Analog-to-Digital Conversion
Digital-to-Analog Conversion (ADC/DAC)
Lab 2

Luis Cardenas
E.E. 470
Lab Instructor: Gary Hill

Table of Contents

Introduction

Experiment involves the conversation of a signal going from an Analog to Digital control and then from Digital to Analog control. The sine wave signal will going through the Arduino Duemilanove (ADC) which will convert the analog signal into a digital signal.(Using 10 bit) Then the MCP4921 microchip (DAC) will converted the signal from digital(Using 12 bit) back to analog signal. We will also check the frequency at which the Arduino (ADC) is not able to produce the analog signal correct because we begin to see distortion in the output of the signal as the MCP4921 microchip converts the digital to analog.

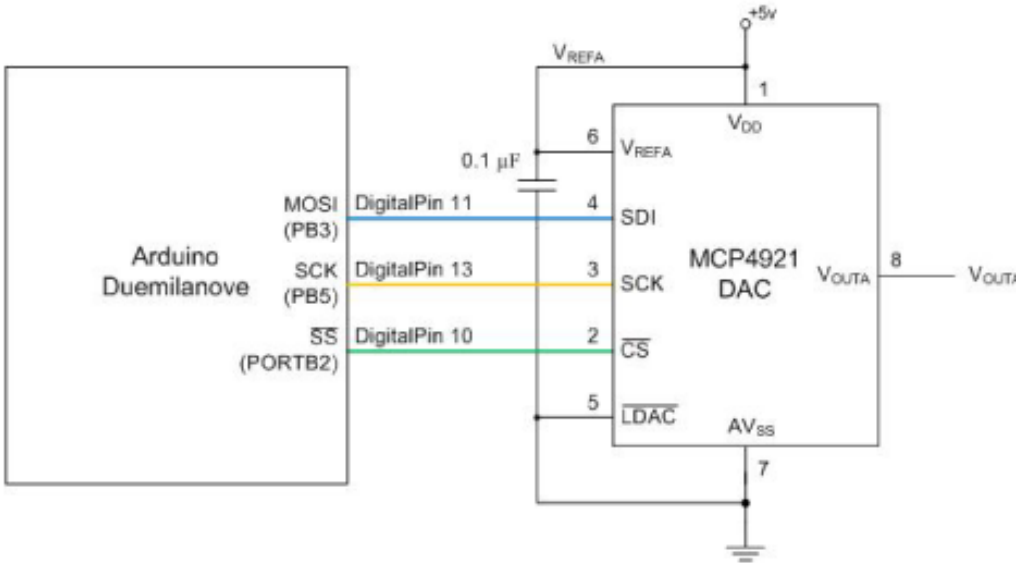
List of items needed for the Lab

1. Arduino Duemilanove: ATmeg328P ADC
2. DAC Microchip: MCP4921
3. micro Farad
4. 9v power adapter (to Arduino Uno), 5 volts voltage supplied (coming from the Arduino)
5. Parallax USB Oscilloscope
6. Function Generator Model FG-500

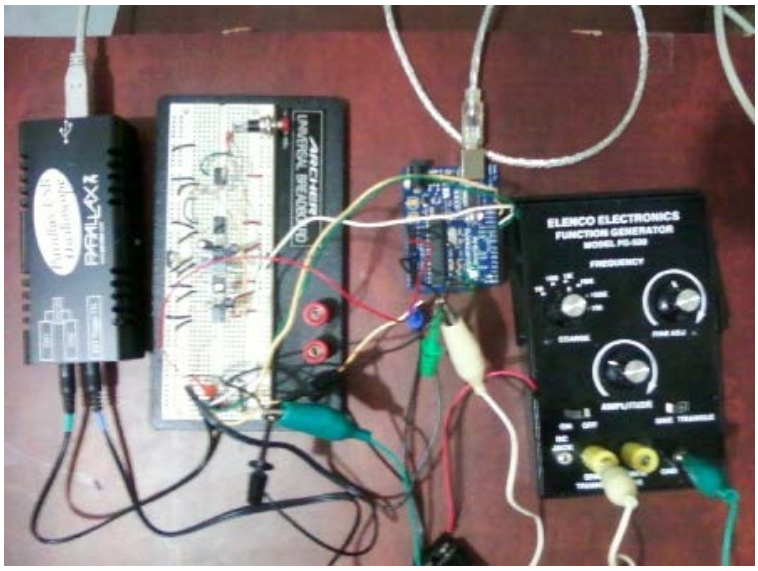
If external signal generator used.

7. 2.2 K resistor
8. 2 Shottky diodes

Circuit Schematics



Complete Setup



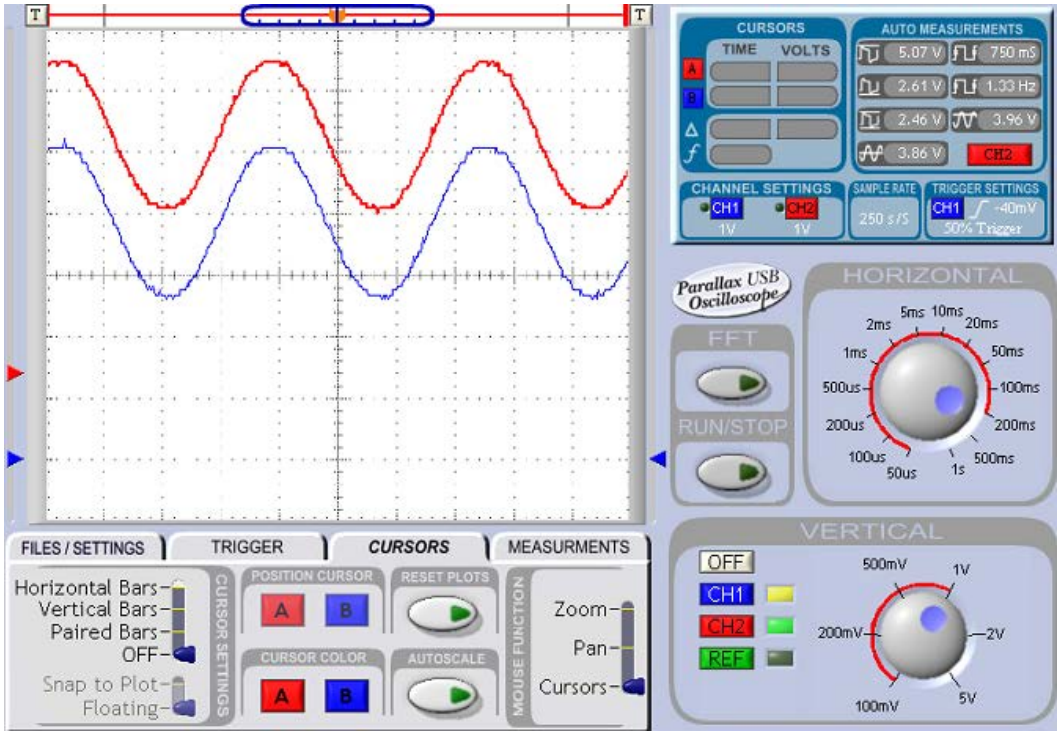
Program

```
//************************************************************//
//  Name        : Analog In to 12-bit SPI DAC Microchip MCP4921 //
//  Author      : Gary Hill                                     //
//  Date        : 14 March, 2010                                //
//  Version     : 1.0                                           //
//  Reference(s): http://www.arduino.cc/playground/Code/Spi     //
//  Notes       : Download Spi folder and place in              //
//              : arduino-00nn/hardware/libraries folder        //
//************************************************************

// SPI Interface     SS_PIN(PB2), SCK_PIN(PB5), MOSI_PIN(PB3), MISO_PIN
// Arduino Pin       10           13            11             12
// MCP4921 DAC       SS           SCK           MOSI           n/a
#include <Spi.h>

// ATmega328P ADC
int analogPin = 0;     // analog input channel

// ADC analog input value
word sensorValue = 0;  // equivalent to unsigned int

// Byte of data to output to DAC
byte data = 0;

void setup() {
  //set pin(s) to input and output
  pinMode(analogPin, INPUT);
}
void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(analogPin); // comment out this line to test DAC
  // sensorValue = 0x0200; // 0x03FF = Vref, 0x0200 = 1/2 Vref, 0x0000 = 0
  sensorValue = sensorValue << 2 ; // 10 bit ADC to 12-bit DAC word
  // set SS pin low, beginning 16-bit (2 byte) data transfer to DAC
  digitalWrite(SS_PIN, LOW);
  // send high byte
  data = highByte(sensorValue);
  data = 0b00001111 & data;     // clear 4-bit command field (optional)
  // set command: 0 = DACA, 0 = buffered, 1 = 1x, 1 = output buffer enabled
  data = 0b00110000 | data;
  Spi.transfer(data);   // alternate: shiftOut(MOSI, SCK, MSBFIRST, data);
  // send low byte
  data = lowByte(sensorValue);
  Spi.transfer(data);   // alternate: shiftOut(MOSI, SCK, MSBFIRST, data);
  // set SS pin high, completing 16-bit transfer to DAC
  digitalWrite(SS_PIN, HIGH);  }
```

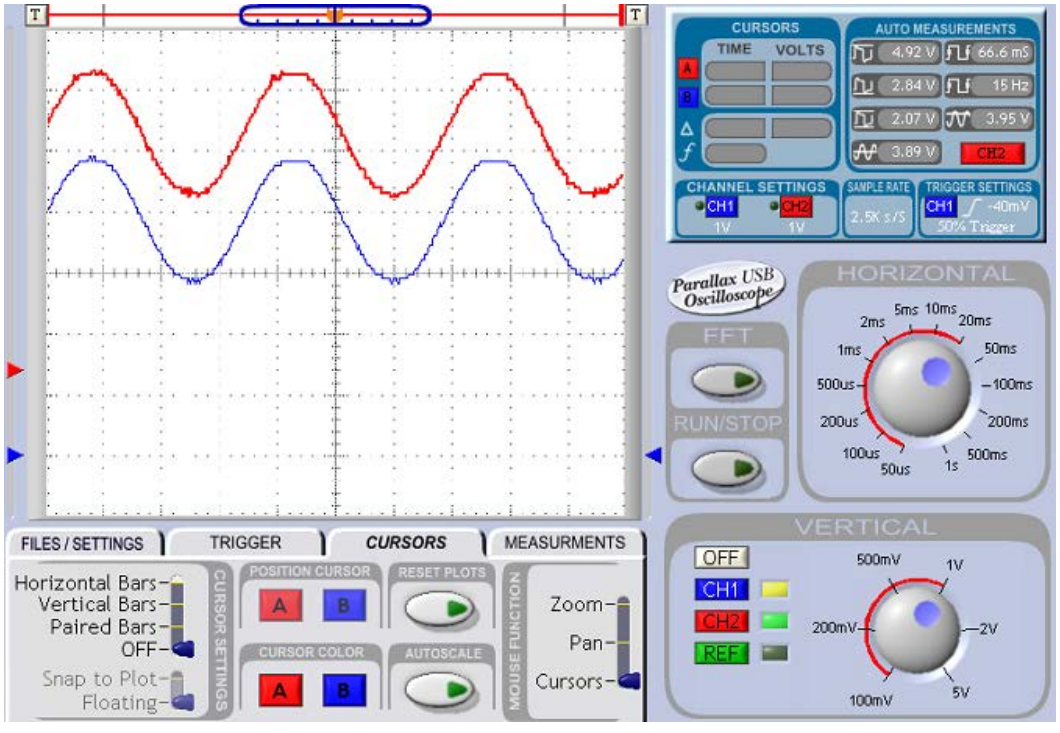
Test Results

INPUT (Blue/Bottom) and OUTPUT(Red/Top)

At 10 Hz

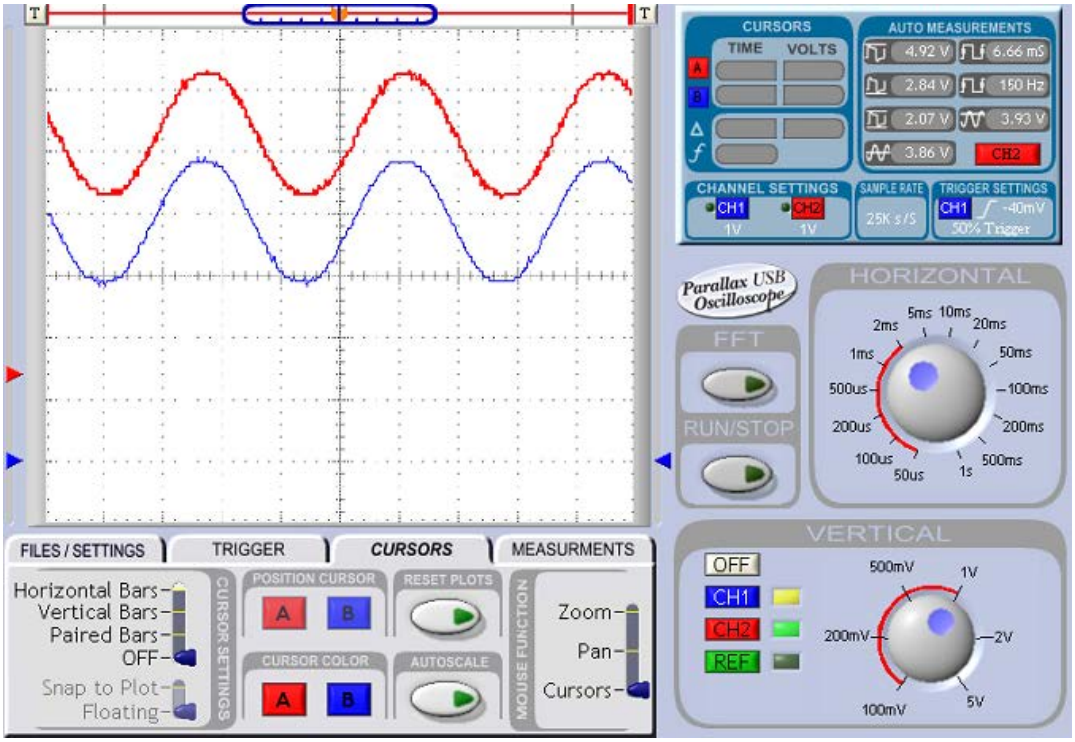


At 100 Hz

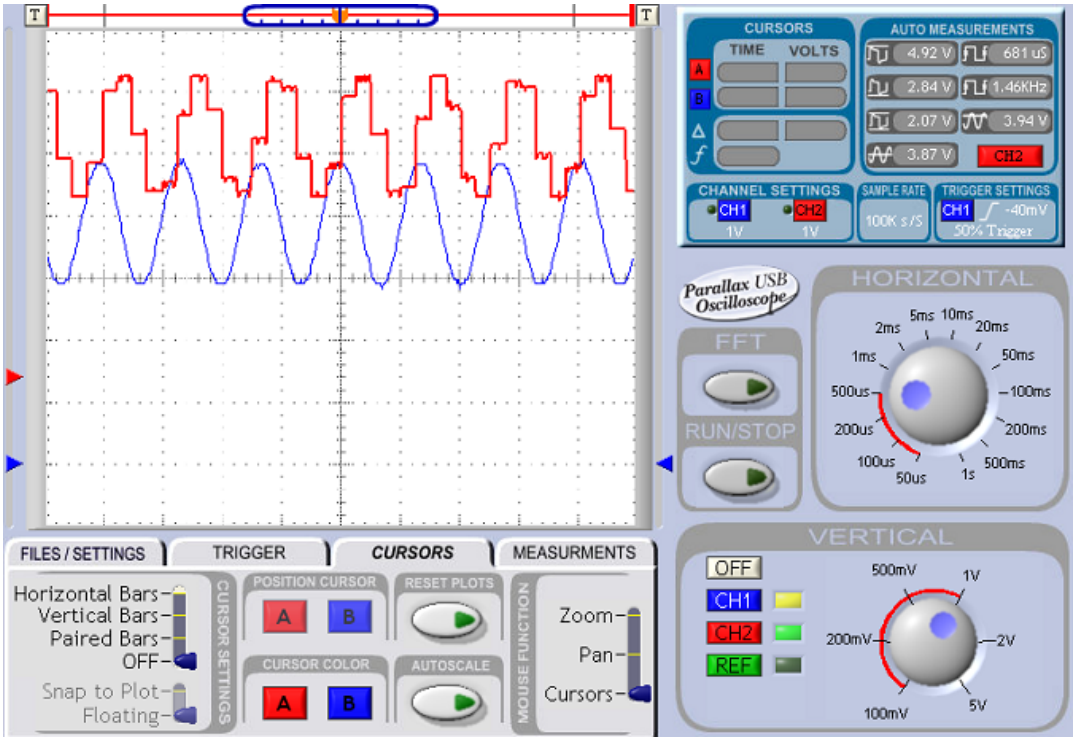


INPUT (Blue/Bottom) and OUTPUT(Red/Top)

At 1 kHz :



At 10 kHz



In the above output waveform we can clearly see eight digitized states per cycle. At a frequency of F = 10 khz and a period T = 1/F = 0.1 msec, we therefore have a sample frequency of approximately 0.1 msec / 8 samples = 125 usec /sample.

Conclusion

I was able to compare the input/output of the Arduino and MAC chip by using different frequency . At 10 Hz, 100 Hz and 1k Hz the output is able to be reproduce . Once we go to 10 k Hz and 100 k Hz the output begins to be distorted So as the frequency got higher, it reach a point where the DAC chip was not able to reproduce the exact copy of the analog signal coming from Arduino. This was cause by the Ardruino not being able to keep up with the higher frequency. The best frequency to use for the Arduino to handle an analog signal is at 1KHz.