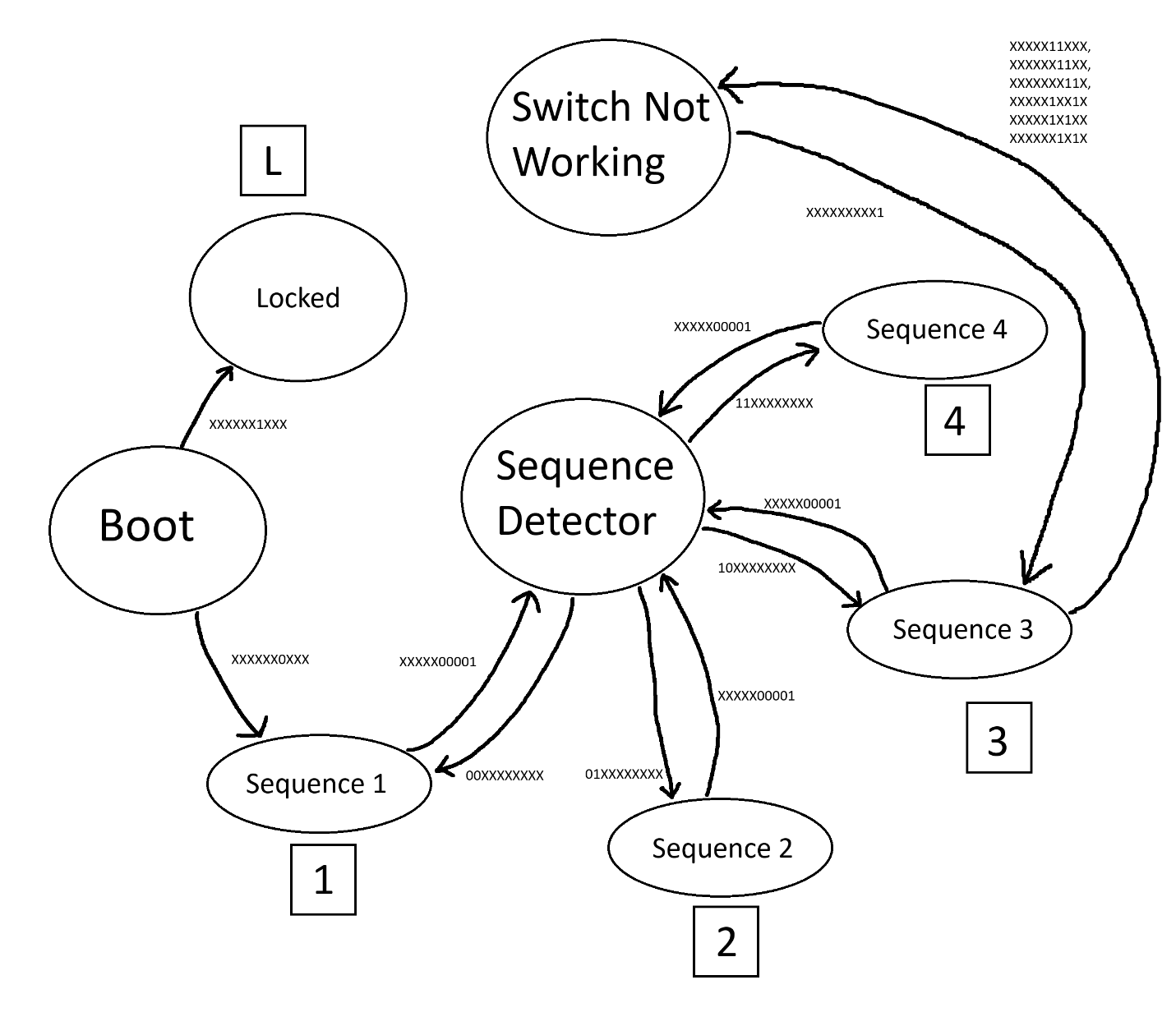
**Magic Box Guidelines:**

You might want to follow the same state diagram:  


For simplicity, we omitted the output of the following states: Locked, Sequence 1, Sequence 2, Sequence 3 and Sequence 4; and we decided to list them below:

**Locked (L):**

* Switch 1 🡪 LED 1
* Switch 2 🡪 LED 2
* Switch 3 🡪 LED 3
* Switch 4 🡪 LED 4

**Sequence 1:**

* Switch 1 🡪 LED 1
* Switch 3 🡪 LED 2
* Switch 2 🡪 LED 3
* Switch 4 🡪 LED 4

**Sequence 2:**

* Switch 2 🡪 LED 2
* Switch 3 🡪 LED 3
* Switch 1 🡪 LED 4
* Switch 4 🡪 LED 1

**Sequence 3:**

* Switch 4 🡪 LED 3
* Switch 3 🡪 LED 4
* Switch 2 🡪 LED 1
* Switch 1 🡪 LED 2

**Sequence 4:**

* Switch 1 🡪 LED 4
* Switch 2 🡪 LED 3
* Switch 3 🡪 LED 2
* Switch 4 🡪 LED 1

To be able to derive our equations, we filled up a huge truth table on excel. (The excel file is available on GitHub too). The input variables were listed as follows:



**Memory:**

M1, M0: These two bits represent the last switch that was turned off.

00 means the 1st, 01 means the 2nd, 10 means the 3rd, and 11 means the 4th.

**Current State:**

A(t), B(t), C(t): As the name suggests, these three bits represent the current state of the state machine. As represented by the state diagram, we have 8 states, which forced us to use at least 3 bits.

**Inputs:**

S1, S2, S3, S4: These inputs represent our switches.

**Timer:**

T: This is a timer that gives a high signal (1) 4 seconds after it is triggered.

Based on our design, and shown diagram, the circuit should enter locked mode if it is booted with switch 2 being high (5 V). Once locked, the machine stays locked, until reset by removing the power supply and rebooting the circuit.

If the machine did not enter the locked state, and went into the sequence detector state, it will start at Sequence 1 by default, then the FSM will remember the last switch that got turned off once all the switches are off and will change states accordingly.

Using a python code to process our variables (also available in the GitHub repository!) we obtained the following equations:

A(t+1) = (A ∧ ¬C) ∨ (A ∧ B ∧ ¬T) ∨ (A ∧ M1 ∧ ¬B) ∨ (S2 ∧ ¬B ∧ ¬C) ∨

(B ∧ C ∧ S3 ∧ ¬A ∧ ¬S1 ∧ ¬S2 ∧ ¬S4) ∨ (B ∧ T ∧ ¬A ∧ ¬S1 ∧ ¬S2 ∧ ¬S3 ∧ ¬S4) ∨

(C ∧ T ∧ ¬A ∧ ¬S1 ∧ ¬S2 ∧ ¬S3 ∧ ¬S4)

B(t+1) = (A ∧ B) ∨ (B ∧ S1) ∨ (B ∧ S2) ∨ (B ∧ S3) ∨ (B ∧ S4) ∨ (B ∧ ¬T) ∨

(S2 ∧ ¬A ∧ ¬C) ∨ (A ∧ C ∧ M0 ∧ ¬M1)

C(t+1) = (B ∧ C) ∨ (C ∧ ¬A) ∨ (C ∧ ¬M0) ∨ (¬A ∧ ¬B ∧ ¬S2) ∨

(T ∧ ¬A ∧ ¬S1 ∧ ¬S2 ∧ ¬S3 ∧ ¬S4) ∨ (T ∧ ¬B ∧ ¬C ∧ ¬S1 ∧ ¬S2 ∧ ¬S3 ∧ ¬S4)

L1 = (B ∧ C ∧ S2) ∨ (A ∧ B ∧ S1 ∧ ¬C) ∨ (A ∧ S4 ∧ ¬B ∧ ¬C) ∨

(B ∧ S4 ∧ ¬A ∧ ¬C) ∨ (C ∧ S1 ∧ ¬A ∧ ¬B)

L2 = (B ∧ C ∧ S1) ∨ (B ∧ S2 ∧ ¬C) ∨ (A ∧ S3 ∧ ¬B ∧ ¬C) ∨

(C ∧ S3 ∧ ¬A ∧ ¬B)

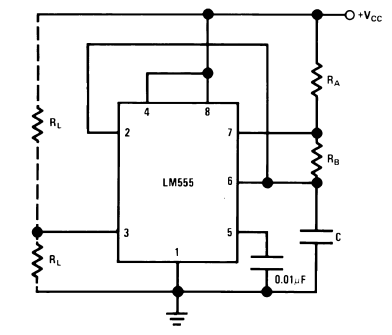
L3 = (B ∧ S3 ∧ ¬C) ∨ (B ∧ C ∧ S4 ∧ T) ∨ (B ∧ C ∧ S4 ∧ ¬A) ∨

(A ∧ S2 ∧ ¬B ∧ ¬C) ∨ (C ∧ S2 ∧ ¬A ∧ ¬B)

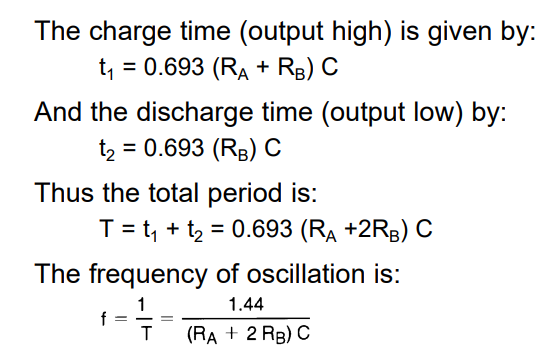
L4 = (B ∧ C ∧ S3) ∨ (A ∧ B ∧ S4 ∧ ¬C) ∨ (A ∧ S1 ∧ ¬B ∧ ¬C) ∨

(B ∧ S1 ∧ ¬A ∧ ¬C) ∨ (C ∧ S4 ∧ ¬A ∧ ¬B)

Astable was used to generate a clock signal using the following configuration:

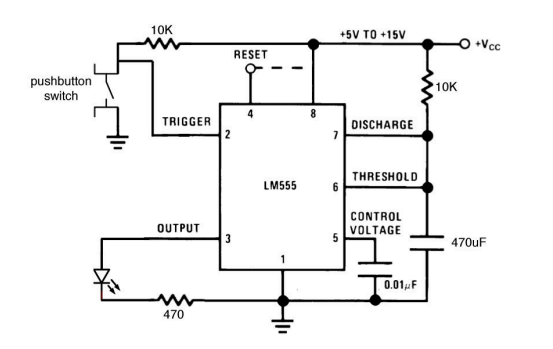


This functionality was achieved using the following equations:



By fixing (RA + RB) C and RB C to be 0.3607 s, we can generate a clock signal of 0.5 seconds.

The second configuration is the monostable configuration that we used to create a timer:



This functionality was achieved by using the following equations:



By choosing RC to be 3.63, we can get a timer of 4 seconds.