Real-Time Remote Control of Droege High Voltage Power Supplies

Mesut Çalışkan

The University of Chicago, USA

Abstract

High Voltage Power Supplies for use with Multi-Wire Proportional Chambers (MWPC), otherwise known as Droege, are still useful today in applications which require positive or negative high voltage output (DC) with low current, fast trip, or good transient response. However, local control of the output lacks precision and is inefficient in terms of time, while the traditional method of remote controlling is problematic since the connectors, the cables, and the remote control modules no longer exits. The purpose of this research is to develop a new method of real-time remote controlling Droege over the internet such that a specific voltage or an increasing/decreasing order of voltages with exact increments can be supplied with high precision for the desired duration of time. This new method has been developed using a micro-controller board (Arduino Uno), 10-bit analog-to-digital and 12-bit digital-to-analog converters, and a Raspberry Pi and by writing the appropriate firmware and software. As a result, Droege can be controlled from anywhere via the internet with a range of output of 0 to 7.5 kV with ± 2.19 V uncertainty. In addition to this, the true output voltage and current can be remotely monitored with a range of 0 to 5.0 kV with ± 4.96 V uncertainty and with a range of 0 to 500 μA with ± 122 nA uncertainty. The desired voltage and time values can be set at each use without any modification of the code. Lastly, the method also enables the user to set an upper limit to the voltage and current as a safety measure.

1 Background and Introduction

ES-7109 and ES-7125 are dual high voltage power supplies (negative and positive respectively) for use with multi-wire proportional chambers (MWPC). They are also known as "Droege" due to the inventor of these power supplies, Thomas F. Droege. In addition to being a high voltage power supply for use with MWPC due to its abilities such as high internal resistance, which limits the amount of energy that can be delivered to a chamber under spark conditions, fast trip, and good transient response, Droege is also a convenient high voltage power supply for applications which require low current, and it can output voltages in the range of $0 - 7.5$ kV. [1, 2]

There are two ways to control the output of Droege: local, and remote. Local control is problematic because it requires one to turn a potentiometer knob in order to adjust the level of voltage output, which is time consuming. In addition to this, local control comes with a high uncertainty. In applications which a series of high voltages must be outputted, such as creating an IV curve, outputting the desired voltage in each step is hard.

Remote controlling is possible via applying a potential difference to two of the pins on the back panel of Droege. This method works in a differential way and the potential difference corresponds to the output voltage. This way of remote controlling can be achieved by using a DC power supply, the appropriate 8-pin connector, Burndy GOB 12-88 PNE, and the cable. However, this requires the user to be at the same place as Droege. Also, this method requires wire connection between the DC power supply and Droege. Furthermore, the 8-pin connector and the cable no longer exist Therefore, there is a need of a new way of remote-controlling these devices.

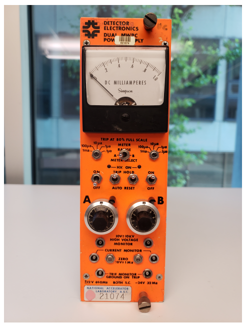


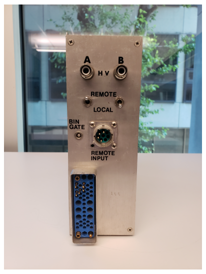
Figure 1: The Front Panel of Droege

It should be noted that a useful method of remote controlling has been developed by Jean-François Caron (University of Houston), and his colleagues by using a micro controller. [3]

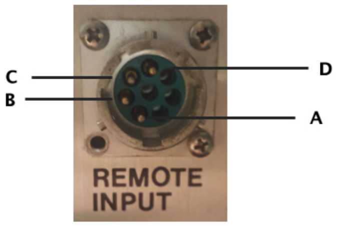
The purpose of this paper is to describe the development of a new method of real-time remote controlling Droege over the internet such that a specific voltage or an increasing/decreasing order of voltages with exact increments can be supplied with high precision for the desired duration of time. This new method has been developed using a micro-controller board (Arduino Uno), 12-bit analog-to-digital and digital-to-analog converters, and a Raspberry Pi and by writing the appropriate firmware and software.

2 Method

On the back panel of Droege, there are two channels of HV outputs (named A and B), two switches for swapping between local and remote control, a bin gate, connections to the NIM crate, and the remote control input as seen in Figure 2a.



(a) The Components of the Back Panel



(b) Pins of the Remote Input

Figure 2: The Back Panel of Droege

The remote control input has four pins, labeled A, B, C, and D as seen in Figure 2b.

The pins A and B are pins are for HV channel A, and the rest are for channel B. The mechanism of remote controlling works based on the difference of voltages applied on the pins. To explain this mechanism, let us assume voltages V_A and V_B has been applied to pins A and B respectively. ($V_A, V_B \geq 0$) Then, the output of channel A, V_1, is given by the following equation:

$$V_1 = 2000(V_A - V_B)$$

It should be noted that the pins A and C are the ones to which the higher (relative to the other pin for that channel) voltage should be applied. For example, if $V_A \leq V_B$, then V_1 would be equal to 0 V.

The previous paragraph shows that if the required voltages that are applied on the pins can be controlled remotely, and if these voltages are applied on the pins using the suitable cables and the connector, the output of Droege can be controlled remotely.

2.1 Hardware

Since the cable and the connector do not exist anymore, a cable with the appropriate pins has been built. For this, two AMP 6630-1-ND female pins were soldered on to a half-meter of shielded cable. Two solid male tips (22AWG) were soldered on to the other side of the cable.

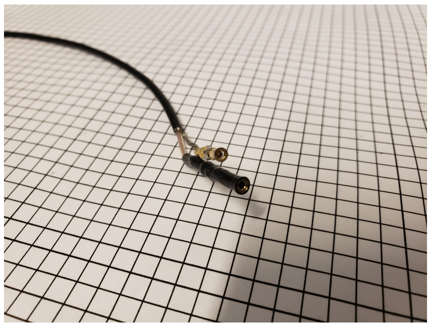


Figure 3: Connector Cable with AMO 6630-1-ND Pins

In order to create the voltage applied to the pins, I have decided to use a micro-controller board called Arduino UNO. [4] Arduino UNO has both digital and analog input pins and digital output pins. The range of voltage for both the inputs and outputs is $0 - 5$ V. The output pins can use pulse width modulation to imitate a voltage in between $0 - 5$ V by creating a square wave with 5 V of amplitude. But this means they cannot actually create a true analog signal with a voltage value in between $0 - 5$ V. Therefore, I have decided to use a digital-to-analog converter (DAC) so that a true analog signal could be created. For this purpose, MCP4725 12-bit DAC was used. [5] Arduino is controlled by using its own language based on C++. Using the right libraries the desired voltage can be created by DAC based on the command put into Arduino. Then, this voltage was applied on to the pins using the cable mentioned above.

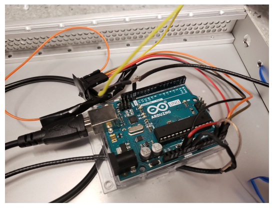
MCP4725 has been found to be suitable for our purpose because 12-bit of resolution corresponds to increments of 2.5 V, and this is a relatively good resolution considering our HV output range of $0-7.5$ kV. On the other hand, Arduino has been chosen due to it being:

easy to programme, relatively reliable since it does not have an operating system, compact, and low in cost.

Droege has a lemo output on the front panel for monitoring the output voltage. Also, Arduino has built-in 10-bit analog-to-digital converters (ADC). Two solid male tips (22AWG) have been connected to one side of a lemo cable and this cable was used to get the voltage monitor reading into Arduino. The resolution corresponds to increments of 5.0 V. Droege has a current monitor as well. The output of this monitor is read with the same technique as the voltage monitor described above.

In order to control, send code, or receive data from Arduino remotely, Arduino will be connected to a Raspberry Pi 3 Model B (the Pi) and remote connection to the Pi will be achieved via using SSH Protocol (Secure Shell).

The final part of hardware was putting all the components together and relieving strain. For this, an empty NIM box was assembled as seen in Figure 4b. This box is called “The Remote Control Unit.” First, DAC and Arduino were connected using stranded 22AWG jumper cables as seen in Figure 4a. Arduino and DAC were put inside the control unit and secured by double sided tape. The Pi has been left outside the control unit so that the only things need to be dedicated to Droege are Arduino and DAC. This allows the Pi to be taken away and be used for other purposes when Droege is not in use. Then, the Type A side of a female USB Type A to male USB Type B cable was put on the front panel by creating a rectangular opening on the front panel with a drill. The male USB Type B connector was connected to Arduino. The cable coming from the Pi was connected to the USB connector on the front panel of the control unit as seen in Figure 4b. Then, a circular hole was created by using a drill and a female lemo connector was put on the front panel. This connector was connected to Arduino inside the control unit, and a lemo cable was put in between the connector and the voltage monitor of Droege. Afterwards, the same process was repeated for the current monitor as well. Then, the Pi was placed the NIM crate. The final setting of Droege, the control unit and the Pi can be seen in Figure 4b.


(a) Connection of DAC and Arduino

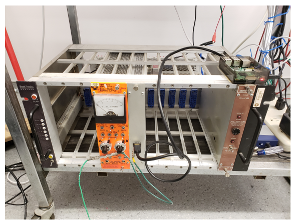

(b) Droege, NIM Box and the Pi

Figure 4: Droege, Remote Control Unit, and Pi on the NIM Crate

2.2 Software and Process

The purpose of the software written was that the user can determine the output voltage of the Droege, the time, any limitations on the voltage, whether the output will be constant or an increasing ramp, etc. without any modification on the code. So that the user can control the Droege, remotely from anywhere with an internet connection, easily by just putting in values to the command line. Also, another purpose is that the user can receive data from Droege regarding the value of the output voltage as well as a time-stamp associated with the voltage outputted at the time. In the following paragprahs, I will explain how the software part of the remote controlling work while trying to make my explanation in a way that it can be reproduced by the reader.

The control being remote and real-time has been achieved by the use of Raspberry Pi. The code, which was written in the language of Arduino (based on C++) was written and put into the Pi. The code written can be found on a GitHub repository. [6] Then, the Arduino was connected to the Pi as explained in the previous section.

To connect to the Pi, the SSH Protocol (Secure Shell) was used. With this, a user can access the Pi from the command line of their computer. For this, just use the the command "ssh" followed by the user name associated with your Pi and its IP address. The command could look like this:

```
$ ssh pi@128.***.***.***
```

Then, if there is a password for the account, the terminal will ask for it. After this step, you have the control of the Pi on your command line.

Next step is making the Pi, which works with Python, compile and and upload the code for the Arduino, which is written in C++ instead of Python. This is a tricky point. However, luckily, there exists a third party application called "Ino". The information on how to set up Ino for Pi can be found on Ino's website. [7]

After the installation the next step is compiling the code for Arduino. For this, first, make a directory on the Pi. Then, one can use the command "ino init" in order to create two files in the directory named "lib" and "src". The directory named "lib" is for the libraries that the code may be using. If it is the case, you can put the library file in "lib". The directory "src" is for the code. One should put the code for Arduino in "src". The process can be seen below:

```
pi@raspberrypi:~ $ mkdir example
pi@raspberrypi:~ $ cd example
pi@raspberrypi:~/example $ ino init
pi@raspberrypi:~/example $ ls
lib  src
```

The only external library that was used in the code was the library named "Adafruit_MCP4725" and it can be found on a GitHub repository. [8] This library should be put inside the directory "lib".

The following step is compiling the code. For this, go to the directory you have created. In our example, the name of the directory is "example". While in the directory, use the following code:

```
pi@raspberrypi:~/example $ ino build
```

Then Ino will compile the code, and will display the information about it on the command line. Now, the code is ready to be uploaded to Arduino. For this, use:

```
pi@raspberrypi:~/example $ ino upload
```

Ino will guess the serial port. In our example, the message presented on the command line was:

```
Guessing serial port ... /dev/ttyACM0
```

Now, the program is ready to be run. There are currently two programmes on the GitHub repository, which are accessible by the public. These are called “constant” and “ramp”. The program “constant” allows one to output a certain voltage for a certain amount of time while collecting the data of output voltage reading. The program “ramp” is to have an increasing order of voltages. In this program, the user can determine the start voltage, the end voltage, voltage increase for each interval and the amount of time for each interval as well as deciding an upper limit to the voltage outputted, and collecting the outputted voltage data and writing it to the command line.

To run the program, while in the main directory, i.e. “example” in our example, use the following code:

```
pi@raspberrypi:~/droege_main/ramp $ ino serial
Guessing serial port ... /dev/ttyACM0
picocom v1.7

port is        : /dev/ttyACM0
flowcontrol    : none
baudrate is    : 9600
parity is      : none
databits are   : 8
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
nolock is      : yes
send_cmd is    : sz -vv
receive_cmd is : rz -vv
imap is        :
omap is        :
emap is        : crcrlf,delbs,

Terminal ready
```

After this, the program will start to run. An example run can be found in the Appendix.

3 Calibration

In order to achieve better accuracy in terms of the voltage output and the voltage reading the Droege was calibrated. This section will be about the method used to calibrate remote control system of the Droege and results of the calibration.

3.1 Calibration of Voltage Output

Since the DAC has 12-bit resolution, Arduino could write a bit value between $0 - 4095$ as a voltage output. The value 0 would mean 0V of output out of Arduino and 4095 would mean the maximum amount of voltage the Arduino can output.

In order to calibrate the voltage output, a known bit value was written by the Arduino to the channel A of the remote input while 0 bit was written to channel B. The voltage applied to channel A was varied, and the output of the Droege was measured using a DC voltmeter via the voltage monitor of the Droege. The voltage monitor of the Droege has 1000 to 1 ratio with the actual output, i.e. if the real output was 1496 V, the voltage monitor would show the value 1.496 V. With this method, the voltage output was calibrated and the result can be found in Figure 5. It should be noted that the error bars are too small to be visible.

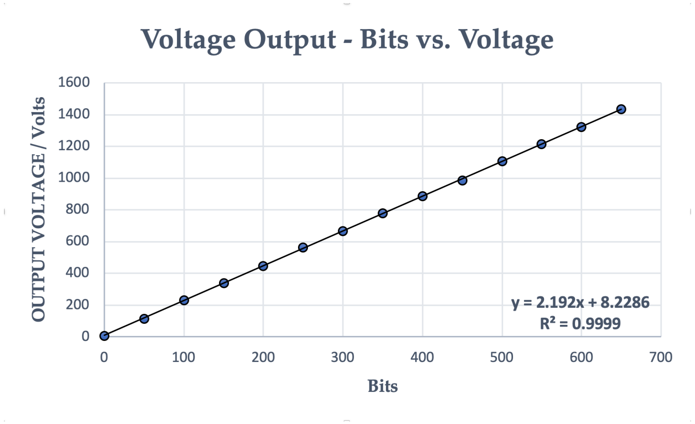


Figure 5: Calibration Results of Voltage Output

3.2 Calibration of Voltage Reading

Since the ADC has 10-bit resolution, Arduino could read a true DC voltage and interpret it as a bit value in between $0 - 1023$ similar to the previous subsection.

A known voltage value which was measured using a DC voltmeter, was read by the Arduino and the corresponding bit value was recorded. Then, these values were graphed. The result can be found in Figure 6.

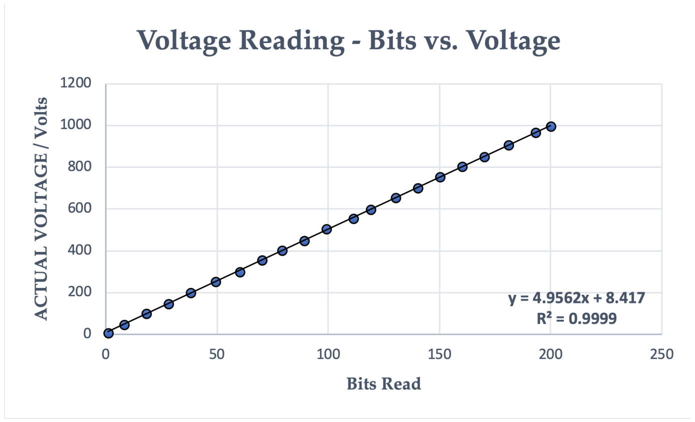


Figure 6: Calibration Results of Voltage Reading

4 Results

The results are the following:

- Droege can be controlled from anywhere via the internet with a range of output of 0 to 7.5 kV with ± 2.19 V uncertainty
- The true output voltage can be remotely monitored with a range of 0 to 5.0 kV with ± 4.96 V uncertainty
- The true output current can be remotely monitored with a range of 0 to 500 μA with ± 122 nA uncertainty
- The values of true output voltage and current can be collected to the terminal line of the user with a time-stamp
- The desired voltage and time values can be set at each use without any modification of the code
- The method enables the user to set an upper limit to the voltage and the current as a safety measure
- The code for the programmes "ramp" and "constant voltage" are publicly available

5 Conclusions

In conclusion, a new method of real-time remote controlling Droege over the internet such that a specific voltage or an increasing/decreasing order of voltages with exact increments can be supplied with high precision for the desired duration of time has been developed.

The new method has advantages. For example, there is no need of modification of the code, i.e. the user can determine the specific values they will use in a run by entering the values into the terminal instead of modifying the code every time. Also, there is no need of uploading or compiling the code to the micro-controller each time. This saves time and will allow someone who is not familiar with the code to use this method. Since the codes are readily available, the user may start remote-controlling the Droege directly, or use the available codes as a basis to the code suitable for their specific application. In addition to this, since the user can determine an upper boundary for the output voltage as a safety measure, the equipment used may be protected or a possible accident may be prevented.

On the other hand, this method of controlling the Droege being remote and real time enables the user to control the output from any location with an internet connection. The user can remotely collected voltage and current data in real-time with a time stamp as well. Therefore, whether for the purpose of controlling the output, or collecting data, the user and the Droege do not have be in close proximity.

I sincerely hope that this little project may be useful for somebody.

6 Acknowledgments

I would like to thank Prof. Henry J. Frisch of the Department of Physics, Enrico Fermi Institute and the College of The University of Chicago for all the help, guidance, and offering me this project. I also would like to thank Evan Angelico, Eric Spielgan, Mark Zaskowski, and Charles Whitmer for their suggestions and support, and Gary Drake for providing information and documentation about the Droege.

References

[1] Thomas F. Droege. “HIGH VOLTAGE DC POWER SUPPLY”. 4,888,673. Dec. 19, 1989. URL: https://patentimages.storage.googleapis.com/c1/99/cf/f81b130a4df97c/US4888673.pdf.

[2] *ES-7125 Positive High Voltage Power Supply.* Oct. 20, 2008. URL: http://prep.fnal.gov/catalog/hardware_info/fermilab/nim/es7125.html (visited on 06/07/2018).

[3] Jean-Francois Caron, Fawzi Abusalma, and Gabe Cobos. “Droege Remote Control with a Microcontroller”. Mar. 26, 2018. URL: http://mu2e-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=16637&filename=report.pdf&version=1.

[4] *ARDUINO UNO REV3.* Dec. 21, 2018. URL: https://store.arduino.cc/usa/arduino-uno-rev3.

[5] *MCP4725 Breakout Board - 12-Bit DAC w/I2C Interface.* URL: https://www.adafruit.com/product/935.

[6] *GitHub Repository of Mesut Caliskan.* URL: https://github.com/MesutCaliskan.

[7] *Ino.* URL: http://inotool.org.

[8] *Adafruit_MCP4725.* URL: https://github.com/adafruit/Adafruit_MCP4725.

7 Appendix

An example run of the program called “ramp” can be found below:

```
Dual MWPC High Voltage Power Supply (Droege) Ramp Programme v1.2

Written by Mesut Caliskan on August 13 at The University of Chicago

Before the process starts, you enter the values which specify your ramp...

Please enter the INITIAL VOLTAGE value (in volts):

INITIAL VOLTAGE value (in volts) is: 50 V
This value corresponds to 10 bits


Please enter the FINAL VOLTAGE value (in volts):

FINAL VOLTAGE value (in volts) is: 500 V
This value corresponds to 205 bits


Please enter the TIME VALUE (in ms) for each step:

TIME VALUE for each step is: 1000 ms


Please enter the VOLTAGE INCREASE (in volts) with each step:

Each INCREMENT is: 5 V
This value corresponds to 2 bits


Your values are like the following:
INITIAL VOLTAGE:  50 V
FINAL VOLTAGE:  500 V
TIME VALUE:  1000 ms
INCREMENT:  5 V

ARE THE VALUES CORRECT? - (y/n)
THE PROCESS IS STARTING in 5...
THE PROCESS IS STARTING in 4...
THE PROCESS IS STARTING in 3...
THE PROCESS IS STARTING in 2...
THE PROCESS IS STARTING in 1...

BEGIN...
Bits written (b) | Voltage written (w) | Voltage Read (r) | Time (t)

b: 10  |  w: 50  |  r: 51  |  t: 35117

b: 12  |  w: 55  |  r: 54  |  t: 36127

b: 14  |  w: 60  |  r: 62  |  t: 37128

b: 16  |  w: 64  |  r: 66  |  t: 38129

b: 18  |  w: 69  |  r: 70  |  t: 39131

b: 20  |  w: 74  |  r: 76  |  t: 40132

b: 22  |  w: 78  |  r: 80  |  t: 41134

b: 24  |  w: 83  |  r: 85  |  t: 42135

b: 26  |  w: 87  |  r: 91  |  t: 43136

b: 28  |  w: 92  |  r: 96  |  t: 44137

...

b: 186  |  w: 456  |  r: 463  |  t: 123249

b: 188  |  w: 461  |  r: 468  |  t: 124251

b: 190  |  w: 466  |  r: 473  |  t: 125252

b: 192  |  w: 470  |  r: 478  |  t: 126254

b: 194  |  w: 475  |  r: 482  |  t: 127255

b: 196  |  w: 479  |  r: 486  |  t: 128257

b: 198  |  w: 484  |  r: 491  |  t: 129258

b: 200  |  w: 489  |  r: 496  |  t: 130259

b: 202  |  w: 493  |  r: 502  |  t: 131261

b: 204  |  w: 498  |  r: 506  |  t: 132262

RAMP HAS ENDED SUCCESSFULLY...

To exit please press Ctrl A + Ctrl X
```