

Viktor Rask
Amanda Håkansson
DT509G

Lab 1

Computer Architectures for MSc in Engineering



Table of Contents

| | |
|------------------------------|----|
| Introduction..... | 2 |
| Task 1..... | 3 |
| Setup..... | 3 |
| Result..... | 3 |
| Method of verification | 5 |
| Task 2..... | 5 |
| Setup..... | 5 |
| Result..... | 6 |
| Method of verification | 7 |
| Task 3..... | 8 |
| Setup..... | 8 |
| Result..... | 9 |
| Method of verification | 11 |

Introduction

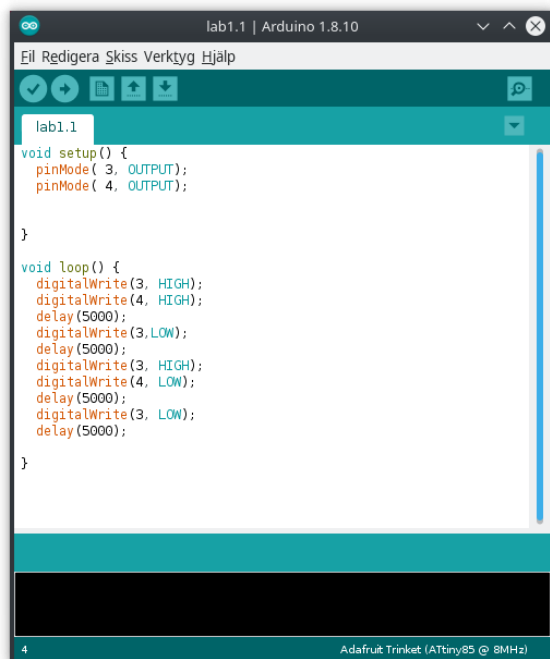
The purpose of this lab was to create a 2-bit adder with the ultimate goal of learning how to build digital logic circuits on a breadboard and how to interface to a microcontroller. The 2-bit adder were supposed to be created following 3 tasks.

Task 1

Task 1 was to setup the breadboard, a NAND gate and to test the NAND gate.

Setup

First the required materials breadboard, NAND chip, Adafruit trinket board and cables were gathered. Then all wires got connected according to instructions. A LED was used to test if there were power going through the circuit. After verifying whether power was going through, the NAND chip was added, and a NAND gate was connected. The a and b inputs of the NAND gate got connected to pin 4 and 3 on the Adafruit trinket board to be able to give high and low inputs. For final circuit in Task 1 see figure 1. Arduino ide was used to program the different inputs to the NAND gate where high was equal to a 1 and low was equal to 0, the code can be seen below.



```
lab1.1 | Arduino 1.8.10
Fil Redigera Skiss Verkttyg Hjalp

lab1.1
void setup() {
  pinMode( 3, OUTPUT);
  pinMode( 4, OUTPUT);
}

void loop() {
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  delay(5000);
  digitalWrite(3, LOW);
  delay(5000);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  delay(5000);
  digitalWrite(3, LOW);
  delay(5000);
}

4 Adafruit Trinket (ATTiny85 @ 8MHz)
```

Result

The result of task 1 was a functioning circuit, a working NAND gate and code which was used to test the NAND gate.

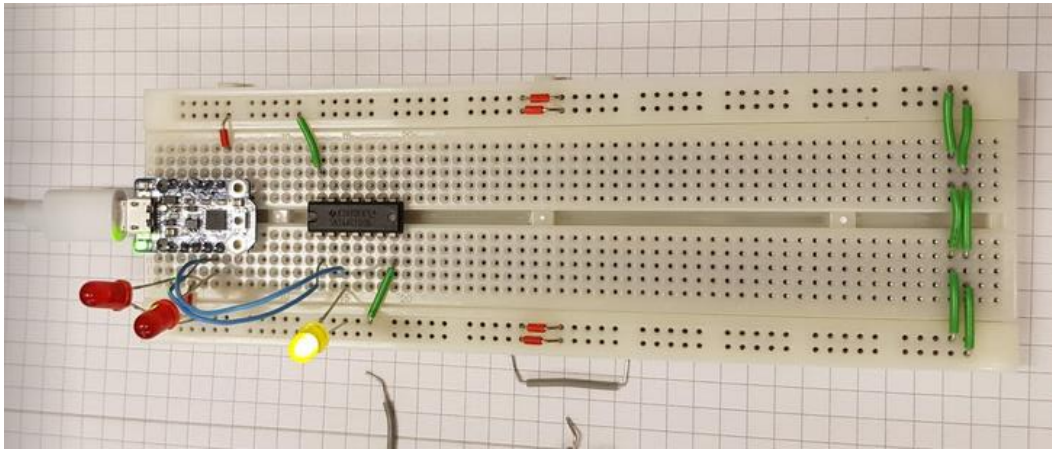


Figure 1, $NAND = 1$ when $A = 0$ and $B = 0$

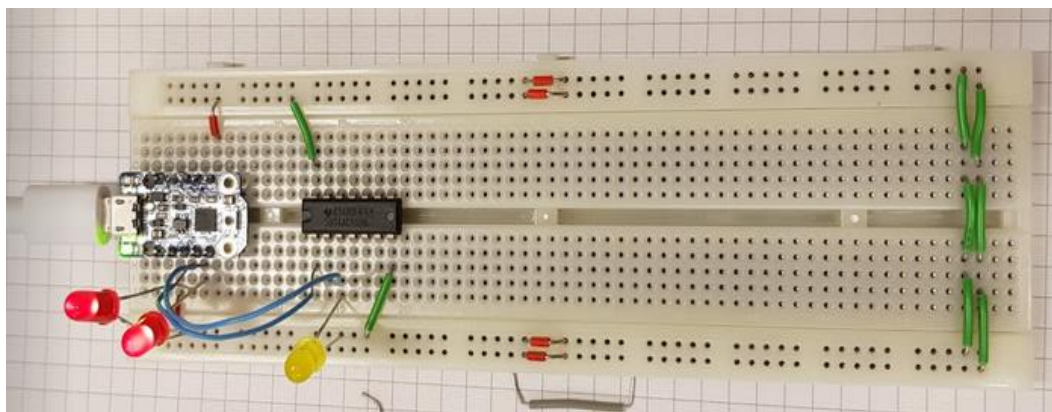


Figure 2, $NAND = 0$ when $A = 1$ and $B = 1$

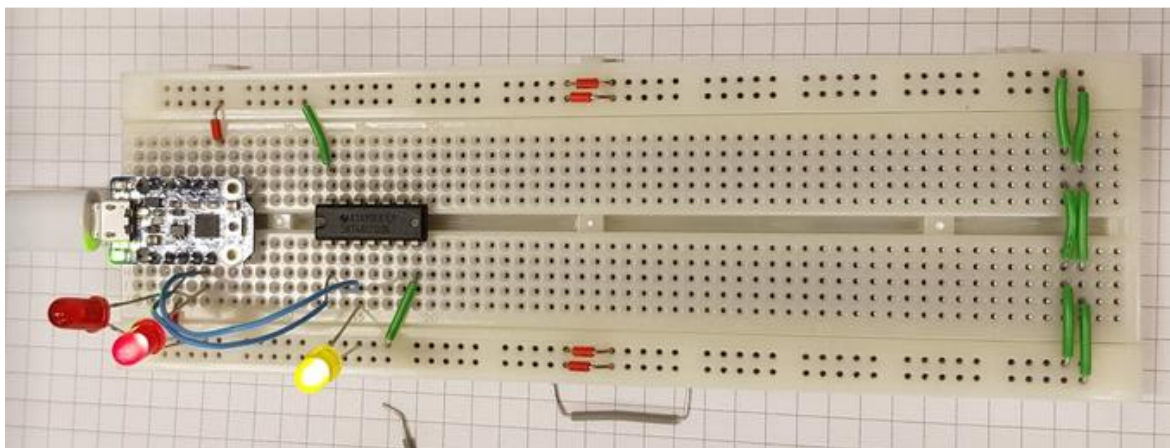


Figure 3, $NAND = 1$ when $A = 0$ and $B = 0$

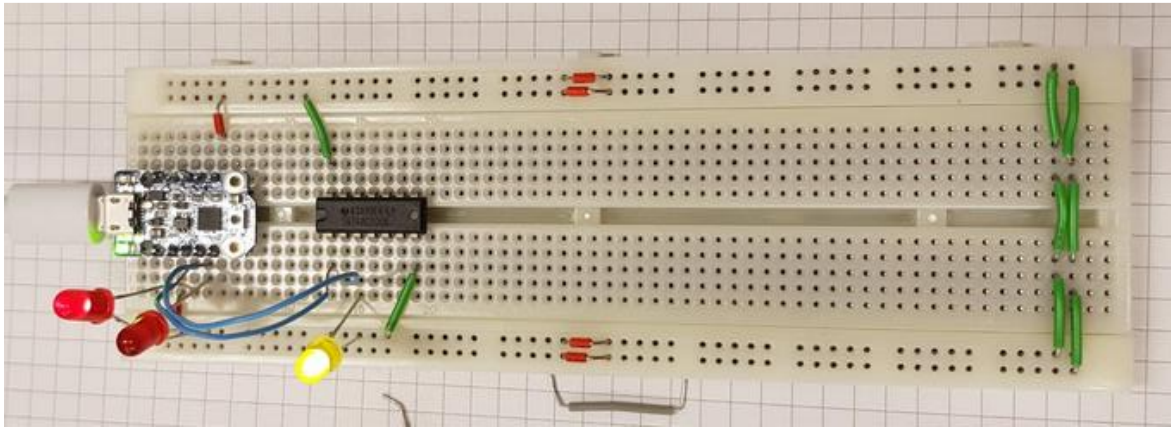


Figure 4, $NAND = 1$ when $A = 1$ and $B = 0$

Method of verification

To verify the first part of task 1 which was to see if power was going through the circuit a LED was used. To test the NAND gate more LED: s where used but this time they were set up at the NAND gate output and at the number 3 and 4 pins of the Arduino trinket board. This was to see the different outputs depending on what input pin 3 and 4 gave. Basically, the method of verification for the NAND gate was to see if the LED: s was acting according to the truth table of a NAND gate where a 1 was equal to the LED lighting up.

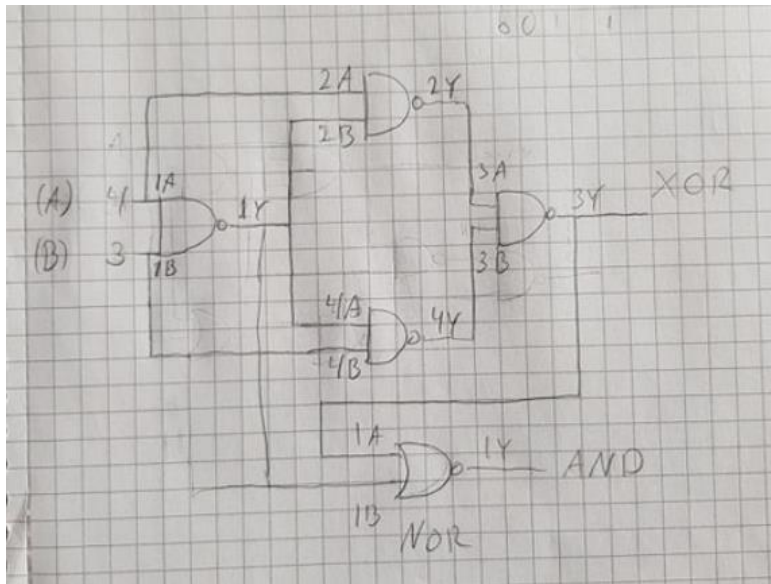
| A | B | NAND |
|---|---|------|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Task 2

Task 2 was to construct a half-adder circuit that took two inputs (bits A and B) and delivered two outputs, the sum and the carry. And only use a maximum of 4 NAND gates and 1 NOR gate.

Setup

The first step was to rewrite the half-adder circuit that was implemented with one XOR and one AND gate to a half-adder circuit implemented with four NAND and one NOR gate. That was done by looking at the truth table for the circuit.



The circuit from task 1, one NOR chip, some cables and one additional LED was used to create the half-adder circuit. The LEDs was connected to the inputs, A and B, and the outputs, sum and carry, to verify that the circuit behaved as intended. The code that provided the inputs to the circuit was the same code as in Task 1.

Result

The result was a working half-adder circuit that was implemented using only four NAND and one NOR gate.

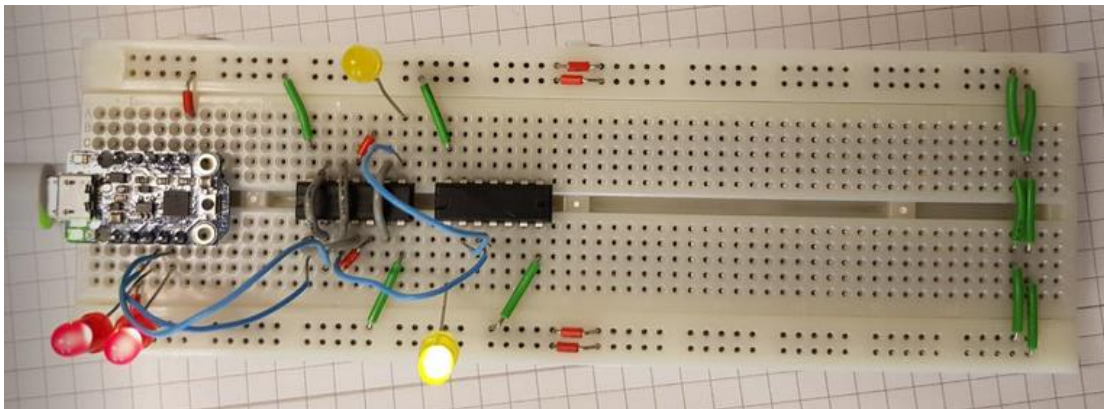


Figure 5, AND (carry) = 1 and XOR (sum) = 0 when A = 1 and B = 1

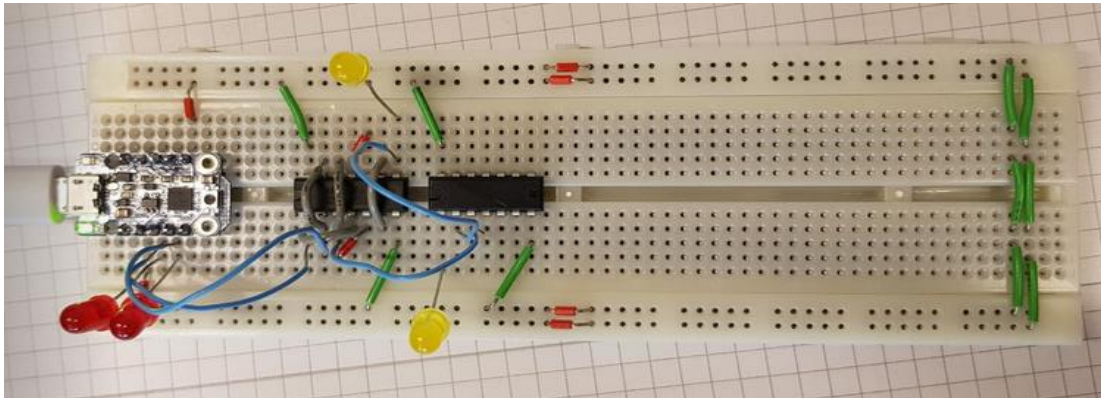


Figure 6, AND (carry) = 0 and XOR (sum) = 0 when A = 0 and B = 0

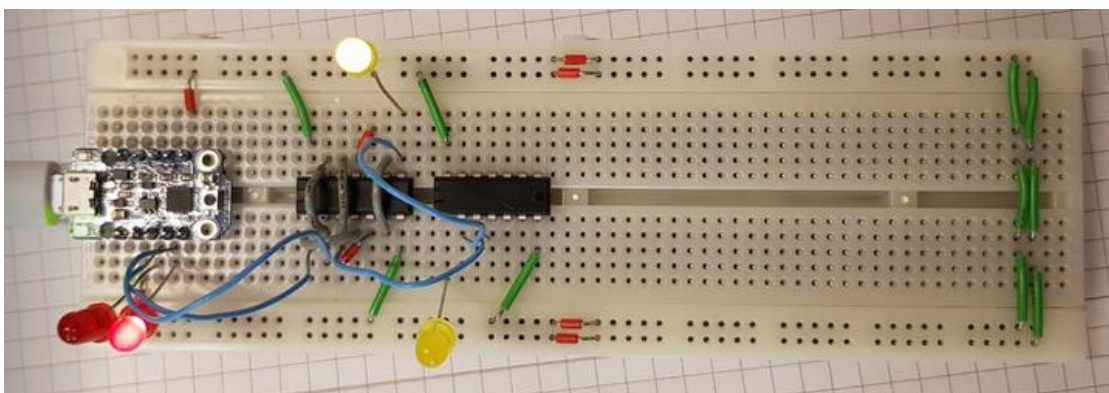


Figure 7, AND (carry) = 0 and XOR (sum) = 1 when A = 0 and B = 1

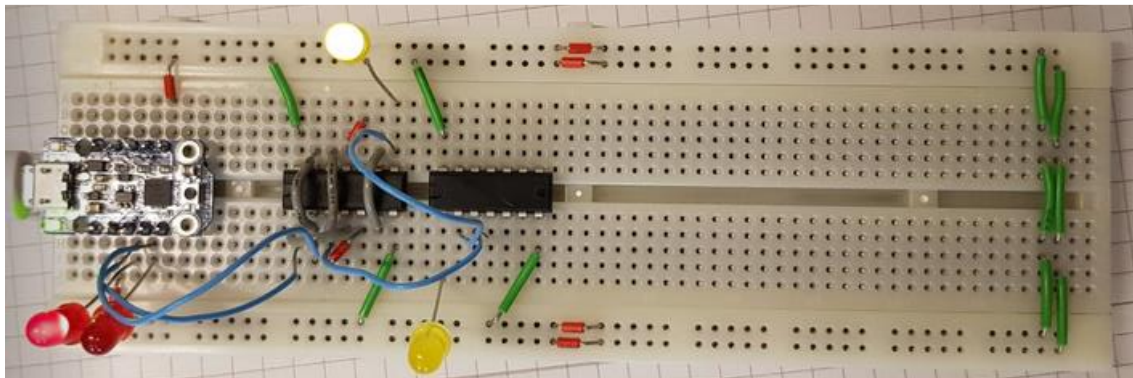


Figure 8, AND (carry) = 0 and XOR (sum) = 1 when A = 1 and B = 0

Method of verification

To verify that the circuit worked as expected, four LEDs was used. The red LEDs represented the input and the yellow LEDs represented the sum and the carry. The figures 5-8 represent the four different states corresponding to the following truth table.

| A | B | Sum | Carry |
|---|---|-----|-------|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

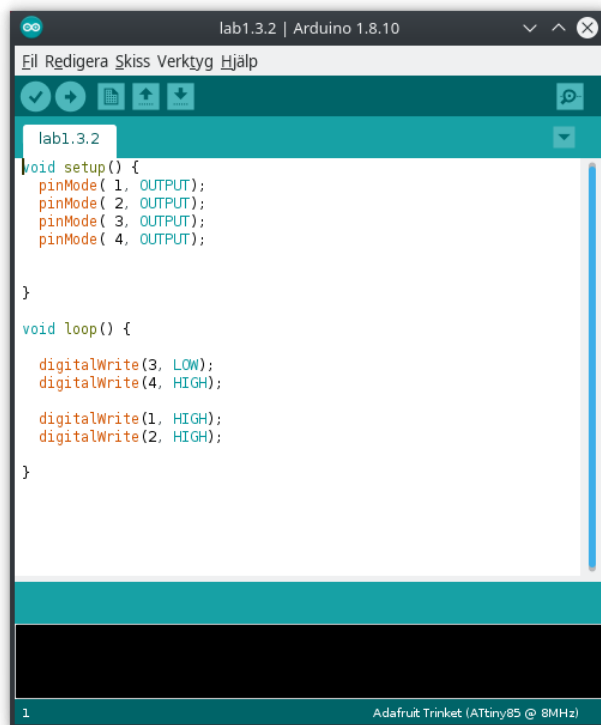
Task 3

Purpose of task 3 was to set up a 2-bit adder by combining a half adder and a full adder.

Setup

The first step for this task was to create a full-adder which is created by combining 2 half-adders. Since one half-adder was constructed in task 2 that circuit was simply copied to achieve two half-adders. Then the next step was to combine them where the sum of the first half-adder was the A input for the other one. The B input for the other half-adder got connected to pin 1 on the Adafruit trinket board where Arduino ide was used to simulate the carry in input for the full-adder.

When that circuit was done and tested via truth table and LED: s the next step was to add another half-adders carry out into the full-adders carry. The half-adder was constructed in the same way the other 2 were and then the carry out was connected to the full-adder carry in. The last step was to use two more pins from the Adafruit trinket board to get 4 different inputs which would simulate a 2-bit number. The pins 1 and 2 got connected to the A and B inputs of the last half-adder. The code that was used for one of the inputs can be seen down below. All the different states were tested in this way.



```
lab1.3.2 | Arduino 1.8.10
Eil Redigera Skiss Verktyg Hjälp

lab1.3.2
void setup() {
  pinMode( 1, OUTPUT);
  pinMode( 2, OUTPUT);
  pinMode( 3, OUTPUT);
  pinMode( 4, OUTPUT);
}

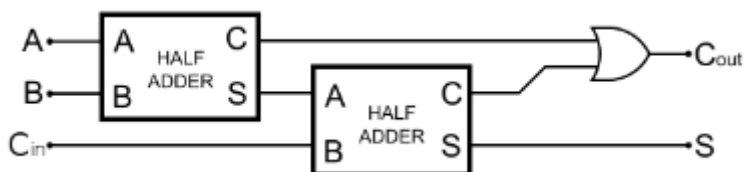
void loop() {
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);

  digitalWrite(1, HIGH);
  digitalWrite(2, HIGH);
}

1 Adafruit Trinket (ATtiny85 @ 8MHz)
```

Result

Our result became a circuit which allows for 2-bit addition with a half-adder and a full-adder. The yellow LED:s got put at the wrong places because of a confusion in the lab instructions. The task was to show the outputs for the sum of the half-adder and the full-adder but since the instructions failed to be clear on what they meant by C0 and C1 we put them on the carry for the half-adders inside the full-adder. The confusion sparked from this picture:



Here we thought C0 was the carry for the first half-adder of the full-adder and that C1 was the carry for the second half-adder of the full-adder. Although this simply means that we did not check if the sum was correct but did check if all half-adders and the full-adder had the correct carry which in the end gives the same verification that the 2-bit adder works.

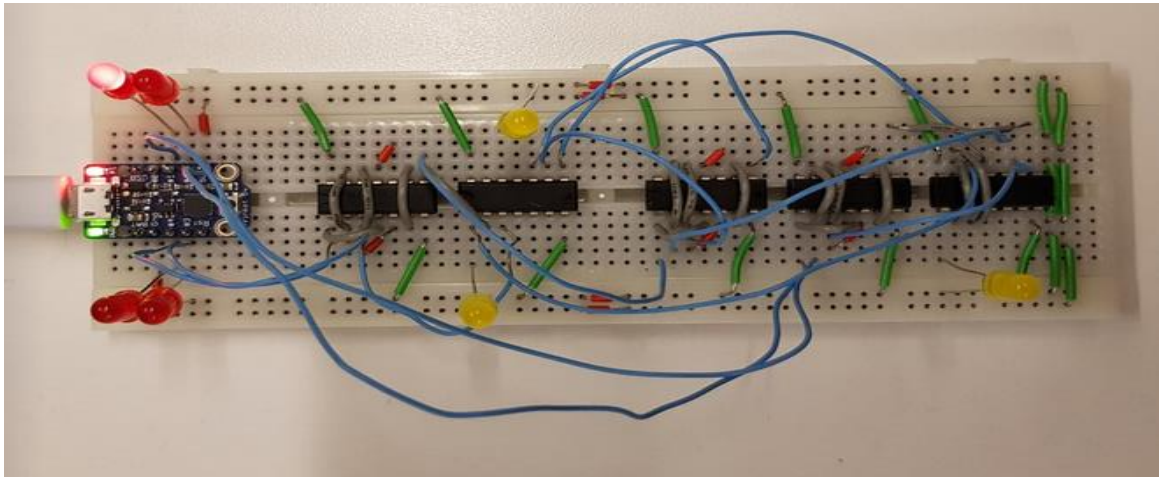


Figure 9, $C0 = 0$, $C1 = 0$, $C\text{-out} = 0$ when $a0 = 0$, $a1 = 0$ and $b0 = 1$, $b1 = 0$

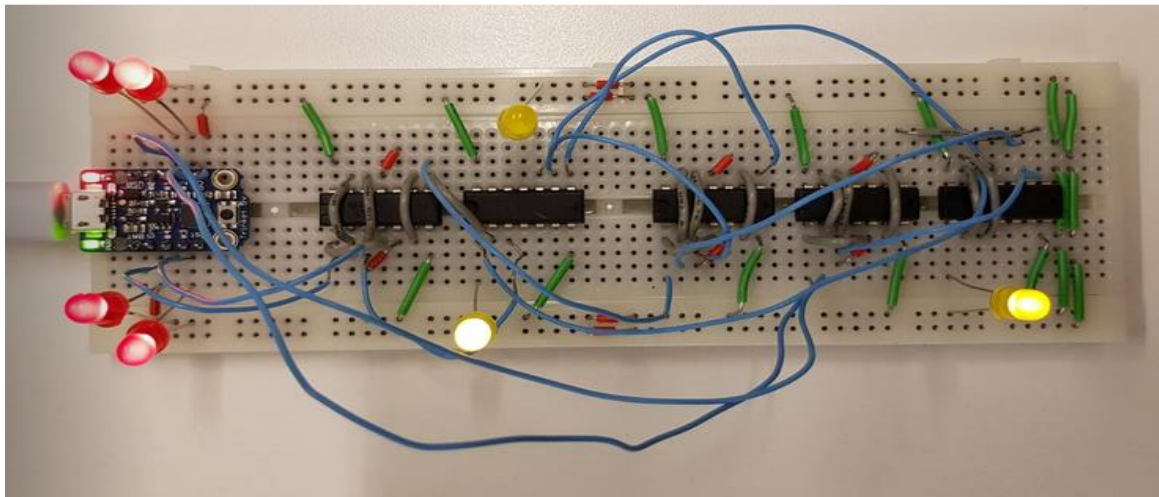


Figure 10, $C0 = 1$, $C1 = 0$, $C\text{-out} = 1$ when $a0 = 1$, $a1 = 1$ and $b0 = 1$, $b1 = 1$

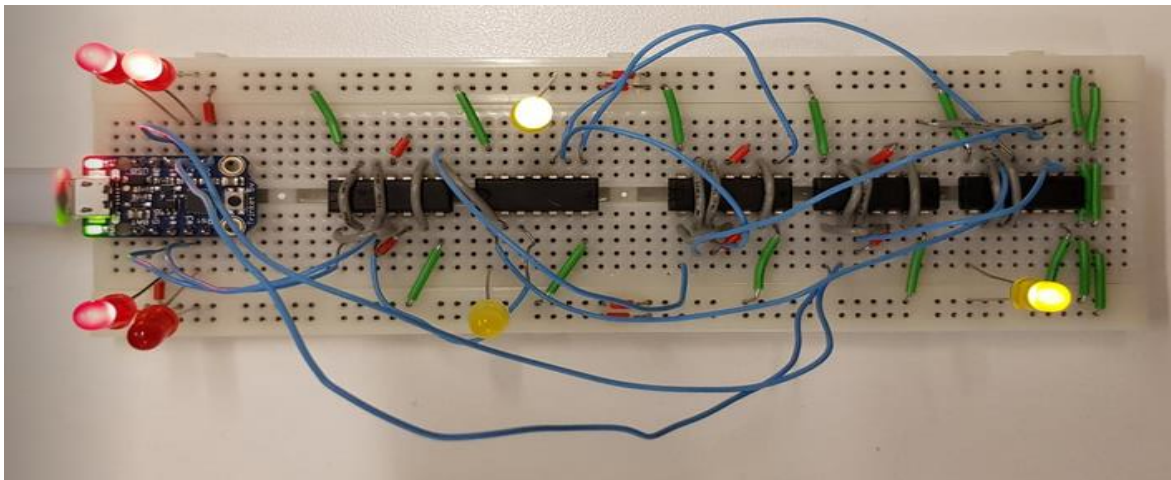


Figure 2, $C0 = 0$, $C1 = 1$, $C\text{-out} = 1$ when $a0 = 1$, $a1 = 0$ and $b0 = 1$, $b1 = 1$

Method of verification

The method used in verifying that the 2-bit adder was working properly was the same method used in the last 2 tasks which was to use truth table and LED: s to see if they showed the same answers. Because of some confusion from the instructions our truth table shows the carry output (C0 and C1) of two different half-adders and their collective full-adder carry out. A more correct method would be to put LED: s at the full-adders sum and the half-adders sum. Our truth table matched the outputs of the LED: s when different inputs were programmed into the Adafruit trinket board. Although our method did not clearly show the 2-bit additions sum, we can gather that the 2-bit adder circuit works because all half adders carry, and the full adders carry were correct.

| a0 | b0 | a1 | b1 | C0 | C1 | C-out |
|----|----|----|----|----|----|-------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |