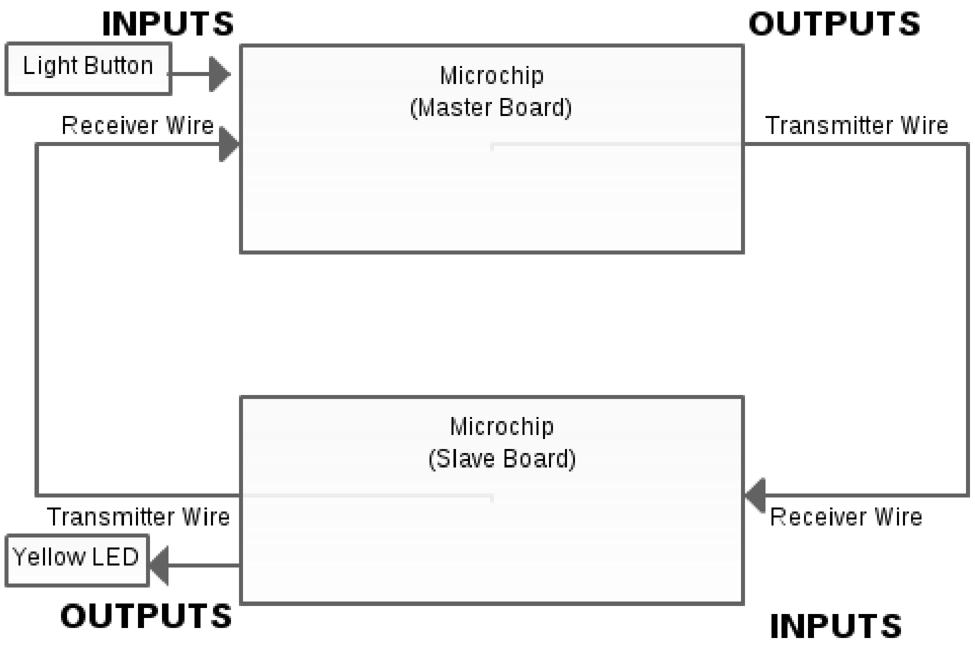


Figure 1: input/output chart

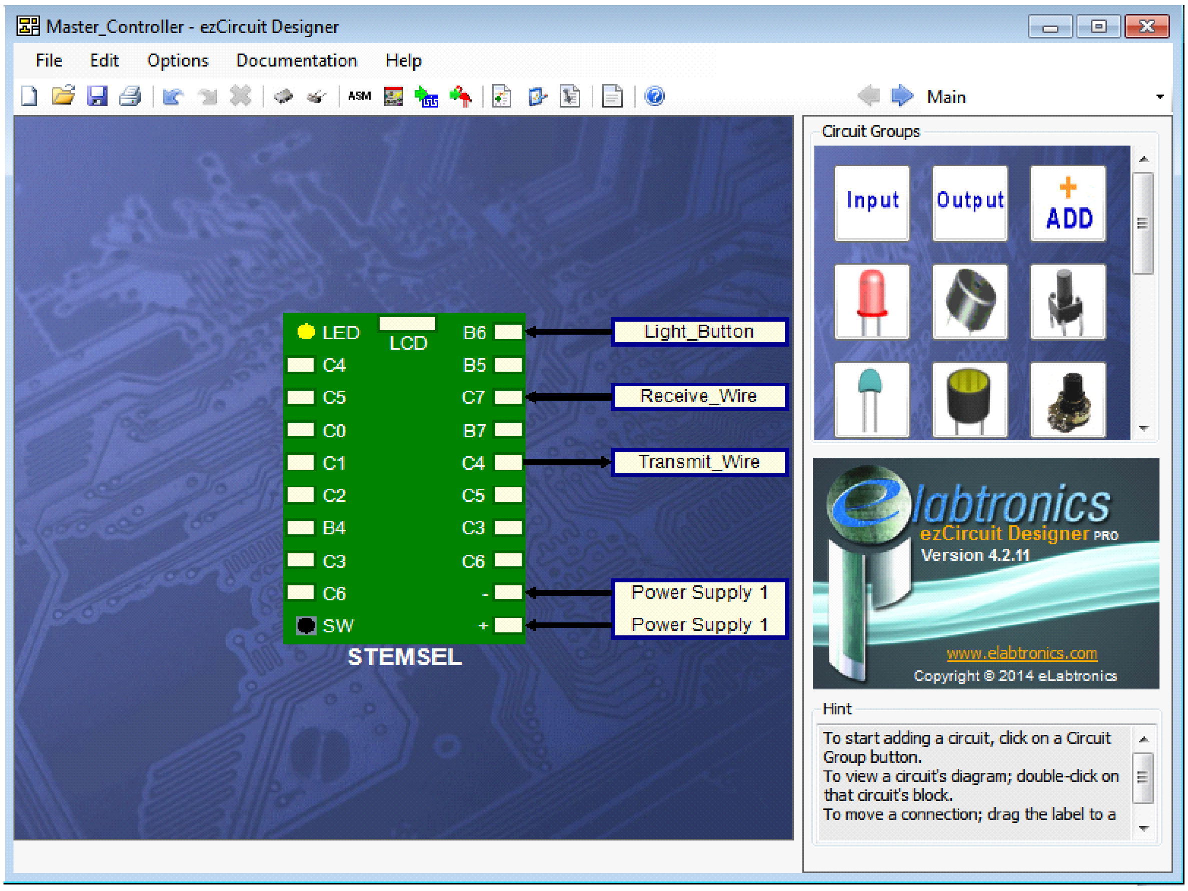
When the button is pressed on the Master board, it sends a signal from its output port to the Slave boards input port. The Slave board receives this signal, activates its output ports, including a signal back to the Master board to say that the message was received. This signal will either be a high 4V (the light is now on) or a low 0V (the light is now off).

As this project is the first where more than one STEMSEL board is needed, we will build it up in several steps. This means we need to open two copies of ezSystem!

Design

Board 1: Master Board

Open ezCircuit and create a new project called Master_Controller. From the plan, we can see that on this board we need to add a button to act as a switch, and transmit and receive wires. Use the Button icon to add the button to the circuit design. The wires don't have a special icon so these needed to be added in a different way. For the Receive_Wire click the input button, then select **Analog Input** from the circuit lists. For the Transmit_Wire click the Outputs button and select **PWM output** from the circuit list.

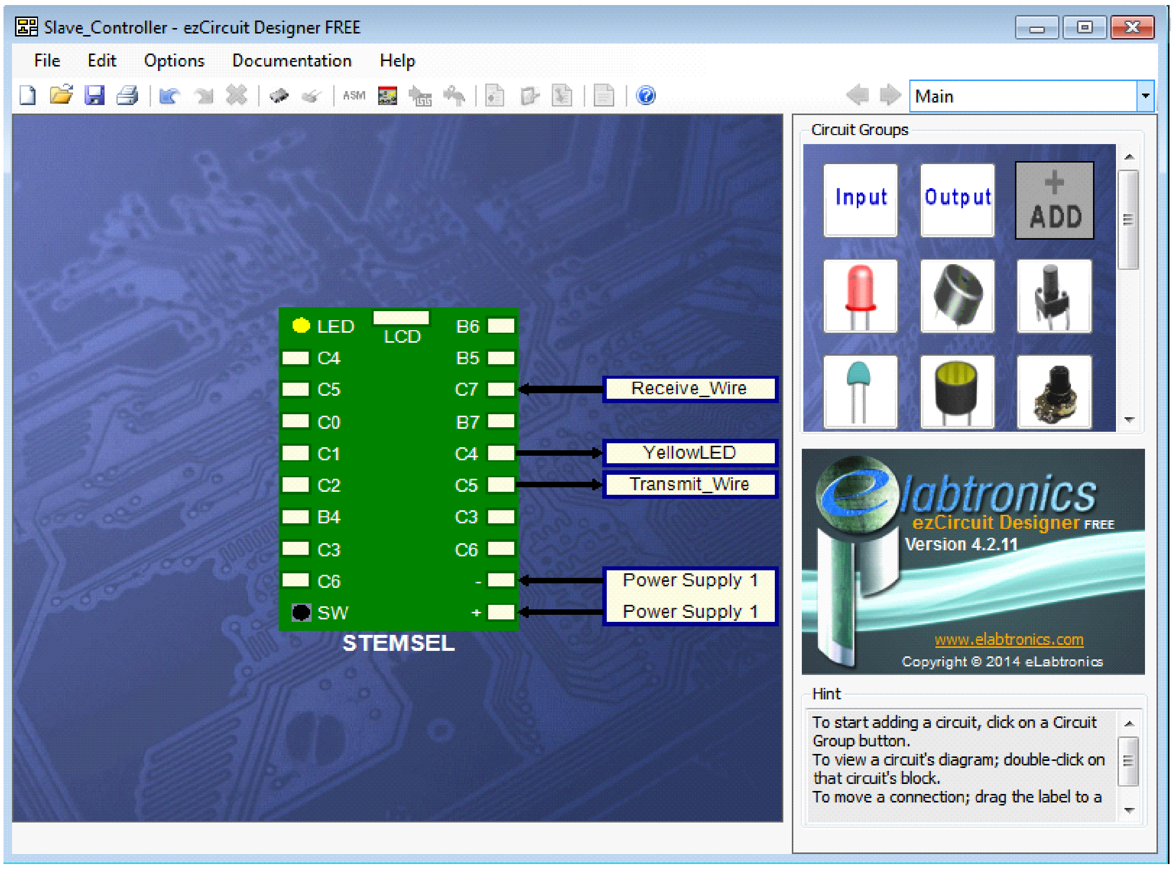


To start adding a circuit, click on a Circuit Group button.
To view a circuit's diagram; double-click on that circuit's block.
To move a connection; drag the label to a

Figure 2: Master Board Circuit Design

Board 2: Slave Board

Open another ezCircuit and create a new project called Slave_Controller. From the plan, we can see that on this board we need to add an LED to act as the car headlight, and transmit and receive wires. Use the LED button to add the LED to the circuit design. Follow the instructions outlined when designing Board 1 to implement the wires, but be sure to put them in the correct ports for this board.



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: Slave Board Circuit Design

Build the circuit

When you are finished the design, it's time to assemble the circuit using your kit.

Board 1: Master Board

Put the white wire from the button into B6, the red wire into the positive terminal and the black wire into the negative terminal. Put the input wire (the wire to go to the output of the Slave board) into C7 and the output wire (the wire to go to the input of the Slave board) into C4.

Board 2: Slave Board

Put the white wire from the yellow LED into C4 and the black wire into the negative terminal. Put the input wire (the wire to go to the output of the Master board) into C7 and the output wire (the wire to go to the input of the Master board) into C4.

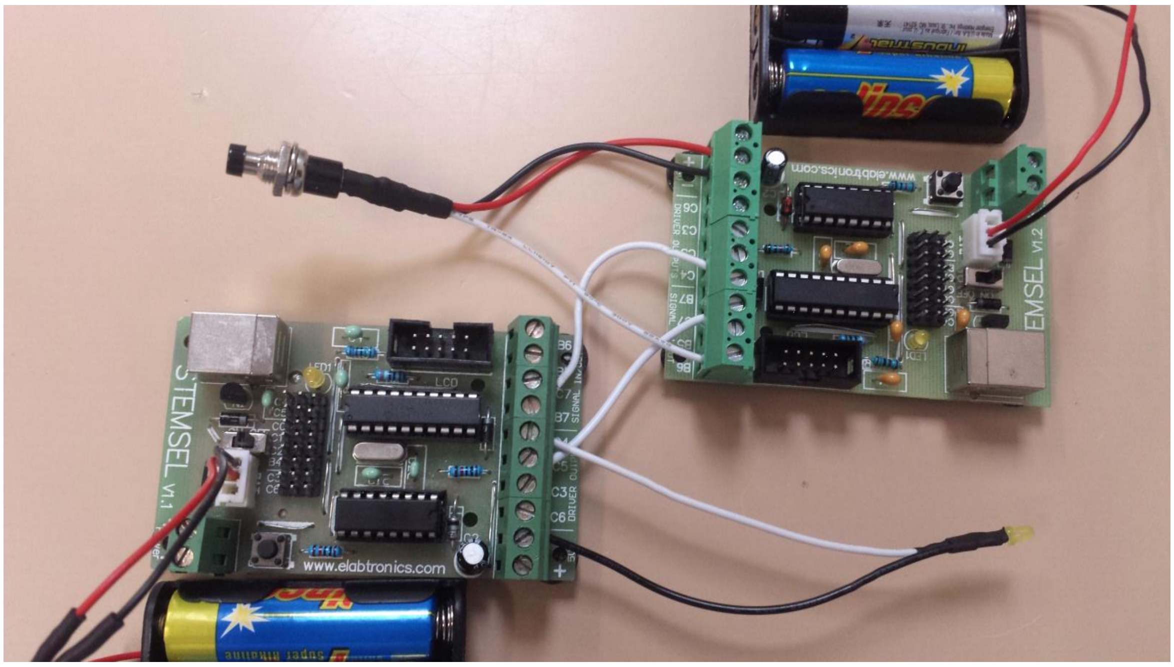


Figure 4: circuit

Programming

Once you have assembled the circuit and connected the two boards together, you are ready to program them. If you have two computers available, perhaps it would be easiest to program the Master board on one and the Slave board on the other. For both boards send the design to Corechart by clicking the "Send to Corechart" button. Then for the Slave board only click the "Send Program To Chip" button to program the microchip with the test routine for the LED. Once you have tested the LED, delete the test routine so we can write our own code.

We will start with the Master board. Here are some steps for reference:

1. Implement a loop so the program will only run fully when Light_Button is pressed.

2. Click on the "**Inputs**" button and select "**Analog_In**" in the "Icon Properties" window and add it after the Decision block. Double click it and store data from "**Receive_Wire**" as "**From_Slave**".

3. Use a Compare icon to compare "**From_Slave**" with 4V as mentioned in the plan to see if the light is on or off.

4. If "**From_Slave**" is greater than 4V, this means the light is on, so a press of the button now should turn the light off. To turn the light off we need to turn the "**Transmit_Wire**" off.

5. If "**From_Slave**" is less than 4V the light is off so it should be turned on. To turn the light on, we need to turn the"**Transmit_Wire**" on.

6. Add a "**GOTO START**" before "**End Main**".

Now we will look at the Slave board. Here are some steps for reference:

1. Click on the "**Inputs**" button and select "**Analog_In**" in the "Icon Properties" window and add it after the "**START MAIN**" block. Double click it and store data from "**Receive_Wire**" as "**From_Master**".

2. Use a Compare icon to compare "**From_Master**" with 4V to see if the output of the Master board has driven "**From_Master**" high by a button press.

3. If "**From_Master**" is greater than 4V turn the LED on and the "**Transmit_Wire**" on. Create a group called "Lights On" and add these two blocks.

4. If "**From_Master**" is below 4V, turn the LED off and the "**Transmit_Wire**" off. Create a group called "Lights Off" and add these two blocks.

5. Add a "**GOTO START**" before "**End Main**".

However now look at the full Master board program diagram shown in figure 5a. In the design you will notice that there is a delay of 20ms right after the Light_Button decision block. Can you think of why we have needed to introduce this into our circuit? You may only be pressing the button once but the STEMSEL chips are so fast they are processing that one button press as many, many button presses(how many approx?). This means that when you let go, we won't know how many times the processors think the button has been pressed. So we introduce a small time delay which only allows the switch to turn the light on or off every 20ms.

So the final step for the Master board is to add this time delay. Here is the last step of the Master board for reference:

> 7. Click on the "**Numbers**" button and select "**Time_Delay**". Add this time delay after the Light_Button decision block. Double click on it and enter 20 into the "**Hundredths**" box.

Note: If you are having trouble and can find an error in you could consider lowering the comparison voltage.

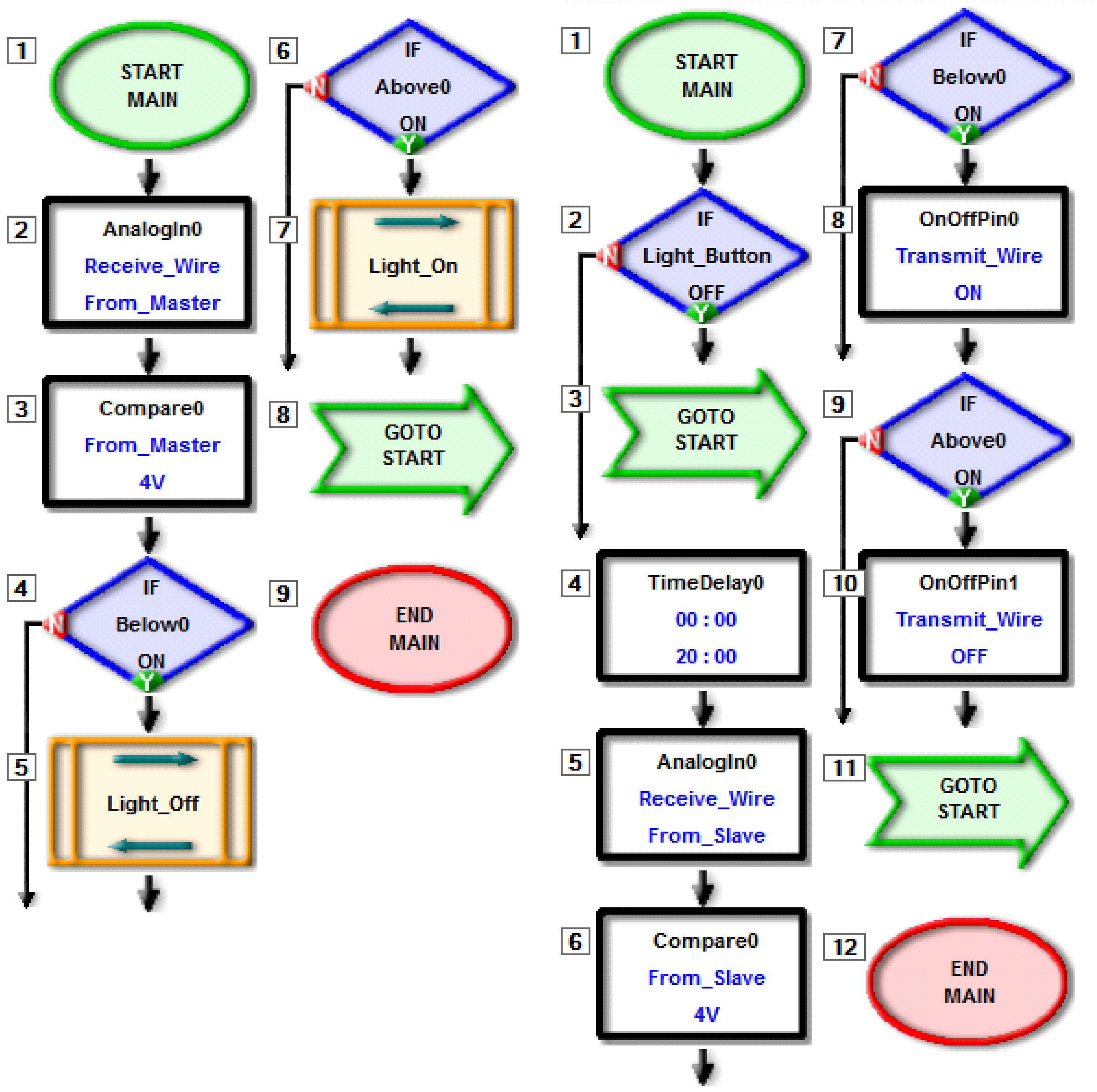


Figure 5a: Completed Program for Master Board

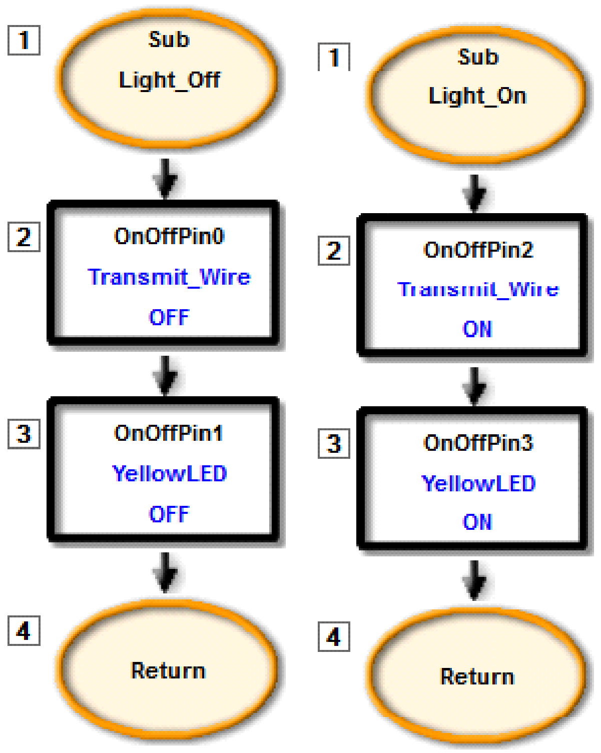


Figure 5b: Completed Program for Slave Board

Now that your program is complete, you can click "Send Program To Chip" for both the Master and Slave boards. Be careful that you are sending the correct program to the correct board. After you send the program to the microchips, press the button on the Master board once and the light on the Slave board should turn on. Press the button again and the light should turn off.

Activity/extension

How many STEMSEL boards can you connect together? Join with another group and see if you make one Master board control many Slave boards. This will mean that any board connecting with two other boards will require two inputs and two outputs, one of each for both boads.

Summary

This tutorial has allowed us to see how processors are able to talk to each other to form networks which can perform a wide range of tasks. We have developed a headlight system for a car which required two processors, just imagine how complex a system must be for a car which has approximately 200 processors talking to each other.