Arduino-Based Piezo Driver

Kiko Galvez and Shannon Zachow

July 2015

1 Introduction

Single-photon experiments require the scan of the path-length difference of an interferometer. The acquisition system works as shown in Fig. 1. The single photons are detected by avalanche photodiode detectors (APD). The digital pulses coming from the detectors are received by an Altera board. The latter counts the electric pulses coming from avalanche photodiode (APD), records those counts, the coincidences between selected inputs, and sends those results to a computer (PC). A labview program running on the PC reads the Altera board. The program must also scan the interferometer phase so we can record oscillations in the photon counts as evidence of single-photon interference. For our Mach-Zender, non-polarizing-type interferometer, this is done via a piezo-electric transducer. We apply a voltage to a piezo-electric, which pushes one of the mirrors of the interferometer, and thereby changing the phase. This voltage varies typically from 0 to 150 V. Since this voltage goes beyond what a PC can put out, we much amplify it with a high-voltage amplifier (A). The Labview program must then be able to scan stepwise the voltage that is applied to the piezo-electric, and record the photon counts after every step.

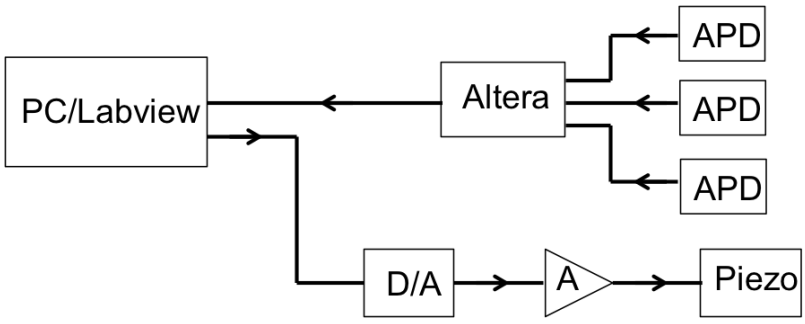

Figure 1: Block diagram of the computer automation of single-photon interference experiments.

In the past we had a cumbersome and expensive system: a GPIB board which controlled a digital to analog conversion NIM module. We have replaced this by a

very inexpensive and easy to use system. It is based on the Arduino-UNO board (a $25 piece of electronics), which is controlled by Labview via a USB link.

2 Digital to Analog

We have made two designs. A first one connects the digital I/O pins of the Arduino board to a R-2R resistor ladder that acts as a digital to analog converter. The second design uses a commercial D/A integrated circuit and a home-made voltage amplifier. Next we discuss the two designs in detail.

2.1 Design 1: Home-made D/A

This circuit uses 10 digital I/O pins of the Arduino board to generate a voltage out via an R-2R ladder. The output voltage is further amplified. The picture of the box with the arduino board and circuit is shown in Fig. 2. Because we use two operational amplifiers to amplify the output of the D/A, we need ± 12 V supply input.

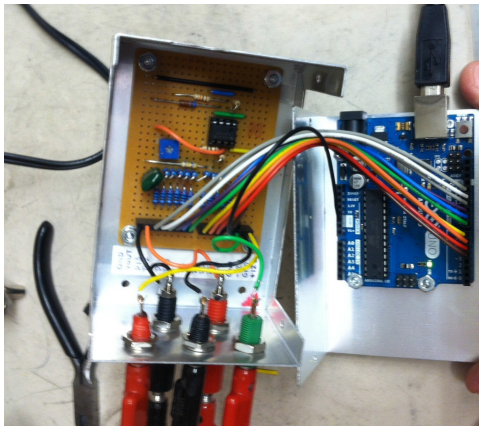

Figure 2: Picture of the home-made box with the Arduino UNO board and the custom D/A circuit.

The circuit diagram wiring of the R-2R d/A is shown in Fig. 3. It is a very basic circuit. The output of the ladder is amplified by two cascaded operational amplifiers (TL072) so that the output goes from 0 to 10 V. The resistors in the ladder are 1% resistors.

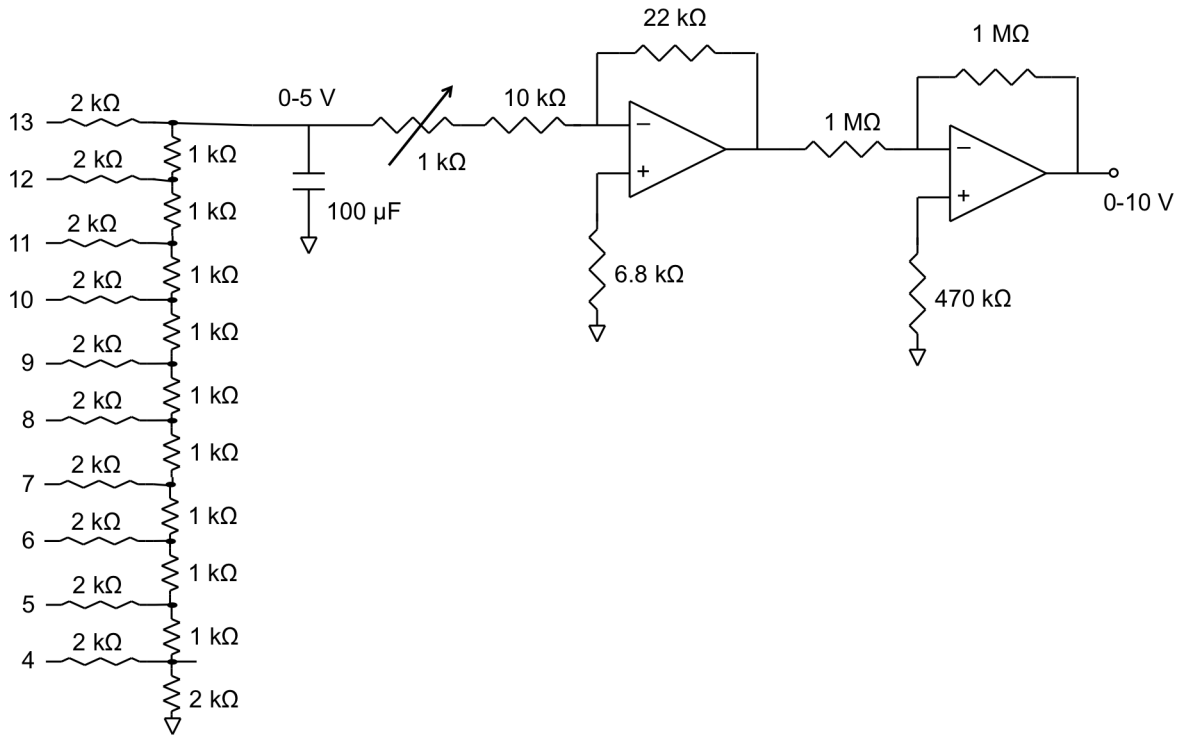

Figure 3: Circuit diagram of the R-2R digital to analog circuit.

The next step is the putting the interface together is the program that is loaded onto the Arduino board. It is listed below:

```
#define PIN_COUNT 10 // using this many digital pins for output
#define PIN_BASE 4 // this is the lowest digital pin number used for output
#define BIGGEST_WORD (1<<PIN_COUNT) // largest # we can output = 2^PIN_COUNT
#define BUFLEN 128
#define OUTMIN (0.0)
#define OUTMAX (10.0)
void setup()
int i,p;
// designate pins we are using for output.
for(p=0; p<PIN_COUNT; p++)
pinMode(p + PIN_BASE, OUTPUT);
// start Arduino's serial port @ 115200 baud.
```

```
Serial.begin(115200);
// set all output pins initially to ZERO.
for(p=0; p<PIN_COUNT; p++)
digitalWrite(p + PIN_BASE, LOW);
// digitalWriteWord(int)
//
// utility function to put a binary word bitwise onto all the output pins.
// 'w' is the word we wish to express on the output pins.
// 'p' variable counts up through pins.
// 'm' variable is a mask, with only one bit set HIGH.
// effectively a shift register, 'm' shifts left (up to LSB) 1 bit position
// each time through loop.
//
void digitalWriteWord(int w)
int p,m;
if(w >= BIGGEST_WORD)
w = BIGGEST_WORD - 1;
else if(w < 0)
w = 0;
for(p=0, m=1; p<PIN_COUNT; p++, m<<=1)
digitalWrite(p + PIN_BASE, (w&m ? HIGH : LOW));
void loop()
int cmd;
char b[BUFLEN];
float x,y;
if(Serial.available() > 0) // if bytes are waiting for us in the serial port...
x = Serial.parseFloat(); // try to read them as a floating point number
if(x >= OUTMIN
&& x <= OUTMAX)
y = BIGGEST_WORD * ((x - OUTMIN) / (OUTMAX - OUTMIN));
// range mapping
digitalWriteWord((int)y); // express that voltage at the output.
```

```
else // out of range
// send an error message back through the serial port.
// You won't see this, unless you have opened the Serial Monitor in
// the arduino IDE, while the program is running.
// found hint online that Serial.print is implicated in
// slow-down of serial coms – don't do this!
/**
Serial.print("Range Error: asked to output ");
Serial.println(x);
Serial.print(" OUTMIN = ");
Serial.println(OUTMIN);
Serial.print(" OUTMAX = ");
Serial.println(OUTMAX);
**/
Serial.readBytesUntil('\ n',b,BUFLEN); // read & discard newline
```

2.2 Design 2: Commercial D/A with Home-made Amplifier

The commercial version of the D/A is easy as well. We use the 12-bit LTC1257 digital to analog converter ($ 10.75 each), which takes the digital input serially and puts out a voltage 0-9 V. The IC runs at 1.4 MHz maximum, so we divide the Arduino clock (16 MHz) by 16. The circuit is shown in Fig. 4.

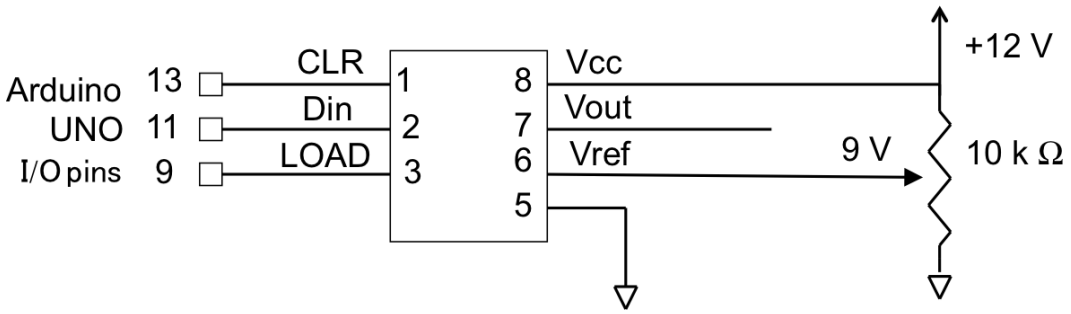

Figure 4: Circuit diagram of the commercial digital to analog.

A photo of the circuit is shown in Fig. 5.

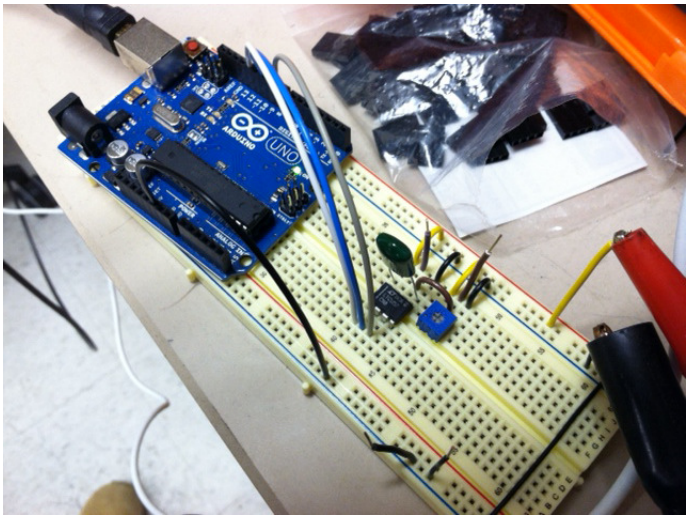

Figure 5: Photo of the circuit with the of the commercial digital to analog converter.

The program that we load onto the Arduino is given below.

```
#include <SPI.h> // use Arduino's SPI library
// digital pins used by the SPI library:
#define PIN_MOSI 11 // Connected to PIN 2 Din of DAC
#define PIN_MISO 12 // Not used
#define PIN_SCK 13 // Connected to PIN 1 of DAC
#define PIN_SS 10 // Not used
#define PIN_nLOAD 9 // Connected to PIN 3 of DAC
#define VOLTAGE_MAX (9.0) // max output of DAC = Vref pin 6
#define DIGITAL_MAX (4095) // biggest 12bit integer + 1.
long loop_count = 0;
// setup() is automatically called just once at startup or reset
void setup ()
pinMode(PIN_nLOAD, OUTPUT); // configure our load pin
SPI.begin(); // start SPI bus (configs all SPI pins automatically)
SPI.setClockDivider(SPI_CLOCK_DIV16);
// arduino runs at 16MHz
// LTC1257 DAC max speed 1.4MHz
// divide arduino clock speed by 16 to clock LTC1257 @ 1MHz
```

```
SPI.setBitOrder(MSBFIRST); // LTC1257 expects bits to arrive MSB first
SPI.setDataMode(SPI_MODE0);
// start Arduino's serial port @ 115200 baud.
Serial.begin(115200);
void loop()
int cmd;
char b[BUFLEN];
float x,y;
if(Serial.available() > 0) // if bytes are waiting for us in the serial port...
x = Serial.parseFloat(); // try to read them as a floating point number
if(x >= 0
&& x <= VOLTAGE_MAX)
y = transferFloat(x);
void transferFloat(float v)
int d, lob, hib, bits, bitshift, Nmax, volt, voltlow;
float voltf;
bits = 12;
bitshift = 12-bits;
volt = (int) DIGITAL_MAX * (v/VOLTAGE_MAX);
voltlow = volt >> bitshift;
Nmax = DIGITAL_MAX >> bitshift;
voltf = (float) voltlow/Nmax;
d = (int) (DIGITAL_MAX * voltf);
// SPI library is set up to transfer one byte (8bits) at a time.
// we have 12bits, so we need 2 bytes.
// separate the integer value 'd' into a low-byte and a high-byte.
lob = (d & 0x00ff);
hib = (d & 0x0f00) >> 8;
// LTC1257 expects MSB first, so make sure we ship 'hib' first!
SPI.transfer(hib);
SPI.transfer(lob);
```

// 12bits are now in LTC1257's serial-in-shift-register.

// Toggle LOAD pin so this is latched to LTC1257's DAC output register.

// minimum toggle time = 150ns.

// include delayMicroseconds(1); call just to make sure.

digitalWrite(PIN_nLOAD, LOW);

delayMicroseconds(1);

digitalWrite(PIN_nLOAD, HIGH);

// if our peripheral had a Chip-Select pin, we would need to disable that here.

// digitalWrite(pinCS, HIGH);

3 Voltage Amplifier

The output of the Arduino box is fed into a voltage amplifier circuit that uses EMCO model Q02-24 voltage amplifier (cost is $81). The circuit to drive this device is shown in Fig. 6. It uses an LM7815 voltage regulator that feeds into the voltage amplifier.

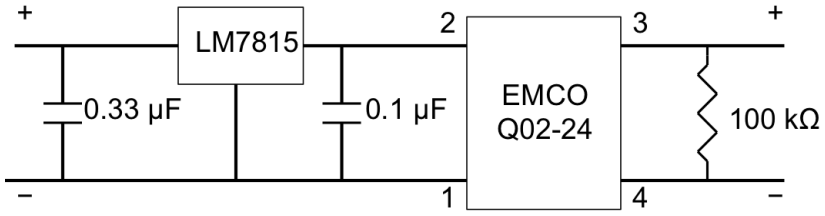

Figure 6: Circuit diagram of the voltage amplifier.