In my next circuit I was able to break the program by not connecting all the nodes together and causing a divide by zero error in the programme. This stopped it from being able to correctly calculate the voltage over the resistors.

```
PS C:\Users\ludde\Desktop\3.3_Circuit> .\analyse .\mycircuit3.cir
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 5
           Nodes: 6
----------------------------
 Node   0 =  -1.#IND00 V
 Node   1 =  -1.#IND00 V
 Node   2 =  -1.#IND00 V
 Node   3 =  -1.#IND00 V
 Node   4 =  -1.#IND00 V
 Node   5 =  -1.#IND00 V
----------------------------
 I(V1)    =  -1.#IND00 A
----------------------------
```

10/11/20

https://secure.ecs.soton.ac.uk/notes/ellabs/1/c6/c6.pdf
https://blackboard.soton.ac.uk/webapps/blackboard/execute/content/file?cmd=view&content_id=_4912064_1&course_id=_190559_1
https://secure.ecs.soton.ac.uk/notes/ellabs/1/c6/SC08-11GWA.pdf
https://secure.ecs.soton.ac.uk/notes/ellabs/1/c6/pec11.pdf
https://secure.ecs.soton.ac.uk/notes/ellabs/1/c6/AST-030C0MR-R.pdf
https://en.wikipedia.org/wiki/USB#Power
https://www.dynapar.com/technology/encoder_basics/incremental_encoder/
https://www.dynapar.com/technology/encoder_basics/quadrature_encoder/

When working on an embedded system you should use avr libraries instead of std libraries. The library to control embedded inputs and outputs is <avr/io.h>.

To set a port to an input or output you must use the assign a value to DDR, like so:

```
DDRx = 0x00;        //Sets port “x” as inputs
PORTx = 0xFF;       //Enables inbuilt pull-up resistors

DDRy = 0xFF;        //Sets port “y” as outputs
```

Current limiting resistor:

$$V_S = V_F + V_R$$
$$V_R = V_S - V_F$$
$$V_R = 3.3 - 2.2$$
$$V_R = 1.1$$

$$V_R = IR$$
$$R = \frac{V_R}{I}$$
$$R = \frac{3.3}{10 \times 10^{-3}}$$
$$R = 110\Omega$$

It is not advised to replace the seven segment display current limiting resistors with a single resistor in the common line because every display output has a different amount of segments being lit. This means a different total voltage being provided to the display. If the display had one current limiting resistor then it will mean a different current for different numbers of segments, at best this will result in some numbers having less luminosity than others.

4. Justify that your board can supply enough current to power the display when all segments are illuminated, when powered from a USB port.

For low power devices USB can provide 100mA of current. In my X2 Lab I found that the Il Matto and power LED had a current draw of 7mA.

If each segment is lit then the display will have a current draw of 8x10mA. This is below the remaining current being delivered by the USB therefore the development board is able to drive the seven segment display.

USB power standards

Specification	Current	Voltage	Power (max.)
Low-power device	100 mA	5 V[a]	0.50 W
Low-power SuperSpeed (USB 3.0) device	150 mA	5 V[a]	0.75 W
High-power device	500 mA[b]	5 V	2.5 W
High-power SuperSpeed (USB 3.0) device	900 mA[c]	5 V	4.5 W
Multi-lane SuperSpeed (USB 3.2 Gen 2) device	1.5 A[d]	5 V	7.5 W
Battery Charging (BC) 1.1	1.5 A	5 V	7.5 W
Battery Charging (BC) 1.2	5 A	5 V	25 W
USB-C	1.5 A	5 V	7.5 W
	3 A	5 V	15 W
Power Delivery 1.0 Micro-USB	3 A	20 V	60 W
Power Delivery 1.0 Type-A/B	5 A	20 V	100 W
Power Delivery 2.0/3.0 Type-C	5 A[e]	20 V	100 W

a. ^ [a] [b] The V_{BUS} supply from a low-powered hub port may drop to 4.40 V.
b. ^ Up to five unit loads; with non-SuperSpeed devices, one unit load is 100 mA.
c. ^ Up to six unit loads; with SuperSpeed devices, one unit load is 150 mA.
d. ^ Up to six unit loads; with multi-lane devices, one unit load is 250 mA.
e. ^ > 3 A (60 W) operation requires an electronically marked cable rated at 5 A.

```c
#include <avr/io.h>
#include <util/delay.h>

/*
Pin     7 6 5 4 3 2 1 0
Segment a b c d e f g dp

0       1 1 1 1 1 1 0 0     0xFC
1       0 1 1 0 0 0 0 0     0x60
2       1 1 0 1 1 0 1 0     0xDA
3       1 1 1 1 0 0 1 0     0xF2
4       0 1 1 0 0 1 1 0     0x66
5       1 0 1 1 0 1 1 0     0xB6
6       1 0 1 1 1 1 1 0     0xBE
7       1 1 1 0 0 0 0 0     0xE0
8       1 1 1 1 1 1 1 0     0xFE
9       1 1 1 0 0 1 1 0     0xE6
*/

const uint8_t segments[10] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE0, 0xFE, 0xE6};
```

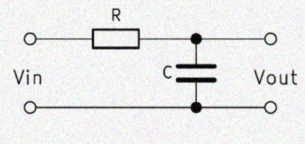
It is a good practice to enable the internal pull-up resistors (~35kΩ) to pull the input high (and not floating) when the button is not being pressed.

On the input pins voltages of less than 0.99V are read as low and voltages of more than 1.98V are read as high.

Switch bounce is noise created at the input due to the mechanical contact not instantaneously connecting.

To debounce a switch you can achieve this using hardware or software (or a combination of both).

An effective hardware solution to switch debouncing is to apply a low pass filter across the output as this blocks the high frequencies of the switch bouncing.

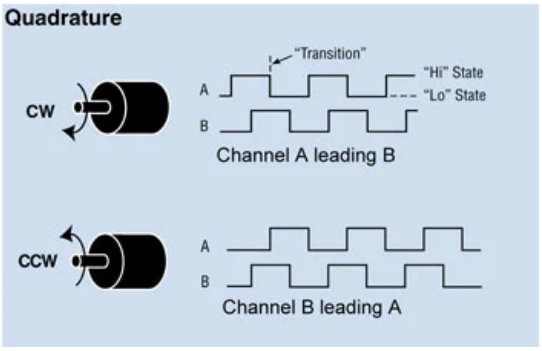


A software solution to switch debouncing is to temporarily disable the input when first activated. This will allow bounce to pass before reading from the input again.

A software solution is usually more inexpensive than a hardware solution as it does not require any additional components. However, a hardware solution is preferable when memory is at a premium.

If the switch bounce is particularly bad then both solutions may have to be used.

The rotary encoder has two output pins, they both produce a digital output when the encoder is turned. Depending on the direction that the encoder is turned then one channel will lead the other in outputting a response. From this the controller can determine which direction.



Piezo Speaker:

$$V_P = iZ$$
$$i = \frac{V_p}{Z}$$
$$i = \frac{3.3}{100}$$
$$i = 33mA$$

This is below the maximum pindrive of the Il Matto at 40mA. This means the piezo speaker can adequately drive the speaker.