Lab 1

Test case

In all the test cases we used a simple code the iterated through all the possible states that two bits can provide. The following states are activated with a delay between all the states with one second.

- A and B = HIGH
- A = LOW, B = HIGH
- A = HIGH B = LOW
- A and B = LOW

This way we can observe all the possible states and verify that our code works as intended. All the solutions were observed and compared to a truth table to ensure that the circuit works as intended.

Task 1: Breadboard Setup and a NAND gate

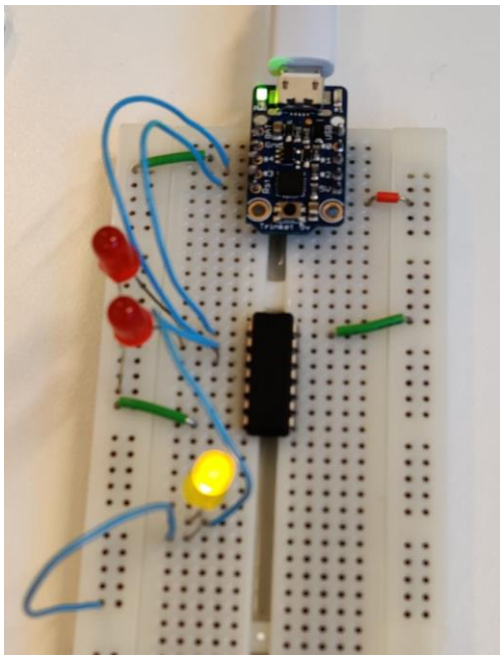
The main task was to ensure that the trinket Arduino board was working, and that you were able to upload your code and run it.

After that we connected an array of NAND gates and made sure it had both power and ground connected. Then we connected two of the Arduino pins to the inputs to one of the NAND gates and made sure that the gate was operating as expected.

The truth table for a NAND gate is as following:

A	B	S
0	0	1
1	0	0
0	1	0
1	1	0

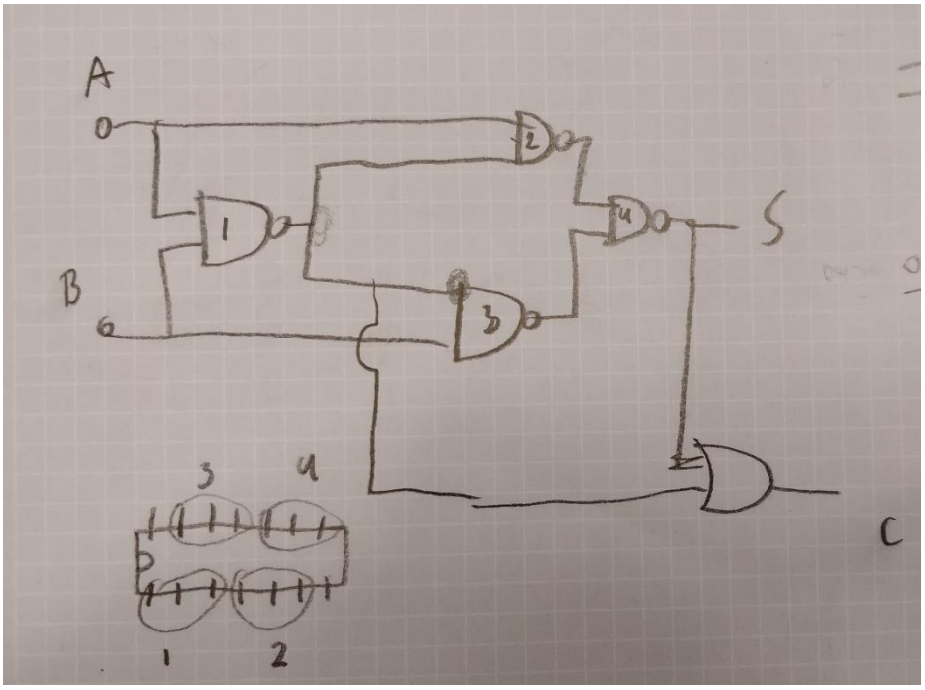
As seen in *Picture 1* we made the state of both A and B visible by having red LEDs show if the were either HIGH or LOW. The only case in which and NAND gate results in HIGH is the case in which A and B is set to LOW. The result at the gate out put can be seen on the yellow LED.



Picture 1

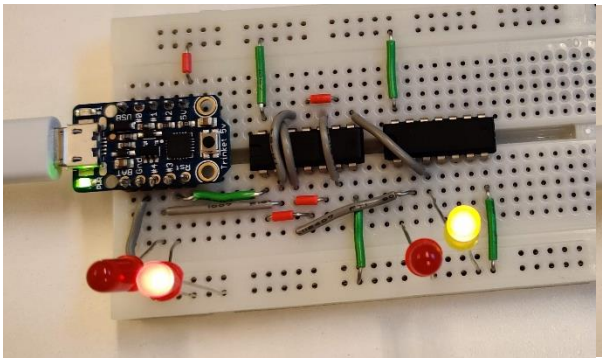
Task 2: Half-adder

This task was to create a half adder with four NAND and one NOR gate. The adder should accept two input bits and output the sum bit and a carry bit.

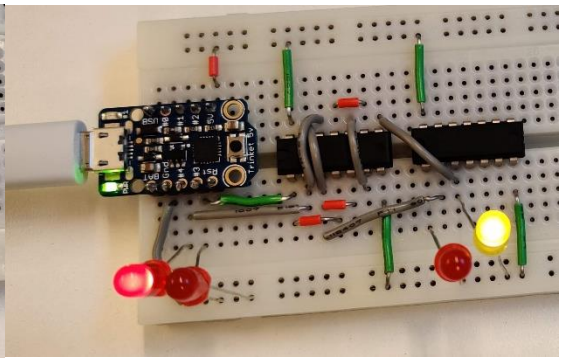


Picture 2

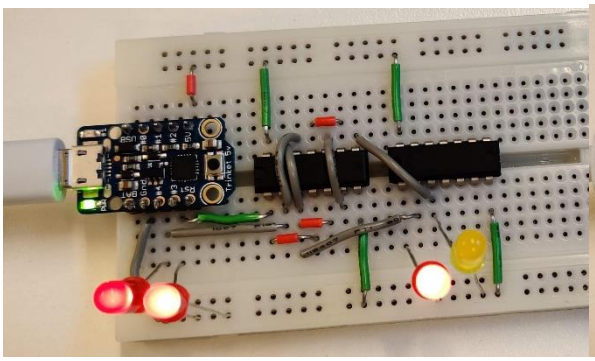
We used the schematic seen in *Picture 2* to create a one-bit half adder.



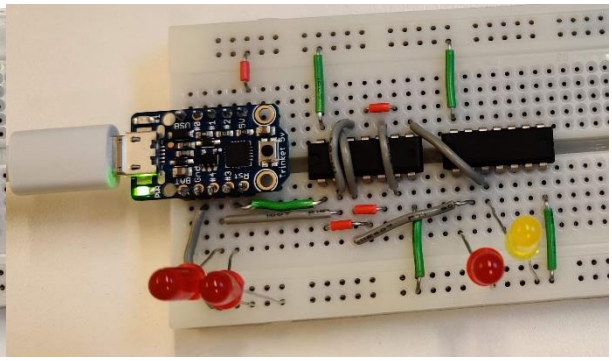
Bread board 1



Bread board 2



Bread board 3



Bread board 4

We tested the adder with all different inputs to verify that it worked correctly. In the pictures, the leftmost red LEDs are the input, the yellow LED to the right shows the sum and the red LED shows the carry. This leads to the table:

A0	B0	S	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

According to our test, the half adder was correct.

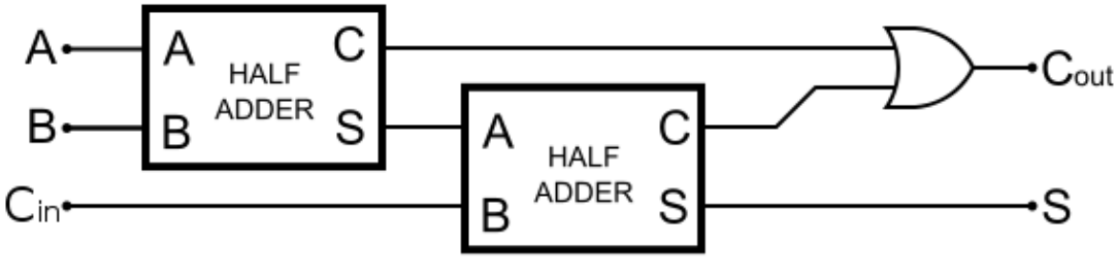
```
int a = 3; // Pin 3 and 4
int b = 4;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);

}

// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH );
    delay(1000);
    digitalWrite(a, LOW);
    delay(1000);
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    delay(1000);
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    delay(1000);
}
```

Code 1

The code we wrote to test our half adder can be seen in *Code 1*. This was done by initialize two of the outputs on the Trinket and alternating them between low and high in different configurations.

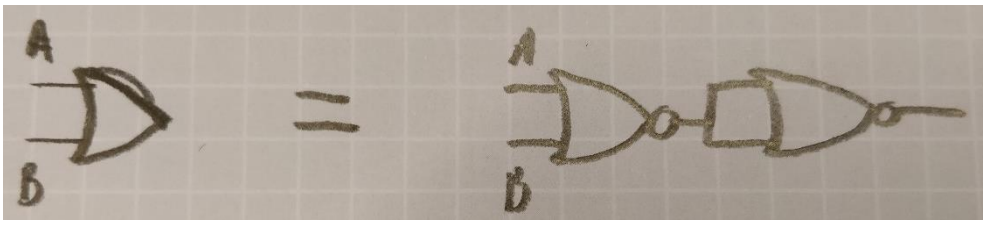
Task 3: Full-adder and 2-bit adder circuit

Having devised a half adder with NAND and NOR gates in the previous task and with the tip of using two half adders and a NOR gate to create a full adder (See *Picture 3)*, We decided to create the full adder by simply hooking up to of the same half adders and connecting them in the appropriate manner to form a Full adder.



Picture 3

The full adder provided is using an OR gate but none was provided. We solved this by making a wiring two NOR gates in series in the manner seen in PICTURE X



Picture 4

The Arduino code was modified and a third input was added to simulate getting a carry in bit. All the three inputs had an LED to indicate where they showed if it was HIGH or LOW.

After wiring up an additional full adder using two quad NAND gates and a quad NOR gate we verified that the result was the same as the ones found in truth table

The truth table for a one-bit full adder is as following:

A0	B0	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The case seen in *Picture 4* is the case in where:

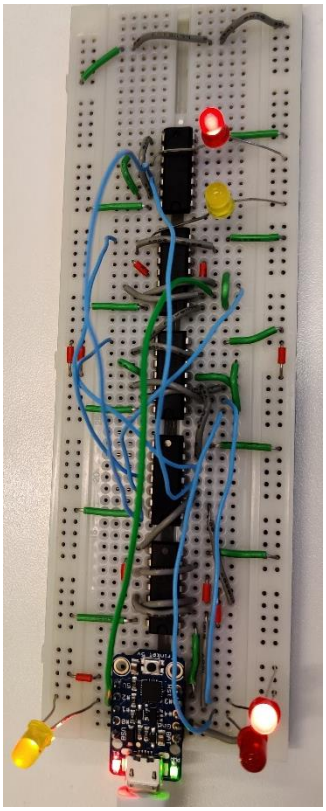
A = 0, B = 1 and Carry in = 1

Wich results in the expected manner: Sum = 0, Carry out = 1

The code wich runs the full adder can be seen in *Code 2*.

```
int a = 3; // Pin 3 and 4
int b = 4;
int c = 1;
bool check = true;
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);

}

// the loop routine runs over and over again forever:
void loop()
{
  if(check)
  {
    digitalWrite(c, HIGH);
    check = false;
  }
  else
  {
    digitalWrite(c, LOW);
    check = true;
  }
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH );
  delay(1000);
  digitalWrite(a, LOW);
  delay(1000);
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  delay(1000);
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  delay(1000);
}
```

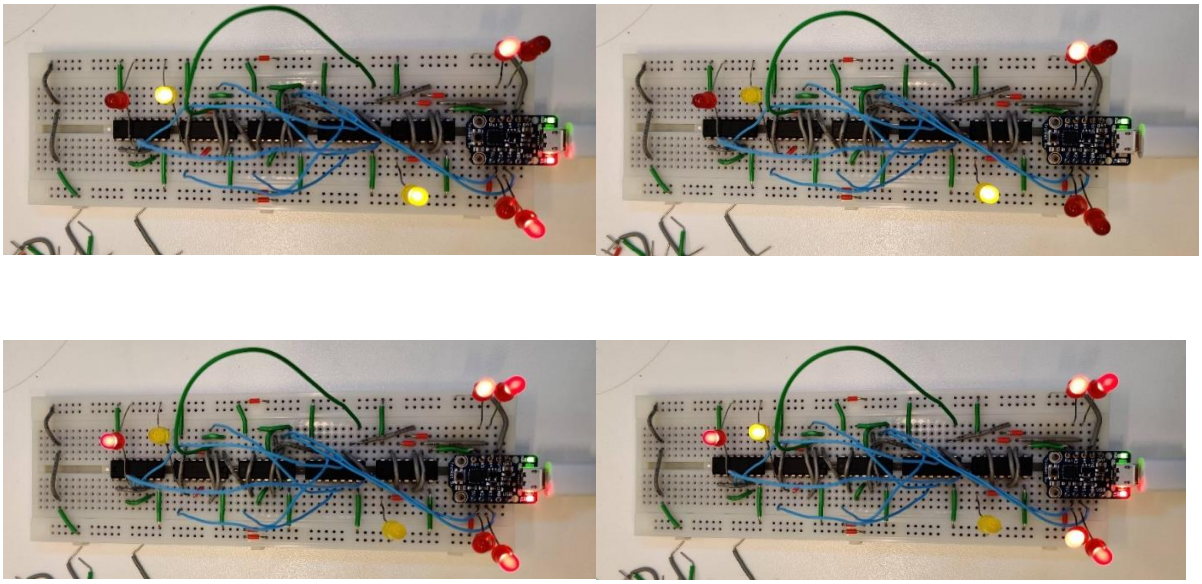
Code 2



Picture 5

2 bit adder

The last task was to implement a 2 bit adder. This was done by adding the carry out from the half adder in task 1 to the carry in on the full adder. Then, by rewiring the inputs so that a0 and b0 was connected to the half adder, and a1 and b1 connected to the full adder, we ended up with a 2 bit adder.



Our breadboard containing a working 2 bit adder. Here showing four different cases. The LEDs from the left: Carry (red), S1 (yellow), S0 (yellow).

A1	A0	B1	B0	S0	S1	Carry
1	0	0	1	1	1	0
0	1	0	0	1	0	0
1	1	0	1	0	0	1
1	1	1	1	1	1	0

```
int a0 = 3; // Pin 3 and 4
int a1 = 1;
int b0 = 4;
int b1 = 2;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(a0, OUTPUT);
  pinMode(a1, OUTPUT);
  pinMode(b0, OUTPUT);
  pinMode(b1, OUTPUT);

}

// the loop routine runs over and over again forever:
void loop()
{
  // Bör bli 1 0 + 0 1 = 1 1 = 3
  digitalWrite(a0, HIGH);
  digitalWrite(b1, LOW);
  digitalWrite(a1, HIGH);
  digitalWrite(b0, LOW);
  delay(6000);
  digitalWrite(a0, HIGH);
  digitalWrite(b1, LOW);
  digitalWrite(a1, LOW);
  digitalWrite(b0, LOW);
  delay(6000);
  digitalWrite(a0, HIGH);
  digitalWrite(b1, LOW);
  digitalWrite(a1, HIGH);
  digitalWrite(b0, HIGH);
  delay(6000);
  digitalWrite(a0, HIGH);
  digitalWrite(b1, HIGH);
  digitalWrite(a1, HIGH);
  digitalWrite(b0, HIGH);
  delay(6000);
}
```

Code 3, the code for running and testing the 2 bit adder.