Programmable Analog Input Node (PAIN) Master Semester Thesis, Nicolo D'Anna.

ETH Zürich June 19, 2017

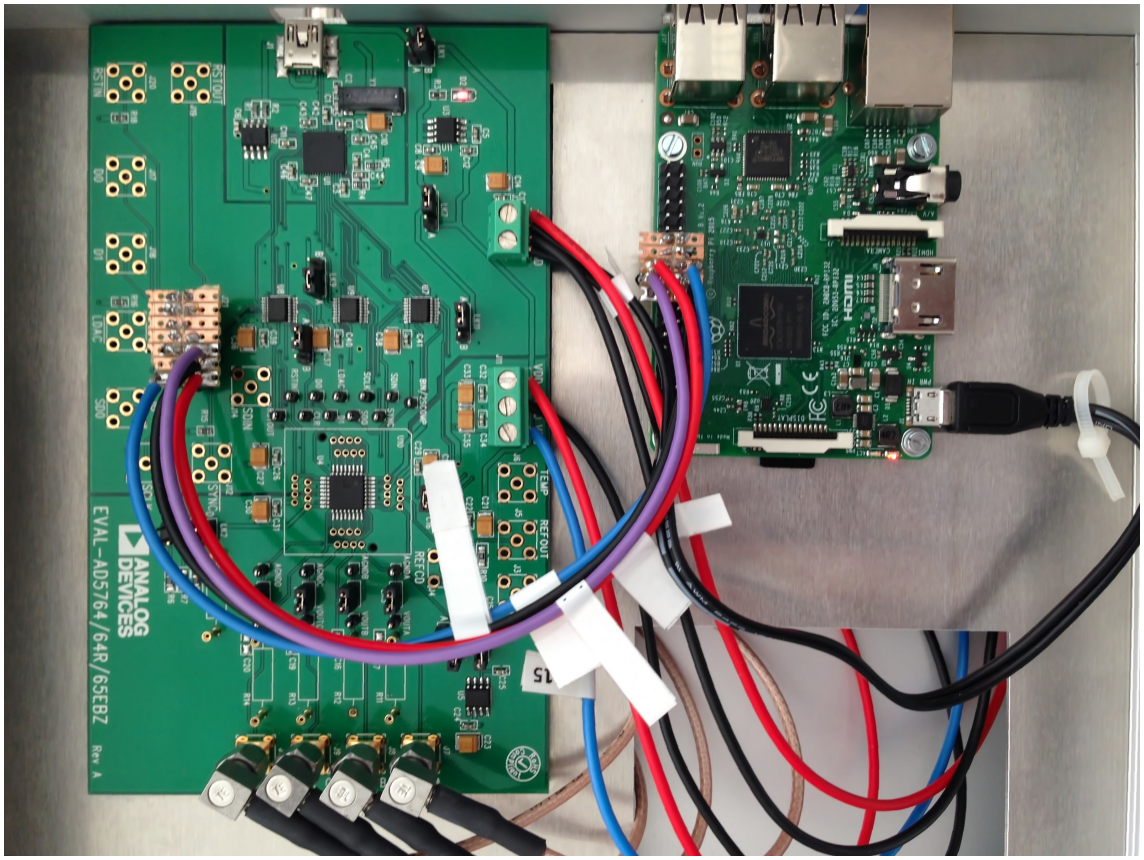

Figure 1: *Image of the wire connections between the Pi and Evaluation board.*

Contents

1 Introduction

The aim of this work is to build a box and a software that allow to easily create and control four electric voltages. This is done using a small computer called Raspberry Pi 3 and a Digital to Analog Converter (DAC) circuit. The Raspberry Pi 3 can communicate via a four wires bus called SPI (Serial Peripheral Interface bus) to the DAC and thus control its outputs. More precisely here an Evaluation Board EVAL-AD5764 is used that allows access and programation of the DAC. In the end measurements are given to characterise the system.

2 Installation

2.1 Raspberry Pi 3 Connection to EVAL-AD5764 The Raspberry Pi 3 needs a 5V input via its micro USB input port. This is done by using a micro USB cable that is stripped on the other end such that the ground and 5V wire can be separated and connected to the DGND and +5V pin respectively (figure 1 and 2), where a second set of wires will eventually be connected coming from an external 5V source.

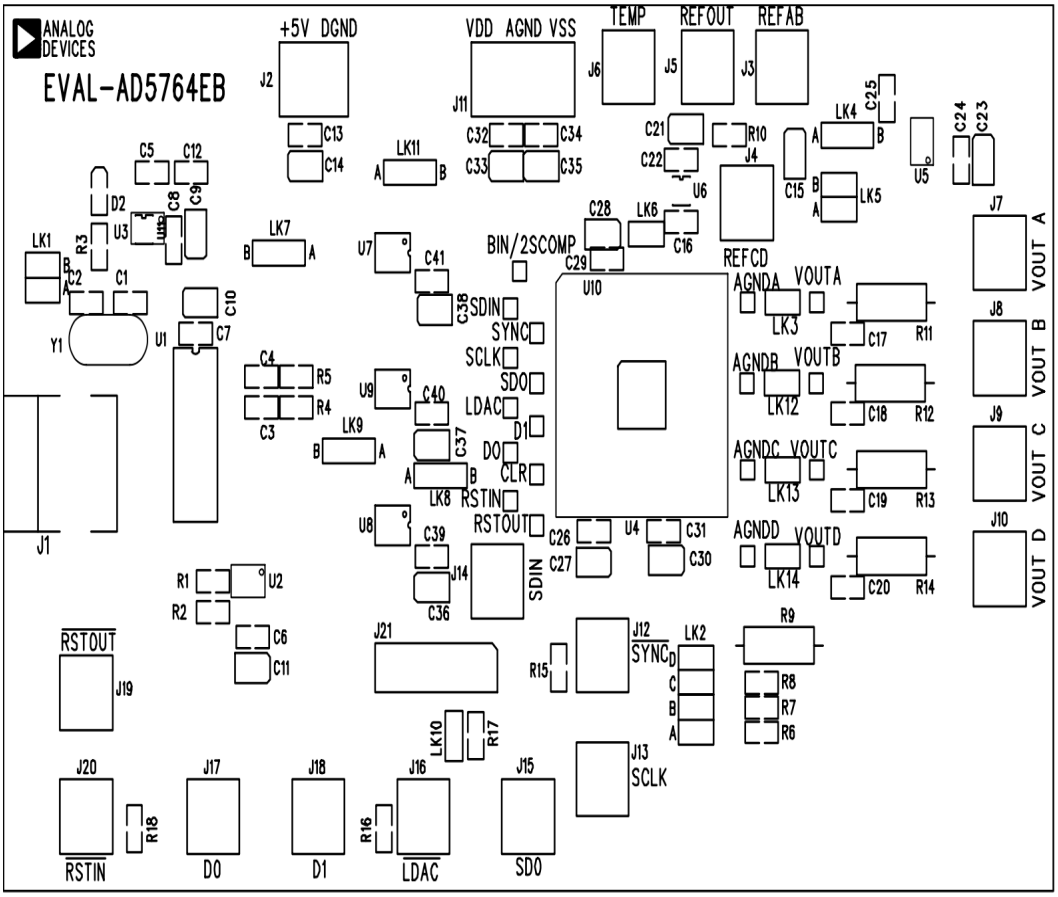

Figure 2: *Image of the Evaluation board components as seen from above, with the names of the all components. Taken from [1].*

Four more wires need to be connected from the Pi's output pins to the Evaluation Board's input J21 connectors (figure 3 and figure 4);
CLK: Pi GPIO 11 Pin 23 to DAC J21 2.
SDIN: Pi GPIO 09 Pin 21 to DAC J21 3.
$\overline{\text{SYNC}}$: Pi GPIO 08 Pin 24 to DAC J21 1.
$\overline{\text{LDAC}}$: Pi GPIO 25 Pin 22 to DAC J21 5.

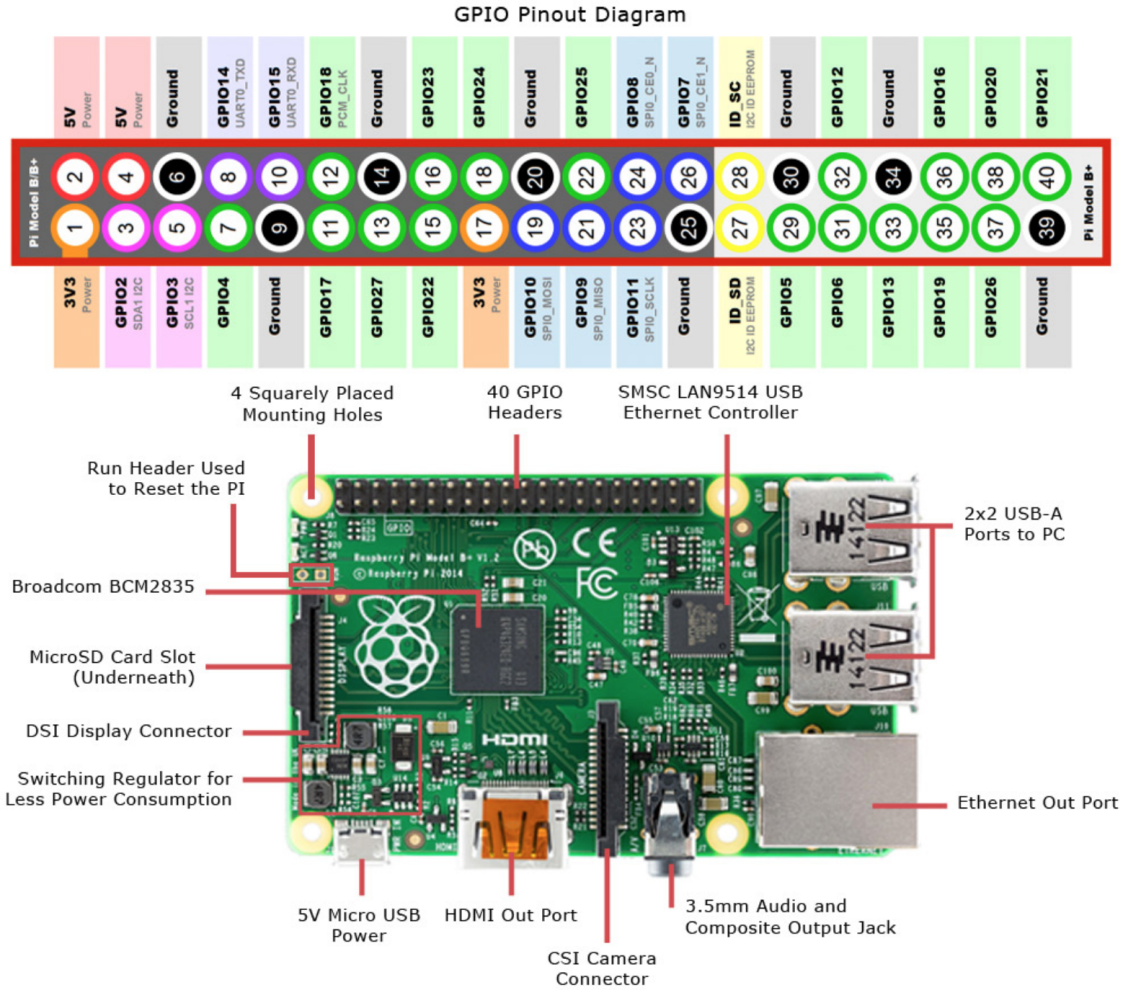

Figure 3: *Image of the Raspberry Pi 3 from [2]; on the bottom is a picture of the where the GPIO pins can be seen. On the top is a description of the GPIO pins function and number.*

2.2 Evaluation Board Setup The Evaluation Board needs two power supplies connected to it, one of 5V connected with a ground and a +5V wire to the DGND and +5V pin respectively (the same as the Pi), and one of 15 V connected with a +15V, a ground and a -15V wire to the VDD, AGND and VSS pin respectively (see figure 2) .

Connector J21 Pin Descriptions

Table 2. Connector J21[1] Pin Configuration

13	**11**	**9**	**7**	**5**	**3**	**1**
14	12	10	8	6	4	2

[1] LK11 must be in Position A to enable the use of J21.

Table 3. Connector J21 Pin Descriptions

Pin No.	Function
1	$\overline{\text{SYNC}}$
2	SCLK
3	SDIN
4	SDO
5	$\overline{\text{LDAC}}$
6	$\overline{\text{CLR}}$
7	D0
8	D1
9	$\overline{\text{RSTOUT}}$
10	$\overline{\text{RSTIN}}$
11	BIN/$\overline{\text{2sCOMP}}$
12	DGND
13	DGND
14	DGND

Figure 4: *Image of J21 component on the Evaluation Board. The location of the J21 can be seen on figure 2. The top part of the image shows the J21 pin nomenclature and the bottom part shows the corresponding functions.*

The Evaluation Board also has 14 links and switches options that need to be in the following positions to control the DAC from the Raspberry Pi;

Link No.	Option	Function
LK1	B	Power supply selection
LK2	C	Short circuit resistor value selection
LK3	on	Connect AGNDA to 0 V
LK4	B	Select reference source
LK5	A,B	Connect reference source to REFAB and/or REFCD
LK6	on	TEMP pin selection
LK7	B	Select supply voltage value for AD5764, U7, U8, U9
LK8	A	Select state of $\overline{\text{LDAC}}$
LK9	A	Select state of BIN/$\overline{\text{2sCOMP}}$ pin
LK10	removed	Select state of $\overline{\text{CLR}}$
LK11	A	select Eval Board control source (PC or J21)
LK12	on	Connect AGNDB to 0 V
LK13	on	Connect AGNDC to 0 V
LK14	on	Connect AGNDD to 0 V

2.3 Raspberry Pi 3 Software and Communication Setup Create a Static IP on the Pi (connect a keyboard and screen to do so):

```
sudo systemctl enable systemd-networkd.service
sudo nano /etc/systemd/network/25-wired.network
```

Then write into the file:

```
[Match]
Name=eth0

[Network]
Address=10.0.0.200/24
Gateway=10.0.0.1
```

End writing by closing with Ctrl+x. And restart the Raspberry Pi.

Then on a Linux PC connected to the Pi using the ethernet port, make the computer interface static using Microsoft Windows "view network connections", then right click "properties", then select "Internet Protocol Version 4" and click "Properties", then set the static IP to the following values:

```
 IP=10.0.0.1
 SUBNET MASK=255.255.255.0
```

The Pi can now be accessed by writing:

```
ssh tiqi@10.0.0.200
```

To be able to access the Pi from the PC without having to give a password each time, add an SSH key; first generate a new key on the PC:

```
ssh-keygen -t rsa -C tiqi@10.0.0.200
```

Save it in the default location by pressing enter.

Then enter the wanted passphrase.

Now in the .ssh directory there are two files, one is your private key: id_rsa. And the other is the public one you put on the Pi: id_rsa.pub

On the home directory on Pi create:

```
mkdir ssh
```

And to copy the key to the Raspberry Pi:

```
cat ~/.ssh/id_rsa.pub | ssh tiqi@10.0.0.200 'cat >> .ssh/authorized_keys'
```

Now each time the Pi is turned on the key has to be added from the PC's terminal:

```
eval $(ssh-agent)
ssh-add
```

and enter the passphrase. The Pi is then accessed by:

```
ssh tiqi@10.0.0.200
```

In order for the PAIN box using the PAIN program to work a few things have to be installed on a new Raspberry Pi. This is done the following way:
Using the a Windows Command Processor terminal (python wheels is installed on it) create a folder called wheels:

```
mkdir C:\scratch\pain\wheels
```

Where you then download the upgrades and packages that will be sent to the Pi:

```
pip download  -d "C:\scratch\pain\wheels" spidev
pip download  -d "C:\scratch\pain\wheels" RPI.GPIO
```

Then do on the Git Bash terminal:

```
scp * tiqi@10.0.0.200/home/tiqi
```

Once this is done connect to the Pi and do:

```
sudo pip install spidev-3.2.tar.gz
sudo pip install RPi.GPIO-0.6.7.tar.gz
```

Note that the name of the files might be different.

Enable SPI on archlinux:

```
sudo nano /boot/config.txt
```

Where you write;

```
device_tree_param=spi=on
```

After that reboot the Pi, and check that it works by looking if the files are there with:

```
ls /dev/spi*
```

Use mkdir to create a ”code” folder in the PI.

Finally to upload the python file from the PC terminal when being in the file’s directory use:

```
scp /*.py tiqi@10.0.0.200:/home/tiqi/code/
```

and to run the code from the Pi use:

```
sudo python ./file_name.py
```

2.4 Communication The pulses outputted to the DAC are shown in figure 5. They have to be in the right order; SCLK starts high and SDIN comes on SCLK (serial clock) falling edge (this corresponds to spi.mode(3) in spidev package), $\overline{\text{LDAC}}$ (load DAC) has to be high before SDIN (serial data input) and brought high after SDIN to upload the DAC, and $\overline{\text{SYNC}}$ (synchronise) also has to be first high then low during the input and then high again and it is set by the spi.mode.

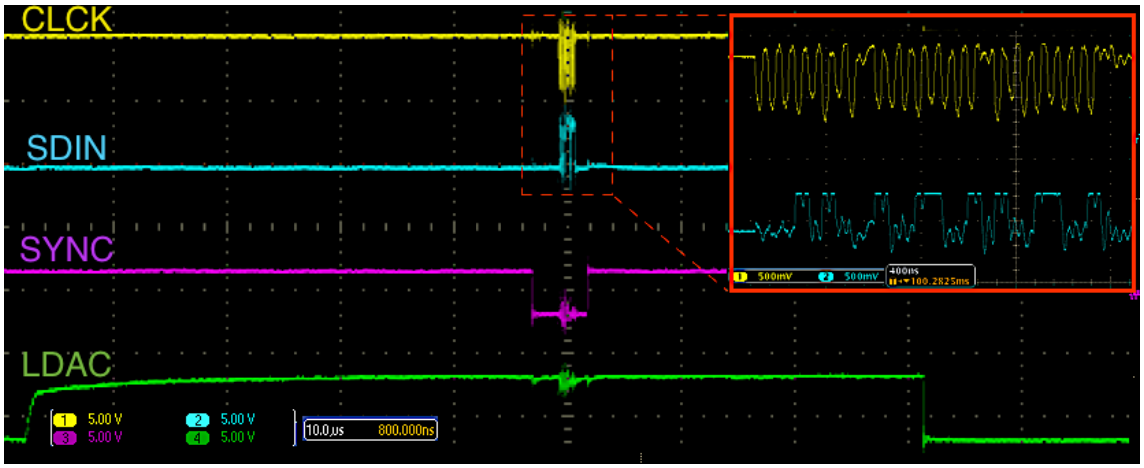

Figure 5: *Pulses sent to the DAC. Four signals are needed; one serial clock signal (SCLK) containing 24 pulses, one input word (SDIN) made up of 24 bits that come on the clock falling edges, one synchronisation signal (SYNC) that is brought low when the clock and input are being used and one input loading signal (LDAC) that is brought high when all the previous signals are on.*

The data input is coded in a 24 bit word, with the 8 most significant bits (MSB) coding for the register and operation choice, and the 16 least significant bits coding the output voltage in hexadecimal value. The 24 bits are sent in 3 groups of 8 bits because the Raspberry Pi can only sent up to 8 bits in one go.

The code for the Raspberry Pi can be found on Gitlab in the project called 'pain' in the folder called 'finalCode'. The code itself is python code called 'PainCode.py'.

3 Characterisation

The characterisations of interest are how fast can a DAC change its voltage and what is the precision with which the voltage can be controlled.

The time it takes for a DAC to change its voltage is a function of the step size (difference of voltage). Figure 7 shows the dependence of the delay in function of the size of a step. More in detail the definition of the delay is shown in the figure 6: it is the time from when the DAC receives the $\overline{\text{LDAC}}$ signal prompting it to update the voltage to when 95% of the step is accomplished. The minimum value of 0.9 μs corresponds to what the manufacturer claims.

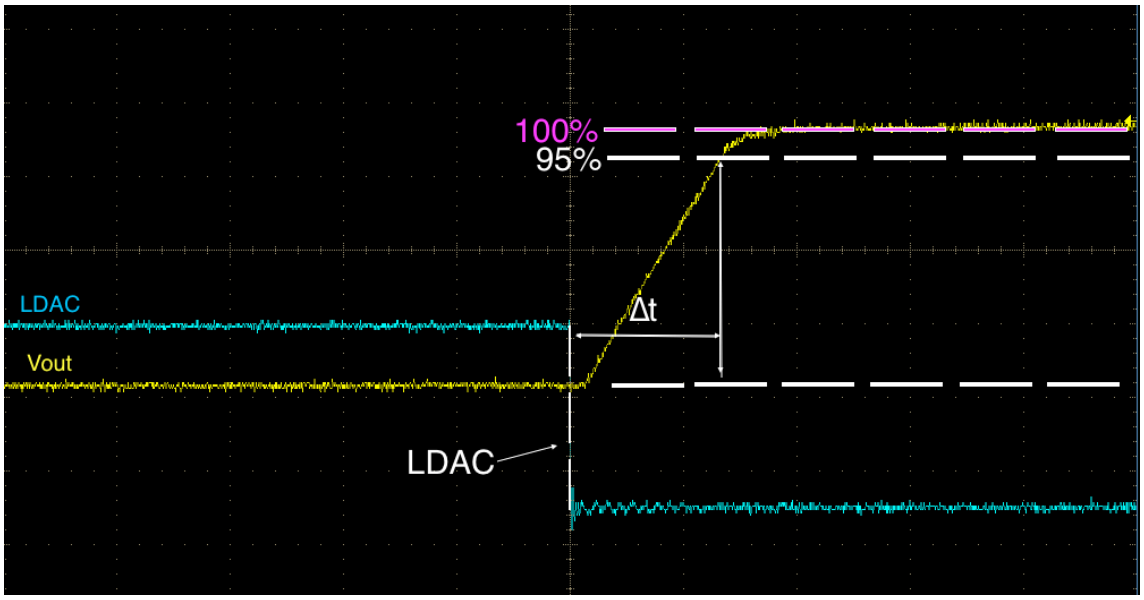

Figure 6: *Definition used to measure the delay Δt for the next figure (7); the delay is taken to be the time it takes for the voltage to reach 95% of the programmed change.*

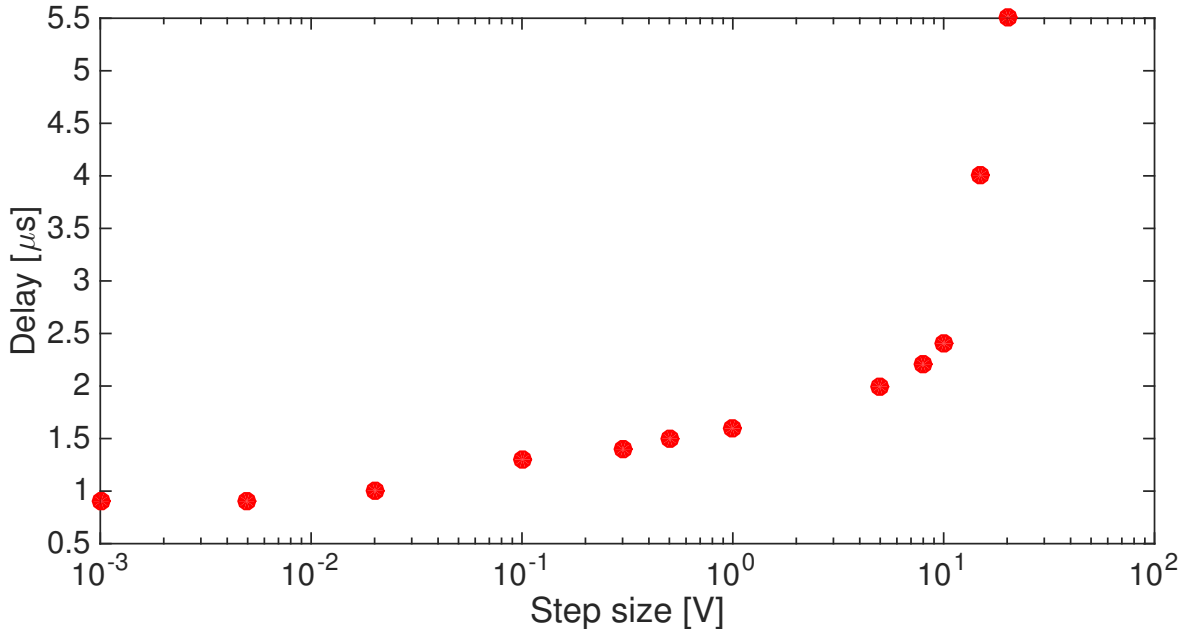

Figure 7: *Delay in function of the size of a step; the maximum possible step is of 20 V and the minimum of 0.03 mV. However the delay does not diminish anymore for steps smaller than 1 mV.*

The theoretical minimum step possible is given by the way the voltage is coded in hexadecimals and is of 0.03 mV. In practice one can see on figure 8 that it is effectively the case, but that at those scales the noise becomes significant. In fact the noise has a standard deviation of 0.25 mV which is almost the same as the precision of the DAC.

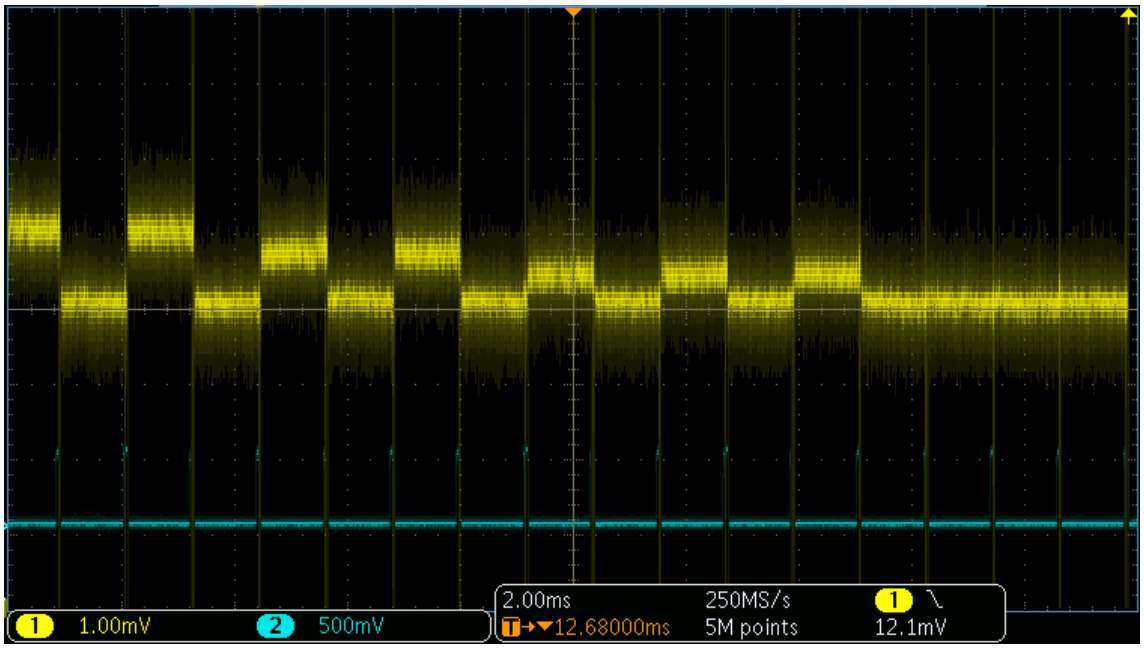

Figure 8: *The output shown here is of the smallest possible voltage steps in yellow and the* $\overline{LDAC}$ *signal in blue . The smallest step is the result of a 0.2 mV input in the program, then moving to the left the increment is of 0.1 mV per step until 0.6 mV then the last two steps increase of 0.2 mV, to reach the biggest step at 1 mV. Inputs smaller than 0.2 mV do not create any result as is the case on the far right of the figure.*

A bigger issue can be seen in figure 9, where an unwanted disordered pulse happens 30 μs before a change in voltage. This phenomena which lasts about 8 μs with a peak to peak maximum of around 15 mV always happens when changing voltage. The same thing but on a smaller time scale happens when the voltage changes and can be seen in figure 10, it is about 200 ns long and has the same peak to peak amplitude.

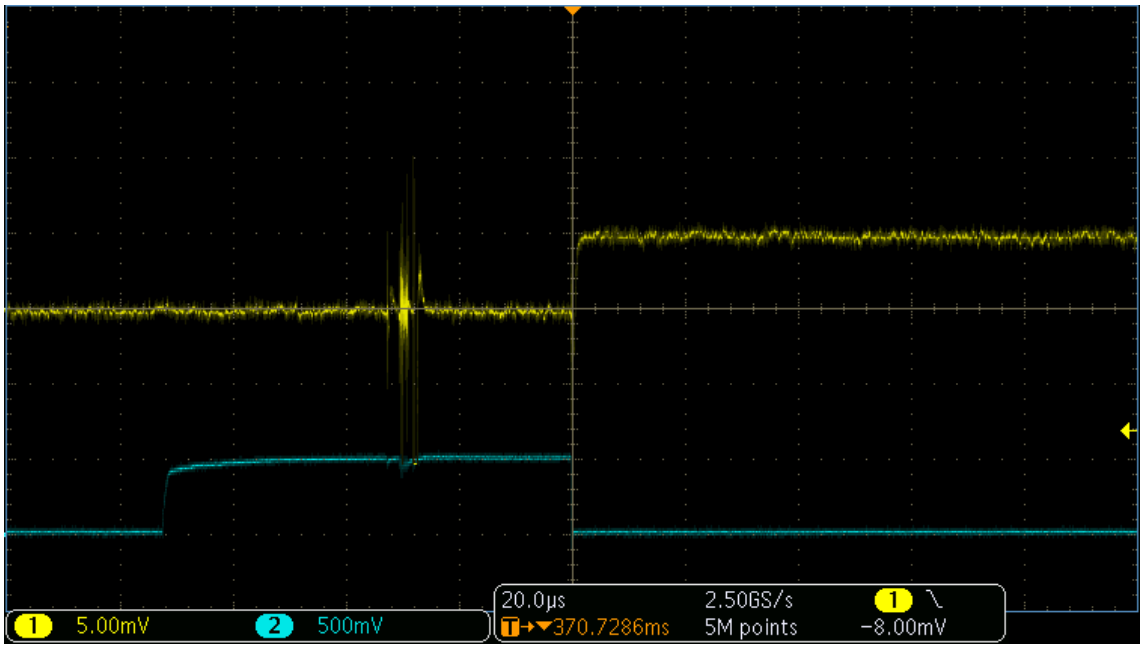

Figure 9: *In yellow the output voltage and in blue* $\overline{LDAC}$*. One can see the noise burst that happens about 30* μs *before the change in voltage.*

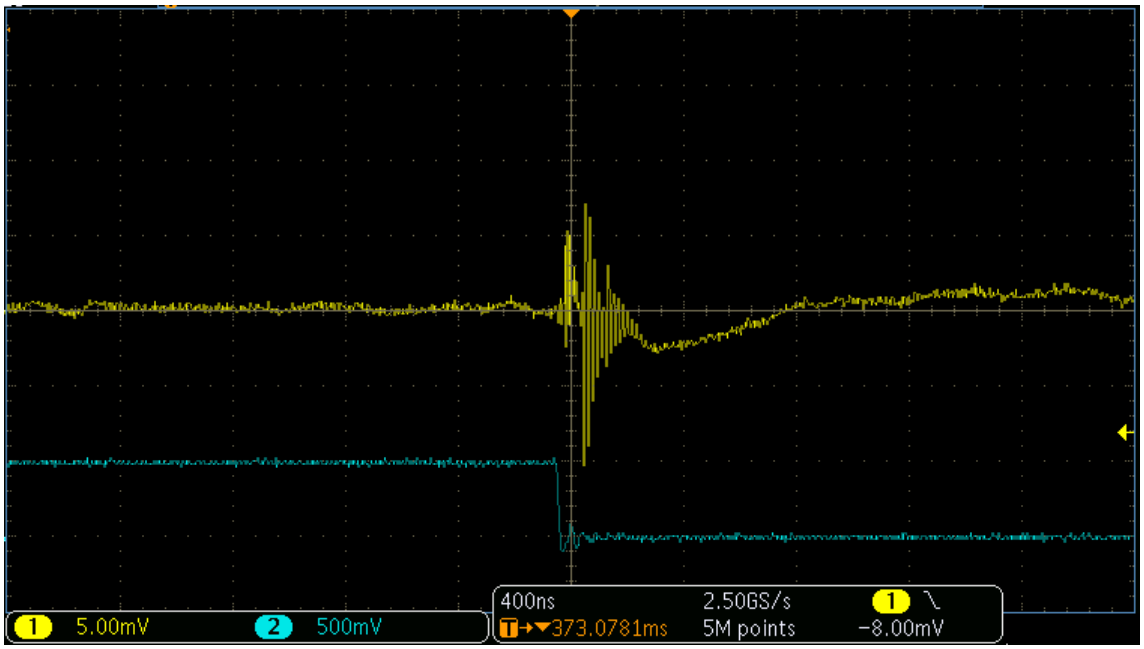

Figure 10: *Here we see precisely what happens at the transition between two outputs, with in yellow the output voltage and in blue* $\overline{LDAC}$*. A burst of noise is characteristic of the transition.*

4 Conclusion

The setup presented here where a Raspberry Pi 3 is used to control an Evaluation Board EVAL-AD5764 to create an analog voltage output with a 10V to -10V range works well with an output precision of 0.3 mV, a noise of 0.1 mV and the rate of change from a voltage to another can reach 1 μs.

Many experimental devices, e.g. the high-current source that drives magnetic field coils, offer analog voltage inputs for control. This project is the basis for further development in the direction of arbitrarily shaped analog voltage ramps with the presented hardware.

References

[1] Technical Manual EVAL-AD5764EB, Analog Devices Inc., Available at: http://www.analog.com/media/en/technical-documentation/evaluation-documentation/106758267AD5764EB_0.pdf, Accessed 19.06.2017.

[2] Raspberry Pi 3 Pinout Image, Available at: https://i.stack.imgur.com/sVvsB.jpg, Accessed 19.06.2017.