**Using a Microprocessor in a Simple Electronic Device**

Two weeks ago, we introduced electronic design by using two push-button switches to control a two-color LED.

![Circuit Diagram](image)

This circuit provides red light when the left switch is closed and green light when the right switch is closed.

Last week, we put a microprocessor between the switches and LED.

![Circuit Diagram](image)

→ more complex and potentially more useful behaviors

For example, the lights could flash and/or the buttons could toggle.

**Simple Simon**

Start by considering a simpler game.

The game starts with a flash of red or green light (randomly chosen).

- If red is flashed, the player should press button 1.
- If green is flashed, the player should press button 2.

If the response is correct, the controller flashes green four times.

If the response is wrong, the controller flashes red four times.

Then the cycle repeats.
Simple Simon

This flowchart illustrates the controller view of the game.

Simple Simon

Each iteration of the game can be coded as one pass through loop.

Simple Simon

Consider the flash green four times box.

Simple Simon

Here is code for flash green four times.

Simple Simon

Use the random function to generate a random number.

Simple Simon

The function random(min,max) returns a random integer such that

\[ \text{min} \leq \text{random(min,max)} < \text{max} \]

Notice that min is a possible outcome but max is not.

The following code will set the variable c to 0 or 1 or 2 or 3.

... c = random(0,4)
...
**Simple Simon**

The *wait for button push* box is more complicated.

![Diagram of Simple Simon logic](image)

**Code for *wait for button push***.

```c
... int button;
...
while(1){
  if(analogRead(SENSOR1)<128){
    button = 1;
    break;
  }
  if(analogRead(SENSOR2)<128){
    button = 2;
    break;
  }
}
...```

**Part 1 of this Week’s Lab**

Part 1: *Write a program* to implement Simple Simon.

![Diagram of Simple Simon logic](image)

Make/upload a video of your working Simple Simon game.

**Part 2 of this Week’s Lab**

Part 1: Write a Program to Implement Simple Simon.
Part 2: *Rework hardware* for Simon game (4 switches + 4 LEDs)

![Image of hardware layout](image)
Rework Hardware for Simon Game

You can cut resistor and LED leads to fit in less space.

Leave at least 1/4" of bare lead to insert into the protoboard.

Rework Hardware for Simon Game

The photos below show a working implementation.

Check that your hardware works by writing a program to light each LED when the corresponding button is pressed.

Breakout Groups

We will divide up now to work in small groups to work on this week’s projects, which are described under the week 4 lab tab:

http://mit.edu/6.a01

Part 1: Write a Program to Implement Simple Simon.
   - Upload a video to demonstrate the Simple Simon game.

Part 2: Rework hardware for Simon game (4 switches + 4 LEDs)
   - Upload a picture of your upgraded hardware.

Next week: Write a program to implement the Simon game.