LAB REPORT-5

Name: Snigdha stp

Roll number: 2023102036

Group number: G6

Experiment-1

Objective: Preparing SR Latch using NOR Gates

Electronic components:

1) One NOR Gate (IC-7402)

2) Two LEDs

3) Two Resistors

4) One DIP Switch SPSTx4

Procedure:

1) Take Inputs 1A and 2A of the DIP Switch as R and S respectively.

2) Connect R and S to inputs 1A and 2A of the NOR Gate.

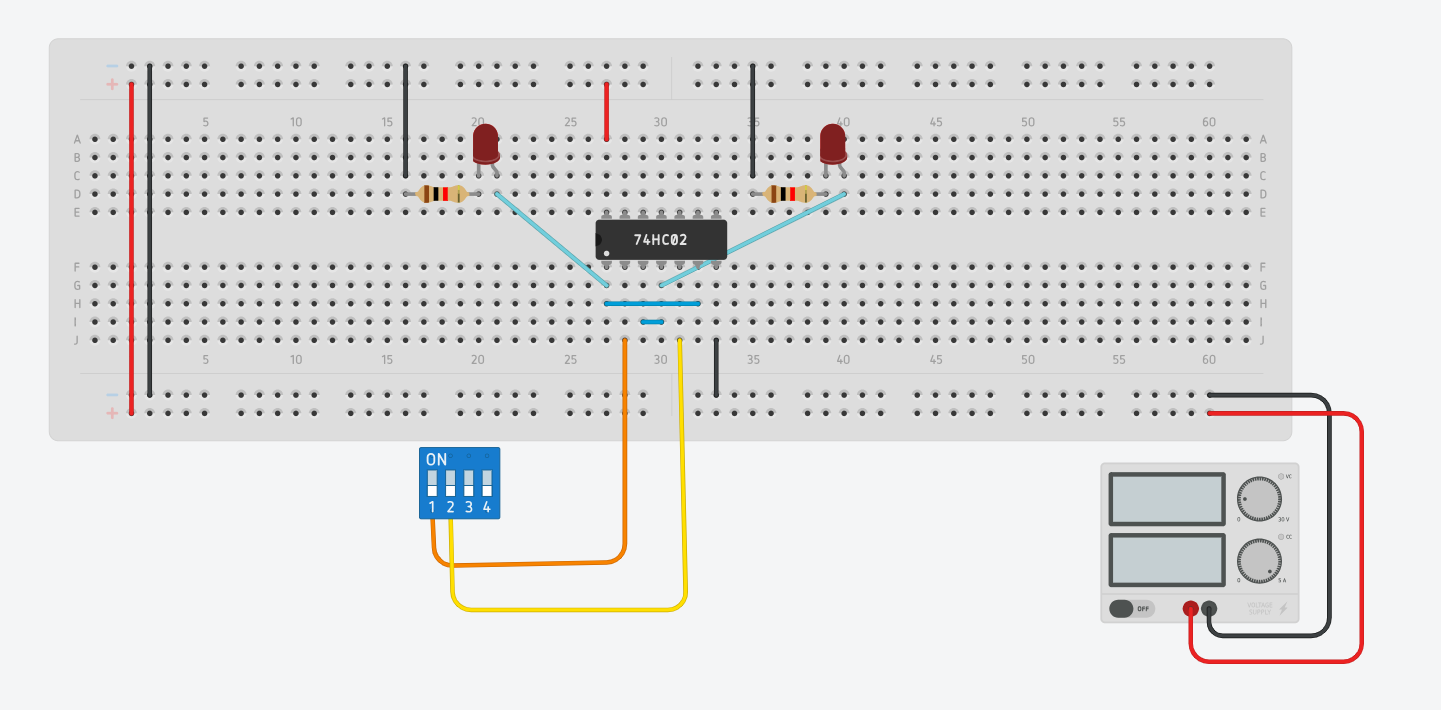
3) Connect outputs 1 and 2 of the gate to Inputs 2B and 1B of the same gate.

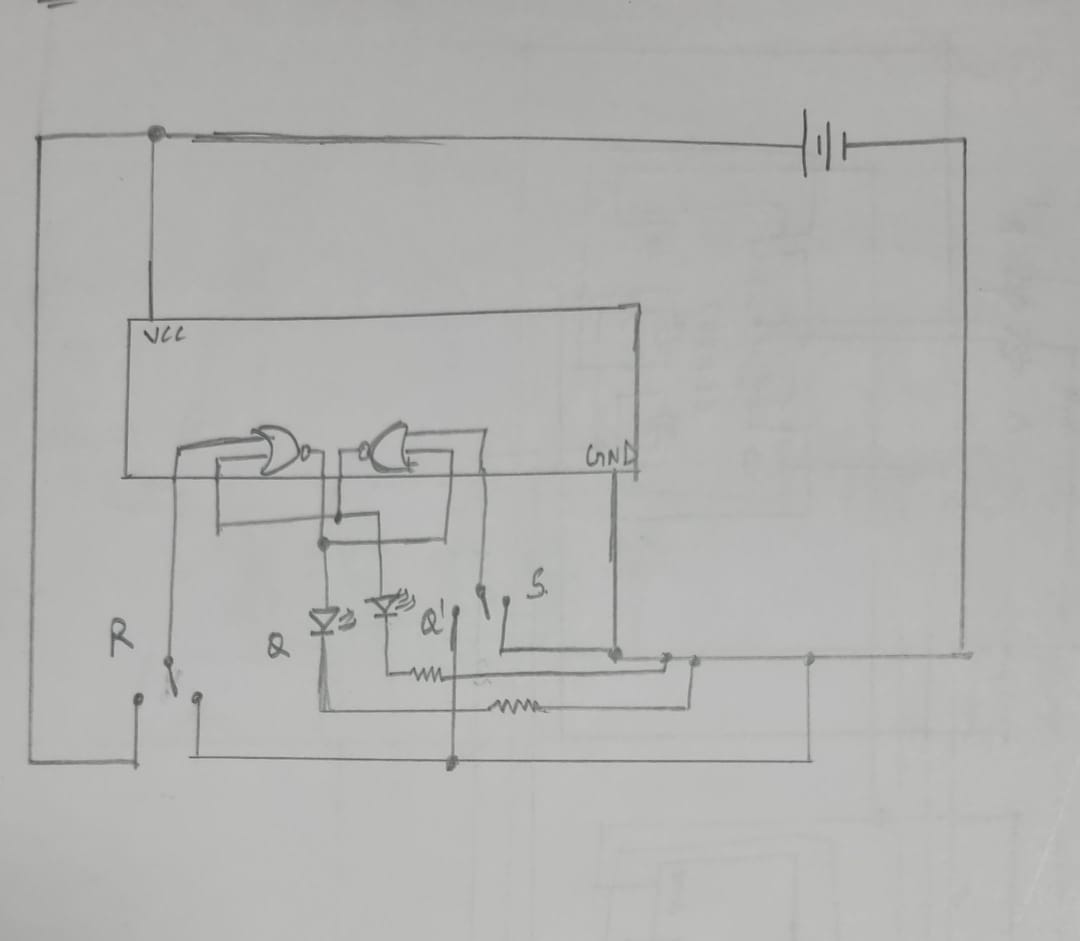
4) Connect Outputs 1 and 2 of the NOR gate to the Anode of the LEDs and the Cathode of the LEDs to terminal 2 of the resistor such that terminal 1 of those resistors is grounded.

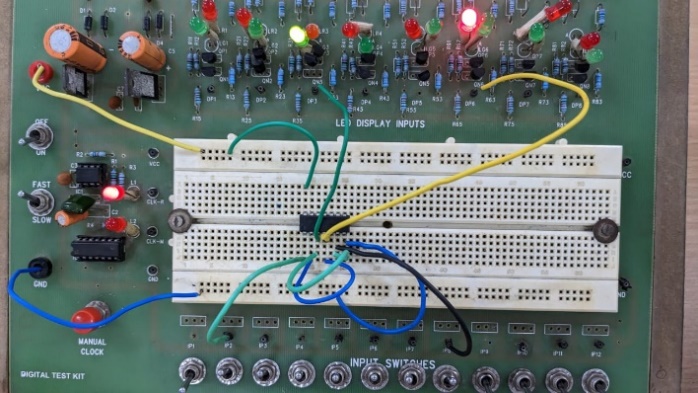
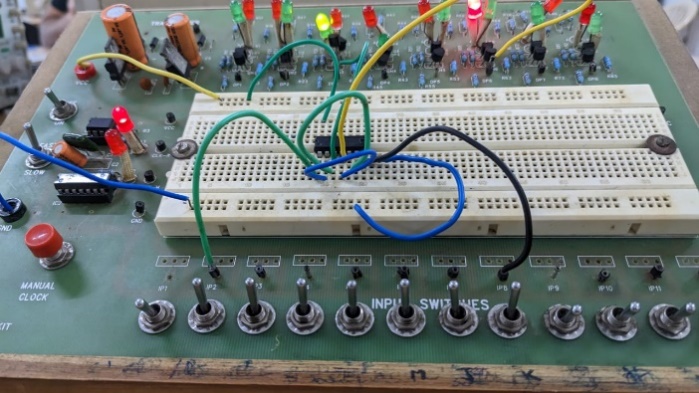
Observation/Conclusion:

|  |  |  |  |
| --- | --- | --- | --- |
| S | R | Q | Q’ |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
|  |  |  |  |

Reference Circuits:







Tinkercad link:

<https://www.tinkercad.com/things/atRLFmCCa8M-swanky-amur/editel?sharecode=OL1DiWJgSFc83RwfWMCv-NUHauciLkmhzuLtw4unYLA>

Note: We can’t change (11 to 00) or (10 to 01) immediately in SR Latch i.e., we can’t change two inputs at a time

Experiment-2

Objective: Producing JK Master-Slave Flip-Flop using normal gates.

Electronic components:

1) Two 4-Input NAND Gates

3) Two 2-Input NOR gate

4) Two LEDs

5) Two Resistors

6) One DIP Switch SPSTx4

Procedure:

1)Take 1A,2A, and 3A of the DIP Switch as Clock, J, and K respectively.

2)Take the left LED as a result of Q and the right LED as a result of Q’ respectively.

3)Connect the unused inputs of the 4-input NAND gates to the HIGH (VCC) level.

4)Build the Flip-flop as shown in the below reference circuit.

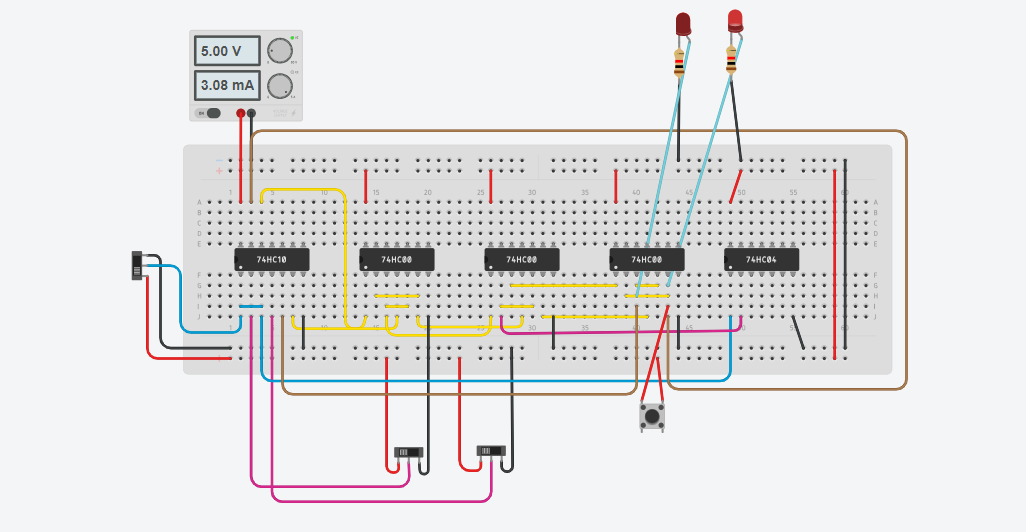
Observation/Conclusion:

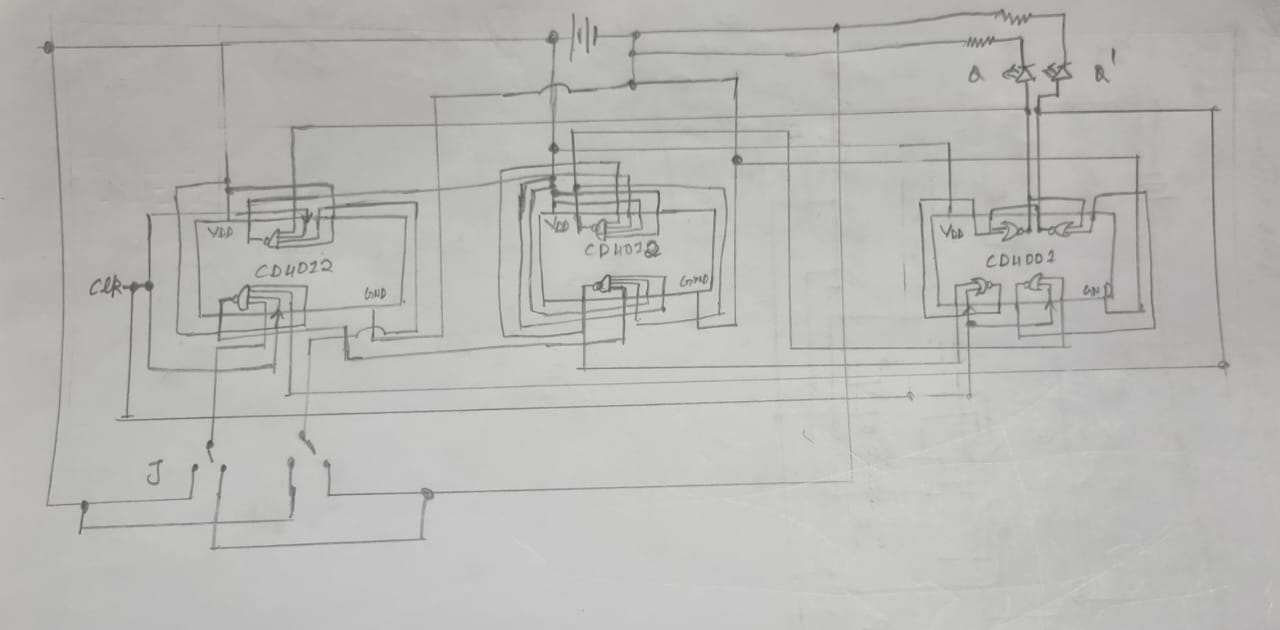


Equations:

1. RM = K • Q • CK
2. SM = J • Q’ • CK
3. SS = QM • CK’
4. RS = QM’• CK’

Reference Circuits:



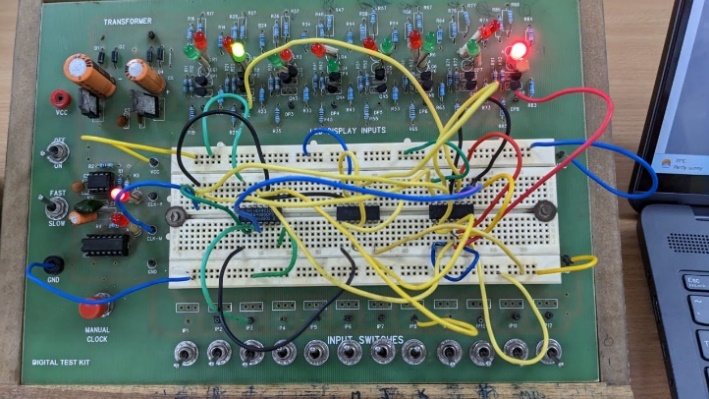






A circuit board with wires

Description automatically generated with low confidence



Tinkercad link:

<https://www.tinkercad.com/things/iD1ttlFQuHC-magnificent-borwo/editel?sharecode=FdUUx7nsdxhN8oSoRBGiUsEZrMtDRMr6Ls-sIUVd6l8>

Note: While stimulating the circuit the website could crash. Please wait for some time without moving the mouse.

EXPERIMENT-3

Objective: Preparing a 4-bit Up-Down Counter using Arduino

Electronic components:

1) One Arduino Uno R3

2) Four LEDs

3) Four Resistors

Procedure:

1) Connect GND and 5V of Arduino to the breadboard as shown in the diagram.

2) Connect Pins 2,3,4 and 5 on the Digital of the Arduino to the Anode of the LEDs and the Cathode of the LEDs to terminal 2 of the resistors such that terminal 1 of those resistors is grounded.

3) Execute the following code,

#ifndef Event\_h

#define Event\_h

#include <inttypes.h>

#define EVENT\_NONE 0

#define EVENT\_EVERY 1

#define EVENT\_OSCILLATE 2

class Event

{

public:

Event(void);

void update(void);

void update(unsigned long now);

int8\_t eventType;

unsigned long period;int repeatCount;

uint8\_t pin;

uint8\_t pinState;

void (\*callback)(void);

unsigned long lastEventTime;

int count;

};

#endif

#ifndef Timer\_h

#define Timer\_h

#include <inttypes.h>

#define MAX\_NUMBER\_OF\_EVENTS (10)

#define TIMER\_NOT\_AN\_EVENT (-2)

#define NO\_TIMER\_AVAILABLE (-1)

class Timer

{

public:

Timer(void);

int8\_t every(unsigned long period, void (\*callback)(void));

int8\_t every(unsigned long period, void (\*callback)(void), int repeatCount);

int8\_t after(unsigned long duration, void (\*callback)(void));

int8\_t oscillate(uint8\_t pin, unsigned long period, uint8\_t startingValue);

int8\_t oscillate(uint8\_t pin, unsigned long period, uint8\_t startingValue, int repeatCount);

/\* This method will generate a pulse of !startingValue, occuring period after the

\* call of this method and lasting for period. The Pin will be left in !startingValue.

\*/

int8\_t pulse(uint8\_t pin, unsigned long period, uint8\_t startingValue);

/\*\*

\* This method will generate a pulse of pulseValue, starting immediately and of

\* length period. The pin will be left in the !pulseValue state

\*/

int8\_t pulseImmediate(uint8\_t pin, unsigned long period, uint8\_t pulseValue);

void stop(int8\_t id);

void update(void);

void update(unsigned long now);

protected:

Event \_events[MAX\_NUMBER\_OF\_EVENTS];

int8\_t findFreeEventIndex(void);

};

#endif

//// YOUR CODE STARTS HERE

Timer t;

int count = 0;

int state = 0;

// "every" X milliseconds

void stopAllTimers() {

t.stop(3);

t.stop(2);

t.stop(5);

t.stop(4);

t.stop(1);state = state == 0 ? 1 : 0;

t.oscillate(2, 500, state);

t.oscillate(3, 1000, state);

t.oscillate(4, 2000, state);

t.oscillate(5, 4600, state);

}

void setup() {

Serial.begin(9600);

pinMode(2, OUTPUT);

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

pinMode(5, OUTPUT);

t.every(8000, stopAllTimers);

t.oscillate(2, 500, state);

t.oscillate(3, 1000, state);

t.oscillate(4, 2000, state);

t.oscillate(5, 4600, state);

}

/\* // starts from high, oscillates on the pin every 500ms

int eventId = t.oscillate(pin, 500, HIGH);

if (eventId < 0) {

Serial.println("Could not initialize timer");

}

} \*/

// 1 unit of your timer = 500ms in real time

void loop() {

t.update();

}

//// YOUR CODE ENDS HERE// For Arduino 1.0 and earlier

// #if defined(ARDUINO) && ARDUINO >= 100

// #include "Arduino.h"

// #else

// #include "WProgram.h"

// #endif

Event::Event(void)

{

eventType = EVENT\_NONE;

}

void Event::update(void)

{

unsigned long now = millis();

update(now);

}

void Event::update(unsigned long now)

{

if (now - lastEventTime >= period)

{

switch (eventType)

{

case EVENT\_EVERY:

(\*callback)();

break;

case EVENT\_OSCILLATE:

pinState = ! pinState;

digitalWrite(pin, pinState);

break;

}

lastEventTime = now;count++;

}

if (repeatCount > -1 && count >= repeatCount)

{

eventType = EVENT\_NONE;

}

}

Timer::Timer(void)

{

}

int8\_t Timer::every(unsigned long period, void (\*callback)(), int repeatCount)

{

int8\_t i = findFreeEventIndex();

if (i == -1) return -1;

\_events[i].eventType = EVENT\_EVERY;

\_events[i].period = period;

\_events[i].repeatCount = repeatCount;

\_events[i].callback = callback;

\_events[i].lastEventTime = millis();

\_events[i].count = 0;

return i;

}

int8\_t Timer::every(unsigned long period, void (\*callback)())

{

return every(period, callback, -1); // - means forever

}

int8\_t Timer::after(unsigned long period, void (\*callback)())

{return every(period, callback, 1);

}

int8\_t Timer::oscillate(uint8\_t pin, unsigned long period, uint8\_t startingValue, int repeatCount)

{

int8\_t i = findFreeEventIndex();

if (i == NO\_TIMER\_AVAILABLE) return NO\_TIMER\_AVAILABLE;

\_events[i].eventType = EVENT\_OSCILLATE;

\_events[i].pin = pin;

\_events[i].period = period;

\_events[i].pinState = startingValue;

digitalWrite(pin, startingValue);

\_events[i].repeatCount = repeatCount \* 2; // full cycles not transitions

\_events[i].lastEventTime = millis();

\_events[i].count = 0;

return i;

}

int8\_t Timer::oscillate(uint8\_t pin, unsigned long period, uint8\_t startingValue)

{

return oscillate(pin, period, startingValue, -1); // forever

}

/\*\*

\* This method will generate a pulse of !startingValue, occuring period after the

\* call of this method and lasting for period. The Pin will be left in !startingValue.

\*/

int8\_t Timer::pulse(uint8\_t pin, unsigned long period, uint8\_t startingValue)

{

return oscillate(pin, period, startingValue, 1); // once

}

/\* This method will generate a pulse of startingValue, starting immediately and of

\* length period. The pin will be left in the !startingValue state

\*/

int8\_t Timer::pulseImmediate(uint8\_t pin, unsigned long period, uint8\_t pulseValue)

{

int8\_t id(oscillate(pin, period, pulseValue, 1));

// now fix the repeat count

if (id >= 0 && id < MAX\_NUMBER\_OF\_EVENTS) {

\_events[id].repeatCount = 1;

}

return id;

}

void Timer::stop(int8\_t id)

{

if (id >= 0 && id < MAX\_NUMBER\_OF\_EVENTS) {

\_events[id].eventType = EVENT\_NONE;

}

}

void Timer::update(void)

{

unsigned long now = millis();

update(now);

}

void Timer::update(unsigned long now)

{

for (int8\_t i = 0; i < MAX\_NUMBER\_OF\_EVENTS; i++)

{

if (\_events[i].eventType != EVENT\_NONE)

{

\_events[i].update(now);}

}

}

int8\_t Timer::findFreeEventIndex(void)

{

for (int8\_t i = 0; i < MAX\_NUMBER\_OF\_EVENTS; i++)

{

if (\_events[i].eventType == EVENT\_NONE)

{

return i;

}

}

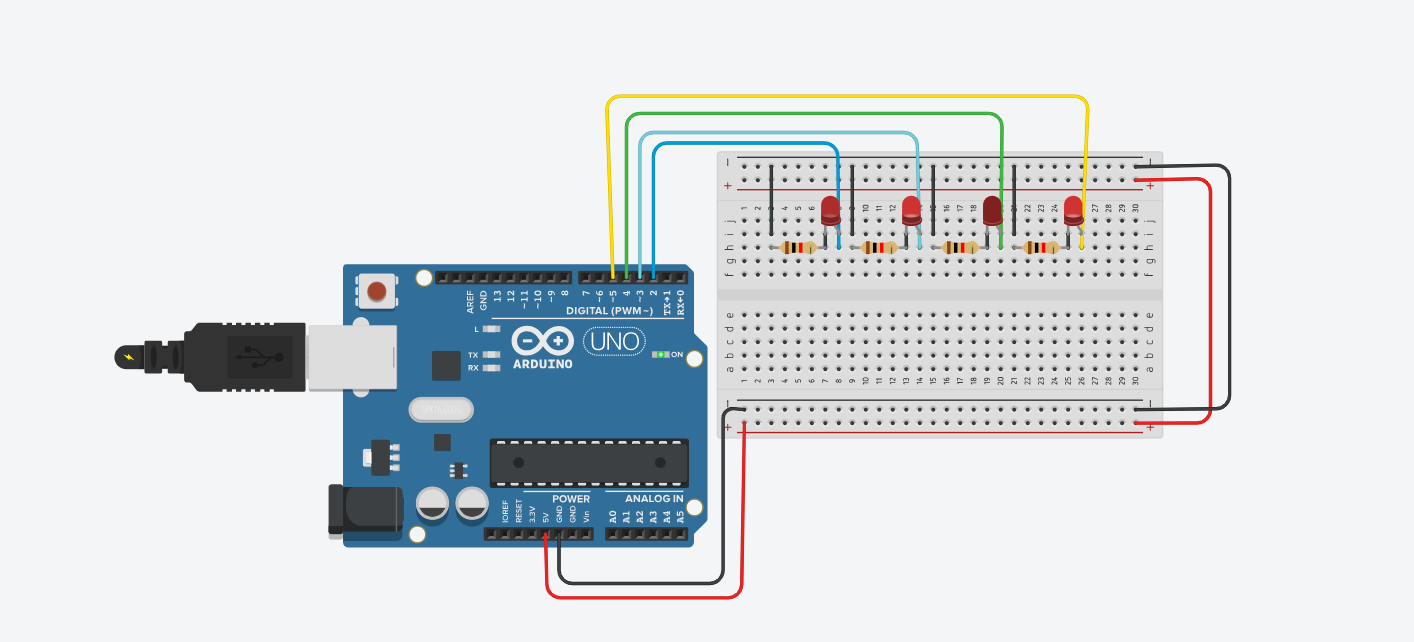
return NO\_TIMER\_AVAILABLE;

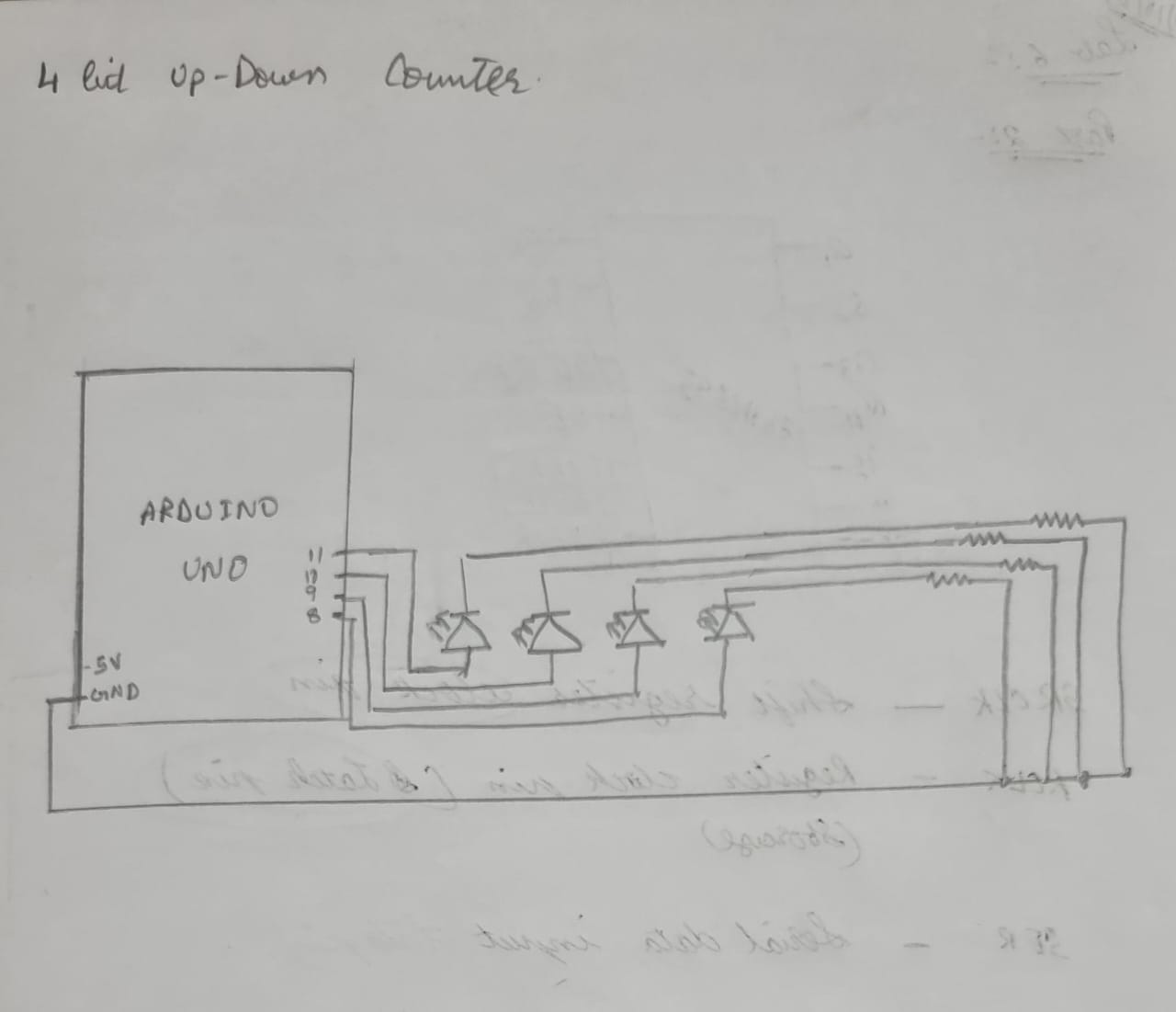
}

Observation: First, the bulbs will glow in such a way that it represents the numbers 0 to 15 in the binary form (which shows that it is acting as a 4-bit up counter), then the bulbs will glow in such a way that it represents the numbers 15 to 0 in the binary form (which shows that it is acting as a 4-bit down counter).

Note: If a bulb glows, it represents the number 1; if it doesn’t glow, assume that it represents the number 0.

Reference Circuit:







Tinkercad link:

<https://www.tinkercad.com/things/9EqfbUqJvin-ingenious-esboo-lappi/editel?sharecode=GmV14qCj2R9nYJRacmAPqQ2qVoDF0Zg1g2rSzhBn290>