Electronics inside an electric fan motor.

Electronics inside a Tesla Model 3 Motor.
In Lab 2, we introduced electronic design by using two pushbutton 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.
In Lab 3, we put a microprocessor between switches and LED.

→ more complex and potentially more useful behaviors

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

Last time, we wrote a program to implement Simple Simon:

The game starts with a flash of red or green light (randomly chosen). If red is flashed, the player should press button 0. If green is flashed, the player should press button 1. 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.

Today we will write a program to turn four switches and four LEDs into an electronic game: Simon.
Reflection on Last Week

One way to code Simple Simon is to **enumerate** all possibilities.

Diagram:
- **Loop**
- **Randomly choose**
  - **Flash red**
    - **Wait for button**
      - **0**
        - **Flash green 4 times**
      - **1**
        - **Flash red 4 times**
  - **Red**
  - **Green**
    - **Flash green**
      - **Wait for button**
        - **0**
          - **Flash green 4 times**
        - **1**
          - **Flash red 4 times**
  - **Exit**
Enumeration is fine for Simple Simon, since there are just two possible outcomes: i.e., there is either one red flash or one green flash.

However, there are many possible sequences in each iteration of Simon. Here is the algorithm:

- **Choose** a challenge sequence $c$ of four random integers $\in [0,3]$.
- **Play** the challenge sequence $c$:
- **Wait** for user to enter a response sequence $r$.
- If $c == r$, flash green light from all LEDs four times.
- If $c != r$, flash red light from all LEDs four times.
- Repeat this cycle.
Enumeration

Enumeration is fine for Simple Simon, since there are just two possible outcomes: i.e., there is either one red flash or one green flash.

However, there are many possible sequences in each iteration of Simon.

Q: How many possible sequences are there?
Modular Design

This **flowchart** illustrates a high-level view of the Simon algorithm.

Our goal is to structure the code with corresponding **modularity**.
The **loop** function invokes high-level abstractions.

```c
void loop(){
    int c[4],r[4]; // save challenge and response in arrays
    issue_challenge(c); // create and display a challenge
    get_response(r); // get the player’s response
    give_feedback(c,r); // flash LEDs to indicate success/failure
}
```
The first abstraction is to create a challenge `c` and issue it.

```c
void issue_challenge(int c[4]){
    int i;
    for(i=0; i<4; i++) { // there are four trials in each challenge
        c[i] = random(0,4); // each trial = randomly selected position c[i]
        led_green(c[i]);   // turn the LED at position c[i] green
        delay(200);
        led_off(c[i]);     // turn the LED at position c[i] off
        delay(200);
    }
} // on return, c[] will contain the challenge
```
Get the player’s response \( r \).

```c
int get_one_button(){
    int i;
    while(1){
        for(i=0; i<4; i++) { // loop over buttons
            if(analogRead(button[i])<128){ // test if button \( i \) was pushed
                led_green(i); // light LED to let player know
                delay(100); // wait for 0.1 seconds
                // REPLACE THIS LINE by code to wait for button[i] to be released
                led_off(i); // turn off LED
                delay(100);
                return i; // return the position of the pressed button
            }
        }
    }
}

void get_response(int r[4]){
    int i;
    for(i=0; i<4; i++){
        r[i] = get_one_button();
    }
} // on return, \( r[] \) will contain the response
```
Determine feedback $f$ and issue it.

```c
void give_feedback(int c[4], int r[4]){
    int i, j, f;
    // REPLACE THIS LINE by code to set $f = 1$ if response $r$ is correct else 0
    delay(400);
    for(j=0; j<4; j++) { // j = flash number
        for(i=0; i<4; i++) { // i = led position
            if(f){
                led_green(i);
            } else {
                led_red(i);
            }
        }
        delay(100);
        for(i=0; i<4; i++) led_off(i);
        delay(100);
    }
    delay(400);
}
```
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 5 lab tab:

http://mit.edu/6.a01

Write a Program to Implement Simon.
– **Upload a video** to demonstrate the Simon game.