Lab1

Vide Hopper, Humam Amouri & Martin Karlsson

Task 1: Breadboard Setup and a NAND gate

We started with gathering the following necessary parts needed under the lab:

- *A solderless breadboard.*
- *An Adafruit Trinket microcontroller board.*
- *Wiring kit.*
- *Micro USB cable.*
- *LED set.*
- *3x 7400-series (SN74ACT00) quad NAND gate arrays.*
- *2x 7402-series (SN74HCT02) quad NOR gate arrays.*

Then we proceeded following the instructions provided in task 1.

First we connected power to the Arduino as well as the 5V output and the ground output to the breadboard. We confirmed the current by connecting a LED to the bredboards HIGH and LOW. After that we connected pin 3 and pin 4 as outputs from the Arduino.

The next step was to connect a NAND gate using the SN74ACT00 chip to the breadboard.

We connected a red LED to the output of the gate and two yellow LEDs to the outputs of the Arduino and loaded *Code snippet 1* to the microcontroller.

```
1.  void setup() {
2.   // initialize the digital pin as an output.
3.   pinMode(4, OUTPUT);
4.   pinMode(3, OUTPUT);
5.  }
6.
7.  // the loop routine runs over and over again forever:
8.  void loop() {
9.     digitalWrite(4, LOW);
10.    digitalWrite(3, LOW);
11.    delay(1000);
12.
13.    digitalWrite(4, HIGH);
14.    digitalWrite(3, LOW);
15.    delay(1000);
16.
17.    digitalWrite(4, LOW);
18.    digitalWrite(3, HIGH);
19.    delay(1000)
20.
21.    digitalWrite(4, HIGH);
22.    digitalWrite(3, HIGH);
23.    delay(1000)
24. }
```

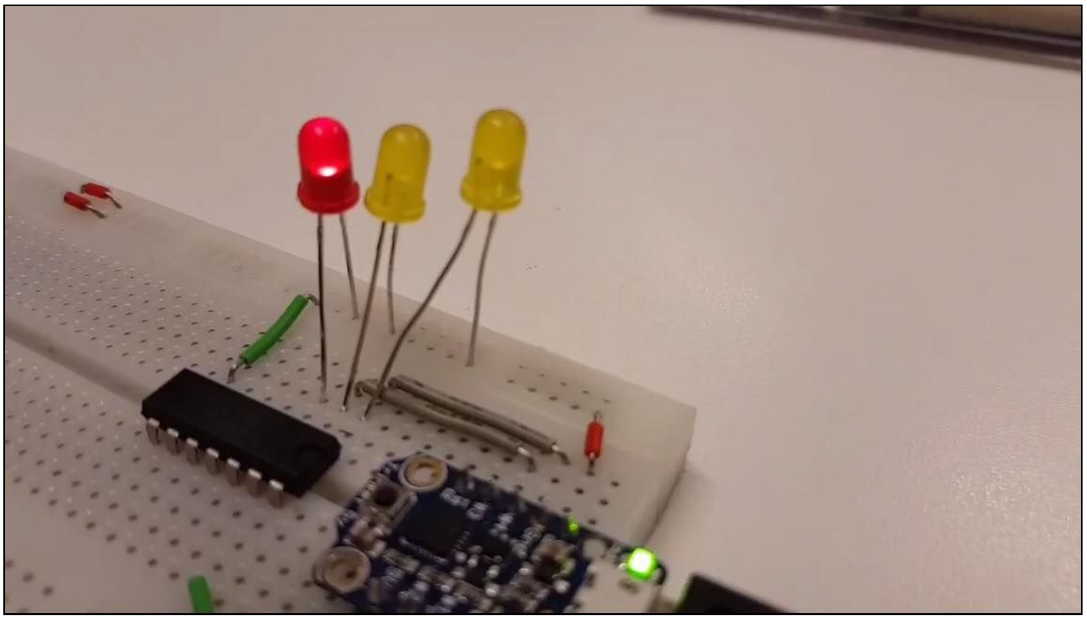
Code snippet 1, the Arduino code that was used during the lab

The code in *Code snippet 1* firstly initialize pin 3 and pin 4 as output ports. This makes it possible to send 5V (henceforth referred as a 1) through the pins.

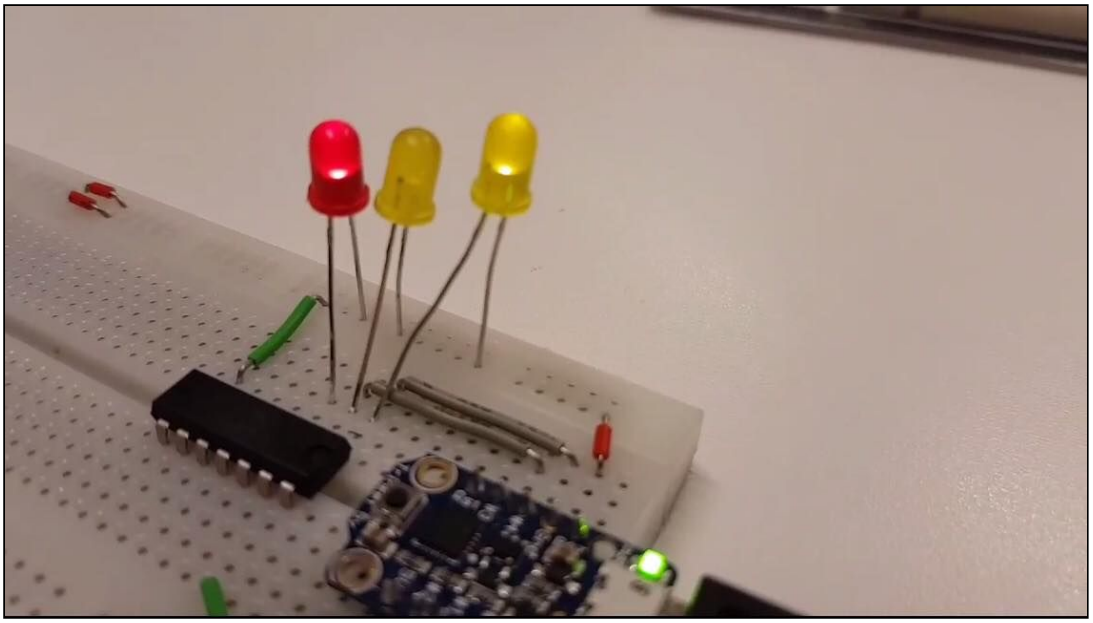
In the loop section we run through all different permutations as seen on *Table 1*. We use the delay function to stay on each output for 1000 milliseconds. This means that within 4 seconds, we've looped through the loop function.

Input	Input	Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

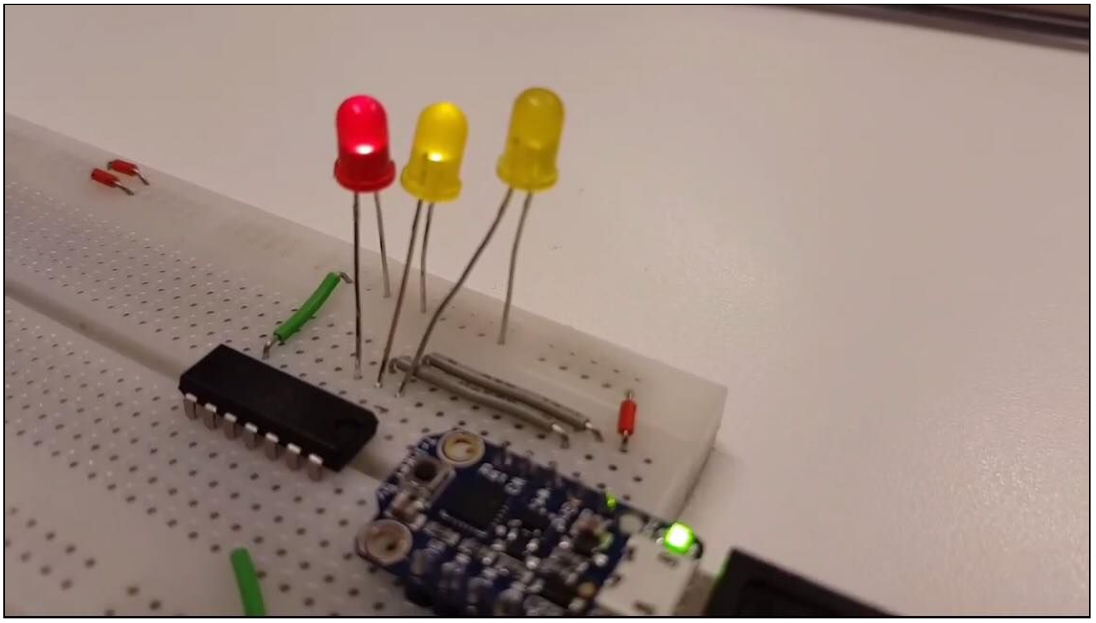
Table 1, A = OUTPUT 4, B=OUTPUT 3



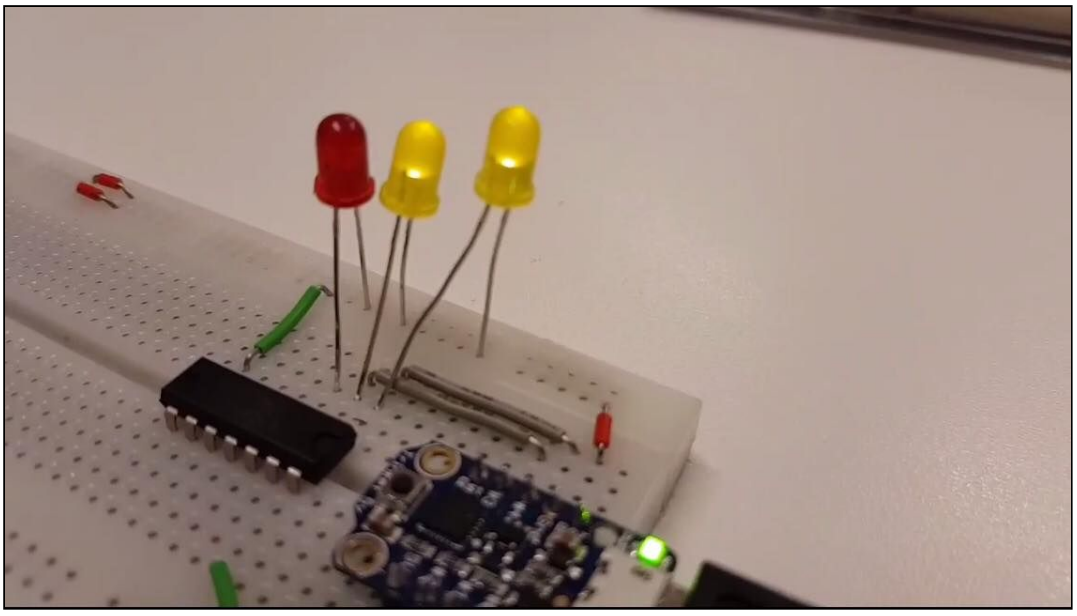
Picture 1, NAND (A = B = 0)



Picture 2, NAND (A =1 B = 0)



Picture 3, NAND (A =0 B = 1)

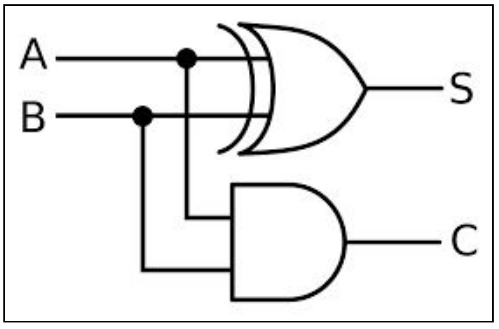


Picture 4, NAND (A =1 B = 1)

Our LEDs lights up as we loop through our code and the red LED lights up everytime the output of the NAND gate is 1. Pictures 1-4 proves that our circuit follow the truth table correctly.

Task 2: Half-adder

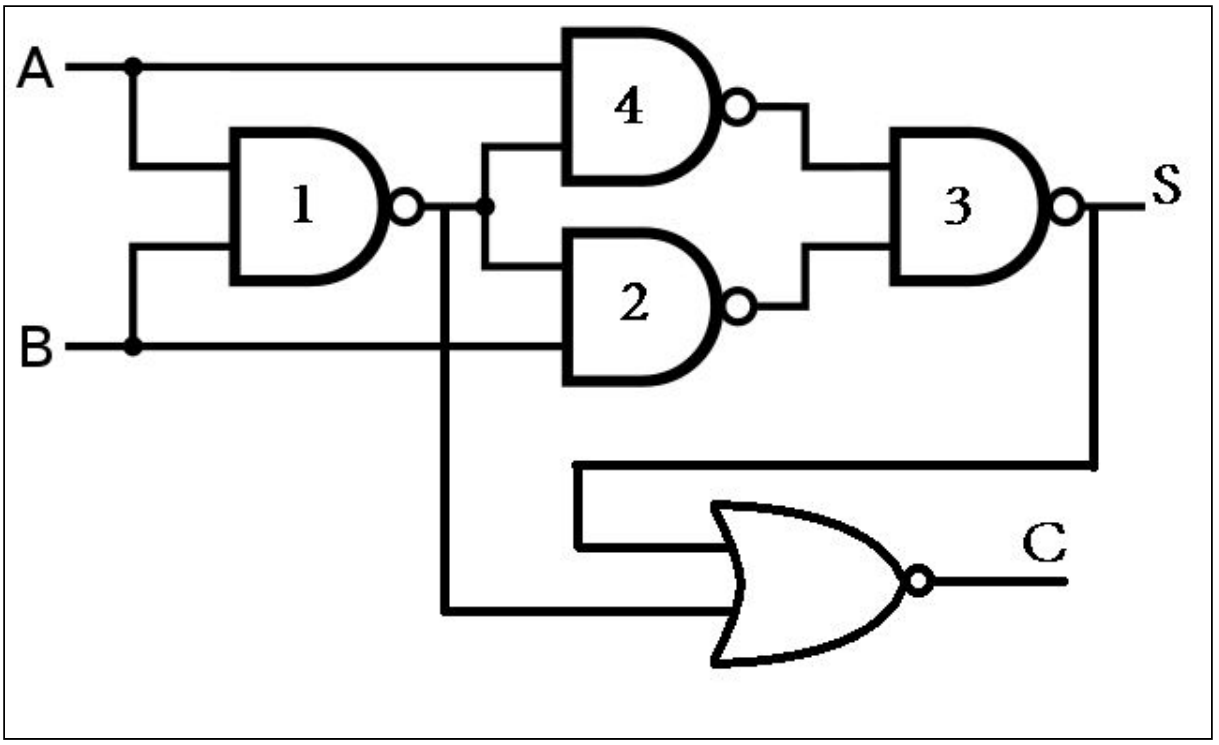
This task was designed to implement a Half-adder circuit. But instead of using one XOR and one AND gate as described in *Picture 5* we were given NAND and NOR.



Picture 5, Half adder

We came up with the solution seen on *Picture 6* and we can see that the result is equivalent to *Picture 5*. Our solution contains four NAND gates and one NOR gate. The four NAND gates operates as the XOR which calculates the sum while the NOR gate works as the carry. Each NAND gate was labeled from the chipsets manual page to keep track of the gates input and output. All 4 NAND gates on the chipset were in use and a NOR chipset was connected

where we utilized one of the NOR gates according to *Picture 6*. The circuit can be viewed in *Picture 8, 10, 12*.



Picture 6, Half adder

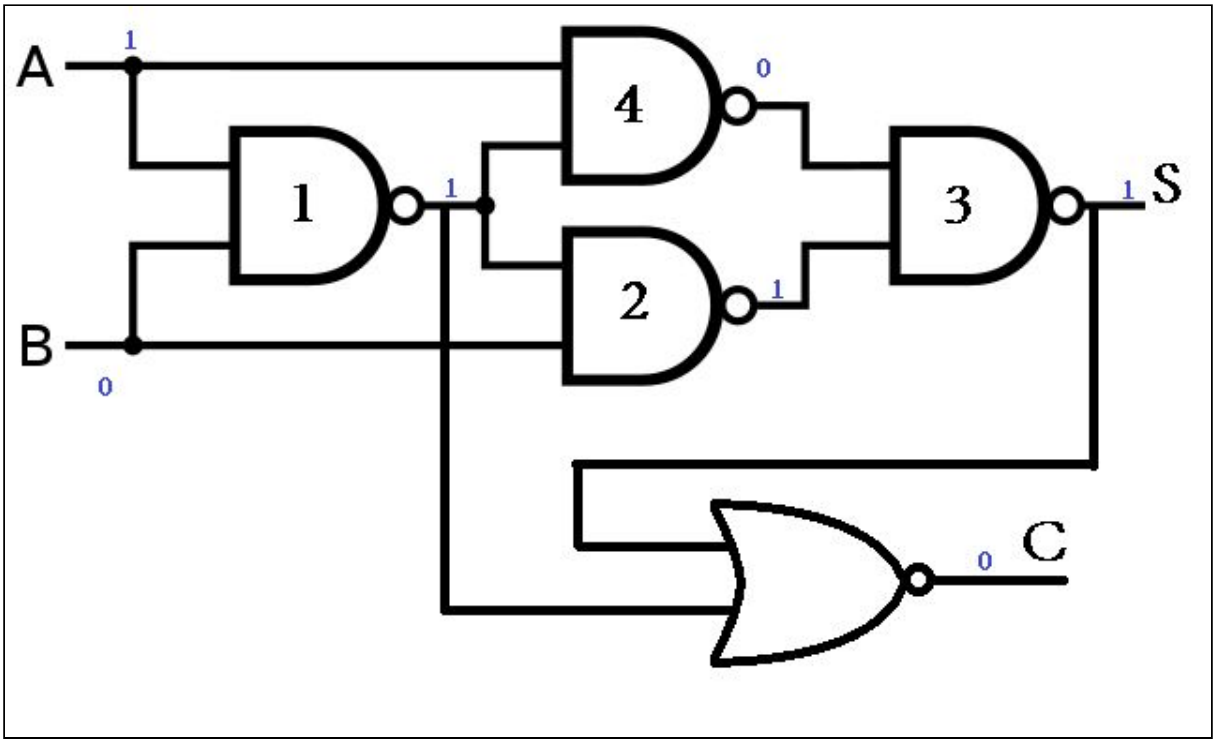
In our circuit we connected one red LED for the sum output and one yellow LED for the carry output. We used the same code as before, *Code snippet 1*, that cycled through the different possible inputs. To demonstrate that our solution works as a Half-adder we take a closer look at the three different outcomes represented in *Table 2* below.

Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 2, Half adder truth table

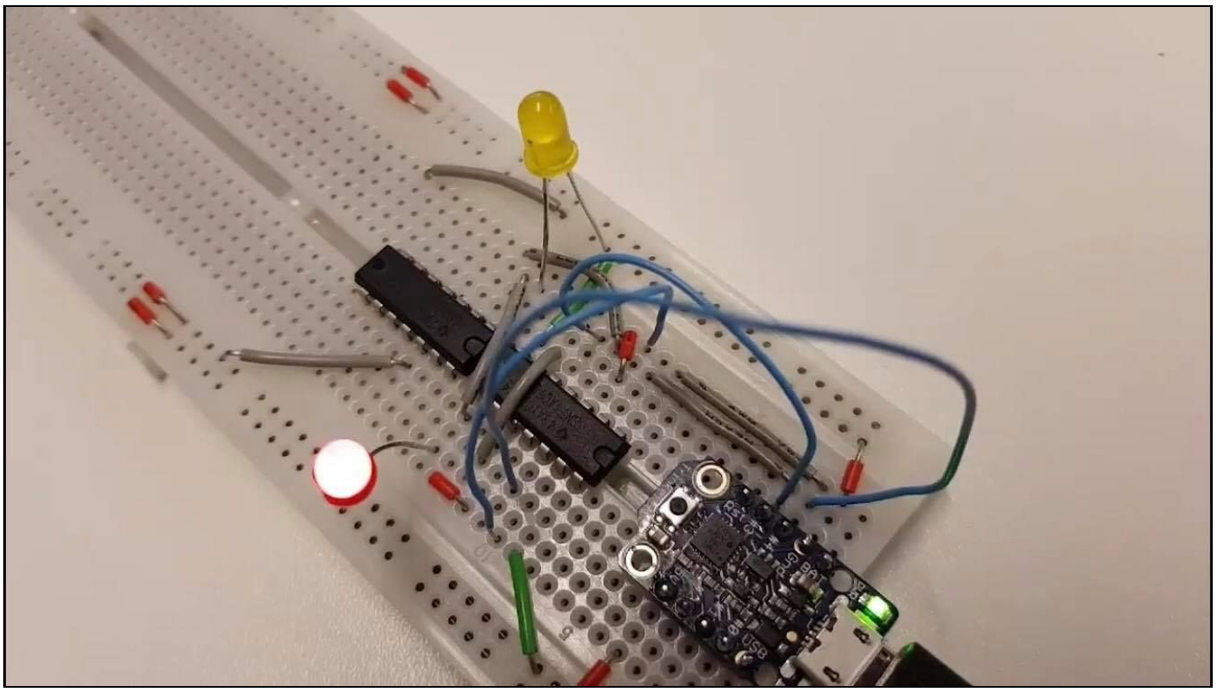
Outcome 1:

When the sum is 1 and the carry is 0, which means that either A or B is 1 and the other is 0.



Picture 7, Half adder

The diagram in *picture 7* demonstrates this outcome. The input in this picture is A=1 & B=0 while the output is 1 for the sum and 0 for the carry.

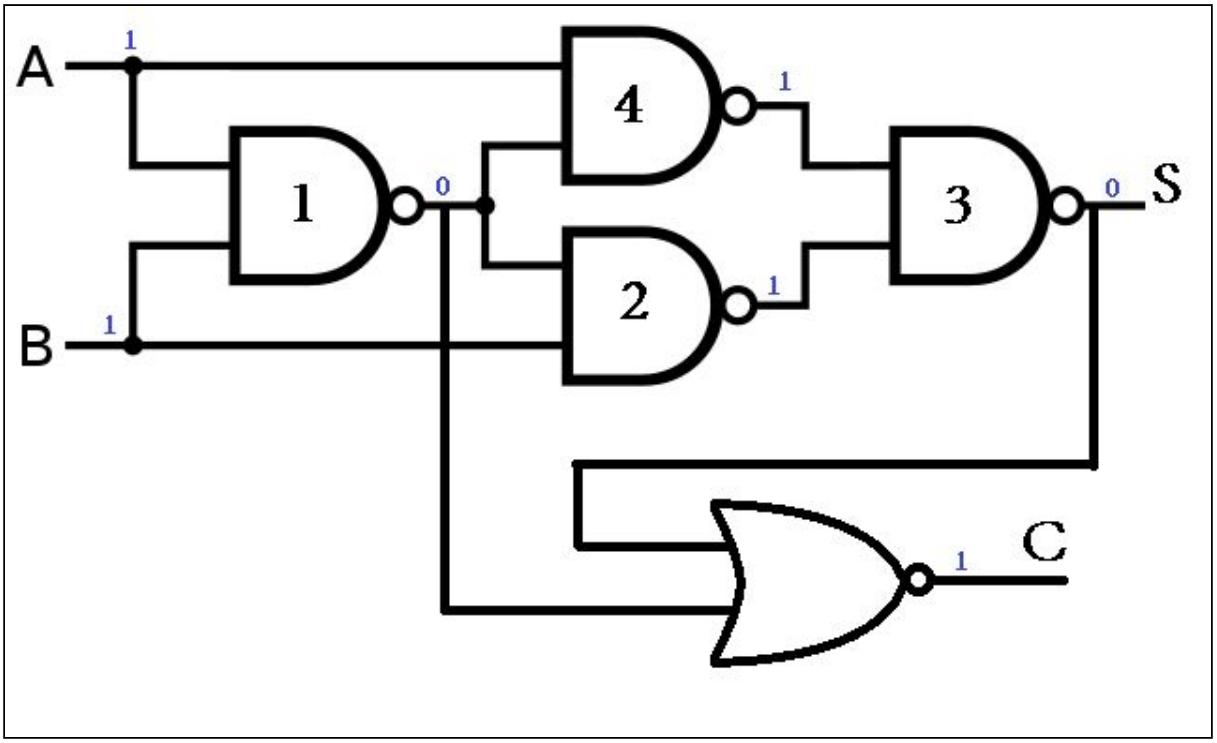


Picture 8, Outcome 1

When looking at the truth table in *Table 2* where A = 1 and B = 0 you can see that the sum is 1 and the carry is 0. In *Picture 8* you can see that the sum LED is lit up and the carry LED is not, just as predicted in *Picture 7*.

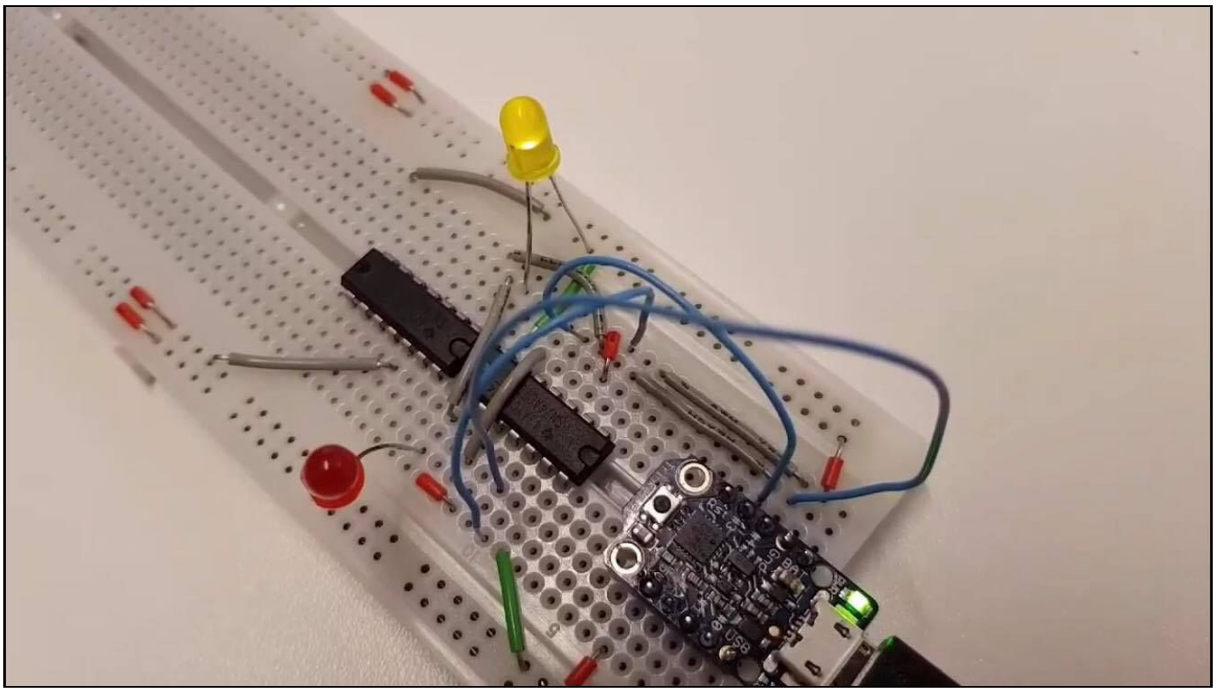
Outcome 2:

When A=B=1. This should result in the sum being 0 and the carry to be 1.



Picture 9, Half adder

Row 4 in *Table 2* illustrates *Outcome 2*. You can read that A = B = 1 should give 1 as carry and 0 as the sum. This is the same situation as you can follow on *Picture 9*.

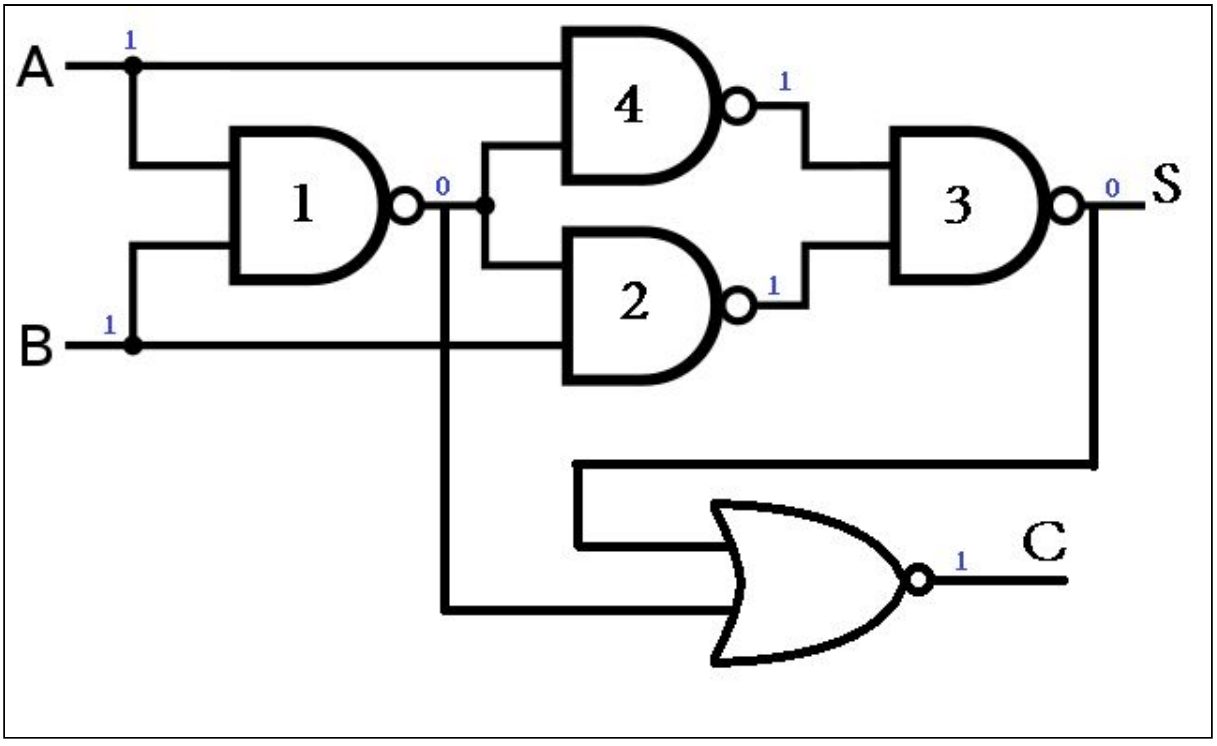


Picture 10, Outcome 2

When circulating through *Code Snippet 1* the yellow LED lit up only once per cycle and that was in this outcome. As can see on *picture 10* the LED (red) of the sum is not lit which means that it follows the outputs of *picture 9.*

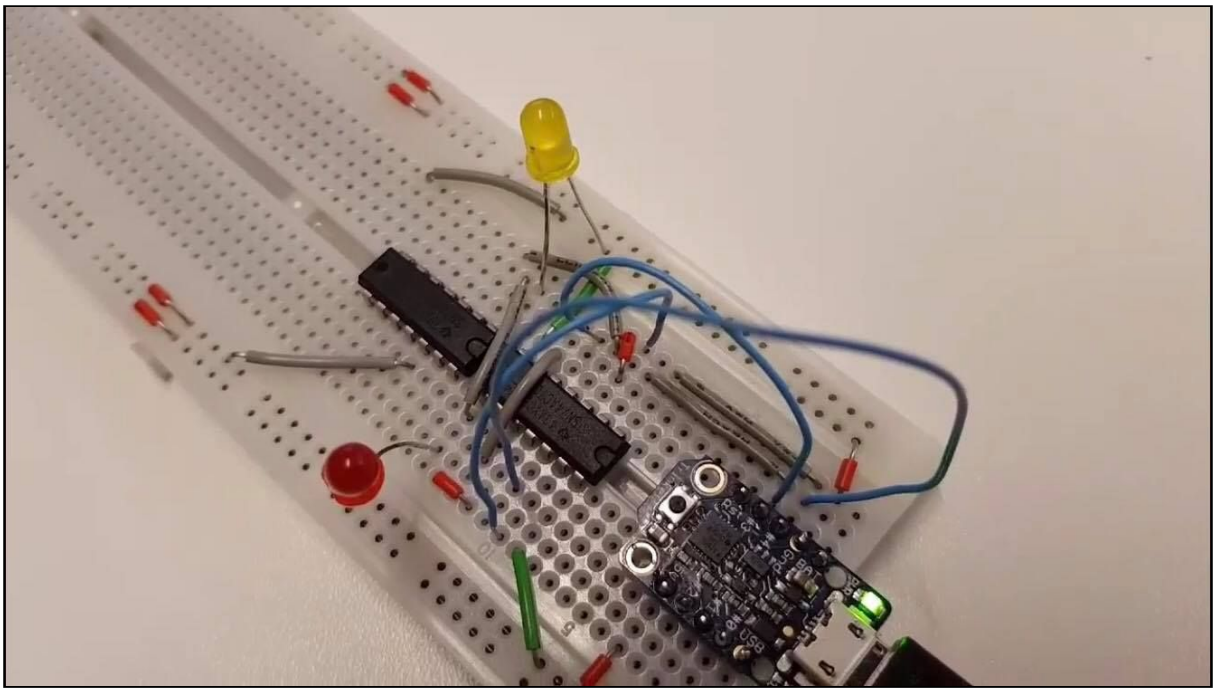
Outcome 3:

Where both input A=B=0.



Picture 11, Half adder

As described in *Picture 11,* the output should be 0 on both sum and carry.



Picture 12, Outcome 3

In this case both LEDs should be off which *picture 12* shows.

Conclusion

Pictures 1-4 indicates that our NAND gate is connected correctly and when looping through all permutations it follows the truth table of an NAND gate. The LED output lights up 3 times throughout the cycle.

Our solution for the half-adder using NAND and NOR gates, as shown in *Picture 6,* corresponds to the truth table for a half-adder. When cycling through the permutations the carry LED only lit up once while the sum LED lit up twice throughout the cycle. This indicates that are solution is a valid option to the solution shown in *Picture 5*.