Lab 1: Digital Logic Circuits

Task 1: Breadboard Setup and a NAND gate

We implemented the circuit as described in the instructions. A LED was used to see if power was flowing through the circuit. The picture below shows that the LED is lit (Figure 1).

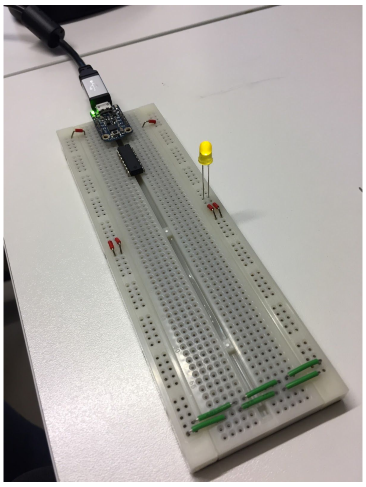


Figure 1: LED is lit

Next step was to see if the NAND chip worked correctly. The circuit was connected according to the chips documentation (7400 series) and connected #4 to 1A and #3 to 1B. A script was made to test the chip. The script loops through all combinations for the input signals. Table below shows the result and works as intended.

```
void setup()
{
  pinMode(4, OUTPUT);
  pinMode(3, OUTPUT);
}

void loop()
{
  digitalWrite(4, LOW); // Input: 0 0
  digitalWrite(3, LOW);
  delay(1000);

  digitalWrite(4, HIGH); // Input: 1 0
  digitalWrite(3, LOW);
  delay(1000);

  digitalWrite(4, LOW); // Input: 0 1
  digitalWrite(3, HIGH);
  delay(1000);

  digitalWrite(4, HIGH); // Input: 1 1
  digitalWrite(3, HIGH);
  delay(1000);
}
```

A(#4)	B(#3)	Output (NAND)	LED
0	0	1	ON
0	1	1	ON
1	0	1	ON
1	1	0	OFF

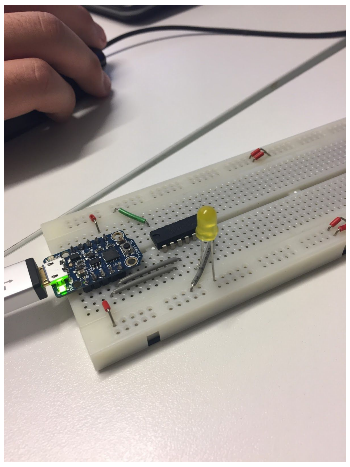
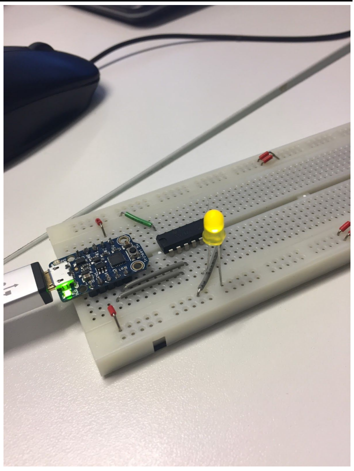



Figure 2 Figure 3

Task 2: Half-adder

The second task was to construct an half-adder circuit using only 4 NAND and one NOR gates.

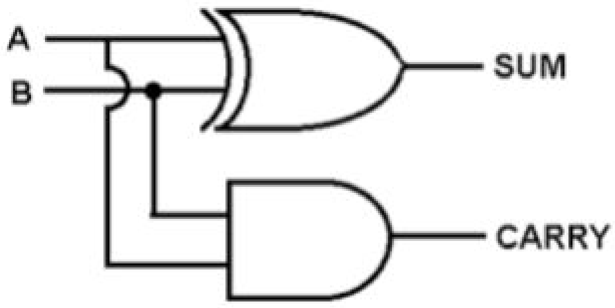


Figure 4: An Half-adder circuit

An half-adder (Figure 4) consists of an XOR gate and an AND gate, the different gate logics can be simulated using other gates. The base of the new circuit was made using four NAND gates coupled in order to simulate the logic of an XOR gate.

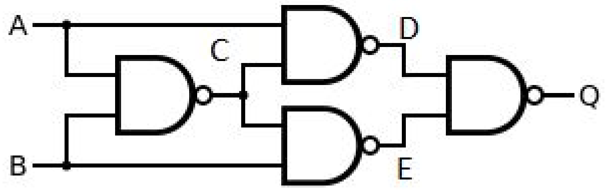


Figure 5: XOR gate constructed using NAND gates

This circuit simulates the logic of an XOR gate and grants the correct SUM value but lacks the carry output. To get the carry the remaining NOR gate has to be connected with specific input nodes, such so the output corresponds with an regular AND gate with A and B as inputs.

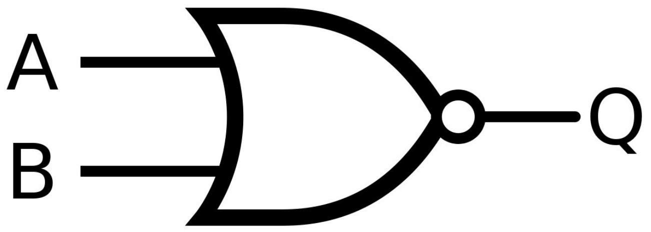


Figure 6: NOR gate.

Carry	A	B
0	0	0
0	0	1
0	1	0
1	1	1

As shown in the table the we need to find nodes that give the right values depending on the state of A and B.

Carry	**NOR**	A	B	**C**	D	E	**Q**
0	**0**	0	0	**1**	1	1	**0**
0	**0**	0	1	**1**	1	0	**1**
0	**0**	1	0	**1**	0	1	**1**
1	**1**	1	1	**0**	1	1	**0**

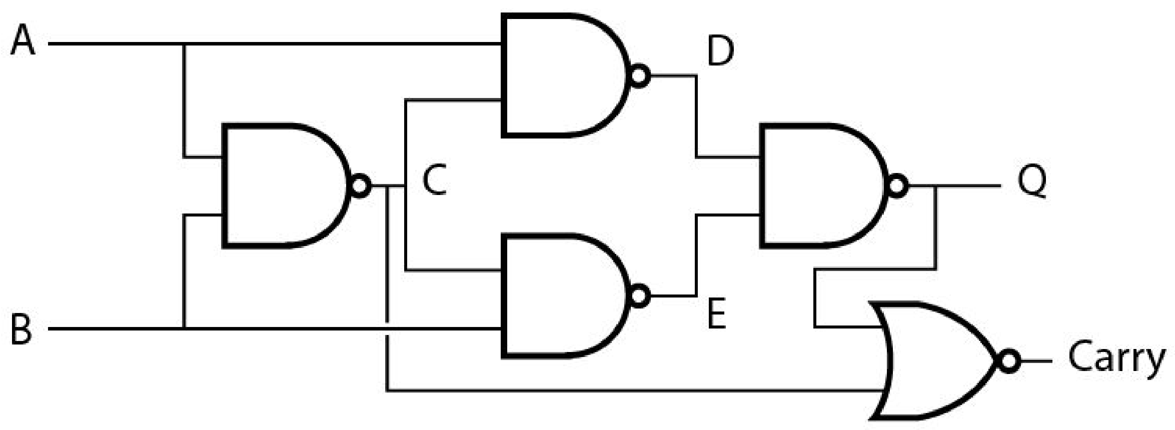
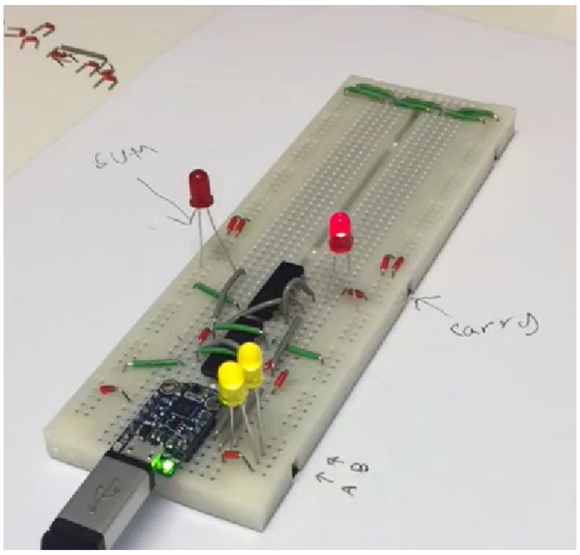
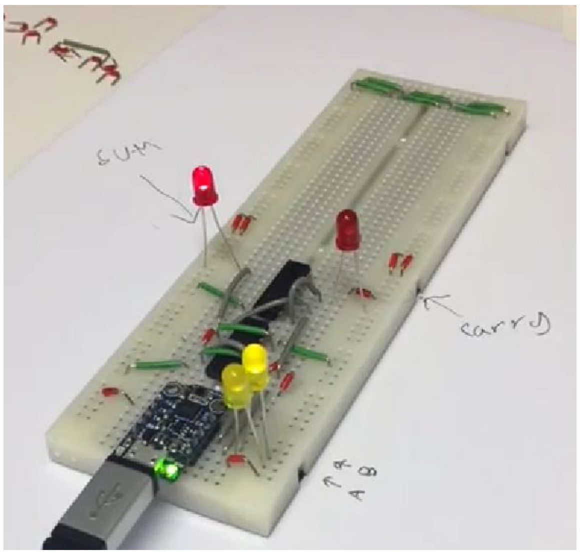


Figure 7: Half-adder consisting of 4 NAND gates and 1 NOR gate.

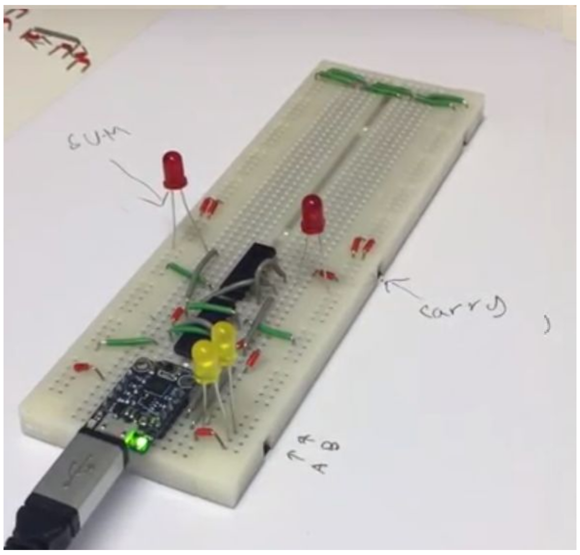
This table confirms that having nodes C and Q as inputs in the NOR gate will yield correct carry values.



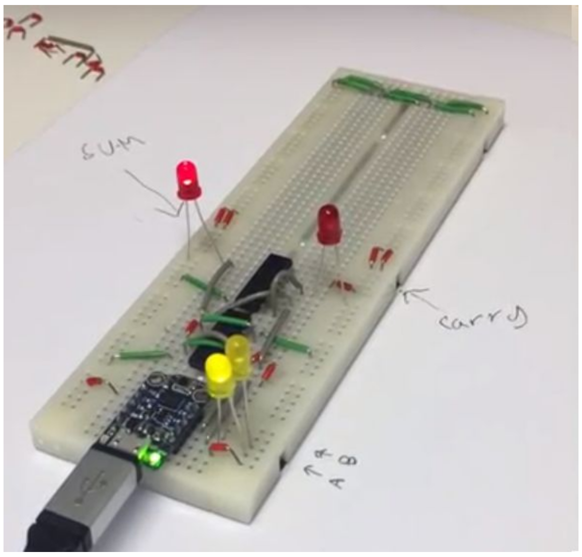
Input: 1 1



Input 0 1



Input 0 0



Input 1 0

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Task 3: Full-adder and 2-bit circuit

For the implementation of the full-adder the schematic from the lab assignment was used (figure 8).

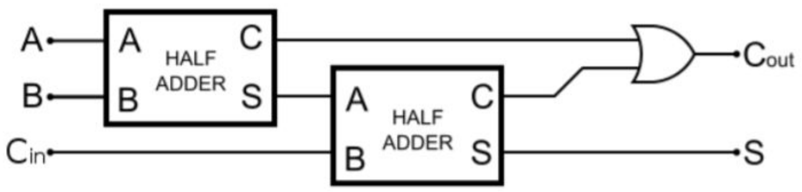


Figure 8: Full-adder basic design.

The circuit was built using only NAND and NOR gates (figure 9). Two more half-adders were added onto the circuit and to get the correct Cout value from the circuit two NOR gates were used to emulate the logic of a OR gate.

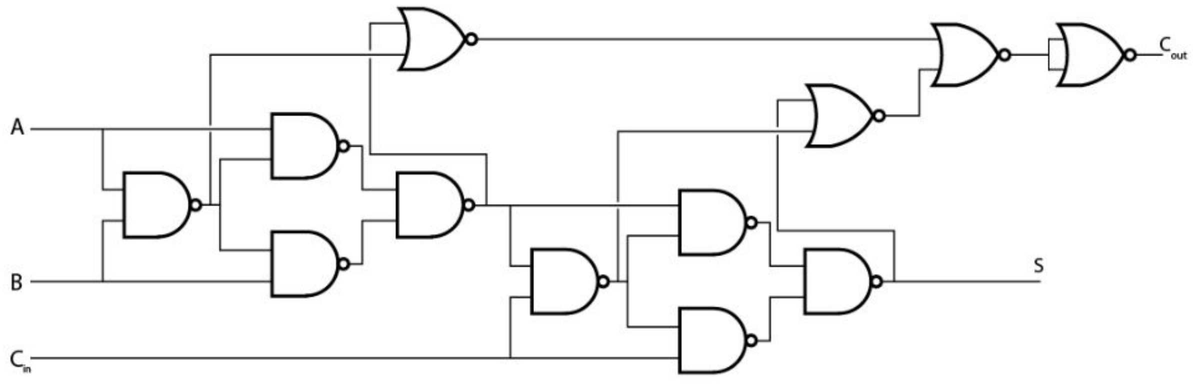


Figure 9: Full-adder with NAND and NOR gates.

As stated in the task another input Cin was added onto the arduino to test the circuit prior to actually connecting the output from the half-adder as the carry-in into the full-adder.

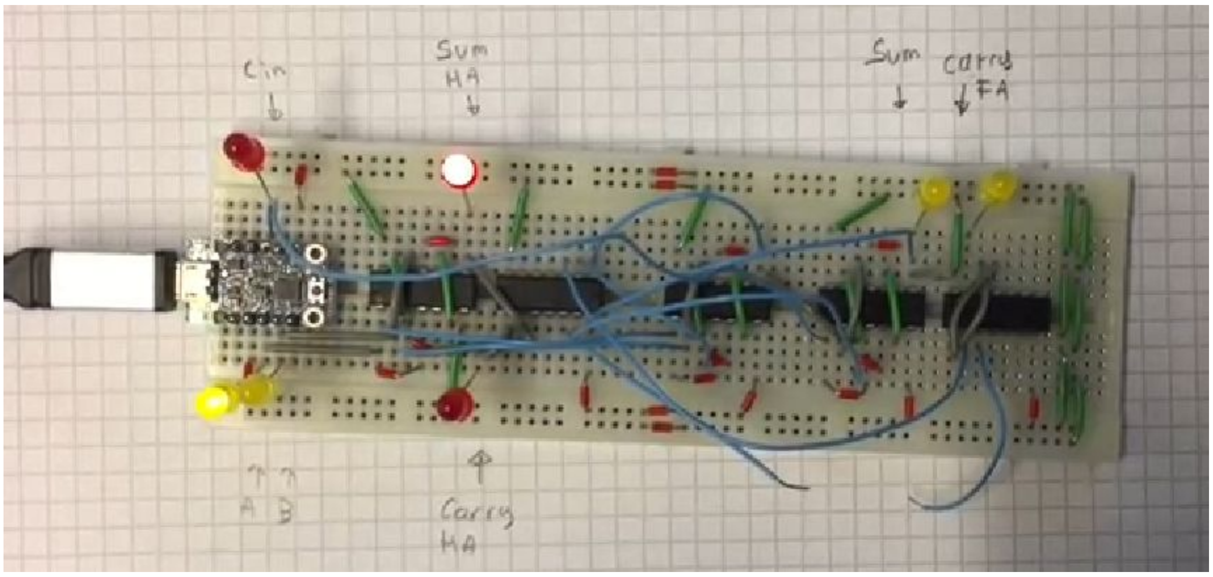


Figure 10: Full-adder and a half-adder on a breadboard with Cin as input from the arduino.

Table below shows the full-adders inputs A, B, Cin and the outputs Cout and Sum.

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1