Patrick Harrington
Maxim Allen
ECEN 2020
Homework 5
October 15, 2013

1 Analog-to-Digital Conversion Techniques

To convert an analog signal into something digital and readable, the microcontroller samples the signal at a rate that is at least twice the frequency of the signal. When deciding what values can be represented from the incomming data, we can use $2^n - 1$ bits where n decides the total number of different 'levels' we can represent. if we had 1 bit total, we can represent 2 different levels, on or off. when we have $n = 8$, we get 127 levels that we can represent from the digital signal. the more bits we assign the better. Analog to digital conversion is a lot like approximating an integral in calculus, you can take a sample of what the data is at a given change in time and calculate the value at that point.

2 Voltage Division

Using `analogRead()`, the voltage across a pair of resistors $R_1 = 22[k\Omega]$ and $R_2 = 270[\Omega]$. From Equation (1), we have calculated the expected voltage from the circuit shown in Figure 1.

$$V_o = \frac{R_2}{R_1 + R_2} \cdot 5[V] = 60.6[mV] \tag{1}$$

We also tested again with two equal resistors found an expected drop of $2.5[V]$, since two resistors in parallel each drop half of the supplied voltage. We also noticed that the Arduino was less accurate with our original resistor values because the voltage dropped was so significant, that the tiny detected voltage was subject to considerable noise.

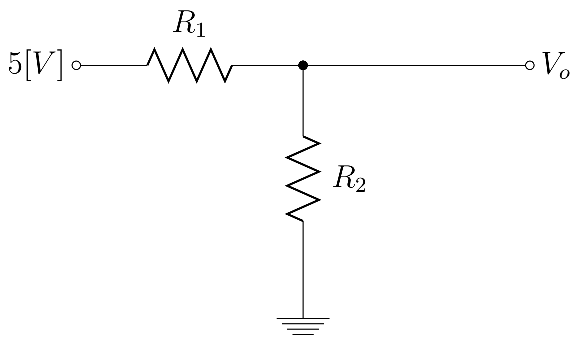

Figure 1: Voltage Divider Circuit

The code used to measure this quantity is attached below:

```
/****************************************************************
* Analog Reading
****************************************************************/

//  Global Variables

const int analogPin = 5;
volatile int value  = 0;     //  A value from 0 - 1023
volatile float voltage = 0; //  A voltage

// R1 = R2 = 12 kOhm
// Expected output voltage is R2/(R1+R2)5=60.6 mV
// Measured before plugging everything in was 60.9 mV

void setup()
{
  //  We'll need Serial to see the voltage
  Serial.begin(9600);
}

void loop()
{
  //  Voltage divider at analog pin 5
  value = analogRead(A5);
  //  analogRead returns values from 0-1023
  voltage = value*(5/1023.0);
  Serial.print(voltage);
  Serial.println("[V]");
  delay(2000);
}
```

3 Advantages of Working at the Register Level

This question asked about manipulating ports on a register level rather than writing their states with functions. This is beneficial because libraries can take up a lot of memory space. especially when you call them multiple times. the following code snippet calls digital write 4 times:

```
  digitalWrite(0, HIGH);
  digitalWrite(1, HIGH);
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
```

This takes up a lot more memory and processing power than `PORTA |= 0x0F;` this method requires no extra space for libraries and is very minimalistic in it processing requirements.

4 Time Elapsed since Program Execution

As directed, the following code includes use of the serial monitor and the Arduino libraries.

```
/****************************************************************
*  Runtime
****************************************************************/

volatile unsigned long time = 0;
volatile unsigned long check = 50;

void setup()
{
  Serial.begin(9600);
  Serial.println("Program␣Begin");
}

void loop()
{
  time = millis();
  //  at start of each loop, we check the time
  //  every 100ms print stuff
  Serial.print(((float)time)/1000,1);
  Serial.println("s");
  delay(100);
}
```

5 Double Blink

Functional code was developed, but this depended upon the libraries. Also included in the homework submission is a copy of an attempt to use ISRs to generate seperate clock signals, but one of the timers (TCCR2A) was problematic because it was an 8-bit timer.

```

const int led1 = 12;
const int led2 = 13;
// have we got values for the rates of each LED?
boolean val1 = false;
boolean val2 = false;
// rates to be chosen by the user
float freq1 = 10;
float freq2 = 2;
//——————————————————————————————————————
void setup(){
pinMode(led1,OUTPUT);
pinMode(led2,OUTPUT);
Serial.begin(9600);

//  get the rate for the first LED
if(!val1 && !val2){
  Serial.println("Please␣enter␣a␣value␣for␣the␣first␣LED");

  }
}
int get_num() { //  Get a number from the user
  char number[20];
  int index = 0;

  while(Serial.available() <= 0);
  delay(50);
  while(Serial.available() > 0) {
     number[index] = Serial.read();
     index++;
  }
  number[index] = 0;

  return atoi(number);
}


void loop(){
digitalWrite(led1,LOW);
digitalWrite(led2,LOW);
delay((int)((1/freq1)*1000));
digitalWrite(led1,HIGH);
digitalWrite(led2,HIGH);
delay((int)((1/freq1)*1000));
}
```

6 Signal Processing

(a)

The period of this wave is $T = 6ms$, so it is oscillating at $f \approx 166[Hz]$. thus, we need to sample it at at least 332 hz to get a good reading.

(b)

If you only checked this signal once per second, you are skipping 165 cycles of possible data change. The resultant sampled signal would be unrecognizable!

(c)

Aliasing in signal processing is when an analog signal is converted to digital and reconstructed using the digital values. during this reconstruction, data points are rounded off or grouped together making the reconstructed signal rough in comparison to what the actual signal is. Aliasing also happens when the signal isnt sampled fast enough so some data points are completely neglected. An aliased signal may appear completely different than the actual signal being sampled.

(d)

Using the same logic from Equation (1), $10[k\Omega]$ is the required R_2 value. When voltage dividing into half the voltage you have, both resistors must be equal.

(e)

Protecting the arduino from high voltages involves a voltage divider. Protecting the arduino from large currents is a little bit tougher, but can be done with just a current divider. The equation for current division is shown in Equation (2).

$$I_o = \frac{G_x}{G_{total}} \cdot I_t \tag{2}$$

Where $G_x = \frac{1}{R_x}$ and G_{total} for n resistors is defined as:

$$G_{total} = \sum_{1}^{n} \frac{1}{R_i}$$