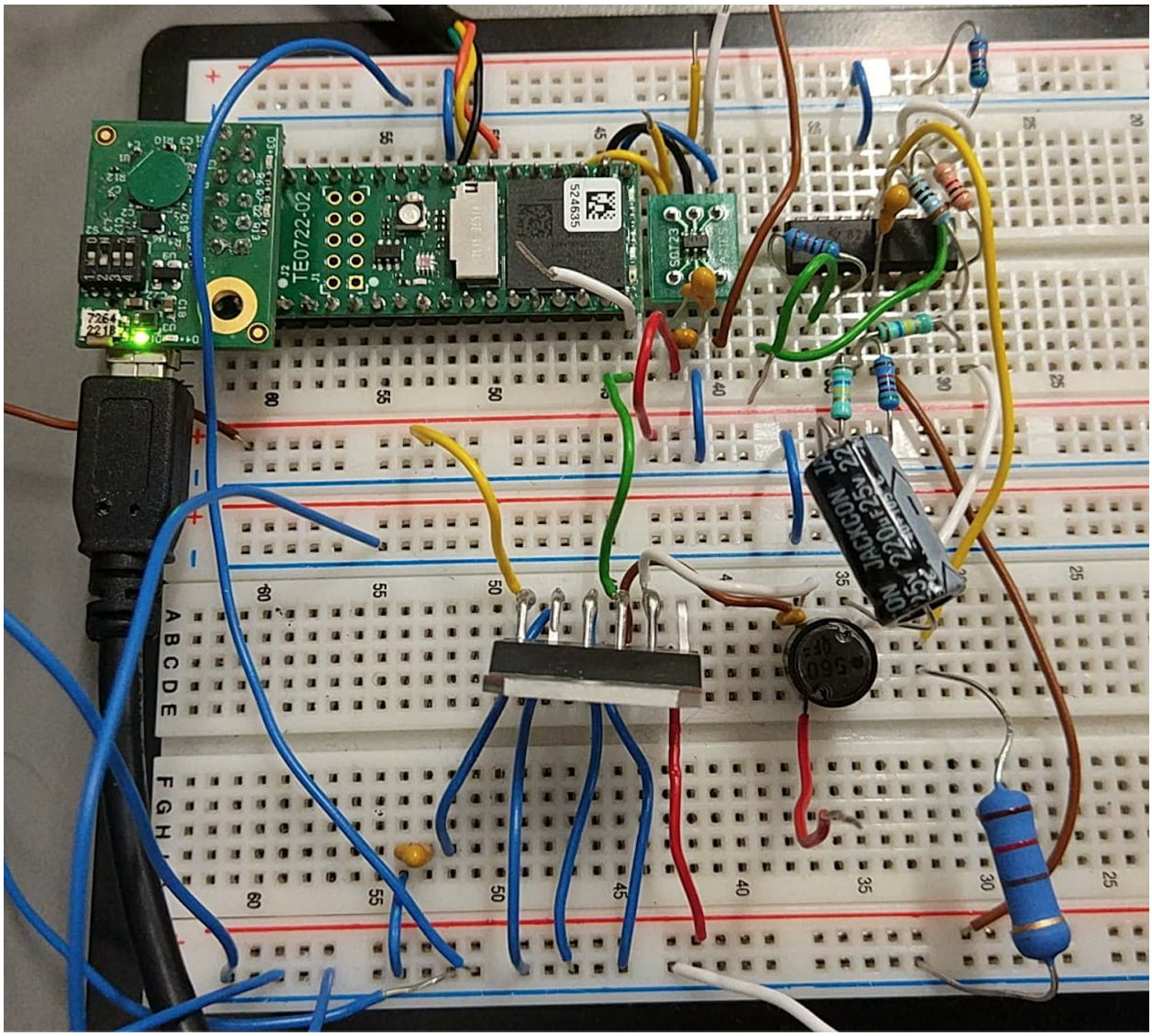
Robot Electronics - Final project

Group 2:
Thor Gunnlaugsson Jensen - thorj14@student.sdu.dk
Jorge Cebollada - joceb18@student.sdu.dk
Pauline Steiner - paste18@student.sdu.dk

January 8, 2019



Contents

1 Introduction

The aim of this final project is to create a system which works as a **signal generator**, creating a sinusoidal load-voltage or current where frequency, amplitude and offset should be controlled from a PC. Besides our FPGA some ADC, a full bridge and electronic components are used in order to get it. A general idea of the whole system is showed in figure 1.

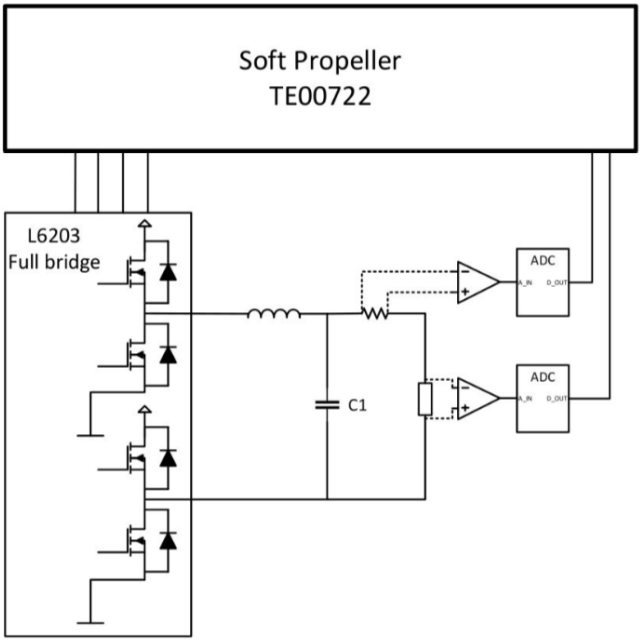


Figure 1: General idea of the target system

2 Component list

1. Trenz TE0722 "Soft propeller" with Xilinx Zynq-7010.
2. Trenz TE0790 XMOD FTDI JTAG Adapter.
3. ADC Ads7885.
4. Full bridge L6203.
5. Ad627 Differential Amplifier.
6. Low-pass filter (see table below).
7. Current measuring (see table below).

Number	Components	Value	Units	Use
x1	Resistance	1	Ω	Current measuring
x1	Resistance	1	kΩ	Current measuring
x1	Resistance	50	kΩ	Current measuring
x2	Resistance	165	kΩ	Differential amplifier (R1, R2)
x2	Resistance	50	kΩ	Differential amplifier (R3, R4)
x1	Coil	50	μH	Low-pass filter
x1	Capacitor	100	μF	Low-pass filter
x1	Capacitor	0,22	μF	Half bridge connections
x1	Capacitor	10	nF	Bootstrap in half bridge
x1	Power resistance	10	Ω	Load

3 System operation

In order to analyse clearly the system operation, two parts have to be separated: FPGA and electronic circuit.

3.1 FPGA

First of all, a general view inside the FPGA most important module (PL TOP) can be seen in figure 2.

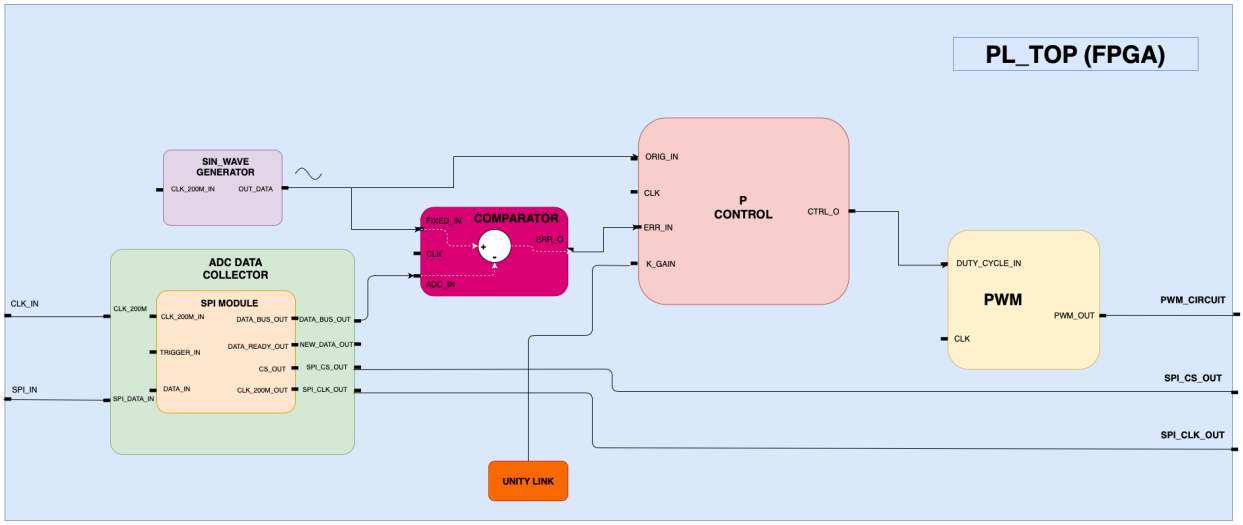


Figure 2: VHDL Modules inside PL TOP

3.1.1 ADC Data Collector and SPI

- **Description**

SPI is a common communication protocol used by many different devices. One unique benefit of SPI is the fact that data can be transferred without interruption. Devices communicating via SPI are in a master-slave relationship. **The master** is the controlling device (our **ADC**), while **the slave** (our **ADC Data Collector**) takes instruction from the master. In this case we have the simplest configuration of SPI: single master and single slave. The ADS7885 Interface Timing Diagram for a normal operation can be seen in figure 3.

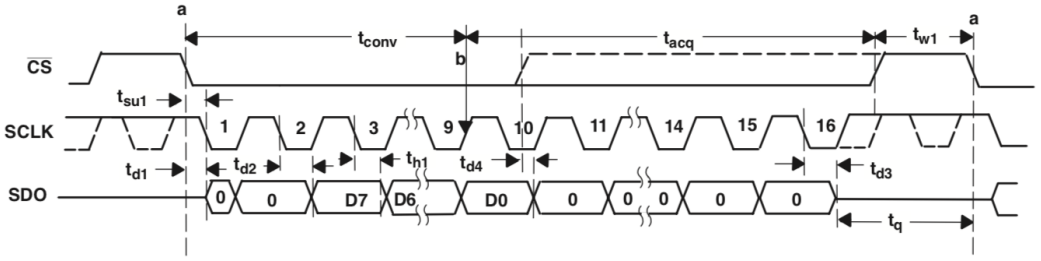


Figure 3: ADS7885 Interface Timing Diagram

- **State diagram of FSM**

In figure 4 the states diagram of the FSM from the implemented code is showed.

Each cycle on the ADC starts by the rising edge of the cs out which triggers the start of a cycle and changes the state from **'idle' to 'cs pulse'**.
This period is enough to sample the leading zeros which are ignored during the time it takes to sample these signals: 't cs pulse'. Next, the **'conv setup'** state does the conversion of the 8-bit data which finishes after waiting 't adc setup'.
After this period, the **'spi clk low'** module becomes active and goes back and forth with the **'spi clk high'** module until all the 8-bit data signals are successfully sampled.
After all that, the 4 lagging zeros are ignored in the **'busy state'** during the 'timer reg' period. When it finishes, the state goes back to the 'idle' state.

- **Test**

Unity link was used in order to check if the SPI and ADC collector modules were implemented correctly. Through *Unity link*, the data from ADC collector was read. The ADC input can fluctuate between 0 to 3.3 V. We tested several input values. As expected, the values read by the *Unity link* changed accordingly, as seen in the figure 5. The *'FF'* value corresponds to an input of 3.3 V and the '0' value corresponds to an input of 0 V.

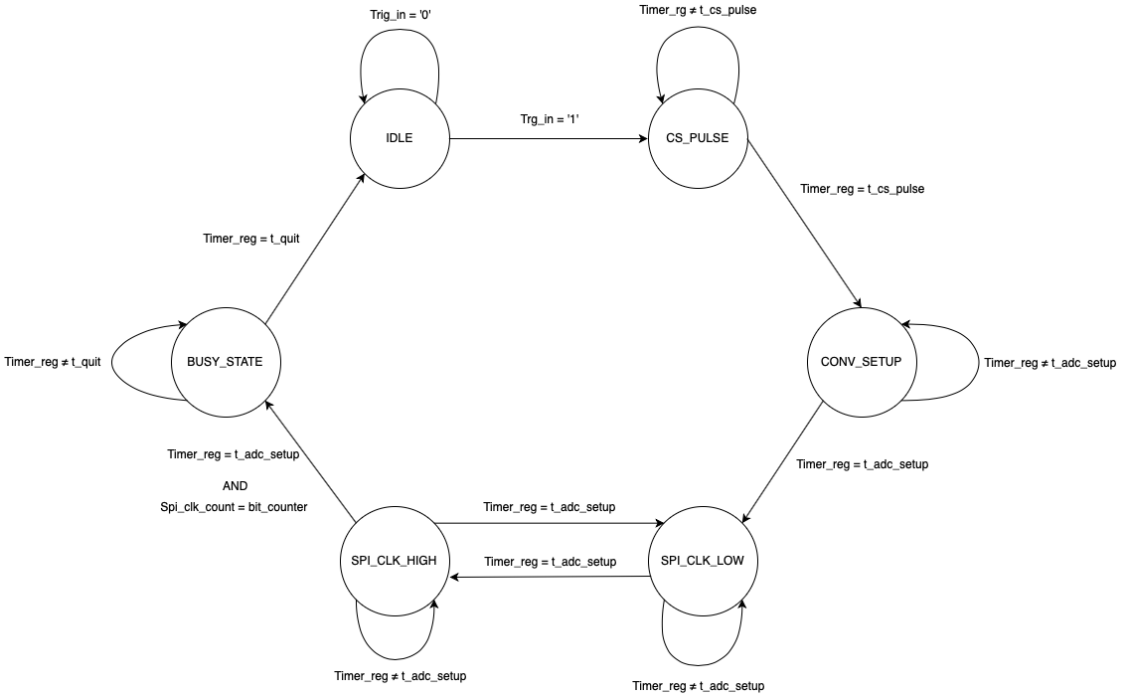


Figure 4: States diagram of the FSM

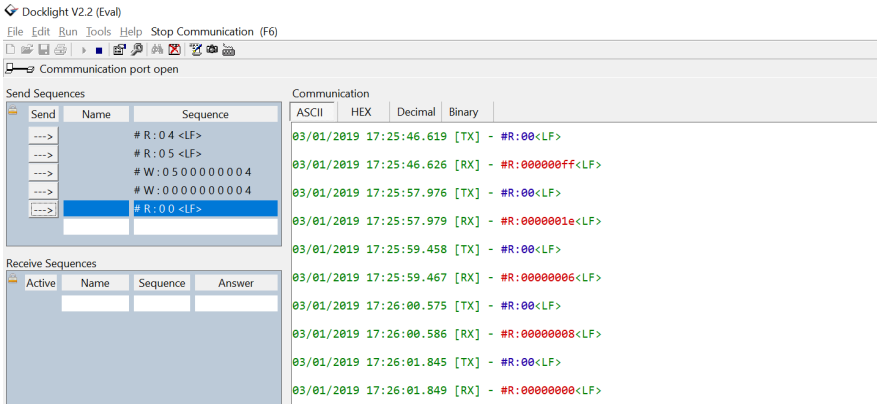


Figure 5: ADC data collector values checked through the Unity link

3.1.2 Sinusoidal wave generator

- **Description**

This module is creates a sinusoidal wave going from the integer value 0 to 255 in its basic form and is then being outputed to the PWM module. That makes the sinusodial controlling the PWM by changing the duty cycle.

- **Simulation**

We wanted to control the frequency of our sinusoidal wave through the *Unity link*. As seen in the figure 6, we simulate it. The variable freq-in corresponds to the value we are sending through *Unity link*. Here it is equal to 2, so the frequency of the signal will be divide by two, and the resulting period will be twice more important than the original signal. As

expected, we can observe a period that is two times the previous one. We have T-modif=320 ns while T-origin=160 ns.

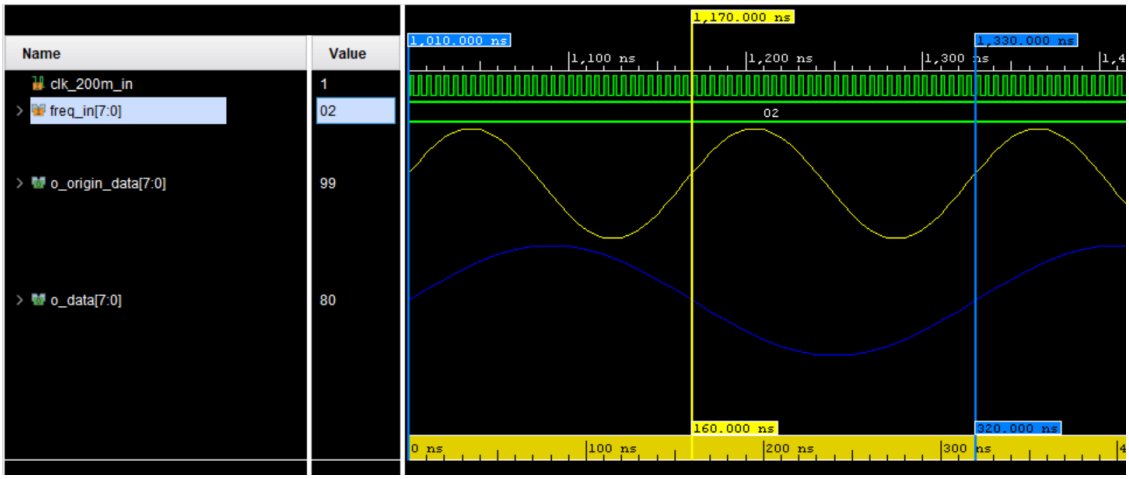


Figure 6: Frequency control in the Wave Generator

To control the amplitude of the signal, we also do it through *Unity link* with the variable ampl-bus that we have set up to 2 as well. With a 2 as input, the amplitude should double, as seen in the figure 7. The modify signal has a maximum amplitude of 1fe, that is indeed the double of the original amplitude which is ff.

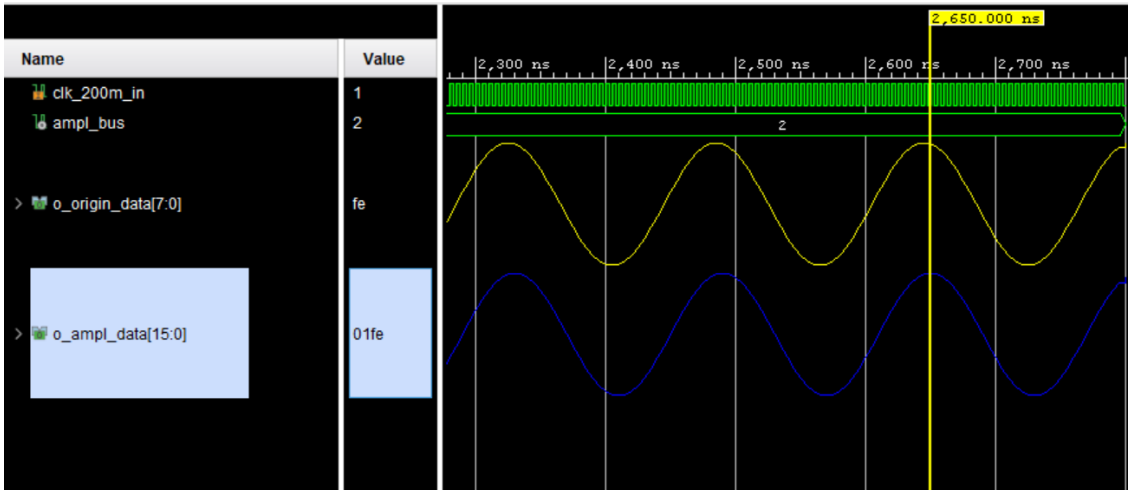


Figure 7: Amplitude control in Wave Generator

The sinosodial function was then connected to the duty cycle of a PWM module. The PWM signal was then led to a H-bridge fitted with a lowpass filter. The resulting signal can be seen in figure **??** which show that there are room for improvement within the anolog part of the system. One can almost imagine a sine wave with a little fantasy.

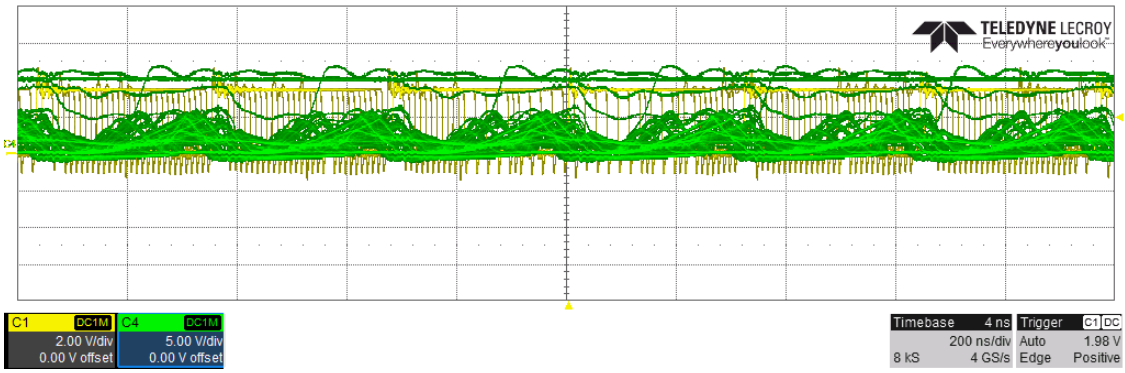


Figure 8: The effect of the sinusoidal wave input measured after the lowpassfilter.
??

3.1.3 P Control

A P-controller module was created in an attempt to make up for the errors within the designed system. The gain of the controller is controlled through Unity link. The P-controller module takes the error calculated from a module before it and tries to reduce said error to zero. The formula for a P-controller is seen in equation 1. p0 is the output without any errors in.

$$P_{out} = K * error + p0 \tag{1}$$

- **Simulation**

The P-controller was tested in simulation before released onto the FPGA. One of these simulatiosn can be seen in figure 9. The gain (k) is set to 2 and the error is 2 also giving the new $P_{out} = 9$ as seen in *ctrl_o*.

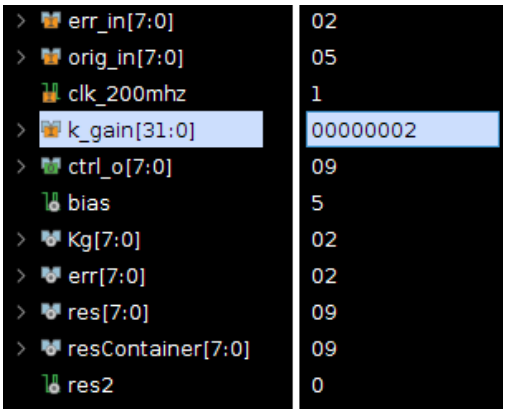


Figure 9: P Control simulation

3.1.4 PWM

As well as for the previous modules, we linked the input duty-cycle port of the PWM module to *Unity link*. So that any value could be given to the PWM module really easily through *Docklight* as seen in figure 11. In figure 10 the connections for testing is showed.

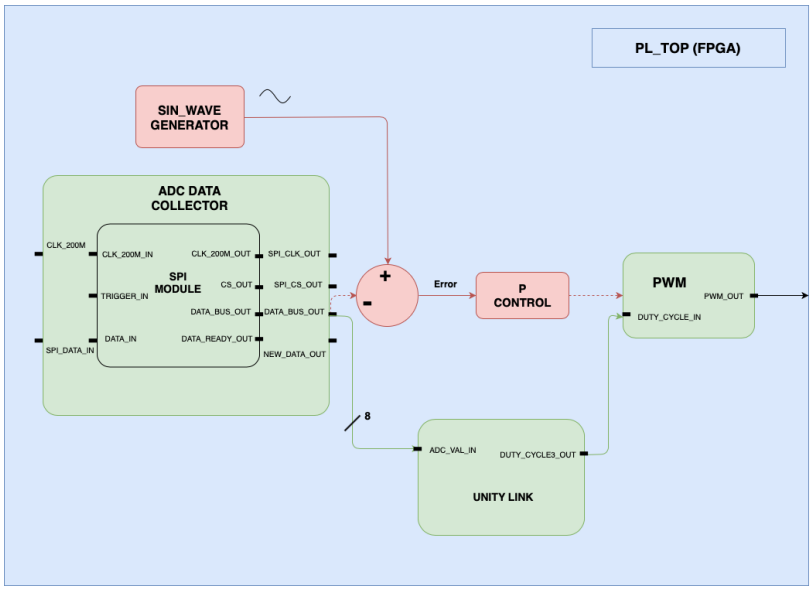


Figure 10: Test diagram

To know whether it works correctly we check the PWM output signal with a scope, as seen in figures 12 and 13. First it is sent the value *'FF'* in the eight bit vector that corresponds to a 100% duty-cycle, as *'11111111b'='FFx'*. And then the second sent value is '80' that corresponds to a 50% duty-cycle, as seen in figure 13.

```
05/01/2019 09:59:49.138 [TX] - #R:0C<LF>

05/01/2019 09:59:49.142 [RX] - #R:00000000<LF>

05/01/2019 09:59:53.130 [TX] - #W:0C000000Ff<LF>

05/01/2019 09:59:53.139 [RX] - #W<LF>
05/01/2019 10:03:57.156 [TX] - #W:0C00000080<LF>

05/01/2019 10:03:57.159 [RX] - #W<LF>
```

Figure 11: PWM Unity link Test

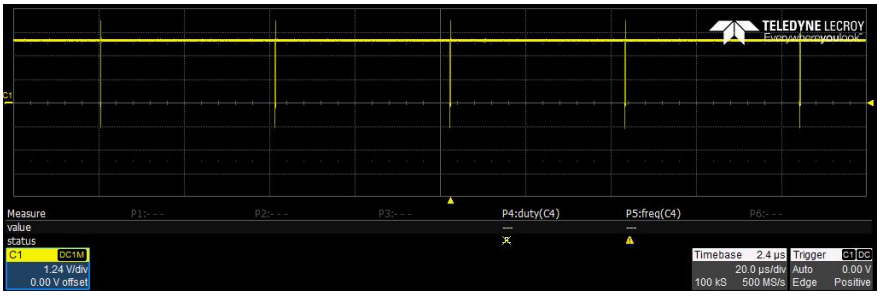


Figure 12: PWM output with a 100% duty-cycle

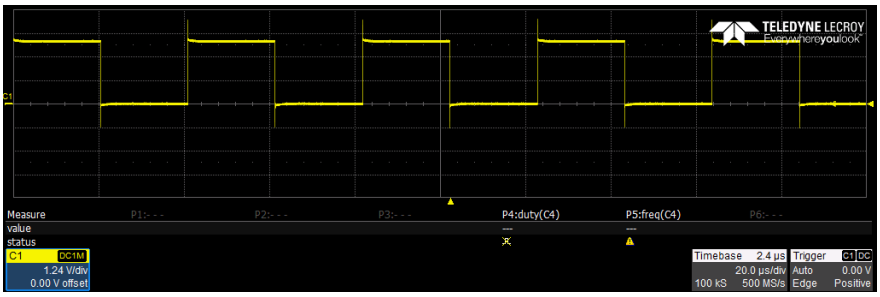


Figure 13: PWM output with a 50% duty-cycle

3.2 Analogue electronics

The output from the FPGA, which is also the output from the PWM goes into the *DMOS Full Bridge Driver*. The output from this driver goes through a low-pass filter in order to reduce the signal noise before go into the power resistance. The voltage in the power resistance as well as the current through it go into the ADCs, where are converted from analog to digital and sent again into the FPGA.
A diagram of all of these electronics can be seen in figure 14.

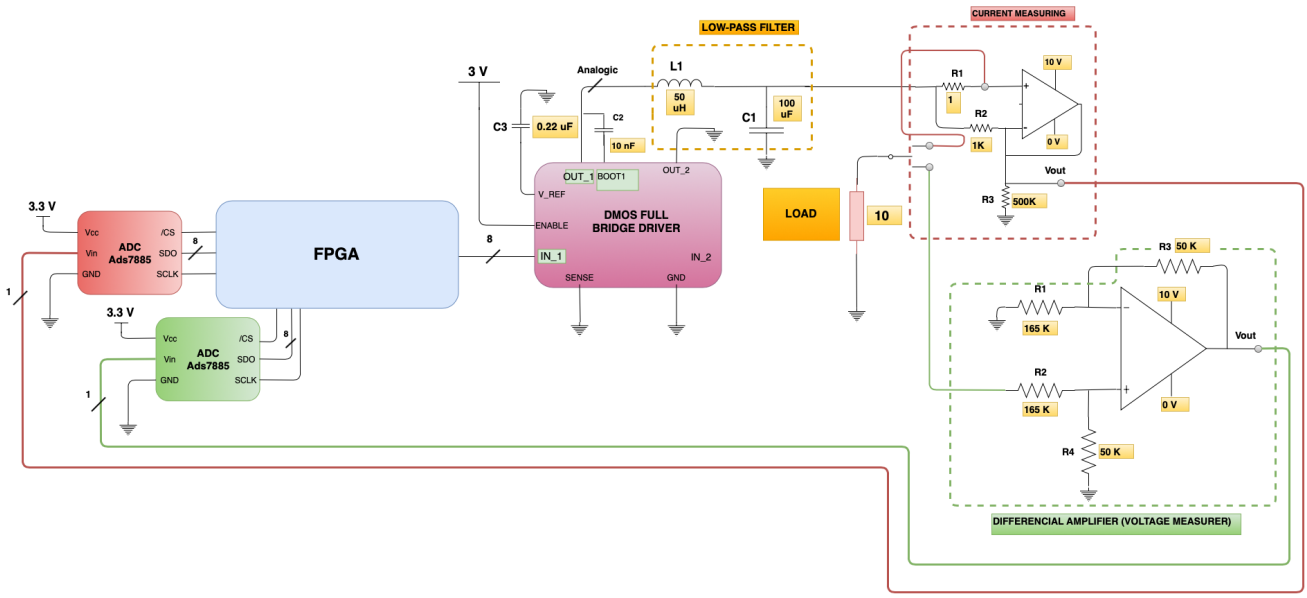


Figure 14: Electronics in the circuit

3.2.1 Analog to Digital Converter

An ADC Ads7885 figure 15 is used for converting the analog signal either voltage or amp measurement and turns it into a 8-bit signal which can be read by the FPGA.

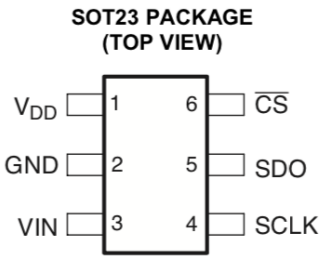


Figure 15: ADC inputs/outputs

The three oscilliscope seens in figure 16 is the output from the ADC with different voltages as input.

3.2.2 DMOS Full Bridge

We use a ***DMOS Full Bridge L6203*** for transforming the PWM signal in a controllable voltage/current which depends on the PWM duty cycle. In this case, **only one half-bridge** is used while the other half-bridge is configured as a constant connection to ground. This will create only positive voltages/current. The pin configuration can be seen in figure 17.

As a consequence of only use one half-bridge, **pins nº 1,6,7,8** are grounded. Also the **pin nº 10 (sense)** because we do not need that feedback. **Pin nº 9 (Reference voltage)** is connected with a 0,22 μF capacitor between it and ground in order to by-pass the internal reference voltage. **Pin nº 11 (enable)** is connected to 3,3 V (high) and then the transistors are enabled to be selectively driven by IN1 and IN2. **Pin nº 2 (Supply voltage)** is connected to 20 V. **Pin nº 5 (IN1)** receives the PWM signal and **pin nº 3 (OUT1)** sends the new signal to the low-pass filter. **Pin nº 4 (BOOT1)** is connected with a 10 nF bootstrap capacitor between it and OUT1 in order to ensure efficient driving of the upper POWER DMOS transistor.

The half-bridge output voltage depends on the outcome of the duty-cycle which we can control through *Unity link* and *Docklight*. We made several measurements as seen in the figures 18, 19 and 20.

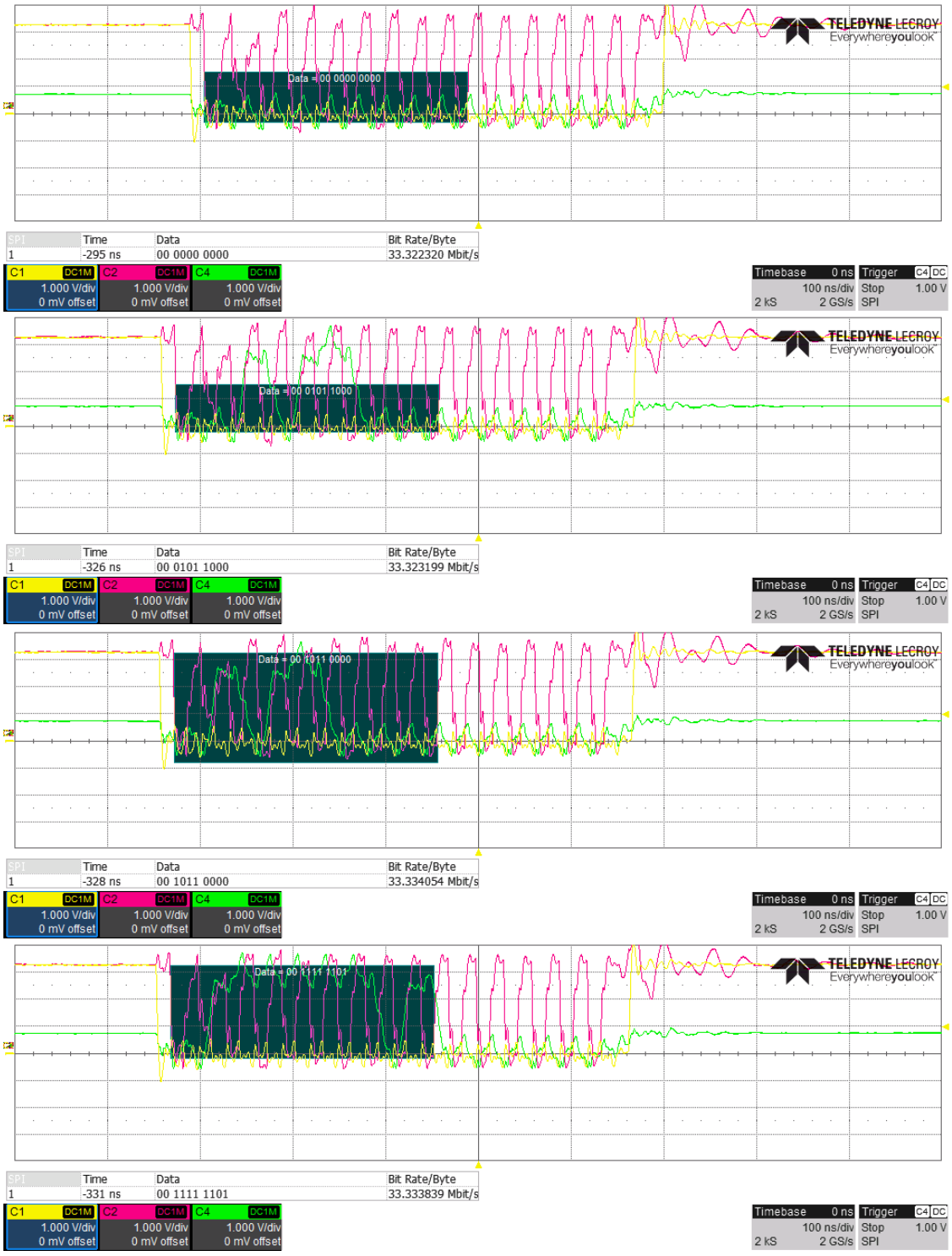


Figure 16: ADC output read from the oscillioscope with when appling different voltages. The relation between the bit(0-255) and voltage(0-3) is as it should be.

3.2.3 Low-pass filter

A lowpass filter was applied to the PWM signal coming from the H-bridge and placed just before the power resistor. The power resistor acts as a load on the system. The values of

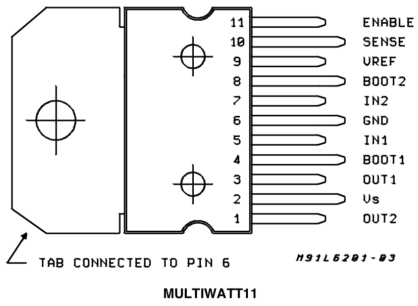


Figure 17: L6203 Pin mapping

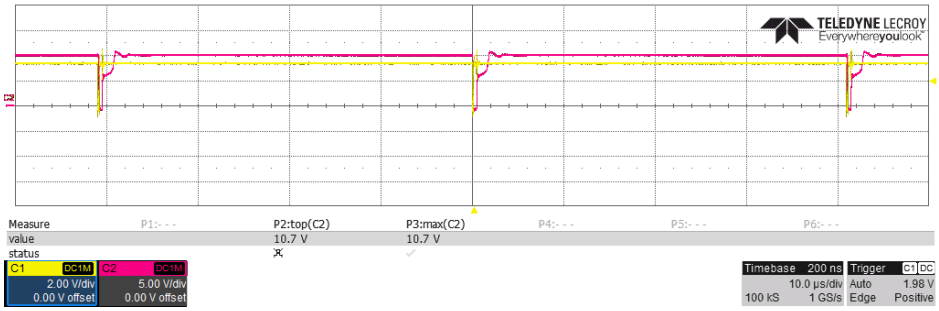


Figure 18: Half-bridge output with a 100% duty-cycle

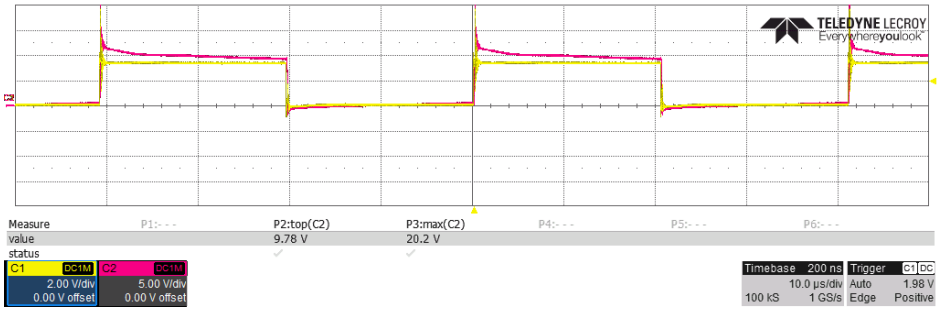


Figure 19: Half-bridge output with a 50% duty-cycle

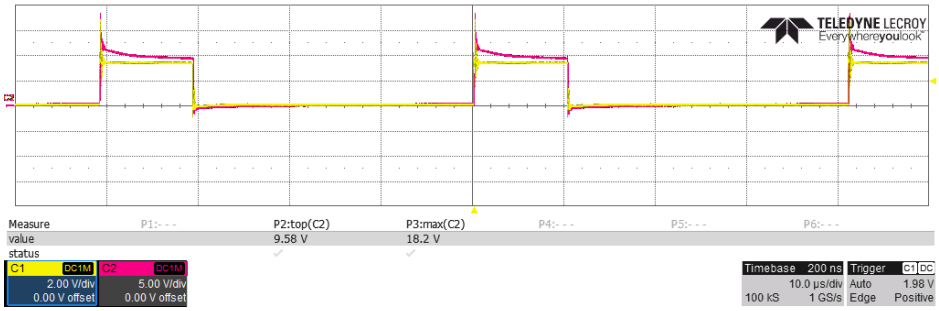


Figure 20: Half-bridge output with a 25% duty-cycle

the components used can be seen in section 2: Components.

First consideration was the frequency of the PWM signal which is around 25 kHz as seen in the figure **??**. The reason for this consideration was to not get below the cutoff frequency

which was calculated using equation 2.

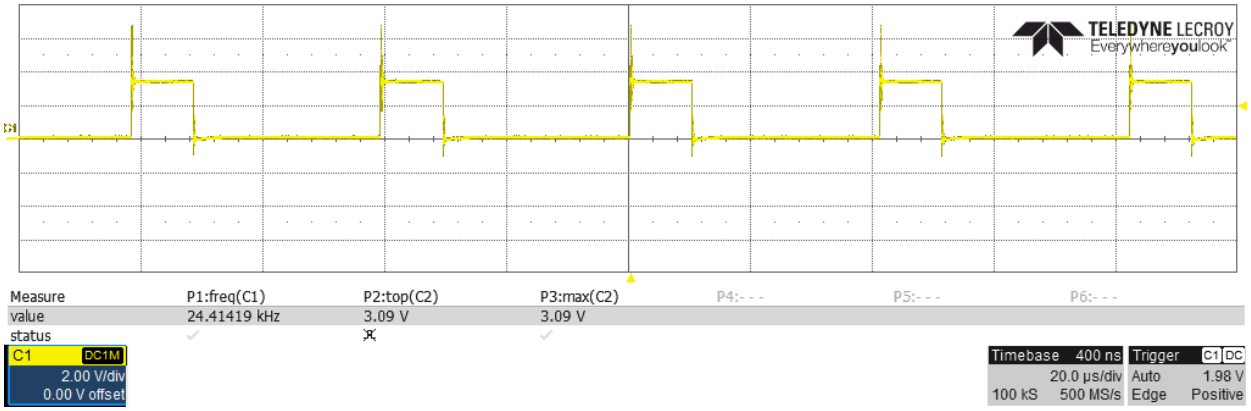


Figure 21: *PWM signal from the FPGA. From the measurement table one sees the signal has a frequency of 24kHz.*

By holding the cutoff frequincy similar to the PWM signals frequency and using the coil given from the teachers ($50\mu H$) a capacitance of approximately $87\mu F$. For the final circuit a capacitor of $200\mu F$ was used because it had a more desirable outcome to the signal. The increment of the capacitor took the PWM signal further away from the calculated cutoff frequency thereby "smoothing out" the voltage.

$$F_{cutoff} = \frac{1}{2 * \pi * \sqrt{coil \times capacitor}} \tag{2}$$

In figures 22 and 23 the signal after passing through the filter is showed, where yellow is the duty-cycle and pink is the low-pass output. It can be observed that with a smaller duty-cycle, the output signal voltage from the low-pass is also smaller. In figure 24 a duty cycle versus voltage graphic is showed.

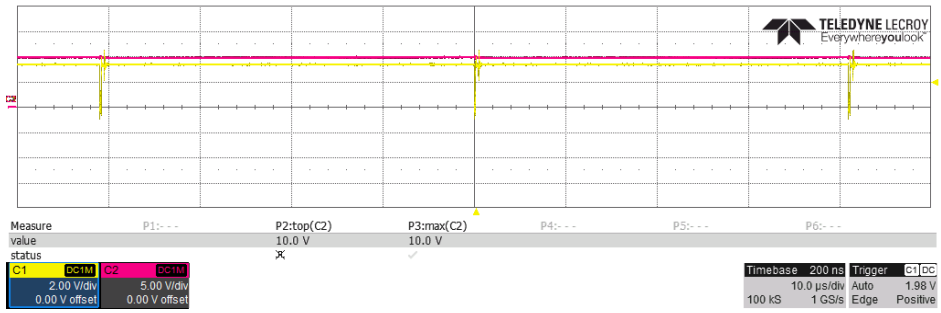


Figure 22: Low-pass output with a 100% duty-cycle

After the lowpass was applied the voltage was measure and compared to the fixed duty-cycle and the result can be seen in figure 24. The relation between dutycycle and voltage looks linear.

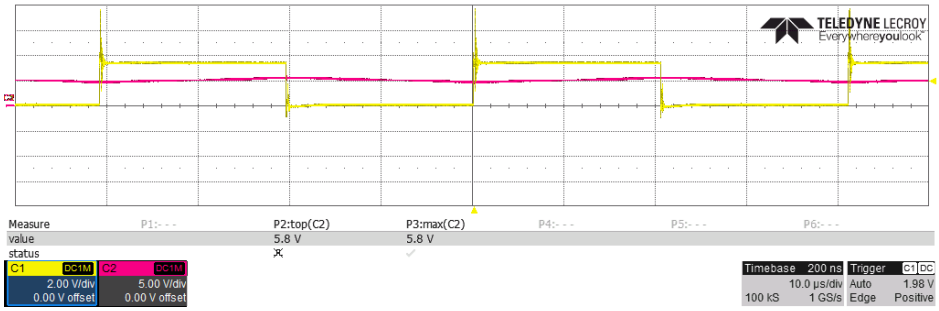


Figure 23: Low-pass output with a 50% duty-cycle

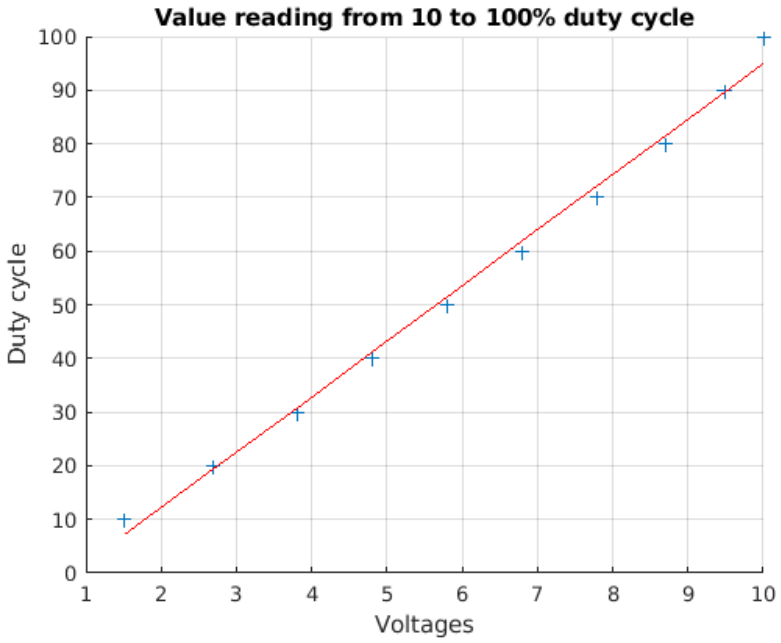


Figure 24: The data are marked with blue crosses and a linear approximation is marked with the red line.

3.2.4 Differential Amplifier - Voltage

Two different differential amplifiers are created with the LM324 chip which contains 4 operational amplifiers. The voltage difference is then scaled down from 10V to 3V since 3V is the ADCs maximum input voltage.

- **Differential amplifier equations**

The equations below are used to figure out the size of the resistors used for the differential amplifier.

$$V_{out} = (R_3/R_1) * (V_2 - V_1) \tag{3}$$

$$R_1 = R_2 \tag{4}$$

$$R_3 = R_4 \tag{5}$$

We assume that we have 10 V in the entrance and we want to get 3 V as V_{out}.

One input is then grounded: $V_1 = 0$

$$3 = (R_3/R_1) * (10 - 0) \tag{6}$$

$$R_1 * 3 = R_3 * 10 \tag{7}$$

$$R_1 = R_3 * (10/3) \tag{8}$$

$$R_1 = R_3 * 3,3 \tag{9}$$

Respecting this relation between the two resistors, the values are then chosen to be very high in order to get the lower current as possible at the entrance.

$$R_1 = R_2 = 165k\Omega \tag{10}$$

$$R_3 = R_4 = 50k\Omega \tag{11}$$

- **Circuit**

The differential amplifier is constructed following the circuit showed in figure 25

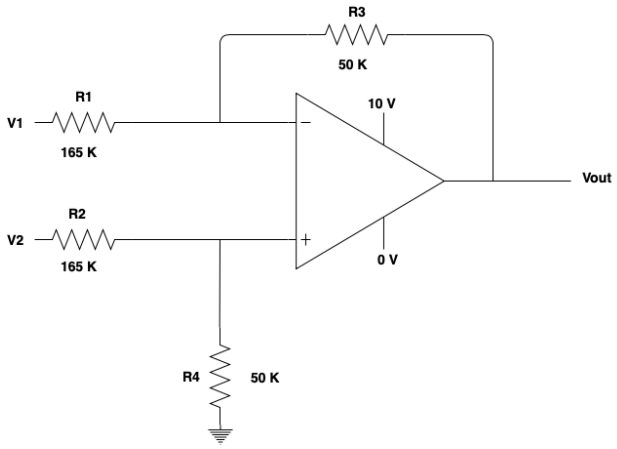


Figure 25: Differential Amplifier Circuit

- **Test**

The differential amplifier is then tested as seen in figure 26. When 10 volts are applied to the load the amplifier outputs 3 volt. So the amplifier is working correctly.

A test of the circuit without the controller connected was constructed were the duty cycle would be controlled by the user and the voltage across the load was measured by the system itself. An external multimeter measuring the same voltage would act as ground truth. The results and difference can be seen in table 1.

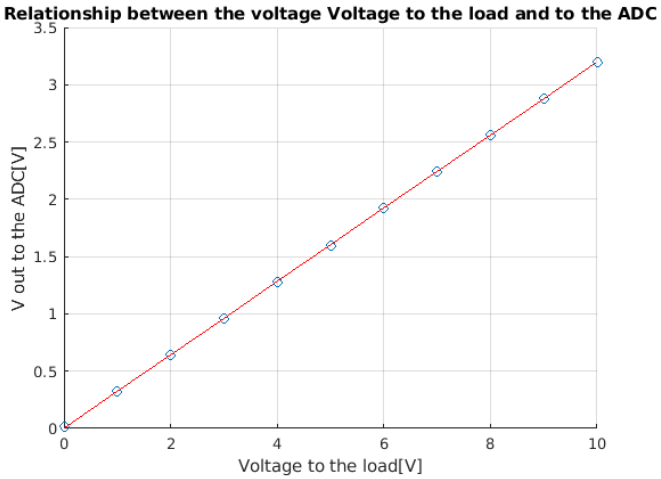


Figure 26: Voltages relation before and after the differential amplifier

Duty cycle	Duty cycle [decimal]	ADC [decimal]	Difference
10	25.5	29	-3.5
20	51	21	30
30	76.5	81	-4.5
40	102	112	-10
50	127.5	146	-18.5
60	153	161	-8
70	178.5	188	-9.5
80	204	218	-14
90	229.5	244	-14.5
100	255	255	0

Table 1: This table shows the desired duty cycle and the actual duty cycle. Both are converted to decimal for reading convenience.

3.2.5 Differential Amplifier - Current

- **Description**

In order to measure the current consumed by the load "a current measuring" has been performed. For building it, a operational amplifier and some resistors are used, as seen in figure 14.

The presence of negative feedback means that we can apply the virtual short approximation, i.e., we can assume that the voltage at the inverting input terminal is equal to the supply voltage minus (supply current R1). Since the upper terminal of both R1 and R2 is tied to the supply voltage, the virtual-short assumption tells us that an equal voltage appears across both of these resistors, and consequently the current through R2 is equal to the current through R1 divided by the ratio of R2 to R1. The current through R3 is more or less equal to the current through R2.
As a consequence, this is the expression for V_{out}:

$$V_{out} = \frac{R_1 * R_3}{R_2} * I_{load} \tag{12}$$

- **Test**

A curve of the amps limited by the system and resulting voltage from the differential amplifier can be seen in figure 27. These measurement was made with no load applied. These result could however not be recreated with a load applied to the system.

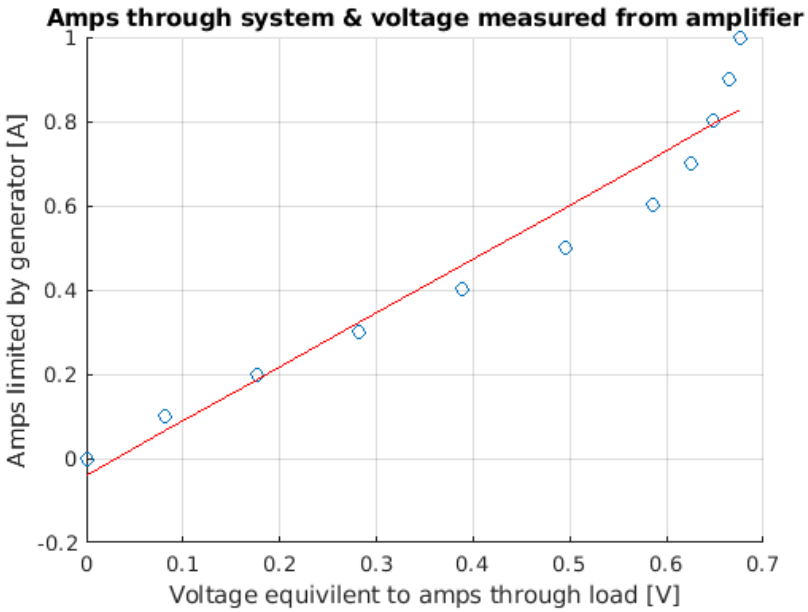


Figure 27: Voltages relation before and after the differential amplifier

4 Conclusion

A sinusodial function capable to control a PWM signal was coded and implemented which allowed the used to change the frequency and amplitude of the the sine wave. However this implementation performed poorly and would need a lot of optimization before working properly.

Two differential amplifiers was created using operation amplifiers. The amplifier tasked with measure voltages work great. The differential amplifier tasked with measuring current did not work as planned when a load was applied which was discovered to late in the project period to fix.

A lowpass filter was applied to a H-bridge and the resulting output voltage could be controlled by user input. The user could also extract the measure voltage from an ADC connected to a FPGA through a unity link.

The loop was closed and a control module was coded tested and verified in simulation but was however not implemented in the finished system due to poor time management from the authors and random complexity. Therefor a fully autonomous control loop was not build and verified.

The presented system is by far fully optimized with its components which has been left for future work to finish.

5 References

1. **SPI Communication (3/1/2019)**: http://www.circuitbasics.com/basics-of-the-spi-commu
2. **L6203 Datasheet (4/1/2019)**: http://users.ece.utexas.edu/~valvano/Datasheets/L6203.pdf
3. **Ads7885 Datasheet (4/1/2019)**: http://www.ti.com/lit/ds/symlink/ads7884.pdf
4. **AD627 Datasheet (6/1/2019)**: https://www.analog.com/media/en/technical-documentatio data-sheets/AD627.pdf
5. **How to Monitor Current with an Op-Amp, a BJT, and Three Resistors (7/1/2019):**
https://www.allaboutcircuits.com/technical-articles/how-to-monitor-current-with-an?fbclid=IwAR2CvX3Ef60abSmBKXeW_7wgicqVrxnIUnG869tb9Vn_mB7pH_M_3FM27Dk/