PROOF OF WORK

What was our GOAL for this MODULE?

You have acquired a deeper and practical understanding of validation concepts in the blockchain framework. These concepts are critical for the security of the blockchain information and thus kids learn how the miners race to get the blocks validated.

What did we ACHIEVE in the class TODAY?

- Used the sandbox for implementation and for understanding the nonce concept and performed the small code to demo the nonce concept.

Which CONCEPTS/ CODING did we cover today?

- Learn the concept of Proof-of-work and Nonce.
- Demo of nonce using the sandbox
- Demo of code on nonce

How did we DO the activities?

Python code to generate a nonce and understand the concept of generating nonce:

```
for nonce in range(1, 20):
    equation =20+4-nonce
    if equation == 18:
            print("After calculations, we get the result which is number 18, at a number", nonce ,"hence the nonce is: ", nonce)
            break
    else:
            print("At number", nonce , " we didn't get the result as number 18")
```

Explaining above code:

1) **Wrote for loop to iterate a given range:**
    - Iteration means repetition of a set of statements or code, for a given range of sequence.

    - In the above code, our aim was to get the nonce, which is a number, which will satisfy the mathematical problem.
    - For this we started by writing the for loop.
    - We used the for loop only, because, to solve the mathematical problem,we have to get a missing value. So we have to specify a range of number from which we can obtain that value.So, in order to get all the values of that range, we have to set an iteration, for this purpose we have to use the for loop.
    - Like this
        - We wrote the for keyword `for`
        - Then to keep a record of iterateration of the for loop, we set a variable called **nonce.** `for nonce`
        - Then we wrote an in operator, which is used to check the iteration value in a specified range. Like this `for nonce in`
        - Then we specified a range in which we wanted the for loop to iterate. Means how many times, or till which value we have to iterate the for loop.
        - So, in our python module, while learning the for loop, we had learnt the **range()** function concept, which takes two values; start and stop. Where start indicates the value from which we have to start iterating the loop. And the stop indicates till which value we have to stop iterating the loop.
        - So, we specified the **range()** function like this `for nonce in range()`

- And inside this range() function we added the start value as 1 and stop value as 20. This is because, we had to get a missing value of the equation which resulted us to achieve the target value as 18
- Like this `for nonce in range(1, 20):`
- And after writing the colon, we started adding the below code given in the next point.

- Then , we wrote an equation. This equation, we wrote to generate a specific number, using a nonce value.
- For this we wrote a variable name as an **equation,** and inside it, we stored an equation as 20+4-**nonce.** Here we generated a value of equation, based on the nonce value.Like this:

```
equation =20+4-nonce
```

- Then, we added a condition, to check, if the value of an equation is equal to a a target we set. Let's suppose, we set a target as 18. Now we want to find that at which iteration we will get the target 18.
-
- So, we wrote the if condition as follows:
  - We wrote the if keyword. `if`
  - Then we wrote a condition to check if value of equation is equal to number 18 like this: `if equation == 18:`
  - Then if we get the value of the condition as 18, which means that the condition is true, then we printed a statement or text as **"After calculations, we got the result which is number 18, at a number"**, followed by variable **nonce**, then followed by the text as again text **"hence the nonce is: ",** then again followed by the variable as **nonce.** Like this:

```
if equation == 18:
        print("After calculations, we get the result which is number 18, at a number", nonce ,"hence the nonce is: ", nonce)
```

  - This gave us the nonce value which solved the mathematical problem or the equation in this case.
  - Then we broke the loop using break keyword. Which meant it should exit the if condition and come out of the loop.The break keyword, helps us to come out of the loop, once we get the condition as true. Like this

```
for nonce in range(1, 20):
    equation =20+4-nonce
    if equation == 18:
            print("After calculations, we get the result which is number 18, at a number", nonce ,"hence the nonce is: ", nonce)
            break
```

  - And if we do not get the result as 18, after testing all the numbers from 1 to 20 from the equation, then it will go to the else part, and print the

text as **"At number",** followed by a variable as **nonce,** then again followed by the text as **“we didn't get the result as number 18” .** Like this:

```
else:
        print("At number", nonce , " we didn't get the result as number 18")
```

```
for nonce in range(1, 20):
    equation =20+4-nonce
    if equation == 18:
        print("After calculations, we get the result which is number 18, at a number", nonce ,"hence the nonce is: ", nonce)
        break
    else:
        print("At number", nonce , " we didn't get the result as number 18")
```

Output:

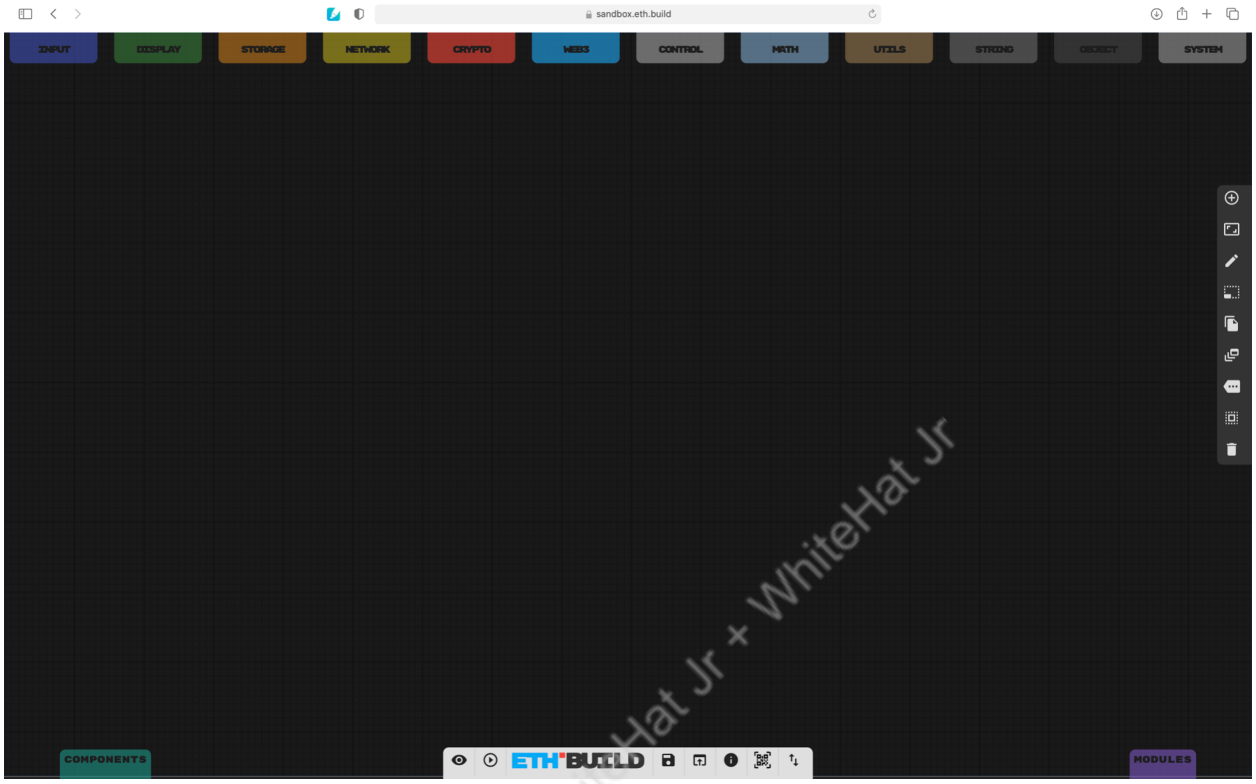
```
(base) C:\Users\ASUS\Documents\BLOCKCHAIN>class_C247.py
At number 1  we didn't get the result as number 18
At number 2  we didn't get the result as number 18
At number 3  we didn't get the result as number 18
At number 4  we didn't get the result as number 18
At number 5  we didn't get the result as number 18
After calculations, we get the result which is number 18, at a number 6 hence the nonce is:  6
```

So the mathematical problem was solved using number 6 and hence our nonce is number 6. Which is nothing but our proof of work.

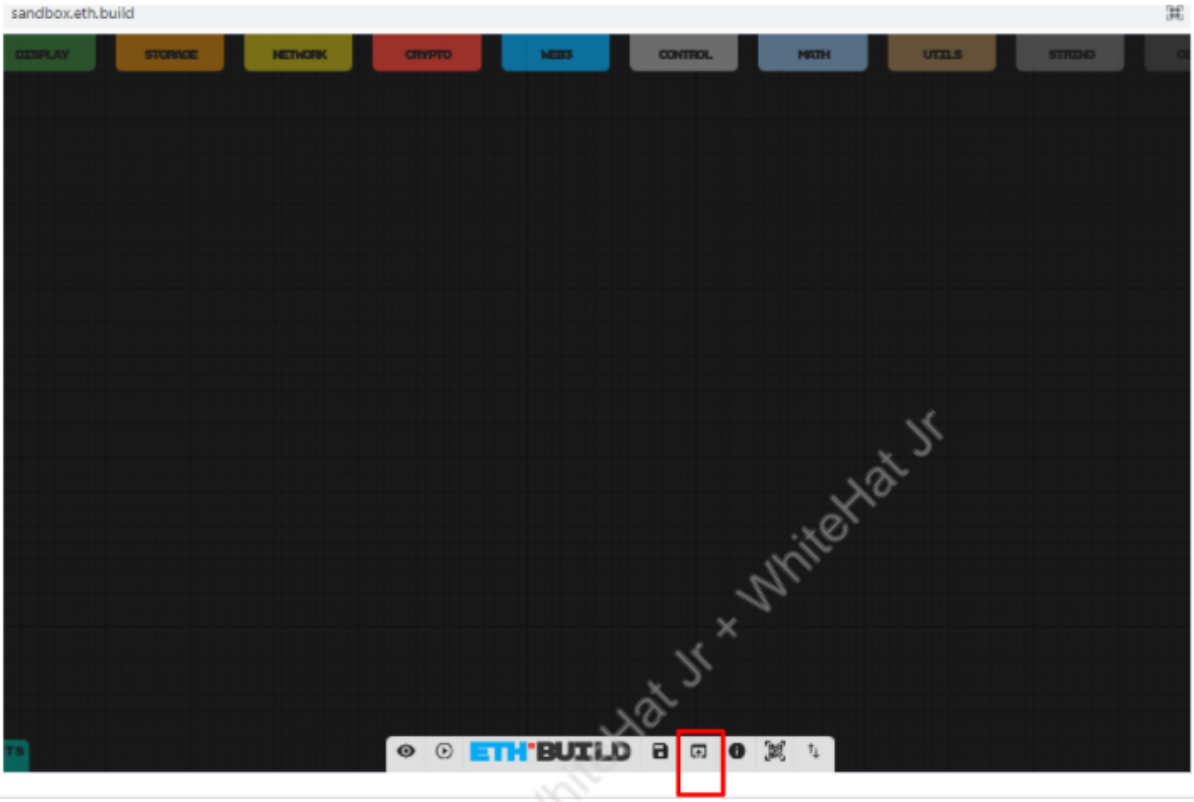
Sandbox Implementation -

We implemented nonce into Sandbox to see how blocks are validated in block coding.

1. **Implemented nonce into sandbox**
    - Once you open the sandbox you can see something like this



- Then we clicked on **load** to upload the file we downloaded Nonce_demo.webloc , like this



- We can see something like this, it contains a predefined network block, which holds the transactions data.

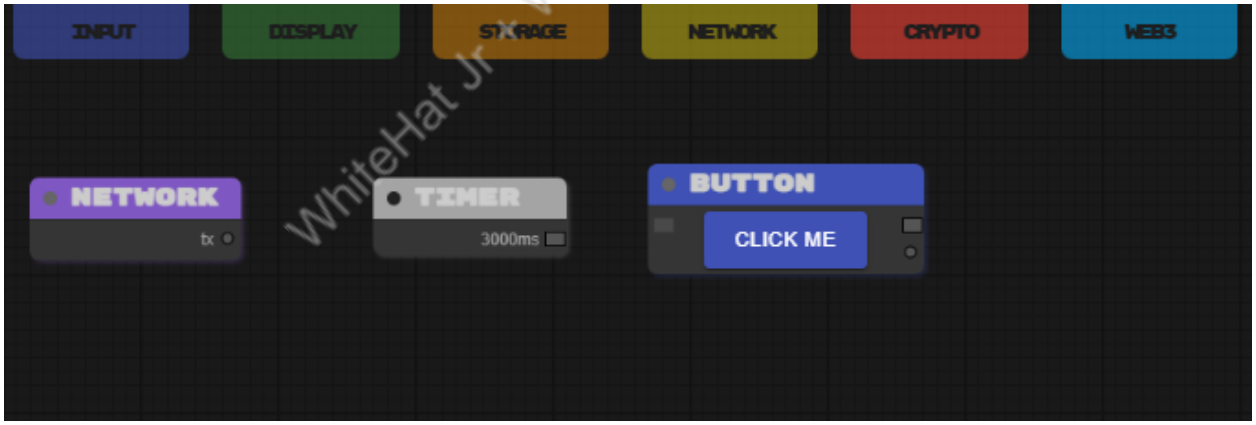
This network block consists of different transactions, all over the network, which are yet to be verified in the blockchain. So, this block sends the transactions continuously to the block chain for verifying purposes, and here we used a Ledger block, which behaves like a database or transactions.

- Then we pressed the spacebar and then searched for Timer and pressed enter. Like this

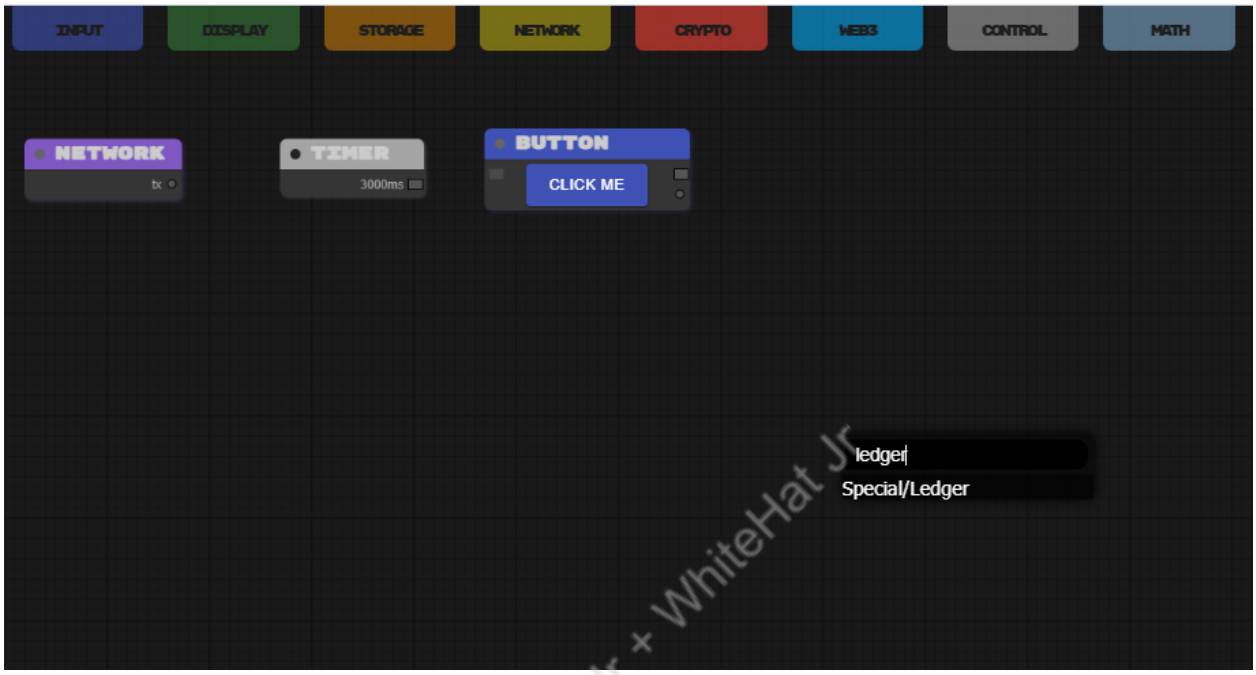
- A timer block is added like this

- Then, again we pressed spacebar and searched for Button like this:

- A Button block gets added like this:

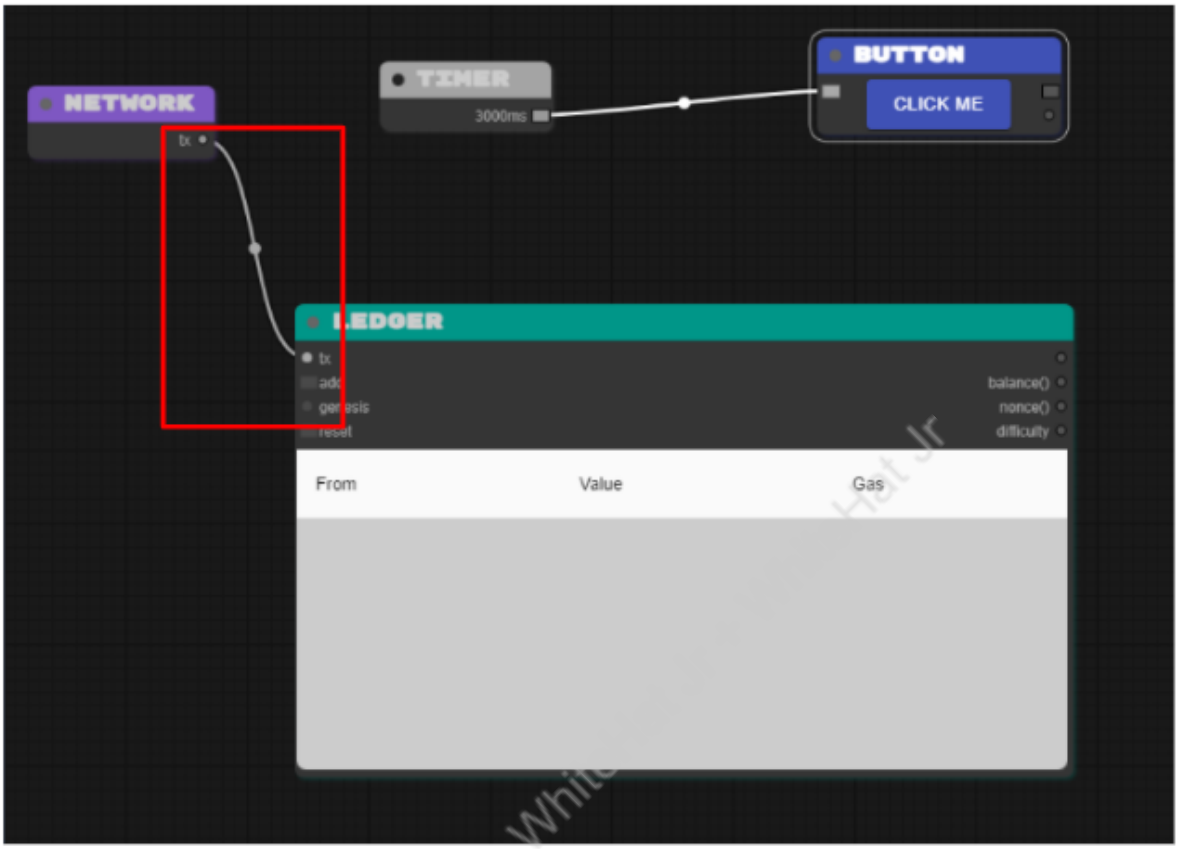


- Then, again we pressed spacebar and search for Special/Ledger like this

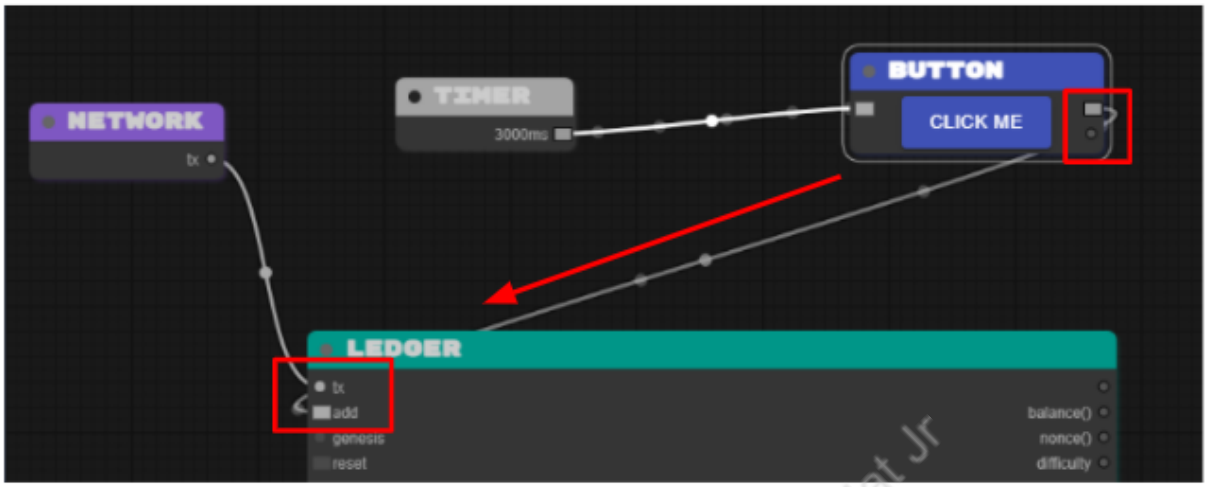


- A Ledger block gets added like this:

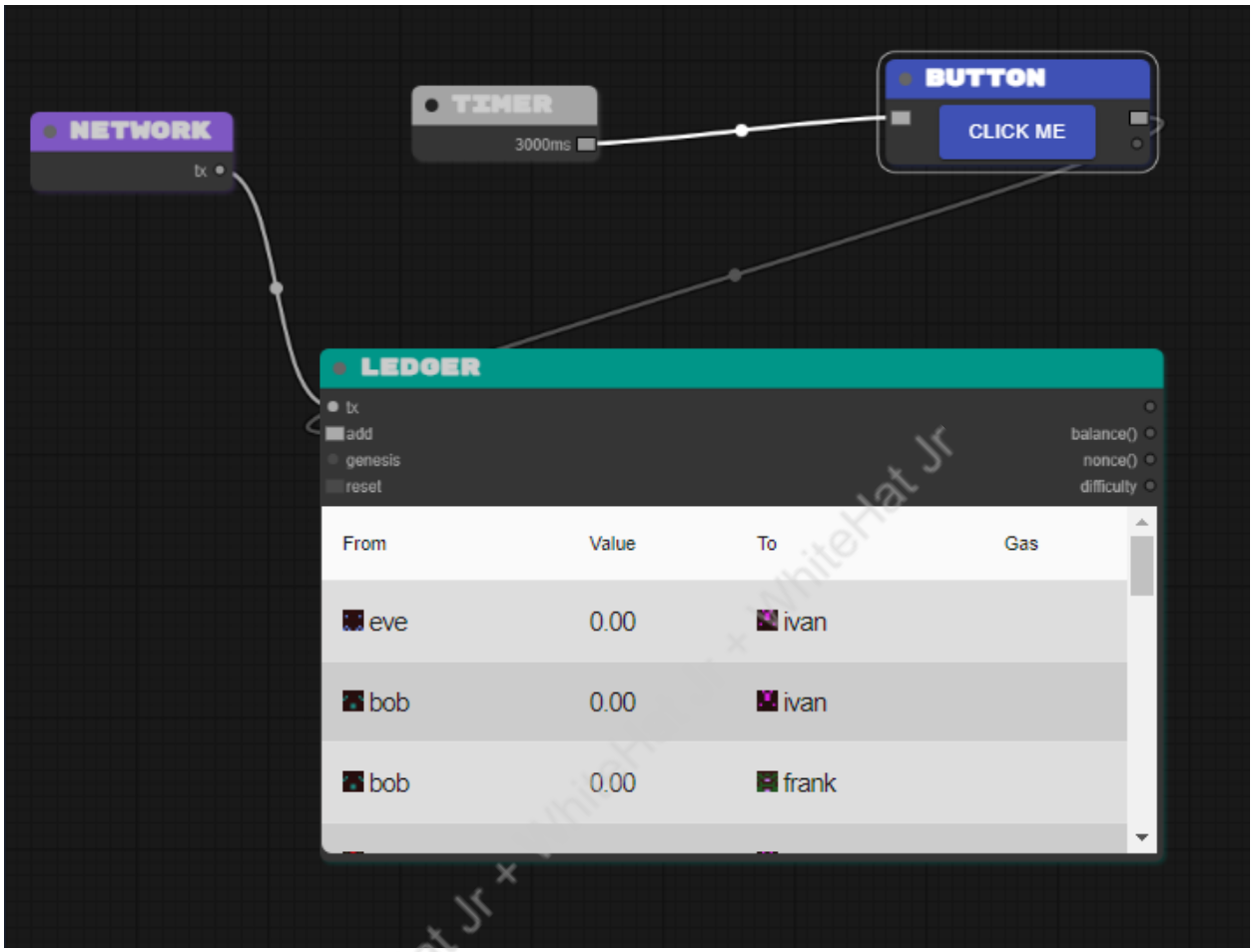
- Ledger is a block which contains the continuous transaction data, which needs to be validated in the block.
- Then, we had to get the transactions from the network block, which are continuously getting generated. And which needs to be verified in the block chain.
- then,we connected all the components:
    - First connect the network component with the ledger, by connecting the tx of the network to the tx of the network.
      And connect the timer component to the button like this:



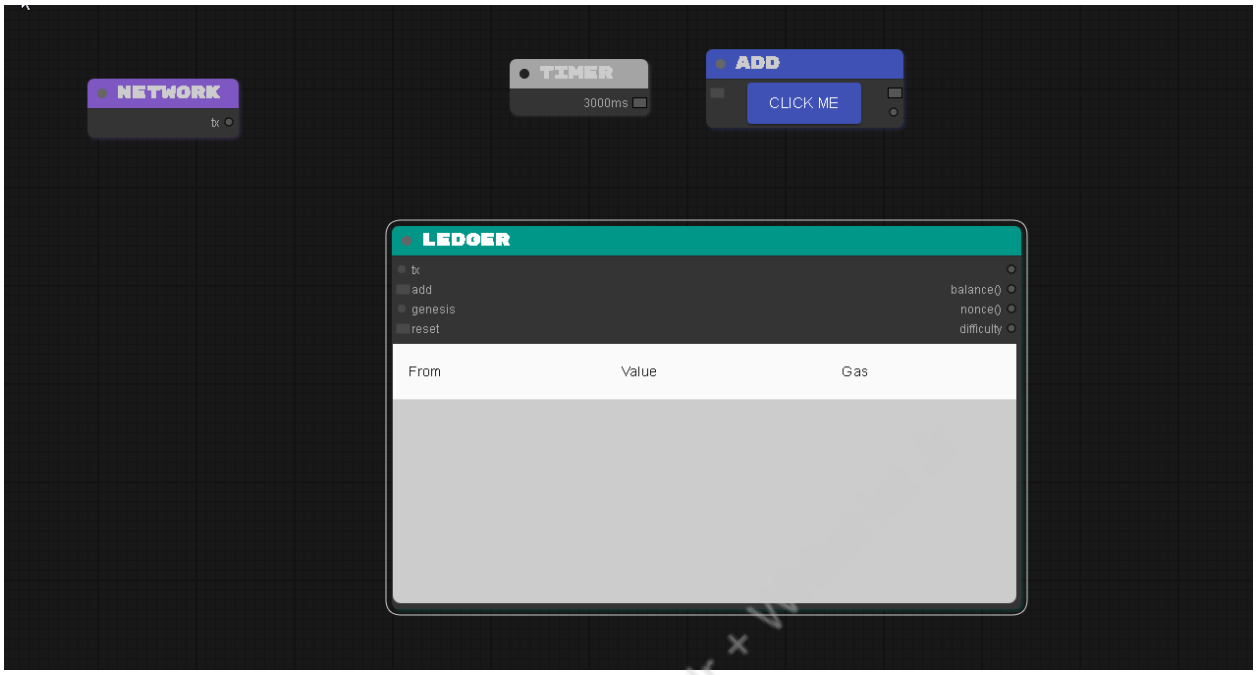
Here we connected the network block with the ledger to get access to all the transactions present in the network. Then using the timer which is set for 3000 ms, we got the transactions in the ledger block. Here the timer after every 3000 ms automatically presses the button and the transaction gets added to the ledger from the network.



- So, once we connected the button to the ledger,the button automatically functioned after every 3000 ms and displayed the transactions in the Ledger. like this:

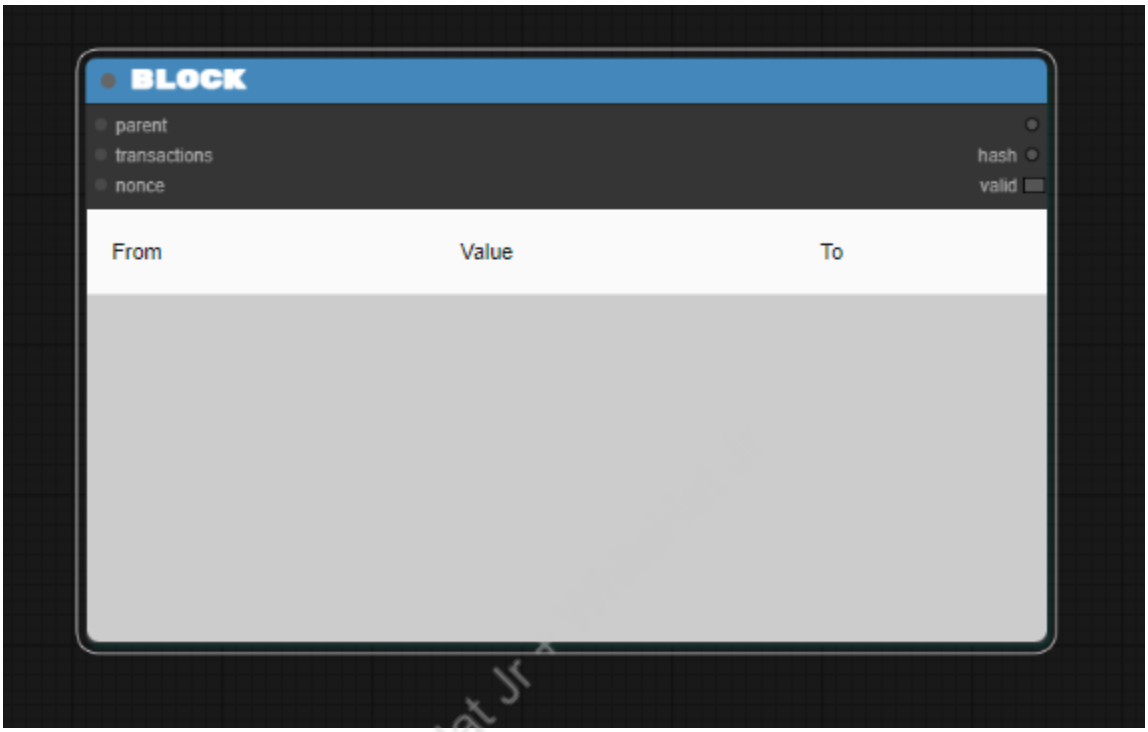


Below is the gif image of how, dragging and connecting is done in the Sandbox, we follow, the same way, while connecting with other blocks.:



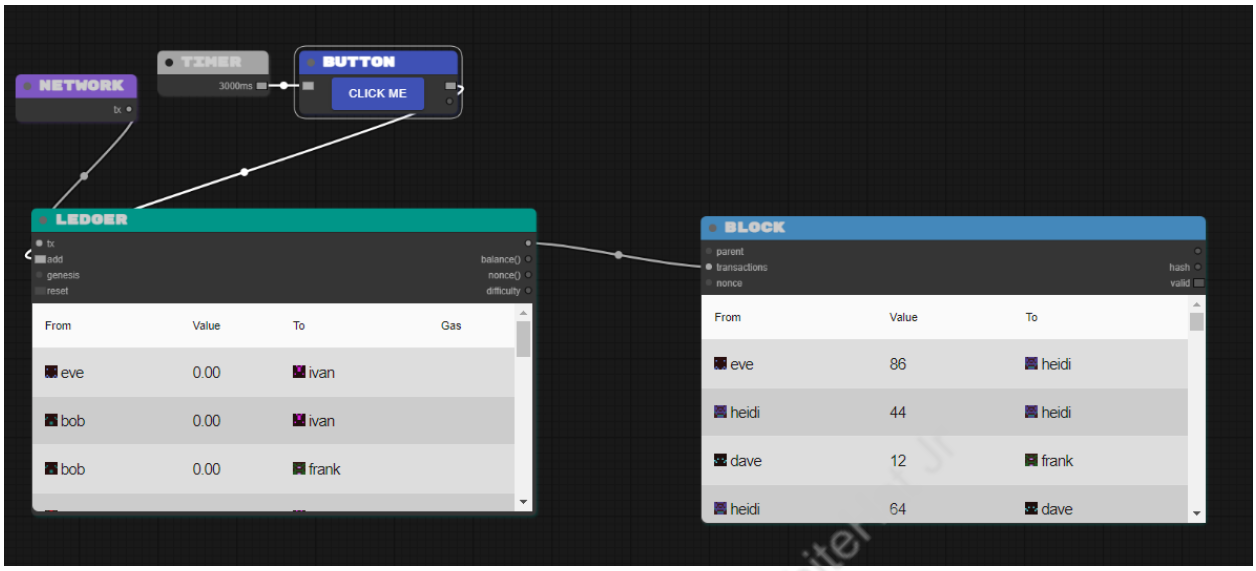
- Then we created a new block, by giving a space bar, and an input box appeared. Then,we wrote there as special block and press enter:

- It displayed the following like this:

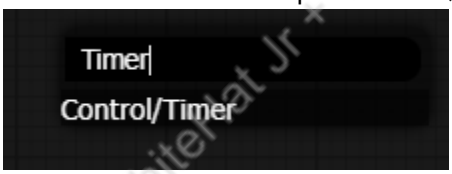


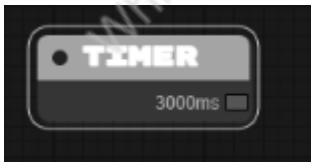
- Then , we connected the ledger and the new block like this:

- When we connected the ledger and the block, then it showed all the transactions from ledger into the block like this:

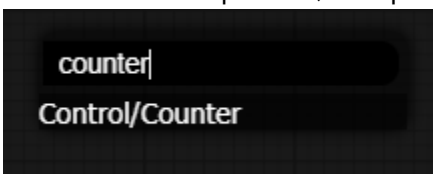


- Then , we validated this block, for this we again took a timer block, and a counter block. Here the counter block acted as our nonce value, and this value gets changed for every 3000ms till the block gets validated. We can consider this timer as the for loop, which keeps on iterating until we get the nonce value, similarly, the timer value keeps on changing every 3000 ms till we got the nonce value, and the block gets verified.
- Let's first take Timer. So,we clicked on space bar, and in the input box, we wrote Timer on it and press Enter, like this:

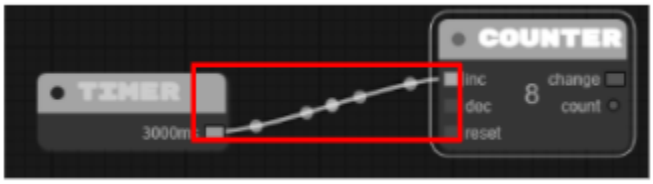




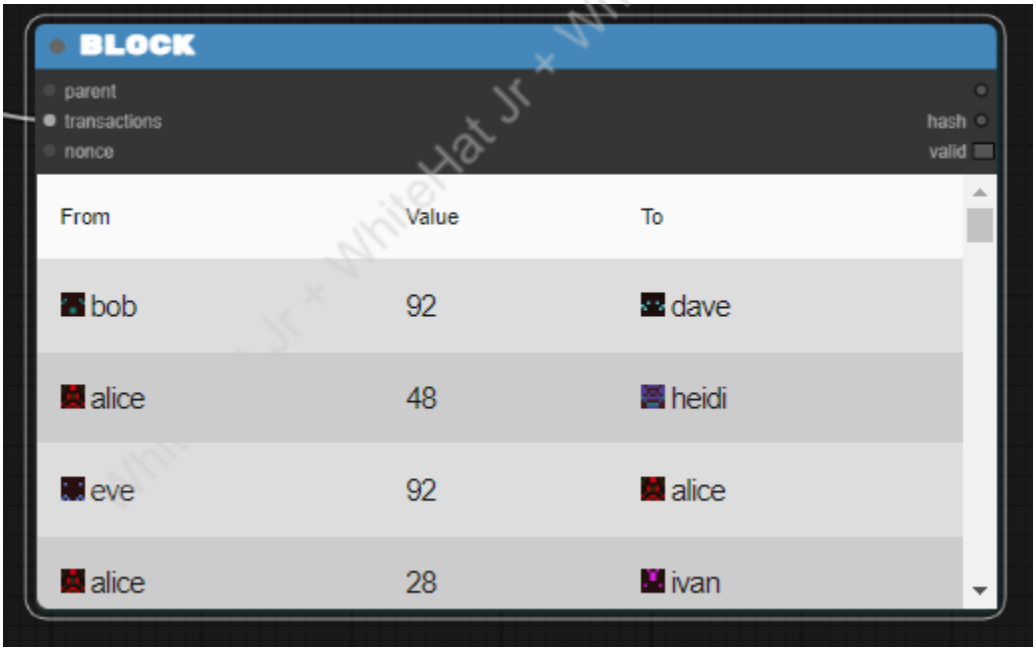
- Then we added counter block. So again we clicked on spacebar and wrote Counter in the inputbox, and pressed enter, like this:



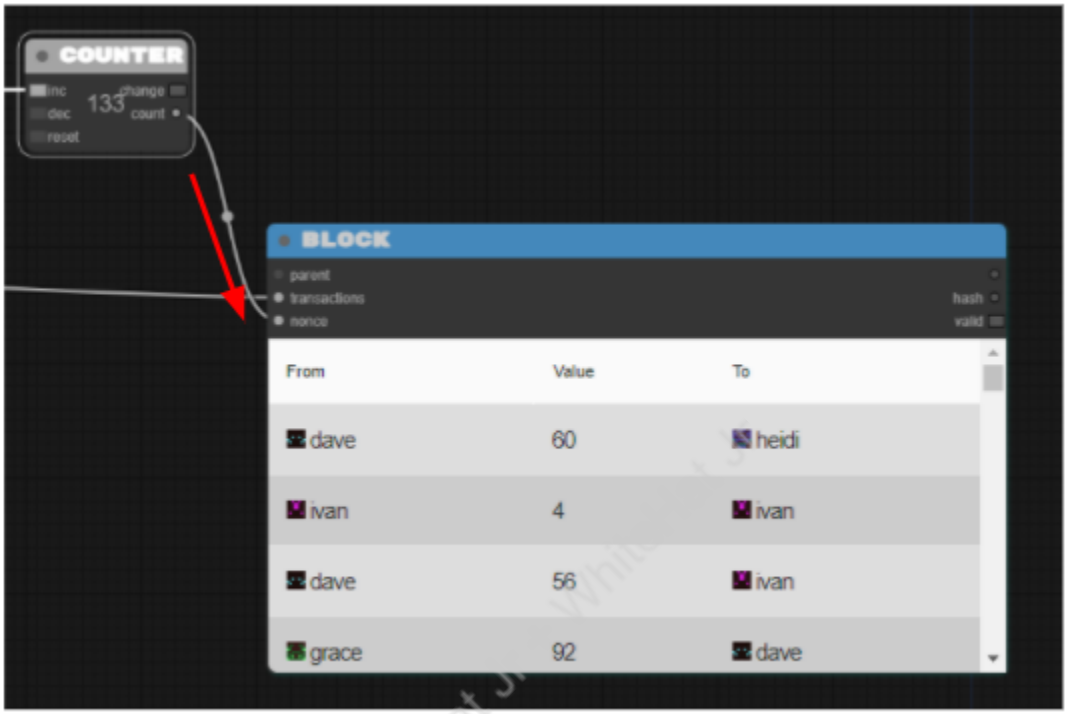
- Then, lets connect the Timer block and the Counter block. Like this:



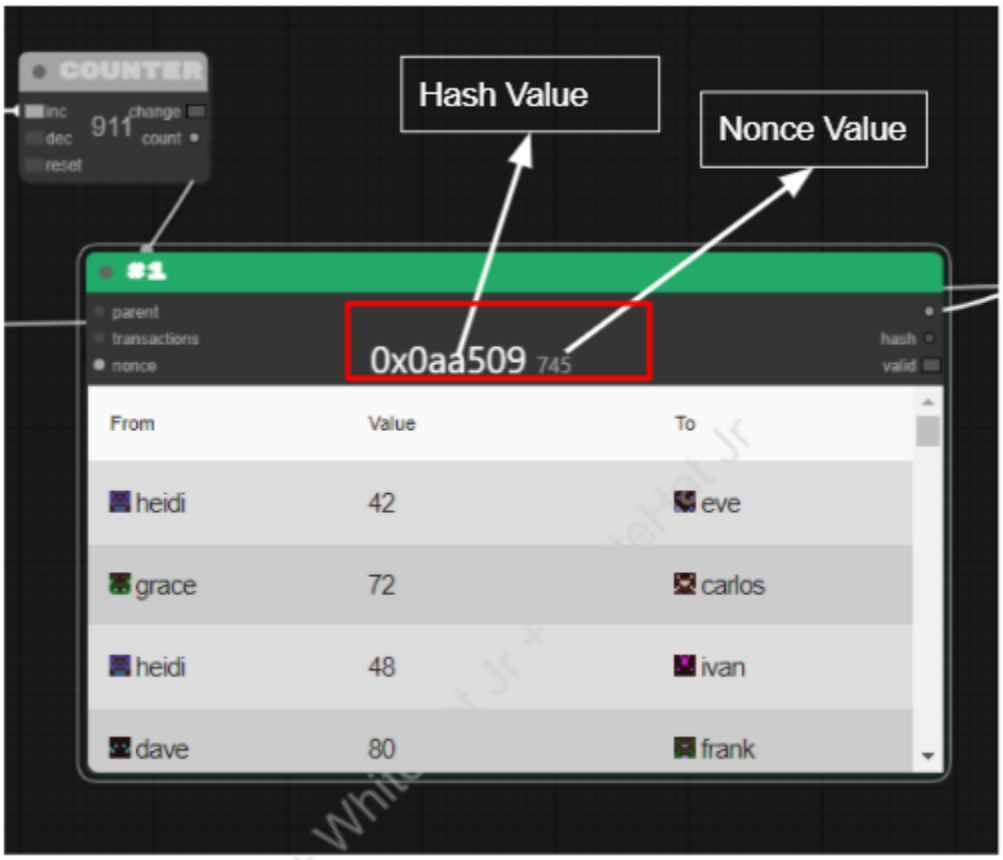
- Then we validated our block. For this, we connected the counter to the block. And if you observe the block, we cant see the block hash present on the block like this:



- So, then,we connected the counter to the block. For this from the count field of the counter block,we dragged a line towards the nonce field of the block. Like this



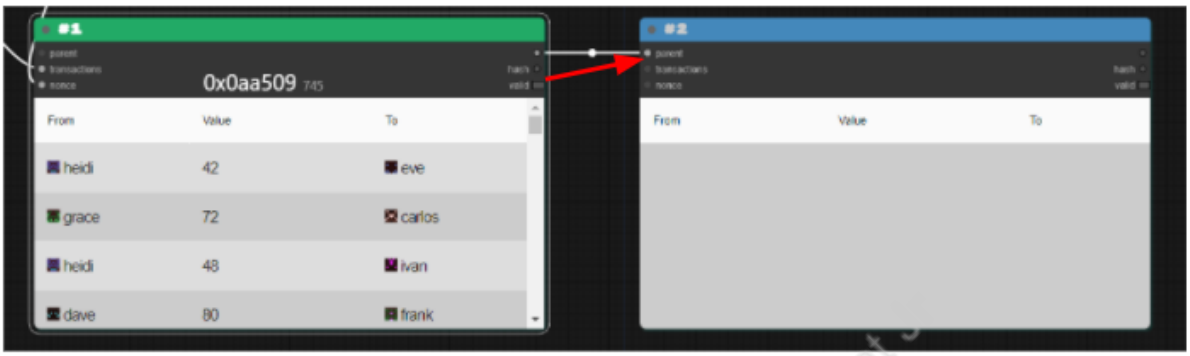
- If we observed the block, we see that the color of the block is changed from blue to green. And also there are two values being displayed on the block, which are the hash of the block and the nonce value. Like this:



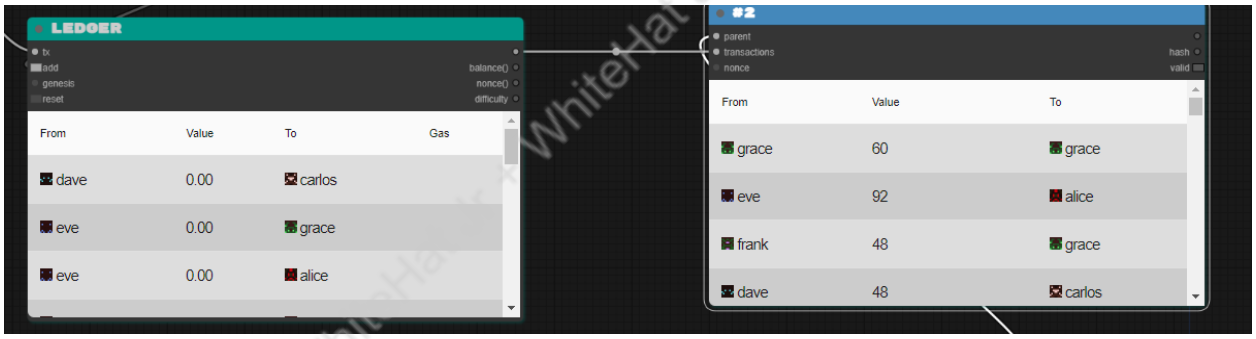
- Once the color of the block changes to green and a hash is generated. It means that the block gets verified and added to the network. And this happened as the proof of work which is that the nonce generates the hash value to the block.So the hash value displayed on the block is achieved at a nonce value 745.

- Then,we took a new block. So, again type spacebar, and type Special/Block in the input field, and press enter. like this

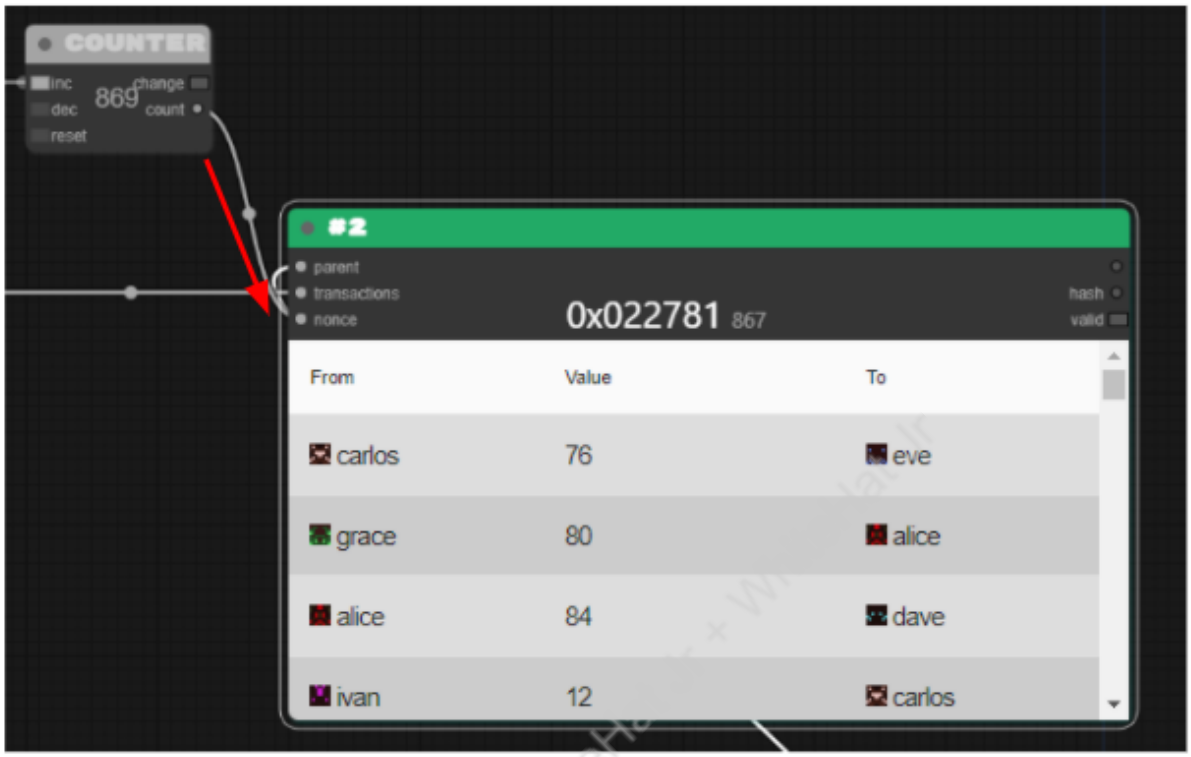
- Then,we connected the first block to the parent field of the second block. This we are doing to verify the new transactions which we are getting added in the network. Like this :



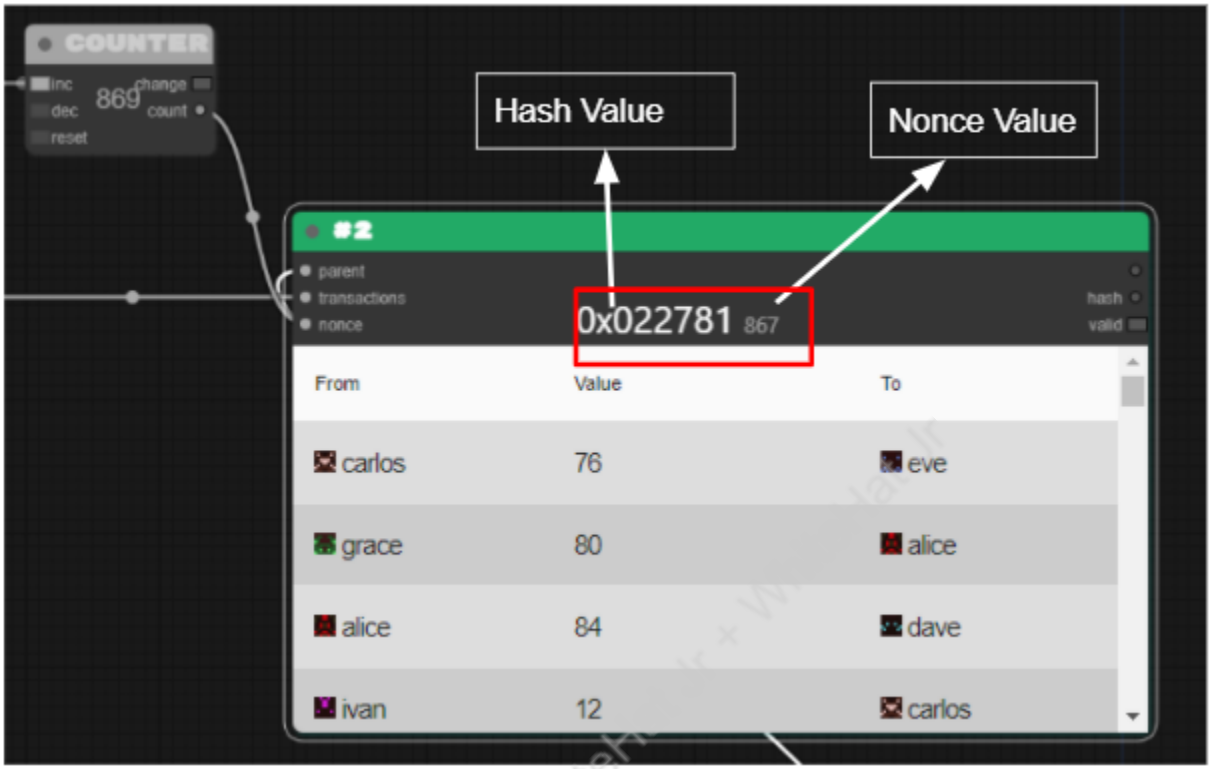
- Then we removed the connection of the ledger and the first block, and connected the Ledger and the new or the second block to the ledger.
- This is because, then we had to establish a connection between the counter and the new block,so as to show the flow of transactions. As shown below:



- Then,we removed the connection between the counter block and the first block and connected the second block and the counter block.
- For this from the count field of the counter block we dragged a line and connected it to the nonce field of the second block.



- As soon as we connected, the block took some time to verify and we see, that the block gets validated and generated a hash at a given nonce value like this



- Hence the block got verified and generated a hash at the nonce value of

What's NEXT?

In the next class, we will learn how to implement proof of work in the code, using the real time transactions.