Department of Biomedical Engineering
BME403 Senior Design Report
Fall 2014

Phase Measurement Device

Team Members
Xiaoyan Tian
Zhexuan Zhang

Mentors:
Dr. Chun-Yuh Charles Huang

Institutions:
University of Miami College of Engineering
Department of Biomedical Engineering

Date:
Dec 17th, 2014

Table of Contents

1. Abstract

Oxygen measurement is important in biomedical laboratory experiment and clinical practice, since oxygen concentration can represent some essential physiological parameters (such as glucose, ATP, etc.) with the help of certain enzymes. The ultimate goal of our project is using a specifically designed circuitry and algorithm to measure the oxygen concentration in real time. Oxygen concentration has a direct relationship with the quenching decay-time between oxygen and Ruthenium complex. A microprocessor with Analog to Digital Converter can extract the time signal, so that the oxygen concentration can be calculated with pre-defined algorithm. We have built up a phase measurement device, which can measure the decay-time and display the data on an Liquid Crystal Display. Capacitor has an exponential decay-time, which is similar to the quenching decay-time. So we use capacitor as our model for the quenching decay-time due to time constrain. The final goal will be implemented in next semester.

2. Objectives

2.1 General Objective

The purpose of this project is to develop a real-time oxygen concentration measurement device. Since the oxygen concentration can be linked to other physiological parameters via specific enzymes, the device can also measure other material's concentration indirectly (by specific calibration). Currently, concentration can be determined by test strip or titration. The disadvantages of those are obvious. Test strip is not accurate enough, and titration is time costing. Also, other devices that utilize the same decay-time principle are too expensive for general application. The commercial price for a typical phase measurement device, *NeoFox* from Ocean Optics, can be up to 3,000 US dollar. And the data measured by *NeoFox* PMD can only be displayed on a computer with Microsoft operating system.

Overall, several problems exist in current concentration measurement techniques. Some cheap methods are time consuming and with low precision; High accuracy and real-time device could be too costly for common use; And, almost all methods are not portable.

An ideal PMD should achieve five objectives. First, the phase measurement device (PMD) should be able to detect and extract the decay-time signal accurately; second, the PMD should be able to analyze and display the data by itself; third, the concentration determination should be fast enough for real-time monitoring; fourth, the PMD should be portable; fifth, the PMD should be cost-efficient.

2.2 Expected Problem Identification documentation

Problem 1: Data Acquisition

The purpose of the project is to measure substances' concentration by measuring concentration of oxygen. We adopt the principle that the concentration of oxygen can be determined by the decay-time of Oxygen-Ruthenium quenching. The signal can be extract by a microprocessor with Analog to Digital Converter (ADC).

Problem 2: Precession

Since we use a microprocessor as the function part of our project, the quality of the data is greatly depend on the number of bit of the ADC. More number of bit means higher resolution of the signal that we can acquire, and also more expensive. For our case, an 8-bit ADC balances the precession and cost-efficiency.

Problem 3: Time consuming

In order to observe the concentration change under certain conditions, the data acquisition and processing should be fast. The decay-time of Oxygen-Ruthenium quenching will only take several microseconds, so does the processing time of microprocessor.

Problem 4: Portability

The function of phase measurement device is to measure the decay-time, and calculate the concentration accordingly. This process can be implemented by a microprocessor, and also the data can be displayed on a LCD screen connected serially to the microprocessor.

3. Background

The measurement of oxygen concentration in solution, air, and tissue sample is important in biomedical research. Oxygen-sensing systems have also been utilized for monitoring physiological parameters, such as glucose, ATP[1]. Because the biochemical reaction of these substance causing either the production of oxygen or consumption of oxygen. By monitoring the altered oxygen concentration, the concentration of biological parameters could be determined. A lot of researches have been done to study the method of oxygen concentration determination. Researchers found that some transient metal can quench with oxygen, and emit luminescence. Based on the intensity of the light, the oxygen concentration can be determined. Luminescence quenching methods of analysis are based on difference in the emission intensity of luminescence lifetimes in the presence of oxygen and in the absence of oxygen. When oxygen quench with transient metal complex, the complex enters an excited state, and begins to emit luminescence[2]. After an amount of time, called lifetime, the emission of light will come to a steady excited state. If luminescence quenching is entirely diffusional, the excited-state lifetimes or luminescence intensities are related to the oxygen concentration by Stern-Volmer equations:

$$\frac{\tau_0}{\tau} = 1 + K_{SV}[O_2] \qquad \text{Equation 3.1}$$

$$\frac{I_0}{I} = 1 + K_{SV}[O_2] \qquad \text{Equation 3.2}$$

$$K_{SV} = k_2\tau_0 \qquad \text{Equation 3.3}$$

In the above equations, τ_0 and I_0 are luminescence lifetime and intensity in the absence of oxygen; τ and I are luminescence lifetime and intensity in the presence of certain oxygen concentration. K_{SV} is the Stern-Volmer quenching constant, which has a specific magnitude for each materials combination, and k_2 is the bimolecular quenching constant[3].

Researchers utilize the luminescence quenching theory in some biological parameter detection with certain biosensors. A reliable ATP biosensor was developed by C.Wang et al in University of Miami[4]. The basic principle is to relate the ATP concentration with oxygen concentration, and to extract the luminescence signal for analysis.

ATP + Glycerol $\rightarrow$ Glycerol-3-phosphate + ADP Equation 3.4
Glycerol-3-phosphate +O_2 $\rightarrow$ Dihydroxyacetone phosphate + H_2O_2 Equation 3.5

The two steps above are happened under catalyzation of glycerol kinase. During this chemical reaction, the consumption of oxygen alters the oxygen concentration in the solution; thus, the concentration of ATP can be determined by the concentration of oxygen.

The biosensor was a needle type sensor (Figure 3.1). Ruthenium complex (transient metal complex which can quench with oxygen) and enzyme were coated on the tip of the needle. An optical fiber was inserted into the middle of the needle for luminescence transmission. The needle tip was inserted into a solution with ATP. ATP on the tip was catalyzed and consumed oxygen. The consumption of oxygen influenced the luminescence quenching with Ruthenium complex, which could be transmitted through the optic fiber to the phase measurement device. They used a commercial phase measurement device to measure the luminescence decay-time, and the result confirmed that oxygen-related biological parameters could be tested through the quenching decay-time.

The laboratory experiment inspired us to develop a similar phase measurement device based on the same principle, but with a simpler function so that it can be cost-efficient. Also, since the luminescence quenching decay is a single exponential function, we can use simple RC circuit to test the viability of our device.

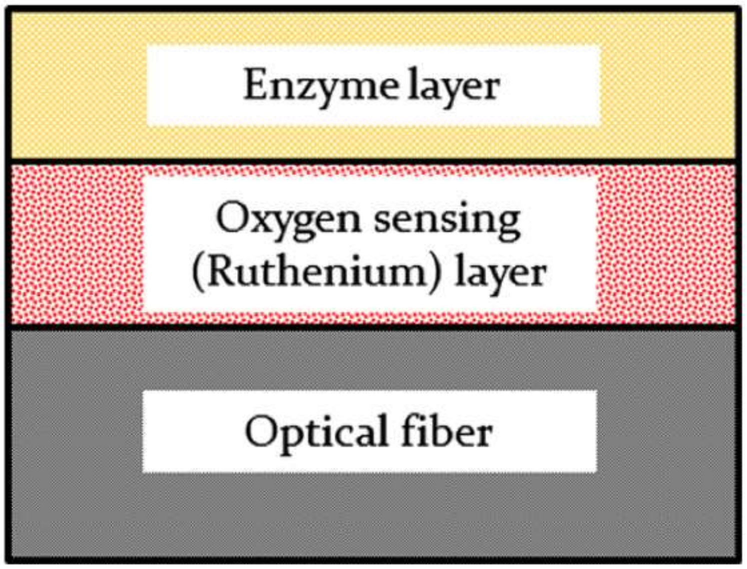


Figure 3.1 ATP Biosensor. Enzyme and Ruthenium complex are coated on the needle tip; the optical fiber is inserted in the middle of needle.

4. Design specifications

MoSCoW Prioritisation			
Project Name:	Phase Measurement System		
Project Manager:	Xiaoyan Tian, Zhexuan Zhang		
Project Description:	A description of the project, or work package to be undertaken.		
	The project aims to develop a phase measurement device based on Oxygen-Ruthenium quenching. The device is able to measure the decay-time of the quenching, calculate the value of oxygen concentration, and display it on a LCD screen.		
Limitations:	Time:	Budget:	Resource:
	12/01/2013-12/17/2014 **381days**	**$123.66**	**Stem Cell and Orthopaedic Bioengineering Lab** **Advisors:** **Dr. Chun-Yuh Huang** **Dr. Nelson Salas**
M(ust have):	What must be delivered, i.e. it is essential for this phase?		
	Capacitor decay-time (model for Oxygen-Ruthenium quenching) must be acquired by this device properly. Capacitor decay-time determination algorithm must be implemented by the microprocessor. Capacitor decay-time must be displayed on the LCD screen.		
S(hould have):	What should be delivered as a high priority but not essential?		
	The device should be able to acquire data from Oxygen-Ruthenium quenching. Oxygen concentration determination algorithm should be effective.		
C(ould have):	What could be delivered if there was available time / budget / resource?		
	Calibration system could be incorporated into the microprocessor. The whole device could be integrated on a single specially designed chip. More predefined parameter for other substances' concentration could be included in the device.		
W(ould have):	What would be have if time / budget / resource was unlimited?		
	Touch screen would be more user-friendly. Higher resolution would be achieved by a more expensive microprocessor. A corresponding software would be developed for personalize the device (i.e. user can define parameters for some substances, which are not included originally in the device.)		

Specification	Acceptable	Desired
Resolution	0.1μs	0.001μs
Processing time	5s	Instant
Portability	Portable	Portable

The difference between acceptable and desired value of resolution is mainly due to budget constrain. And according to our observation in laboratory, 0.1us is sufficient enough for glucose concentration measurement. The 5-second processing time is caused by the sample averaging algorithm for error reduction to achieve high stability. Also 5-second processing time also provide a sufficient time window for displaying data on the LCD screen for user to record the data.

5. Design constraints

Electronics are involved in the Phase Measurement Device. The device would be integrated in a single chip for mass production. Safety protocol related to electrical devices should be followed. Since the device mainly consists of electronic component, it should never be operated in high humidity environment, should never be operated in high temperature, and should never be opened without authorization to avoid the hazard of short circuit and explosion. And the depreciated device should be recycled by specific company to avoid contamination to the environment as it contains battery.
A similar device (*NeoFox*, Ocean Optics) exists on the commercial market. So it should go through the FDA 510(k) clearance before putting into market.

6. Design Development

The principle of oxygen concentration detection is Oxygen-Ruthenium quenching. The quenching produces an exponentially decaying light signal that can be picked up by a high sensitive photodiode with low response time. The decaying light signal is then converted to an electric current, which has the same characteristic decay time with the light decay. By measuring the decay time of the electric current produced by the photodiode, the concentration of oxygen can be determined. Due to time constrain and limitation of our microprocessor and photodiode, the oxygen measurement device cannot be built in this semester. Alternatively, we use several capacitors to model the Oxygen-Ruthenium quenching, since the capacitor can also produce an exponentially decaying electric signal. By using capacitor, the measurement algorithm can be tested and improved. And we developed a new theoretical method to measure the signal.

Phase1: Microprocessor-weighted Method

The microprocessor we are using is a ATmega32u4 based module (called *Leonardo*), designed by Arduino. Arduino has developed a widely used C++ library for people who work on their Arduino product. The library is for general purpose, and the microprocessor on the module can also be directly programmed with C++ language. An understanding in the ATmega32u4 datasheet would greatly help with the more advanced programing.

Analog to digital converter (ADC) cannot record the electric signal continuously. The Arduino library offers it’s own ADC function called *analogRead()*, which has a 10kHz sampling

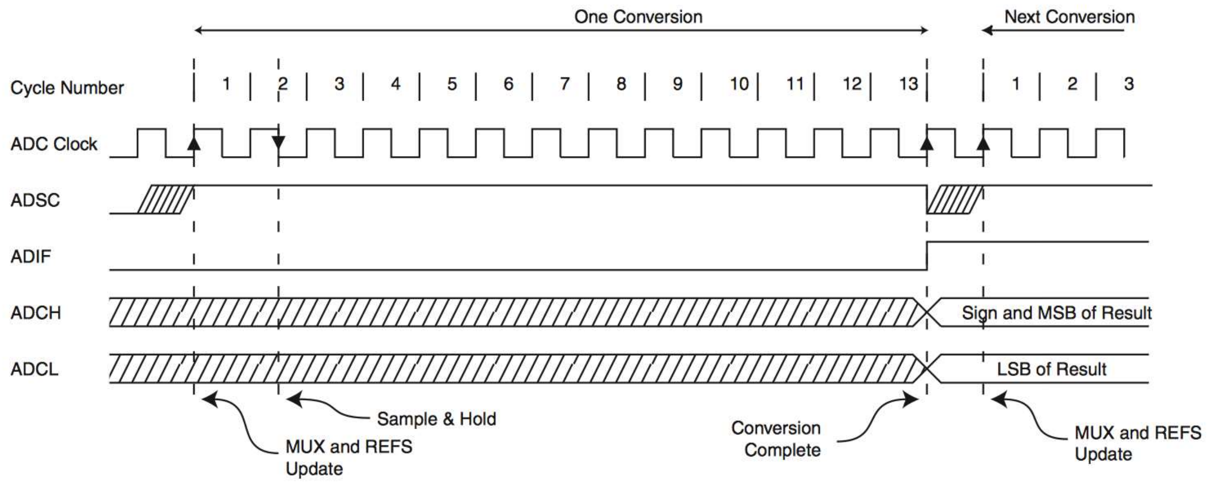


Figure 6.1 A normal analog-to-digital conversion. *Sample&hold* happens at 1.5 ADC clock cycles.

frequency(100 microseconds per cycle). The decay-time of O_2-Ru quenching is microsecond level (around 1-5 microseconds, MHz level). The analogRead() function won't be able to catch any decay signal. But the microprocessor on the Arduino board can be overridden to accelerate its sampling frequency by using C language to directly access it. The highest sampling frequency of the ADC can be used is typically 77.8KHz (13 microsecond per cycle). According to the Nyquist Sampling Theorem, the sampling frequency should be at least twice of the signal frequency, so that the full behavior of the signal can be recovered. But clearly, 77.8kHz is not enough to recover the full behavior of the signal. We further studied the microprocessor data sheet, and we found out the working principle of the ADC.

A normal analog-to-digital conversion takes 13 ADC clock cycles. The shortest ADC clock cycles can be adjusted to 1 microsecond (μs) without compromising its accuracy. In other words, a normal ADC takes 13μs to fulfill a valid conversion. We notice that the 13μs consists of three actions. First action is multiplexer and reference update; second action is *Sample&Hold*; third action is conversion (Figure 6.1). The most important action is the *Sample&Hold*, which is the moment that the microprocessor sample the analog signal and hold it for conversion, and it takes place exactly at 1.5μs from the beginning of the conversion. 1.5μs is a great improvement comparing to 13μs, but it's still greater than 1μs (Nyquist frequency).

After a discussion with our mentor, Dr. Huang, we concluded that we don't have to follow the Nyquist Sampling Theorem, since the Nyquist frequency is only used for recover the full behavior of the analog signal. In our case, the behavior of the analog signal is already known --- an exponentially decaying signal ---, so we don't have to use such a high sampling frequency to determine the decay time. We only need to extract two points from the analog signal, which are a signal before decay, and a signal just after the decay (Figure 6.2). Once the signals have been extracted, and the time period between them are known (1.5μs), the decay time can be calculated by Equation 6.1, and Equation 6.2.

$$V_t = V_0 \cdot e^{-\frac{t}{\tau}} \qquad \text{Equation 6.1}$$

$$\tau = \frac{-t}{\ln(\frac{V_t}{V_0})} \qquad \text{Equation 6.2}$$

In the equation, V_t is the signal at 1.5μs after the beginning of the decay; V_0 is the stable signal before decay; t is the moment at which the ADC pick up the analog signal, 1.5μs; τ is the decay-time constant, which is the key parameter in concentration determination.

The theoretical method for extracting the signal has been developed, but we still need to consider how to produce a series of this decaying signal for further analysis (i.e. averaging signal to reduce noise). Considering the decay time of O_2-Ru quenching is around 1-5 microseconds, as long as the time interval between each pulse is greater than ten times the decay, which is 50 microseconds, the decay can be extracted by the ADC. A pulse frequency if 30Hz (60Hz for on/off switch) is a good choice since it can be easily done with Arduino language, and averaging all samples in a 5 second time window to reduce the noise. So the next step is to develop a steady pulse generator with the Arduino.

The *Leonardo* (microprocessor we are using) has a built-in time function in its programming library. Our first thought was to implement the frequency pulse with the Arduino library. The Arduino time function is more like a general function, which means that it can be used very conveniently in relatively large time scale (miliseconds to hours), but works poorly in small time scale (microseconds). The error introduced by the Arduino time function is about 4 microseconds. Considering the decay time is only 1-5 microsecond, the 4-microsecond error is unacceptable,

Again, by reviewing the ATmega32u4 datasheet, we found a highly reliable function of the microprocessor called *Timer/Counter*. The Timer/Counter is basically a delicate timing function, which can be access by basic C++ language (not Arduino). The timing is based on a 16MHz crystal oscillator; in other words, the resolution of the Timer/Counter is 0.0625μs. Since we only need a 60Hz switch frequency, we decided to set a 8-presale multiplexer to the *Timer/Counter*, and make the resolution to be 0.5μs. The purpose of the multiplexer is to reduce the power consumption and increase the stability of the crystal oscillator. Then we set the *Timer/Counter* to switch the polarity of the outlet at 60Hz frequency to create a 30Hz pulse signal.

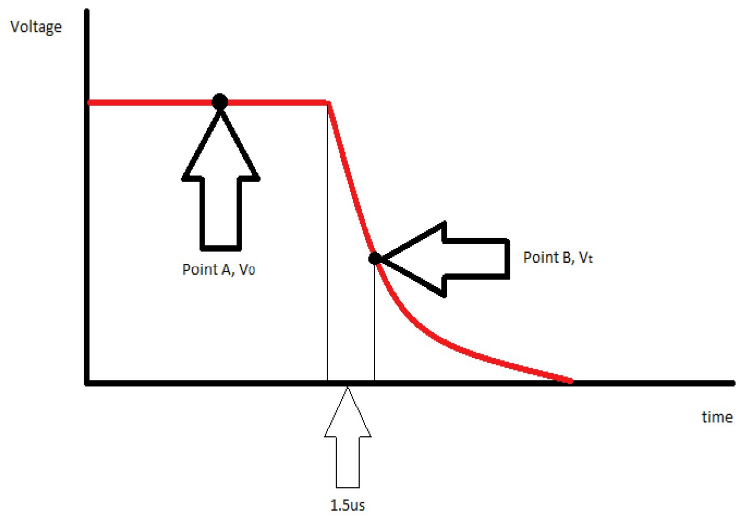


Figure 6.2 The method to determine decay

The last step before finishing the prototype concept is to connect a liquid crystal display (LCD) to the microprocessor for data display. Arduino has a LiquidCrystal Library, which allows us to control the LCD compatible with the Hitachi HD44780 driver. The process of controlling the display involves putting the data of what we want to display into the microprocessor's data registers, then putting instructions in the instruction register. The Arduino LiquidCrystal Library simplifiers these steps so that the code is much simpler. The connection of LCD follows a circuit specifically designed for Arduino microprocessor module (Figure 6.3).

The design has been conceptually developed. Three main parts have been included: data acquisition, pulse generating, and data display. The microprocessor uses its *Timer/Counter* unit to generate a series of electrical pulses, and sends to the subject to stimulate corresponding decay signals. Then the decay signals are picked up by the built-in Analog-to-Digital converter with specially designed algorithm. After calculation, the decay time should be properly displayed on the LCD screen. Implementation of the concepts should be resulting in a desired prototype (Figure 6.4).

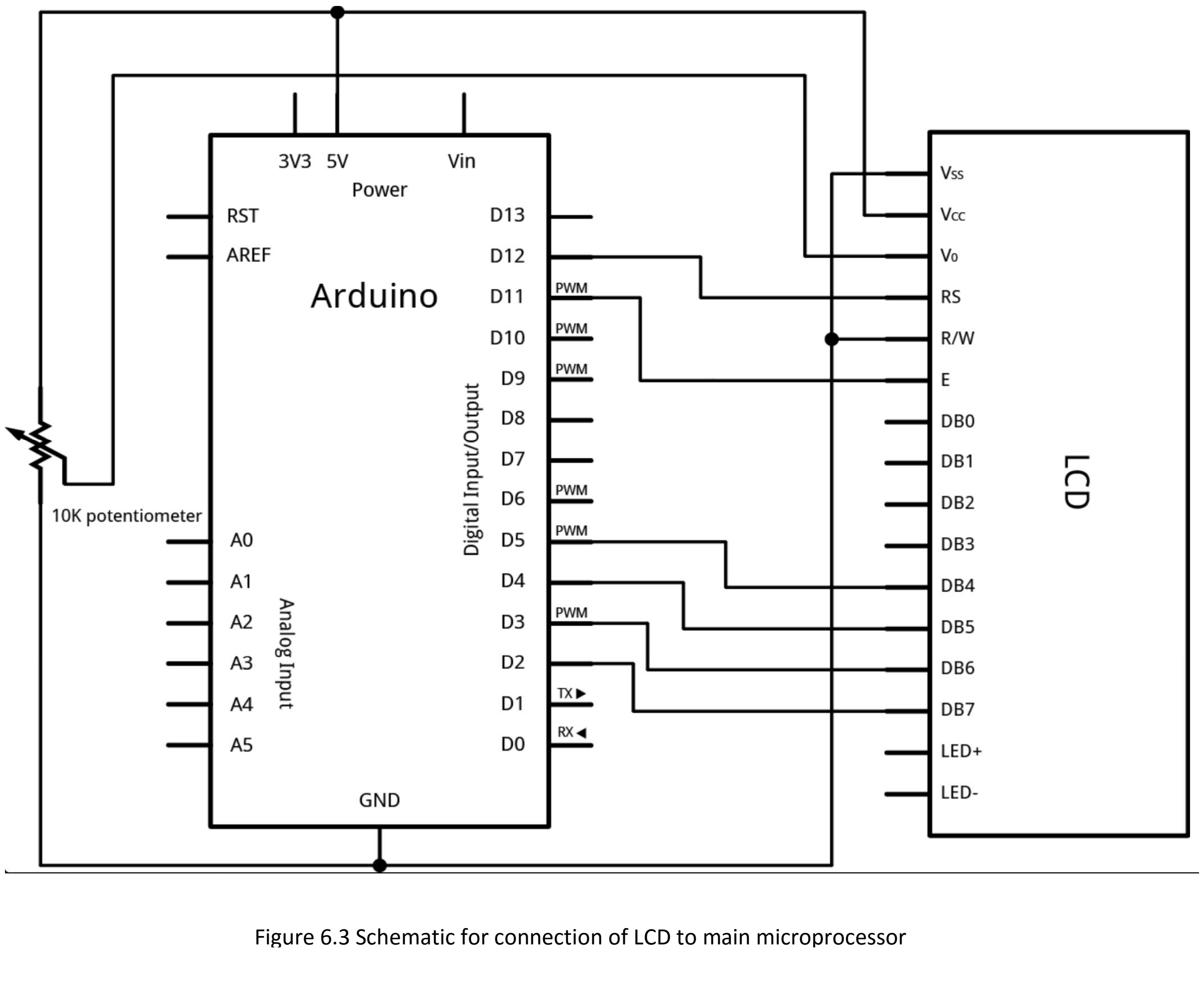


Figure 6.3 Schematic for connection of LCD to main microprocessor

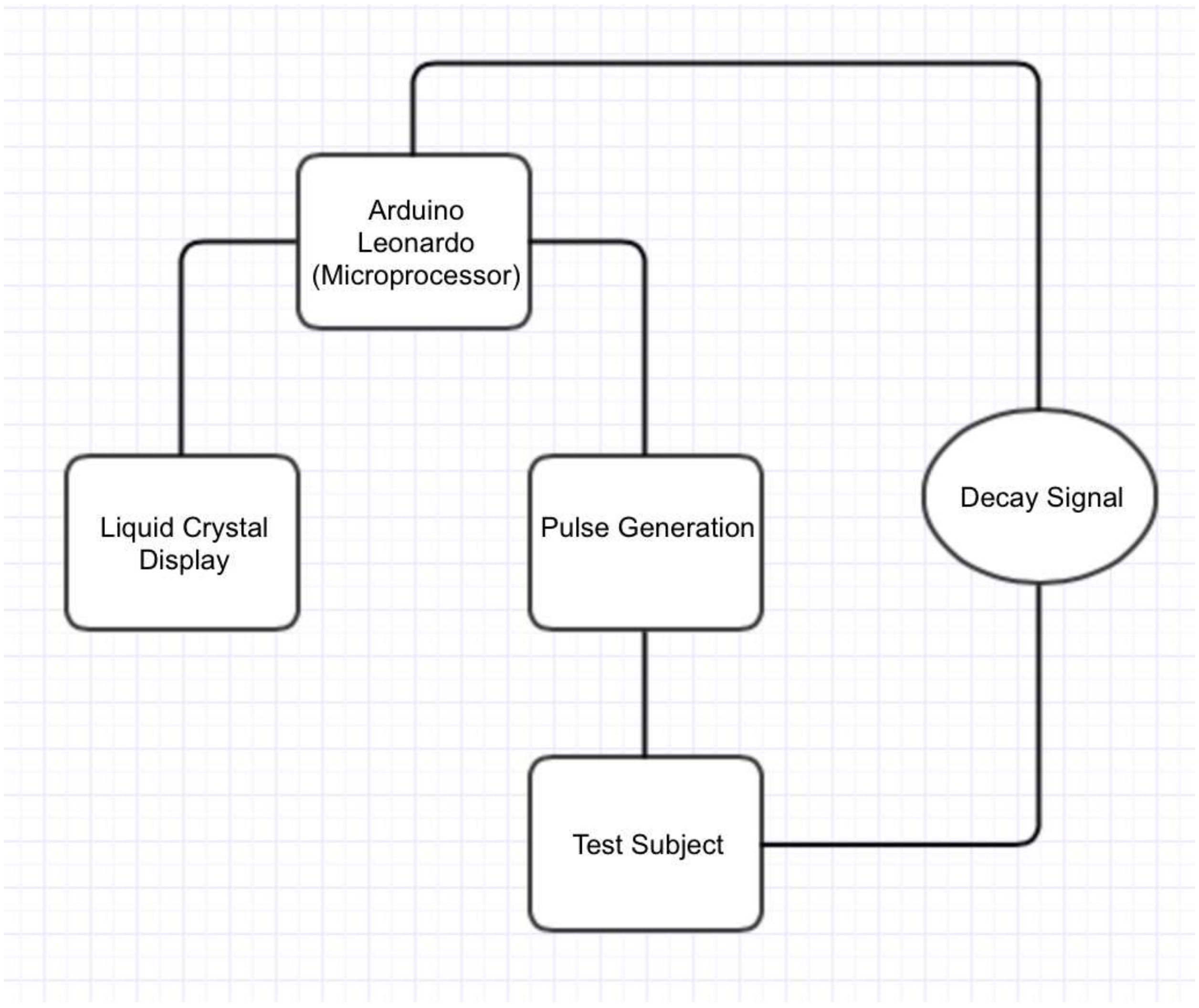


Figure 6.4 Block Diagram for conceptual prototype.

RC circuit has been used as our test subject to simulate the decay of orange light(O_2- Ru quenching). The test results shows that our algorithm is viable. And phase 1 development has completed.

Phase 2

A short-wavelength type Avalanche photodiode (Hamamatsu APD module C12702) has been provided by Dr. Huang to be our light sensor. By reading the datasheet, we found out that it has a 4KHz high-pass filter, which is used to eliminate environmental light noise. But in our case, it also eliminates our 30Hz light signal. A simulation on Matlab is done, and shows how the APD distorts our signal (Figure 6.5). In microsecond level (Figure 6.6), it seems that there is some correlation between APD's output and orange light, but the relation is obscure, since the datasheet does not provide us the high pass filter detail, also Arduino cannot measure negative voltage.

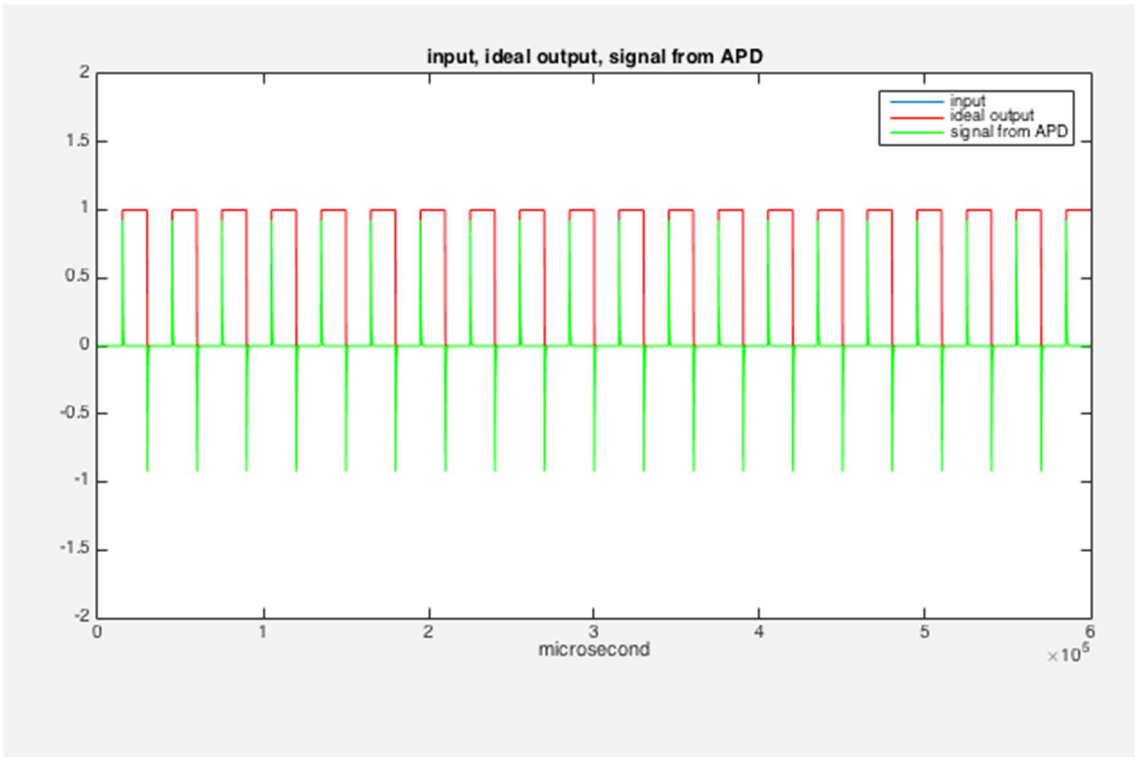


Figure 6.5 MatLab simulation of how the HPF on APD distorted our signal. Blue line is our input blue light signal, Red line is the orange light coming back from probe, the Green line is the APD's output signal when it receives orange light.

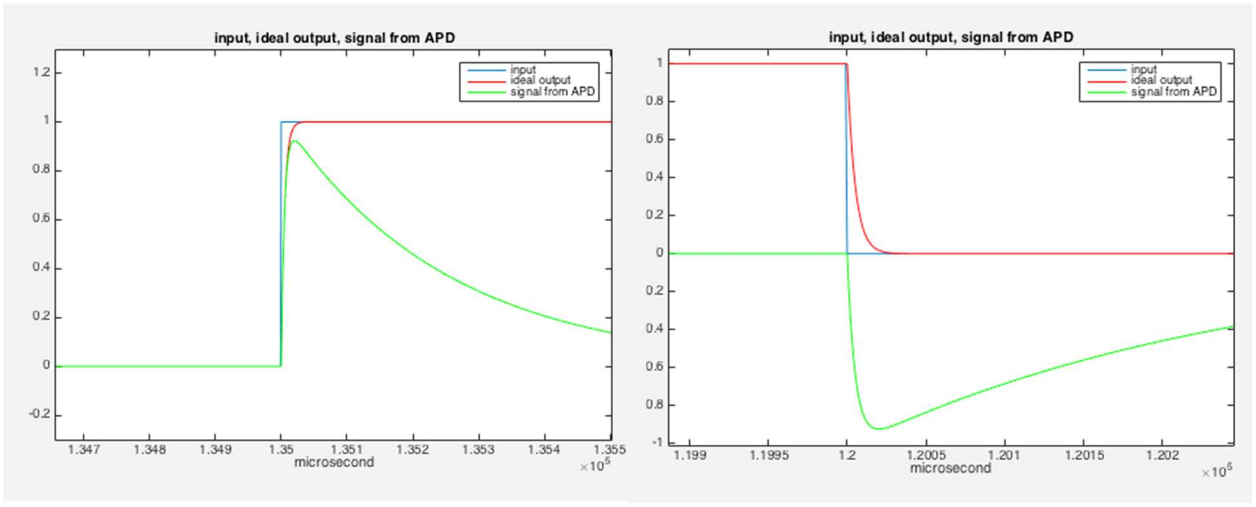


Figure 6.6 MatLab simulation of how the HPF on APD distorted our signal (microsecond level). Blue line is our input blue light signal, Red line is the orange light coming back from probe, the Green line is the APD's output signal when it receives orange light.

After consulting with Dr. Bohorquez, he suggested if we could use high frequency light impulse to accommodate the APD. We used function generator to test the frequency level when the APD can fully recover the light signal it received. The results showed that at hundred-kHz level, the APD can recover the light signal without distortion (figure 6.7).

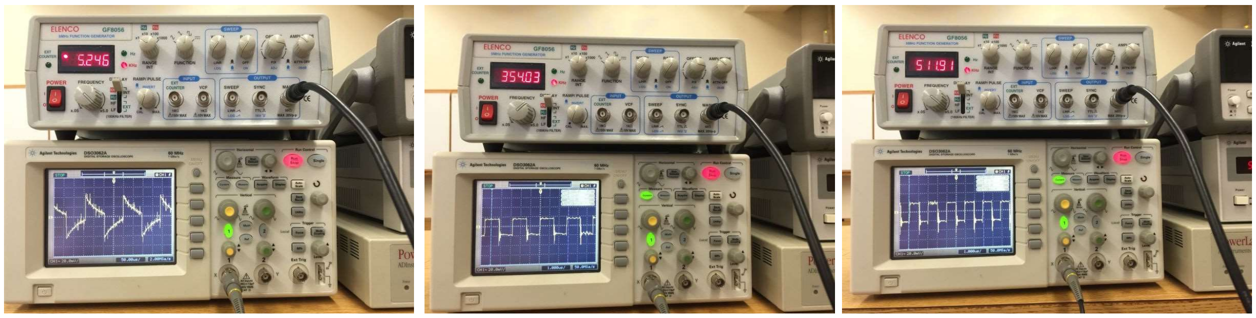


Figure 6.7 Test frequency response of APD. Left is at 5KHz, middle is at 300kHz, right is at 500kHz. From the picture, hundred-KHz frequency should be sufficient for APD to recover signal without distortion.

Again, by using MatLab simulation, we can predict the behavior of quenching and APD response under 100kHz light pulse (Figure 6.8). The APD can fully recover the light signal it received. Since the time interval between each light pulse is less than the time light signal stabilized, so the response of APD is a triangular waveform. If the decay of the light is short, than in each time interval, it will rise and decay much more than the long decay signal, which means that by measuring the amplitude of the triangular wave, we can determine the decay time.

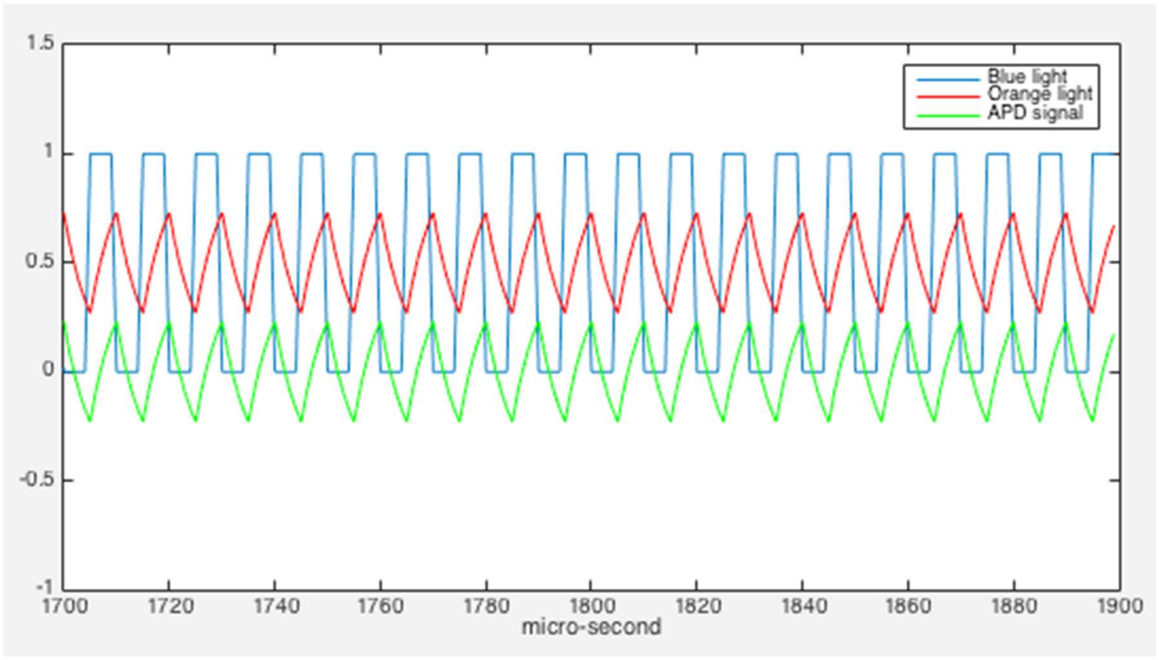


Figure 6.8 MatLab simulation of APD response in high frequency light pulse. Blue line is blue light sending out to probe; Red line is orange light that received by APD; green line is APD's output signal. In this frequency, APD can recover full behavior of the light signal it receives.

For the purpose of amplitude measurement, a rectifier and a low pass filter can be implemented after the APD output, so that the triangular signal can be soothed to a DC signal (Figure 6.9). The magnitude of the DC signal is related to the decay time.

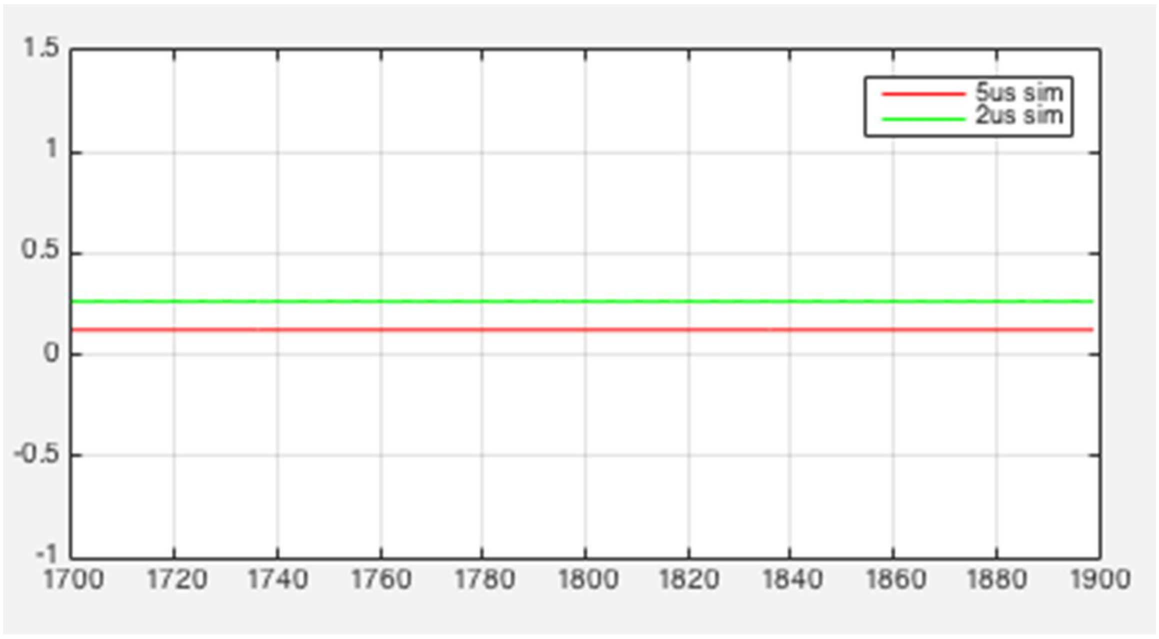


Figure 6.8 MatLab simulation of LPF. Green line is the DC signal for 2us quenching decay time, red line represents the DC signal for 5us quenching decay time

7. Proof of concept and Test Result

The prototype consists of two main parts, circuitry and program. Since the prototype should fulfill three functions, which are pulse generation, decay-time detection, and LCD connection, we broke down the construction into three stages, and each stage a program along with its corresponding circuitry was implemented.

First stage was LCD connection and programing. This connection was done by following the instruction shown in Figure 6.3. And the code for LCD was easily set up with the help of Arduino LiquidCrystal Library (Figure7.1).

```
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int  lcd_start()
{
// set up the LCD's number of columns and rows:
  lcd.begin(16,2);
  // Print a message to the LCD.
  lcd.print("hello, world!!");
  //pinMode(ledPin, OUTPUT);
}
```

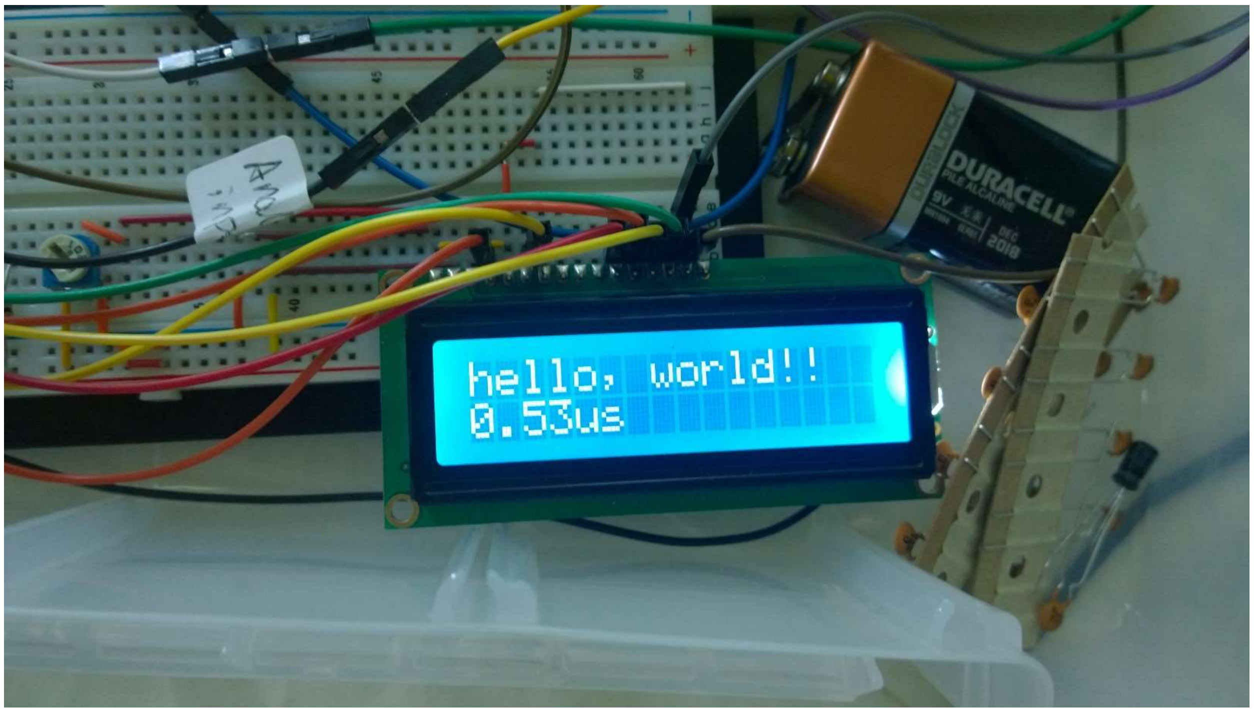


Figure 7.1 LCD display on breadboard

Second stage was pulse generation and programing. The Timer/Counter function has been adjusted to work at 2MHz (0.5μs time interval) by setting the prescale to be 8. Since a 30Hz pulse was needed in the prototype, the lead polarity should be switching at 60Hz (The on/off switch action should be twice the frequency as the pulse frequency). The observation of pulse generation was done visually and experimentally. Once the pulse generation code has been completed, we connected a light emitting diode (LED) to the pulse lead. A blinking LED showed that the coding was working (Figure 7.2). Also we used an Oscilloscope to monitor if the pulse was well formed. The result showed that the pulses produced by the microprocessor are perfect

```
DDRB  |=  (1<<5);//setting Pin 9 as LED output
TCCR1B  |=  (1<<CS11); //prescalar 8, initializing Timer/counter1
for (;;)
{
int LED_flag = 0;□ //flag for LED status. 0 means no light, 1 means lighting up.
TCNT1 =0;□ //Clear Timer/Counter cache
while (LED_flag ==0){ //LED dark
if (TCNT1 >= 33332){ //counting
PORTB |= (1<<5);  //LED light up
TCNT1 = 0; //Reset timer value LED_flag = 1;
} }
while (LED_flag == 1){ //LED light up
if (TCNT1 >= 33332){
PORTB &= ~(1<<5); //LED shut down
TCNT1 = 0; //reset timer value
} }
}
```

square waves, with respond time less than 5 nanosecond.

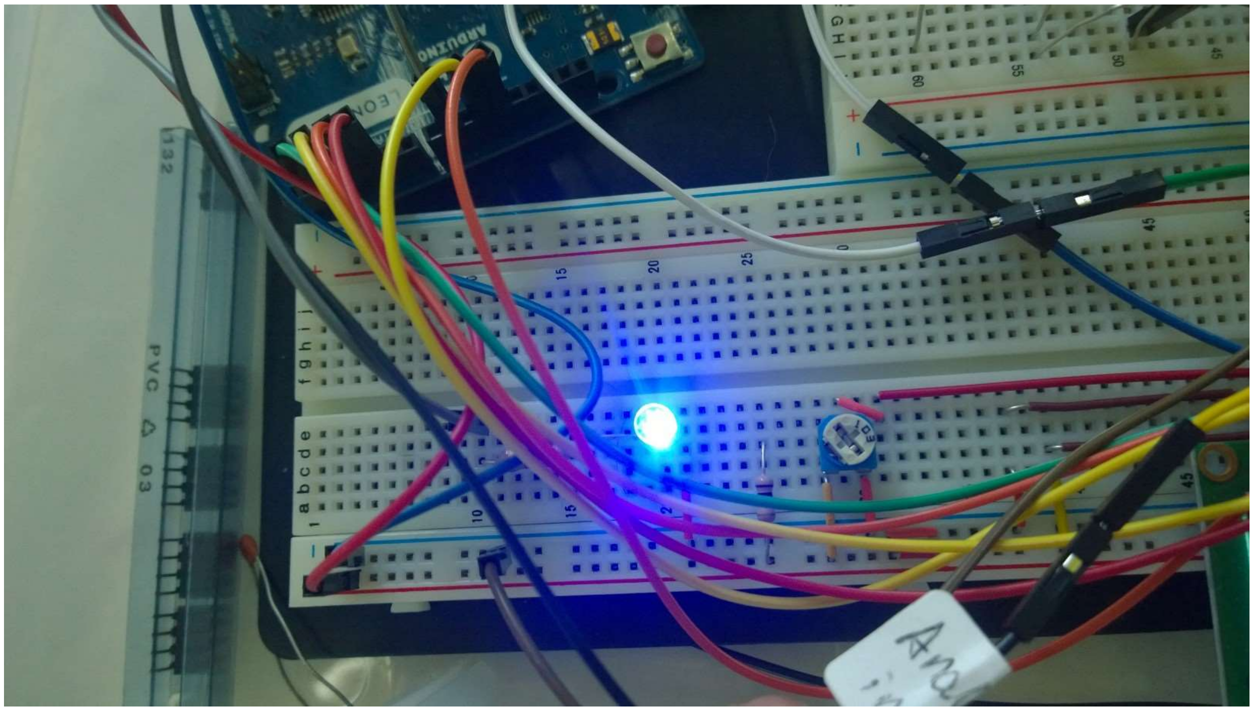


Figure 7.2 LED blinking showing the pulse generator works properly.

Third stage was signal detection programing. This part only involved programing. The analog-to-digital converter has been adjusted to work at 1MHz, with an ADC clock equals to 1μs. Also, the signal detection program was cohered into the Timer/Counter program, in order to extract the decay signal exactly at 1.5μs.

```
int  adc_decay_start() //setup ADC7(A0) with normal conversion
{
ADCSRA |= (1<<ADPS2); //prescalar=16; ADC clk freq=1MHz;//1 ADC clock = 1us;
ADMUX |= 0x47; //AVcc with external capacitor on AREF pin. right adjustment to
maximize //set ADC7 as input channel
ADCSRA |= (1<<ADEN); //ADC enable
ADCSRA |= (1<<ADSC); //ADC start conversion
}
```

Tests were taken to examine if the prototype can successfully display a reasonable decay time of the test subject. The test subjects were three capacitors; each capacitor was connected in a standard RC circuit (Figure 7.3 Figure 7.4). The decay time displayed on the LCD screen were recorded for further analysis (Table 7.1, Figure 7.5).

The results of the tests showed an acceptable conclusion. Each capacitor has a one-to-one relationship with each decay-time displayed on the LCD screen.

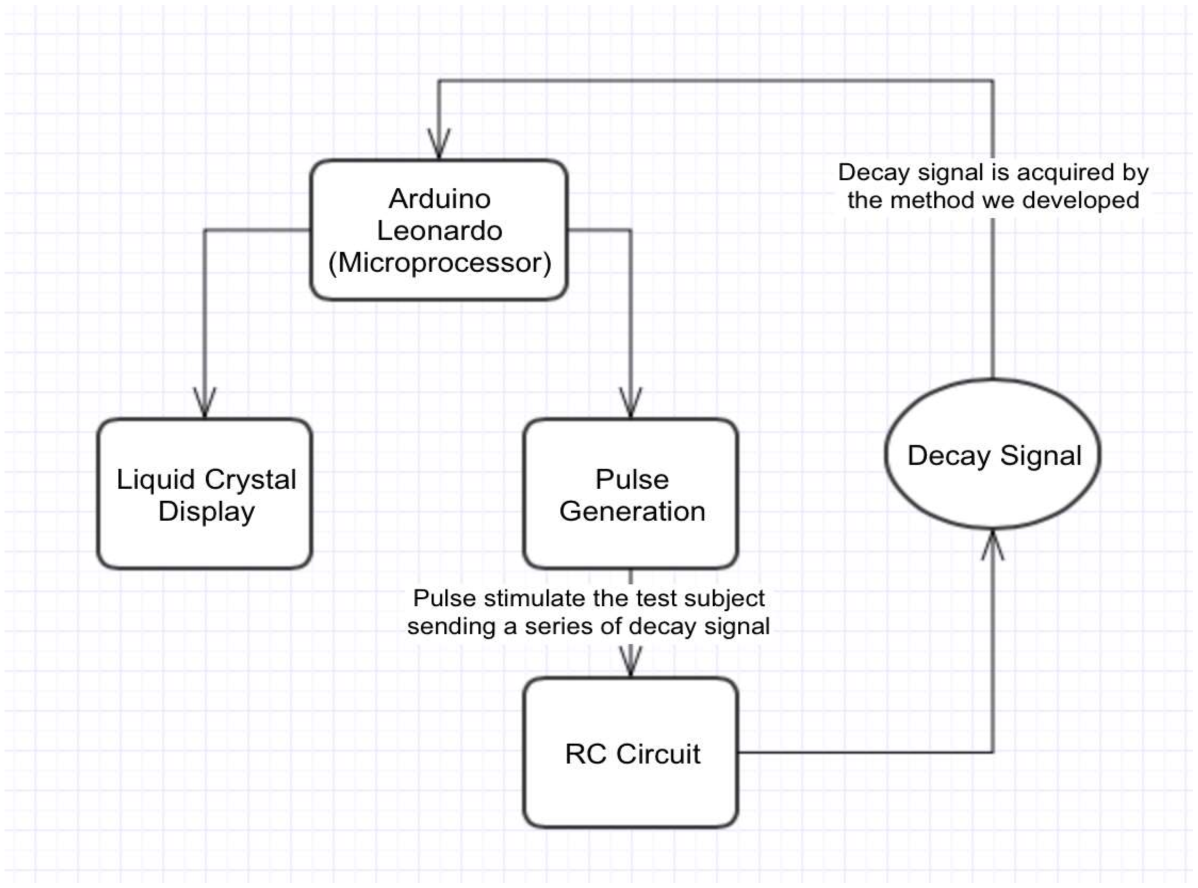

Arduino Leonardo (Microprocessor)
Liquid Crystal Display
Pulse Generation
Decay Signal
Decay signal is acquired by the method we developed
Pulse stimulate the test subject sending a series of decay signal
RC Circuit

Figure 7.3 Schematic for Prototype. Test subject is now RC circuit.

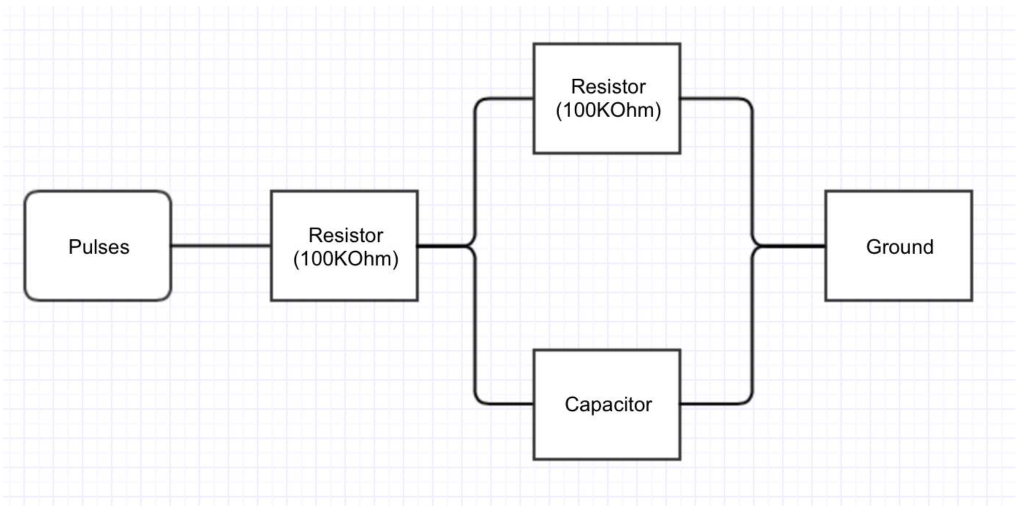


Figure 7.4 Circuit for RC connection

Table 7.1 Capacitance (pF) V.S. Decay Time (μs) Displayed on LCD screen.

Capacitance	100pF	220pF	470pF
1	0.18	0.55	0.97
2	0.20	0.57	0.94
3	0.20	0.56	0.95
4	0.19	0.56	0.96
5	0.21	0.57	0.95
6	0.20	0.55	0.94
7	0.23	0.56	0.94
8	0.19	0.56	0.93
9	0.19	0.55	0.95
10	0.22	0.57	0.96
Average	0.201	0.56	0.949
Standard Deviation	0.01523884	0.00816497	0.01197219

The standard deviation of the result is around 0.01 microseconds, which is better than the resolution we planed to achieve. We also run student-t test between each groups(Table 7.2). The result shows that they're not paired. The 100pF and 470pF capacitor have a proportional relationship with their corresponding decay-time (Equation 7.1). And the 220pF capacitor also has a proportional relationship with its decay-time, but with a different coefficient (Equation 7.2).

$$DecayTime = 2 * \frac{Capacitance}{1000} \qquad \text{Equation 7.1}$$

$$DecayTime = 2.54 * \frac{Capacitance}{1000} \qquad \text{Equation 7.2}$$

In the above equation, the DecayTime is measured in microseconds, and Capacitance is in picoFarad.
The deviation of the coefficients in the two equations may be caused by the manufacture error in the capacitor. The capacitance of the capacitor may not be the same as its label, which is an common error for small capacitance capacitor production. The distortion if the decay-time is mainly due to electrical noise in surrounding environment (powerline, cell-phone, etc). A deviation around 0.01 microseconds is good enough, since the time interval is so small, any minor interference would distort the signal.

When we're doing Phase 2 experiment. We found out that there're two limitations due to the microprocessor and the APD. The microprocessor always randomly misses some waves when it tries to produce 100kHz square wave (Figure7.5). And the APD simply cannot detect the light signal.

Table 7.2 student t-test between each group

groups	100pF&220pF	100pF&470pF	220pF470pF
t-value	-65.7	-122	-84.9
std	0.0122	0.0137	0.0102
Probability they're paired	0	0	0

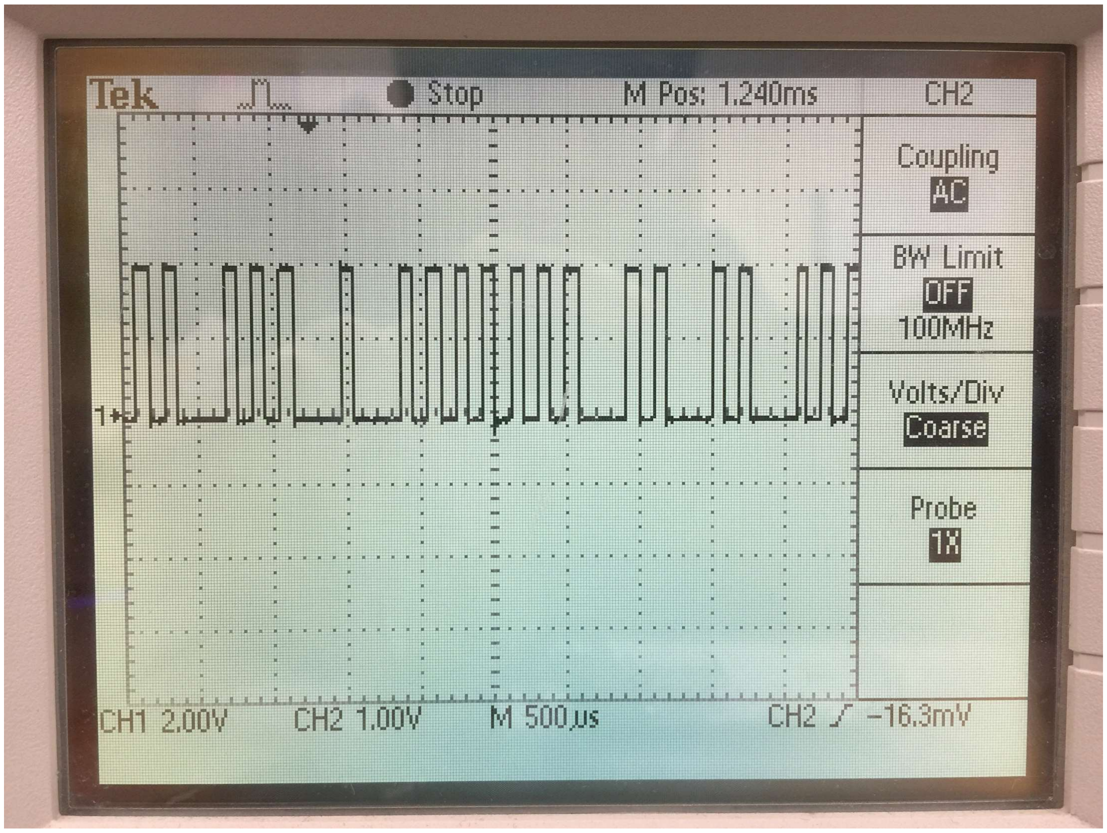


Figure 7.5 Microprocessor misses waves when trying to produce high frequency square wave.

A PARETO analyses (Table 7.3, Figure 7.6) was done to determine the critical parameters in the prototype construction. According to the 80/20 rule, the concept development, sensor selection, Timer and ADC coding, Circuit construction and system perfection are the essential elements in the construction of prototype. We followed the PARETO analysis, and put most our efforts in these elements.

Table 7.3 PARETO Analyses for Phase Measurement Device

Parameters	Efforts should be made	Cumulative %
Concept Development	30	30%
Sensor selection	20	50%
Timer and ADC coding	10	60%
Circuit construction	10	70%
System Perfection	10	80%
Testing	10	90%
Decay-time Algorithm	5	95%
LCD coding	5	100%

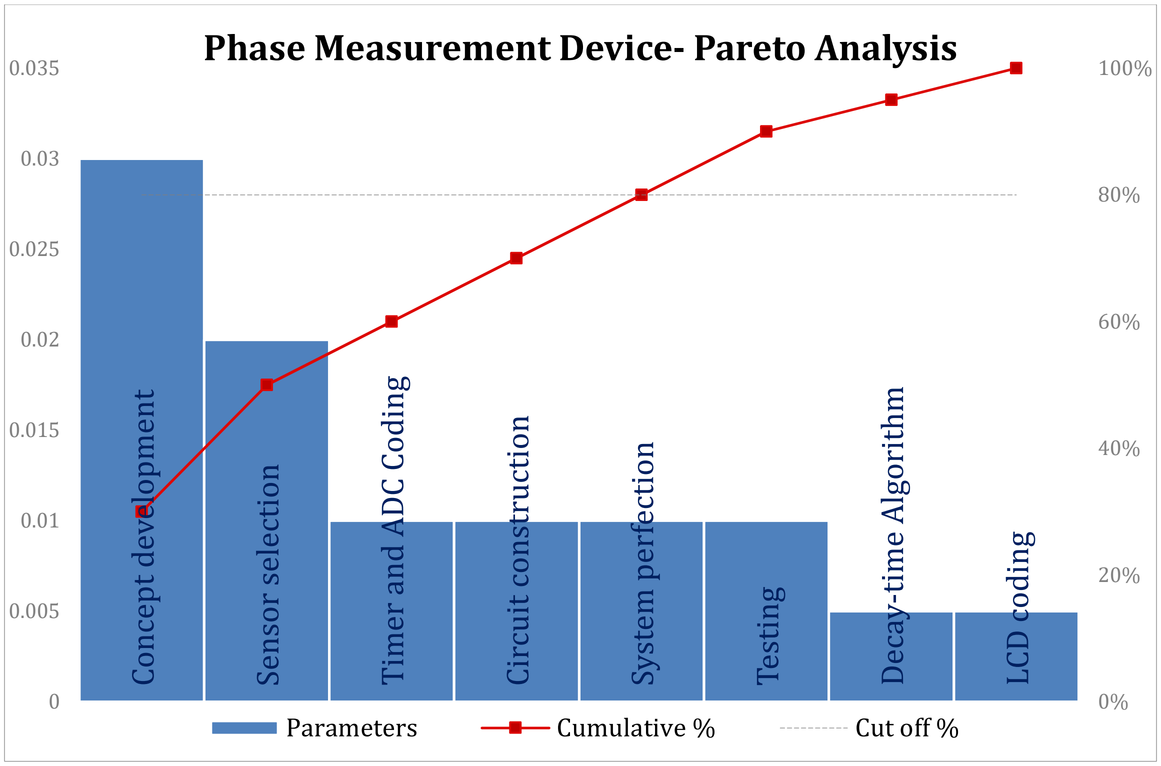


Figure 7.6 PARETO Analyses for Phase Measurement Device

8. Conclusion

We've proven our algorithm by RC circuit simulation, but due to the limitation of Arduino and Avalanche photodiode, we can't get proper signal outcome. A new design with rectifier and low pass filter can be incorporate to the system to accommodate the Avalanche photodiode, and the simulation done by MatLab has proven its viability.

9. Acknowledgements

We would like to thank our mentor Dr. Chun-Yuh Charles Huang for guiding us through the project. And thanks Dr. Nelson Salas, Dr. Jorge Bohorquez, Dr. Ramon Montero, Zhiqiang Liu, Yufu Wang for their great help!.

10. References

1. Mariano L. Bossi, M.E.D., Pedro F. Aramendia, *Luminescence quenching of Ru(II) complexes in polydimethylsiloxane sensors for oxygen.* Journal of Photochemistry and Photobiology A, 1998. **Chamistry 120**(15-21).
2. J.R. Bacon, J.N.D., *Determination of Oxygen Concentrations by Luminescence Quenching of a Polymer-Immobilized Transition-Metal Complex.* Anal. Chem., 1987. **59**(23).
3. Max E. Lippitsch, J.P., Marco J.P. Leiner, Otto S. Wolfbeis, *Fiber-Optic Oxygen Sensor with the Fluorescence Decay Time as the Information Carrier.* Analytica Chimica Acta, 1987. **205**(1-6).

4. Wang, C., C.Y. Huang, and W.C. Lin, *Optical ATP biosensor for extracellular ATP measurement.* Biosens Bioelectron, 2013. **43**: p. 355-61.

Appendices

Code wrote on microprocessor

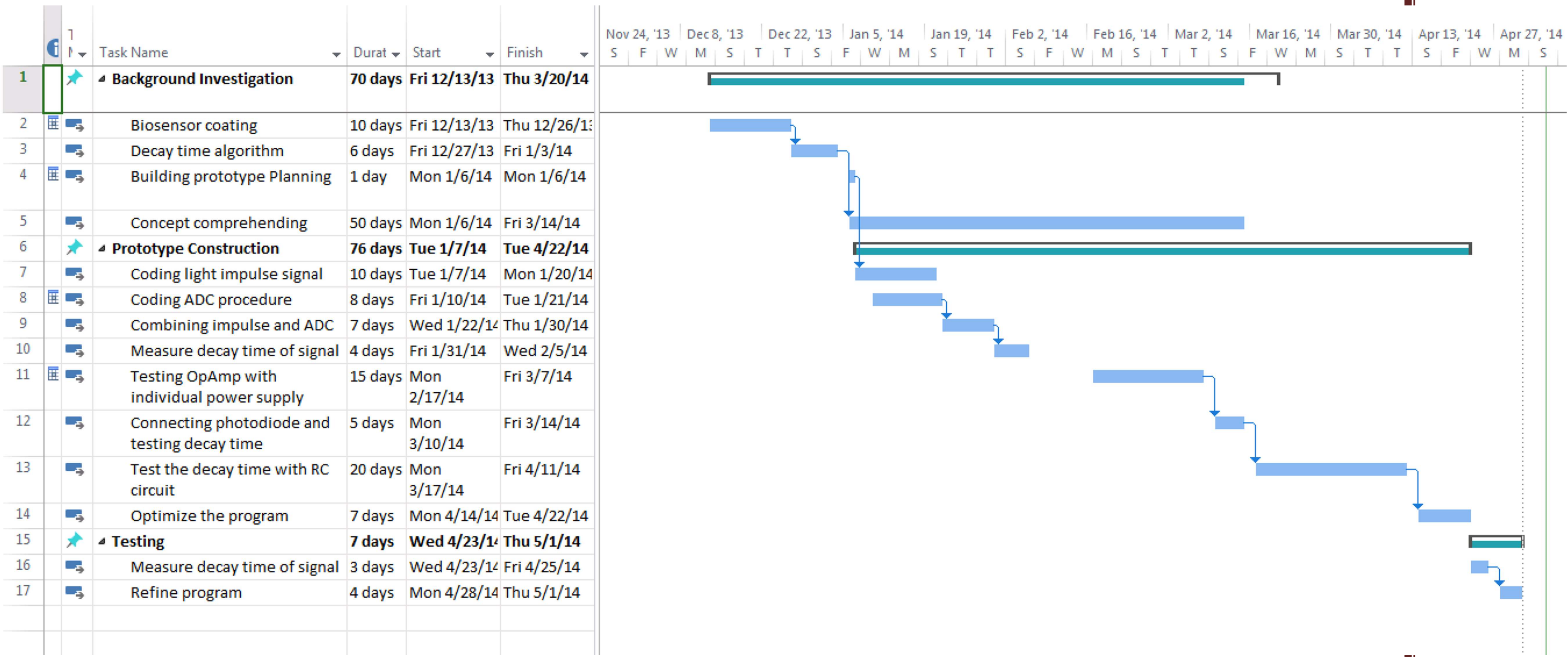
```
// include the library code:
#include <LiquidCrystal.h>
#include <avr/io.h>
#include <math.h>
#define F_CPU 16000000L

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//const int ledPin = 9;
double Vcc = 4.95;
int  lcd_start()
{
// set up the LCD's number of columns and rows:

  lcd.begin(16,2);
  // Print a message to the LCD.
  lcd.print("hello, world!!");
  //pinMode(ledPin, OUTPUT);
}

int  adc_normal_start()//setup ADC7(A0) with normal conversion
{
ADCSRA |= (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);//prescalar=128; ADC clk freq=125KHz;
//1 ADC clock = 8us; 25ADCclk=0.2ms

ADMUX |= 0x47;//AVcc with external capacitor on AREF pin. right adjustment to maximize
//set ADC7 as input channel

ADCSRA |= (1<<ADEN);//ADC enable

ADCSRA |= (1<<ADSC);//ADC start conversion //normal conversion does not manipulate
ADCSRB

}

int  adc_decay_start()//setup ADC7(A0) with normal conversion
{
ADCSRA |= (1<<ADPS2);//prescalar=16; ADC clk freq=1MHz;//1 ADC clock = 1us;

ADMUX |= 0x47;//AVcc with external capacitor on AREF pin. right adjustment to maximize
//set ADC7 as input channel

ADCSRA |= (1<<ADEN);//ADC enable

ADCSRA |= (1<<ADSC);//ADC start conversion //normal conversion does not manipulate
ADCSRB

}

double adc_normal()
{
int sum = 0;//use to average normal_ADC reading adc_normal_start();for (int i=0; i<9;
i++){

    while(ADCSRA  &  (1<<ADSC));//wait till ADC complete
    sum = sum+ADC;
    ADCSRA |= (1<<ADSC);
    }
  sum = sum+ADC;
  ADCSRA = ADCSRA & 0xF8;//clear prescalar
  ADCSRA &= ~(1<<ADEN);
  double normal_ADC_ave = sum/10.0/1023.0*1000*Vcc;
  return normal_ADC_ave;
}

int main (void)
{
lcd_start();
DDRB  |=  (1<<5);//setting Pin 9 as LED output
TCCR1B  |=  (1<<CS11);//prescalar 8, initializing Timer/counter1
for (;;)
{
double tau_sum = 0;
double tau_ave = 0;
for (int freq = 0; freq < 150; freq++)
{
double normal_ave = 0;□double decay_2us = 0;//actually it's 1.5us double tau = 0;□int
LED_flag = 0;□TCNT1 =0;□while (LED_flag ==0)//LED dark□{

if (TCNT1 >= 33332)//counting {

PORTB |= (1<<5);//LED light up TCNT1 = 0;//Reset timer value LED_flag = 1;

} }

while (LED_flag == 1)//LED light up {

    if (TCNT1 >=20000 & TCNT1 < 22000)
    {
    normal_ave = adc_normal();
    }
    else if (TCNT1 >= 32000 & TCNT1 < 32040)
    {
         adc_decay_start();
    }
    else if (TCNT1 >= 33332)
    {
PORTB &= ~(1<<5);//LED shut down adc_decay_start();□while(ADCSRA & (1<<ADSC));//wait
till ADC complete decay_2us = ADC/1023.0*1000*Vcc;□ADCSRA = ADCSRA & 0xF8;//clear
prescalar□LED_flag =0;

TCNT1 = 0;//reset timer value

tau = (0.0-1.5)/(log(decay_2us/normal_ave)); tau_sum = tau_sum + tau;

} }

}
tau_ave = tau_sum/150.0;
lcd.setCursor(0,1);
lcd.print(tau_ave);
lcd.print("us     ");
}
}
//}
```

MatLab simulation

```
%% simulation 33ms
clear all; close all;
input=[zeros(1,15000) ones(1,15000)];
input=repmat(input,[1 10]);
t=0:(30000*10-1);
sys=tf(1,[3 1]);
% sys2=tf([250 0],[250 1]);
output=lsim(sys,input,t);
% oo=lsim(sys2,output,t);
figure(1)
subplot(211);plot(t,input)
axis([0 length(t) -0.5 1.5]);xlabel('micro-sec');
title('pulse generated by Microprocessor')
subplot(212);plot(t,input,t(1:end-1),output(2:end),'r')
axis([0 length(t) -0.5 1.5]);xlabel('micro-sec');
legend('blue light signal','orange light signal')
title('ideal blue/orange light signal')

grid on;



%% simulation 5us w/photodiode
clear all; close all;
input=[zeros(1,5) ones(1,5)];
input=repmat(input,[1 200]);
t=0:(10*200-1);
sys=tf(1,[5 1]);
sys2=tf([250 0],[250 1]);
output=lsim(sys,input,t);
oo=lsim(sys2,output,t);
figure(1)
subplot(311);plot(t(1700:1900),input(1700:1900)...
    ,t(1700:1900),output(1700:1900),'r',...
    t(1700:1900),oo(1700:1900),'g');
legend('Blue light','Orange light', 'APD signal')
xlabel('micro-second')
axis([1700 1900 -1 1.5])
wc=0.001*2*pi;
sys3=tf(1,[1/wc 1]);
ooo=lsim(sys3,abs(oo),t);

subplot(312);plot(t(1700:1900),output(1700:1900)...
,'r',t(1700:1900),abs(oo(1700:1900)),'g',...
t(1700:1900),ooo(1700:1900),'b');
legend('orange light','rectified APD signal','LPF')
axis([1700 1900 -1 1.5])
grid on;

sys4=tf(1,[2 1]);
output=lsim(sys4,input,t);
APD2=lsim(sys2,output,t);
APD2r=abs(APD2);
APD2=lsim(sys3,APD2r,t);
```

```
subplot(313);plot(t(1700:1900),abs(oo(1700:1900))...
,'r',t(1700:1900),APD2r(1700:1900),'g');
legend('5us sim','2us sim')
axis([1700 1900 -1 1.5])
grid on;

figure(100)

subplot(211);plot(t(1700:1900),ooo(1700:1900)...
,'r',t(1700:1900),APD2(1700:1900),'g');
legend('5us sim','2us sim')
axis([1700 1900 -1 1.5])
grid on;

sys3us=tf(1,[3 1]);
output=lsim(sys3us,input,t);
APD3=lsim(sys2,output,t);
APD3r=abs(APD3);
APD3=lsim(sys3,APD3r,t);
sys4us=tf(1,[4 1]);
output=lsim(sys4us,input,t);
APD4=lsim(sys2,output,t);
APD4r=abs(APD4);
APD4=lsim(sys3,APD4r,t);
APD2us=mean(APD2(1700:1900));
APD3us=mean(APD3(1700:1900));
APD4us=mean(APD4(1700:1900));
APD5us=mean(ooo(1700:1900));
subplot(212);plot([2 3 4 5],[APD2us APD3us APD4us APD5us])
title('Magnitude vs. Decay');
xlabel('decay (microsecond)');
grid on;

%% simulation 5us w/photodiode
clear all; close all;
input=[zeros(1,15000) ones(1,15000)];
input=repmat(input,[1 20]);
t=0:(30000*20-1);
sys=tf(1,[5 1]);
sys2=tf([250 0],[250 1]);
output=lsim(sys,input,t);
oo=lsim(sys2,output,t);
figure(1)
subplot(211);plot(t,input,t,output,'r',t,oo,'g');
title('input, ideal output, signal from APD')
axis([0 600000 -2 2])
xlabel('microsecond')
legend('input', 'ideal output', 'signal from APD')
wc=0.001*2*pi;
sys3=tf(1,[1/wc 1]);
ooo=lsim(sys3,abs(oo),t);

subplot(212);plot(t,output,'r',t,abs(oo),'g',t,ooo,'b');
axis([0 600000 -2 2])
grid on;
```

GANTT CHART



	Task Name	Duration	Start	Finish
1	**Background Investigation**	**70 days**	**Fri 12/13/13**	**Thu 3/20/14**
2	Biosensor coating	10 days	Fri 12/13/13	Thu 12/26/1
3	Decay time algorithm	6 days	Fri 12/27/13	Fri 1/3/14
4	Building prototype Planning	1 day	Mon 1/6/14	Mon 1/6/14
5	Concept comprehending	50 days	Mon 1/6/14	Fri 3/14/14
6	**Prototype Construction**	**76 days**	**Tue 1/7/14**	**Tue 4/22/14**
7	Coding light impulse signal	10 days	Tue 1/7/14	Mon 1/20/14
8	Coding ADC procedure	8 days	Fri 1/10/14	Tue 1/21/14
9	Combining impulse and ADC	7 days	Wed 1/22/14	Thu 1/30/14
10	Measure decay time of signal	4 days	Fri 1/31/14	Wed 2/5/14
11	Testing OpAmp with individual power supply	15 days	Mon 2/17/14	Fri 3/7/14
12	Connecting photodiode and testing decay time	5 days	Mon 3/10/14	Fri 3/14/14
13	Test the decay time with RC circuit	20 days	Mon 3/17/14	Fri 4/11/14
14	Optimize the program	7 days	Mon 4/14/14	Tue 4/22/14
15	**Testing**	**7 days**	**Wed 4/23/1**	**Thu 5/1/14**
16	Measure decay time of signal	3 days	Wed 4/23/14	Fri 4/25/14
17	Refine program	4 days	Mon 4/28/14	Thu 5/1/14