**LAB REPORT 5**

**Name:- Ayush Kumar Gupta**

**ROLL NO:- 2023114001**

**1.OBJECTIVE –** To Form RS OR Latch and check input results.

**ELCTRONIC COMPONENTS REQUIRED -**

1. Digital test kit.
2. NOR Gate.

**PROCEDURE:**

1. Test the ICs, LED Lights and Switches.
2. Connect ICs with GND, Power.
3. ­­­­­­­Connect the circuit as given in reference circuit. A diagram of a block diagram

   Description automatically generated
4. Check the output with following order:

S R = 01, 00, 10, 00, 01, 10, 01, 00, 11, 00, 10, 11, 00, 01, 11, 00.

Observe the output and make the Table.

|  |  |  |  |
| --- | --- | --- | --- |
| S | R | Output | Output’ |
| 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 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |

Conclusion:

On SR value of 01, Latch gets reset to 01.

On SR value of 00, Latch retains the previous value.

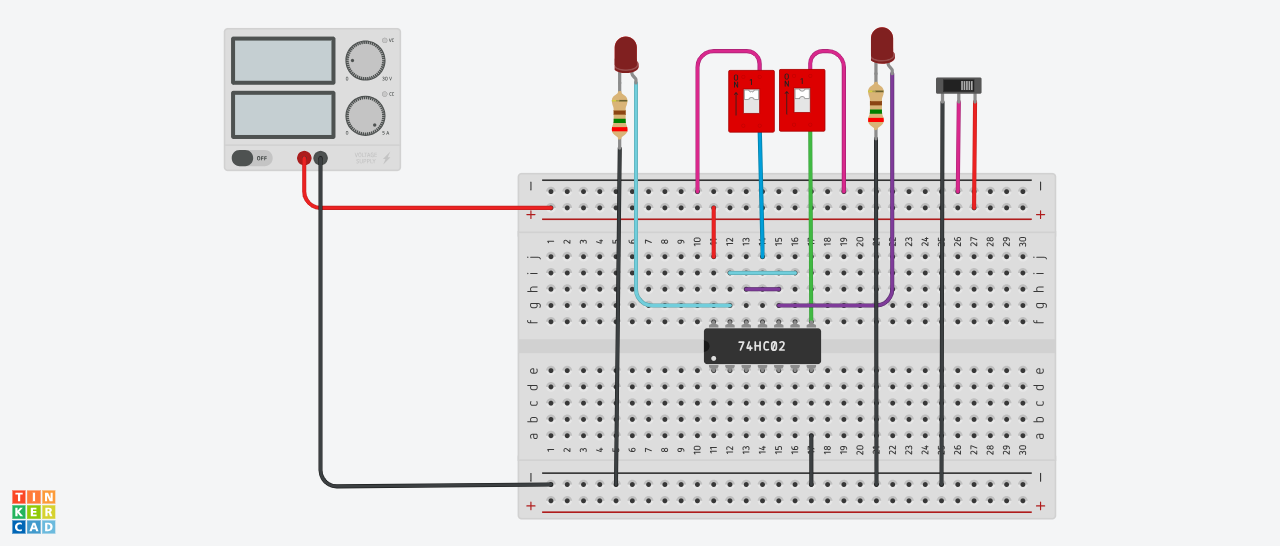
On SR value of 10, Latch gets set to 10.

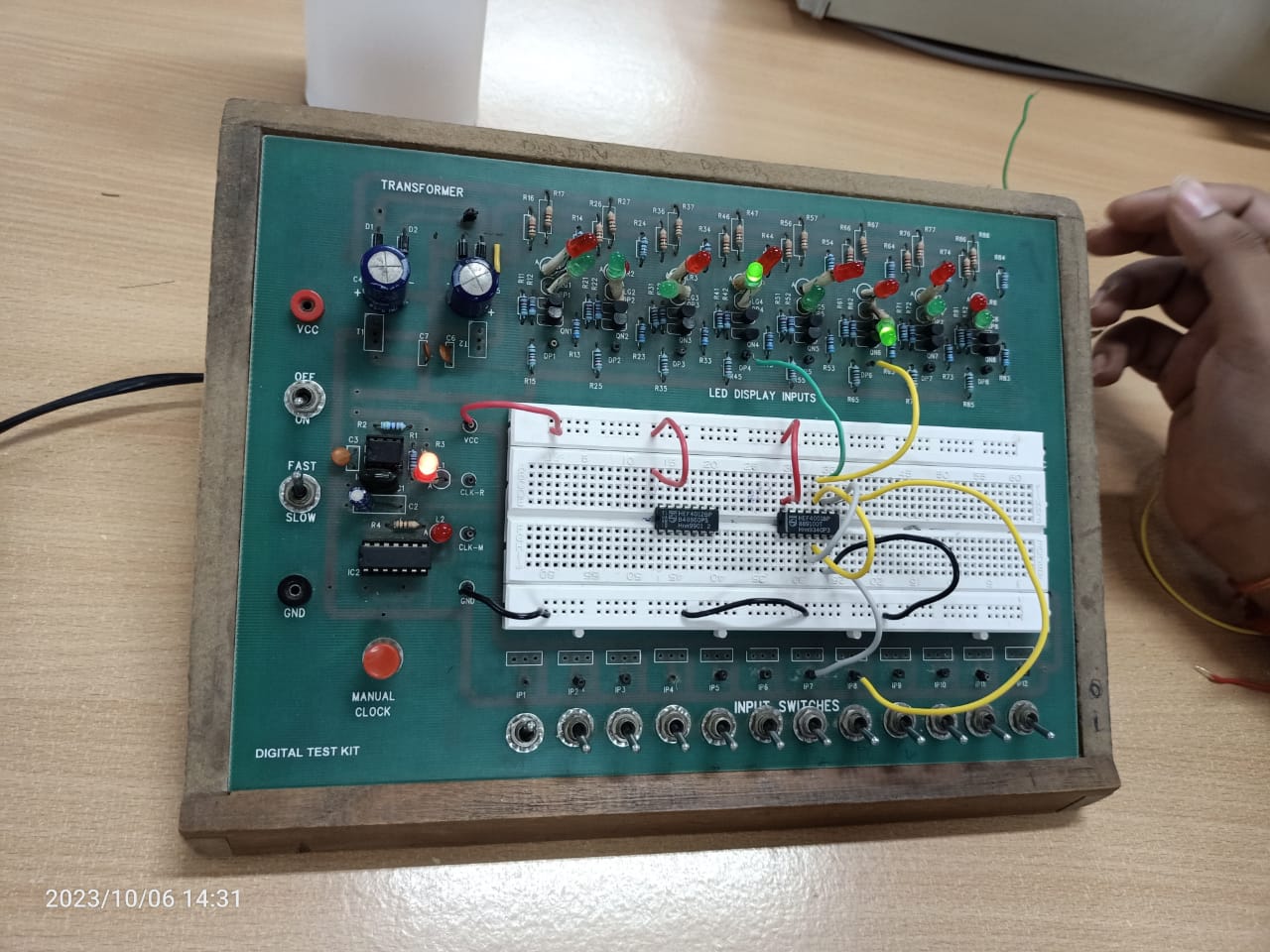
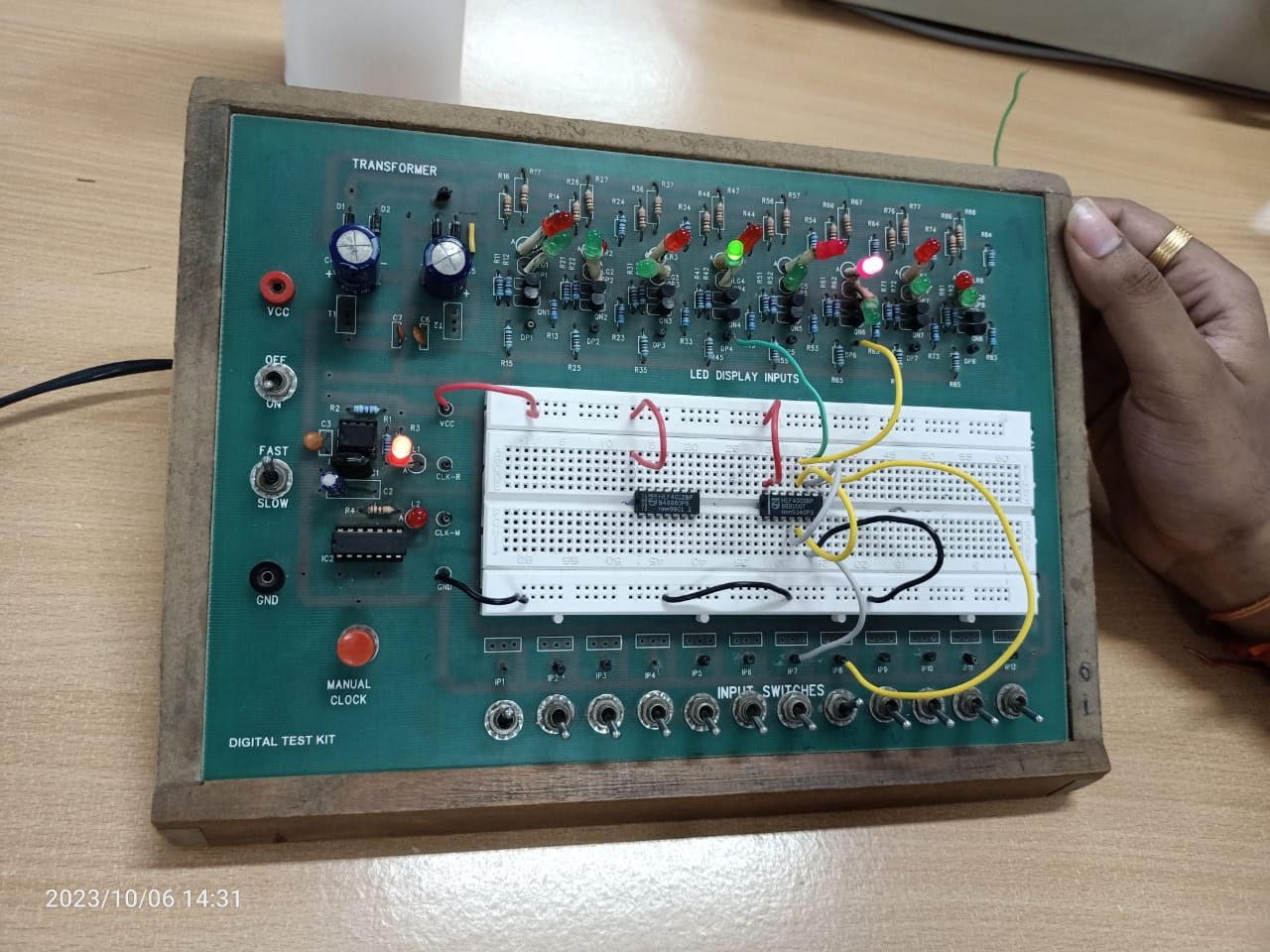
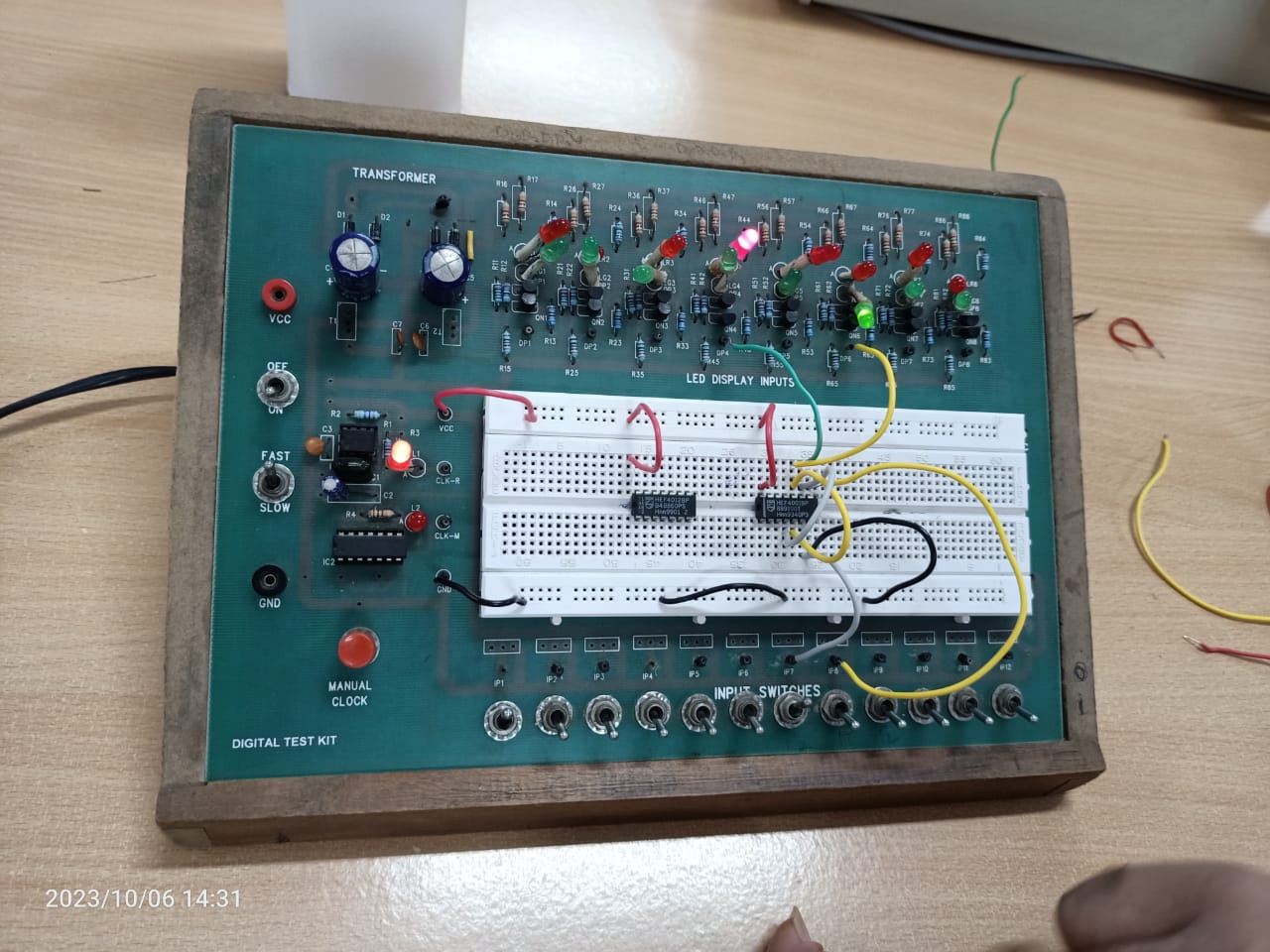
On SR value of 11, Latch gets to forbidden state of 00, which changes differently on applying 00.

**LINK FOR TINKERCAD SIMULATION :**

[**https://www.tinkercad.com/things/7r0Zb17dzcr-sr-latch/editel?sharecode=izayfHMT5YSZk3DXaK5ZqqhrrDPLI9wkxXfVaCVNNW0**](https://www.tinkercad.com/things/7r0Zb17dzcr-sr-latch/editel?sharecode=izayfHMT5YSZk3DXaK5ZqqhrrDPLI9wkxXfVaCVNNW0)

**TINKERCARD :**



LAB:

**2.OBJECTIVE –** To Form JK Flip Flop and Observe its Output.

**ELCTRONIC COMPONENTS REQUIRED –**

1. Digital test kit.
2. 2-Input And Gate IC.
3. 3-Input And gate IC.
4. Inverter IC.
5. LEDs.

**PROCEDURE:**

* Test the ICs, LED Lights and Switches.
* Connect ICs with GND, Power.
* ­­­­­­­Connect the circuit as given in reference circuit.

A diagram of a circuit

Description automatically generated

Observe the output and make the Table.

|  |  |  |  |
| --- | --- | --- | --- |
| J | K | Enable | Q(t+1) |
| 0 | 0 | 0 | Qt |
| 0 | 0 | 1 | Qt |
| 0 | 1 | 0 | Qt |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | Qt |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | Qt |
| 1 | 1 | 1 | Toggle |

Conclusion:

On JK value of 01, It gets reset to 01.

On JK value of 00, It retains the previous value.

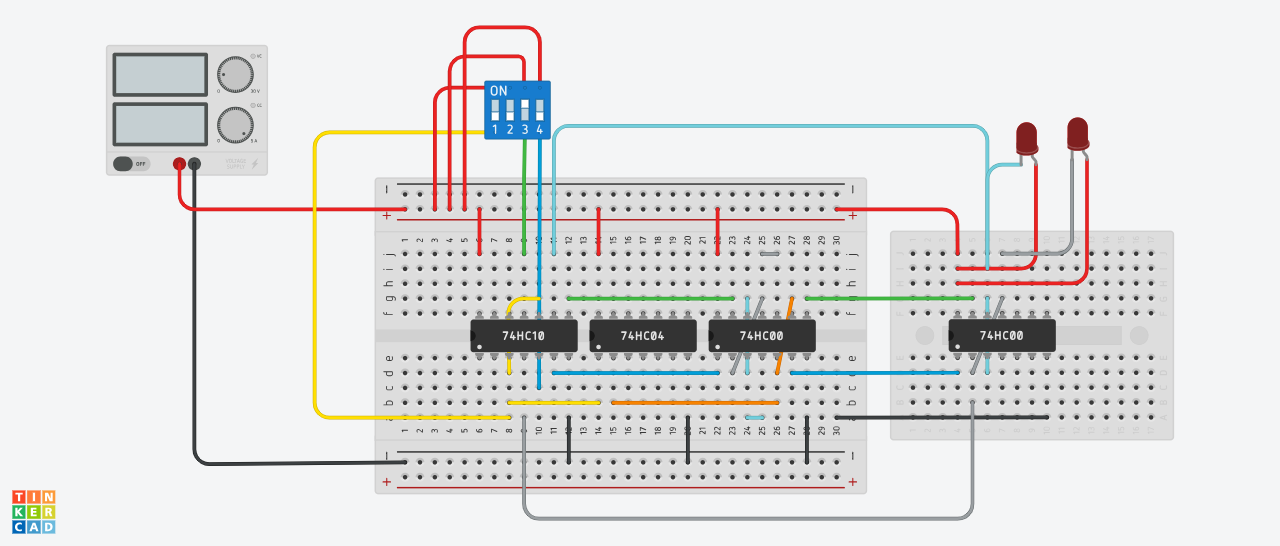
On JK value of 10, It gets set to 10.

On JK value of 11, It gets to toggle state.

**LINK FOR TINKERCAD SIMULATION :**

<https://www.tinkercad.com/things/i5eHVFlc7cu-fantastic-wluff-snaget/editel?sharecode=lXAk7PaWkzR1lc40NGXpDJyytDkwnx5tTiaxJykD9FI>

**TINKERCARD :**



LAB:A green circuit board with wires and a white board

Description automatically generatedA green circuit board with wires and lights

Description automatically generatedA green circuit board with wires and wires

Description automatically generatedA green circuit board with many wires

Description automatically generatedA circuit board with wires and wires

Description automatically generatedA green electronic board with many wires and a white board

Description automatically generated

**3.OBJECTIVE –** To Form 4-Bity UP Down Counter.

**ELCTRONIC COMPONENTS REQUIRED –**

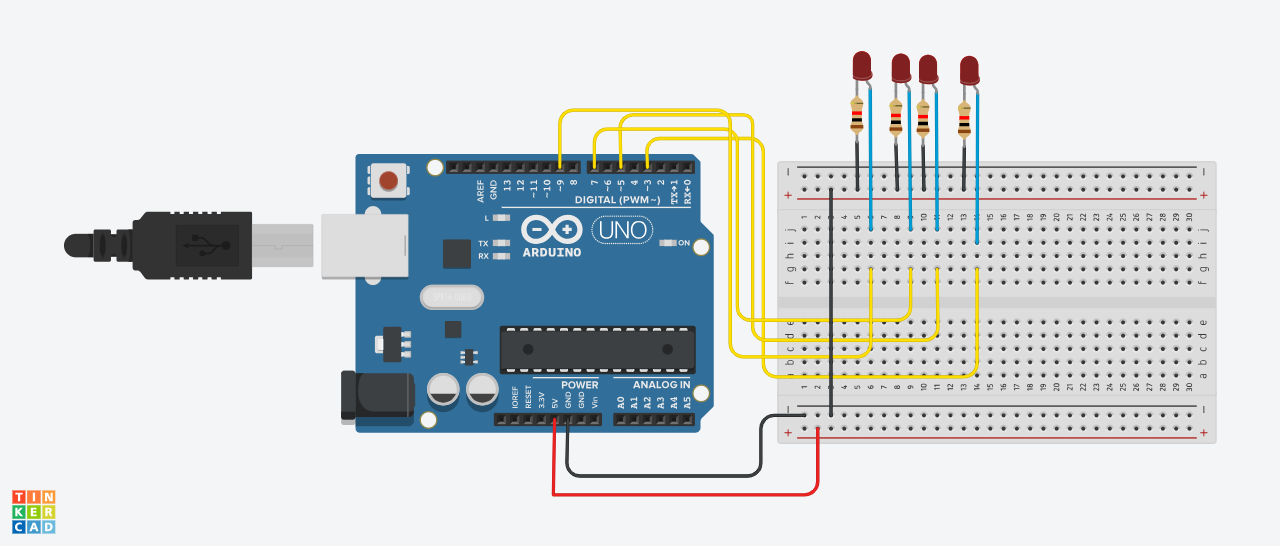
1. Digital test kit.
2. Arduino.
3. LEDs.

**PROCEDURE:**

* Test the LED Lights and Arduino.
* Connect Arduino with GND, Power.
* ­­­­­­­Connect the Input to Arduino and Output it to LEDs.
* Write Code to print them.

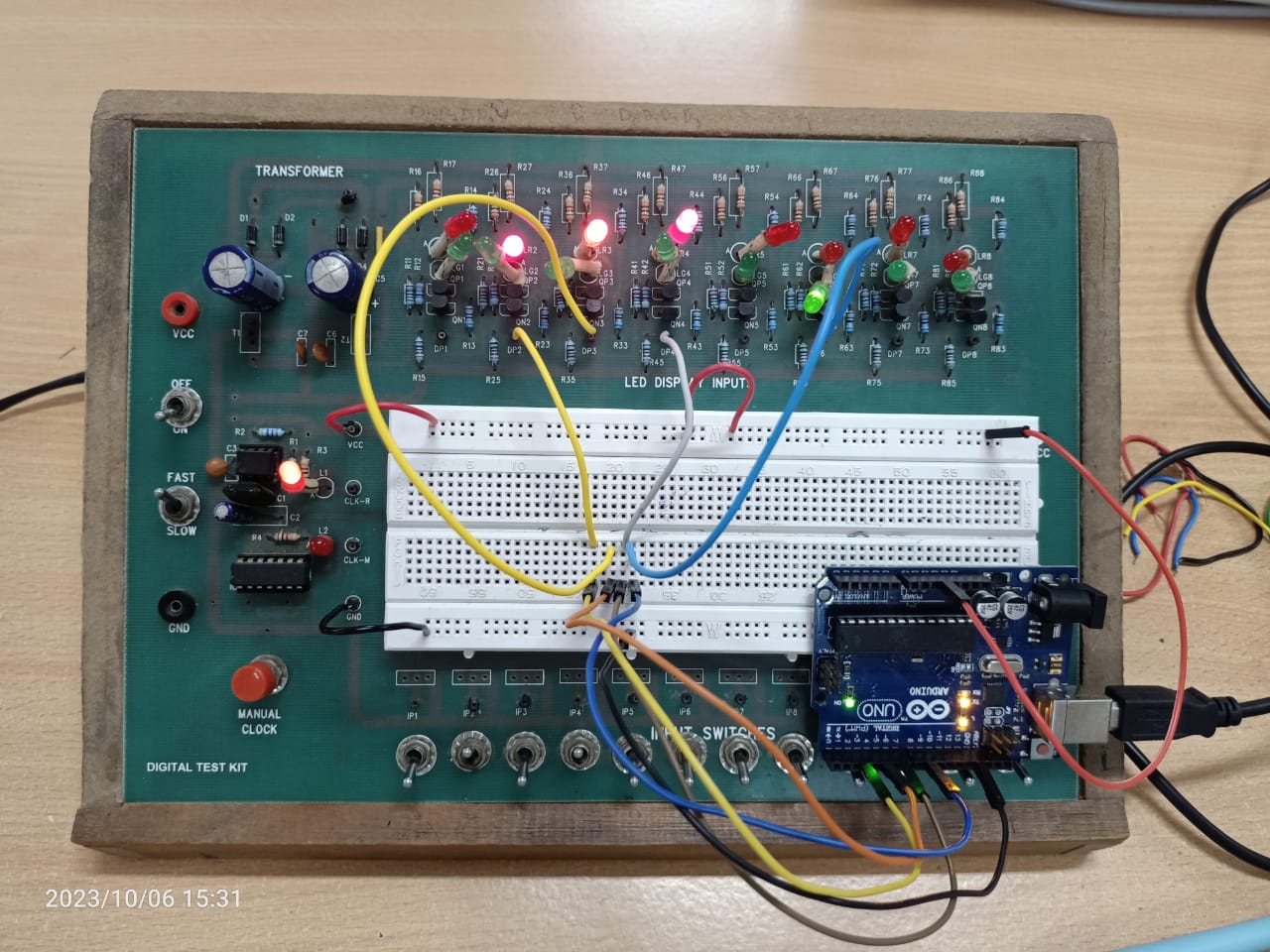
**LINK FOR TINKERCAD SIMULATION :**

<https://www.tinkercad.com/things/0u2KkKv7p1h-printer/editel?sharecode=DUlNu-SgCi8KDesXdnVg3dwje29ufYw0DCeeE1x53o0>

**TINKERCARD :**

LAB:A green board with wires and wires

Description automatically generatedA green board with wires and wires

Description automatically generatedA green electronic board with wires and wires

Description automatically generatedA green circuit board with wires and wires

Description automatically generatedA green board with wires and wires

Description automatically generated

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);

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

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

Timer t;

int pin1 = 3;

int pin2 = 5;

int pin3 = 7;

int pin4 = 9;

int eventId1;

int eventId2;

int eventId3;

int eventId4;

void setup() {

Serial.begin(9600);

pinMode(pin1, OUTPUT);

pinMode(pin2, OUTPUT);

pinMode(pin3, OUTPUT);

pinMode(pin4, OUTPUT);

eventId1 = t.oscillate(pin1, 1000, LOW);

if (eventId1 < 0) {

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

eventId2 = t.oscillate(pin2, 2000, LOW);

if (eventId2 < 0) {

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

eventId3 = t.oscillate(pin3, 4000, LOW);

if (eventId3 < 0) {

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

eventId4 = t.oscillate(pin4, 8000, LOW);

if (eventId4 < 0) {

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

t.every(16000, takeReading);

}

void loop() {

t.update();

}

void takeReading(){

eventId1 = t.oscillate(pin1, 1000, HIGH);

if (eventId1 < 0) {

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

eventId2 = t.oscillate(pin2, 2000, HIGH);

if (eventId2 < 0) {

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

eventId3 = t.oscillate(pin3, 4000, HIGH);

if (eventId3 < 0) {

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

eventId4 = t.oscillate(pin4, 8000, HIGH);

if (eventId4 < 0) {

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

}

void stopAllTimers() {

}

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

}

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

{

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

}

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;

}