Simple Simon
Electronics inside an electric fan motor.

Electronics inside a Tesla Model 3 Motor.
Two weeks ago, we introduced electronic design by using two push-button switches to control a two-color LED.

This circuit provides red light when the left switch is closed and green light when the right switch is closed.
Using a Microprocessor in a Simple Electronic Device

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

→ more complex and potentially more useful behaviors

For example, the lights could flash and/or the buttons could toggle.
Today, we will use the same hardware/microprocessor configuration to implement a video game.
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.

```
start

let c = random(red, green)

flash color c

wait for button push

does button=c ?

flash red four times

flash green four times

```
Simple Simon

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

```
let c = random(red,green)
flash color c
wait for button push
does button=c?
flash red four times
flash green four times
yesno
loop
exit
```

Coding most of these boxes is straightforward.
Consider the **flash green four times** box.

```
loop

let c = random(red,green)
flash color c
wait for button push
does button=c ?
flash red four times
flash green four times
yesno
loop
exit
```
Here is code for **flash green four times**.

```c
... int i;
...
for(i=0; i<4; i++){
    digitalWrite(AIN1,LOW);
    digitalWrite(AIN2,HIGH);
    delay(100)
    digitalWrite(AIN1,LOW);
    digitalWrite(AIN2,LOW);
    delay(100)
}
...
Simple Simon

Use the `random` function to generate a random number.

```
loop

let c = random(red,green)
flash color c
wait for button push
does button=c ?
flash red four times
flash green four times
yesno
loop
exit
```
The function \( \text{random}(\text{min}, \text{max}) \) returns a random integer such that

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

Notice that \( \text{min} \) is a possible outcome but \( \text{max} \) is not.

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

\[
c = \text{random}(0,4)
\]
Simple Simon

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

```
loop

let c = random(red,green)
flash color c
wait for button push
does
button=c?
flash red
four times
flash green
four times
yesno
loop
exit
```
Simple Simon

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

1. **Wait for button push**
2. **Is button 1 pushed?**
   - Yes: **Set button = 2**
   - No: Go to next step
3. **Is button 2 pushed?**
   - Yes: **Set button = 1**
   - No: Exit
Simple Simon

Code for \textbf{wait for button push}.

...  
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.

```plaintext
let c = random(red,green)
flash color c
wait for button push
does button=c?
flash red four times
flash green four times
yesno
loop
exit

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)
Part 2 of this Week’s Lab

Tight fit? Here’s a possible layout for 1 switch + 1 LED.
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.