

An Inexpensive Solar Scintillation Seeing Monitor Circuit with Arduino Interface

E. J. Seykora

Department of Physics

East Carolina University

Greenville, NC

Introduction

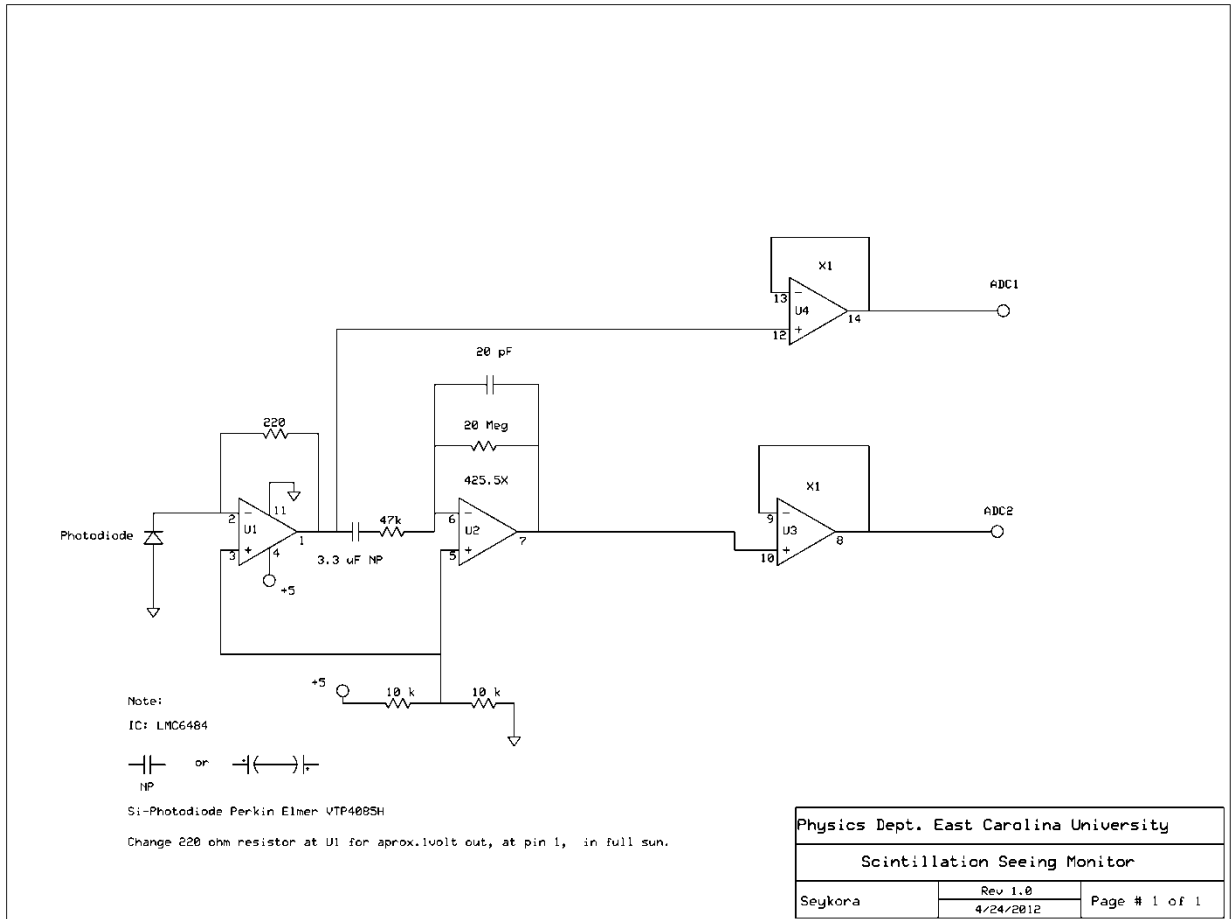
The measurement of solar scintillations has been used for a number of years to assess the amount of atmospheric turbulence at ground or near ground level. Such measurements have been useful in determining the quality of the atmospheric “seeing” during high resolution solar observations. A simple circuit is presented which may be used to measure the RMS scintillation intensity and its interface to an Arduino microprocessor. An Arduino program is also presented which allows seeing data to be transferred to any computer using a USB emulated serial port.

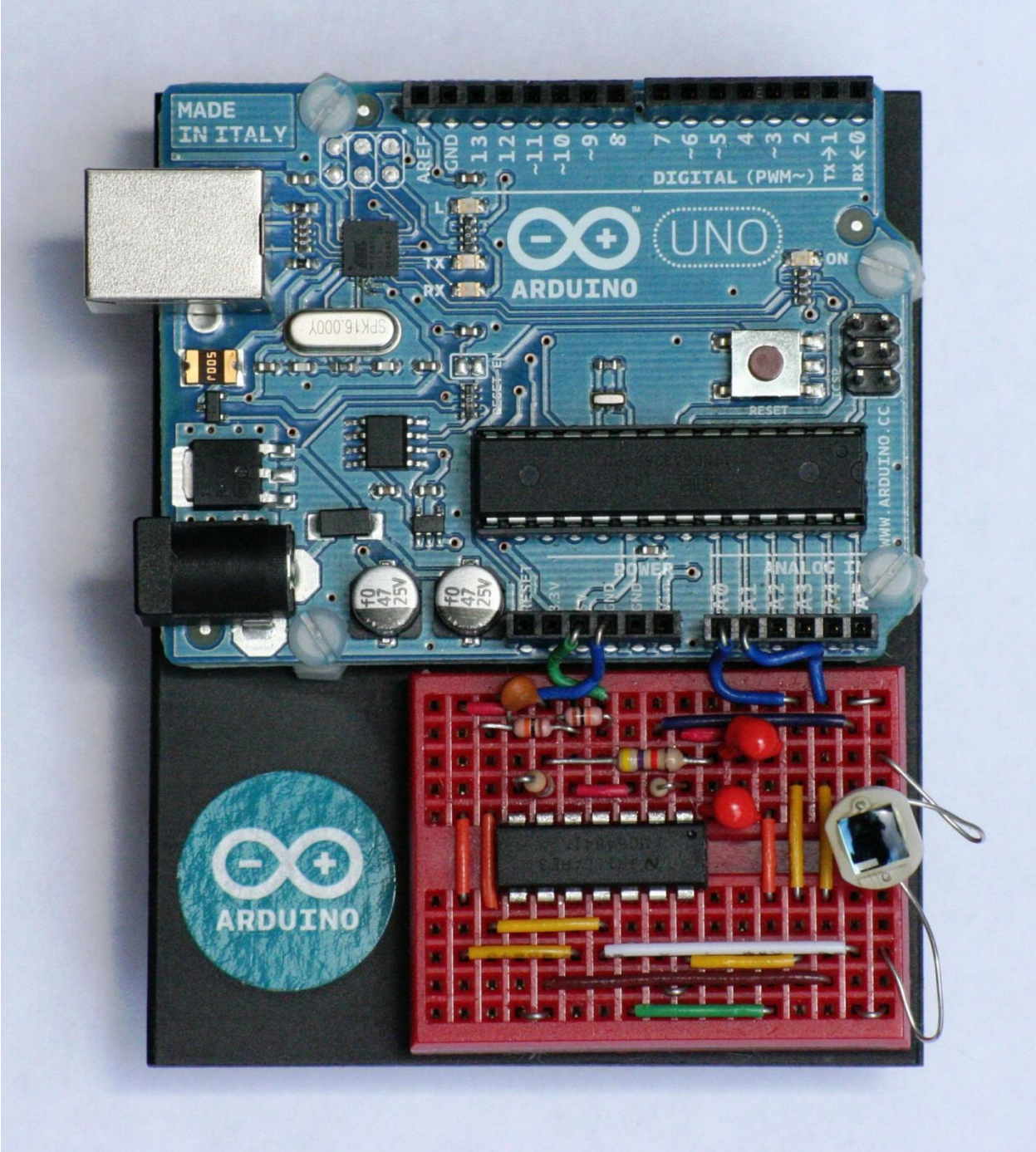
Circuit

A schematic diagram of the circuit is shown below which consists of a small Si Photodiode and amplifier circuit. The amplifier consists of a LMC6484 CMOS quad rail-to-rail input and output operational amplifier. Here the first stage at u1 is a current to voltage converter which converts the current output of the photodiode to a voltage which is proportional to the light intensity. The feedback resistor of this stage, between pin 2 and 1, is normally adjusted such that the voltage output registers a one volt increase when the photodiode is in full sun. This output is used to measure transparency variations as well as angle-of-arrival variations during the day. The following amplifier stage consists of an ac-coupled gain stage with a gain of 425.5X at u2. The final stages consist of a unity gain non-inverting drivers. Here the output of the drivers is used to calculate the ac-coupled RMS voltage or intensity variation. The outputs pins, pin14 or ADC1 and pin8, ADC2 are connected to the A0 and A1 analog input pins of the Arduino Uno Board. Power for the circuit is connected to the +5 volt supply pin of the Arduino Uno Board as well as the ground. This circuit along with the Arduino Board may be powered through the USB port of an external computer or from an external power source. Also shown below is an image of the circuit “bread board” mounted next to the Arduino Board. The photodiode shown is normally mounted in a weather resistant container where the detector is in full view of the sun, for most of the day, from a fixed position. It should be noted here that the photodiode current can drive a long co-ax cable with no loss of performance. Care should be taken that the cable is grounded only at the circuit board and not at the detector. This insures that no ground loops are formed, and reduces the chance of noise being injected into the circuit.

As shown in the circuit and image below, the Arduino Uno board, and Breadboard Mini, are very inexpensive and may be perched from SparkFun Electronics or other distributors. The IC is carried by many part distributors along with the resistors and capacitors. Also the Si

photodiode, VTP4085H, is available from Allied Electronics. Total parts cost is under forty dollars for the circuit shown. It should be also noted that a Bluetooth modem “BlueSMiRF” and a “Duck Antenna” may be added to the Arduino board for Bluetooth transmission of serial data. Micro SD-Card Shields, which allow on board memory of large amounts of data, are also available.





Arduino Program

The Arduino microcontroller is programmed using the Arduino Programming language which is available at www.arduino.cc along with references and all the information needed to get started. Once the Arduino programming language is placed on a computer the "Sketch" or program shown below may be copied compiled and uploaded to the Arduino board.

This program first sets-up the baud rate (115200) for serial i/o then defines the Number of points used to sample the solar intensity and ac-coupled intensity variation with the circuit gain of 425.5X. Voltage values are recorded by the ADC on inputs A0 and A1. All voltage values recorded by the ADC's are scaled from their unipolar output of 0 to 1023 to a bipolar output of -2.5 to +2.5 volts. The rms value of the ac-coupled intensity variation is calculated as well as the average intensity variation. Finally the value for the "seeing" ($4.46 \times \text{RMS Intensity variation}$) / $\langle I \rangle$ is printed to the serial port. In the initial set-up the value of $\langle I \rangle$ from the first op-amp may be displayed by un-commenting (remove //) and uploading the sketch. The value should be approximately 1 Volt in full sun at or near maximum intensity. The value is adjusted by changing the feedback resistor on IC u1. The value of $\langle I \rangle$ is used to normalize the RMS seeing when there are transparency variations or angular position variations of the sun. Once the resistor value is set the line may be commented out (//), the sketch uploaded and run.

The output of the monitor may be viewed using the Serial Monitor on the Arduino 1.0 programming tool or any other serial i/o program on a PC. Tera Term <http://ttssh2.sourceforge.jp/> is one such program which may be used via the USB port emulation of a serial port. The Sketch below is set for a Baud rate: 115200, Data: 8 Bit, Parity: none, Stop: 1 bit. The serial terminal program must be set to the same com port# that is set when the Arduino board is mounted on the PC. A simple Windows PC serial program which may be used to read (only) the serial output is included as the last c-program. This program must be configured to the proper baud rate and com port #, and then compiled. Create a new folder c:\\DATA to save the data on the PC. This SerialComm.c program may be compiled using Dev-C++ version 4.9.9.2 from <http://www.bloodshed.net> .

"Sketch"

```
/*
```

```
Solar Scintillation Seeing Monitor Sketch
```

```
Read 'Number' of analog input readings on pin 'A1' and 'A0' and find the value of  
4.46*RMS(Intensity_A1)/AVE-Intensity_A0. The result is sent to serial monitor or TXRX  
output. External Scintillation Monitor circuit gain set to 425.5X. Remove "//" to print solar  
intensity during set-up, set to aprox. 1.0 for full sun by changing FB resistor on U1.
```

```
*/
```

```

void setup() {
    Serial.begin(115200);
    #define Number 10000 // number of samples used to find RMS value
}
void loop() {
int i,sensorValue1,sensorValue2;
float Value,ValueA,ValueB,IValue,IValueB;
ValueB=0;
IValue=0;
IValueB=0;
for ( i=0; i< Number; i++)
    {
        sensorValue2 = analogRead(A1); // find the RMS value
        Value=(sensorValue2-511.)*2.5/511.;
        Value=sqrt(Value);
        ValueB=ValueB+Value;
        sensorValue1 = analogRead(A0); // find the average value of solar intensity
        IValue=(sensorValue1-511.)*2.5/511.;
        IValueB=IValueB+IValue;
    }
    IValueB=IValueB/Number+.0; // + or – small dc off-set
    //Serial.println(IValueB,2);
    ValueB=ValueB/Number;
    ValueB=sqrt(ValueB);
    ValueB=4.46*ValueB;
    ValueB=ValueB-.05; // + or – small dc off-set
    ValueB=ValueB/IValueB;
    if (IValueB<0.5) ValueB=0; // Discriminate for clouds – set output to “0”
    if (ValueB>10.0) ValueB=0; // Discriminate for clouds – set output to “0”
    Serial.println(ValueB,2);
}

```

Serial I/O Terminal C-program

```
/*  
SerialCom.c is a program used to receive (only) data transmitted on a com or  
virtual com port. The COM# and baud rate may be changed in the two lines  
indicated in the program. SerialCom.c uses standard win32 API calls. The  
received data, text file, is saved at c:\\DATA\\%Y%m%d.txt and c:\\DATA\\  
T%Y%m%d.txt. Data file %Y%m%d.txt is a text data file only, and T%Y%m%d.txt  
has time "H:M:S" plus data. Data may be plotted using gnuplot or wgnuplot.  
SerialCom.c may be compiled using Dev-C++ from http://www.bloodshed.net  
*/
```

```
#include <windows.h>  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>  
#include <time.h>  

```

```

{
    buf[n] = 0;
    for(i=0; i < n; i++)
    {
        if(buf[i] < 32)
        {
            buf[i] = ' ';
        }
    }
}
/////////////////////////////////////////////////////////////////
//Print data to stdio
printf(buf1);
printf(" ");
printf("%s\n", (char *)buf);
/////////////////////////////////////////////////////////////////
//Save time + data file at c:\\DATA\\T%Y%m%d.txt
fp=fopen(buf2,"a+");
fprintf(fp,buf1);
fprintf(fp," ");
fprintf(fp,"%s\n",(char *)buf);
fclose(fp);
/////////////////////////////////////////////////////////////////
//Save data file at c:\\DATA\\%Y%m%d.txt
fp=fopen(buf3,"a+");
fprintf(fp,"%s\n",(char *)buf);
fclose(fp);
}
Sleep(200);
}
return(0);
}
/////////////////////////////////////////////////////////////////
void TXRX(void)
{
    DCB dcbCommPort;
    //change "COM#" here and compile
    hComm = CreateFile ("COM3",
        GENERIC_READ | GENERIC_WRITE,
        0,
        0,
        OPEN_EXISTING,
        0,
        0);
    GetCommTimeouts(hComm,&ctmoOld);
}

```

```
ctmoNew.ReadTotalTimeoutConstant = 100;
ctmoNew.ReadTotalTimeoutMultiplier = 0;
ctmoNew.WriteTotalTimeoutMultiplier = 0;
ctmoNew.WriteTotalTimeoutConstant = 0;
SetCommTimeouts(hComm, &ctmoNew);
dcbCommPort.DCBlength = sizeof(DCB);
GetCommState(hComm, &dcbCommPort);
//change "baud=115200 parity=N data=8 stop=1" here and compile
BuildCommDCB("baud=115200 parity=N data=8 stop=1", &dcbCommPort);
SetCommState(hComm, &dcbCommPort);
EscapeCommFunction(hComm, CLRDTR);
EscapeCommFunction(hComm, SETDTR);
```

```
}
int PollComport(unsigned char *buf, int size)
{
    int n;
    if(size>4096) size = 4096;
    ReadFile(hComm, buf, size, (LPDWORD)((void *)&n), NULL);
    return(n);
}
```

////////////////////////////////////