

Janelle Shane
ECE 480 – Design team 4
Michigan State University
East Lansing, MI

Application note:

Mini LED reflection spectrometer using an 18F4520

March 30, 2007

Keywords: mini spectrometer spectrophotometer spectroscopy reflection reflectance LED 18F4520 three color identification matching

Executive summary:

A reflectance spectrometer measures the amount of light an object reflects at different wavelengths. This application note provides an introduction to the principles of color and light involved in a reflectance spectrometer, and describes how to build a simple reflectance spectrometer and interface it with a PC. The spectrometer illuminates a sample with a red, green, and blue LED one at a time, and collects total signal (the reflectance) for each color illumination with a photoresistor. An algorithm is outlined by which the reflectance spectrometer can use these three reflectances to match a sample's color to a set of calibration samples. The intended application of this spectrometer is as an automatic reader of urinalysis test strips.

Intended audience:

An undergraduate electrical engineering student with some basic experience in wiring, circuit design (solving simple circuit equations), and very limited C programming. No prior experience with spectrometers or color/light is required. Access to PIC programming/debugging tools (such as MPLAB IDE and the MPLAB ICD 2) is required, as well as a 5V power supply. A complete parts list is found in section 1.

1 Parts List

Amount	Name	Description
Microcontroller/communication		
1	PIC 18F4520 microcontroller	Microcontroller
1	MXO45IIS Clock	Clock for the microcontroller
1	MAX232 Serial Buffer	Used for RS-232 communication
1	Winford PBC6P6C	Programs microcontroller
1	Winford PDC9F	Connects to RS-232
1	RS-232 cable	Connects to PC
1	PC with RS-232 port	Some newer computers may not have RS-232 ports
1	Breadboard/perf board	At least 4x6"
4	10 μ F capacitor	Standard capacitors
1	10k Ω resistor	Standard resistor
2	1k Ω resistor	Standard resistor
Reflectance detection		
1	NSL-6110 photoresistor	Or similar
1	U LM324 chip	Contains four op-amps
4	Resistors	Values will vary
Red, green LED illumination		
1	Mouser 351-3230-RC	Red LED, water clear lens
1	Mouser 351-3240-RC	Green LED, water clear lens
2	390 Ω resistor	LED current-limiting resistors
Blue LED illumination		
1	Mouser 351-3310-RC	Blue LED, water clear lens
1	2N2222 transistor	Switches LED power supply
1	100 Ω resistor	Standard resistor
1	39 Ω resistor	Standard resistor
1	15 Ω resistor	Standard resistor

2 Introduction

2.1 Introduction to the reflectance spectrometer

Color is used as an indicator in many applications, both within and outside the laboratory. A color change in a solvent, for example, may indicate the presence and progress of a chemical reaction, yielding information on the amounts of reactants present in a sample. Urinalysis test strips use the color

change resulting from reaction with the chemicals on a small test square to indicate the presence and concentration of various substances within the urine.

The simplest way to measure the color of a test strip is to use the human visual system: a human visually compares the test strip to a color chart (Figure 1) and reads the corresponding reactant concentration of the closest color match.



Figure 1. Top: Sample urinalysis color chart showing test strip colors and the corresponding glucose concentration. Bottom: manually comparing a test strip to the chart on the bottle. Image from (top) <http://www.cotc.edu/csimpson/urinary.htm>, (bottom) <http://www.irvingcrowley.com/cls/urin.htm>.

Let's examine how a human does this color identification, as it will form the basis for the design of a reflectance spectrometer that does it automatically. When a human perceives the "color" of an object, they are simply observing the composition of light that the object reflects. For example, if an object appears red in color, this is because it reflects red light and absorbs all other colors. A white object reflects all colors equally, and a black object absorbs all wavelengths equally. The human visual system determines an object's color based on the total reflected light from that object, detected by each of the eye's three types of color receptors: red, green and blue cone cells (Figure 2).

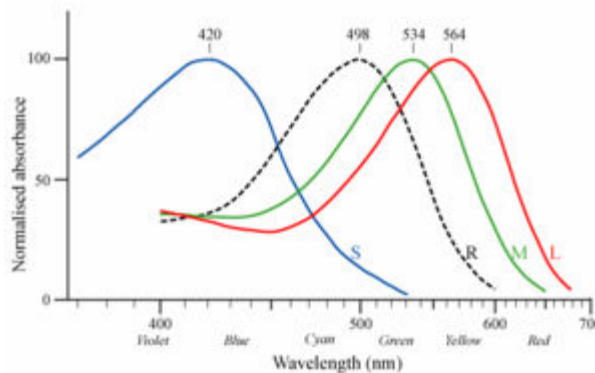


Figure 2. Human red, blue, and green photoreceptors. The visual system determines color by comparing the total reflected light detected by each receptor type. Image source: www.answers.com

The reflectance spectrometer (a type of spectrophotometer) can be used to automatically detect color changes in urinalysis test strips, saving time and reducing the chance for human error. A reflectance spectrometer measures the composition of light reflected from a urinalysis test strip (or any other object). The spectrometers can be designed to measure reflected light at a range of wavelengths, not necessarily in the visible range (approx. 350-700nm) (Figure 3).

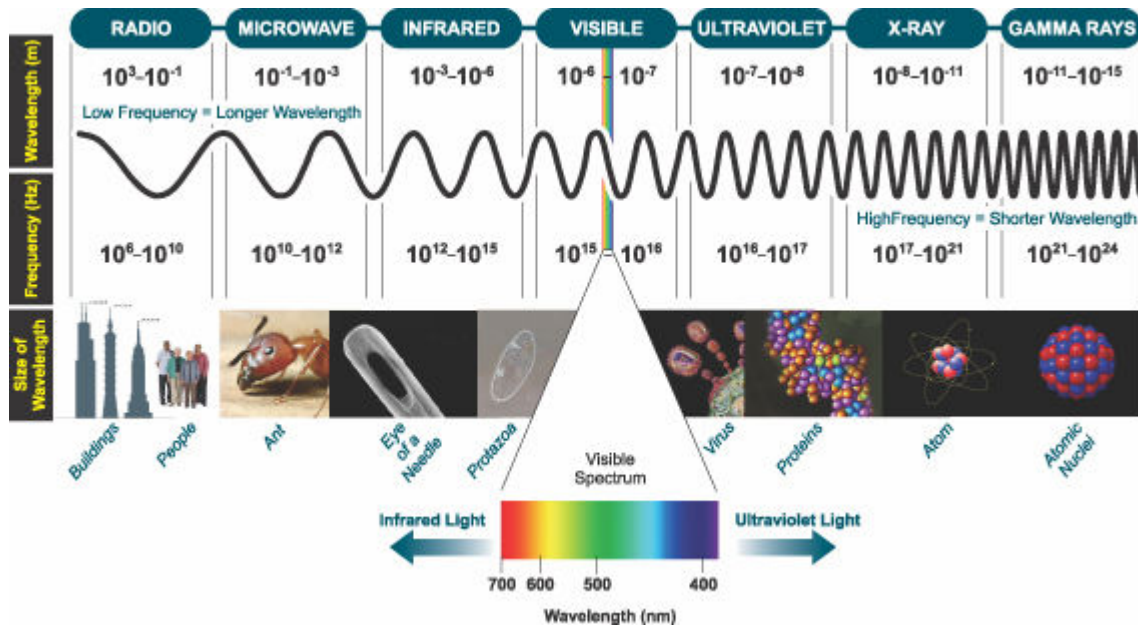


Figure 3. Electromagnetic radiation. The visible wavelength range extends from approximately 350-700nm. Image source: <http://www.andor.com/library/light/>

Unlike the human eye, a reflectance spectrometer provides its own illumination. To measure how much red light is reflected from a sample, for example, a spectrometer has two primary alternatives:

1. It may illuminate the sample with red light only and measure the total amount of light that is reflected.
2. It may illuminate the sample with white light and filter the reflected light to obtain red light only.

Typically a spectrometer detects reflected light at not just three wavelengths, like the human eye, but at an array of evenly-spaced wavelengths, creating a full spectrum of the object. This full spectrum is useful for research purposes but unnecessary for basic color identification; like the human eye, a spectrometer can use just three wavelengths to distinguish colors. This application note will describe a reflection spectrometer that determines color using three different wavelengths of illumination.

Closely related to the reflectance spectrometer is the transmission spectrometer, which measures the light transmitted through a partially transparent object. Many commercial spectrometers can measure both reflectance and transmission with a simple change of geometry.

The quantity a reflectance spectrometer measures at each illumination wavelength is reflectance, defined as the ratio:

$$\text{Reflectance}(\lambda) = \frac{I_{obj}(\lambda)}{I_{white}(\lambda)},$$

where I_{obj} is the intensity of light reflected by the object, and I_{white} is the intensity of light reflected by a perfectly white object. For a truly white object, I_{white} will be constant, but in practice it is best to measure I_{white} at each wavelength to account for wavelength-dependent variations in detector sensitivity, interfering light, or illumination intensity.

3 Objective

The objective of this application note is to describe the design, construction, and programming of a simple reflectance spectrometer. This mini-spectrometer does not obtain reflectance values over an entire spectrum, but just at three selected wavelengths. The mini-spectrometer does this by illuminating the sample with selected LEDs and recording reflectance using a photoresistor detector. This mini-spectrometer is designed to read the colors of urinalysis test strips, but could perform other color-matching functions as well. The application note will provide instructions for developing a spectrometer circuit and program based on an 18F4520 PIC, and for allowing this program to communicate via RS-232. Additionally, an algorithm for color-matching will be briefly outlined, revealing how this design can be used to read chemical indicator test strips.

4 Body

4.1 Basic reflection spectrometer design

As described above, a reflection spectrometer consists of an illumination source and a detector. To get color-specific reflectance, the illumination or the detection has to be monochromatic. The reflection spectrometer described in this application note uses monochromatic illumination: the sample is illuminated in sequence by each of three differently-colored LEDs: red, green, and blue. For each illumination color, the total reflectance from the sample is detected using a photoresistor. An 18F4520 PIC controls the LEDs and measures the voltage drop across the photoresistor, sending this data over RS-232 to the PC. The PC stores the reflectance values and processes them for color identification, if necessary. (See block diagram in Figure 4). The following sections will describe each portion of the spectrometer in greater detail. Appendix A contains a schematic of the entire spectrometer.

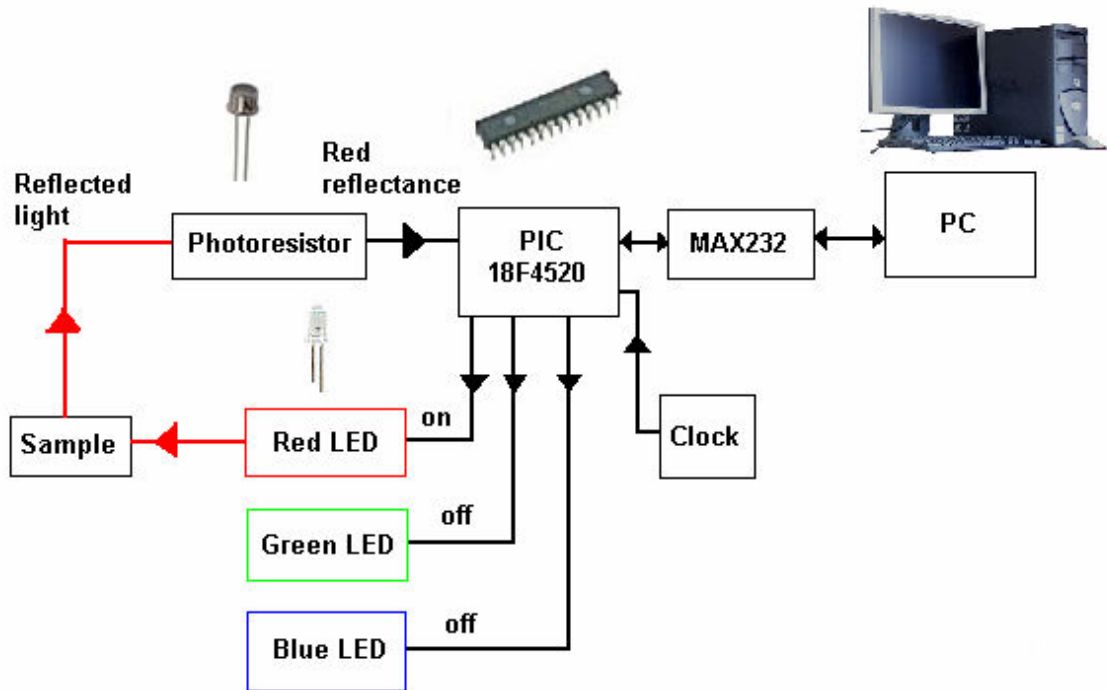


Figure 4. Block diagram of mini reflectance spectrometer

4.2 LED illumination

4.2.1 LED geometry

The LEDs must illuminate the sample without sending light directly into the photoresistor. To do this, we use the geometry shown in Figure 5. The LED must have a narrow illumination angle (about 30°) or it will leak light into the photoresistor. For this reason, it is best to use LEDs with “water clear” lenses, rather than diffuse coated lenses, which scatter the LED’s light. Some amount of leakage is acceptable, since this leakage will be constant for any given sample. If the leakage is too high, it will obscure the true signal.

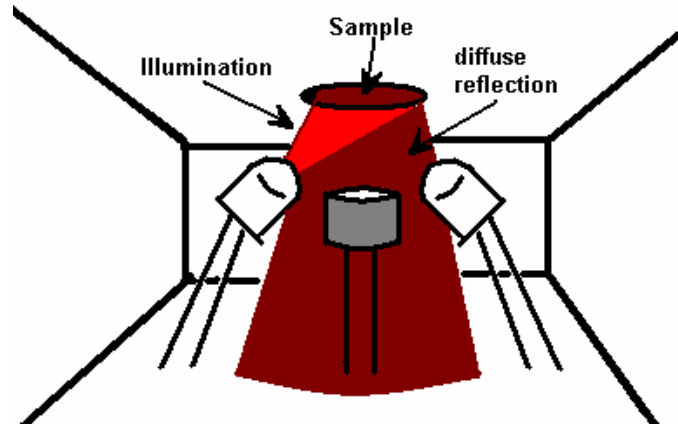


Figure 5. Spectrometer geometry. The LEDs are arranged at angles around the photodiode, directing light in a narrow beam toward the sample. The entire setup is enclosed in an opaque box to prevent interfering light from entering. A hole at the top allows access to the sample.

4.2.2 Powering the LEDs

As will be described later, the LEDs are powered directly by one of the 18F4520 PIC's output pins. If blue LEDs are used, they will likely require more power than the PIC's output pins can supply. In this case, the output signal can be used to control a transistor switch that turns on and off a 5V supply. The transistor switch used in the mini-spectrometer is shown in Figure 6.

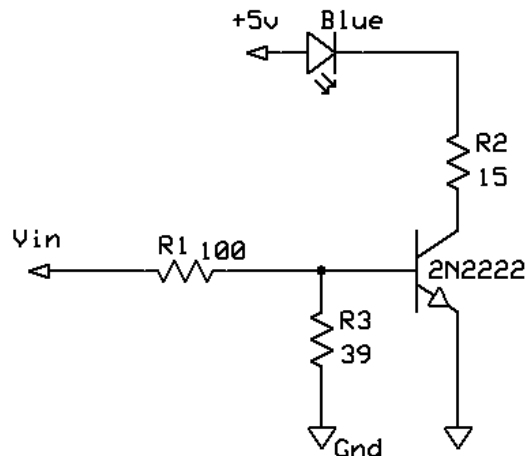


Figure 6. Transistor switch used to power an LED. Values shown are for a 5V – 0V input and a Mouser 351-3310-RC “ultra-bright” blue LED.

Biassing the transistor: The switch is based on a 2N2222 transistor. When the voltage across the base is above 1.2V, the switch will turn on, completing the circuit between R₂ and ground so that current may flow through the LED from the 5V power supply. The input voltage V_{in} is 5V when high and 0V when low, so the 5V high value must be converted to about 1.5V at the transistor base. This follows the equation:

$$V_B = V_{in} \frac{R_3}{R_1 + R_3} = 1.5V$$

If R_1 is chosen to be 100Ω , then, using $V_{in} = 5V$, $R_3 = 42.8\Omega$. The closest standard resistor value is 39Ω .

Adjusting the voltage drop across the LED: The resistor R_2 serves as a current limiter, and also adjusts the voltage drop across the LED. Its value will depend on the current I_d and voltage drop V_d required by the blue LED, following the equation

$$R_2 = \frac{5V - V_d}{I_d}$$

For the Mouser 351-3310-RC blue LED, $I_d = 20mA$ and $V_d = 4.7V$ (at the upper end of the LED's voltage rating, but due to resistances in wires and in the LED, the actual voltage is a bit lower). This gives $R_2 = 15\Omega$.

4.3 The detector

The reflected light from the sample is detected with a photoresistor (although a photodiode could also be used). The more light the photoresistor sees, the lower its resistance, and therefore the lower the voltage drop across it. The voltage at the output to the photoresistor is the quantity that needs to be sent to the PIC for conversion to digital.

For the PIC's analog-to-digital converter (ADC) to detect the most variation in the photoresistor voltage, these voltage values should cover the entire ADC input range, from 0 to 5V. However, the voltage drop across the photodiode usually will instead vary by a small amount around a nonzero voltage value. For example, it might vary from 1.5 to 2V. A voltage divider circuit can be used to reduce the photodiode voltage to a small range near zero (say, 0 to 0.5V). Then, a noninverting amplifier can be used to amplify this voltage (for example, a gain of 10 would stretch the voltage range to 0 - 5V). (See Figure 7) The equation for the voltage divider is:

$$V_1 = 5 \frac{R_s}{R_p + R_s},$$

and the equation for the noninverting amplifier is

$$V_2 = \left(1 + \frac{R_2}{R_1}\right) V_1.$$

The exact resistor values needed will vary from setup to setup, depending on the variation in R_p , which depends on factors including the proximity of the sample and LEDs, and the amount of light that leaks into the photoresistor.

Another requirement of the 18F4520 ADC is that the input resistance be no more than $2.5k\Omega$. A buffer circuit will reduce the input resistance to effectively zero. The full photodiode circuit is shown in Figure 7.

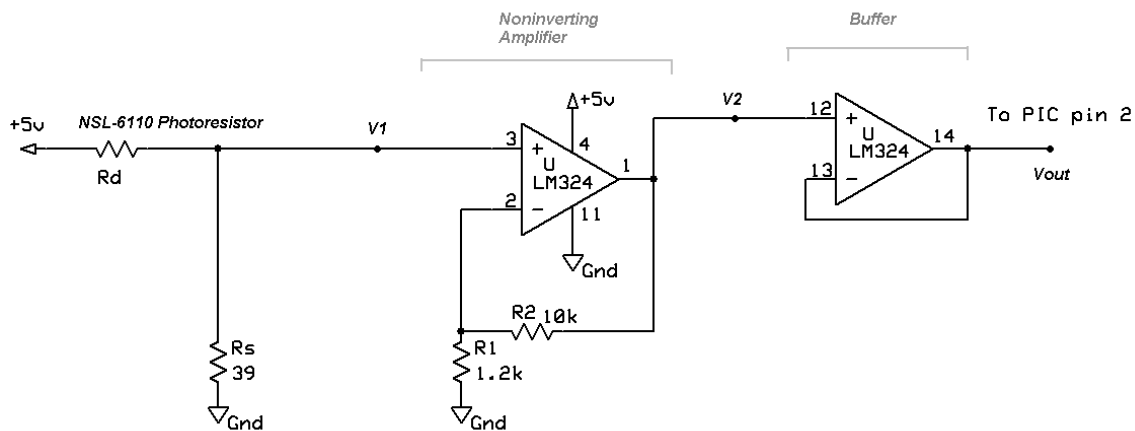


Figure 7. Amplifier circuit for photoresistor signal. The LM324 chip contains four op-amps; two are used here, with pin numbers shown.

The NSL-6110 Photoresistor will for this application. The main requirement for whatever photoresistor is used is that it must be sensitive to wavelengths emitted by the red, green, and blue LEDs. Some photoresistors detect mainly in the red to infrared, so it is important to choose one that will be sensitive to blue. The NSL-6110 is more sensitive to red than to blue wavelengths, but can still respond to blue light.

4.4 Central unit: the 18F4520

The 18F4520 PIC controls the operation of the spectrometer. Please see the following documents for an introduction to using the 18F4520 in a somewhat similar configuration (the circuit diagrams and code in this application note are adapted from these documents):

- <http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/Lab3.pdf>
- <http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/Lab4.pdf> with supporting file <http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/LAB4VB.zip>

A datasheet for the 18F4520 can be found at:

<http://ww1.microchip.com/downloads/en/DeviceDoc/39631a.pdf>

A brief description of the 18F4520 features that are relevant to the spectrometer construction:

Input/output ports: The 18F4520 has a port, “PortD” that uses four pins (pins 18, 19, 20, and 21). The spectrometer uses these four pins as digital outputs, each of which switches one LED on and off. Note that only four LEDs can be controlled in this way; a decoder switch would be needed for more LEDs

A/D Conversion: The 18F4520 has a 13-bit analog-to-digital converter (ADC) located at pin 2. The ADC measures analog input voltages over the range 0 to 5V and maps them onto 256 integer values. The resolution is therefore 5/256

= 0.02V. The spectrometer makes use of this A/D conversion to read the analog photoresistor detector voltage.

Clock: The 18F4520 requires a clock, which can have speeds up to 40MHz. This spectrometer uses a MX045HS clock.

Adaptation for RS232 communication: The spectrometer communicates with the computer via RS-232 ports. Although slower and less ubiquitous than USB, RS-232 makes it much easier for a user to send and receive data directly from a peripheral device and is therefore ideal in this application. The 18F4520 outputs data with standard TTL logic (0V = low, 5V = high), but RS-232 uses the rather strange voltages: -15 to -3: high, 3 to 15: low. A MAX232 chip will do the necessary voltage conversion and logic inversion.

The introductory 18F4520 documents listed above explain step-by-step how to use these features, as well as sample microcontroller and GUI programs. It is recommended to try the quick experiments in these documents as a good introduction. Appendix A contains a full schematic for the spectrometer circuit, showing the proper way to connect power, ground, clock, MAX232, etc.

4.5 Controlling software

4.5.1 Algorithm & code for reading reflectance

Using MPLAB IDE, the 18F4520 can be programmed with C code that will allow it to operate the LEDs and read the detector voltage. The basic algorithm for reading reflectance from each sample is as follows:

1. Wait for the PC to send a character over the RS-232 line
2. If the character is 'r':
 - a. Turn on red LED
 - b. Read photoresistor voltage & send to PC
3. If the character is 'b':
 - a. Turn on blue LED
 - b. Read photoresistor voltage & send to PC
4. If the character is 'g':
 - a. Turn on green LED
 - b. Read photoresistor voltage & send to PC
5. Return to step 1

The full code for the 18F4520 can be found in Appendix B. This code operates based on commands from the PC, which sends single-character commands over the RS-232 line (see section 4.5.2 for an example). The spectrometer reads the three reflectance values and sends them to the PC for display or further analysis.

The following code excerpt provides an example of how to read reflectance at a single wavelength:

```

unsigned char c;
c =getcUSART(); //get a single character off the USART line

if (c == 'r') //Turn red LED on; read reflectance
{
    //Make sure blue, green LEDs are off
    PORTDbits.RD1 = 0; //blue LED off
    PORTDbits.RD3 = 0; //green LED off

    //Turn red LED on
    PORTDbits.RD0 = 1;
    for(dloop=0;dloop<waittime;dloop++); //wait a bit
    //Read photoresistor voltage
    while(BusyUSART());
    ConvertADC(); //perform ADC conversion
    while(BusyADC()); //wait for result
    red = ReadADC(); //get ADC result
    PORTDbits.RD0 = 0; //Turn red LED back off

    data = red >> 2;
    putcUSART (data); //put a single character on the USART line
    PIR1bits.RCIF = 0; //reset the ISR flag.
}

```

The red LED is connected to pin 18, which is pin 1 of PORTD. Writing a 1 to this port will turn the LED on, and writing a 0 to the port will turn the LED off. The function ConvertADC() tells the PIC to read the analog voltage at pin 2 and convert this voltage to a digital value. The loop while(BusyADC()) makes the program wait until the ADC has finished conversion. ReadADC() gets the conversion result, which will be a value from 0 to 255, with 0 corresponding to 0V, and 255 corresponding to 5V. At this point, this value can either be sent right away to the PC, or stored by the PIC for more calculations. The code can easily be modified to perform all three reflectance measurements at a single “start” command from the PC, perform color matching, and return a single value representing the detected color.

4.5.2 A Graphical User Interface

As mentioned earlier, the PIC communicates with a PC (or other processor) by sending and receiving 8-bit characters over an RS-232 line. A graphical user interface (GUI) will allow a user to send commands to the spectrometer and to read the data that is returned.

One program for GUI development is Microsoft Visual Basic.net. The following documents provide a tutorial on setting up a GUI in Visual Basic:

- <http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/Lab3.pdf>
- <http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/Lab4.pdf> with supporting file <http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/LAB4VB.zip>

An example spectrometer GUI is pictured below:

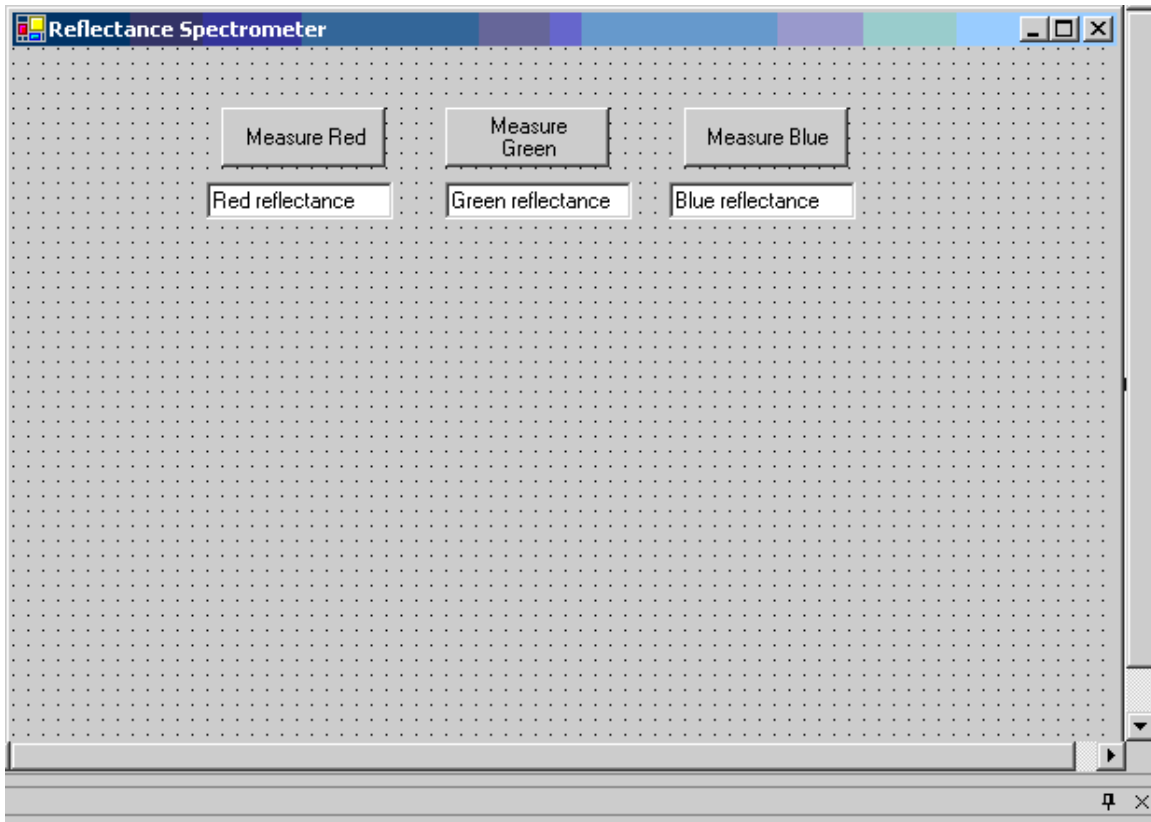


Figure 8. Example Visual Basic GUI

“Measure Red” is Button1, “Red reflectance” is Textbox1; “Measure Green” is Button2, “Green reflectance” is Textbox2; “Measure Blue” is Button3, “Blue reflectance” is Textbox3. The Visual Basic code in Appendix C takes red, green, and blue reflectance readings and displays them.

4.6 Color matching software

This section outlines the principles of an algorithm that could be used for color matching: the K Nearest Neighbor (KNN) algorithm. The example application of the color matching algorithm is in matching urinalysis test strip colors, as described in the introduction. The urinalysis test used is that for glucose (see Figure 1).

The KNN algorithm is summarized as follows:

1. **Training phase:** *Set up a series of known instances for each of the classification categories.* For example, in glucose concentration identification, a series of glucose solutions of known concentration are made. Test strips are dipped in each solution, and their red, green, and blue reflectance values are measured with the spectrometer. Each test strip, then, gets mapped to a point in (red, green, blue) coordinate space. Test strips exposed to the same glucose concentration should cluster in one region of (red, green, blue) coordinate space.

2. **Classification phase:** Compute the Euclidean distance from an unknown sample to the known points. The most prevalent classification category among the unknown sample's K nearest neighbors is assigned to the unknown sample. In the glucose concentration example, measure the (red, green, blue) reflectances of an unknown test strip, mapping it to a point $(red_t, green_t, blue_t)$. Compute the Euclidean distance between $(red_t, green_t, blue_t)$ and the known $(red_i, green_i, blue_i)$ test strips:

$$DISTANCE_i = \sqrt{(red_t - red_i)^2 + (green_t - green_i)^2 + (blue_t - blue_i)^2}$$

Make a list of the K known points with the smallest $DISTANCE_i$ (the test point's K nearest neighbors). The most prevalent classification among these K points is the correct classification for the test strip.

For more information on the KNN matching algorithm, please see, for example, <http://people.revoledu.com/kardi/tutorial/KNN/index.html>.

5 Conclusions/additional considerations

5.1 Summary

This application note described the construction, programming, and use of a three-color reflection spectrometer. It should contain all the information necessary to allow an ECE undergraduate student with sufficient technical background to reproduce this work, and to adapt it to other applications.

5.2 Additional considerations

5.2.1 Compensating temperature dependence

A refinement that could be added to this spectrometer is compensation for temperature-dependent variations in LED intensity, photoresistor response, or test strip chemistry. For a description of such a refinement, please see US patent number 5972715, available for example at

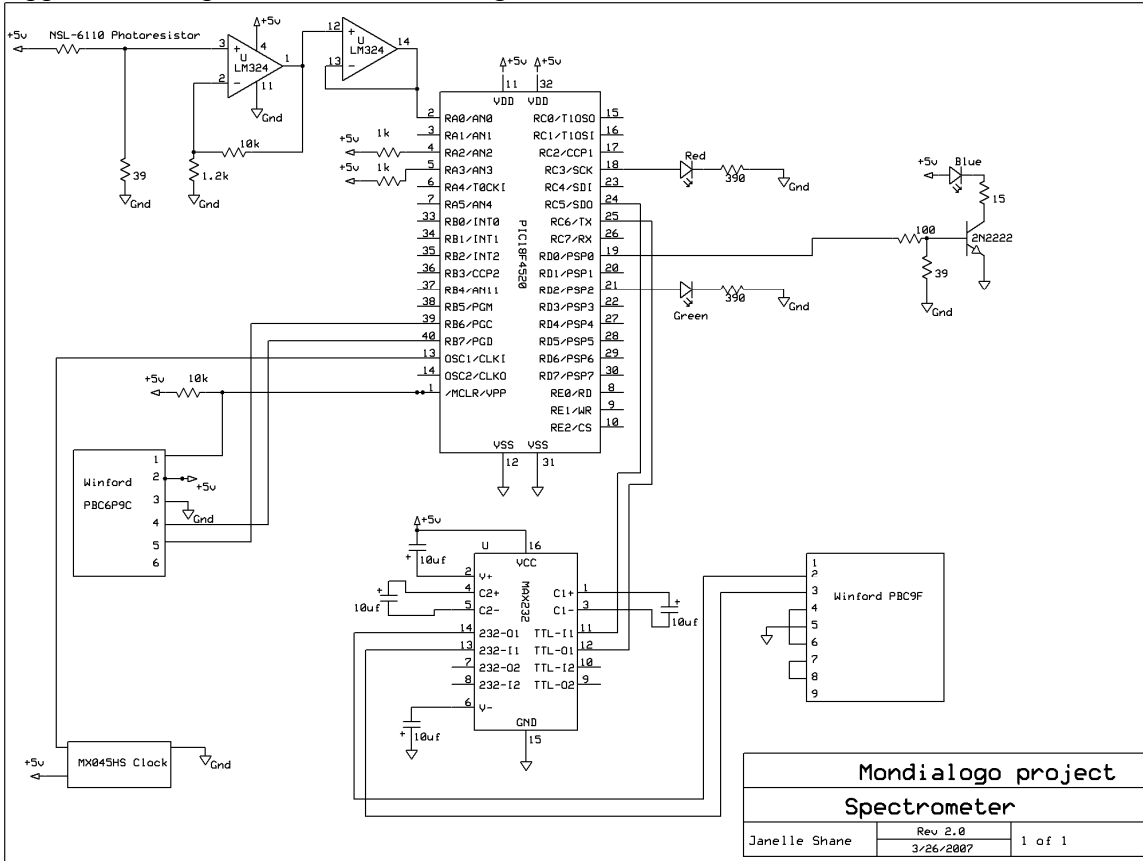
<http://www.patentstorm.us/patents/5972715-description.html>.

6 References

- For an excellent introduction to principles of color and light (as well as other aspects of optics) please see Hecht, Eugene. Optics. Addison Wesley Publishing Company.
- <http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/Lab3.pdf>
- <http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/Lab4.pdf> with supporting file
<http://www.egr.msu.edu/classes/ece480/goodman/ForMiniprojects/LAB4VB.zip>
- <http://ww1.microchip.com/downloads/en/DeviceDoc/39631a.pdf>

- <http://people.revoledu.com/kardi/tutorial/KNN/index.html>.
- Similar patents:
 - US patent number 5972715
 - US patent number 6043893

Appendix A. Spectrometer circuit diagram.



Appendix B. Microcontroller code

```
#include <p18cxxx.h>
#include <usart.h>
#include <ADC.h>
#pragma config LVP=OFF
#pragma config WDT=OFF
void rx_handler (void); //Declare the ISR function
long int count;
int adc_result, blue, red, green;
int ratio;
unsigned char data;
long int dloop = 0;
long int waittime = 200000; //time to wait before reading reflectance
void main()
{
    OpenUSART (USART_TX_INT_OFF & USART_RX_INT_ON &
    USART_ASYNCH_MODE & USART_EIGHT_BIT &
    USART_CONT_RX & USART_BRGH_LOW, 63);
    RCONbits.IPEN = 1; /* Enable interrupt priority */
    IPR1bits.RCIP = 1; /* Make receive interrupt high priority */
    INTCONbits.GIEH = 1; /* Enable all high priority interrupts */
    OpenADC(ADC_FOSC_32 & ADC_RIGHT_JUST & ADC_12_TAD,
    ADC_CH0 & ADC_INT_OFF, 0); //open adc port for reading
    ADCON1 = 0x00; //set VREF+ to VDD and VREF- to GND (VSS)
    TRISD = 0x04;
    PORTDbits.RD0 = 0; //red LED off
    PORTDbits.RD1 = 0; //blue LED off
    PORTDbits.RD3 = 0; //green LED off
    while(1) //keeps the program constantly running
    {
    }
}

//Lets the compiler know the location of the ISR
#pragma code rx_interrupt = 0x8
void rx_int (void)
{
    _asm goto rx_handler _endasm
}
#pragma code

//Lets the compiler know that this is the ISR
#pragma interrupt rx_handler
void rx_handler (void)
{
```

```

unsigned char c;
c = getchUSART(); //get a single character off the USART line

if (c == 'r') //Turn red LED on; read reflectance
{
    //Make sure blue, green LEDs are off
    PORTDbits.RD1 = 0; //blue LED off
    PORTDbits.RD3 = 0; //green LED off

    //Turn red LED on
    PORTDbits.RD0 = 1;
    for(dloop=0;dloop<waittime;dloop++); //wait a bit
    //Read photoresistor voltage
    while(BusyUSART());
    ConvertADC(); //perform ADC conversion
    while(BusyADC()); //wait for result
    red = ReadADC(); //get ADC result
    PORTDbits.RD0 = 0; //Turn red LED back off

    data = red >> 2;
    putcUSART (data); //put a single character on the USART line
    PIR1bits.RCIF = 0; //reset the ISR flag.
}

if (c == 'g') //Turn green LED on; read reflectance
{
    //Make sure red, blue LEDs are off
    PORTDbits.RD0 = 0; //red LED off
    PORTDbits.RD1 = 0; //blue LED off

    //Turn green LED on
    PORTDbits.RD3 = 1;
    for(dloop=0;dloop<waittime;dloop++); //wait a bit
    //Read photoresistor voltage
    while(BusyUSART());
    ConvertADC(); //perform ADC conversion
    while(BusyADC()); //wait for result
    green = ReadADC(); //get ADC result
    PORTDbits.RD0 = 0; //Turn green LED back off

    data = green >> 2;
    putcUSART (data); //put a single character on the USART line
    PIR1bits.RCIF = 0; //reset the ISR flag.
}

if (c == 'b') //Turn blue LED on; read reflectance

```

```
{
    //Make sure red, green LEDs are off
    PORTDbits.RD0 = 0; //red LED off
    PORTDbits.RD3 = 0; //green LED off

    //Turn blue LED on
    PORTDbits.RD1 = 1;
    for(dloop=0;dloop<waittime;dloop++); //wait a bit
    //Read photoresistor voltage
    while(BusyUSART());
    ConvertADC(); //perform ADC conversion
    while(BusyADC()); //wait for result
    blue = ReadADC(); //get ADC result
    PORTDbits.RD1 = 0; //Turn blue LED back off

    data = blue >> 2;
    putcUSART (data); //put a single character on the USART line
    PIR1bits.RCIF = 0; //reset the ISR flag.
}
}
```

Appendix C – GUI Code

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Dim setRs232 As New Rs232
    Dim serial_in As String
    Dim x As Integer
    Dim s As Integer
    Dim red As Integer
    Dim blue As Integer
    Dim ratio As Integer
    Dim samples As Integer
    Declare Sub Sleep Lib "kernel32" Alias "Sleep" (ByVal
dwMilliseconds As Long)

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overrides Sub Dispose(ByVal disposing As
Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form
Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Button2 As System.Windows.Forms.Button
    Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
    Friend WithEvents Button1 As System.Windows.Forms.Button
    Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
    Friend WithEvents Button3 As System.Windows.Forms.Button
    Friend WithEvents TextBox3 As System.Windows.Forms.TextBox
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.Button2 = New System.Windows.Forms.Button
        Me.TextBox2 = New System.Windows.Forms.TextBox
        Me.Button1 = New System.Windows.Forms.Button
```

```

Me.TextBox1 = New System.Windows.Forms.TextBox
Me.Button3 = New System.Windows.Forms.Button
Me.TextBox3 = New System.Windows.Forms.TextBox
Me.SuspendLayout()
'
'Button2
'
Me.Button2.Location = New System.Drawing.Point(232, 32)
Me.Button2.Name = "Button2"
Me.Button2.Size = New System.Drawing.Size(88, 32)
Me.Button2.TabIndex = 0
Me.Button2.Text = "Measure Green"
'
'TextBox2
'
Me.TextBox2.Location = New System.Drawing.Point(232, 72)
Me.TextBox2.Name = "TextBox2"
Me.TextBox2.TabIndex = 2
Me.TextBox2.Text = "Green reflectance"
'
'Button1
'
Me.Button1.Location = New System.Drawing.Point(112, 32)
Me.Button1.Name = "Button1"
Me.Button1.Size = New System.Drawing.Size(88, 32)
Me.Button1.TabIndex = 4
Me.Button1.Text = "Measure Red"
'
'TextBox1
'
Me.TextBox1.Location = New System.Drawing.Point(104, 72)
Me.TextBox1.Name = "TextBox1"
Me.TextBox1.TabIndex = 2
Me.TextBox1.Text = "Red reflectance"
'
'Button3
'
Me.Button3.Location = New System.Drawing.Point(376, 32)
Me.Button3.Name = "Button3"
Me.Button3.Size = New System.Drawing.Size(88, 32)
Me.Button3.TabIndex = 4
Me.Button3.Text = "Measure Blue"
'
'TextBox3
'
Me.TextBox3.Location = New System.Drawing.Point(368, 72)
Me.TextBox3.Name = "TextBox3"
Me.TextBox3.TabIndex = 2
Me.TextBox3.Text = "Blue reflectance"
'
'Form1
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.ClientSize = New System.Drawing.Size(592, 381)
Me.Controls.Add(Me.Button2)
Me.Controls.Add(Me.TextBox2)
Me.Controls.Add(Me.Button1)

```

```

        Me.Controls.Add(Me.TextBox1)
        Me.Controls.Add(Me.Button3)
        Me.Controls.Add(Me.TextBox3)
        Me.Name = "Form1"
        Me.Text = "Reflectance Spectrometer"
        Me.ResumeLayout(False)

    End Sub

#End Region

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        With setRs232
            .Port = 1
            .BaudRate = 9600
            .DataBit = 8
            .StopBit = Rs232.DataStopBit.StopBit_1
            .Parity = Rs232.DataParity.Parity_None
            .Timeout = 10000
        End With
        setRs232.Open()

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        setRs232.Write("r") 'Measure red reflectance'
        setRs232.Read(1)
        serial_in = setRs232.InputStreamString
        x = AscW(CChar(serial_in))
        TextBox1.Text = (x / 256) * 5
    End Sub
    Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox1.TextChanged

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        setRs232.Write("g") 'Read green reflectance'
        setRs232.Read(1)
        serial_in = setRs232.InputStreamString
        x = AscW(CChar(serial_in))
        TextBox2.Text = (x / 256) * 5
    End Sub
    Private Sub TextBox2_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox2.TextChanged

    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        setRs232.Write("b") 'Measure blue reflectance'
        setRs232.Read(1)
        serial_in = setRs232.InputStreamString
        x = AscW(CChar(serial_in))

```

```
        TextBox3.Text = (x / 256) * 5
    End Sub
    Private Sub TextBox3_TextChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles TextBox3.TextChanged

        End Sub
End Class
```