LAB REPORT 5

Name:- Ayush Kumar Gupta

ROLL NO:- 2023114001

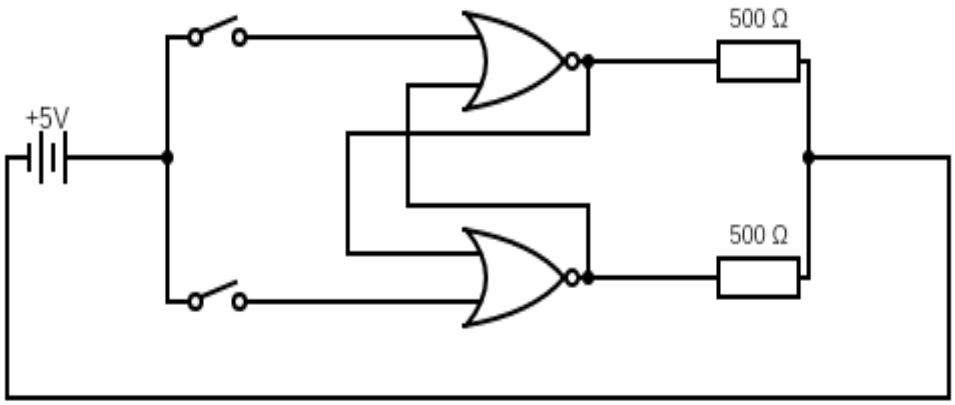
1.OBJECTIVE – To Form RS OR Latch and check input results.

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.



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.

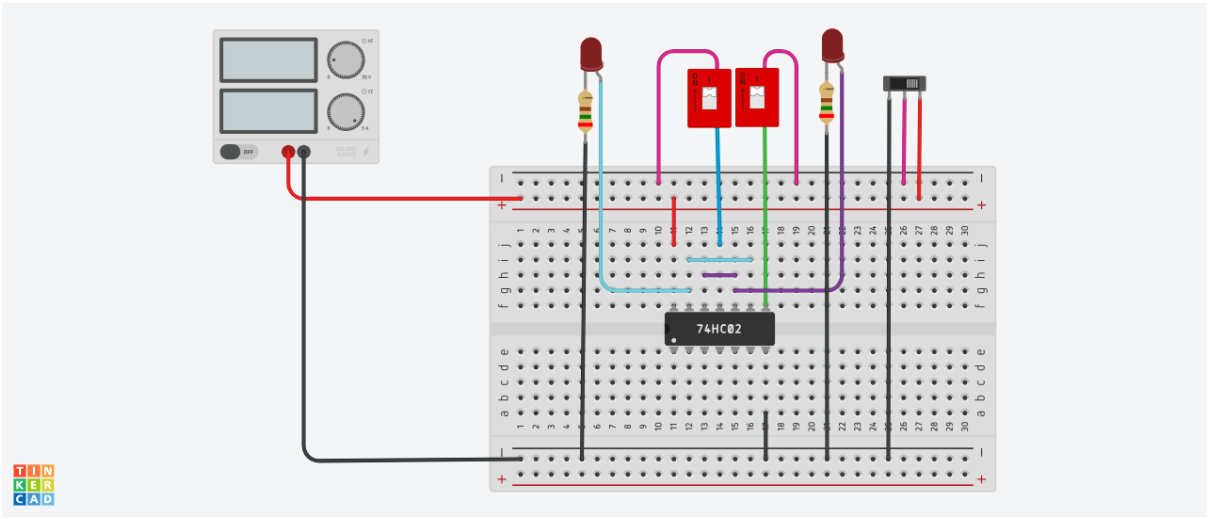
On SR value of 10, Latch gets set to 10.

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

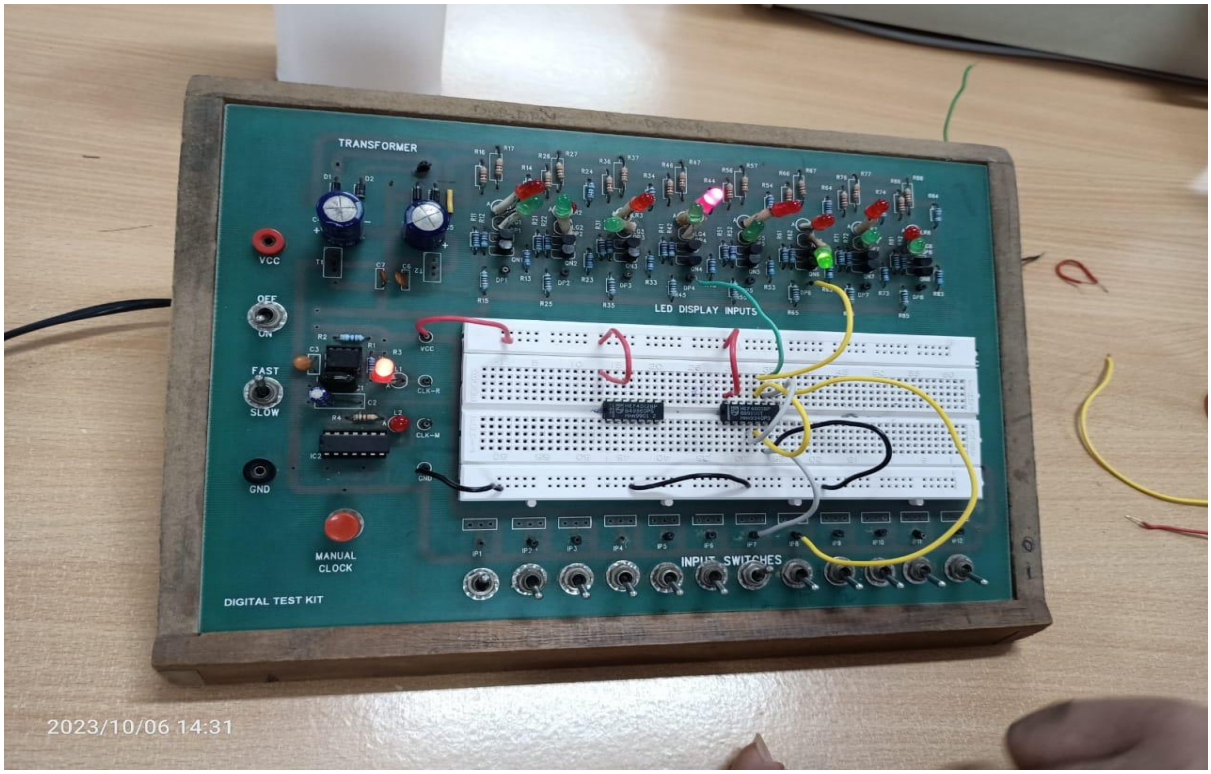
LINK FOR TINKERCAD SIMULATION :

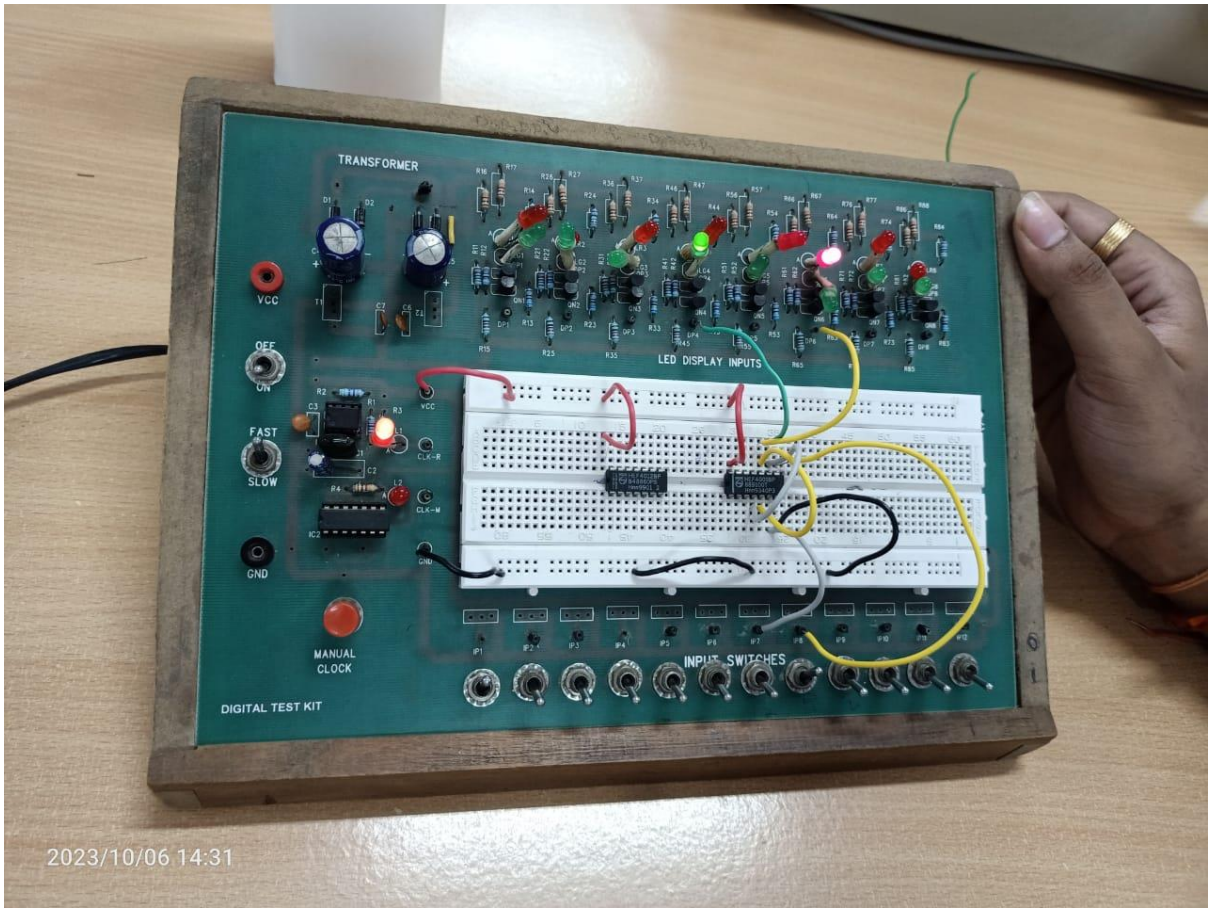
https://www.tinkercad.com/things/7r0Zb17dzcr-sr-latch/editel?sharecode=izayfHMT5YSZk3DXaK5ZqqhrrDPLI9wkxXfVaCVNNW0

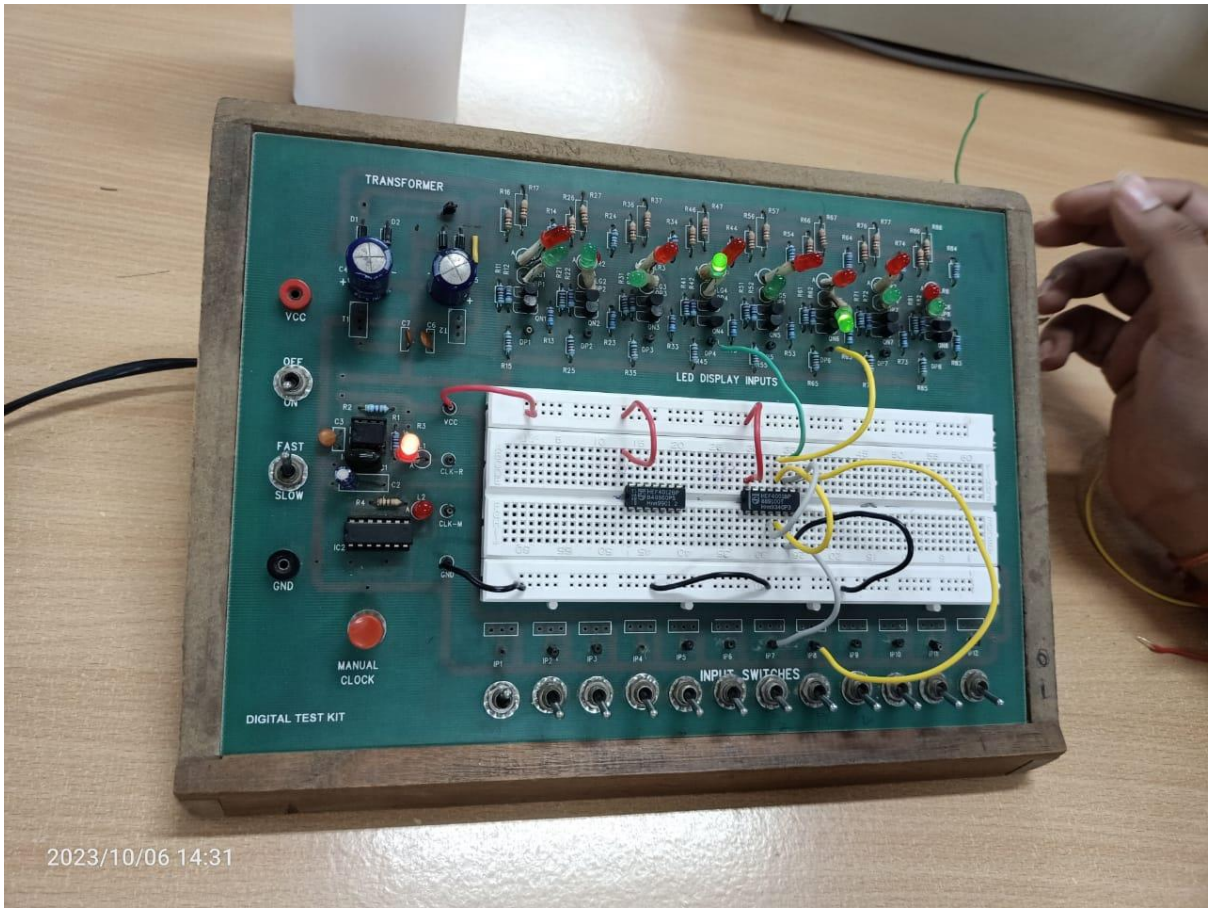
TINKERCARD :



LAB:




TRANSFORMER
VCC
OFF
ON
FAST
SLOW
GND
MANUAL
CLOCK
LED DISPLAY INPUTS
INPUT SWITCHES
DIGITAL TEST KIT
2023/10/06 14:31


TRANSFORMER
VCC
OFF
ON
FAST
SLOW
GND
MANUAL
CLOCK
LED DISPLAY INPUTS
INPUT SWITCHES
DIGITAL TEST KIT
2023/10/06 14:31

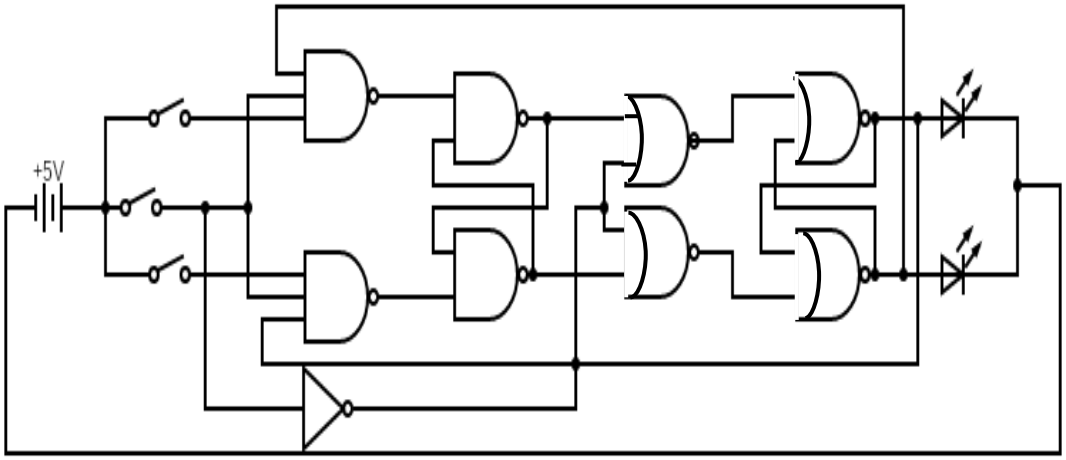
2.OBJECTIVE – To Form JK Flip Flop and Observe its Output.

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.



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.

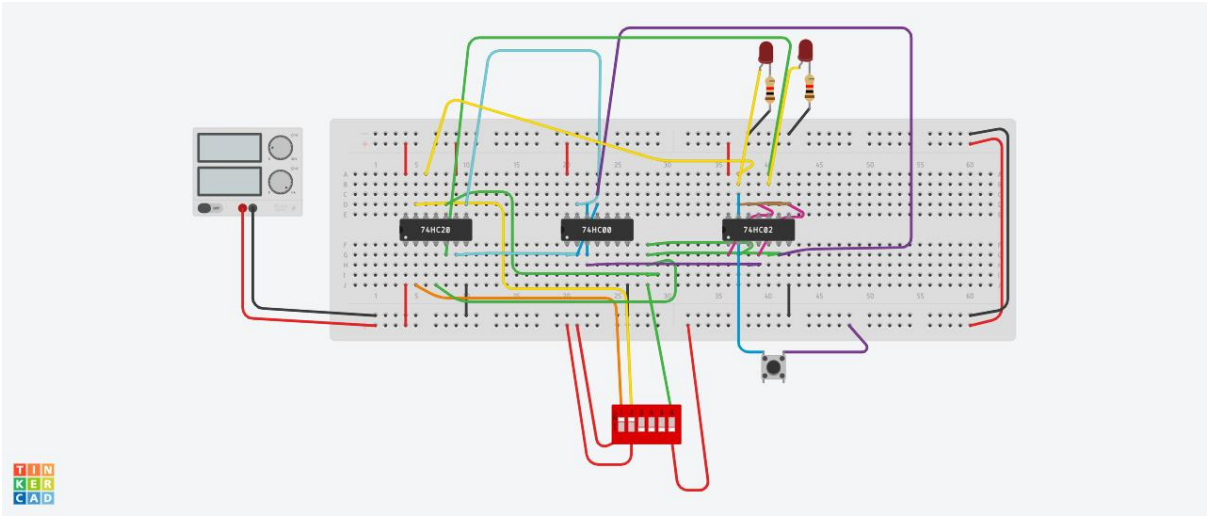
On JK value of 10, It gets set to 10.

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

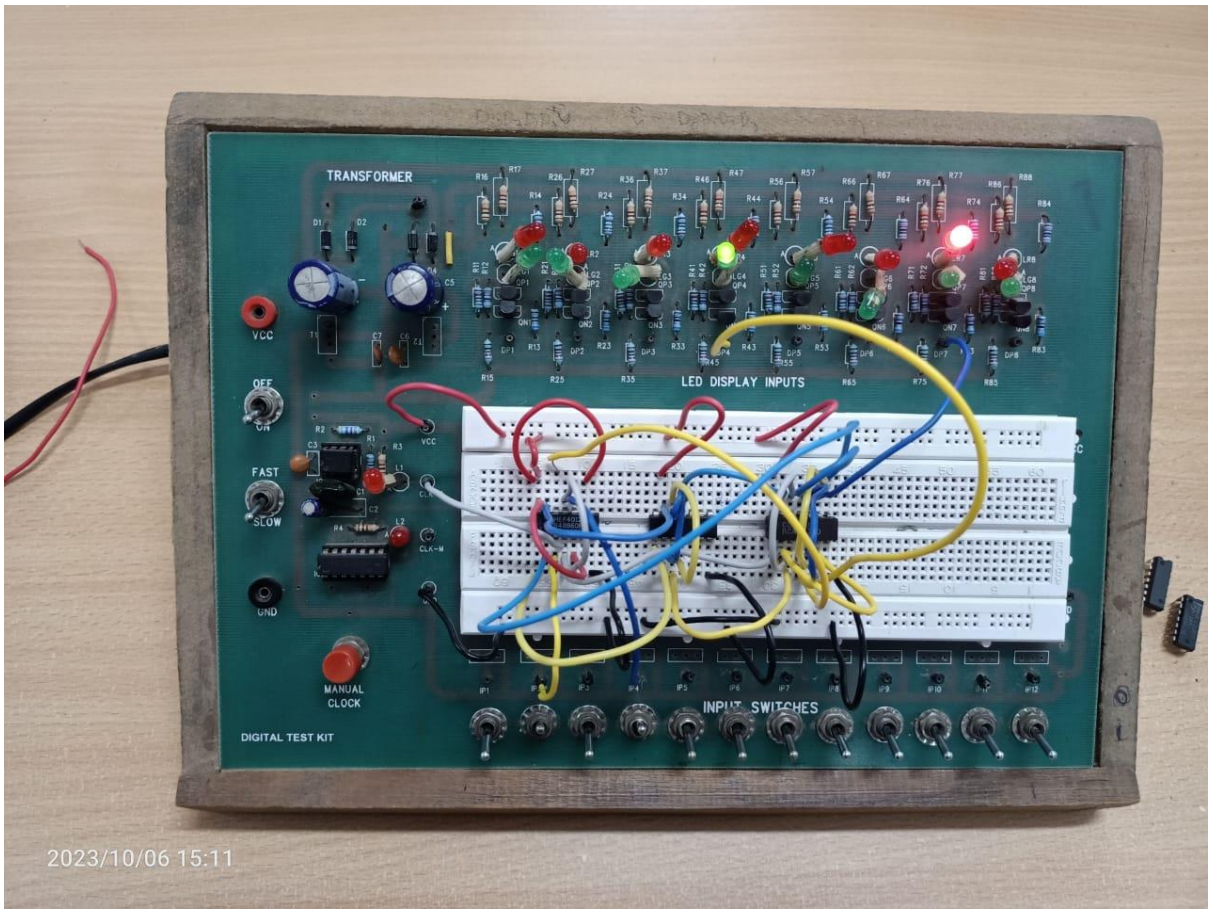
LINK FOR TINKERCAD SIMULATION :

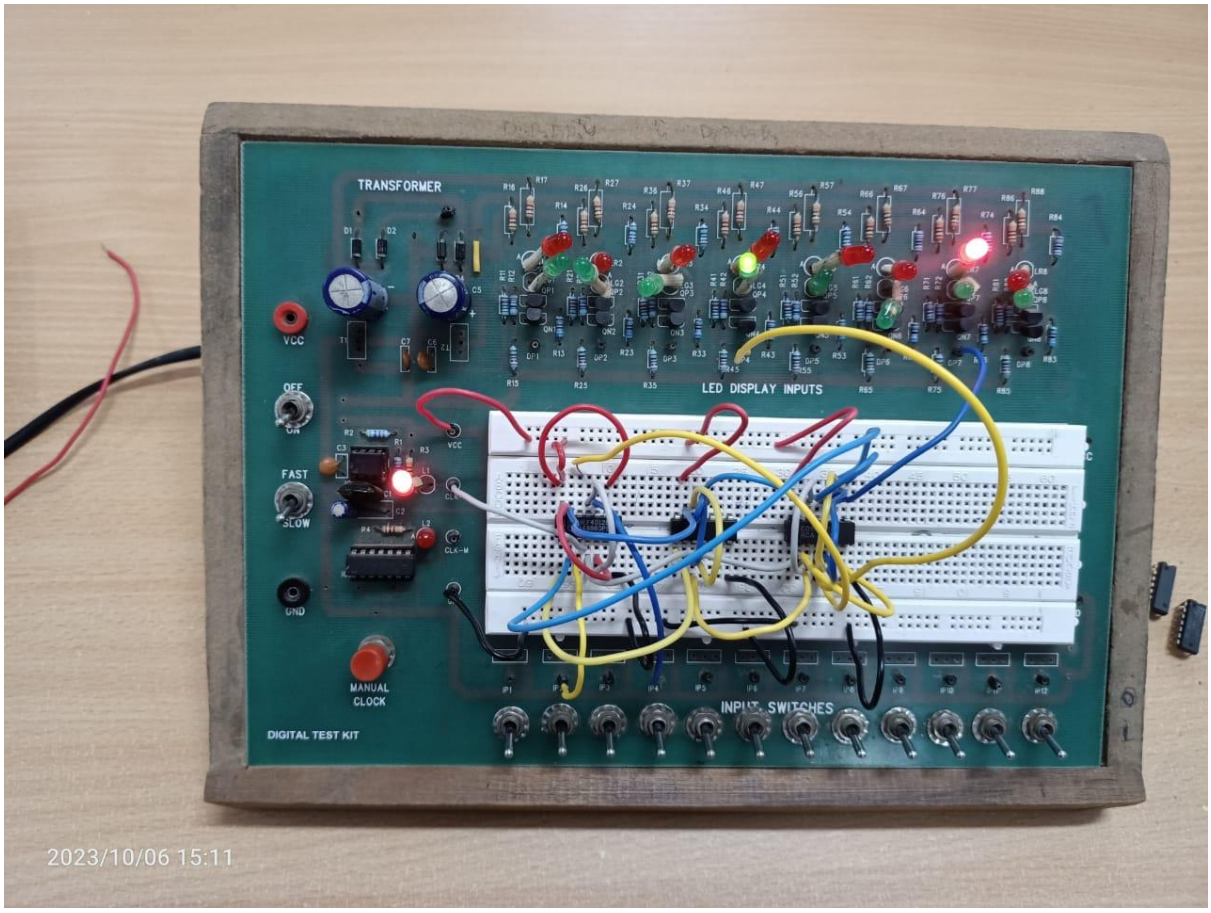
https://www.tinkercad.com/things/i5eHVFlc7cu-fantastic-wluff-snaget/editel?sharecode=lXAk7PaWkzR1lc40NGXpDJyytDkwnx5tTiaxJykD9FI

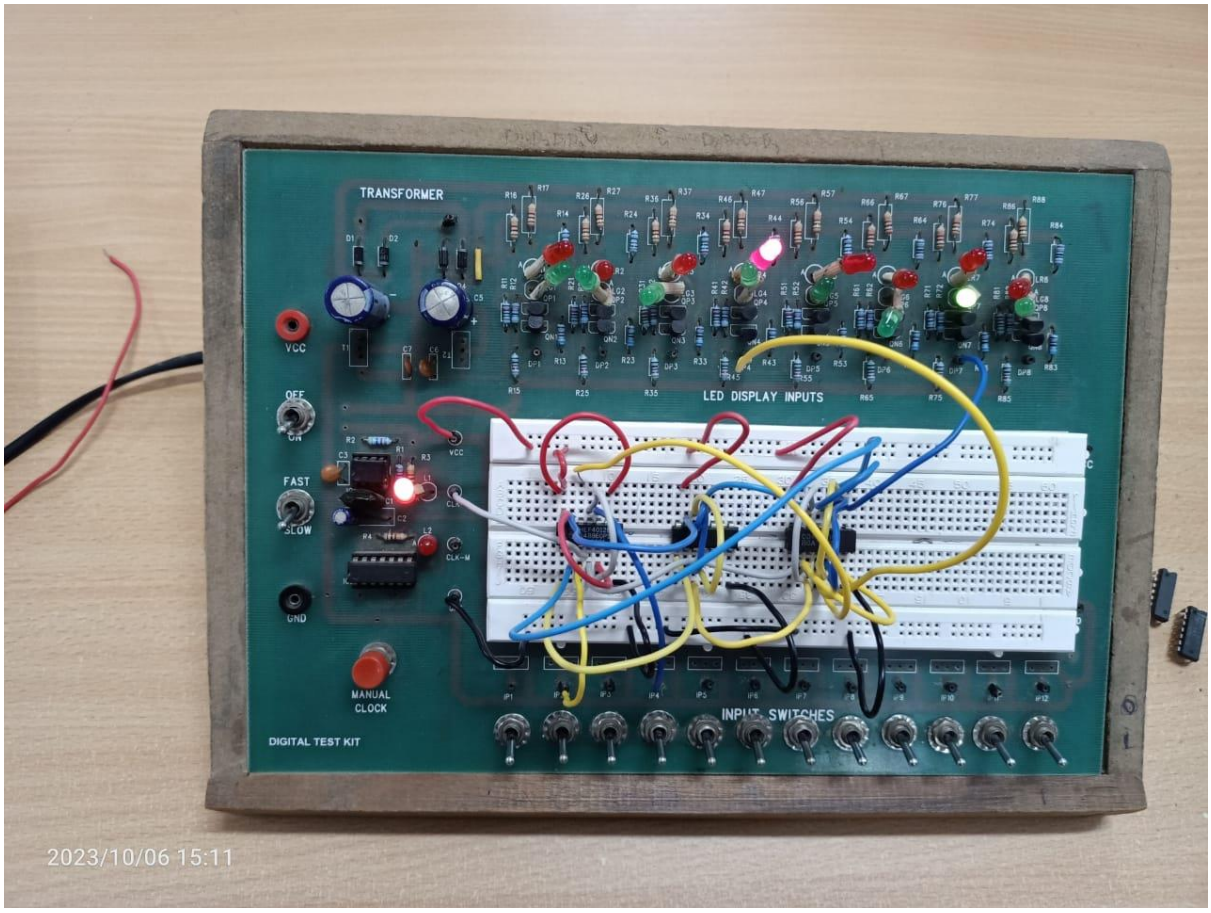
TINKERCARD :

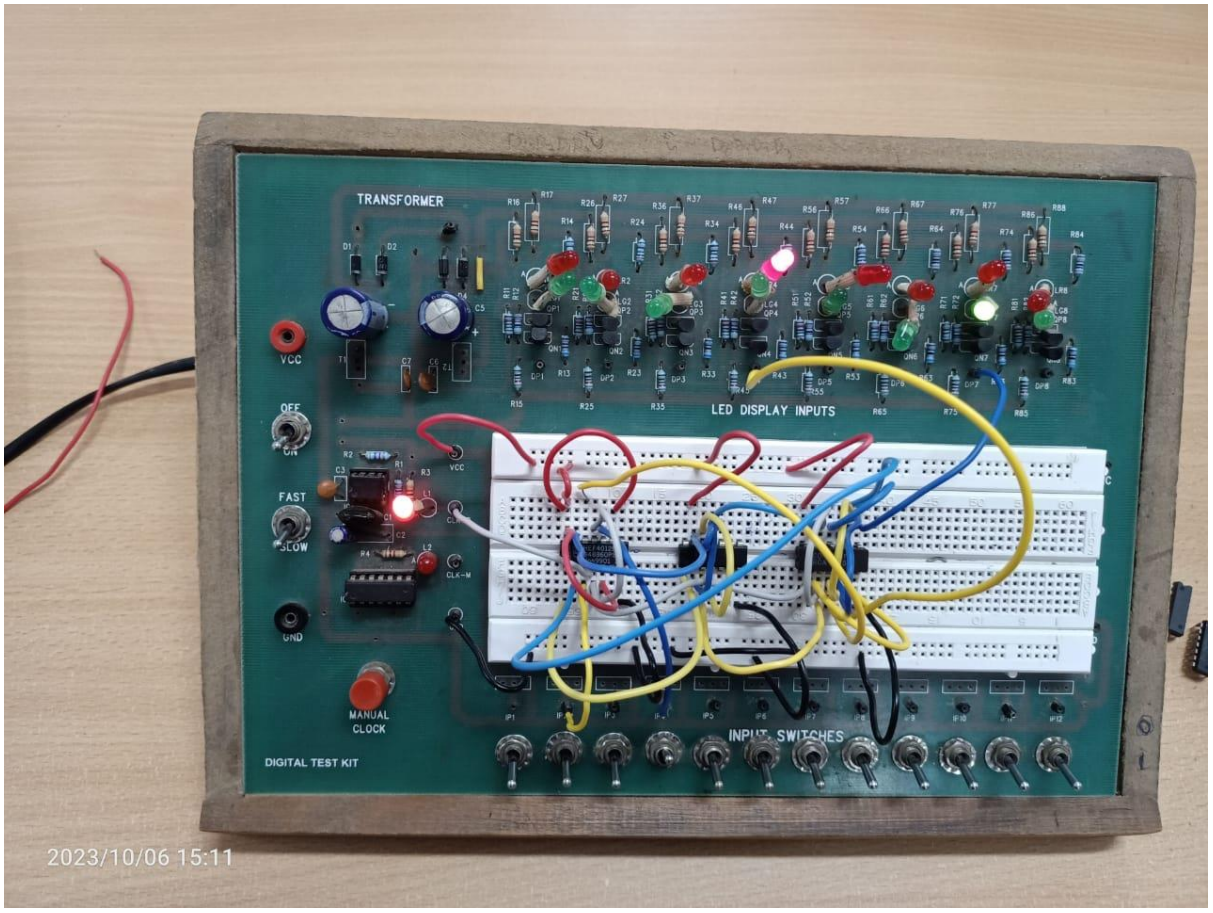


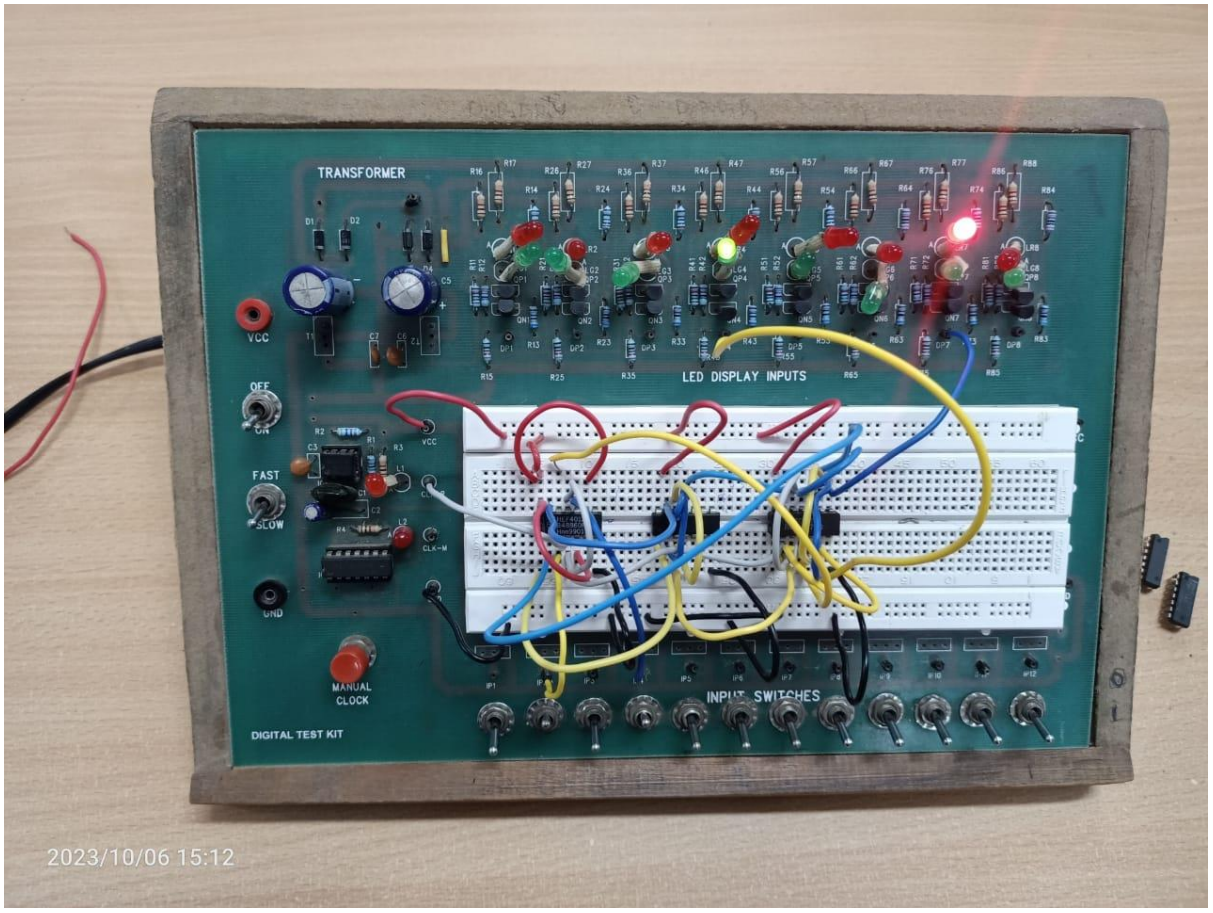
LAB:

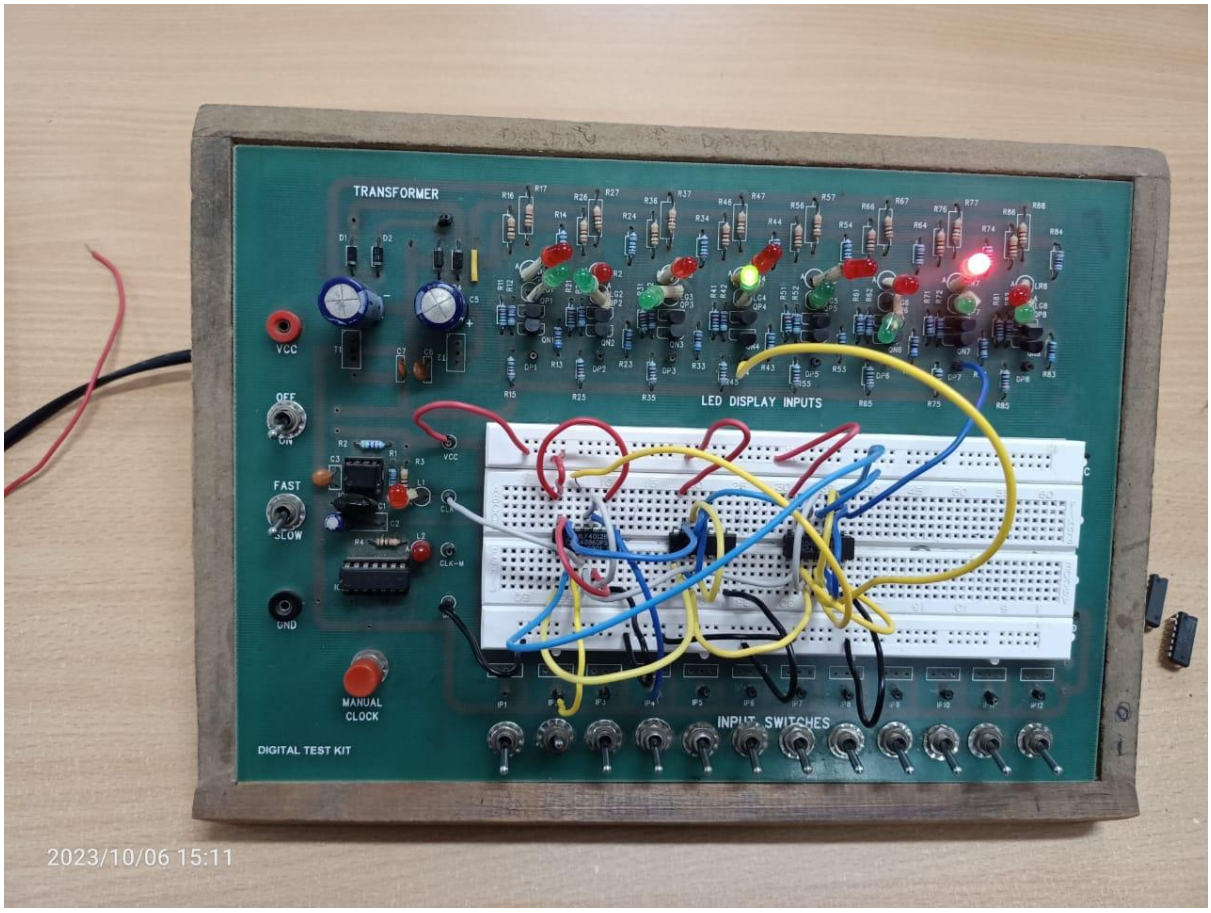



TRANSFORMER
LED DISPLAY INPUTS
VCC
OFF
ON
FAST
SLOW
GND
CLK-M
MANUAL
CLOCK
INPUT SWITCHES
DIGITAL TEST KIT
2023/10/06 15:11


TRANSFORMER
LED DISPLAY INPUTS
VCC
OFF
ON
FAST
SLOW
GND
CLK-M
MANUAL
CLOCK
INPUT SWITCHES
DIGITAL TEST KIT
2023/10/06 15:11


TRANSFORMER
VCC
OFF
ON
FAST
SLOW
GND
MANUAL
CLOCK
LED DISPLAY INPUTS
INPUT SWITCHES
DIGITAL TEST KIT
2023/10/06 15:11


TRANSFORMER
VCC
OFF
ON
FAST
SLOW
GND
MANUAL
CLOCK
LED DISPLAY INPUTS
INPUT SWITCHES
DIGITAL TEST KIT
2023/10/06 15:12



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

ELCTRONIC COMPONENTS REQUIRED –

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

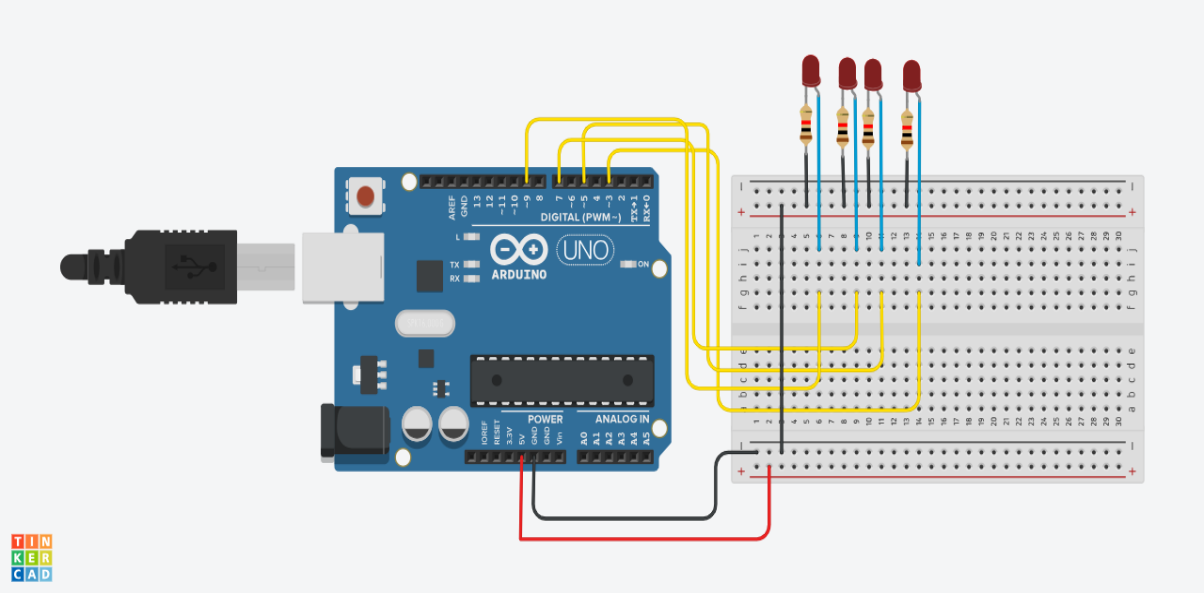
PROCEDURE:

- 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.

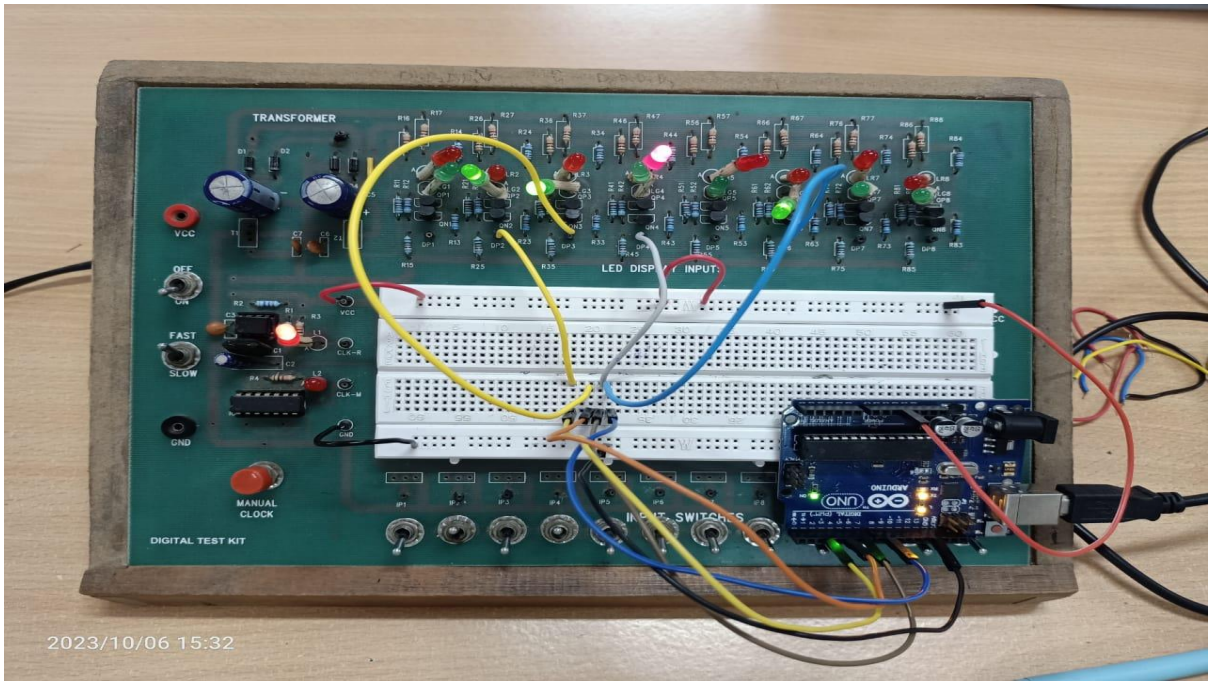
LINK FOR TINKERCAD SIMULATION :

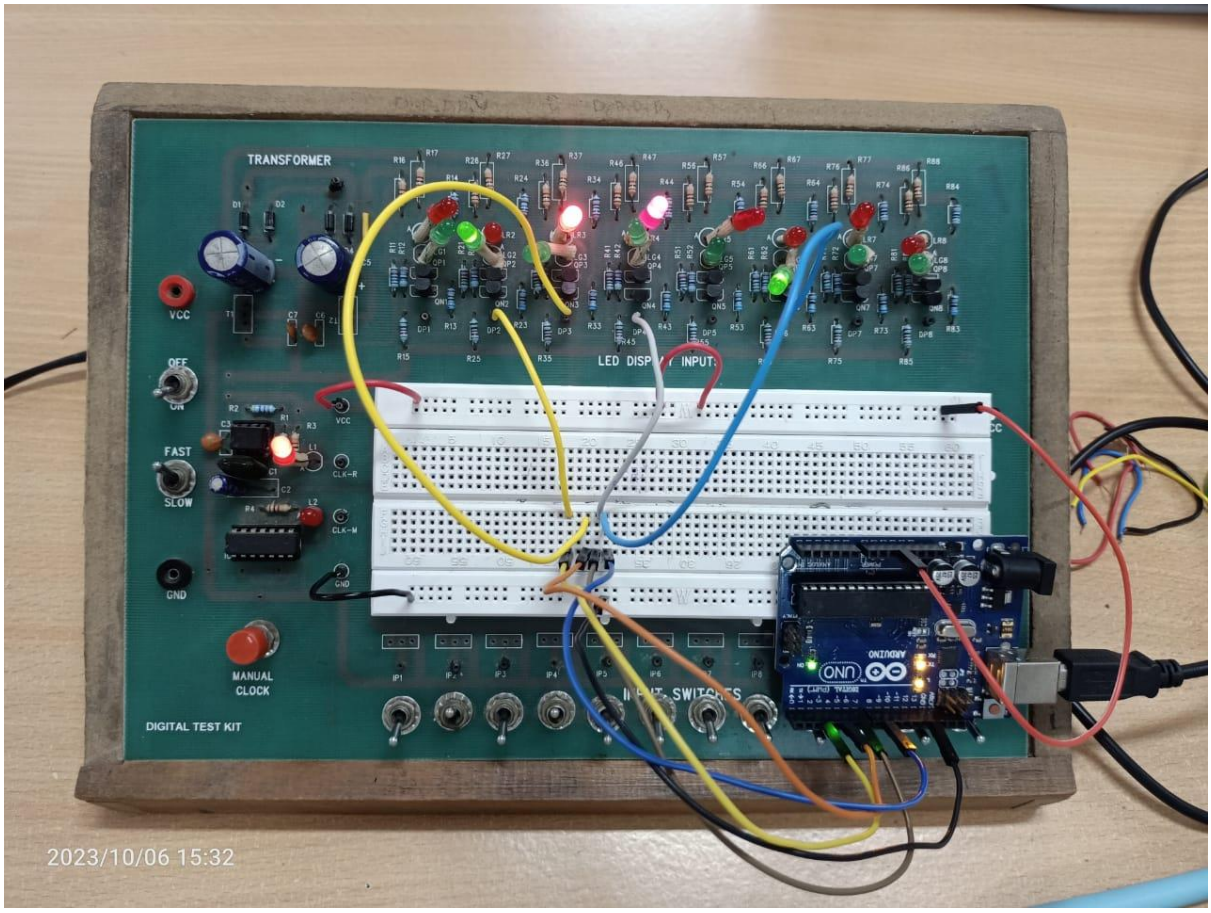
https://www.tinkercad.com/things/0u2KkKv7p1h-printer/editel?sharecode=DUlNu-SgCi8KDesXdnVg3dwje29ufYw0DCeeE1x53o0

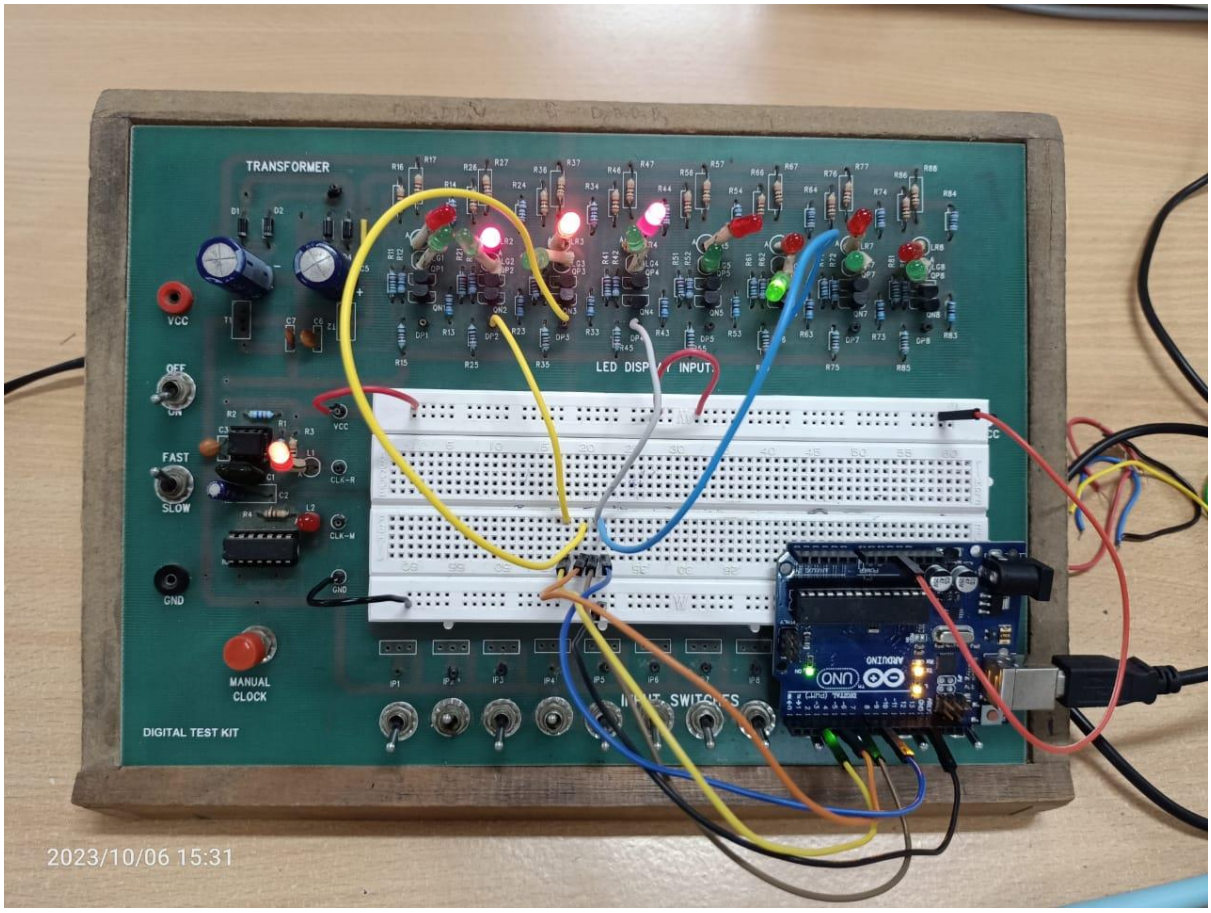
TINKERCARD :

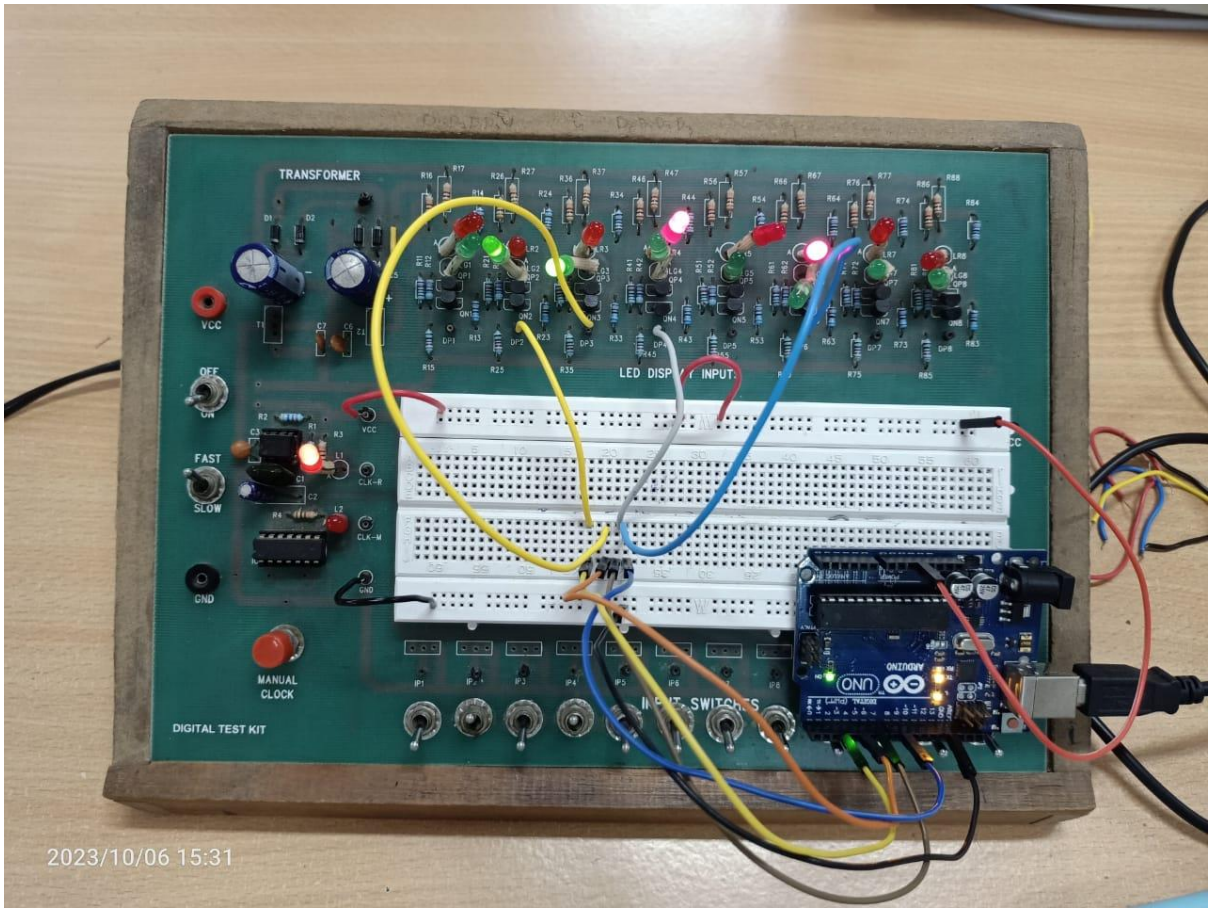


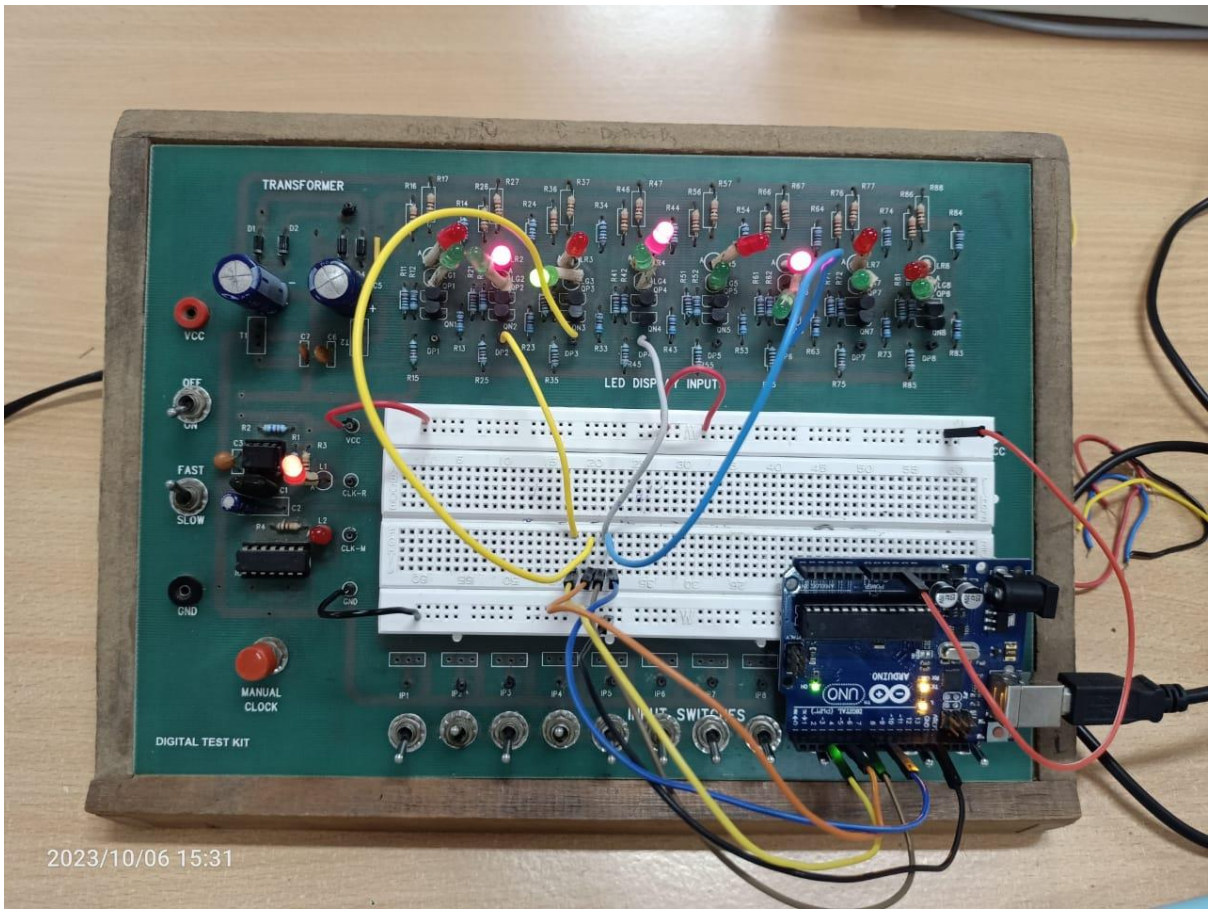
LAB:

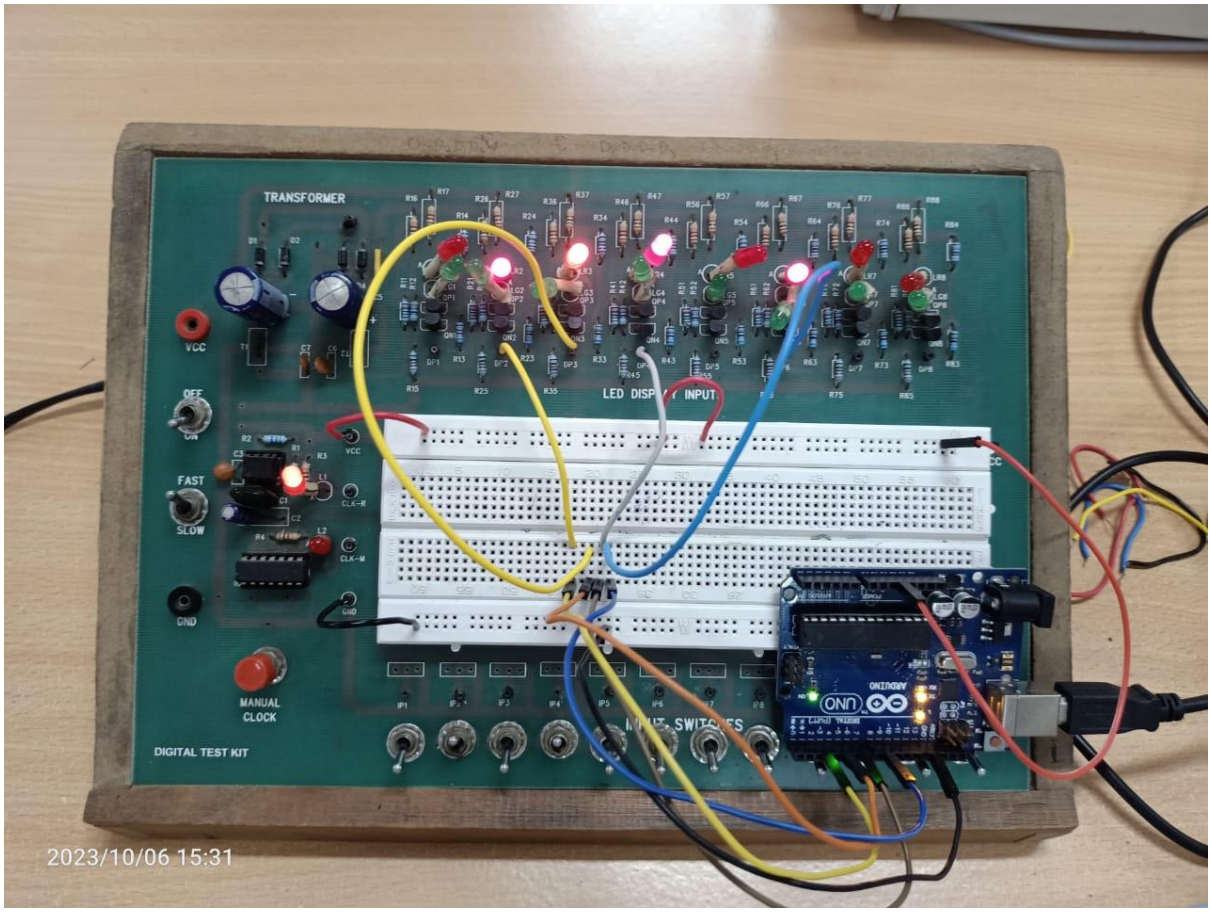



TRANSFORMER
VCC
OFF
ON
FAST
SLOW
GND
MANUAL
CLOCK
DIGITAL TEST KIT
2023/10/06 15:32


TRANSFORMER
VCC
OFF
ON
FAST
SLOW
GND
MANUAL
CLOCK
DIGITAL TEST KIT
2023/10/06 15:31


TRANSFORMER
VCC
OFF
ON
FAST
SLOW
GND
MANUAL
CLOCK
DIGITAL TEST KIT
2023/10/06 15:31


TRANSFORMER
VCC
OFF
ON
FAST
SLOW
GND
MANUAL
CLOCK
DIGITAL TEST KIT
2023/10/06 15:31



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;
}
```