HAND HELD DATA ACQUISITION DEVICE

FOR SOLAR PV VI CHARACTERISTICS UNDER FLASHING LIGHT

BY

STEVEN GRUSPIER

A THESIS
SUBMITTED TO THE FACULTY OF

ALFRED UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

ALFRED, NEW YORK

SEPTEMBER, 2015

HAND HELD DATA ACQUISITION DEVICE FOR SOLAR PV VI CHARACTERISTICS UNDER FLASHING LIGHT

BY

STEVEN GRUSPIER

B.S. ALFRED STATE COLLEGE (2012)

SIGNATURE OF AUTHOR____

APPROVED BY____

XINGWU WANG, ADVISOR

____

WALLACE LEIGH, ADVISORY COMMITTEE

____

JOSEPH ROSICZKOWSKI, ADVISORY COMMITTEE

____

CHAIR, ORAL THESIS DEFENSE

ACCEPTED BY____

DOREEN D. EDWARDS, DEAN

KAZUO INAMORI SCHOOL OF ENGINEERING

ACKNOWLEDGEMENTS

I would like to thank my Advisor, Dr. Xingwu Wang, for his constant support, assistance, advice, and encouragement throughout the process of completing this thesis.

I also would like to thank Dr. Leigh for his assistance and support.

Thanks to other faculty/staff and other students, including Rosalie DiRaimondo, Dr. Jianxin Tang, John Gallipeau, and Janelle Gallipeau.

Another special thanks to my friends and family for their encouragement through the entire process.

TABLE OF CONTENTS

Page

Acknowledgments …... iii
Table of Contents …... iv
List of Tables …... vi
List of Figures …... vii
Abstract …... x

I INTRODUCTION …... 1
A. Hand Held Data Acquisition Device Criteria …... 3
B. Solar Panels …... 4
C. Microcontroller ….. 5
D. Oscilloscope …... 6
F. Solar simulator.. 7

II FIRST DESIGN OF ACQUISITION DEVICE ….. 8
A. Determining Time Constraints 8
B. Controlling The Solar Simulator 9
C. Choosing a Load Resistor 11
D. Main Circuit Design 12

III SOFTWARE ….. 13
A. AD9850 Programming ….. 13
B. Analog Data …... 14
C. Serial Programming ….. 15
D. Main Acquisition Program …... 15
E. Receiving Serial Data …... 18

IV PRELIMINARY EXPERIMENTATION 21
A. Different Intensities …... 21
B. Amplitude and Offset …... 22
C. Different Waveforms ….. 24
D. Different Frequencies …... 27
E. Different Resistor ….. ..30
F. Shielding …………………………………………………………………………32

V PROBLEMS ENCOUNTERED & POSSIBLE SOLUTIONS ……………...…33
A. Load Resistor ………………………………………………………………….33
B. Shielding ………………………………………………………………………34
C. Choosing a Transistor …………………………………………………………34
D. Simultaneous Data Acquisition ……………………………………………….34
E. Fast Acquisition Code ………………………………………………………...37

VI FINAL DESIGN, TESTING, & VERIFICATION 39
A. Signal Generator 39
B. Batteries …... 41
C. Analog Inputs …... 42

D. Microcontroller …..….. 42
E. Electrical Circuitry …...….. 43
F. Packaging …..…44
G. Final Testing ……………………………………………………………..45
VII RESULTS AND DISCUSSION …... 48
VIII CONCLUSION …... 50
IX FUTURE WORK …... 51
REFERENCES …... 52
APPENDIX A Solar Simulator Flashing Equipment **…... 54**
APPENDIX B Oscilloscope DSO1002A Program **….. 57**
APPENDIX C Oscilloscope Capturing for Solar Simulator **.. 60**
APPENDIX D Arduino Program Code (Master)**.. 63**
APPENDIX E Arduino Program Code (Slave)**.. 69**
APPENDIX F Python Program Code **…..73**
APPENDIX G Analog to Digital Converter (Atmega328p)**.. 75**

LIST OF TABLES

Table 1. Input/Output Table………………………………………………………….5

LIST OF FIGURES

Figure 1. Schematic for solar panel acquisition circuit..2

Figure 2. Extrapolation of an ideal solar cell…………………………………………...3

Figure 3. SX-30U solar panel datasheet..4

Figure 4. Voltage output from 30 watt panel..6

Figure 5. Solar simulator hardware trigger...8

Figure 6. Solar simulator software trigger...10

Figure 7. Flowchart for hand held program..17

Figure 8. Flowchart for python serial transfer program...19

Figure 9. Voltage response at 2.5 Vp and 1.25 VDC offset...22

Figure 10. Voltage response at 3.0 Vp and 1.50 VDC offset...23

Figure 11. Voltage response at 3.5 Vp and 1.75 VDC offset...23

Figure 12. Voltage response at 4.0 Vp and 2.00 VDC offset...24

Figure 13. Voltage response for a sinusoidal waveform..25

Figure 14. Voltage response for a square waveform..25

Figure 15. Voltage response for a triangular waveform (50% symmetry).........................26

Figure 16. Voltage response for a triangular waveform (100% symmetry).......................26

Figure 17. Voltage output with 1 kHz signal...27

Figure 18. Voltage output with 900 Hz signal...28

Figure 19. Voltage output with 800 Hz signal...28

Figure 20. Voltage output with 700 Hz signal...29

Figure 21. Voltage output with 600 Hz signal...29

Figure 22. Voltage output with 500 Hz signal..30

Figure 23. Voltage output using an IRF530N Transistor..30

Figure 24. Voltage output using an TIP31 Transistor..31

Figure 25. Voltage output using an BS170 Transistor...31

Figure 26. Voltage without a metal enclosure..32

Figure 27. Voltage with a metal enclosure...32

Figure 28. Voltage output with a 2.5 ohm and 1.0 ohm load..33

Figure 29. Schematic using Arduino master-slave config...35

Figure 30. Flowchart for Arduino master-slave config...36

Figure 31. Schematic for signal amplification circuit...40

Figure 32. Completed amplification circuit..41

Figure 33. Final prototype before packaging...45

Figure 34. Data captured from hand held device..46

Figure 35. Voltage output from multisim 11 simulation...46

Figure 36. IV Curve from generated from hand held device...47

Figure 37. Elinchrom Digital 2400RX Power Pack...54

Figure 38. Settings for Elinchrom Digital 2400 RX Power Pack......................................54

Figure 39. Solar Simulator Light Source using Xenon Arc Lamp....................................55

Figure 40. Overhead view of collimation of light rays..56

Figure 41. Agilent IntuiLink Software Scope Selection...58

Figure 42. Agilent IntuiLink Software Connecting to Oscilloscope.................................59

Figure 43. Agilent 1002A Oscilloscope Representation from Manual.............................60

Figure 44. Agilent 33210A Function Generator Representation from Manual................62

Figure 45. Analog to Digital Converter (Atmega328p)...75

ABSTRACT

A device was developed that is specifically designed to acquire data from a solar panel within this short time frame with the use of a microcontroller. The microcontroller was used to make a portable device that could capture and transfer this data to a computer for further analysis. Circuitry was developed to connect between a solar panel and a fixed load resistance. The voltage across the load resistor, and other internal components would be captured by two microcontrollers to simultaneously acquire data. This data could be then used to illustrate the IV curve of a solar panel connected to this apparatus. Two prototype designs were created for acquisition. The first being larger scale, using an oscilloscope and function generator. The second design eliminated the need for these larger pieces of equipment by replacing them with microcontrollers, and a direct digital synthesizer (DDS) respectively. Experiments were performed on two solar panels under the ideal solar radiation of 1000 w / m^2, using different input signals. One prototype using two different oscilloscopes, and one hand held prototype were designed and compared. The results proved favorable as the three experiments gathered similar data to the solar panel manufacturers data.

INTRODUCTION

Hand held devices have recently, become very popular for acquiring data for analysis of electrical systems. For solar photovoltaic applications, devices have been designed and created, but have not yet seen wide adoption at the installation level. Due to the growing concerns of faulty solar panels, it is vital that each panel is well tested to prevent the installation of such a panel into a solar array. The New York Times suggests that the solar industry is facing a quality crisis and that this is a more urgent matter than most realize. It was reported that a solar panels located on the rooftop of a warehouse in the Inland Empire region east of Los Angeles failed two years into the expected 25 year life span. Coatings that were designed to protect the solar panels had disintegrated in addition to other defects, which caused a fire that took the system offline. The article further explains “There are no industrywide figures about defective solar panels.”, and furthermore “...when defects are discovered, confidentiality agreements often keep the manufacturer’s identity secret, making accountability in the industry all the more difficult”.[1]

The data acquisition device is intended to be used for a flash based solar simulator due to the ability to effectively acquire data within a window of time two milliseconds long. The process between capturing the data, and exporting the data to a computer takes less than 5 seconds. As a result of this characteristic, a hand held acquisition device is created that can acquire data instantaneously could potentially eliminate the installation of faulty panels by indicating significant deviations from a standard solar panel manufacturer specification during the installation process by allowing installers to acquire data within a few seconds.

Existing solutions use a method of capturing IV Curve data that relies on varying the resistance while taking a voltage and current reading at each resistance, from a high impedance to a negligible resistance, to create an IV curve. This method is very easy, but for a flash based solar simulator, is too slow because the resistance can not be instantaneously varied between a low resistance for determining the short-circuit current (Isc) and a high resistance to determine the open-circuit voltage (Voc) while capturing voltage and current data. The method used in this device involves a simplified version of this method, using a resistor in series with a transistor connected to a solar panel. A sinusoidal waveform is used to vary the voltage applied to the gate of the transistor. A voltage probe 1 is connected to the positive lead of the solar panel, and voltage probe 2 is connected to the drain of the transistor. Figure 1 illustrates the basic circuitry for the device. As the solar simulator flashes, the energy produced by the solar panel (in this case, a 30 Watt panel with an open-circuit voltage of 21 Volts) would be applied to the 2.5 ohm load. This particular load resistor configuration was chosen due to the low resistance and power dissipation characteristics. In figure 1, note the node labeled 1 and the node labeled 2. For future reference node 1 will be referred to as vdrain and node 2 as vsolar. When the voltage applied to the gate of the transistor is zero, the voltage at volar and vdrain are equivalent to the open-circuit voltage. As the voltage applied to the gate is increased, the voltage difference between the vsolar and vdrain increases. This difference in voltage divided by the resistance in the circuit allows us to find the current through the circuit. Using the voltage across the resistor and the current through the circuit, an IV-Curve could be produced.

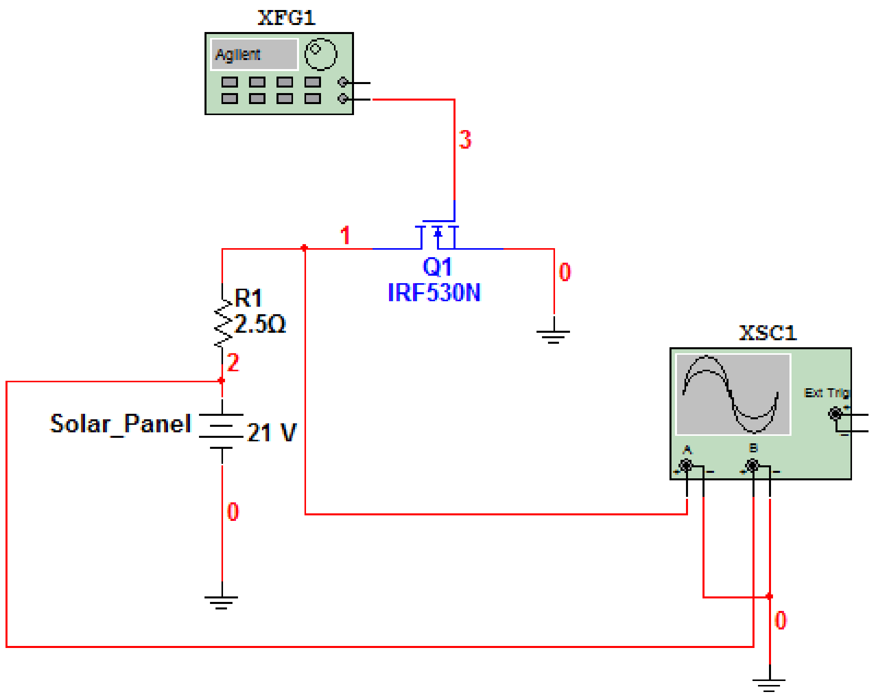


Figure 1. Schematic for basic solar panel acquisition circuit.

The most significant draw back to this method is the inability to reach the short-circuit current. To remedy this problem, the data at that point will be found by extrapolating the data up to that point. Figure 2 illustrates the method of how extrapolation will work.

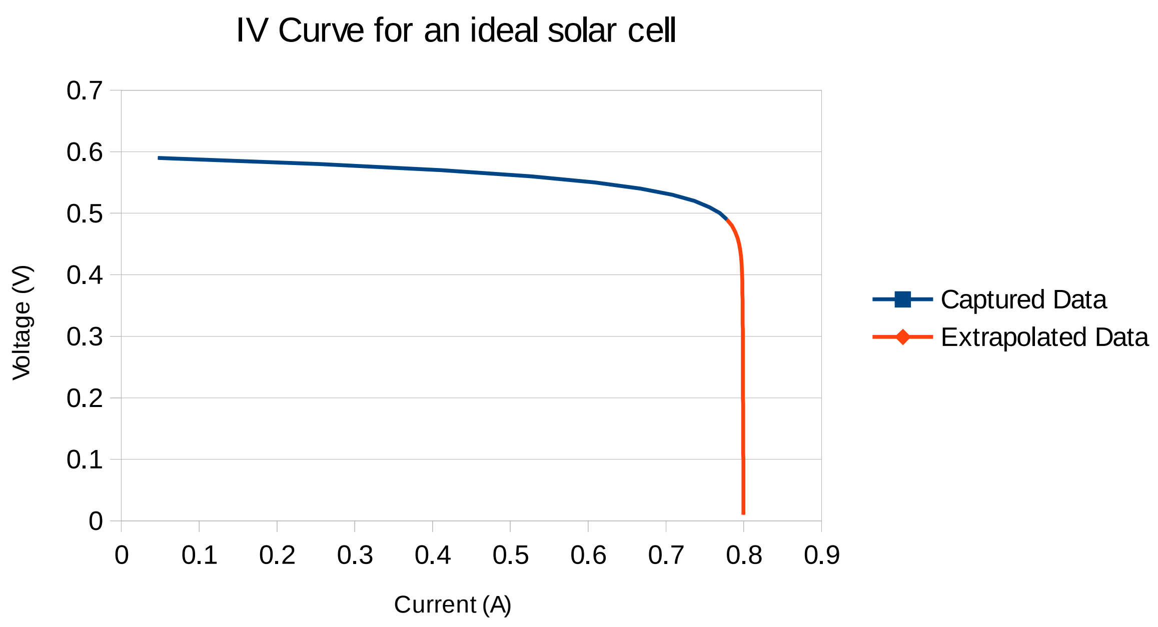


Figure 2. IV Curve with extrapolation performed.

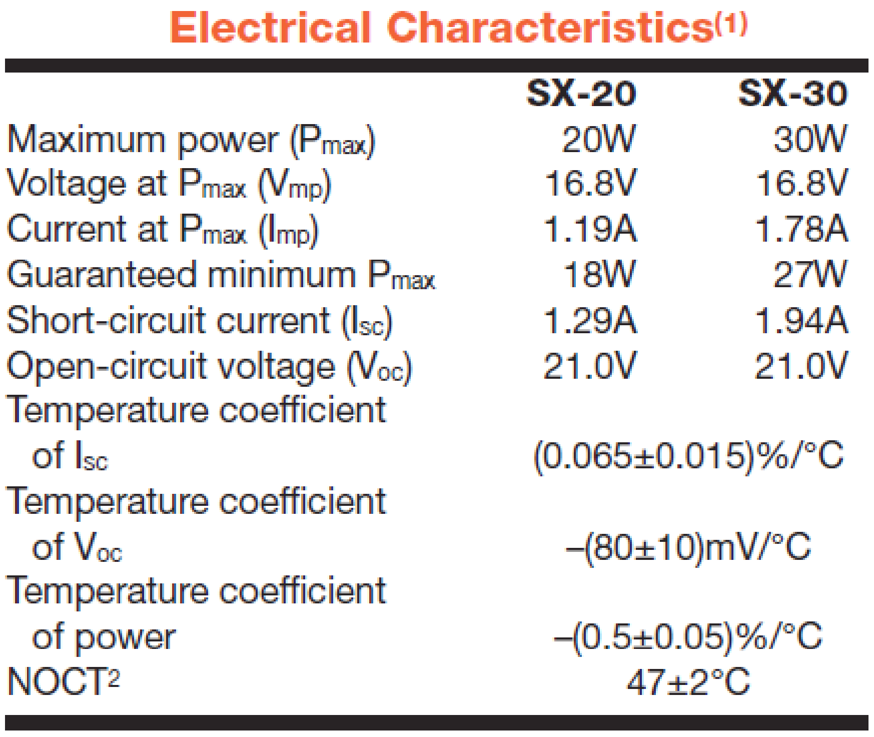
A. Hand Held Data Acquisition Device Criteria

The hand held data acquisition device created in this project needed to meet specific requirements:

- The device should be fast enough to acquire data from a flash based solar simulator
- The device should be easy to use
- The hardware should be portable
- The circuitry should be battery powered to allow for mobility

B. Solar Panels

The SolarexTM SX-30U solar panel was used in this study. To determine if results were accurate, it was imperative that similar panels could be used, and create similar results. The characteristics of the Solarex^{TM} SX-30U solar panel is listed in Figure 3.



Electrical Characteristics(1)

	SX-20	SX-30
Maximum power (P_{max})	20W	30W
Voltage at P_{max} (V_{mp})	16.8V	16.8V
Current at P_{max} (I_{mp})	1.19A	1.78A
Guaranteed minimum P_{max}	18W	27W
Short-circuit current (I_{sc})	1.29A	1.94A
Open-circuit voltage (V_{oc})	21.0V	21.0V
Temperature coefficient of I_{sc}	(0.065±0.015)%/°C	
Temperature coefficient of V_{oc}	–(80±10)mV/°C	
Temperature coefficient of power	–(0.5±0.05)%/°C	
NOCT2	47±2°C	

Figure 3. SX-30U solar panel datasheet[2].

These values provided an approximate way to determine if the captured values from the acquisition device are accurate. These solar panels were used because the power produced from the 30 watt solar panels were significantly lower than other available solar panels in the lab. To test the data acquisition device, it was safer to use solar panels that had a lower chance of damaging components, equipment, and most importantly, injuring individuals working on this project.

C. Microcontroller

A microcontroller is a microprocessor on an integrated circuit, which consists of a CPU, memory, and a combination of inputs and outputs. For the hand held device, the Atmega328P microcontroller is the main component used to control, and acquire data from the solar simulator. The Atmega 328P has 14 digital I/O pins that can be used as well as 6 analog input pins. For this project only 2 analog inputs are needed for data acquisition.[3] Only 2 digital pins are absolutely needed; one to activate the solar simulator, and another indicate the status of the acquisition device. More digital pins were used to indicate non-essential steps of the acquisition process to make it easier for the user. The inputs and outputs are shown in Table 1. The clock speed of the Arduino Uno is also a consideration since the device must be fast enough to capture a reasonable amount of data to create an IV curve. The microcontroller features a 16 MHz ceramic resonator, which allows the device to operate at the maximum frequency specified by the Atmega 328P datasheet.[3]

Table 1. Input/Output Table

Name	Type	Description
Vdrain	Analog Input	Acquires voltage across the IRF530N transistor's drain
Vsolar	Analog Input	Acquires voltage across the solar panel
Push Button	Digital Input	Starts acquisition process
LED Indicator	Digital Output	Allows a user to see when the solar simulator is ready to be triggered again
Solar Simulator	Digital Output	Used for triggering the solar simulator using the sync port on the Elinchrom Power Pack
LED Power	Digital Output	Allows a user to see if the device is on

Between the different microcontrollers that were available, it was determined that the simplest device to use would be the Arduino Uno. The Atmega238P microcontroller was chosen based on criteria including its low power design, availability of analog ports in addition to digital IO ports, and ease of programming. The simplicity of the Integrated Development Environment (IDE) used for writing code for the Microcontroller was a significant factor. Any Arduino microcontroller sold uses the Arduino IDE for writing code. The Arduino IDE is very mature after years of development. Not only is the software for writing code simple to use, but the included libraries make it easy to program by simplifying the uses of the inputs and outputs by using functions. These functions automate processes such as reading inputs or changing the state of an output making writing code easier, by allowing a programmer to focus on the end goal rather than creating functions for themselves to use later. The Arduino IDE is compatible with C and C++ for writing code.

D. Oscilloscope

To test the basic theory of data acquisition, an oscilloscope was used. Two different oscilloscopes were compared against one another to determine which one could acquire the most data points. The Agilent™ DSO1002A oscilloscope and the Agilent DSO3062A oscilloscope were compared to determine which one could produce the best results for testing the method of acquisition. These oscilloscopes share similar qualities in that they both have a 60 MHz bandwidth have a sampling rate of 1 GSa/s per channel. The DSO1002A was used due to portability and because it seemed to perform better overall during testing. For a brief period, Test Equity LLC allowed for one demo unit of their DSO3024A oscilloscope to be used for testing with the solar simulator. The DSO3024A oscilloscope featured a 200 MHz bandwidth with a sample rate of 4GSa/s per channel. Both oscilloscopes have a 8-bit digitizer, which divides a 10 Vpp input range into 2^8 = 256 levels of 39 mV each. The dynamic range is 49.93 dB for both oscilloscopes as well.

E. Solar Simulator

The solar simulator purchased by Alfred University, was custom built by Sciencetech^{TM} Inc. This simulator uses flash based technology to control the source of light. This type of solar simulator has the drawback of having a short span of time that can be used to acquire data. This short time span of time, where the solar simulator light source is triggered, has the added advantage of minimizing the heating effects of the light source, because the solar simulator is active for less than a second. Most acquisition systems use load varying methods as described in the previous section. These systems are too slow in the case of flash based technology, meaning that using this different acquisition method would be beneficial for this application.

FIRST DESIGN OF ACQUISITION DEVICE

The purpose of the first prototype of the data acquisition device was to test the acquisition method and determine any factors that needed to be taken into consideration before creating a hand held prototype. This particular design used the Agilent DSO1002A to acquire the voltage data, and the Agilent 33210A Function Generator to vary the voltage in the circuit.

A. Determining Time Constraints

To begin the design of the hand held data acquisition device, it was important to first understand the voltage transients. Since the voltage signal from the solar panel would only appear for a short amount of time without repeating, the data needed to be recorded by the DSO1002A for analysis. The voltage data was recorded successfully, and yielded the results shown in Figure 4 when the solar simulator was set to an f-stop rating of 90.

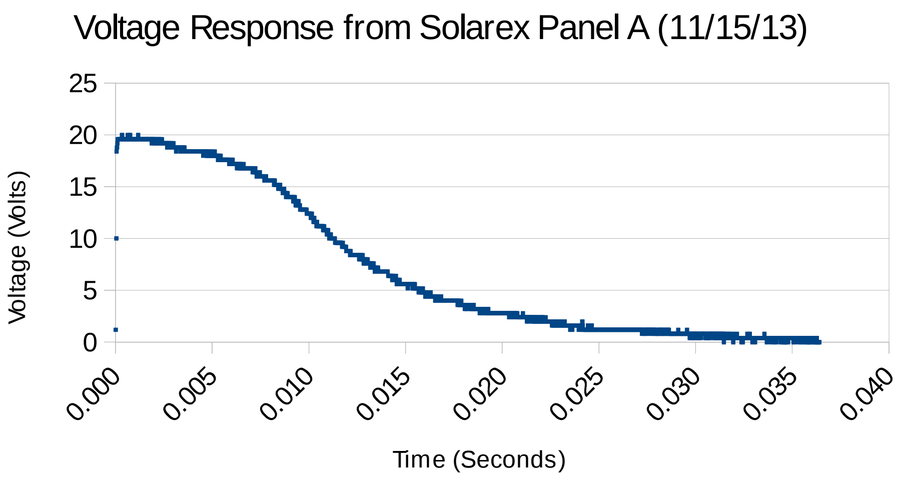


Figure 4. Voltage output from 30 watt solar panel using solar simulator.

By by close observation it was determined that the voltage remained constant for approximately two milliseconds. This meant that all of the data that needed to be captured could only be acquired during these two milliseconds. Another interesting observation is that the voltage curve has two time-constants. One time constant from 0 milliseconds to 11 milliseconds, and from 11 milliseconds to 37 milliseconds. Most likely due to the effects of the light flashing for only a short amount of time, and the capacitance of the solar panel.

B. Controlling The Solar Simulator

Between performing experiments with the solar simulator and communicating with the manufacturer it was determined that the solar simulator should only be triggered after waiting a minimum of 11 seconds after the solar simulator had previously been triggered. The reasoning behind this is that the Elinchrom Digital RX 2400 power pack system contains capacitors that need 11 seconds to charge to a usable voltage before application to the Xenon Arc Lamp. This would prevent possible damage to the equipment.[5]

To safeguard against premature triggering the solar simulator, a circuit was designed based on an RS flip flop circuit in combination with a 555 timer circuit. The circuit based on basic components was designed and then tested in Multisim as seen in Figure 5. The equation $\tau = 1.1\ RC$ was used to determine the time delay for the circuit.[4] In this case, R is noted as R7 and C is noted as C1, within the Multisim circuit design. From this it was determined that using a 100 kΩ resistor and 100 μF capacitor would result in a 11 second delay after triggering the solar simulator before it could be triggered again.

After using this trigger circuit for several weeks and consulting Sciencetech™ Inc., it was determined that for more stable results, a longer delay was needed for the voltage within the capacitors to stabilize and reach a steady state because even though the power pack was charged to usable voltage, this voltage was not completely stable, which produced errors during the acquisition process.[6]

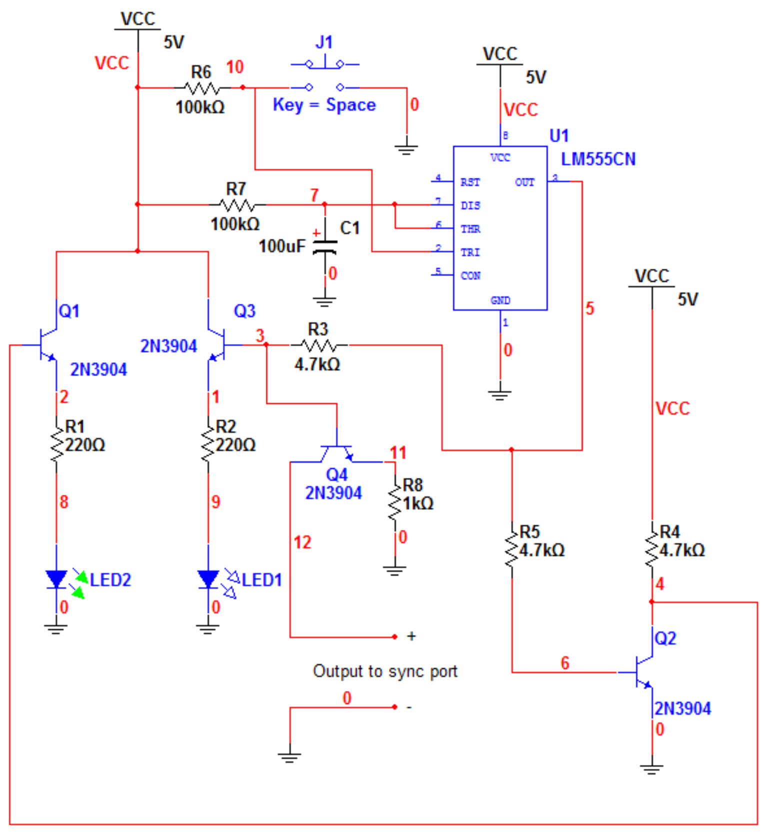


Figure 5. Schematic controlling the solar simulator using basic components.

Due to the static nature of basic circuit components combined with difficulties of finding specific components, a circuit based on the Arduino Uno microcontroller was created instead to replace this circuit. The configuration was simplified significantly as seen in Figure 6. Since time values could easily be altered through software changes rather than hardware changes, this was considered the best approach. After several tests, it was determined that a time delay of 30 seconds yielded consistent results.

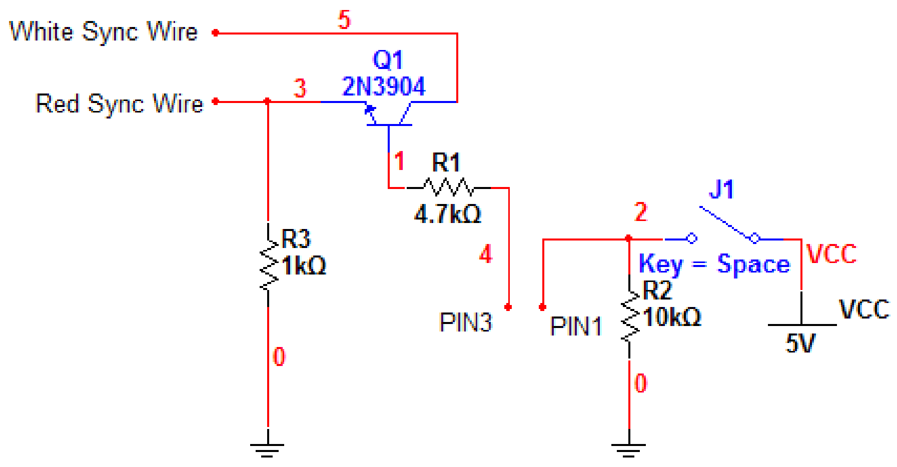


Figure 6. Schematic for Arduino controlled for simplified solar simulator triggering.

C. Choosing a Load Resistor

An IV-Curve begins at the open-circuit voltage and ends at the short-circuit current. To approach the short-circuit current in a typical load varying circuit, the resistance within the circuit starts using a high resistance, and decreases to a point of negligible resistance. Since the method used in this thesis relies on a constant resistance and a varying voltage, the resistance needed to be small, but big enough to measure a voltage difference across the resistor. Another consideration were the voltages and currents that were going to be introduced to this circuit. To make a decision on the maximum power rating of the resistor, the solar panels in the solar simulator lab were used. The biggest solar panel within the solar simulator lab has a maximum power rating of 220 Watts. For added safety, it was determined that using a resistor with a minimum power rating of 300 Watts would be sufficient. After determining this minimum requirement, two Ohmite 5Ω resistors with a power rating of 1 kW were used for use in the solar simulator experimentation. To reduce the resistance, the two resistors were used in parallel to make a total resistance of 2.5 ohms. This configuration is acceptable to use since it is a low resistance, and the resistors used exceed the maximum power ratings more than three times the determined minimum power rating.

D. Main Circuit Design

To vary the load, the Agilent 33210A Function Generator was used to vary the gate voltage while the solar simulator was triggered by the Arduino Uno, and the data was acquired by the Agilent DSO1002A Oscilloscope. To acquire the data, the oscilloscope needed to be configured to detect a rising edge, and record when a certain voltage threshold was reached. In this case, a 1 Volt trigger point was used since the solar panel voltage would quickly reach 21 volts when the solar simulator was triggered due to the flash. The 1 volt trigger point would also eliminate possible electrical interference prematurely triggering the oscilloscope. Due to the lack of light within the solar simulator operating area because of thick curtains in place to prevent external interference from objects and light sources in addition to safety concerns regarding the flash emitted from the Xenon Arc Lamp, the solar panel voltage is zero. The Function Generator needed to be set with specific settings to allow for the transistor to remain within the active region and cut-off regions to acquire useful data.

SOFTWARE

The final design of the hand held device involved much more thought than the first prototype. The final prototype features hardware that must either send or receive signals using the Arduino Uno. Multiple indicator LEDs were used to indicate the status of the data acquisition device, as well as the state of the device (on or off). A push button was used to activate the device. These hardware components, among other parts, needed to be managed by the Arduino Uno microcontroller.

A. AD9850 Programming

The AD9850 circuit board was purchased from Ebay.com, and solved many prototyping problems with soldering such a small component, and the added benefit of having many people who had previously made open source libraries to easily use with this specific chip. In this case, the developer of the libraries used for this project, Radio Artisan made freely available libraries for using the AD9850 Chip with an Arduino microcontroller. The libraries are able to be downloaded through the sourceforge website using the web address: “*http://sourceforge.net/projects/k3ngarduinaddds/files/*”.

To use the AD9850 library, the dds.h and dds.cpp files need to be moved to the proper location to be recognized as usable library files. For windows, the proper location is: My Documents\Arduino\libraries\. For Linux the proper location is /home/[username]/Sketchbook/. The AD9850 library files need to be moved into this location, then placed in a new folder, which in this case, was named “dds” to match the name used. After loading the libraries through the Arduino software, the libraries could be used by simply adding an include in the header of the data acquisition code.

The functions for configuring the AD9850 are straight forward and simple. First, the function *dds ddschip* is used to pass information including, the type of AD98XX series chip being used, the pins being used for data, load, and clock functions, and the frequency of the crystal used for the device. In this case, the complete function used was *dds ddschip(DDS9850, 8, 6, 7, 120000000LL);* The digital IO pin numbers 8, 6, and 7 were used because these were used in the programming example made by the developer of the library. Due to different pins having dual functionality such as pulse width modulation (PWM) in addition to being a normal digital IO, it was thought best to keep this pin configuration the same. The second function used, *ddschip.calibrate(-.0418)* adjusted the frequency to be exact. Due to differences in manufacturing, the crystals used may be slightly different. This difference leads to the output frequency of the AD9850 to be slightly off. To calibrate this, the Agilent DSO1002A oscilloscope was used to find the exact frequency before any calibration was made. The raw frequency of the AD9850 was 1.0017090 kHz. After trial and error, the calibration value was found to be -.0418, which allowed the AD9850 chip to achieve the exact frequency desired. The final function used in the code is *ddschip.setfrequency(1000LL)*. This function was used in the code to turn on and off the AD9850 by setting the frequency (in Hertz) between 0 (off) and a non-zero frequency (on), in this case 1000.[8]

B. Analog Data

When the device is turned on, all of the digital ports used for the AD9850 circuit board are set to either send of receive voltage signals, but the analog ports had the special function of receiving voltage signals from the main solar simulator acquisition circuit. To capture the voltage readings, arrays were used to store the data in memory. This method allowed for the voltages for each analog input to be gathered separately, and temporarily stored.

At this point, the data has been collected, but the data is only useful if it can be viewed, which is not the case if all of the data is stored on the Arduino. To solve this problem, serial communication though the built in USB port on the Arduino Uno was used. By using the built in USB connector, the Arduino Uno was able to be recognized by any operating system.

C. Serial Programming

The benefit of using an Arduino microcontroller, in the programming aspect, is that many functions of the Arduino are simplified into libraries. These libraries and the functions contained within are very well documented, and remove a lot of the hassle of using normal C or C++ code. In the case of serial communication, a readily available library for sending data could be used. Using the *serial.println* function, all of the analog data was pushed through the USB connection. To configure the serial connection, the command *serial.begin(9600);* is used. The purpose of this command is to set the baud rate of the serial connection. It is important to have the same baud rate setting on the sending and receiving ends, otherwise the connection will either be unsuccessful, or have unexpected results.

D. Main Acquisition Program

The functioning of the acquisition program is very straight forward. When power is applied to the circuit, the ports are configured, the AD9850 is configured, the status LED is turned on, and the program then waits for the user to push the push button to start the acquisition process. When the push button is pressed, the Status LED is turned off to indicate that the process has been started, and the AD9850 chip is set to the desired frequency for the signal applied to the transistor. This is done before the solar simulator is triggered because of the small amount of time the flash from the solar simulator provides a stable voltage. After the solar simulator is triggered, the analog ports are read, and the data acquired is inserted into separate arrays. This process is continued until a set number of points are acquired from both analog ports. The amount of points able to be gathered was not made a static number because this device could also be used for solar panel installations, and so having more data would allow for a more accurate representation of data. To change the number of data points gathered by each analog input, change the value of *points* in the code. After acquiring all the data, the program proceeds to export the values over serial communication. After completing this, the AD9850 is turned off by setting the frequency to 0 Hz. To meet time delay requirements, the code waits for 30 seconds before allowing the user to trigger the solar simulator again. After waiting 30 seconds, the program turns on the status LED, and waits for the user to press the push button to re-run the program. To observe the programming logic in the form of a flow chart, see Figure 7.

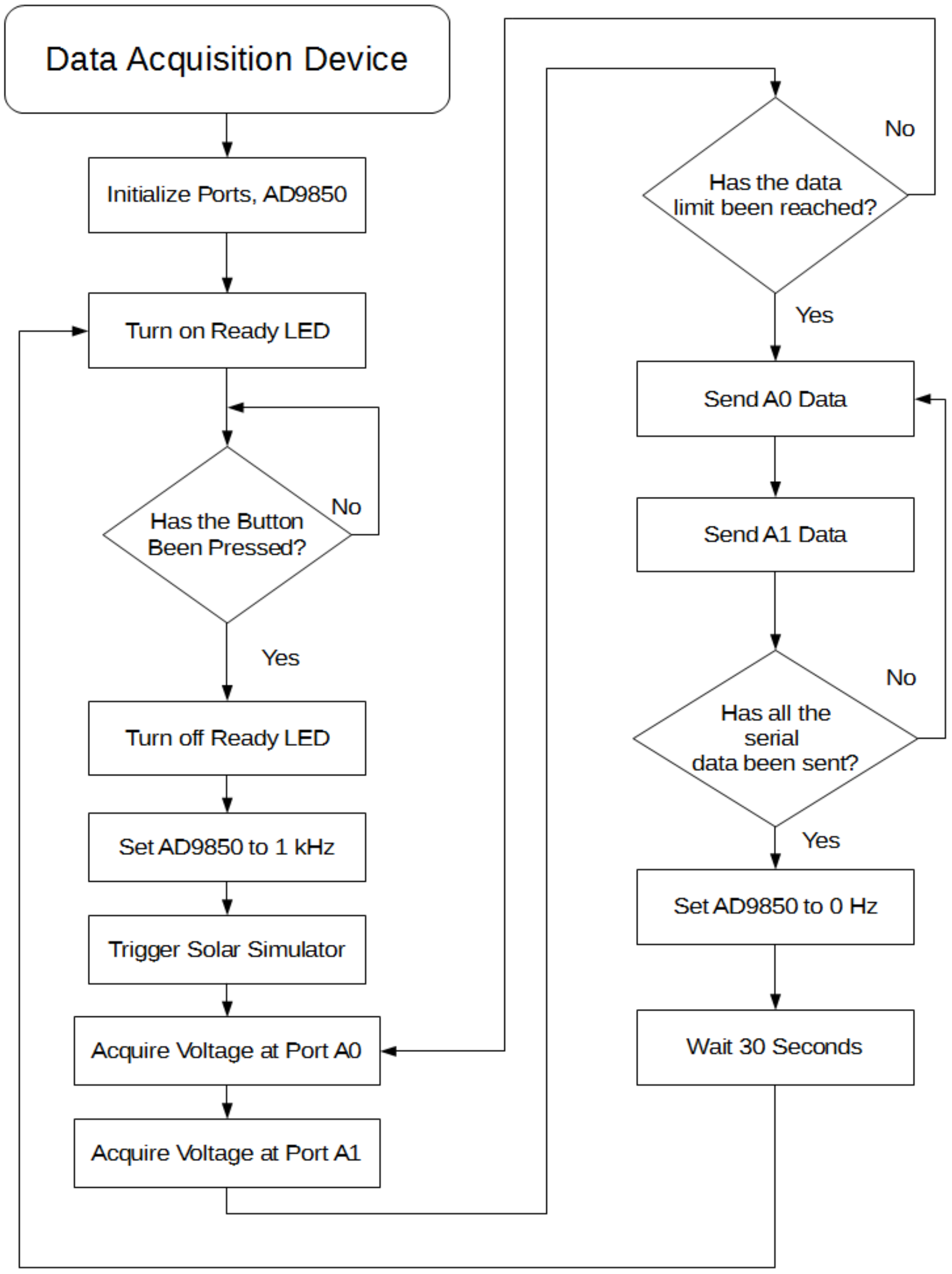


Figure 7. Hand held acquisition device program flowchart.

E. Receiving Serial Data

To receive the serial information from the Arduino Uno, the Python programming language was used. Python was used rather than C or C++ for a few different reasons. Python has the advantage of being very easy to program with because it is a high level language compared to C or C++. Python does not have to deal with special memory allocation, or the need to make complicated functions to solve simple problems. Python also has the benefit of being completely platform independent. This program can be ran on any type of hardware, or operating system, as long as the operating system supports Python. All of the most common operating systems have support for Python (Windows, Mac, and Linux).

Since Python programs are compiled each time the program is executed, it is also easy to change code at any point without having to use special tools. All python files have the file extension *.py*. These *.py* files, which text files that can easily be edited with low level text editors such as Microsoft Notepad, or Notepad++. The Python tools for creating and testing programs can be found online at python.org. It is important to note that the version of Python used to create and run this program is 2.7. The newer 3.4 version of Python does not have legacy support for code that was built for version 2.7. The decision to use version 2.7 stemmed from the fact that the majority of information involving version 2.7 is well established, and still used to program today.[9]

To connect to the Arduino specifically, the Python program must be given the device address of the microcontroller. In Windows, this would be referred to as the com port and is typically *COMX*, where *X* is the number of the communications port. In Linux, this would be called the device address and is typically */dev/ACMX*, where *X* is the number of the port. To communicate with the serial port, the command *serial.Serial("COMX", 9600);* is used to call the serial port within the code. It is important to note that the baud rate used in this command must be the same as the baud rate set for the Arduino Uno serial communication, otherwise the Arduino and the Python program will not communicate with each other.[10] The *read()* function is used read the serial data as each line is transmitted.

The Python program replaces the internal software functions of the Oscilloscope, which store the data into a text file and plotting the data. The program gnuplot is used to graph the data collected by the hand held data acquisition device. A flowchart of the logic behind the device can be seen in Figure 8. After the program is started, the device address is used along with the baud rate of the Arduino Uno to allow the Python program to interact with the microcontroller. A text file is generated (or opened), and then all text is removed from the file. Once this is complete, the program reads the first serial string.

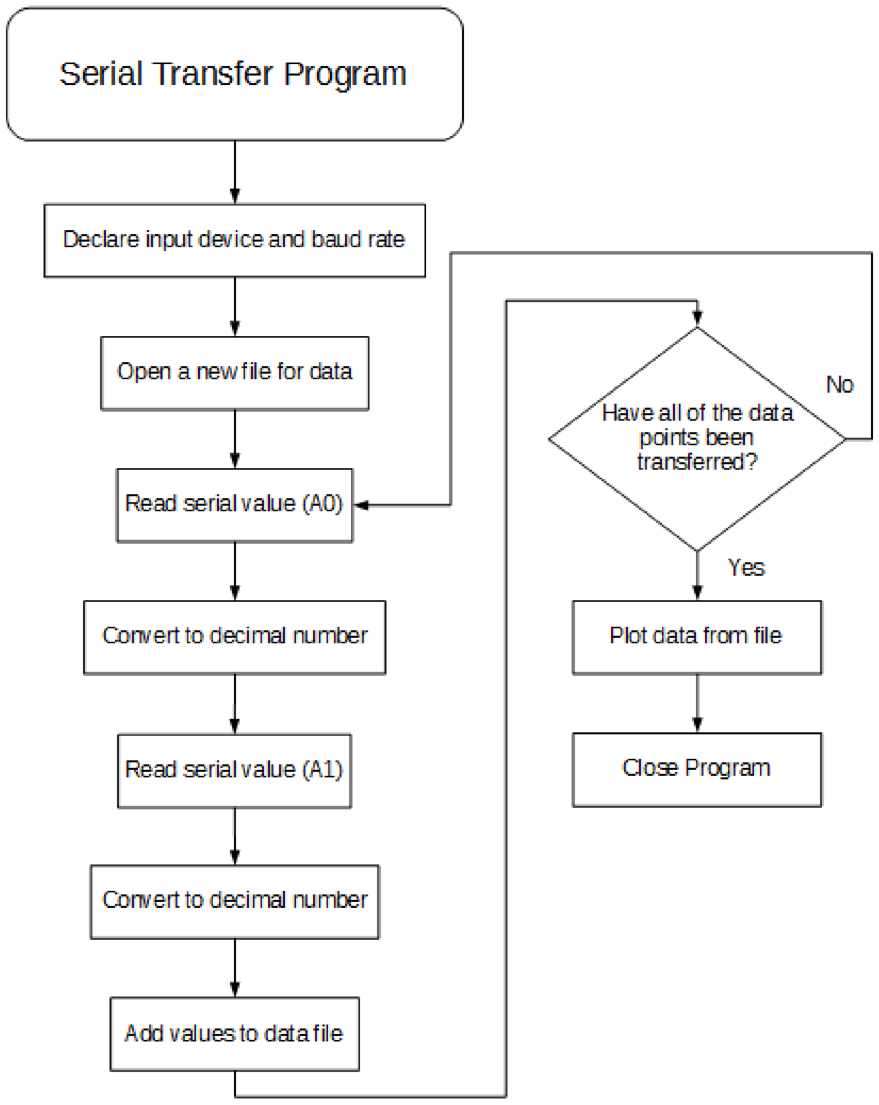


Figure 8. Python serial transfer program flowchart.

This is the first voltage read from analog input A0. The Value of A0 is on a scale of 0 to 1024 because this is the analog to digital converter, which is changes the analog signal into an integer equivalent. For these values to have meaning, these values were converted into decimal numbers. To accomplish this, the integer number was divided by the value 1024, then multiplied by the maximum voltage input.[11] In this case, the data acquisition was made for voltages up to, but not exceeding, 40 Volts.

The output values of the serial communication between the Arduino and python are serial strings. Since they were declared strings, many errors were produced in the Python code when trying to do math with the data from the Arduino because the strings appeared to be integers at first glance. To remedy this situation, the strings were forced to be considered decimal numbers using functions that temporarily allowed the strings to be used in mathematical functions without producing any errors. After the conversion from an integer to a decimal number, the same process is repeated for the analog input A1. These two decimal equivalents of the voltages taken from analog inputs A0 and A1 are then stored in a data file called *rawdata.dat*. The data file is tab delimited so that the file can be easily opened in a spreadsheet program such as Microsoft Excel. The process of receiving and converting the analog inputs repeats until the number of points in the program is reached. Similar to the Arduino program, the Python program also uses a set of points for determining how much data needs to be processed. This number is set within the Python program, and is not taken from the Arduino program. The number of points in each program needs to match, otherwise the Python program will crash with an error, or just continue to wait for more values. The data file can still be read if either case occurs, but this means that the graphing process must be carried out manually.

After all of the data is transferred, the data is plotted using Gnuplot, and the program closes. Gnuplot is a command line utility that is available for Windows, Linux, and Mac. The program is open source, and is freely available online.[12] For windows, MATLab was considered for use as it is a standard software package that is used, and can also be integrated into python.

PRELIMINARY EXPERIMENTATION

Experiments were attempted to maximize the data points that could be acquired while maintaining accurate readings. Each of the following experiments tested different aspects of the circuit, which furthered this progress. The first few experiments were performed using the oscilloscope and function generator to capture the data while trying to determine the best configuration to use for the final hand held device.

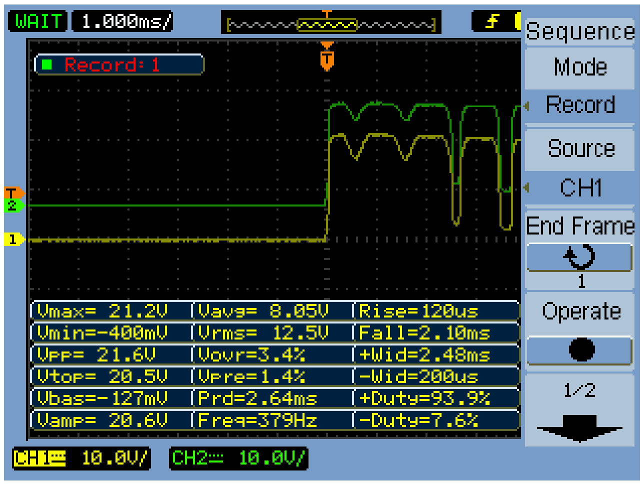
A. Different Intensities

Before tests could be performed with certainty, the intensity of the solar simulator needed to be examined. The intensity of the solar simulator needed to be roughly equivalent to the sun. This number was not calibrated to the fullest extent due to lacking equipment such as a calibrated reference cell to exactly determine if the intensity emitted from the Xenon Arc Lamp is equal to one sun, or 1000 W/m^2. A Solarex 30 Watt solar panel was used for testing. Oscilloscopes were used for all initial experiments leading up to the results of the hand held device.

To determine which setting on the Elinchrom Digital RX 2400 was closest to the value of one sun, the open-circuit voltage at each intensity was examined. The intensity setting was tested using three different settings: 6.5, 7.0, 7.5, and 8.0. The decision to use these values was derived from the previous experiments done to test the triggering of the solar simulator because an intensity of 6.0 produced a voltage that was visibly too low to be accurate. Figure 1 from the Solarex solar panel datasheet shows that the 30 Watt model of the solar panel produces an open-circuit voltage of 21 Volts, which was visibly much higher than the value seen on the oscilloscope during initial testing. To determine which setting was the best, the captured open-circuit voltage values were compared against the rated open-circuit voltage of the solar panel. To perform the experiment, the negative and positive leads of the solar panel were connected directly to an Agilent DSO1002A Oscilloscope. This apparatus ensured that no external factors could change the voltage readings from the solar panel.

B. Amplitude and Offset

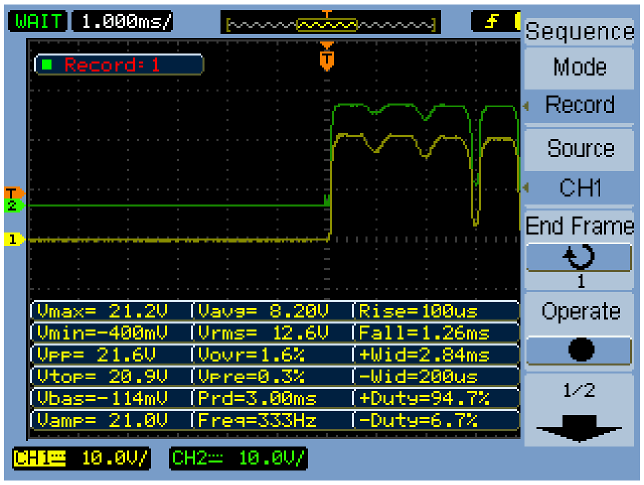
The amplitude of the voltage applied to the gate of the transistor, and the DC offset are vital to obtaining data from a solar panel. A DC offset was needed due to the transistor threshold voltage of 2 to 4 volts. The amplitude was initially varied from 2.5 Vpp to 4 Vpp using specific DC offsets to vary the voltage starting from zero. The 2.5 Vpp signal with a 1.25 Volt DC offset had well defined peaks (seen in Figure 9), and had a stable output. This was the standard setting used for amplitude and DC offset for since the beginning of experimentation. The voltages that were tested were, 2.5 Vpp amplitude with a 1.25 VDC offset, 3.0 Vpp amplitude with a 1.5 VDC offset, 3.5 Vpp amplitude with a 1.75 VDC offset, and a 4.0 Vpp amplitude with a 2.0 VDC offset. This experiment was performed using the first prototype configuration, using the DSO1002A Oscilloscope with the Agilent 33210A Function Generator.



```
Vmax= 21.2V    Vavg= 8.05V    Rise=120us
Vmin=-400mV    Vrms= 12.5V    Fall=2.10ms
Vpp= 21.6V     Vovr=3.4%      +Wid=2.48ms
Vtop= 20.5V    Vpre=1.4%      -Wid=200us
Vbas=-127mV    Prd=2.64ms     +Duty=93.9%
Vamp= 20.6V    Freq=379Hz     -Duty=7.6%
```

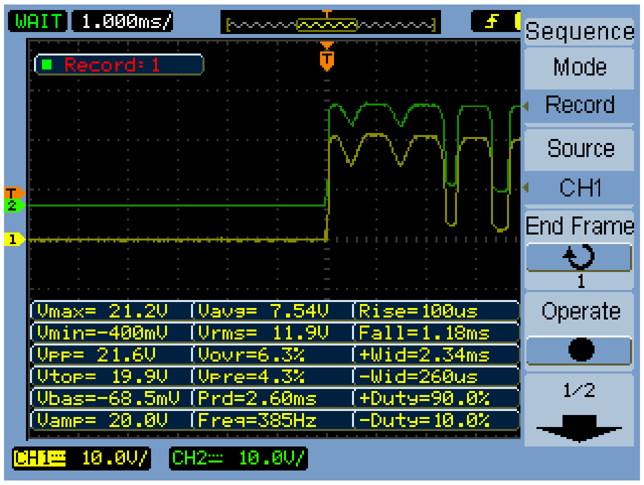
Figure 9. Voltage response using an amplitude of 2.5 and 1.25 VDC offset.

Figure 10 is the result of using 3.0Vp with a DC offset of 1.5 volts. The output voltage data from this portion of the experiment did not extend as low as the results in figure 9. In addition to this, the difference in voltage data acquired from probes 1 and 2 did not provide as much as a voltage difference, meaning that the current captured using these settings did not reach as far towards the short circuit current. Figure 11 reached deeper, but did not have a large difference in voltage between probes 1 and 2.



```
Vmax= 21.2V   Vavg= 8.20V   Rise=100us
Vmin=-400mV   Vrms= 12.6V   Fall=1.26ms
Vpp= 21.6V    Vovr=1.6%     +Wid=2.84ms
Vtop= 20.9V   Vpre=0.3%     -Wid=200us
Vbas=-114mV   Prd=3.00ms    +Duty=94.7%
Vamp= 21.0V   Freq=333Hz    -Duty=6.7%
CH1 10.0V/  CH2 10.0V/
```

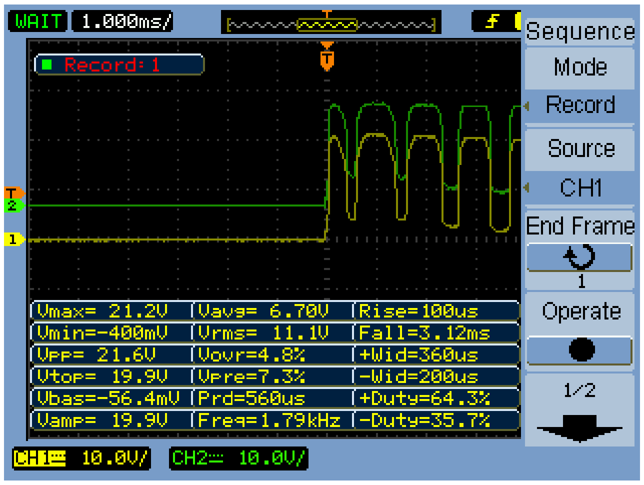
Figure 10. Voltage response using an amplitude of 3.0 Vp and 1.5 VDC offset.



```
Vmax= 21.2V   Vavg= 7.54V   Rise=100us
Vmin=-400mV   Vrms= 11.9V   Fall=1.18ms
Vpp= 21.6V    Vovr=6.3%     +Wid=2.34ms
Vtop= 19.9V   Vpre=4.3%     -Wid=260us
Vbas=-68.5mV  Prd=2.60ms    +Duty=90.0%
Vamp= 20.0V   Freq=385Hz    -Duty=10.0%
CH1 10.0V/  CH2 10.0V/
```

Figure 11. Voltage response using an amplitude of 3.5 Vp and 1.75 VDC offset.

Using 4 volts peak with a DC offset of 2 volts, the voltage applied to the gate was too high. This resulted in the voltages dropping too low, and the voltage sampled from across the solar panel was the same as the voltage across the drain.



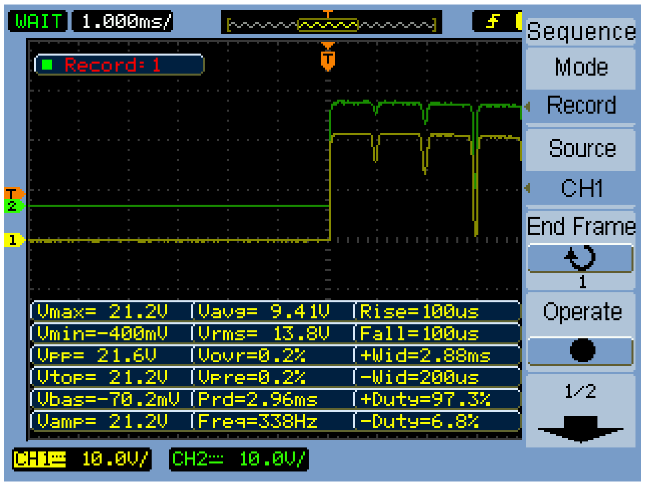
```
Vmax= 21.2V   Vavg= 6.70V   Rise=100us
Vmin=-400mV   Vrms= 11.1V   Fall=3.12ms
Vpp= 21.6V    Vovr=4.8%     +Wid=360us
Vtop= 19.9V   Vpre=7.3%     -Wid=200us
Vbas=-56.4mV  Prd=560us     +Duty=64.3%
Vamp= 19.9V   Freq=1.79kHz  -Duty=35.7%
CH1= 10.0V/   CH2= 10.0V/
```

Figure 12. Voltage response using an amplitude of 4.0 Vp and 2.0 VDC offset.

C. Different Waveforms

The Agilent 33210A Function Generator has many functions to choose from when trying to make a voltage signal. The sinusoidal, rectangular and triangular waveforms were tested to determine which type would produce the most meaningful results. The triangular waveform was performed twice. Once with the symmetry at 100% symmetry, and one at 50% symmetry. For each test, the function generator was set to output a 2.5 Vpp signal with a DC offset of 1.25 volts while set at a frequency of 1 kHz.

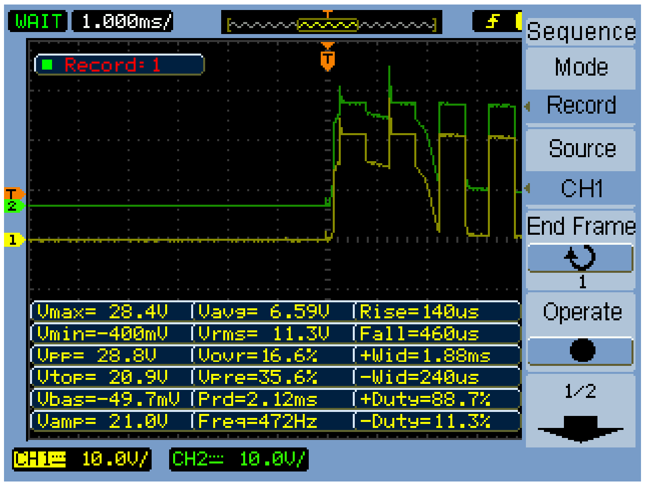
Using a sinusoidal wave produced results where there was a definite voltage difference. The width of the voltage spike was also clearly defined. The results are shown in figure 13.



```
Vmax= 21.2V   Vavg= 9.41V   Rise=100us
Vmin=-400mV   Vrms= 13.8V   Fall=100us
Vpp= 21.6V    Vovr=0.2%     +Wid=2.88ms
Vtop= 21.2V   Vpre=0.2%     -Wid=200us
Vbas=-70.2mV  Prd=2.96ms    +Duty=97.3%
Vamp= 21.2V   Freq=338Hz    -Duty=6.8%
```

Figure 13. Voltage response for a sinusoidal waveform.

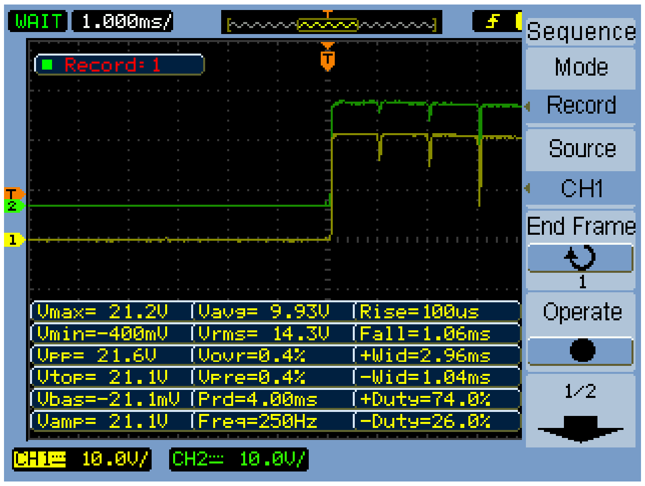
The square waveform in figure 14 produced useless results. There were no voltage differences captured during this part of the experiment.



```
Vmax= 28.4V   Vavg= 6.59V   Rise=140us
Vmin=-400mV   Vrms= 11.3V   Fall=460us
Vpp= 28.8V    Vovr=16.6%    +Wid=1.88ms
Vtop= 20.9V   Vpre=35.6%    -Wid=240us
Vbas=-49.7mV  Prd=2.12ms    +Duty=88.7%
Vamp= 21.0V   Freq=472Hz    -Duty=11.3%
```

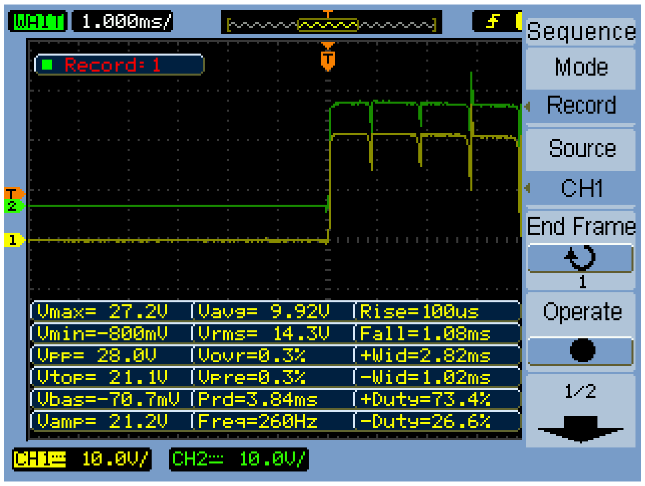
Figure 14. Voltage response for a square waveform.

The triangular waveform set to 50% symmetry in figure 15 had definite voltage peaks, had a small width compared to the sinusoidal wave, meaning that the current captured using a triangular waveform obtains a smaller amount of data and a smaller current than using a sinusoidal wave. With symmetry set to 100% for a triangular wave, the resulting voltage spikes were even more thin.



```
Vmax= 21.2V   Vavg= 9.93V   Rise=100us
Vmin=-400mV   Vrms= 14.3V   Fall=1.06ms
Vpp= 21.6V    Vovr=0.4%     +Wid=2.96ms
Vtop= 21.1V   Vpre=0.4%     -Wid=1.04ms
Vbas=-21.1mV  Prd=4.00ms    +Duty=74.0%
Vamp= 21.1V   Freq=250Hz    -Duty=26.0%
CH1 10.0V/  CH2 10.0V/
```

Figure 15. Voltage response for a triangular waveform (50% symmetry).

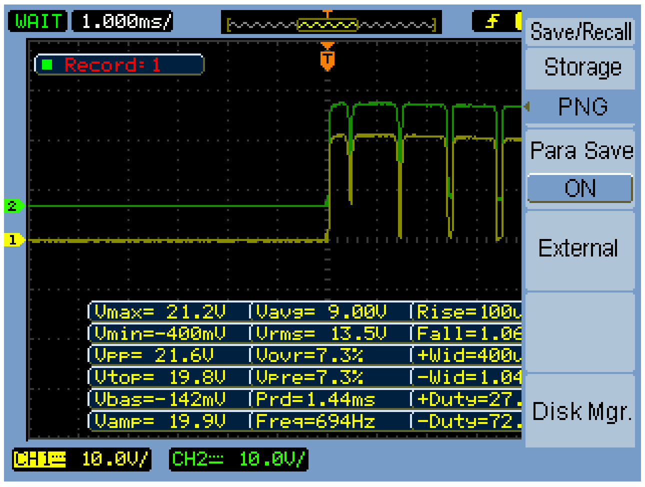


```
Vmax= 27.2V   Vavg= 9.92V   Rise=100us
Vmin=-800mV   Vrms= 14.3V   Fall=1.08ms
Vpp= 28.0V    Vovr=0.3%     +Wid=2.82ms
Vtop= 21.1V   Vpre=0.3%     -Wid=1.02ms
Vbas=-70.7mV  Prd=3.84ms    +Duty=73.4%
Vamp= 21.2V   Freq=260Hz    -Duty=26.6%
CH1 10.0V/  CH2 10.0V/
```

Figure 16. Voltage response for a triangular waveform (100% symmetry).

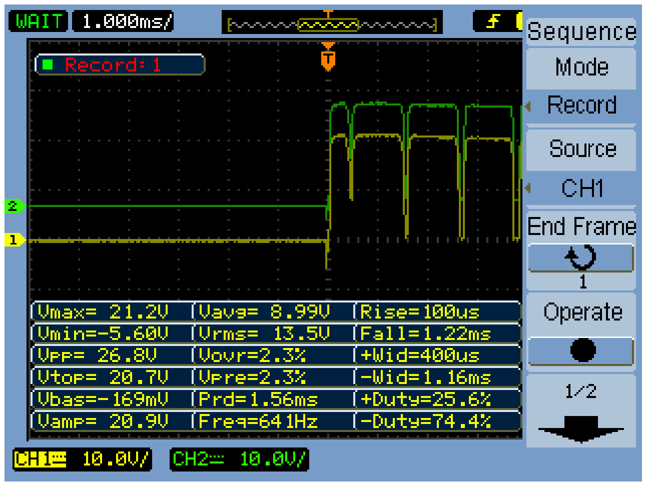
D. Different Frequencies

When acquiring data for this particular system, it is necessary to collect as many data points as possible. The reason being that the data acquired by the hand held device will be much less than can be acquired by an oscilloscope due to the high sample rate of the oscilloscopes used. Smaller frequencies have a longer period,which allows for more data to be acquired. When determining the smallest frequency that could be used for acquisition, the 2 ms time for a stable open-circuit voltage was used, which yielded a frequency of 500 Hz. For each test, the function generator was set to output a 2.5 Vpp signal with a DC offset of 1.25 volts using a sinusoidal waveform.



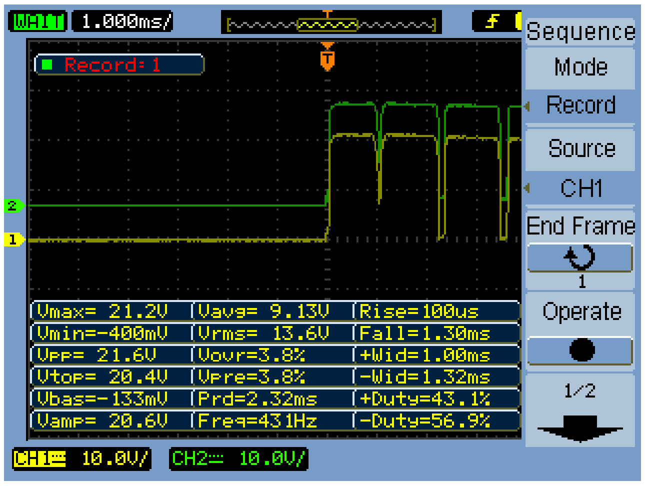
```
Vmax= 21.2V    Vavg= 9.00V    Rise=100u
Vmin=-400mV    Vrms= 13.5V    Fall=1.06
Vpp= 21.6V     Vovr=7.3%      +Wid=400u
Vtop= 19.8V    Vpre=7.3%      -Wid=1.04
Vbas=-142mV    Prd=1.44ms     +Duty=27.
Vamp= 19.9V    Freq=694Hz     -Duty=72.
```

Figure 17. Voltage output with 1 kHz signal.



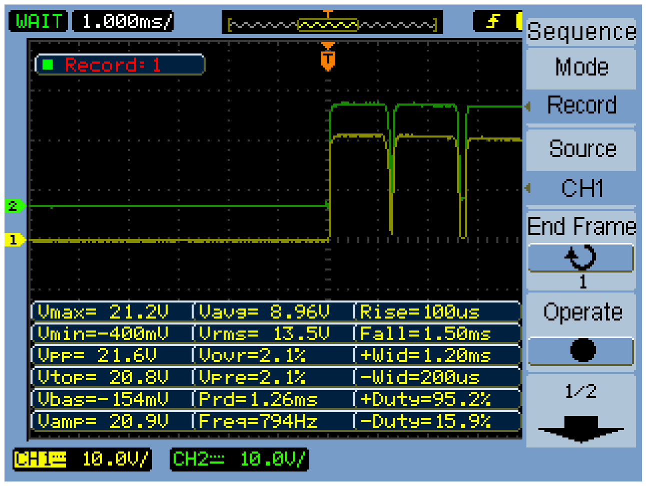
```
Vmax= 21.2V   Vavg= 8.99V   Rise=100us
Vmin=-5.60V   Vrms= 13.5V   Fall=1.22ms
Vpp= 26.8V    Vovr=2.3%     +Wid=400us
Vtop= 20.7V   Vpre=2.3%     -Wid=1.16ms
Vbas=-169mV   Prd=1.56ms    +Duty=25.6%
Vamp= 20.9V   Freq=641Hz    -Duty=74.4%
```

Figure 18. Voltage output with 900 Hz signal.



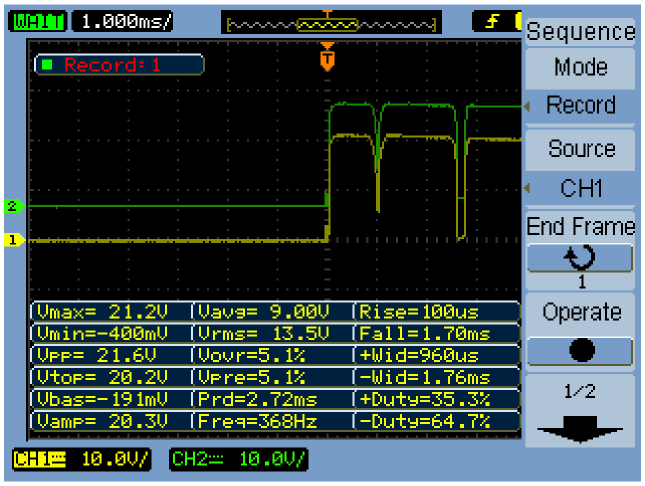
```
Vmax= 21.2V   Vavg= 9.13V   Rise=100us
Vmin=-400mV   Vrms= 13.6V   Fall=1.30ms
Vpp= 21.6V    Vovr=3.8%     +Wid=1.00ms
Vtop= 20.4V   Vpre=3.8%     -Wid=1.32ms
Vbas=-133mV   Prd=2.32ms    +Duty=43.1%
Vamp= 20.6V   Freq=431Hz    -Duty=56.9%
```

Figure 19. Voltage output with 800 Hz signal.



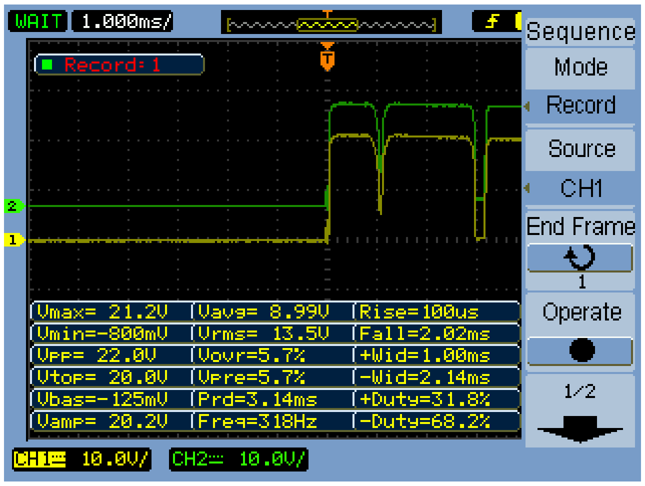
```
Vmax= 21.2V   Vavg= 8.96V   Rise=100us
Vmin=-400mV   Vrms= 13.5V   Fall=1.50ms
Vpp= 21.6V    Vovr=2.1%     +Wid=1.20ms
Vtop= 20.8V   Vpre=2.1%     -Wid=200us
Vbas=-154mV   Prd=1.26ms    +Duty=95.2%
Vamp= 20.9V   Freq=794Hz    -Duty=15.9%
```

Figure 20. Voltage output with 700 Hz signal.



```
Vmax= 21.2V   Vavg= 9.00V   Rise=100us
Vmin=-400mV   Vrms= 13.5V   Fall=1.70ms
Vpp= 21.6V    Vovr=5.1%     +Wid=960us
Vtop= 20.2V   Vpre=5.1%     -Wid=1.76ms
Vbas=-191mV   Prd=2.72ms    +Duty=35.3%
Vamp= 20.3V   Freq=368Hz    -Duty=64.7%
```

Figure 21. Voltage output with 600 Hz signal.

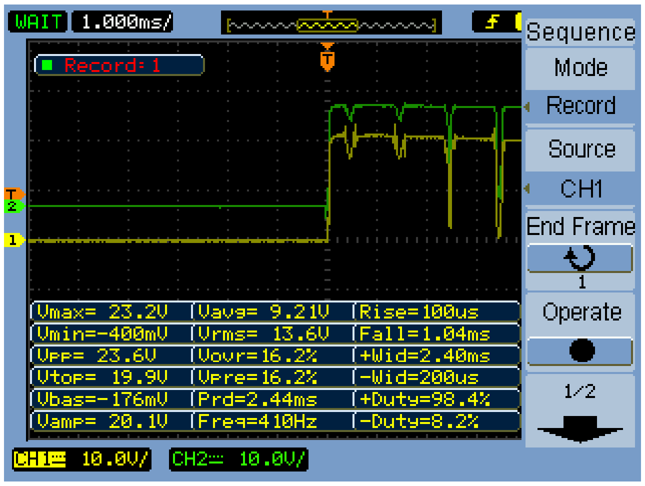


```
Vmax= 21.2V   Vavg= 8.99V   Rise=100us
Vmin=-800mV   Vrms= 13.5V   Fall=2.02ms
Vpp= 22.0V    Vovr=5.7%     +Wid=1.00ms
Vtop= 20.0V   Vpre=5.7%     -Wid=2.14ms
Vbas=-125mV   Prd=3.14ms    +Duty=31.8%
Vamp= 20.2V   Freq=318Hz    -Duty=68.2%
```

Figure 22. Voltage output with 500 Hz signal.

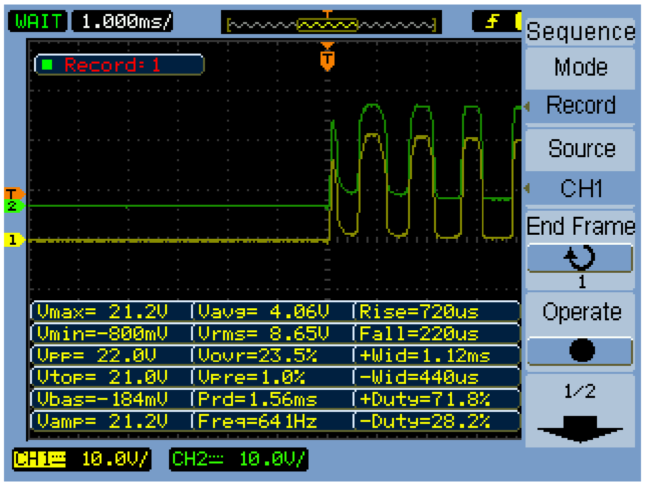
E. Different Transistors

When beginning the data acquisition project, an IRF530N transistor was used. Multiple transistors were tested to determine if other readily available transistors could provide a better response in the system. The IRF530N was tested against a TIP31 and BS170 transistor. For each test, the function generator was set to output a 2.5 Vpp signal with a DC offset of 1.25 volts using a sinusoidal waveform at a frequency of 1 kHz.



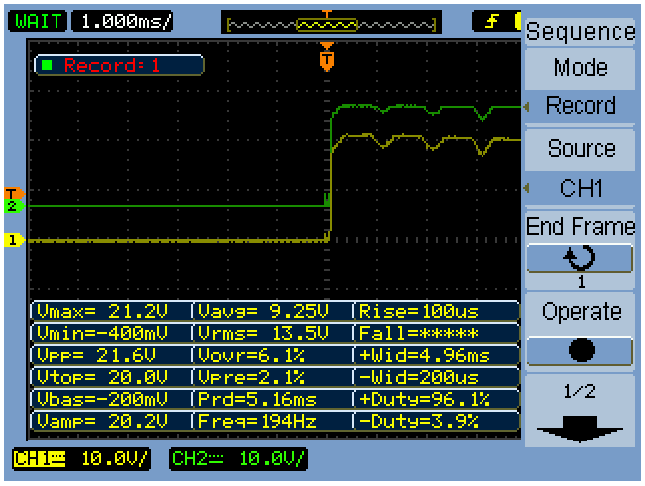
```
Vmax= 23.2V   Vavg= 9.21V   Rise=100us
Vmin=-400mV   Vrms= 13.6V   Fall=1.04ms
Vpp= 23.6V    Vovr=16.2%    +Wid=2.40ms
Vtop= 19.9V   Vpre=16.2%    -Wid=200us
Vbas=-176mV   Prd=2.44ms    +Duty=98.4%
Vamp= 20.1V   Freq=410Hz    -Duty=8.2%
```

Figure 23. Voltage output using an IRF530N Transistor.



```
Vmax= 21.2V   Vavg= 4.06V   Rise=720us
Vmin=-800mV   Vrms= 8.65V   Fall=220us
Vpp= 22.0V    Vovr=23.5%    +Wid=1.12ms
Vtop= 21.0V   Vpre=1.0%     -Wid=440us
Vbas=-184mV   Prd=1.56ms    +Duty=71.8%
Vamp= 21.2V   Freq=641Hz    -Duty=28.2%
CH1= 10.0V/  CH2= 10.0V/
```

Figure 24. Voltage output using a TIP31 Transistor.



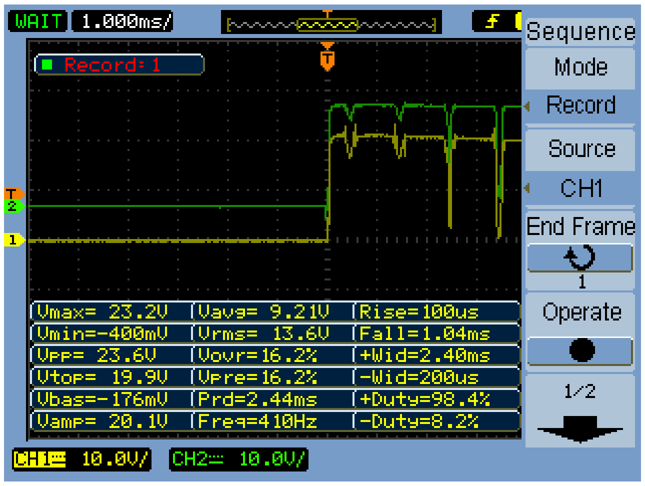
```
Vmax= 21.2V   Vavg= 9.25V   Rise=100us
Vmin=-400mV   Vrms= 13.5V   Fall=*****
Vpp= 21.6V    Vovr=6.1%     +Wid=4.96ms
Vtop= 20.0V   Vpre=2.1%     -Wid=200us
Vbas=-200mV   Prd=5.16ms    +Duty=96.1%
Vamp= 20.2V   Freq=194Hz    -Duty=3.9%
CH1= 10.0V/  CH2= 10.0V/
```

Figure 25. Voltage output using a BS170 Transistor.

The IRF530N was the best transistor to use compared to the BS170 and TIP31. The TIP31 transistor reached saturation very quickly compared to the IRF530N, and did not have a large active region. The BS170 looked promising when examining the oscilloscope graph, but after examining the raw voltage data, it was determined that the resulting current output was significantly lower than with the IRF530N.

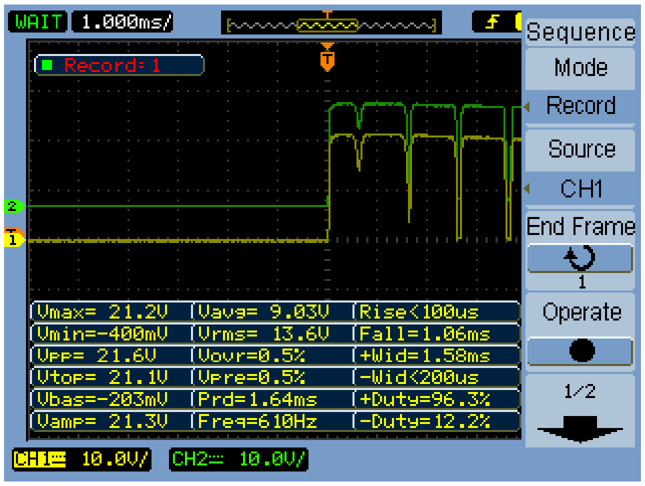
F. Shielding

After changing the load resistance of the circuit, it was apparent that a large amount of noise was present in the circuit. To minimize the noise in the circuit, a metal enclosure was used.



```
Vmax= 23.2V   Vavg= 9.21V   Rise=100us
Vmin=-400mV   Vrms= 13.6V   Fall=1.04ms
Vpp= 23.6V    Vovr=16.2%    +Wid=2.40ms
Vtop= 19.9V   Vpre=16.2%    -Wid=200us
Vbas=-176mV   Prd=2.44ms    +Duty=98.4%
Vamp= 20.1V   Freq=410Hz    -Duty=8.2%
```

Figure 26. Voltage output without a metal enclosure.



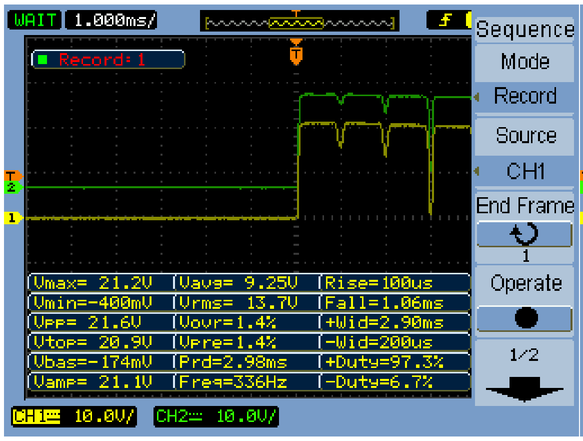
```
Vmax= 21.2V   Vavg= 9.03V   Rise<100us
Vmin=-400mV   Vrms= 13.6V   Fall=1.06ms
Vpp= 21.6V    Vovr=0.5%     +Wid=1.58ms
Vtop= 21.1V   Vpre=0.5%     -Wid<200us
Vbas=-203mV   Prd=1.64ms    +Duty=96.3%
Vamp= 21.3V   Freq=610Hz    -Duty=12.2%
```

Figure 27. Voltage output with a metal enclosure.

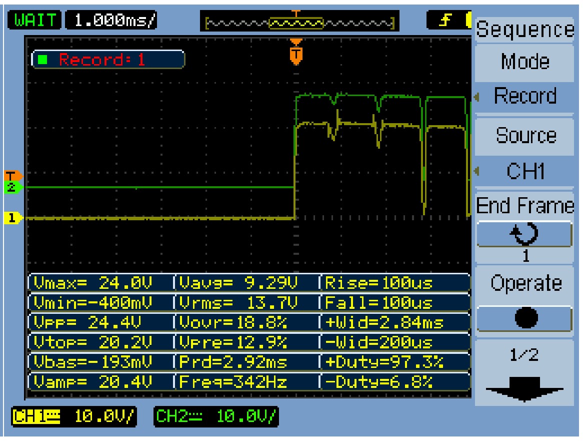
PROBLEMS ENCOUNTERED & POSSIBLE SOLUTIONS

A. Load Resistor

To obtain more data closer to the short-circuit current, a new load resistor was purchased The new resistor met the original power dissipation requirements, dissipating 300 Watts of power with a resistance of 1 ohm. The two 5 ohm resistors were 3” x 3”, which filled a 6” x 3” space. With the additional 0.3” needed to make connections, this resistor configuration required a significant amount of space. The new 1 ohm resistor was a bar cylindrical, 9” long tube with a diameter of 1.2”. Using this new resistor could have allowed for a smaller form factor to be used. Unfortunately, the change resulted in the acquisition circuit being more sensitive to noise with almost no gain in additional data approaching the short circuit current. After proper shielding was provided, there was little difference between the 2.5 ohm and 1 ohm load results. The results of testing both resistors can be seen in Figure 29.



```
Vmax= 21.2V    Vavg= 9.25V   Rise=100us
Vmin=-400mV    Vrms= 13.7V   Fall=1.06ms
Vpp= 21.6V     Vovr=1.4%     +Wid=2.90ms
Vtop= 20.9V    Vpre=1.4%     -Wid=200us
Vbas=-174mV    Prd=2.98ms    +Duty=97.3%
Vamp= 21.1V    Freq=336Hz    -Duty=6.7%
```



```
Vmax= 24.0V    Vavg= 9.29V   Rise=100us
Vmin=-400mV    Vrms= 13.7V   Fall=100us
Vpp= 24.4V     Vovr=18.8%    +Wid=2.84ms
Vtop= 20.2V    Vpre=12.9%    -Wid=200us
Vbas=-193mV    Prd=2.92ms    +Duty=97.3%
Vamp= 20.4V    Freq=342Hz    -Duty=6.8%
```

Figure 28. Voltage output with a 2.5 ohm load (Left) and 1 ohm load (Right).

B. Shielding

During experimentation, a lot of results gathered contained a 0.4 volt noise when operating at 1 kHz. This noise was a significant factor when using the 1 ohm resistor to obtain data points closer to the short-circuit current. When the 1 ohm resistor was first placed into the circuit and tested, strange voltage spikes were very apparent as different voltages were applied to the transistor gate due to switching such a large capacitance. To remedy this situation, a metal enclosure was tested around the first prototype. Since the electrical circuits were exposed without any shielding, the circuit was acting like an antenna, which allowed many sources of noise to penetrate the circuit and caused undesired results. For the final prototype, the enclosure was fitted with aluminum flashing to shield the electrical circuitry from any external electrical interference.

C. Choosing a Transistor

To vary the voltage of the circuit, a transistor needed to be used. The quality of the IRF530N transistor was very consistent and produced good data, but using a better transistor could yield better results. After searching for transistors that could potentially be used for this project, there were several transistors to pick from. After carefully looking at each transistor, three were used for further testing to see which transistor produced the best results. Standard MOSFET transistors were compared to determine the best choice was used for all of the remaining experiments.

D. Simultaneous Data Acquisition

When acquiring data from a solar panel using an Arduino, it is important to understand that each data point acquired in serial by the Arduino microcontroller. This is the case even when using multiple analog ports because the ATMEGA328P IC only contains one analog to digital converter (ADC).[3] This hardware limitation creates a need for another microcontroller; one for acquiring voltage data from the solar panel, and one acquiring data from the drain pin on the IRF530N transistor. This is due to the fact that data must be gathered at the same time to be useful considering these points are used to find the current output from the solar panel at a particular instance in time, which means both sets of data must be acquired in the same time frame.

To accomplish this, a master and slave relationship was established between two Arduino microcontrollers as seen in figure 29.

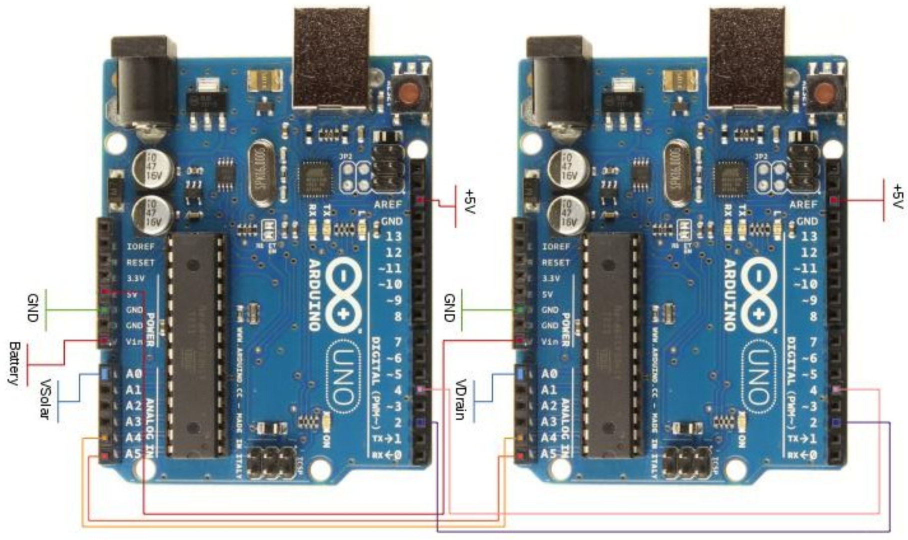


Figure 29. Two Arduinos using the I2C protocol for data transmission.

For each microcontroller, two digital pins, and two analog pins were used to communicate between the master and slave. The digital pins were used to communicate between the master and slave; indicating each when the other was done with a task. To start, the master signals the slave, using a digital pin, to trigger the solar simulator and acquire voltages from the drain of the IRF530N transistor. After the master has signaled the slave, the master is delayed for a short period, then acquires voltages across the solar panel. The delay is to synchronize the time that the voltages are both acquired from the master and slave. The slave then signals the master, using a digital pin, that the voltage acquisition is done. The master then uses the two analog pins (A4 and A5) to transfer the drain voltage data from the slave to the master. To transfer the voltage values from the slave to the master, the I2C communication protocol was used via the provided Wire library.[13] Serial Communication was not used in this case because of the potential interference with the python program, as the python program will accept any serial output (intended or unintended) from the Arduino.

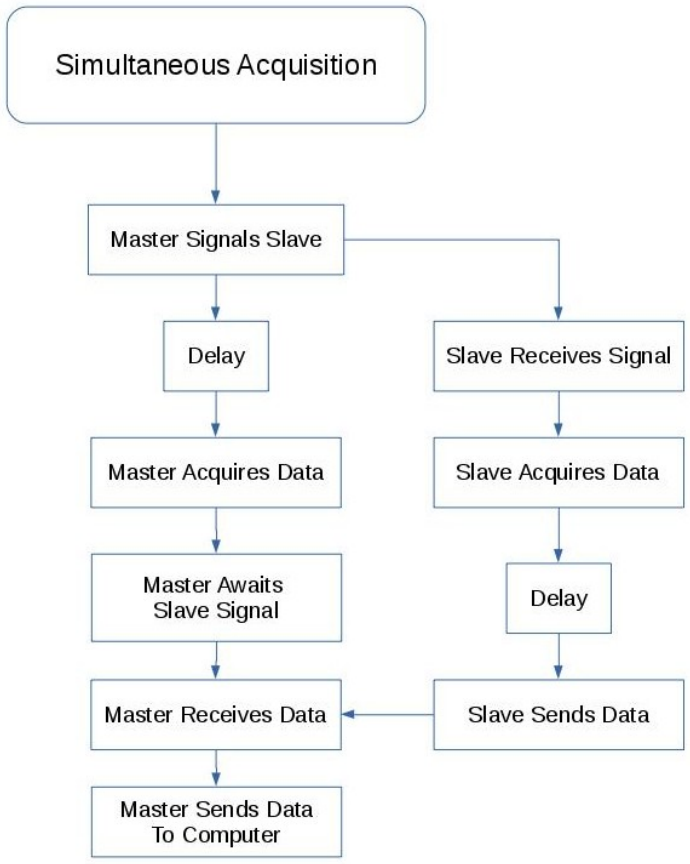


Figure 30. Flowchart for acquisition with a master-slave configuration with Arduinos.

It is vital that the master and slave be synchronized when acquiring data and sending data, since issues can arise from an unsynchronized configuration. There were issues in the development stage where the master was directing the slave to send data values before the slave was ready, which caused the slave to lock-up and become unresponsive. In an attempt fix the issue, communication between the master and slave via the digital pins was used. After merging this code difference, the master would initially signal the slave to start the solar simulator process. The slave would trigger the solar simulator, and start collecting data. At the same time, the master would wait 10 micro-seconds, then acquire data. A 10 micro-second delay was able to supplement the time used for the slave to receive the signal from the master, and trigger the solar simulator. The slave would then use the a digital port to signal to the master that the data collection as finished. The master would receive this signal from the slave, and request the data that was acquired from the slave.

This eliminated several problems that were present when using the wire library to send signals to start processes. Initially when the master would send a signal to start acquisition, the slave would start the acquisition, then suddenly stop as any request from the wire library after that point would lock-up the slave. Using digital pins that are programmed to be exclusively and input or output simplified this process. Unfortunately, issues were still present when trying to obtain information using the wire library. When consulting the documentation for the Atmega320P, there was a section under revision D of the IC which stated there was an issue where using the Atmega328P with a two wire interface (TWI) slave, using a clock frequency over 2 MHz, the data setup time was too short resulting in a false start or stop condition.[3] This issue was remedied by adding a time delay between the digital communication between the slave and master, where the slave was signaling the master that the slave was ready to send the acquired voltage data.

E. Fast Acquisition Coding

Due to various tasks that are ran in the background and default settings built into the analogRead() function provided by the Arduino software, acquisition time for the ATmega328P is not optimized for the fastest possible acquisition from analog ports. To combat this issue, assembly code was embedded into the program code to shorten the acquisition time. The following code provides the fast acquisition:

```
/*-Fast ADC Code---------------------------------------*/
#define FASTADC 1

// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

/*-Fast ADC settings-----------------------------*/
#if FASTADC
  // set prescale to 16
  sbi(ADCSRA,ADPS2) ;
  cbi(ADCSRA,ADPS1) ;
  cbi(ADCSRA,ADPS0) ;
  #endif
```

By setting the ADPS2, and clearing the ADPS1 and ADPS0 bits, the amount of time the ADC takes to approximate the voltage is reduced, and therefore lessens the amount of time for each conversion. This method of speeding up the acquisition has the possible drawback of having a slightly less precise reading, but after many tests between the two methods, there did not appear to be any significant change in precision. A chart illustrating the internal bits of the ADC within an ATMEGA328P IC can be found in APPENDIX G.[14]

FINAL DESIGN, TESTING, & VERIFICATION

A. Signal Generator

The first prototype of the data acquisition device had the advantage of using a Function Generator, which could easily be used to generate a signal with the appropriate amplitude, frequency and DC offset. Due to the necessity of having the device be portable, this Function Generator needed to be replaced with a circuit that could generate a similar signal to the one that was produced by the Function Generator. After searching through many different options, a circuit board based on the AD9850. The AD9850 is a chip based on direct digital synthesizer (DDS) technology.

There were numerous reasons that this particular option was selected for use. The most important reason being that the device was fabricated with the AD9850 installed with male header pins, which was important because the AD9850 came as a surface mount device, and would have been nearly impossible to solder due to the fact that the device pins were microns apart. The device also had freely available libraries available, which were Arduino compatible, which minimized the amount of time needed to properly gain an understanding of how the AD9850 functioned, and how to integrate the chip into the hand held device circuit as well as allow focus to remain on the overall device. Since the AD9850 chip could be programmed using an Arduino, this allowed for the frequency to easily be changed as necessary if desired. The cost of purchasing the device, rather than the chip itself, was much cheaper. By doing a search at mouser.com, the chip costs about twenty-seven dollars compared to about eight dollars for the fully finished device on ebay.com. The device also had the advantage of having low power consumption, which would maximize the life of the batteries in the hand held device.

The only disadvantage to using this device was that the unaltered signal from the AD9850 circuit board is that the voltage signal output is 1 Volt peak-to-peak. The transistor used requires a minimum of 2 Volts applied to the gate to allow electron flow. For the transistor to turn fully on, the voltage needed to be at least 5 Volts to turn the transistor completely on. To remedy this problem, an amplification circuit needed to be created to increase the voltage from 1 Volt to 5 Volts. The LM741 op-amp was used due to it being readily available locally in the case of the IC being damaged. An inverting amplifier configuration was used due to the simplicity of the circuit. Using the equation $A_V = -R_F / R_1$, where A_V is the gain of the input voltage signal to the inverting input.[4] Since readily available resistors were typically within a tolerance of 5%, it was decided that a resistor and potentiometer would be used together to eliminate this problem with the additional benefit of being able to increase and decrease the voltage signal easily. Using a 3.3 kohm resistor with a 10 kohm potentiometer, it was determined that the peak to peak voltage could be varied from 3.3 Volts to 8.3 Volts, which meets the minimum voltage requirement for allowing the transistor to operate in the active region, and saturation region.

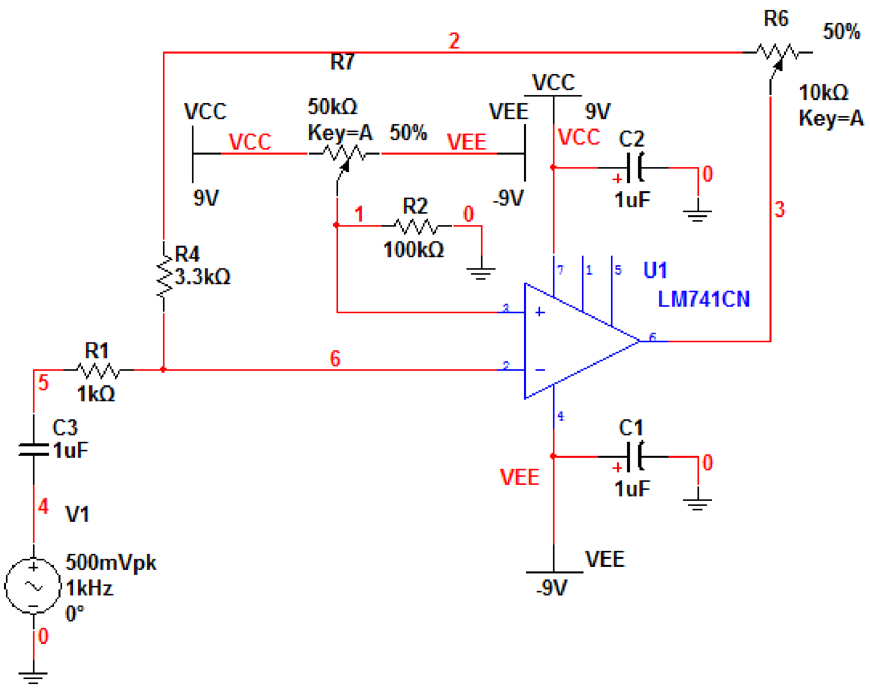


Figure 31. Schematic for signal amplification circuit.

Since the voltage signal applied to the transistor gate needed to vary between 0 and 5 Volts to properly capture data points, a DC offset was needed to allow the voltage from the AD9850 to vary between these voltages. To accomplish this, the voltage supply was connected to a potentiometer to change the voltage applied to the non-inverting input. By changing the voltage applied to the non-inverting input, the reference point of the output signal was altered, changing the DC offset. The final constructed form of this circuit can be seen in Figure 32.

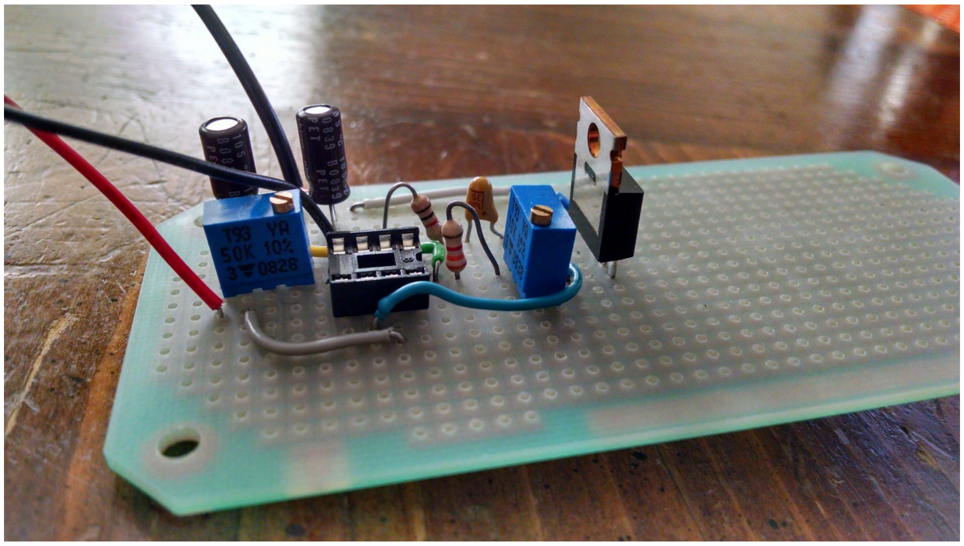


Figure 32. Completed amplification circuit in solderboard.

B. Batteries

Due to the power needs of the Arduino, LM741, and AD9850, it was decided that the hand held device would use two 9 Volt batteries. In addition to these considerations, using 9 Volt batteries were the most compact option compared to using AA or AAA batteries, which would have resulted in the need of a bulky battery pack. The LM741 op-amp requires a split phase supply, which means that a positive and negative voltage supply (typically of the same value) was needed to allow the LM741 to operate. To create a split phase supply, the two 9 Volt batteries were put in series and the center connection was connected to ground, giving the circuit a + 9 Volt, and a – 9 volt supply.

C. Analog Inputs

To replace the use of the oscilloscope, the Arduino Uno was used. The problem with any microcontroller used for this application is the input voltage being too high for the IO ports. Depending on the microcontroller, the maximum allowed voltage input could be as low as 1.7 Volts. In this case, the maximum allowed voltage to be applied to the analog input for an Arduino Uno is 5 Volts. For this application, the maximum voltage output from solar panels could be as high as 36 volts for a 250 Watt solar panel. To remedy this situation, a voltage divider circuit was introduced. Using a maximum value of 40 Volts output from a solar panel, it was determined that the Voltage applied to the analog inputs on the data acquisition device needed to be 1/8 of the original Voltage. To accomplish this, a 51 kohm resistor and a 10 kohm potentiometer were used. To calculate the exact resistance needed in place of the 10 ohm resistor, the formula, $V_{OUT}=\frac{R_2}{R_1+R_2}V_{Source}$, was manipulated to become $R_2=\frac{V_{OUT}\cdot R_1}{V_{Source}-V_{OUT}}$.[7] The resistance was calculated to be 7.29 kohms.[4] Using this base resistance value for the 10 kohm potentiometer, and manually adjusting the circuit until the voltage input to the analog ports, the circuit was properly calibrated to accurately provide the proper voltage values.

D. Microcontroller

The Arduino Uno microcontroller was used for the second prototype. The Beagle Bone Black microcontroller was originally going to be used because it has a 1 GHz processor, multiple Digital IO ports, and built in analog ports, which would be able to replace the Arduino Uno easily without many modifications to the various circuits within this project. Unfortunately, the microcontroller never arrived due to a hardware defect, which caused the shipment to be back-ordered more than six months. If the Beagle Bone Black would have been used, then all of the data analysis could have been done directly from the hand held device because the Beagle Bone Black uses the Linux operating system.

Despite the significant drawback of having a lower sample rate, there were some benefits to using the Arduino Uno. The libraries used for programming the AD9850 circuit board was already written with the intention of being used with Arduino microcontrollers. This had the benefit of not needing to re-write any code to be retrofitted to the formats of a different microcontroller. The AD9850 also required a 3.3 Volt or 5 Volt source, which was supplied by the regulated 5 Volt supply from the Arduino Uno.

E. Electrical Circuitry

For the final prototype, all circuitry regarding the main solar simulator acquisition circuit, and most of the signal generator circuit were soldered to a solder board to minimize the possible points of failure within the circuit, and minimizing the distance between connections. The only connections that were not directly soldered were connections to the Arduino Uno. Besides the analog inputs, all other inputs and outputs were not directly related to the main data acquisition circuitry.

It is important to note that the AD9850 circuit board is VERY sensitive to heat when being soldered. When attempting to solder the AD9850 circuit board into the solder board, it was damaged and was beyond the hope repair. To prevent damage to the AD9850, two 10 pin female header strips were used to solder into the solder board instead of directly soldering the AD9850 circuit board. This solution worked perfectly, and allowed the circuit board to be easily replaced if another hardware failure occurred.

To allow for users to see a visual confirmation that the program had successfully ran and completed, an LED was installed to display when the device had been triggered, and when the device is ready to be triggered again. To shrink the amount of space used for connecting the LED, the 220 ohm resistor used to limit the current flow was soldered directly to the LED. To prevent a short circuit from occurring within the circuitry, heat shrink tubing was used. The heat shrink tubing was able to cover the leads of the LED, the resistor, and all of the exposed wire. A secondary LED was also installed for indicating the state of the device (on or off).

To activate the device, a push button needed to be used. It is important to note that any digital inputs to the Arduino Uno that are used should be connected to ground when not in use. The reason behind this is that the state of the input when ungrounded can possibly be in a state between 5 Volts and ground. This can be problematic since the Arduino may interpret this non-zero signal as a 5 Volt signal. If this happens, then the equipment being controlled, in this case the solar simulator, could trigger randomly. To prevent this from occurring, a 10 kohm pull-down resistor was used to make the voltage at the digital input pin drop to zero when the push button is not pressed. Using this method, it is impossible for the voltage applied to the digital input would be between states.

The ability to adjust the amplitude and DC offset of the sinusoidal signal applied to the transistor gate, for acquiring the voltages, was added to allow more control over the acquisition process in addition to precise adjustments to the voltage output. The circuit depicted in Figure 8 shows the two potentiometers responsible for these settings. The schematic shown in Figure 8 shows R6 as the potentiometer responsible for the voltage amplitude, and R7 as the DC offset voltage.

F. Packaging

To select a properly sized enclosure to store the electrical circuits and load resistor, the dimensions of each circuit needed to be examined. The dimensions of the main circuitry is 4.5” x 3”. Knowing the circuit board and Arduino Uno dimensions, all that remained was the space needed to house the resistor. After quickly researching project boxes, the Radioshack 8” x 6” x 3” project box was selected. After doing simple math, it was calculated that the resistor would have 10” diagonally within the box. After placing the resistor within the box, it was determined that the project box was closer to 9” since the project boxes start from an 8” x 6” base, but narrows as the end of the project box is reached in addition to different shielding and screw holes within the project box, which made the usable area smaller. The 1 ohm resistor was attached to the top of the project box because it was required to be diagonal to allow it to fit in the enclosure. The main circuitry and Arduino Uno were bolted to the bottom of the project enclosure on the provided piece of sheet metal. The sheet metal was also connected to the aluminum flashing inside the project box to make a single shielding around the circuitry. The USB port on the Arduino Uno is exposed through a hole in the enclosure to provide easy access for programming after assembly. The final product before assembly is shown in Figure 33.

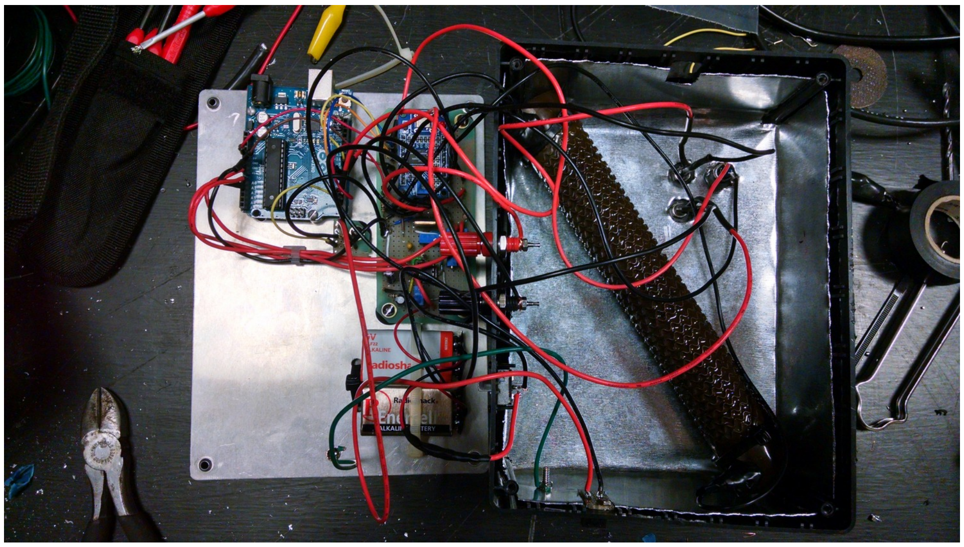


Figure 33. Final prototype before complete packaging.

Due to the small amount of space used for the circuitry and microcontroller, it is possible to easily change microcontrollers in the future for further development.

G. Final Testing

The final design features a 8 x 6 x 4 project box to hold all the components needed for a hand held device. For each test, the DDS chip was set to output a 2 Vpp signal with a DC offset of 4 volts using a sinusoidal waveform at a frequency of 1 kHz. The data collected from the solar simulator was then compared to a simulation designed using Multisim 11. To model the solar panel and flash, a capacitor and pulse input were used.

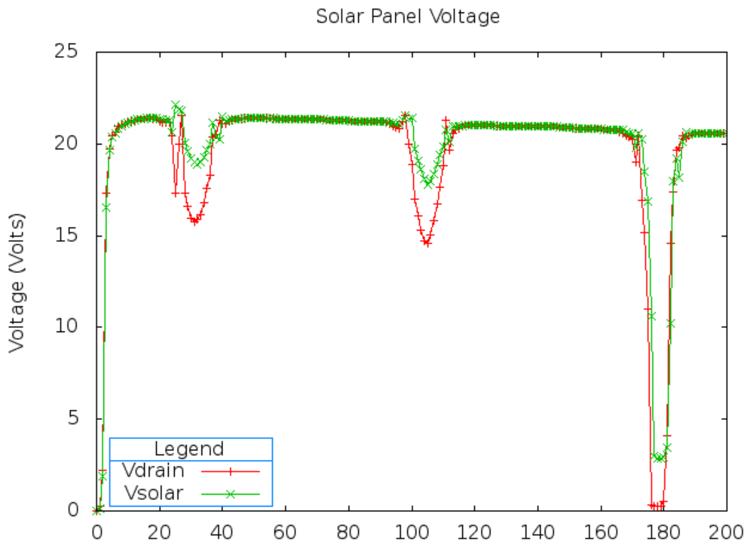


Figure 34. Data captured from hand held device.

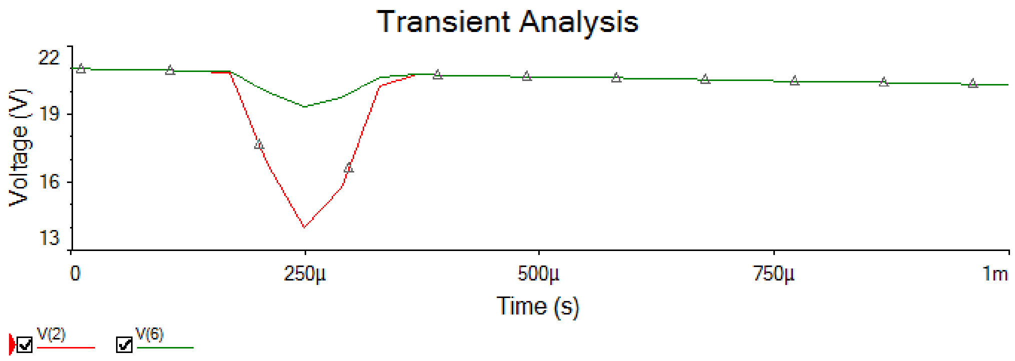


Figure 35. Voltage output from multisim 11 simulation.

The results of the hand held device were then tested against the previous designs as well as the manufacturers IV curve data. The manufacturers IV curve data was taken from the Solarex^{TM} manual for the 30 watt solar panel. The DSO1002A and DSOX3024A data was taken using the first design, but using two different oscilloscopes.

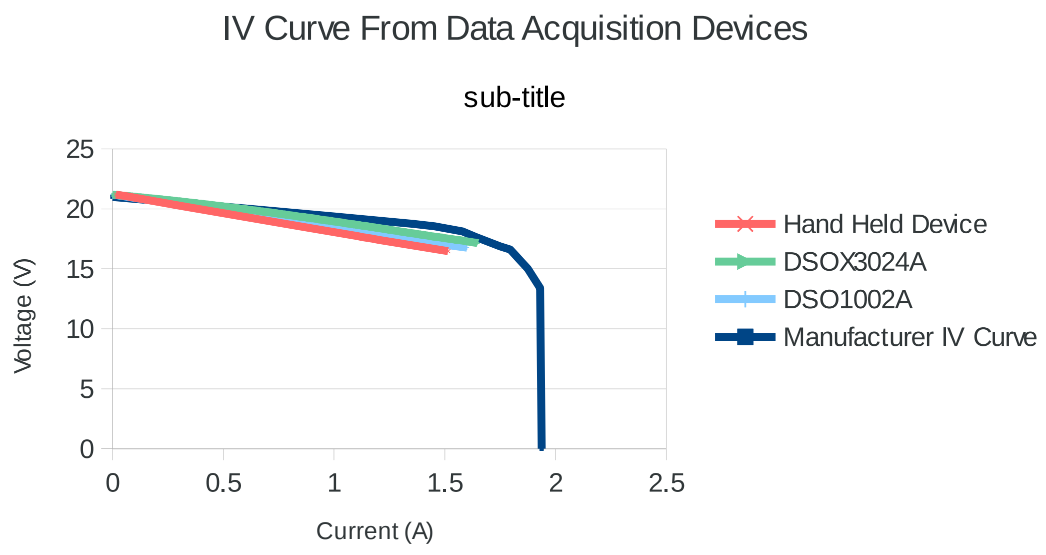


Figure 36. IV Curve from generated from data from hand held device.

RESULTS AND DISCUSSION

The data acquisition device has been designed and optimized. The initial design tested fundamental characteristics of this approach to data acquisition, namely, the signal to drive the transistor. It was determined that the maximum voltage that should be used to drive the IRF530N transistor is 5 volts. This was determined after analyzing the data from the experiment in Chapter VI – Section B. The results appeared to be sufficient for an amplitude of 3.5 volts with a DC offset of 1.75 volts, however, there was no great benefit in using this voltage compared to the results made with an amplitude of 2.5 volts with a DC offset of 1.25 volts. Due to the limitation of power consumption for the hand held device, it was determined that the lowest voltage possible without sacrificing any significant data.

The frequency used to drive the transistor gate was tested to determine the best frequency to use for acquisition. After carefully examining the data, it was determined that the frequency did not make a significant difference. The 1 kHz signal was used for the remainder of the experiments.

It was determined that waveform shape made a significant difference. When using a square wave, an IV curve could not be created because the only data points that could be obtained were from the open circuit voltage and the closest point to short circuit current. With only these two points, it would be impossible to determine the actual curve. The triangular waveform was much better, but lacked a large range of points along the IV curve. The sinusoidal waveform allowed for the maximum amount of points to be extracted from the voltage output data for the IV curve.

The hand held device used all of the results of the previous experiments to create a hand held device. The final device is able to acquire data from a solar panel using the solar simulator. The data used to generate the graph in Figure 36 was obtained from the second time the gate was triggered. This is because the voltage, from the first time the gate was triggered, was still reaching the steady maximum. The third time the gate was triggered, the voltage had decayed slightly, and had reached a critical point where the gate had become saturated, making it difficult to obtain points that would be present on an IV curve. The transient analysis simulated using Multisim 11 was very accurate. The voltage output from the drain of the transistor closely matches the voltage acquired using the hand held device. The acquisition tests show that the voltages acquired from the hand held device are consistent with those of the original circuit using different oscilloscopes. The hand held device had a contact resistance of .5 ohms, which lowered the output voltage. This issue was corrected by adding the voltage across the .5 ohm resistor for each point. The IV curve for each of the three devices also needed to be adjusted for any data that was captured when the voltage from the panel was decreasing. This was modeled by the equation $21 - 21*\cos(125.6t)*e^{-t}$. The resulting voltages were added for data points captured when $.003s < t$. The resulting voltage change was a difference of little more than 100 mV.

CONCLUSION

A solar simulator data acquisition device has been thoroughly examined to create an IV curve as a hand held device. The hardware and software solutions involved in the research of this device has been discussed. Prototypes were designed, beginning with an oscilloscope used to capture data, and a function generator to drive the gate of a transistor to vary the voltage. This method progressed with each test, allowing for the most efficient design to acquire data. These configurations were then compressed into a small form factor that allowed for simplicity in transportation and deployment. The acquisition of data for producing an IV curve has been proven to work effectively using the hand held device.

FUTURE WORK

This hand held acquisition device is many steps ahead of where this project started, but there are still many things that should be considered:

- Improve the python program so that the voltage data can be easily manipulated and transformed into a IV-Curve. The current python code allows for the raw data to be used for calculating the current and generating a new file with voltage and current information, but the output data is not easily graphed. This most likely can be done using MATLab.
- Change to a microcontroller with a better sampling rate. The Arduino was very good for being able to prove that the device works, and was able to make an IV-curve, but does not have a lot of data points. This can easily be corrected by using a different microcontroller.
- Add wireless capability. If a microcontroller that contains an OS is used, there is a possibility that wireless can be used to send data from the hand held device to a laptop. This would benefit anyone who needs to test multiple solar panels, and does not have time to continually move the data to a computer using a USB cable.
- Improve battery life to allow for the device to be used for many hours without needing new batteries.

REFERENCES

1. T. Woody, “Solar Industry Anxious Over Defective Panels” (2013). Accessed on: June 2014. Available at <http://www.nytimes.com/2013/05/29/business/energy-environment/solar-powers-dark-side.html?pagewanted=all&_r=0>

2. Solarex, Solarex SX-30U Manual. The Corporation, Fredrick, Maryland, 1999. Available at <http://www.solarcellsales.com/techinfo/docs/E0503.pdf>

3. Atmel Corporation, Atmega328p datasheet. October, 2013. Accessed on: November, 2013. Available at <http://www.atmel.com/images/Atmel-8271-8-bit-AVR-microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf>

4. D. Bates & A. Malvino, *Electronic Principles, 7th ed.*; pp. 377-381. McGraw-Hill, Boston, MA, 2006.

5. Elinchrom LTD, Elinchrom Digital 2400RX User Manual. The Corporation, Switzerland, 2011. Available at <http://www.fotoflits.com/files/Handleidingen/elinchrom%20digital%201200%20%26%202400%20rx%20%28en%29.pdf>

6. S. Walker, Sciencetech inc, London, Ontario, Canada, November 2013, Private Communication.

7. R. Boylestad, *Introductory Circuit Analysis*, *11th ed.*; pp 149-151. Pearson Prentice Hall, Columbus, Ohio, 2007.

8. Radio Artisan, “Analog Devices DDS arduino Library” (2012). Accessed on: May, 2013. Available at <http://blog.radioartisan.com/analog-devices-dds-arduino-library/>

9. Z. Shaw, *Learn Python the Hard Way, 3rd ed.*, pp. 136-139. Addison-Wesley Professional, Boston, MA, 2013.

10. C. Liechti, “Pyserial Documentation” (2001). Accessed on: May, 2013. Available at <http://pyserial.sourceforge.net/>

11. Arduino, “Arduino Uno”. 2014. Accessed on November 2013. Available at <http://arduino.cc/en/Main/ArduinoBoardUno>

12. C. Kelley & T. Williams, “Gnuplot Homepage” (2013). Accessed on: May 2013. Available at <http://www.gnuplot.info/>

13. N. Gammon, “I2C - Two-Wire Peripheral Interface - for Arduino” (2011). Accessed on: November 2014. Available at <http://www.gammon.com.au/i2c>

14. D. Weiman, “ATmega168 Analog to Digital Converter” (2009) Alfred State College. Accessed on: January 2015. Available at <http://web.alfredstate.edu/weimandn/miscellaneous/atmega_subsystem/ATmega168%20Analog-to-Digital%20Converter%20v01b.gif>

15. Agilent Technologies, DSO1000A/B Series Portable Oscilloscopes Datasheet. The Corporation, USA, 2013. Available at <http://homes.ieu.edu.tr/maskar/EEE331/General/DSO1002%20Oscilloscope%20User%20Manual.pdf>

16. Agilent Technologies, Agilent 33210A 10 MHz Function/Arbitrary Waveform Generator Datasheet. The Corporation, USA, 2012. Available at <http://cp.literature.agilent.com/litweb/pdf/5989-8926EN.pdf>

APPENDIX A

The Elinchrom Digital RX 2400 power pack is one of the most crucial components within the solar simulator. This device powers the Xenon Arc Flash lamp with the amount of power determined by a user. The value on the screen can be used to determine the power being used in Ws, and the f-stop rating.

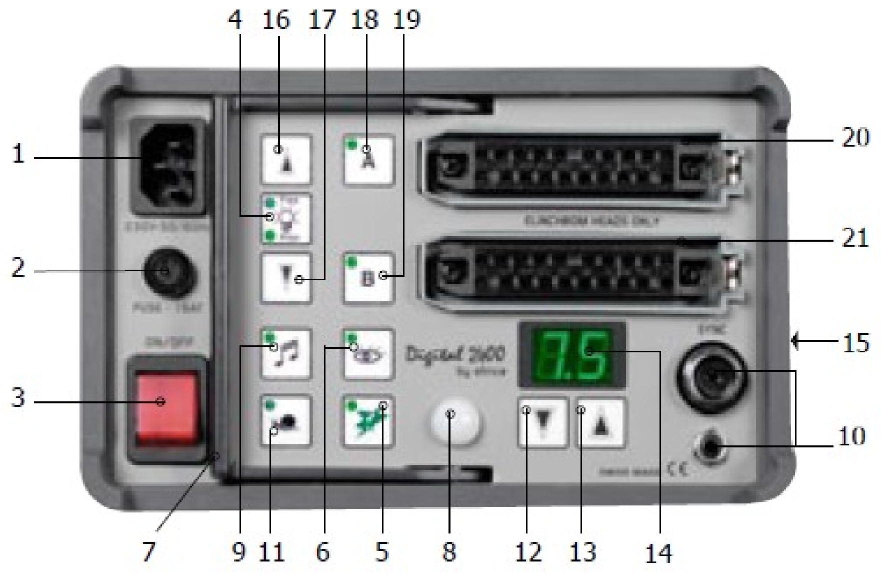


Figure 37. Elinchrom Digital 2400 RX Power Pack.

Digital 2400 RX

Scale	Ws	f-stop	Power
8.5	2400	(180)	1/1
7.5	1200	(128)	1/2
6.5	600	(90)	1/4
5.5	300	(64)	1/8
4.5	150	(45)	1/16
3.5	75	(32)	1/32

Figure 38. Settings for Elinchrom Digital 2400 RX Power Pack[5].

A special cable is used to connect the Digital RX 2400 to the triggering circuit. The cable is plugged into the larger sync port labeled 10 in Figure 40. The end of the cable, which typically would be placed into camera equipment, was cut in order to use a non-proprietary method to trigger the circuit. The cable had two leads: a red lead, and a white lead. The white lead was determined to be a 5 volt signal while the red wire was a lead that was to receive the 5 volt signal in order to trigger the circuit.

To generate the correct full spectrum wavelengths, an airmass filter must be used. Airmass filters allow for the spectral output of the arc lamp to match specific natural solar conditions by filtering the raw light emitted from the xenon arc lamp. Using this airmass filter, the conditions meet the Reference Spectral Irradiance: Air Mass 1.5 Spectra standard set by The American Society for Testing and Materials (ASTM). Only one xenon arc lamp is used to reduce maintenance costs further.

A Xenon Arc Lamp is used for the light source. The light source for the solar simulator is shown in Figure 39. The light source comprised of a xenon arc lamp and an airmass filter. When the light source is placed a focal length *f* away from the light source, the resulting light emitted collimates and is reflected back towards a solar panel as a uniform plain. For this system the focal length is 5 meters.

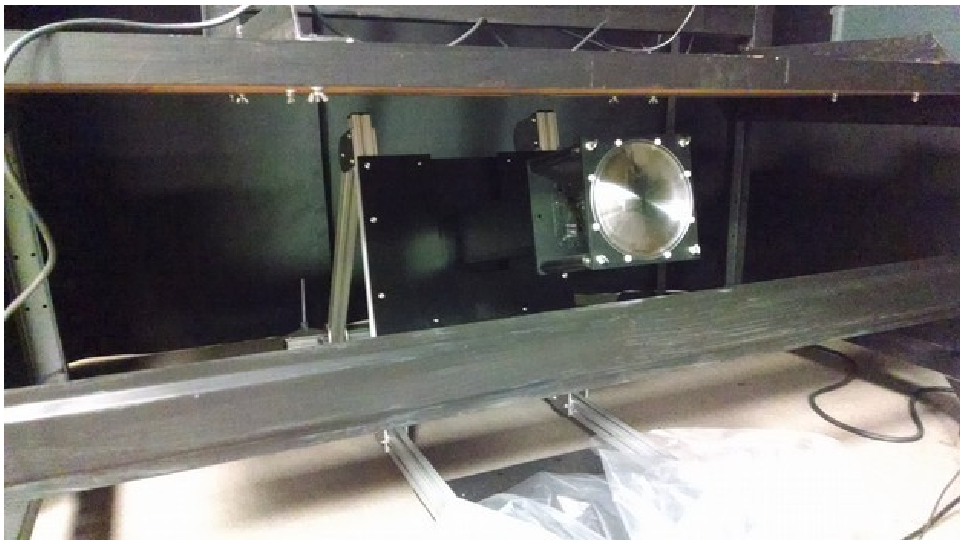


Figure 39. Solar Simulator Light Source using Xenon Arc Lamp.

In place of additional xenon arc lamps, a spherical mirror is used to collimate the light emitted from the xenon arc lamp. The initial cost of the mirror is high at $40,000 dollars, but comes with very little maintenance and will not expire after a limited number of flashes. A top down view is shown in Figure 40.

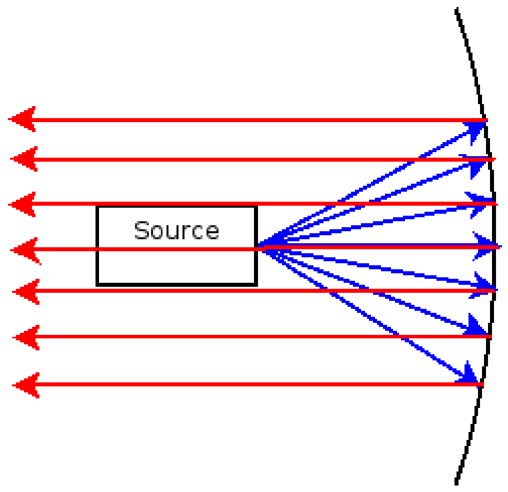


Figure 40. Overhead view of collimation of light rays.

At Alfred University with proper calibration the solar simulator can be classified as a AAA solar simulator in accordance with ASTM (American Society for Testing and Materials). This rating system uses three criteria to determine the quality of the overall solar simulator, namely, spectrum match, irradiance uniformity and stability [11001344].

APPENDIX B

There are numerous ways to acquire data using an oscilloscope. One way that was researched was to use an Agilent software named “Intuilink Data Capture”. This software can be used for capturing data from a source using the oscilloscope, and then transferring the data back to the computer to be examined later. This method was very convenient in that a click on a computer could configure the settings automatically by uploading the correct settings file to the oscilloscope, then capture data from the oscilloscope directly after. Unfortunately the software could not be easily altered to send a signal to the Arduino microcontroller to trigger the solar simulator. This drawback is the reason that the software is not being used anymore.

Acquiring data from the oscilloscope via Agilent software

To manually acquire data from the oscilloscope and transfer it to a PC there are a few steps:

1. Download the “Agilent IO Control” and “Agilent Intuilink Data Capture” software packages. The Agilent IO Control and Agilent Intuilink Data Capture software can be found easily by going to www.agilent.com and using the search box.
2. Install the packages. The Agilent IO Control software should be installed first to ensure the computer has the drivers installed to allow the computer to properly communicate with the oscilloscope. The Agilent Data Capture software can be installed later.
3. Connect the DSO1002A oscilloscope to your PC using USB and turn the oscilloscope on.
4. Use the Agilent IO Control program to configure the driver for the DSO1002A.
5. Open the Agilent Intuilink data capture software.
6. Navigate under the “Instrument” tab and select “Agilent 1000 Series”.

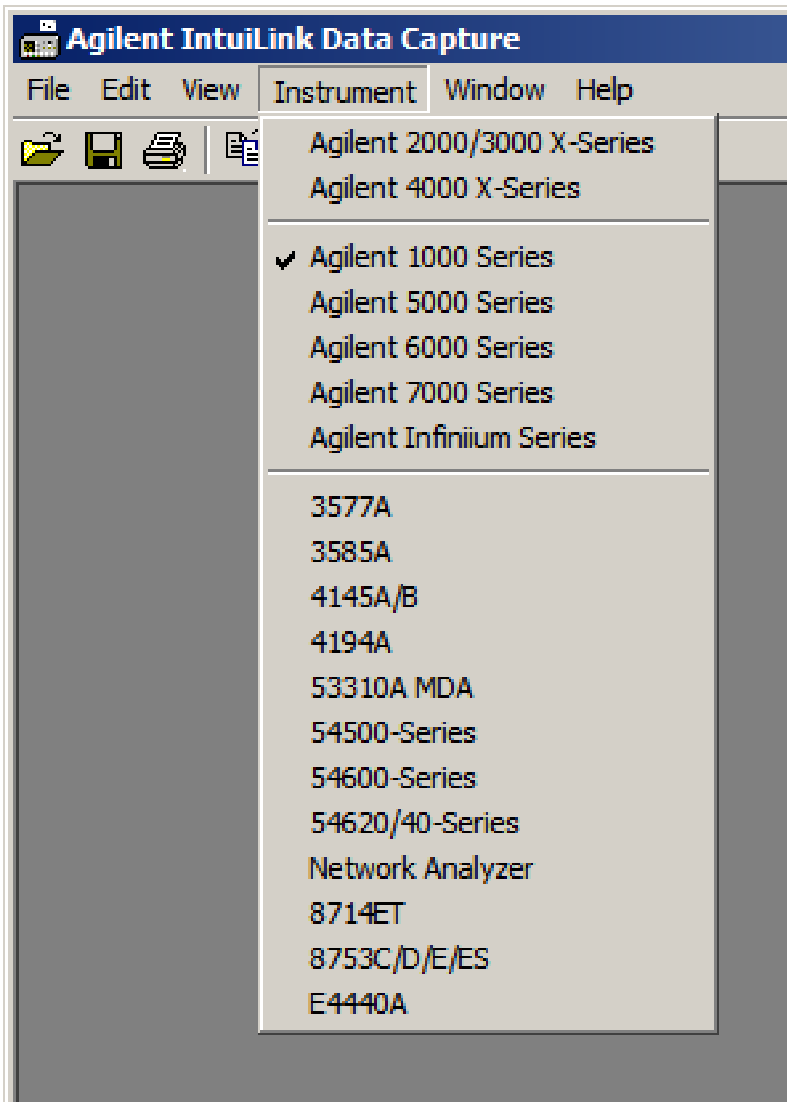
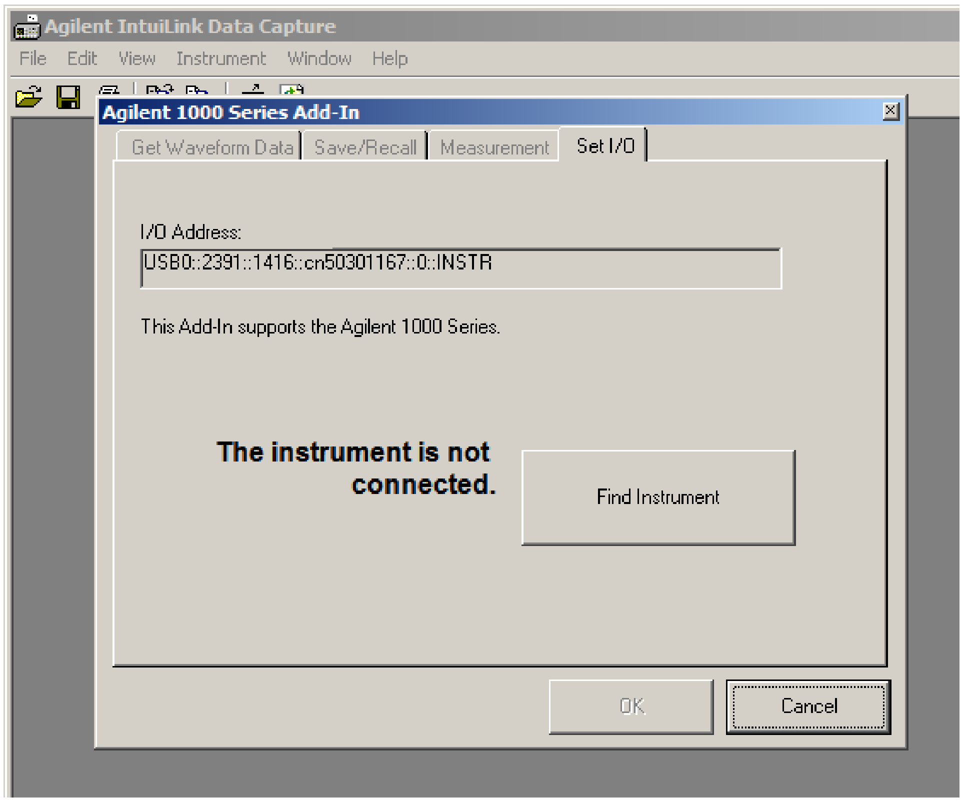


Figure 41. Agilent IntuiLink Software Scope Selection.

7. If the device has not been configured previously, or it needs to be reinitialized, then the following screen will appear:


Agilent IntuiLink Data Capture
File Edit View Instrument Window Help
Agilent 1000 Series Add-In
Get Waveform Data
Save/Recall
Measurement
Set I/O
Find Instrument
OK
Cancel

I/O Address:
USB0::2391::1416::cn50301167::0::INSTR

This Add-In supports the Agilent 1000 Series.

The instrument is not connected.

Figure 42. Agilent IntuiLink Software connecting to Oscilloscope.

8. Click the “Find Instrument” button to search for the device
9. Next click the “Identify Instrument(s)” button to allow for the program to search for the com port that the oscilloscope is connected to. The oscilloscope should be detected at this time. After the oscilloscope has been detected, then it can be highlighted and click the “OK” button.

APPENDIX C

A. Setting Up The Oscilloscope

The Figure 42 is a representation of the oscilloscope that will be used.

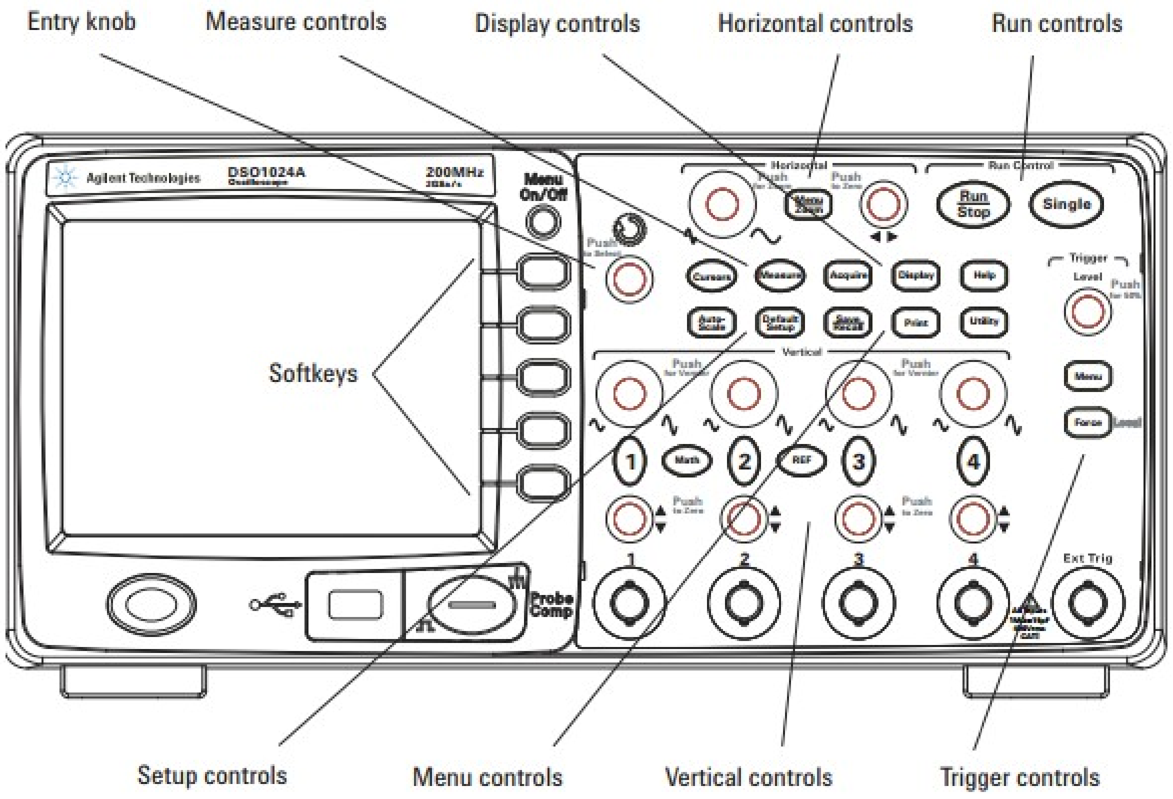


Figure 43. Agilent 1002A Oscilloscope representation from manual[13].

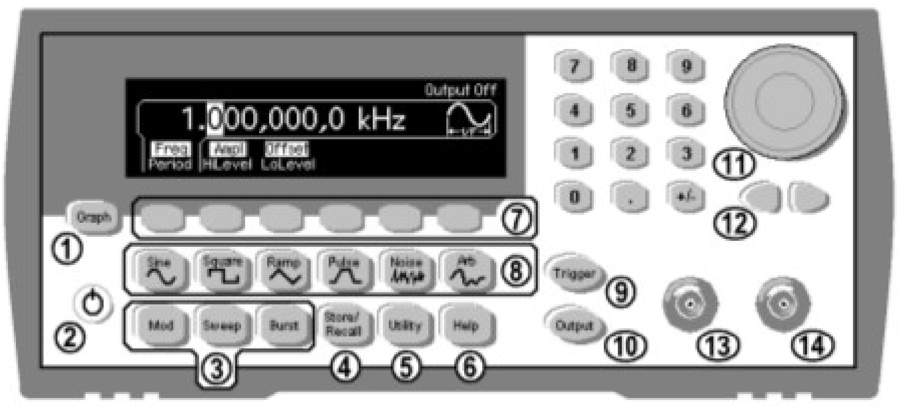
In this case, Alfred University has the DSO1002A oscilloscope, which comes with 2 input channels rather than the 4 channel model shown here. Before acquiring any data, some settings must be altered in order to properly acquire the data. Press the “Default Setup” button to restore the factory settings of the oscilloscope to ensure that no one changed a setting that could alter the results in some way. In this case the Trigger settings will be changed first. To get to the Trigger settings, press the “Menu” button under “Trigger controls”, which are labeled in the Figure above. The using the soft keys and entry knob, the “Mode” setting must be set to “Edge”, the “Source” must be changed to “CH1”, and the “Slope” Must be changed to a rising edge, which looks like an arrow facing upwards only. The trigger “Sweep” must be changed from “Auto” to “Normal” because the oscilloscope will automatically trigger each time the user tries to set the oscilloscope to wait for a rising edge without the signal ever reaching the desired threshold. The rest of the default settings for the trigger do not need to be changed.

Next Channel 1 and Channel 2 must be set to the proper probe settings. In this case 1X probes are used. Press the “1” under vertical adjustment knobs to enter the menu. Change the “Probe” to 1X by pressing the softkey next to “Probe” and change to “1X” and press in using the entry knob. All other default settings for the first channel are fine as is. Repeat these steps for the second channel. The vertical adjustment knobs for channel 1 and 2 need to be set to 10.00V/div in order to view the data from the screen of the oscilloscope before transferring the data to a USB flash drive to be read by the computer.

The final oscilloscope setting that needs to be changed is the acquisition settings. Press the “Acquire” button directly under the horizontal controls. Press the soft key next to “Sequence”. Change the Mode from “OFF” to “Record”. This is all the initial setup that needs to be done to the oscilloscope before recording the data from the solar panel.

B. Setting up the signal generator

To begin use the On/Off Switch to turn on the Agilent 33210A Function Generator. The frequency should be by default set to 1 kHz. If the Function Generator is set to anything other than 1 kHz, use the number pad to press 1 followed by the Menu Operation Softkey underneath kHz. The amplitude needs to be set to 2.5 Vpp. To switch to setting the amplitude press the Menu Operation Softkey underneath Ampl. Next press enter 2.5 on the number pad and press the Menu Operation Softkey underneath Vpp. Next the DC offset needs to be adjusted. To change the DC offset press the Menu Operation Softkey underneath Offset and enter 1.25. Finally press the V_{DC} softkey to enter the value. The Function generator should be set to a sine wave. If not press the sine wave button. Below is a figure of the Function Generator to get a better idea of what needs to be done.[14] To allow the function generator to output the signal to vary the voltage the Output Enable/Disable Key must be pressed.



1 Graph Mode / Local Key	**9** Manual Trigger Key (*used for Sweep and Burst only*)
2 On/Off Switch	**10** Output Enable/Disable Key
3 Modulation/Sweep/Burst Keys	**11** Knob
4 State Storage Menu Key	**12** Cursor Keys
5 Utility Menu Key	**13** Sync Connector
6 Help Menu Key	**14** Output Connector
7 Menu Operation Softkeys	
8 Waveform Selection Keys	

Figure 43. Agilent 33210A Function Generator representation from manual[14].

APPENDIX D

```
//--------------------------------------------------------------------------------
// Hand Held Data Acquisition Device Code
// Created by Steven Gruspier
// Created on May 14th, 2014
// Latest revision on March 12th, 2015
//
// The purpose of this code is to acquire the voltages across the drain and solar panel
//
//--------------------------------------------------------------------------------

// format for dds ddschip(chip, data pin, load pin, clock freq)

#include <stdio.h> // needed for dds library
#include <dds.h> // Library specifically for the AD98XX Series DDS chips
#include <Wire.h>

/*-Fast ADC Code---------------------------------------*/
#define FASTADC 1

// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

/*-IO Table--------------------------------------------*/
// connection to drain voltage moved to slave
```

```
const int vsolar = A0;    // connection to solar panel
//const int simulator = 9;  // connection to solar simulator (on slave now)
const int pushbutton = 3;
const int led_indicator = 10; // indicates if solar simulator is ready to fire

// for sending/receiving data to/from slave Arduino
const int slave_send = 2; // was 9
const int master_receive = 4; // was 10
int data[2];
int parttracker = 0;
int read_tracker = 0;
int receive_state = 0;

/*-Data Capture----------------------------------------*/
const int points = 200;     // This value determines how many actual points
                            // are captured from each analog output
//const long int capt_freq = 0LL; // for debugging
const long int capt_freq = 800LL;
const long int init_freq = 0LL;  // Value for initialization before and after capture

/*-Other important things------------------------------*/
int vdrain_values [points]; // array to store values from the drain of the circuit
int vsolar_values [points]; // array to store values directly from the solar panel

/*-AD9850 Pre-Configuration----------------------------*/
dds ddschip(DDS9850, 8, 6, 7, 120000000LL); // 6,7, and 8 are pin numbers on the
Arduino
// format for dds ddschip(chip, data pin, load pin, clock freq)

void setup()
{
  for(int i = 0; i < points; i++){
    vdrain_values[i] = 0;
    vsolar_values[i] = 0;
  }

  for(int i = 0; i < 2; i++)
    data[i] = 0;

  #if FASTADC
  // set prescale to 16
  sbi(ADCSRA,ADPS2) ;
  cbi(ADCSRA,ADPS1) ;
  cbi(ADCSRA,ADPS0) ;
  #endif

   ddschip.calibrate(-.0418); // calibrates frequency to be exact (dependent on physical
chip)

  pinMode(pushbutton, INPUT);
  pinMode(led_indicator, OUTPUT);
  digitalWrite(led_indicator, HIGH);


   ddschip.setfrequency(init_freq);     // make 0LL during regular functioning, for testing
other
  //delay(2000);
  pinMode(slave_send, OUTPUT);
  digitalWrite(slave_send, LOW);
  pinMode(master_receive, INPUT);

  Wire.begin();
  Serial.begin(9600);
}

void loop()
{
  int state = digitalRead(pushbutton);
  //Serial.println(state);

  if(state == HIGH) {

    ddschip.setfrequency(capt_freq);
    delay(1000);

    digitalWrite( slave_send, HIGH);

    delayMicroseconds(10); // for lag between slave getting signal and starting acq.

    for( int i = 0; i < points; i++)
    vsolar_values[i] = analogRead(vsolar);

    while(receive_state = digitalRead(master_receive) != HIGH)
    {
      //Serial.println("waiting");  // for debugging
      digitalWrite(slave_send, LOW);
    }
    //Serial.println("Done waiting");  // for debugging
    for( int i = 0; i < points; i++)
    {
      Wire.beginTransmission(4);
      int available = Wire.requestFrom(4, 2);
    while(Wire.available())
    {
      data[parttracker] = Wire.read(); // beware of > 0 value on exit
      parttracker++;

      if(parttracker == 2)
      {
        vdrain_values[read_tracker] = data[0]+(data[1]*256);
        //Serial.println(vdrain_values[read_tracker]);
        parttracker = 0;
        read_tracker++;
      }
    }
  }

  read_tracker = 0;

  digitalWrite(led_indicator, LOW);

  ddschip.setfrequency(init_freq); // turn off signal generator

  for (int serial_push = 0; serial_push < points; serial_push++) {
    Serial.println(vdrain_values [serial_push]);
    Serial.println(vsolar_values [serial_push]);
  }
  delay(30000);      // 30000 for solar simulator only (30 secs)

  digitalWrite(led_indicator, HIGH);
}

else
```

```
    digitalWrite(slave_send, LOW);
}
```

APPENDIX E

```
/*********************************************************************

Slave Code For sending and receiving

*********************************************************************/

#include <Wire.h>

/*-Fast ADC Code--------------------------------------*/
#define FASTADC 1

// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

const int data_points = 200;
const int slave_receive = 2;
const int led_power = 3;
const int master_send = 4;
const int analog_voltage = A0; //vdrain
const int simulator = 5;

int send_number = 0;
int receive_state = 0;
int voltage_reading[data_points];
```

```
void setup()
{
  for(int i = 0; i < data_points; i++)
    voltage_reading[i] = 0;

  #if FASTADC
  // set prescale to 16
  sbi(ADCSRA,ADPS2) ;
  cbi(ADCSRA,ADPS1) ;
  cbi(ADCSRA,ADPS0) ;
  #endif

  pinMode(simulator, OUTPUT);
  digitalWrite(simulator, LOW);  // make sure the simulator is not triggered beforehand

  pinMode(led_power, OUTPUT);
  digitalWrite(led_power, HIGH);  // indicate that device is on

  pinMode(slave_receive, INPUT);
  pinMode(master_send, OUTPUT);
  digitalWrite(master_send, LOW);

  Wire.begin(4);
  Serial.begin(9600);
  Wire.onRequest(transfer_data);
}

void loop()
{
  if(receive_state = digitalRead(slave_receive) == HIGH)
  {
    digitalWrite(simulator, HIGH); // triggers solar simulator

    //Serial.println("Master told me to do stuff");  // for debugging
    for(int i = 0; i < data_points; i++)
      voltage_reading[i] = analogRead(analog_voltage);

    //Serial.println("I'm telling the master I am done");  // for debugging

    // WARNING DELAY NEEDED TO ALLOW SLAVE TO BECOME READY FOR
TRANSMISSION TO MASTER
    // WILL LOCK UP MICROCONTROLLER IF A DELAY IS NOT PRESENT DUE
TO MASTER REQUESTING
    // INFORMATION EARLY
    delay(1000);  // needed to ensure slave is ready

    digitalWrite(master_send, HIGH);

    digitalWrite(simulator, LOW); // disables solar simulator trigger

    delay(100); // makes sure master can set slave_receive signal low
  }
  else
    digitalWrite(master_send, LOW);
}

void transfer_data()
{
  digitalWrite(master_send, LOW);
  //Serial.println(send_number);  // for debugging
  //Serial.println(voltage_reading[send_number]);  // for debugging
  Wire.write((byte *)&voltage_reading[send_number], sizeof(int));

  send_number++;

  if(send_number >= data_points) // think about this logic
    send_number = 0;
}
```

APPENDIX F

```
import serial
import sys
import time
import os
import shutil

timestr = time.strftime("%Y%m%d-%H%M%S")
rawfile = timestr + '-rawdata.dat'

ser = serial.Serial('/dev/ttyACM0', 9600)

points = 200    # Points in python code should match
                        # the points in Arduino code
                        # Arduino_pts = Python_pts
count = 0
resistance = 1.15

data = open(rawfile, 'w+')
data.truncate()
data.write('vdrain\tvsolar\n')

ivcurve = open('ivcurve.dat', 'w+')
ivcurve.truncate()

for count in range(0,points):
        # change the first column created into time column
        vdrain = float(ser.readline())
        vdrain = float(vdrain/1024*40)
        vdrain = "%0.3f" % vdrain
        vsolar = float(ser.readline())
        vsolar = float(vsolar/1024*40)
        vsolar = "%0.3f" % vsolar
        vsolar2 = vsolar
        data.write('{}'.format(vdrain))
        data.write('\t')
        data.write('{}'.format(vsolar))
        data.write('\n')
        # This section does the calculation for current to make
  #an IV curve (This needs to be tested in the lab)

        vsolar2 = float(vsolar2)
        vsolar2 = "%0.3f" % vsolar2
        ivcurve.write('{}'.format(vsolar2))

        ivcurve.write('\t')

        current = float((float(vsolar)-float(vdrain))/float(resistance))
        current = float(current)
        current = "%0.3f" % current
        ivcurve.write('{}'.format(current))
        ivcurve.write('\n')

        count = count + 1
data.close()
os.system('cp '+ rawfile + ' rawdata.dat')
os.system("gnuplot rawplot.p")
os.system('cp rawdata.png '+ rawfile + '.png')
#os.system("gnuplot ivplot.p")
```

APPENDIX G

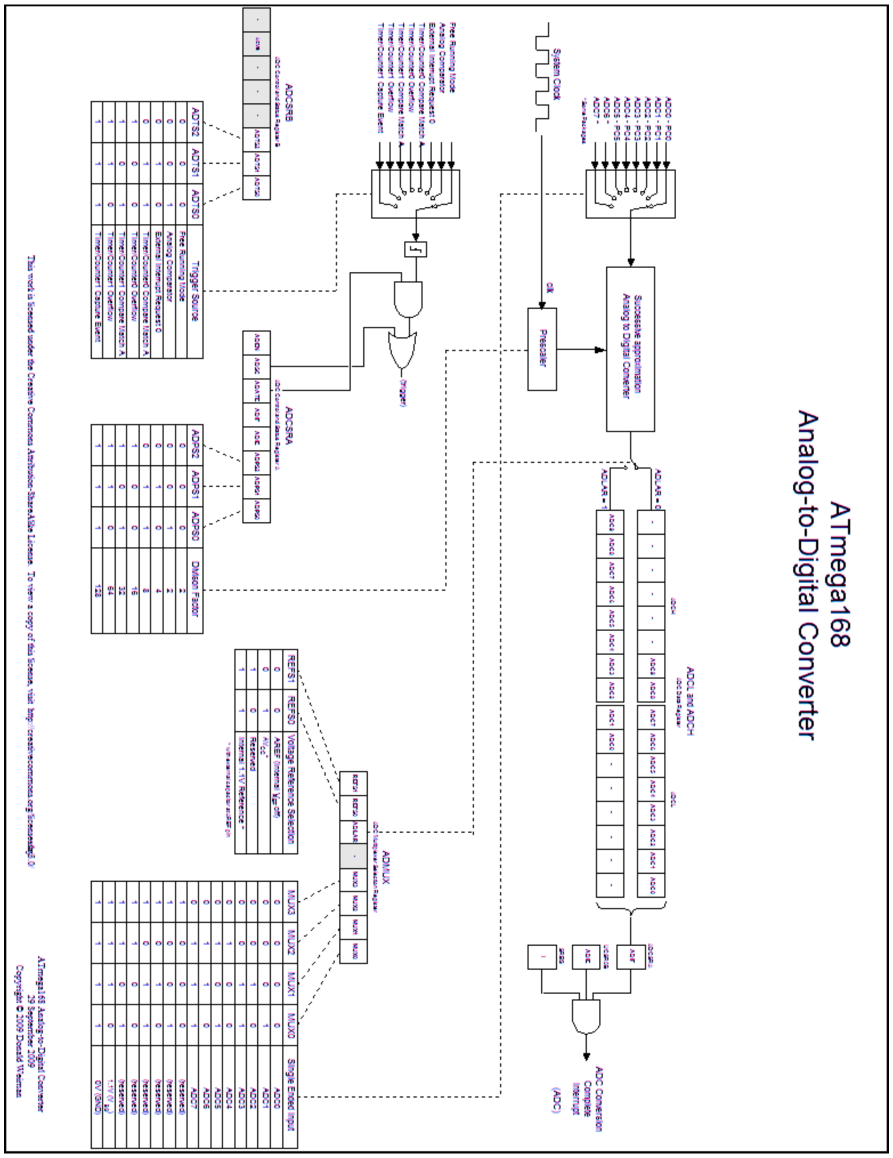

ATmega168
Analog-to-Digital Converter
Successive approximation Analog to Digital Converter
Prescaler
System Clock
ADCSRA
ADCSRB
ADMUX
ADCL and ADCH
Trigger Source
Division Factor
Voltage Reference Selection
Single Ended Input
ADC Conversion Complete Interrupt (ADC)

Figure 44. Analog to Digital Converter (Atmega328p)[12].