Mess Entry Counter System Using Up/Down Counter

Lab Assignment : 08

EE1200: Electrical Circuits Lab

Harshil Rathan Y **EE24BTECH11064**
Y Akhilesh **EE24BTECH11066**

Contents

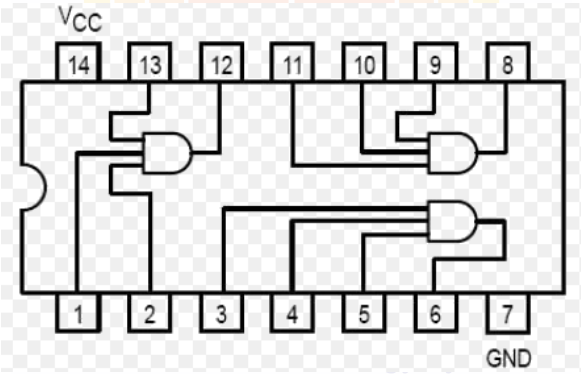
1 Experiment Objective

To design and implement a **digital up/down counter** that displays the number of people currently in the mess during peak lunch hours. The system will help students decide whether they can enter the mess based on the current occupancy. The maximum count is set to **99**.

2 Components Used

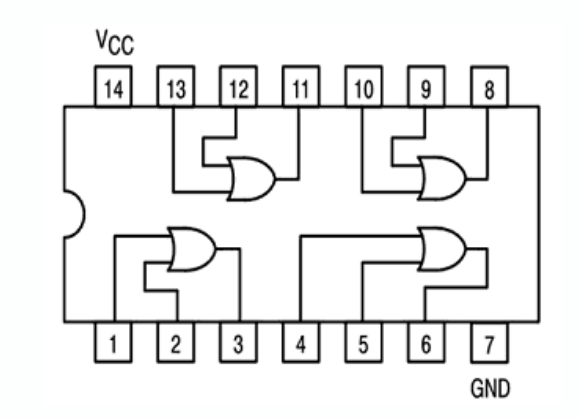
1) Arduino Uno
2) 2 Seven-Segment Displays
3) 2 Push Buttons
4) 4 7476 JK Flip-Flop
5) 11 7411 AND Gate IC's
6) 6 7432 OR Gate IC's
7) 2 **7447** BCD to 7-Segment Decoder IC's
8) Breadboard & Jumper Wires
9) 2 Resistors - 220Ω
10) 2 LEDs

*2.1 **7411** IC*



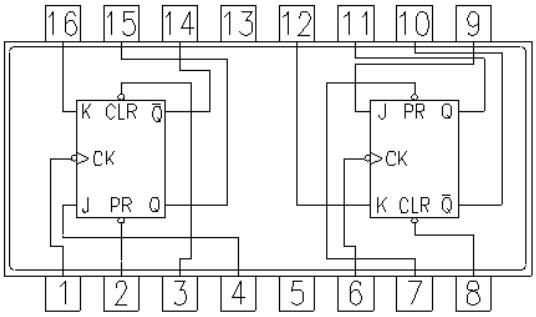
- 7411 Triple 3-Input AND Gate IC .
- We set one input to high when we need only 2 input and gate.
- Used to build multiplication logic

2.2 ***7432*** *IC*



- 7432 Quad 2-Input OR Gate IC .
- It has four independent gates as given in the diagram.
- Used to build addition logic

2.3 ***7476*** *IC*



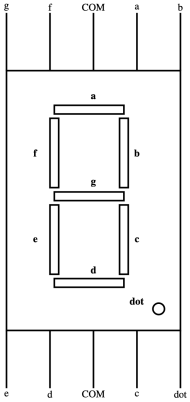
7476
Dual J/K M/S Flip–Flop
with Preset and Clear

- 7476 Dual JK FLip Flop IC's
- Each IC has 2 independent JK Flip Flops
- Used to build increment and decremnet logic based on state diagram
- For this exp we used 4 independent JK Flip flops

2.4 *7447 IC*

- This is a BCD decoder for a 7 segemnt display
- It has 4 inputs A B C and D
- Connected those inputs to Q_A, Q_B, Q_C, Q_D in each condition for a displays
- It controls the outputs accordingly

2.5 *7segment Display*

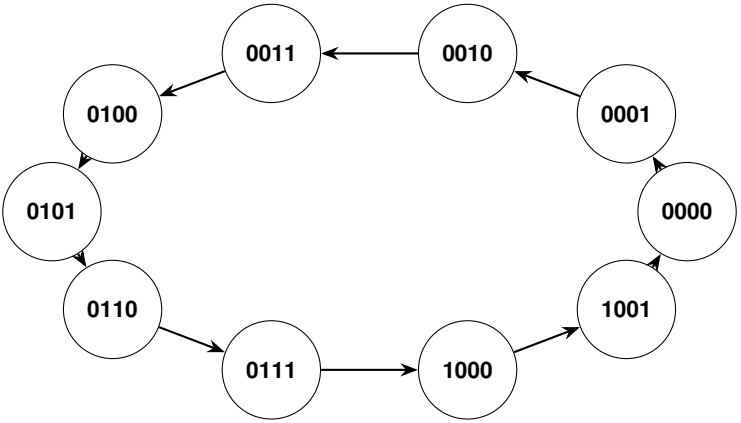


- It takes inputs a, b, c, d, e, f, g and display the digit accordingly
- it can show numbers 0-9, we used 2 such displays

3 Mod-10 Synchronous Up Counter

- No of states count by counter = N = 10.
- Range of counting = 0 to N-1 = 0 to 9.
- Number of Flip Flops = 2 IC's

3.1 State Diagram



3.2 State Table

State	Present State				Next State				Inputs (T Flip-Flops)			
	Q_A	Q_B	Q_C	Q_D	Q_{A+1}	Q_{B+1}	Q_{C+1}	Q_{D+1}	T_A	T_B	T_C	T_D
0	0	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	1	0	0	0	1	1
2	0	0	1	0	0	0	1	1	0	0	0	1
3	0	0	1	1	0	1	0	0	0	1	1	1
4	0	1	0	0	0	1	0	1	0	0	0	1
5	0	1	0	1	0	1	1	0	0	0	1	1
6	0	1	1	0	0	1	1	1	0	0	0	1
7	0	1	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	1	0	0	1	0	0	0	1
9	1	0	0	1	0	0	0	0	1	0	0	1
10	1	0	1	0	x	x	x	x	x	x	x	x
11	1	0	1	1	x	x	x	x	x	x	x	x
12	1	1	0	0	x	x	x	x	x	x	x	x
13	1	1	0	1	x	x	x	x	x	x	x	x
14	1	1	1	0	x	x	x	x	x	x	x	x
15	1	1	1	1	x	x	x	x	x	x	x	x

- Since the 7 segment dispaly displays 0-9 we igonre states from 10-15
- States from 10 to 15 are of **Don't care conditions**

3.3 Expression for T_A, T_B, T_C, T_D

$$T_A = \sum m(7, 9) + d(10, 11, 12, 13, 14, 15)$$
$$T_B = \sum m(3, 7) + d(10, 11, 12, 13, 14, 15)$$
$$T_C = \sum m(1, 2, 5, 7) + d(10, 11, 12, 13, 14, 15)$$
$$T_D = \sum m(0, 1, 2, 3, 4, 5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$$

3.3.1 K-Map for T_A

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	x	x	x	x
10	0	1	x	x

$$T_A = Q_A \cdot Q_D + Q_B \cdot Q_C \cdot Q_D$$

3.3.2 K-Map for T_B

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	x	x	x	x
10	0	0	x	x

$$T_B = Q_C \cdot Q_D$$

3.3.3 K-Map for T_C

$Q_A Q_B$ \ $Q_C Q_D$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	x	x	x	x
10	0	0	x	x

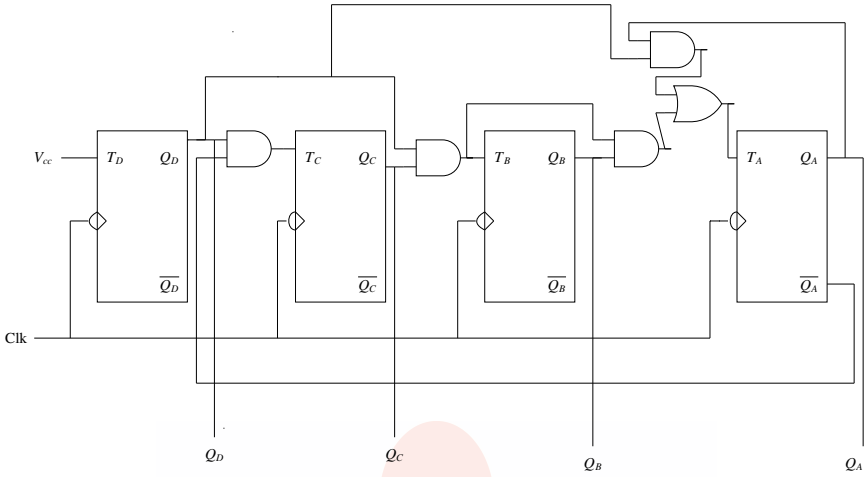
$$T_C = \overline{Q_A} \cdot Q_D$$

3.3.4 K-Map for T_D

$Q_A Q_B$ \ $Q_C Q_D$	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	x	x	x	x
10	1	1	x	x

$$T_D = 1$$

3.4 Circuit Diagram



Circuit Logic Summary

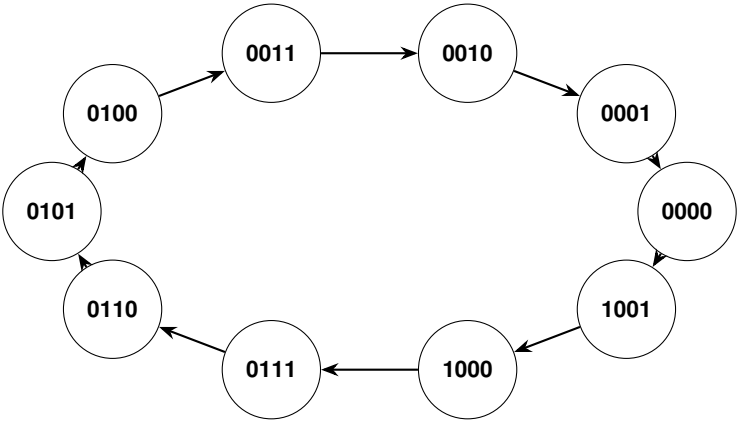
- $$T_D = 1$$
- $$T_C = \overline{Q_A} \cdot Q_D$$
- $$T_B = Q_C \cdot Q_D$$
- $$T_A = Q_A \cdot Q_D + Q_B \cdot Q_C \cdot Q_D$$

Implement and connect the following logics according to the circuit diagram above with the help of the gates

4 Mod-10 Synchronous Down Counter

- No of states count by counter = N = 10.
- Range of counting = N-1 to 0 = 9 to 0.
- Required Flip Flops = 2 IC's

4.1 State Diagram



4.2 State table

State	Present State				Next State				Inputs (T Flip-Flops)			
	Q_A	Q_B	Q_C	Q_D	Q_{A+1}	Q_{B+1}	Q_{C+1}	Q_{D+1}	T_A	T_B	T_C	T_D
9	1	0	0	1	1	0	0	0	1	0	0	1
8	1	0	0	0	0	1	1	1	1	1	1	1
7	0	1	1	1	0	1	1	0	1	0	0	1
6	0	1	1	0	0	1	0	1	1	0	1	1
5	0	1	0	1	0	1	0	0	1	0	0	1
4	0	1	0	0	0	0	1	1	1	1	1	1
3	0	0	1	1	0	0	1	0	1	0	0	1
2	0	0	1	0	0	0	0	1	1	0	1	1
1	0	0	0	1	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	1	0	0	1
10	1	0	1	0	x	x	x	x	x	x	x	x
11	1	0	1	1	x	x	x	x	x	x	x	x
12	1	1	0	0	x	x	x	x	x	x	x	x
13	1	1	0	1	x	x	x	x	x	x	x	x
14	1	1	1	0	x	x	x	x	x	x	x	x
15	1	1	1	1	x	x	x	x	x	x	x	x

4.3 Expression for T_A, T_B, T_C, T_D

$$T_A = \sum m(0, 8) + d(10, 11, 12, 13, 14, 15)$$
$$T_B = \sum m(4, 8) + d(10, 11, 12, 13, 14, 15)$$
$$T_C = \sum m(0, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$$
$$T_D = \sum m(0, 1, 2, 3, 4, 5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$$

4.3.1 K-Map for T_A

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	x	x	x	x
10	1	0	x	x

$$T_A = \overline{Q_B} \cdot \overline{Q_A} \cdot \overline{Q_D}$$

4.3.2 K-Map for T_B

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	x	x	x	x
10	1	0	x	x

$$T_B = (Q_A + Q_B) \cdot \overline{Q_C} \cdot \overline{Q_D}$$

4.3.3 K-Map for T_C

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	0	0	0	1
01	1	0	0	1
11	x	x	x	x
10	1	0	x	x

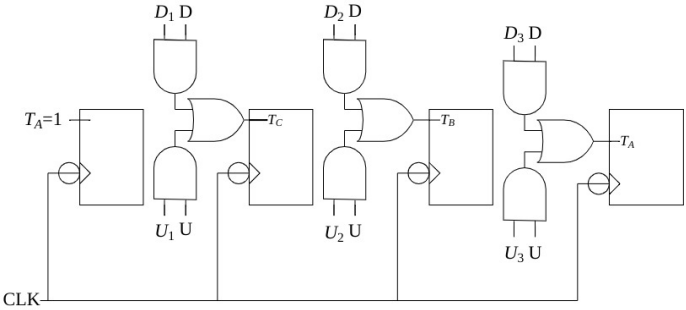
$$T_C = Q_C \cdot \overline{Q_D} + T_B$$

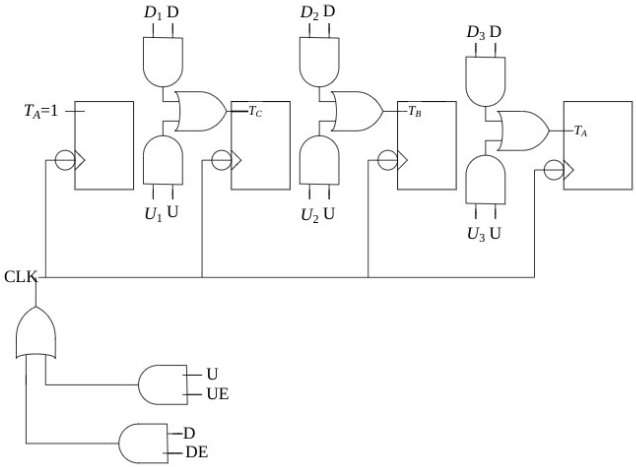
4.3.4 K-Map for T_D

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	x	x	x	x
10	1	1	x	x

$$T_D = 1$$

4.4 Circuit diagram





where,

$$D_1 = Q_B\overline{Q_C}\cdot\overline{Q_D} + Q_A\overline{Q_C}\cdot\overline{Q_D} + Q_C\overline{Q_D}$$

$$D_2 = Q_B\overline{Q_C}\cdot\overline{Q_D} + Q_A\overline{Q_C}\cdot\overline{Q_D}$$

$$D_3 = \overline{Q_B}\cdot\overline{Q_C}\cdot\overline{Q_D}$$

$$U_1 = \overline{Q_A}Q_D$$

$$U_2 = Q_CQ_D$$

$$UE = Q_A\overline{Q_B}\cdot\overline{Q_C}Q_D$$

$$DE = \overline{Q_A} \cdot \overline{Q_B} \cdot \overline{Q_C} \cdot \overline{Q_D}$$

$$U_3 = Q_A Q_D + Q_B Q_C Q_D$$

$$U \rightarrow Up$$

$$D \rightarrow Down$$

Circuit Logic Summary

- $$T_D = 1$$
- $$T_C = Q_C \cdot \overline{Q_D} + T_B$$
- $$T_B = (Q_A + Q_B) \cdot \overline{Q_C} \cdot \overline{Q_D}$$
- $$T_A = \overline{Q_B} \cdot \overline{Q_A} \cdot \overline{Q_D}$$

Implement and connect the following logics according to the circuit diagram above with the help of the gates

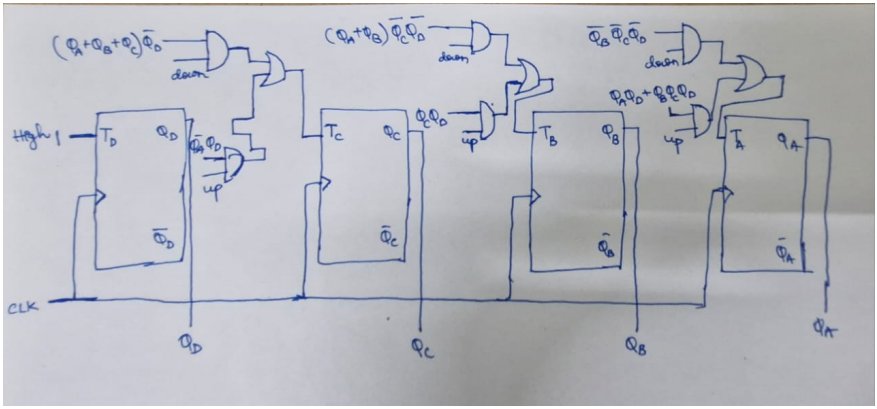
5 Connections

Follow the following connections

- **Step : 1**
  First Make a single digit up/down counter by cascading the up and down counter which counts from 0-9 and 9-0 when buttons are pressed accordingly

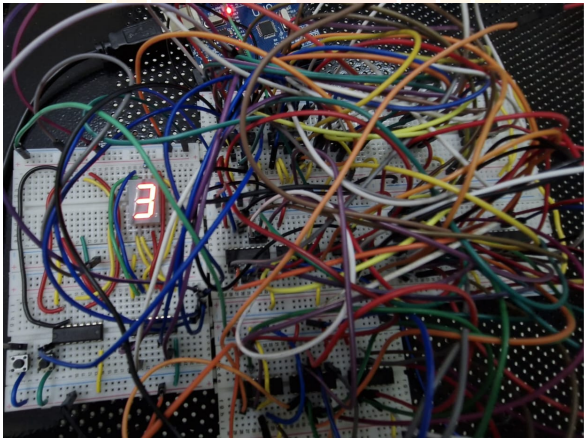
5.1 One's Digit Up Down Counter

Connect the following cascaded counter as in the diagram

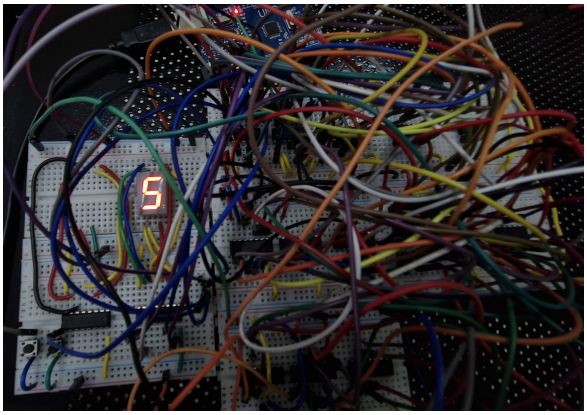


- Take 2 pins from the arduino to control up down buttons
- Connect each up donw pin as an input to an and gate annd the other input as its respective logic
- Give the outputs of each and gate to an or gate and then to each J and K
- Doing this we can control what logic to be performed
- Connect Q_A, Q_B, Q_C, Q_D to the respected 7 segment displays 7447 decoder inputs A B C D

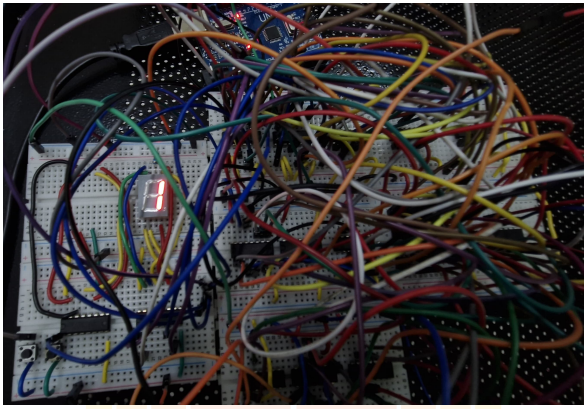
Below are some diagrams of the working of a one's digit up down counter, based on the above diagram



After pressing the up button twice



After pressing the down button four times



5.2 Ten's Digit Up Down Counter

- The connections logic remains the same as the one's digit we need to change the clock siganl inputs so that it works together as a two digit number
- When the one's digit is at 9 and after pressing up button the ten's digit should increment by 1
- When the one's digit is at 0 and after pressign the down button the ten's digit should decrement by 1
- We implement this logic by changing the ten;s digit clock input to the following
- Connect Q_A, Q_B, Q_C, Q_D to the respected 7 segment displays 7447 decoder inputs A B C D

if up logic is to be implemented

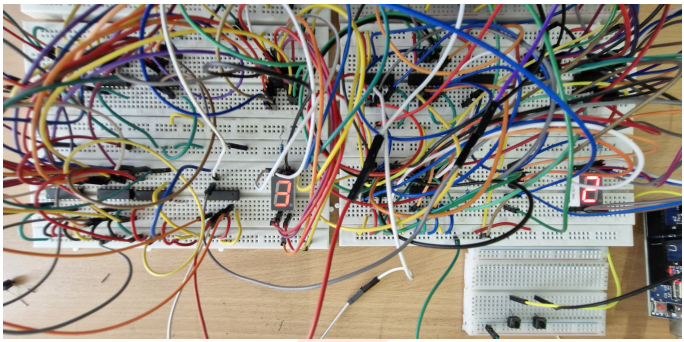
$$CLK = Q_3$$

else if down logic is to be implemented

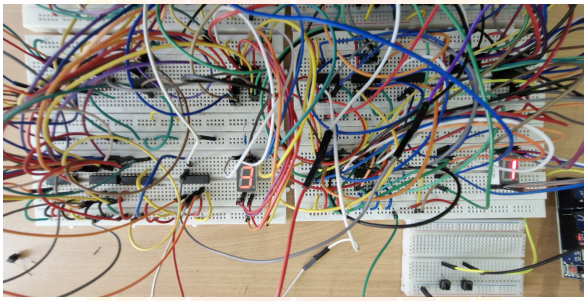
$$CLK = \overline{Q_3}$$

The followig logic can be achieved by using AND and OR gates

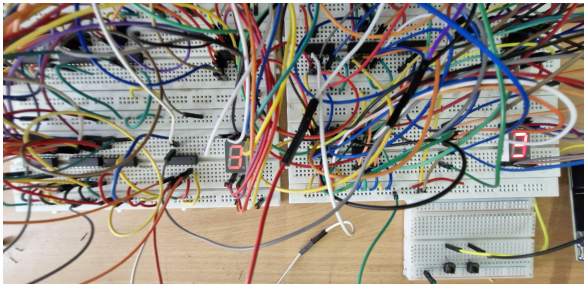
5.3 Working 0-99-0 Up Down Counter



After pressing down button once



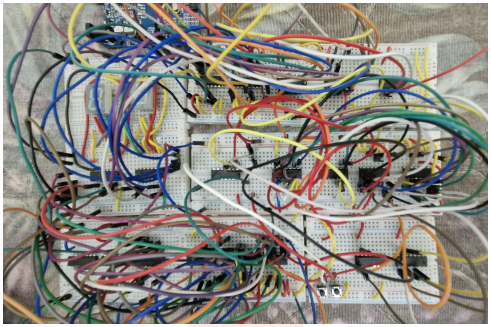
After pressing up button twice



5.4 Complete Circuit Diagram

- We make 2 independent 0-9 up counters and cascade down logic to that circuiy using and and or gates
- We used 2 buttons to for up and down
- Then we connect the clock signal of ten's digit given above logic so that it works together as a 0-99 up down counter

Below is the breadboard connections figure



6 CLK Signal Generation

To control the counting of the one's digit manually, a push button is connected to the Arduino. Each time the button is pressed, the Arduino generates a clock signal and sends it to the T flip-flops. This pulse acts like a clock tick, causing the T flip-flops to toggle and update the count.

- Since push buttons can sometimes cause multiple signals from a single press (known as *bouncing*), the program includes a short delay to ignore such effects. This ensures that one press results in only one proper clock pulse.

Here is the arduino code,

```
const int clockPin = 8;
const int pulseButton = 6;
const int toggleButton = 7;
const int pin12 = 12;
const int pin13 = 13;

bool toggleState = true;

void setup() {
  pinMode(clockPin, OUTPUT);
  pinMode(pulseButton, INPUT_PULLUP);
  pinMode(toggleButton, INPUT_PULLUP);
  pinMode(pin12, OUTPUT);
  pinMode(pin13, OUTPUT);

  digitalWrite(pin12, toggleState);
```

```
  digitalWrite(pin13, !toggleState);
}

void loop() {
  // Handle toggle button (pin 8)
  if (digitalRead(toggleButton) == LOW) {
    delay(50); // Debounce
    if (digitalRead(toggleButton) == LOW) {
      toggleState = !toggleState;
      digitalWrite(pin12, toggleState);
      digitalWrite(pin13, !toggleState);
      while (digitalRead(toggleButton) == LOW); // Wait for release
      delay(50); // Debounce
    }
  }

  // Handle clock pulse button (pin 7)
  if (digitalRead(pulseButton) == LOW) {
    delay(50); // Debounce
    if (digitalRead(pulseButton) == LOW) {
      digitalWrite(clockPin, HIGH);
      delay(100); // Clock pulse duration
      digitalWrite(clockPin, LOW);
      while (digitalRead(pulseButton) == LOW); // Wait for release
      delay(50); // Debounce
    }
  }
}
```

The above code functions as follows

- Initializes `clockPin (9)` as output for generating clock pulses.
- Sets `pulseButton (6)` and `toggleButton (7)` as input with internal pull-up resistors.
- `pin12 (12)` and `pin13 (13)` are outputs to show the state of `toggleState`.
- `toggleState` starts as `true`, so `pin12 = HIGH`, `pin13 = LOW`.
- When `toggleButton` (pin 7) is pressed:
  - Debounces with delay.
  - If still pressed, toggles `toggleState`.
  - Updates `pin12` and `pin13` accordingly: one HIGH, the other LOW.
  - Waits until button is released before continuing.
- When `pulseButton` (pin 6) is pressed:
  - Debounces with delay.
  - If still pressed, generates a HIGH pulse on `clockPin` for 100 ms.
  - Then sets `clockPin` LOW.
  - Waits until button is released before continuing.
- Used for manual control of clock signal and toggling a binary output state.
- Debouncing ensures one response per button press.
- Useful in digital logic testing or manual step-through applications.

7 Problems Faced - Hardware

- Some IC's were not working properly, which was lately realized
- Forgot to give VCC and GND to the gates sometimes
- Breadboard, wiring issues. Faulty Breadboards and jumper cables
- Push Button debouncing issue