runlinc Intermediate Project 6: Combination Lock (STEMSEL version)

Contents

Introduction

Problem

We want to make an electronic lock by using runlinc on the STEMSEL board. How can we use it to make a combination lock?

Background

As the writer Allan Poe said, everyone has secrets. People are always trying to find new ways to keep their secrets safe and will continue to do so in the future. Not only our secrets need to be kept securely; our property, private information and communication security are also needed to be taken care of. In the past, people used complex mechanical methods like lockers, locks, keys and huge chains to guard valuable property. Now we still use locks, but a different kind of lock. This lock is so small that people must insert them into plastic cards to keep them, and they are so important that without them, 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 widespread 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, thereby people don't use anyone else's balance, and how can a car remote receiver recognise if it is the right key thus 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 an 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 an 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 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 in to open the door, but difficult for someone who doesn't know the code to guess. Since the potentiometer can create different values as we turn the knob, we will use the values within the range for the code.

We will use runlinc to submit the value and then allow runlinc to check if the submitted value is correct. For each input level, if the value is correct, the Yellow LED will light up to signify the preparation for the next level of the combination. If the combination is correct, the Green LED will light up. If the code is not correct, the lock should be reset, which we will show it by using the Red LED. Additionally, in this project, we will also write a module that will help us change a combination for the lock dynamically.

The Program Flow Chart on the right side is a visual representation of how the program will work, and the I/O that will be used will be shown below.

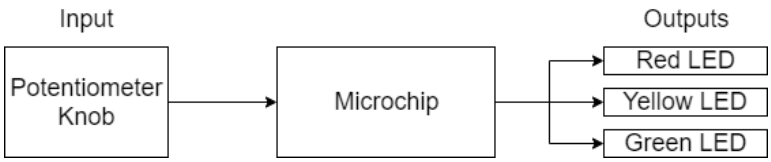

Figure 3 I/O Plan

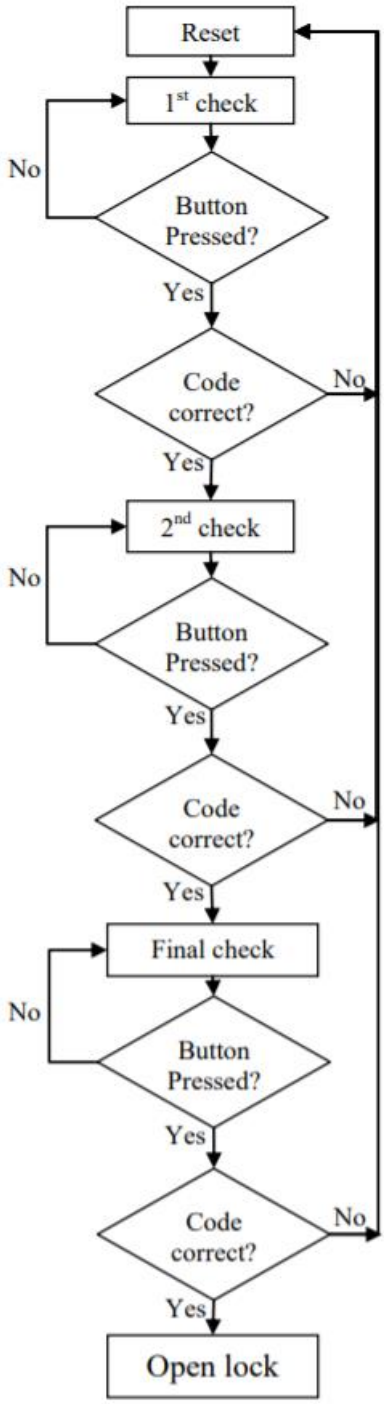

Figure 2 Program Flow Chart

runlinc Background

Runlinc is a web page inside a Wi-Fi chip. The programming is done in the browser and sent to the chip over Wi-Fi. The runlinc web page inside the Wi-Fi chip will command the microchips to do sensing, control, data logging Internet of Things (IoT). It can predict and command.

Part A: Design the Circuit on runlinc

Note: Refer to runlinc Wi-Fi Setup Guide document to connect to runlinc
Use the left side of the runlinc web page to construct an input/output (I/O).

In our circuit design,

- C7 -> ANALOG_IN: Dial
- C3 -> DIGITAL_OUT: Green
- C4 -> DIGITAL_OUT: Red
- C5 -> DIGITAL_OUT: Yellow

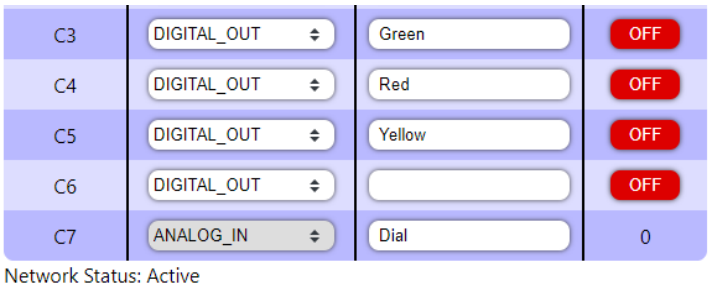

Figure 4 Expected I/O Configuration

Part B: Build the Circuit

Use the runlinc I/O to connect the hardware. Remember that turning the screws clockwise will close the clamps and turning the screws anticlockwise will open them. All black wires should go in the negative (-) terminal, red wires go in the positive (+) terminal, and white wires go in the terminal we designated in the runlinc web page port.

Figure 5 Circuit

Wiring Instructions

- Connect All White Wires to their Respective Pins
  - Potentiometer -> C7
  - Green LED -> C3
  - Yellow LED -> C5
  - Red LED -> C4
- Connect All Black Wires to negative (-) terminal.
- Connect Red Wire to positive (+) terminal.

Part C: Program the Circuit

This project will mainly use the HTML block and the JavaScript block in runlinc. We will first focus on HTML to set-up the basic structure of the application.

HTML

Be sure note about the “id” attributes for each HTML elements as they will be used for the JavaScript portion.

1. We will first align all contents to the centre, at the same time we’ll add a header.

```
<div style="text-align:center;">
   <h1>Security Lock via runlinc</h1>
</div>
```

2. After the header element, we’ll add our first paragraph element block. The block will contain three buttons respectively named “Start”, “Submit”, and “Reset”. Below the buttons will show the value of the dial, and below the dial, the value will explain the different LED light indicator meanings.

```
   <p>
      <button id="Start" onclick="start();">Start</button>
      <button id="Submit" style="visibility: hidden"
         onclick="hasClicked = true;">Submit Dial</button>
      <button onclick="clearlock();">Clear/Lock</button>
      <br> Current Dial Value: <font id="CurrentDial"></font>
      <br>
      <h1>LED Light Indicator</h1>
      Green = Sucess; Yellow = Preparing; Red = Clearing/Locking;
   </p>
```

3. Afterwards, we’ll add another paragraph which will contain elements to show the lock status. The Status block will contain a gauge and status text.

```
   <p>
      Progress
      <br>
      <meter id="Gauge" value="0" min="0" max="3" style="width:25%;"></meter>
      <br>
      <font id="Status"></font>
   </p>
```

4. Then we'll add the last paragraph block, which will contain settings for the lock. The settings will contain values for the lock combination. We'll be using tables for formatting and input element to store values.

```
<p id="Settings">
    Settings: Please note that values will be truncated into values between 0-
    255.
    <br>
    <table id="SettingsTable" align="center">
        <col>
        <th>Enter First Dial Value</th>
        <td><input type="text" id="dialValue_1" value="0"></td>
        <td><font id="dial_1"></font></td>
        </col>
        <col>
        <th>Enter Second Dial Value</th>
        <td><input type="text" id="dialValue_2" value="0"></td>
        <td><font id="dial_2"></font></td>
        </col>
        <col>
        <th>Enter Third Dial Value</th>
        <td><input type="text" id="dialValue_3" value="0"></td>
        <td><font id="dial_3"></font></td>
        </col>
        <col>
        <th>Enter Tolerance Value</th>
        <td><input type="text" id="tolerance"value="0"></td>
        <td><font id="tolStored"></font></td>
        </col>
        <col>
        <td rowspan=2><button id="Store" onclick="store()">Store</button></td>
        </col>
    </table>
</p>
```

Now we will focus on the JavaScript part.

JavaScript

1. We'll start by establishing the following global variables that will be used throughout the program and is not limited within functions. As JavaScript is a single thread programming language, where the instructions are followed step by step, we will have to use setInterval() to continuously retrieve data without freezing the entire program.

```
var dial = new Array(0,0,0);    //Array to store dial values.
var tolerance = 0;              //To store the tolerance of dial values
var currDial;                   //To store current dial value.
var currDialFunc = setInterval(retrieveCurrDial, 100);          //Current dial retrieval.
var hasClicked = false;         //To check whether submit button has been clicked.
var waiting;                    //To check if we are waiting for the submit click.
```

2. We'll then write a function that will store the combination from the setting to the respective variables. Note: in the programming world, what we are doing is flawed security-wisely but this method is used to demonstrate this project's idea.

```
async function store(){
  // The loop is to store dial value and show the stored value.
  for(var i = 0; i < dial.length; i++){
    dial[i]=check(parseInt(document.getElementById("dialValue_" + (i + 1)).value));
    document.getElementById("dial_" + (i + 1)).innerHTML = dial[i];
  }
  // Stores tolerance and show the stored value and show that values are stored.
  tolerance = check(parseInt(document.getElementById("tolerance").value));
  document.getElementById("tolStored").innerHTML = tolerance;
  document.getElementById("Status").innerHTML = "Setting Stored.";
}
```

3. You may have noticed the check(); function; in the previous step. check(); is our own function to cap any values entered for the combination that is exceeding the range.

```
function check(value){
  if(value > 255){ return 255; }
  if(values < 0){ return 0; }
  return value;
}
```

4. When the start button is clicked. It will hide itself and settings block. Then we'll jump to the unlocking process where 0 is the stage level.

```
async function start(){
  document.getElementById("Start").style.visibility = "hidden";
  document.getElementById("Settings").style.visibility = "hidden";
  document.getElementById("SettingsTable").style.display = "none";
  unlockProcess(0);
}
```

5. The unlock process will turnOn() the Yellow LED for 2 seconds then turnOff() it. Then, we will reveal submit button to be allowed to be clicked with a status text to instruct users on what to do. We'll write the waiting function later.

```
async function unlockProcess(index){
  // Flashes Yellow LED for 2 seconds.
  turnOn( Yellow );
  await mSec( 2000 );
  turnOff( Yellow );
  // Reveals submit button and changes status text.
  document.getElementById("Submit").style.visibility = "visible";
  document.getElementById("Status").innerHTML = "You may start dialing, click
submit when you're ready.";
  waiting = setInterval(function(){waitForButton(index);}, 100);
}
```

6. We will continue from the previous unlocking block after the submit button has been clicked. Once the button has been clicked, the submit button will be hidden. Then we'll check if the current dial is within the acceptable range to continue to the next stage of the lock combination. If not acceptable, the lock will be cleared, and the user will have to start over. After checking, we'll have to check if the current stage is the last. If it is not the last stage, continue to the next stage. If it is the last stage, we'll hide the submit button and reveal the setting panel and notify the user that they have unlocked the lock.

```
function unlockProcessContinue(index){
  document.getElementById("Submit").style.visibility = "hidden";
  currDial = analogIn( Dial );

  // Check if the current dial is within acceptable range.
  if( dial[index] - tolerance <= currDial && currDial <= dial[index] + tolerance){
    document.getElementById("Gauge").value = index+1;

    // If it is acceptable, then check if the accepted current stage is not the last
stage of
    // the combination.
    if( index < dial.length - 1 ){

      // If the current stage is not the last stage, proceed to the next stage.
      document.getElementById("Status").innerHTML = "Onto the next Pin";
      unlockProcess(index+1);
    }else{

      // If the current stage is the last stage, then notify the user and reveal the
settings
      // panel and hide the submit button.
      document.getElementById("Status").innerHTML = "You have unlocked the
lock!";
      document.getElementById("Submit").style.visibility = "hidden";
      document.getElementById("Settings").style.visibility = "visible";
      document.getElementById("SettingsTable").style.display = "initial";
      turnOn( Green );
    }
  }else{

    // If the current dial is not acceptable, signify user and clear the lock.
    document.getElementById("Status").innerHTML = "Failed to enter an
acceptable Dial";
    clearlock();
  }
}
```

7. We'll now write the function when we are clearing/locking the lock. It's mainly putting everything back to its original condition except we don't change anything of the setting's variables.

```
async function clearlock(){
  turnOn( Red );
  await mSec( 2000 );
  turnOff( Red );
  document.getElementById("Status").innerHTML = "Reverting to intial condition";
  document.getElementById("Start").style.visibility = "visible";
  document.getElementById("Submit").style.visibility = "hidden";
  document.getElementById("Settings").style.visibility = "hidden";
  document.getElementById("SettingsTable").style.display = "none";
  document.getElementById("Gauge").value = 0;
  turnOff( Green );
  document.getElementById("Status").innerHTML = "Lock is Cleared (Except
Settings) and is Locked.";
}
```

8. Near to the last, let us write the waiting function that will check if the submit button is clicked. If it is clicked, we'll proceed to the latter half of the unlocking process.

```
function waitForButton(index){
  if(hasClicked){
    clearInterval(waiting);
    hasClicked = false;
    unlockProcessContinue(index);
  }
}
```

9. Finally, let's write the function that will continuously retrieve the value of the dial.

```
function retrieveCurrDial(){
  currDial = analogIn( Dial );
  document.getElementById("CurrentDial").innerHTML = currDial;
}
```

If you followed each step and can understand what each line of code does, then you should be able to produce the following page.

Check for Appendix to check entire code.

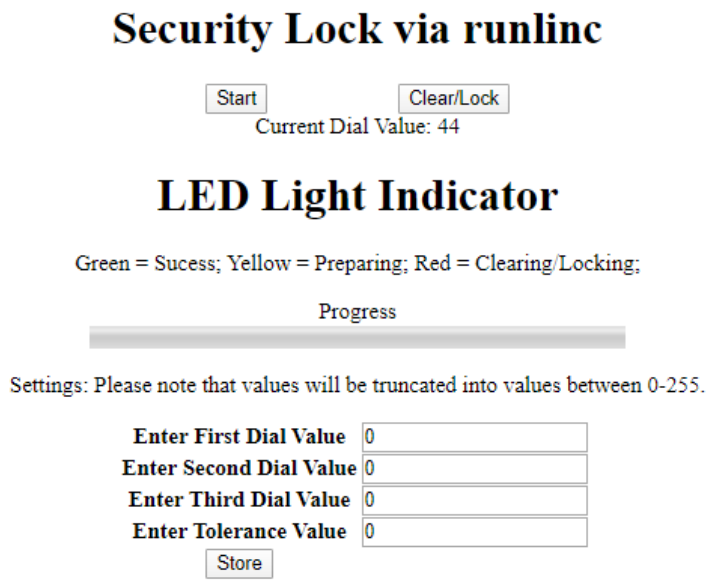

Figure 6 Expected HTML Page Result

Extensions

Try to edit the code such that you can add more stages/elements to the combination of the lock on the webpage. Hint: You can look more into JavaScript arrays.

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.

Appendix – Expected Code

HTML

```
<div style="text-align:center;">
   <h1>Security Lock via runlinc</h1>

      <p>
            <button id="Start" onclick="start();">Start</button>
            <button id="Submit" style="visibility: hidden"
                onclick="hasClicked = true;">Submit Dial</button>
            <button onclick="clearlock();">Clear/Lock</button>
            <br> Current Dial Value: <font id="CurrentDial"></font>
            <br>
            <h1>LED Light Indicator</h1>
            Green = Sucess; Yellow = Preparing; Red = Clearing/Locking;
      </p>

      <p>
            Progress
            <br>
            <meter id="Gauge" value="0" min="0" max="3" style="width:25%;"></meter>
            <br>
            <font id="Status"></font>
       </p>

      <p id="Settings">
            Settings: Please note that values will be truncated into values between 0-255.
            <br>
            <table id="SettingsTable" align="center">
                  <col>
                        <th>Enter First Dial Value</th>
                        <td><input type="text" id="dialValue_1" value="0"></td>
                        <td><font id="dial_1"></font></td>
                  </col>
                  <col>
                        <th>Enter Second Dial Value</th>
                        <td><input type="text" id="dialValue_2" value="0"></td>
                        <td><font id="dial_2"></font></td>
                  </col>
                  <col>
                        <th>Enter Third Dial Value</th>
                        <td><input type="text" id="dialValue_3" value="0"></td>
                        <td><font id="dial_3"></font></td>
                  </col>
                  <col>
                        <th>Enter Tolerance Value</th>
                        <td><input type="text" id="tolerance"value="0"></td>
                        <td><font id="tolStored"></font></td>
                  </col>
                  <col>
                        <td rowspan=2><button id="Store" onclick="store()">Store</button></td>
                  </col>
            </table>
      </p>
</div>
```

JavaScript

```
var dial = new Array(0,0,0);      //Array to store dial values.
var tolerance = 0;                //To store the tolerance of dial values
var currDial;                     //To store current dial value.
var currDialFunc = setInterval(retrieveCurrDial, 100);   //Current dial retrieval.
var hasClicked = false; //To check whether submit button has been clicked.
var waiting;                      //To check if we are waiting for the submit click.

async function store(){

  // The loop is to store dial value and show the stored value.
  for(var i = 0; i < dial.length; i++){
    dial[i]=check(parseInt(document.getElementById("dialValue_" + (i + 1)).value));
    document.getElementById("dial_" + (i + 1)).innerHTML = dial[i];
  }

  // Stores tolerance and show the stored value and show that values are stored.
  tolerance = check(parseInt(document.getElementById("tolerance").value));
  document.getElementById("tolStored").innerHTML = tolerance;
  document.getElementById("Status").innerHTML = "Setting Stored.";
}

function check(value){
  if(value > 255){ return 255; }
  if(value < 0){ return 0; }
  return value;
}

async function start(){

  document.getElementById("Start").style.visibility = "hidden";

  document.getElementById("Settings").style.visibility = "hidden";

  document.getElementById("SettingsTable").style.display = "none";

  unlockProcess(0);

}

async function unlockProcess(index){

  // Flashes Yellow LED for 2 seconds.
  turnOn( Yellow );
  await mSec( 2000 );
  turnOff( Yellow );

  // Reveals submit button and changes status text.
  document.getElementById("Submit").style.visibility = "visible";
  document.getElementById("Status").innerHTML = "You may start dialing, click submit when
you're ready.";
  waiting = setInterval(function(){waitForButton(index);}, 100);
}
```

```
function unlockProcessContinue(index){
  document.getElementById("Submit").style.visibility = "hidden";
  currDial = analogIn( Dial );
  // Check if the current dial is within acceptable range.
  if( dial[index] - tolerance <= currDial && currDial <= dial[index] + tolerance){
    document.getElementById("Gauge").value = index+1;
    // If it is acceptable, then check if the accepted current stage is not the last stage of
    // the combination.
    if( index < dial.length - 1 ){
      // If the current stage is not the last stage, proceed to the next stage.
      document.getElementById("Status").innerHTML = "Onto the next Pin";
      unlockProcess(index+1);
    }else{
      // If the current stage is the last stage, then notify the user and reveal the settings panel
and
      // hide the submit button.
      document.getElementById("Status").innerHTML = "You have unlocked the lock!";
      document.getElementById("Submit").style.visibility = "hidden";
      document.getElementById("Settings").style.visibility = "visible";
      document.getElementById("SettingsTable").style.display = "initial";
      turnOn( Green );
    }
  }else{
    // If the current dial is not acceptable, signify user and clear the lock.
    document.getElementById("Status").innerHTML = "Failed to enter an acceptable Dial";
    clearlock();
  }
}

async function clearlock(){
  turnOn( Red );
  await mSec( 2000 );
  turnOff( Red );
  document.getElementById("Status").innerHTML = "Reverting to intial condition";
  document.getElementById("Start").style.visibility = "visible";
  document.getElementById("Submit").style.visibility = "hidden";
  document.getElementById("Settings").style.visibility = "hidden";
  document.getElementById("SettingsTable").style.display = "none";
  document.getElementById("Gauge").value = 0;
  turnOff( Green );
  document.getElementById("Status").innerHTML = "Lock is Cleared (Except Settings) and is
Locked.";
}

function waitForButton(index){
  if(hasClicked){
    clearInterval(waiting);
    hasClicked = false;
    unlockProcessContinue(index);
  }
}

function retrieveCurrDial(){
  currDial = analogIn( Dial );
  document.getElementById("CurrentDial").innerHTML = currDial;
}
```