STEMSEL Intermediate Project 7: Combination Lock

Problem

We want to make an electronic lock using our STEMSEL controller board. How can we use it to make a combination lock?

Background

As writer Allan Poe said, everyone has secrets. People are always trying to find new ways to keep our secrets safe, and will continue to do so in the future. Not only our secrets need to be kept safe; our property, private information, and communication security also need to be taken care of. In the past people used complex mechanical methods like lockers, locks, keys and huge chains to guard precious property. Now we still use locks, but a different kind of locks. This lock is so small that people have to insert them into plastic cards to keep them, and they are so important that without it the whole modern communication network would be broken down. This lock is also so tough to crack that it would take the most advanced technology millions of years to hack it, yet is so cheap that it can be used by everyone worldwide.

The applications of these locks are very common in our daily life. You may have seen a passenger use a metro card to pay their bus fare, or a driver use a remote control key to unlock their car. But how do the ticket machines identify which card should be connected to which account so people don't use anyone else's balance, and how can a car remote receiver recognize if it is the right key so it won't open its door to everybody who has a remote? Well that is because of RFID chips.

RFID chips are a kind of tiny microchip that usually doesn't have a power source, but it can respond to an electromagnetic field generated by a RFID reader, which can then read the data stored on the chip. If it is the right key (microchip), the RFID reader will execute commands, for example light a green LED and withdraw one dollar from the passenger's account balance, or unlock the door for the driver. Amazing? This is what electronic engineering can do.

Figure 1: some examples of RFID devices

There are three basic modules in an electronic lock; they are input module, decision module and output module. The user provides a sort of key to the input module. The input module transfers this signal to the decision module, which will look at this signal and decide if this is the right key, and then send the result of that decision to the output module. The output module then executes the command to open the door if the input is correct, otherwise it will do nothing or even alert the police.

Ideas

What are some different kinds of locks? There is no keyhole on our STEMSEL controller board, so how can we unlock our lock? What will be the 'key'? We need to be able to open our lock, but we don't want others to be able to open it. How difficult should we make it to open our lock? Since the microchip works at twelve million instructions per second, how will the microchip know when to read the code?

Plan

In this project we will design, program and assemble an electronic combination lock. The combination lock should turn on a green LED if the right combination is entered by a potentiometer.

The potentiometer will be our 'key' which we can use to enter the correct code. This code should be simple enough for you to put into open the door, but difficult for someone who doesn't know the code to guess. Since the potentiometer can create different voltages, we will use voltage ranges as our code. We will use the pushbutton on the board to enter the code, then the microchip will check it to see if it is correct. If so, it should activate an output like a LED to indicate a door opening. If the code is not correct, the lock should be reset which we will show by turning on the on-board yellow LED.

You are encouraged to set your own combination, but please note that the range of our knob is from 0V to 5V. The most anticlockwise position corresponds to 0V and the most clockwise position corresponds to 5V. We don't have a number scale on our dial, so you want to set the combination as a range rather a specific number.

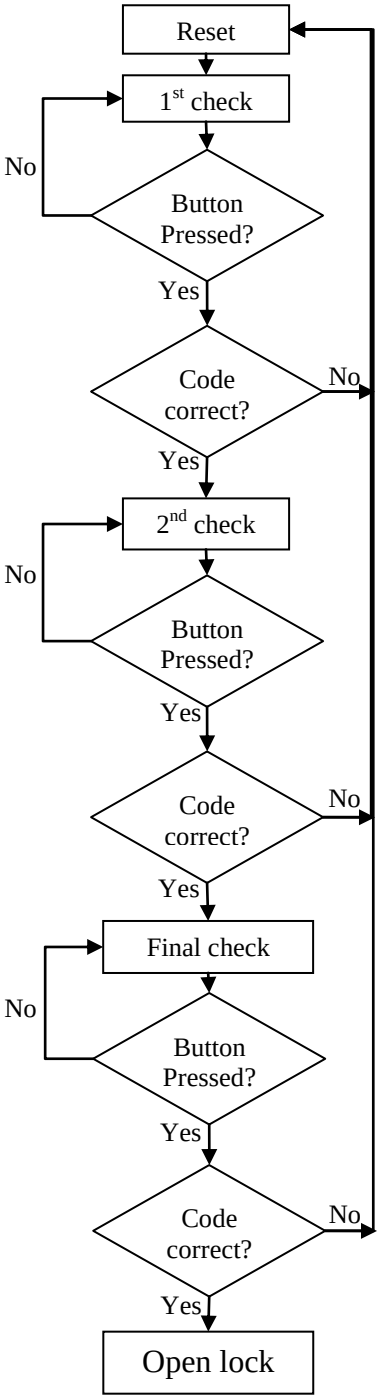

Figure 2: program flowchart

Design

According to the plan, we will need a green LED as an indicator, a potentiometer as an input, and an awesome STEMSEL controller board as a decision module.

Now it's time to open ezCircuit Designer to arrange these components, once you finished design it would like this:

(It's OK to connect a component to another pin as long as the pin was light green when you click the icon.)

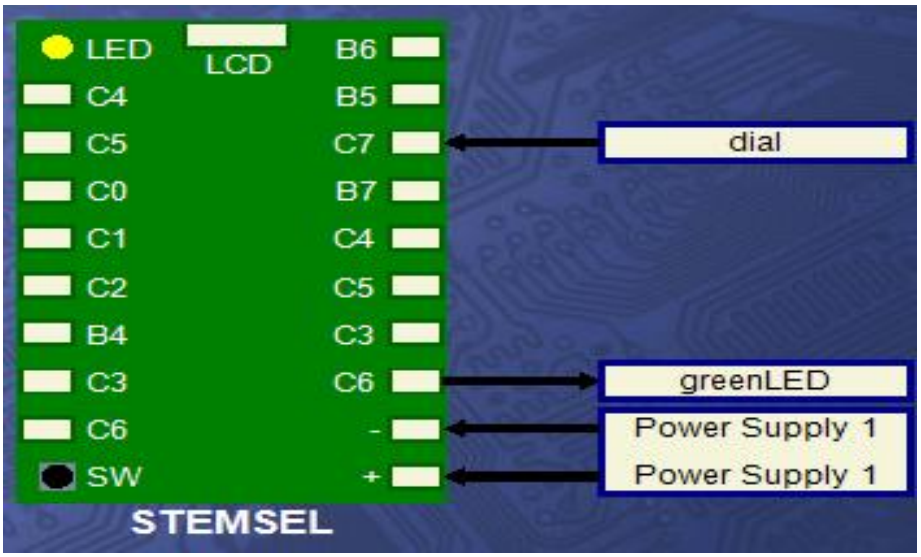

Figure 3: circuit design

Build the circuit

When you have finished, it's time to assemble the circuit using your kit. Put all of the black wires into the negative (-) pin and insert the red lead of the potentiometer into the positive (+) pin. The white wires go into the ports specified in the design.

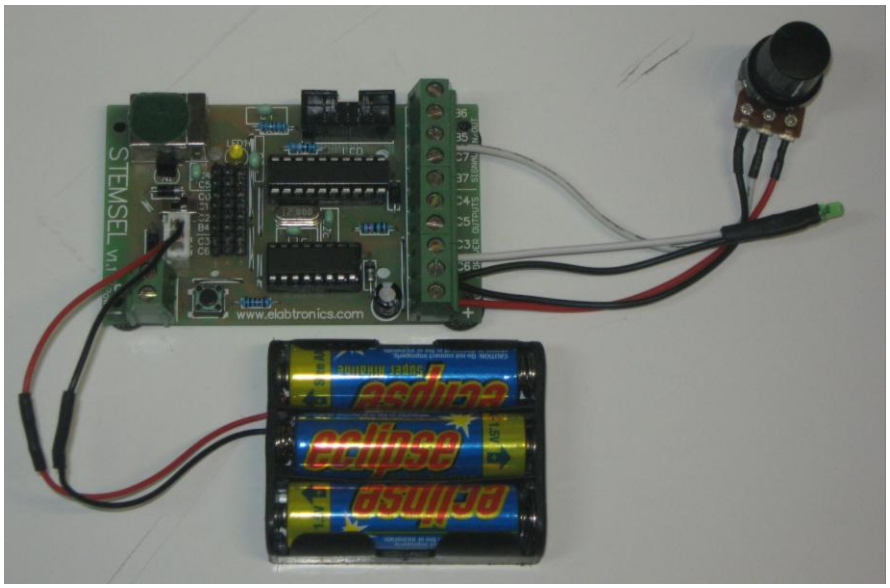

Figure 4: circuit

Programming

Once you have assembled the circuit, send the design to Corechart by clicking the "Send to Corechart" button, then click the "Send Program To Chip" button to program the chip with the LED test program. Once you have verified the LED is working, delete the test routines so we can start writing our program.

The most important thing in our program is to set the right combination, and don't forget to insert some time delays after entering each number.

Here are some steps for reference:

1. The start of the program should be the reset state mentioned in the plan. Set the on-board yellow LED ON, and turn it OFF after 2 seconds to indicate the system is reset.

2. Write a loop to check if the pushbutton is pressed using an Address and a Decision icon to check if the pushbutton is ON (the button is off when it is pressed). If it is not pressed, go back to the address. If the button is pressed, the program should continue and see if the combination is correct.

3. The address, Decision and GoTo icons can all be grouped together into a subroutine called **waitforbutton**.

4. Take the input from the dial using an "**Analog_In**" icon, and store the value as "dailvoltage".

5. Compare the "dailvoltage" and the first number of the combination. In this case it has been set as between 2.5V and 3.5V. This can be done by adding a CompareBetween icon (Numbers -> CompareBetween), then entering 2.5V and 3.5V in "LOWER LIMIT" and "UPPER LIMIT" boxes respectively, and finally ticking the "Not Between" check box in the "DO SOMETHING IF" area. Remember, you should come up with your own code.

6. If the dial is not between 2.5V and 3.5V, then the wrong value has been entered and we want the program to go back to the start, so the GoTo START icon can be left where it is. If the correct value has been entered then the program should continue.

7. Add another time delay (2 seconds), then another "**waitforbutton**" subroutine.

8. Just like in step 3, read the input from the dial and save it as "dailvoltage".

9. Compare the new "dailvoltage" with the second value of the combination. In the example code this is above 4V, though again you can set your own value.

10. If both the values entered are correct, turn on the green LED to show the lock is now unlocked.

Now your program should look like this:

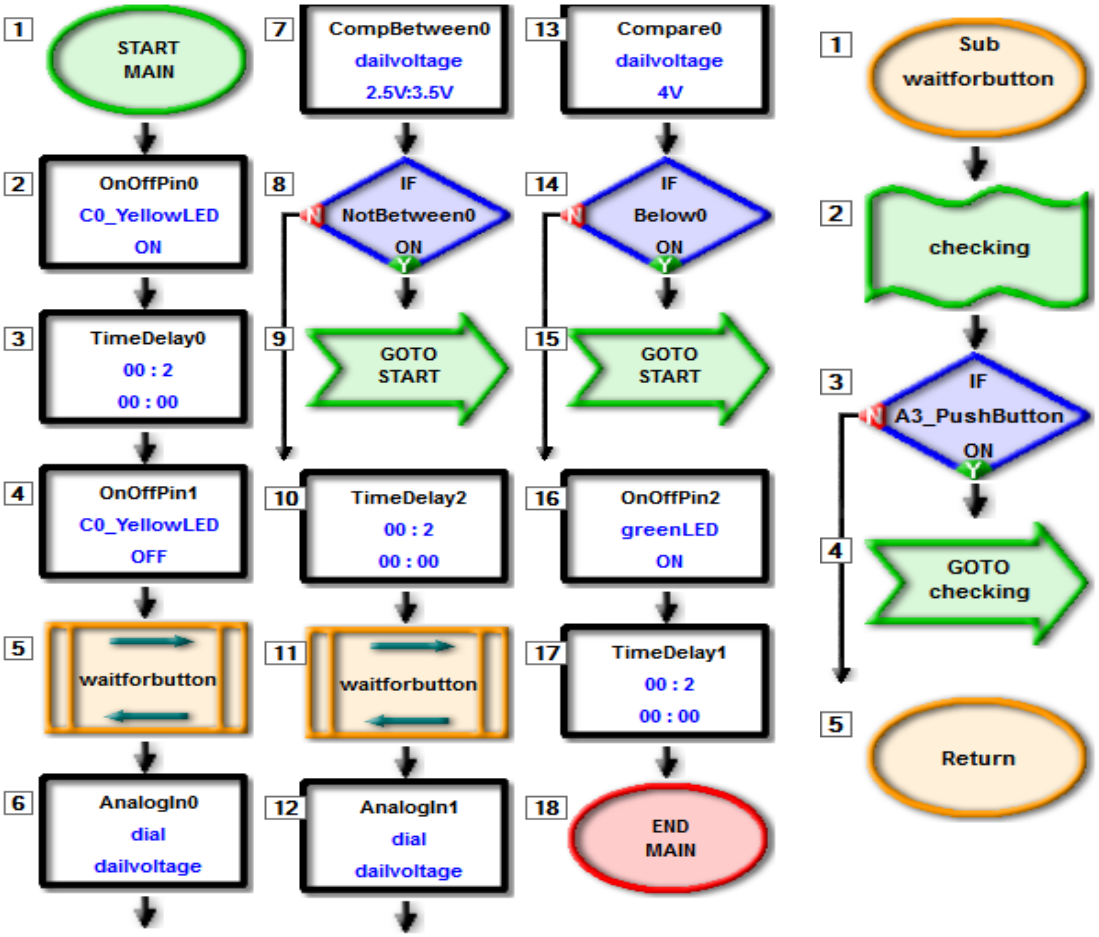

Figure 5: example program

After you send the program to the chip, you can adjust the potentiometer to try your lock. In the example program, the right combination is adjusting the knob to the middle than turning it to the most clockwise position. Now the green LED should be turned on.

Activity

If your lock works properly, set some different combinations and invite your friends to try and open your lock. You can make the code longer (i.e. have more values than need to be entered), or make the voltage ranges bigger or smaller, but remember you still need to be able to unlock it yourself!

To make your code longer, you will simply need to add more of the icons we used before: a **waitforbutton** subroutine, an **AnalogIn** icon to read the new dial value, a **Compare** or **CompareBetween** icon to check if the new value is correct, then a 2 second time delay.

In the example, ranges of about 1V were used (between 2.5 and 3.5 volts, and between 4V and 5V). What happens if you use different voltage ranges, for example if it was only 0.25 volts, or 3 volts? Would it be easier or harder to open your lock?

Summary

By using a microchip, a LED and a potentiometer we made a combination lock. During this project, we learned the basic structure of an electronic lock, and the core principle of the decision module comparing the input value with the correct value. It is simple, yet can be developed into more advanced forms and applied into various fields.