ECE304

Lab3&4 Memo

Bruce Jiang, Siwei Cai

Introduction

The objective of this week's experiment is to transfer a simulation of nulling filter from MATLAB to Arduino. Within the limited storage size on Arduino, applied algorithm should operate on fixed point format instead of floating points. To understand better, experimenting on MATLAB to see what the graph would looks like is the first step.

Plots Results

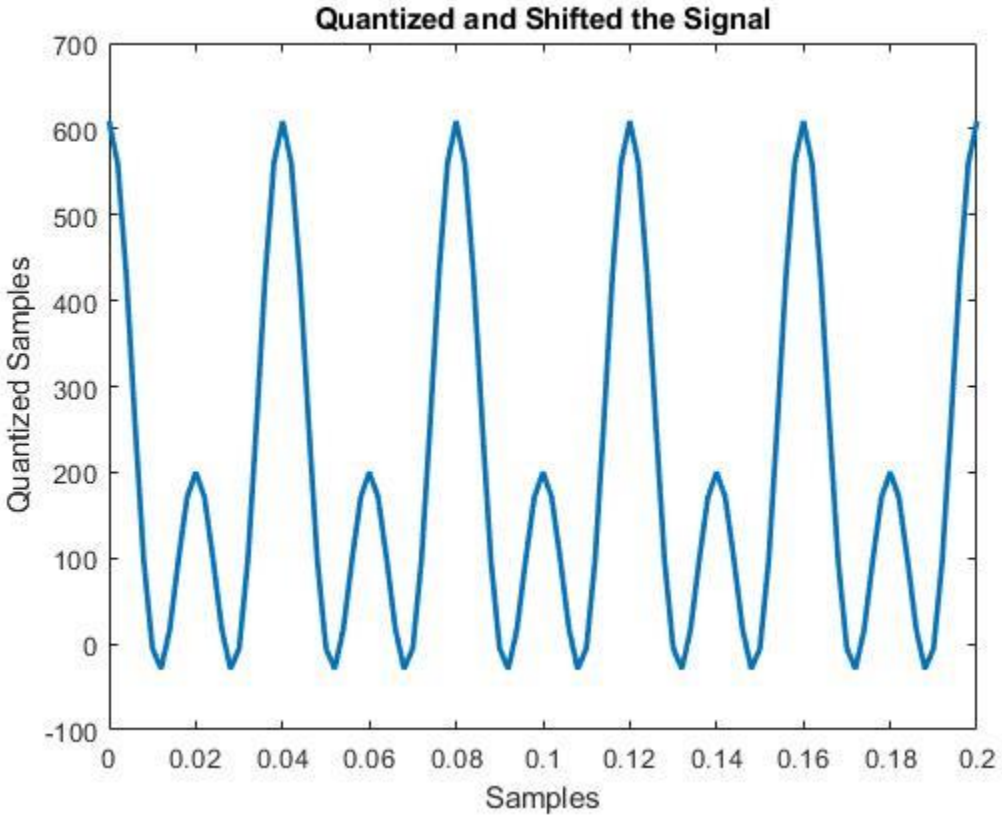

Figure 1 – MATLAB Input & Output Discretized Signals

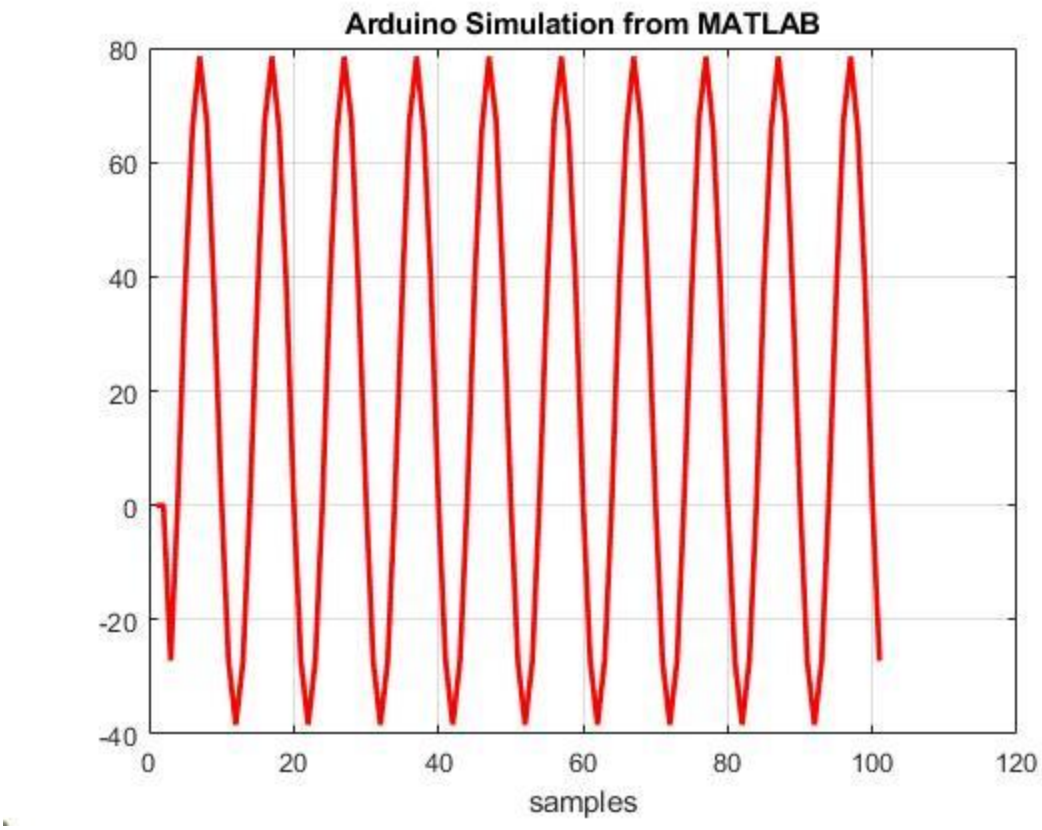

Figure 2 – Arduino Nulling Filter Simulation From MATLAB

From MATLAB, to get figure 1 and 2, we first convert the x value into A/D by multiplying 1023 over 5, then rounded and shifted it. Using the new quantized samples and applied to y function to get nulling filter process done.

```
%%
new_quants = round(x.*(1023/5))+200; %b = [1 -1.9021 1];

for i = 1:101

    fprintf('%d', new_quants(i))
    fprintf(',')
end
```

Figure 3 – MATLAB Code Quantization

Next step implements the new quantization table from above into Arduino and convert them and FIR coefficients properly into Q format.

Below are the Arduino functions to convert a given integer or float number into the fixed format numbers.

Lab4

```
#include<FixedPoints.h>
#include<FixedPointsCommon.h>

SQ15x16 FIR_Co[3] = {1,-1.9021,1};
typedef int16_t fixed_point_t;
//New Quants Table
SQ15x16 x[101] = {609,560,429,257,98,-5,-29,17,98,171,200,171,98,17,-29,-5,
98,257,429,560,609,560,429,257,98,-5,-29,17,98,171,200,171,98,17,-29,-5,98,257,429,560,
609,560,429,257,98,-5,-29,17,98,171,200,171,98,17,-29,-5,98,257,429,560,609,560,429,257,
98,-5,-29,17,98,171,200,171,98,17,-29,-5,98,257,429,560,609,560,429,257,98,-5,-29,17,98,
171,200,171,98,17,-29,-5,98,257,429,560,609};

SQ15x16 y[101] = {};

int n;

void setup() {

  // put your setup code here, to run once:
  int16_t float_to_fixed(float input,int16_t fractional_bits);
  Serial.begin(2000000);

  noInterrupts(); // disable all interrupts
  TCCR5A = 0;
  TCCR5B = 0;
  TIMSK5=0;
  TCNT5 = 0;

  OCR5A = 500; // compare match register 16MHz/256/0.004s=500Hz
  TCCR5B |= (1<<WGM52); // CTC mode
  TCCR5B |= (1<<CS52) | (0<<CS51) | (0<<CS50); // 256 prescaler
  TIMSK5 |= (1<<OCIE5A); // enable timer compare interrupt
  interrupts(); // enable all interrupts
}
```

Figure 4 – Arduino Code

```
void loop() {
   if (n == 101){
      noInterrupts();
      delay(1000);
      for (int j = 0;j<101;j++){
        delay(10);
        Serial.print(static_cast<double>(y[j]));
        Serial.println(" ");
      }
      n+=1;
   }
}

//timer interrupt
ISR(TIMER5_COMPA_vect){
  for (n = 2; n < 101; n++){
    y[n] = static_cast<double>(FIR_Co[0]*x[n] + FIR_Co[1]*x[n-1] + FIR_Co[2]*x[n-2]);
  }
}
```

Figure 5 – Arduino Code

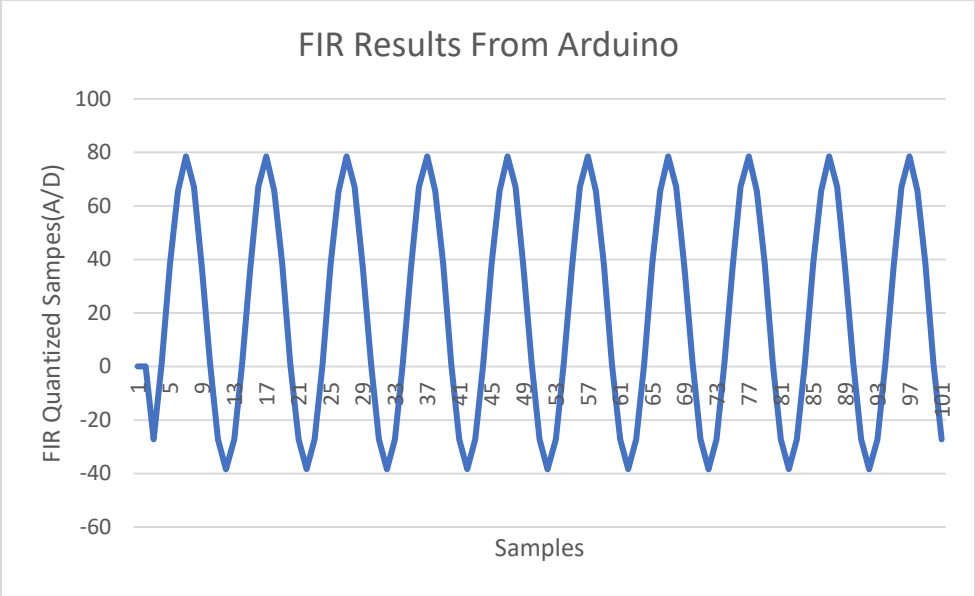

Figure 6 – Arduino FIR Outputs in Excel

Conclusion

In this lab, the nulling coefficient we implemented into Arduino used 15 decimal bits and 16 fractional bits for all the conversions include integer and float points. And in the compelling display the Arduino board have used up 1024 bytes of Global variables which is 8192 bits, 12% of 8192 bytes. Thus, the integer part in x values we implemented from MATLAB could reduce the fractional bits size at the end to save more memory of Global variables. Finally, the Figure 5 looks identical to figure 2, indicating the success of the filter simulating transfer.

Appendix

https://github.com/Pharap/FixedPointsArduino