Contents

1 Introduction

ECG and how it is measured

2 Methods - Circuit

When discussing which filter topology to pick, we weighed complexity of the circuit versus their frequency response. Figure 1 shows fifth order versions of the ones we considered, which made Chebyshev seem particularly interesting. Ultimately settled for a Sallen-Key (not-depicted), due to its simplicity and because we did not have access to inductors.

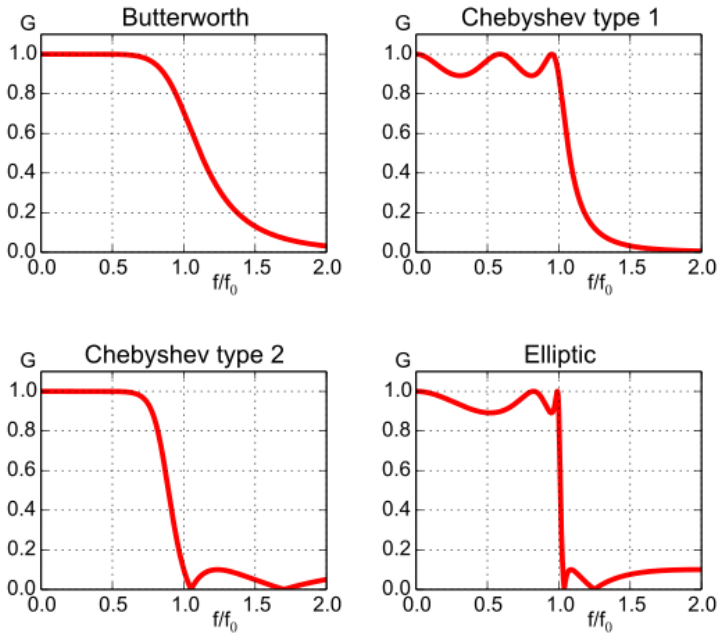

Figure 1: Frequency responses for different fifth order filters. Wik (2016)

The desired cut-off frequencies were 0.5 Hz and 100 Hz for the high-pass and low-pass filters respectively. The actual values deviate from it, but the choice of resistor and capacitor pair was made by picking resistors smaller than MΩ and capacitances larger than nF, that is 10^6 and 10^{-9}, respectively. This is so we are not working in the same order of magnitude as stray capacitances and amplifiers' internal resistances.

The total gain needed was 1000, seeing as the ECG signal had an amplitude of around 5 mV. In order to avoid any risk of saturation, the instrumentation and operational amplifier were given gains of 11 (due to available resistor values) and 100 respectively. No particular reason for the order, since both can achieve gain of 100 without risks.

The presence of the potential divider provides 2.5 V to V_{ref}, allowing for use of the arduino as our single supply. There is a buffer in order to prevent loading.

3 Methods - Program

4 Results

Outputs that you have measured.Discuss the relevance of these results to proving that your device functions as designed.

5 Conclusion

How could this become an actual ECG device? What would be some other considerations that need to be taken into account?

6 Appendix

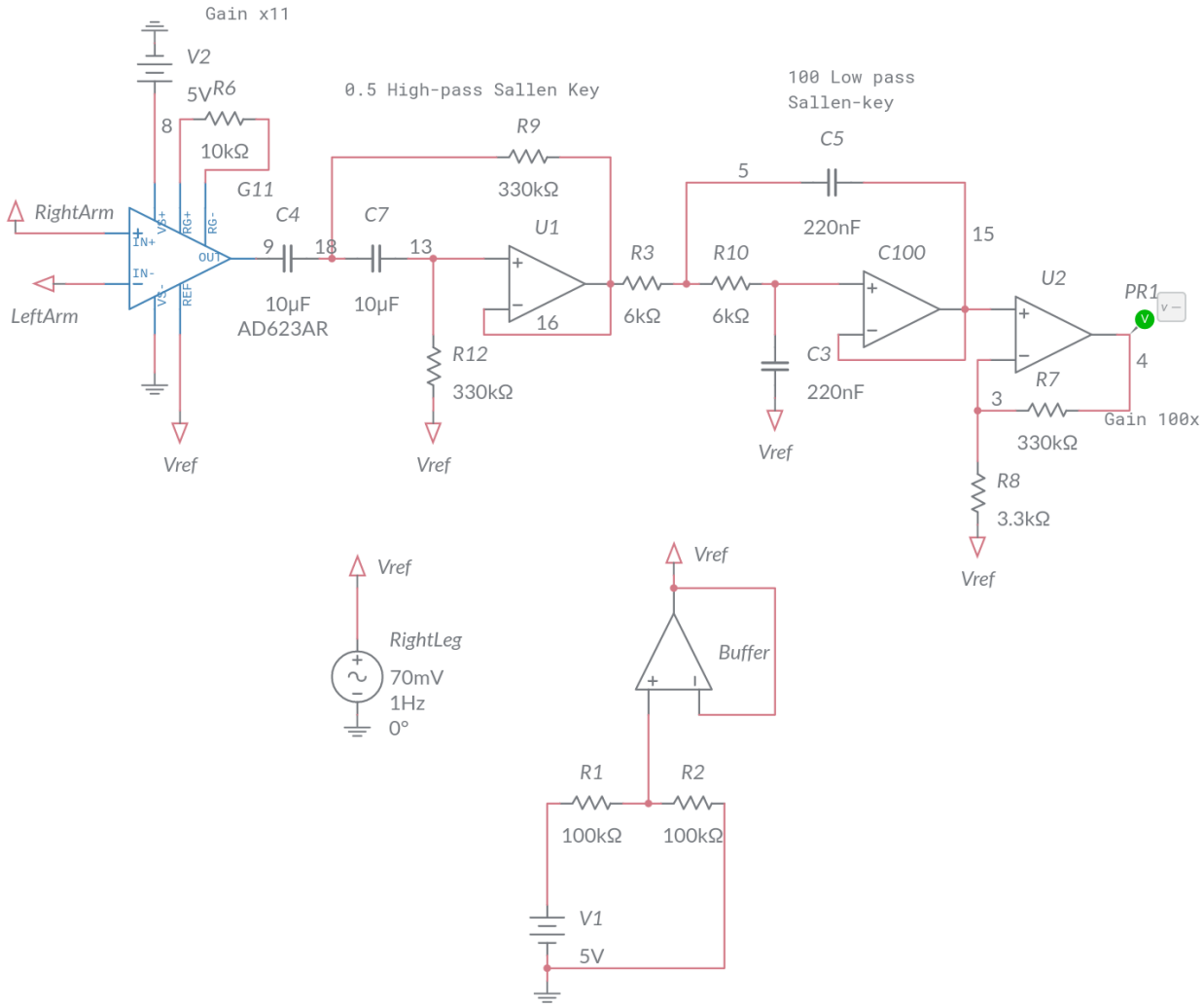

Figure 2: Circuit diagram

Listing 1: Arduino code

```
#include <avr/io.h>

int cyclesPerSample{79}; // 200 Hz = 5ms; 2x maximum frequency of ECG
int _adc{0};             // Private copy of ADC value. Will be switched with every print.
int shouldPrint{0};      // Controlled printing. requires truthy value for print

static inline void initSampler(void) {
    TCCR0B = 0b00000101;         // Prescaler: 1024
    TIMSK0 |= (1 << TOIE1);      // Overflow interrupt enable
```

```
10      TCNT0 = 2^8-cyclesPerSample; //Count 5 ms
11  }
12
13  static inline void initADC(void) {
14      ADMUX  |= (1 << REFS0);                // Set ADC reference voltage to external
15      ADCSRA |= (1 << ADPS1) | (1 << ADPS0); // ADC Prescaler: 8
16      ADCSRA |= (1 << ADIE);                 // ADC Interrupt enabled
17      ADCSRA |= (1 << ADEN);                 // ADC Enable
18      ADCSRA &= ~(1 << ADATE);               // ADC Autotrigger - Make sure it's off,
            otherwise it triggers without interrupts.
19  }
20  // ------ Interrupt Service Routine ------ //
21  ISR(TIMER0_OVF_vect) { //After cyclesPerSample have elapsed, this starts the ADC conversion
        and restarts the counting.
22      ADCSRA |= (1 << ADSC);    //Start Conversion
23      TCNT0=2^8-cyclesPerSample;//Reset timer
24  }
25
26  ISR(ADC_vect) { //Save current ADC value and allow a print
27      _adc = ADC;
28      shouldPrint=1;
29  }
30
31  int main(void) {
32      Serial.begin(115200);
33      noInterrupts();
34      initADC();
35      initSampler();
36      interrupts();
37       while(1) {
38          Serial.flush();
39          if(shouldPrint){
40              Serial.println(_adc);
41              shouldPrint = 0;
42          }
43       }
44      return (0);
45  }
```

References

Filters order 5. Wikipedia, December 2016. URL https://commons.wikimedia.org/wiki/File:Filters_order5.svg.