Lab – Wireless Power Charging Security

**Objective**

This lab is designed to offer an introduction into the concept of Cyber-Physical System Security. The purpose of the experiment is for students to understand how randomness is utilized to achieve wireless power charge security in a CPS system.

**Introduction and Background**

Cyber-physical systems (CPS) are systems that contain computational elements and physical components; both the “cyber” and the “physical” parts of the system interact with each other. One, specific example of CPS’s are implanted medical devices (IMD). A pacemaker is an IMD that has many benefits; however, an issue of security arises.1 General protection schemes cannot be applied to all CPS’s; the security measures taken need to be specifically integrated for that particular the system. For instance, during the wireless power charging of a pacemaker, it is vulnerable to potential cyber attacks. A general protection scheme like cryptography would only be effective in the transfer of data; the wireless charging process is a transfer of electromagnetic energy.2 A solution for security in this case could be achieved through secret resonance frequency tuning.3

Wireless power charging depends on a sender and receiver matching resonant frequency. Hackers could attack a wireless charge if resonant frequency is static. If the frequency was only known to the sender and receiver, and it was unknown to an attacker then security would be achieved. The experiment in this lab will focus on generating a random, resonant frequency using logistic maps.

**Equipment**

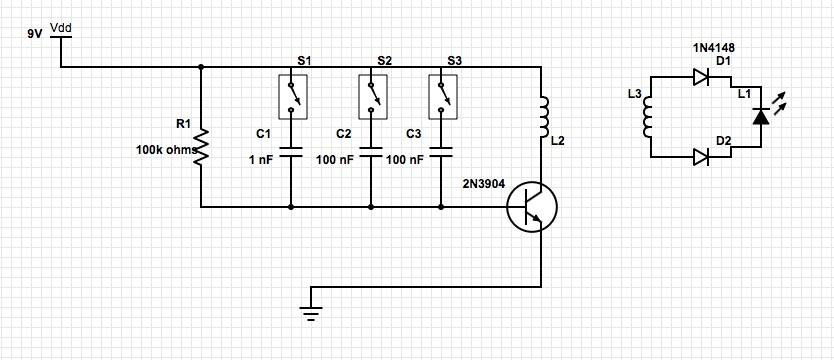
Microcontroller (Arduino), Breadboard, LED’s, Resistors, 5 cm Coils, Capacitors, 2N3904 Transistor, Mechanical Switches, Diodes

**Pre-Lab Assignment**

1. Prepare programming code for the microcontroller that implements a chaotic output from using the logistic map equation:

Xn+1 = A Xn(1 − Xn) ≡ fA(X) (1)5

1. Research typical voltage and current values for the LEDs used in this experiment to find appropriate resistors. Currents can be too high and damage LEDs so it is important to have correct resistance values for the circuit.4
2. Have the wireless power transfer circuit built before coming to the started lab time. The schematic below is reference:



Where L2 and L3 are the 5 cm coils that you made.

**Procedure**

1. First, build an array of LEDs and resistors on the breadboard. The resistors should be in parallel with the LEDs and every branch should connect back to ground. Be sure that you have the proper resistance values that correspond with the LEDs in use. Your circuit should resemble the one displayed in Figure-1 below.

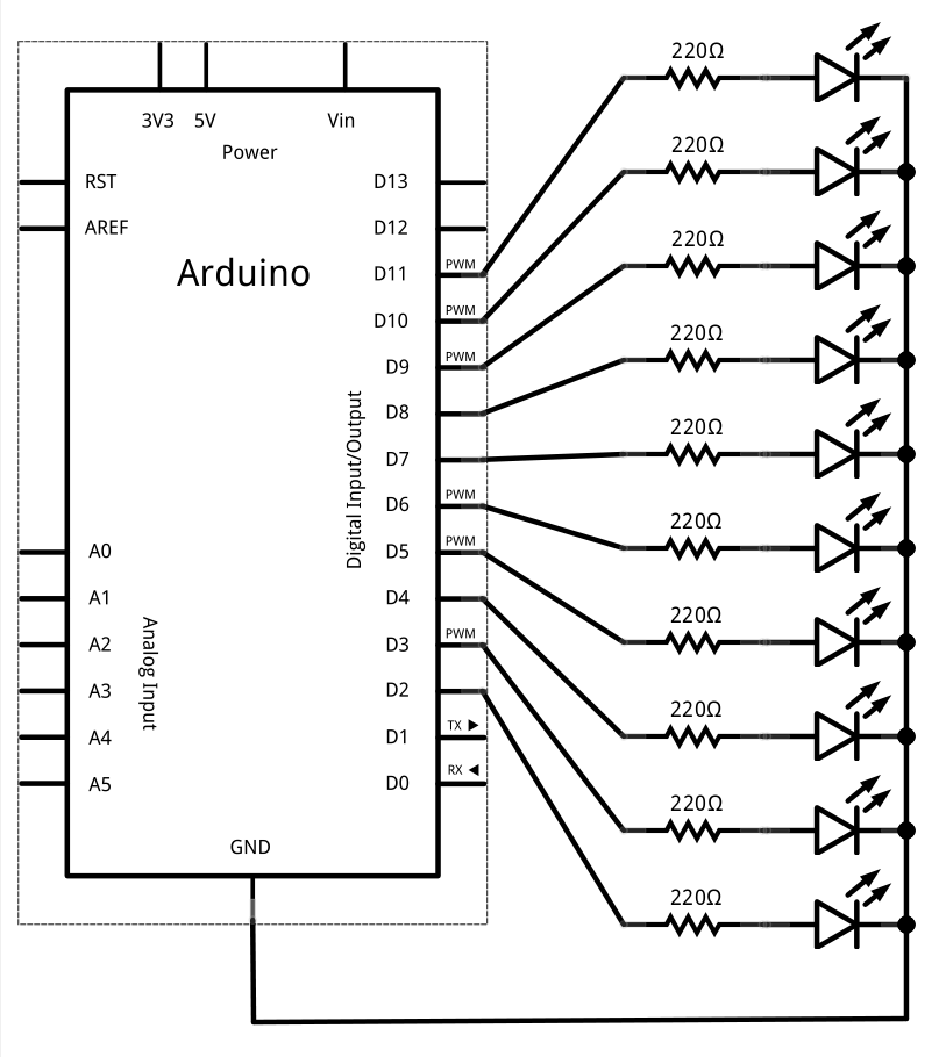


Figure-1 – Circuit Schematic5

This schematic shows 10 LEDs and resistors in parallel; you only need three for this lab project.

1. Connect the microcontroller to a computer with the Arduino environment set up the code you prepared in the pre-lab. Your code should implement the logistic map equation resulting in a random, chaotic output.5 There is an example code provide later in this lab assignment for reference.
2. Run your program on the microcontroller and observe the LEDs in the circuit. Offer analysis of what you observe and characteristics of the logistic map function. Change the const A to different values between 3.5 - 4.0. Write down your observations. Set it to under 3.5 and draw conclusion as to the importance of const A.
3. Set up the wireless power transfer circuit and give the 9V power to the circuit. Experiment with switching different capacitors on and off. What do you observe?

1. Turn all switches on and vary the distance of the two coils. What do you observe?

**References**

1. *Cyber-Physical System Security*. PowerPoint presentation. Department of Electrical Engineering, University of Alabama, Tuscaloosa, AL. 17 Feb. 2014. 26 Apr. 2014
2. Hu, Fei. *Introduction to CPS Security*. PowerPoint presentation. Department of Electrical Engineering, University of Alabama, Tuscaloosa, AL. 26 Apr. 2014
3. Hu, Fei. *Secure Wireless Powering of Implanted Devices via Secret Resonance Frequency Tuning*. Department of Electrical Engineering, University of Alabama 26 Apr. 2014.
4. Kung, Patrick. *Principles of Operational Amplifiers and Application to Control a Light Source*. N.p., n.d. Web. 26 Apr. 2014
5. Serna, Juan D., and Amitabh Joshi. *Visualizing the Logistic Map with a Microcontroller*. N.p., 25 Dec. 2011. Web. 26 Apr. 2014.

Example Microcontroller Code

/ Blinking Logistic Map

// choose the pin for each LED

const int NbrLEDs = 3;

const int LEDpin[] = {4,5,6};

const int wait = 250; // wait for 500 milliseconds

// Logistic Map Parameters

const double A = 4.0; // Logistic map constant

double X0 = 0.1; // Initial position (0 < X0 < 0.8)

double X = X0; // Use X0 as your first calculated point

// setup() initializes the LED pins

void setup() {

for (int i = 0; i < NbrLEDs; i++) {

pinMode(LEDpin[i], OUTPUT);

}

}

// loop() iterates the Logistic Map and turn on/off LEDs

void loop()

{

if (X < 0.1)

{ digitalWrite(LEDpin[0],LOW);

digitalWrite(LEDpin[1],LOW);

digitalWrite(LEDpin[2],LOW);

delay(wait);}

else if ((X >= 0.1) && (X < 0.2))

blinkLED (LEDpin[2]);

else if ((X >= 0.2) && (X < 0.3))

blinkLED (LEDpin[1]);

else if ((X >= 0.3) && (X < 0.4)) //Truth Table Case 3

{digitalWrite(LEDpin[1],HIGH);

digitalWrite(LEDpin[2],HIGH);

delay(wait);

digitalWrite(LEDpin[1],LOW);

digitalWrite(LEDpin[2],LOW);}

else if ((X >= 0.4) && (X < 0.5)) //Truth Table Case 4

blinkLED (LEDpin[0]);

else if ((X >= 0.5) && (X < 0.6)) //Truth Table Case 5

{digitalWrite(LEDpin[0],HIGH);

digitalWrite(LEDpin[2],HIGH);

delay(wait);

digitalWrite(LEDpin[0],LOW);

digitalWrite(LEDpin[2],LOW); }

else if ((X >= 0.6) && (X < 0.7)) //Truth Table Case 6

{digitalWrite(LEDpin[0],HIGH);

digitalWrite(LEDpin[1],HIGH);

delay(wait);

digitalWrite(LEDpin[0],LOW);

digitalWrite(LEDpin[1],LOW); }

else //Truth Table Case 7

{digitalWrite(LEDpin[0],HIGH);

digitalWrite(LEDpin[1],HIGH);

digitalWrite(LEDpin[2],HIGH);

delay(wait);

digitalWrite(LEDpin[0],LOW);

digitalWrite(LEDpin[1],LOW);

digitalWrite(LEDpin[2],LOW);}

// iterates the Logistic Map function

X0 = X;

X = A \* X0 \* (1.0 - X0);

}

// blinkLED function

// turn on/off LEDs

void blinkLED (const int pin) {

digitalWrite(pin, HIGH); // turn LED on

delay(wait); // wait 500 milliseconds

digitalWrite(pin, LOW); // turn LED off

}