The Report Committee for Andrew Edward Hocker
Certifies that this is the approved version of the following report:

EtherLux, A low power wireless display

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: ______________________________
Adnan Aziz

____
Mark McDermott

EtherLux, A low power wireless display

by

Andrew Edward Hocker B.S. ECE

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2009

Dedication

This report is dedicated to my Family: Mom, Dad and my sister and brother for all their support and encouragement raising me and educating me. Finally, this report is dedicated to my wife for giving me the impetus to achieve a Master's degree and having the patience as I went through the Masters program while maintaining a full time job.

Acknowledgements

I would like to acknowledge my supervisor Dr. Adnan Aziz for all of his support and wonderful ideas in developing and expanding this project. This project would never have come to fruition without his aid. I would also like to acknowledge Mark McDermott for reading and reviewing this report.

Dec, 2009

Abstract

EtherLux, A low powered wireless display

Andrew Edward Hocker M.S.E.

The University of Texas at Austin, 2009

Supervisor: Adnan Aziz

Real time information is essential in many businesses and as a method to inform employees and consumers, so that they can make informed decisions. In offices, warehouse and stores it can be advantageous to have tens to hundreds of smaller displays to deliver a variety of information. This paper details the design, implementation and testing of a wireless low power solar powered display system as a solution to deliver real time information. The system uses an Organic LCD to maintain an image for years on no power and uses very little power to update and refresh the display. The system uses off-the-shelf components to achieve multiple updates per day and, with the right lighting conditions, can perform up to one refresh per minute. The system is entirely powered by incandescent light, has a built in radio, and utilizes capacitors to store charge and deliver power, removing the need for rechargeable batteries. The wireless signal works at 2.4GHz and uses the low power 802.15.4 protocol to send and receive data at a range of 75 feet. It has no observable issue operating in environments with 2.4GHz wireless signals, such as 802.11g. The whole system can be built for under $75.00, and takes up an area of 6" x 8" including the photovoltaic cells.

Table of Contents

List of Figures

List of Illustrations

Introduction

It is challenging to come up with a cheap and effective solution to deliver this information. There are a multitude of issues that must be addressed in developing such a system. These include but not limited to, power source, network and communication logistics and protocols, display system, software control.

This project set out to develop and test a prototype that could address these challenges. The system that is built is known as EtherLux because it utilizes light for power (Lux) and the underlying communication founded on the principles of the original Ethernet, as well as play on the old belief that the Ether was propagation medium for E&M waves. The end result uses solar power and capacitive charging to address the power issues by making each system autonomously powered, and decreases maintenance issues by not using rechargeable batteries, which need replacement. A wireless transceiver is used for sending data to the device and the device can transmit status information to help sync data and determine when a device is "alive". Finally an Organic LCD is used to display the data, which has the advantage that it consumes power only when the display is updated with new data. The resulting system can be used for near real time updates through wireless transmission of data.

Chapter 1: *Basic Concepts*

1.1 PHOTOVOLTAIC CELLS

Solar cells use the photovoltaic effect to convert sunlight into voltage. Solar cells are made up of polycrystalline silicon or other exotic materials. Photons with energy higher than the bandgap energy of the material used in the solar cell will generate an electron hole pair also known as carrier generation. The carrier generation is the reason why the solar cell can be used as a voltage source. The major weakness in solar cell design is there efficiency of converting light into energy. Two different types of cells were looked at as potential candidates, CIS (Copper-Indium Selenide) and thin film amorphous silicon made by PowerFilm [1]. CIS cells have achieved efficiencies up to 19.6% [2] whereas PowerFilm achieves an efficiency of about 5% [3]. Due to cost, size and relative performance the PowerFilm cell was chosen.

1.2 802.15.4

The wireless transmission protocol used for this project is based on IEEE 802.15.4 [4]. This particular standard defines the physical layer and the media access control for a low data rate low power transmission scheme. There are multiple options for data rates and transmission band frequencies, and this particular implementation the 2.4GHz 250kbit/s. At the 2.4GHz setting OQPSK (Offset quadrature phase-shift keying) is used so that 2 bits can be sent per symbol time. On top of this 802.15.4 uses DSSS (Direct-Sequence Spread Spectrum), which is a technique for spreading the signal over a larger bandwidth, with a pseudo random modulation scheme. This scheme allows other devices to share the channel because the receiving devices knows what the pseudo random modulation scheme is and can de-spread the signal, any other devices using the same signal but a different pseudo random modulation will just appear as noise. Finally 802.15.4 employs CSMA (Carrier Sense Multiple Access), which check to make sure a channel is clear before sending data.

1.3 Organic LEDs

Organic Light Emitting diodes are similar to traditional inorganic LEDs except that they do not need a backlight, because they are composed of an emissive electroluminescent layer. As a result OLED consume less power and have better contrast ratios. The largest disadvantage is there life times, the organic compounds degrade and the light intensity decreases over time.

1.4 Microcontroller

Microcontrollers function as the CPU and controller for small, embedded devices. They usually offer a minimal instruction set but include significant functionality for analog and digital I/Os, timers, analog to digital converters, serial protocols and some Random Access Memory (RAM). One of the more popular microcontrollers, which is used in this project, is the Peripheral Interface Controller (PIC) made by Microchip Technology. They are low cost, low power, flash-able programming devices and because of their popularity software tools and help is easily available.

Chapter 2: *EtherLux Design*

HIGH LEVEL ARCHITECTURE

At a high level the EtherLux works as follows (See Figure 1). The system first charges up its capacitors until the reach a predefined voltage of approximately 3.06V, at this point the system has enough power to at least send a transmission packet to the base station that it is ready for data and the specific data packet (Starts with packet number 1). The base station at this point will either tell the system to go back to sleep and can give it an exact amount of time to sleep for, between 1-256 seconds, or start transmitting image data. The image data takes 11 packets that carry 100 bytes of actual image data a header and a checksum. Once the packet is received successfully the Etherlux sends an acknowledgement back to the base station and checks to see if their base station received the acknowledgement. If it did send properly it will then wait for the next packet from the base station, unless the voltage has dropped below the threshold of 3.06V in which case the system will have to wait to recharge. In the ideal case all 11 packets are sent back-to-back, i.e. the system has enough power not to recharge in between packets, and the image is refreshed on the LCD. Now in the event the light is generating a small trickle of charge the system may only be able to receive one or two packets at a time and then go back into the low power sleep state while the capacitors charge up. As long as the voltage stays above 2.8V the earlier packets of data will be retained because the image buffer, which is within the organic LCD circuitry, will continue to operate and hold state. As a result the base station can slowly send the data to the system as it continually charges up and discharges and will eventually receive all 11 packets and refresh the image.

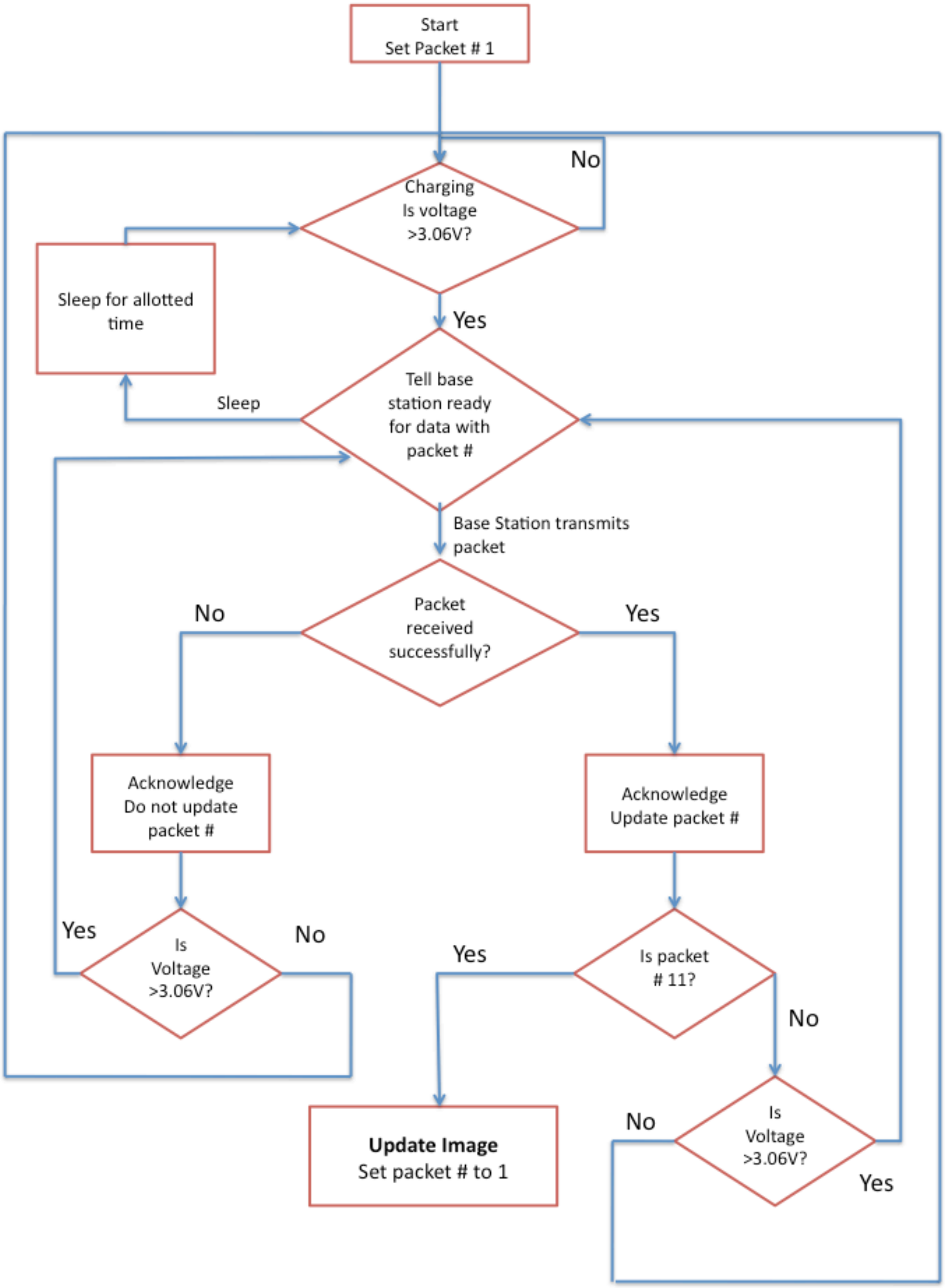


Figure 1: High Level State Diagram.

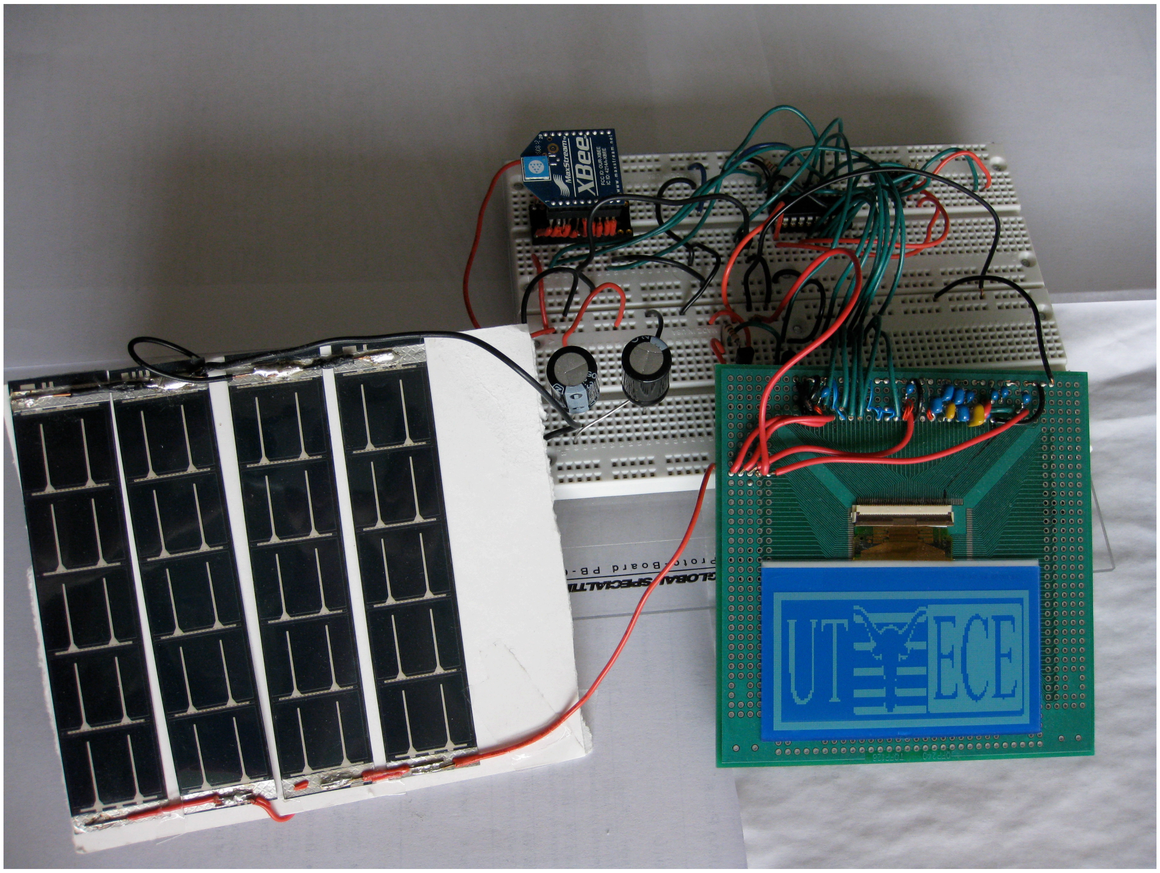


Illustration 1: The EtherLux System.

Hardware components

The use of solar power drove the overall system design, by requiring a minimum number of low power components. Minimal components are necessary to help decrease operating power, keep system cost down and improve reliability. Below is an overview of the major components used in the EtherLux system.

The power source as mentioned earlier is a series of solar cells a 2.35F capacitor and MOS switches which aid in charging and prevent overcharging. The exact operation of this portion of the system will be detailed in Chapter 3. The solar cells are thin film (PowerFilm) and four of them are used, each cell in near full sunlight has a short circuit current of 40mA and an open voltage of 4.1V, with a specification of 25mA current at 3.0V in near full sunlight. In order to achieve higher open voltage and a higher current the 4 panels are divided into two sets of two cells connected in parallel and the two sets are then connected in series.

The organic LCD display is made by Kent Displays and is a Cholesteric LCD with 132x64 pixels, and is encoded in 132x8, 1056 bytes [5]. The display itself only consumes power when refreshing/updating, otherwise it can display an image with no power. The update itself takes about 1-2 seconds depending on temperature with lower temperatures requiring more time. The display includes a LCD driver circuitry, and charge pump, and requires a few external capacitors in order to generate the LCD driver voltage. The entire update consumes about 120mW for 100ms @ 3.3V, and in sleep mode the control circuitry for the LCD, which maintains the image buffer, consumes 1uA.

The wireless transceiver used is a Digi Xbee module. This particular module has a built in antenna and operates at 2.4GHz. In receive mode it consumes 50mA at 3.3V, and during transmission consumes 45mA at 3.3V, there must be more computation required on the receiving side to account for the higher power consumption. The module is able to work up to 90 meters outdoors line of sight and 30 meters indoors. The interface with the module is entirely through UART (universal asynchronous receiver/transmitter) at bandwidths up to 250kbps, which is also the Radio Frequency Data Rate. In the EtherLux the UART frequency is set to 19.2kbps. The system can be setup to be used as drop in UART replacement, which obscures all the wireless details from the end user. In the other mode of operation, known as API mode, the device sends data through packets of less than 110 bytes with header information that more closely models the underlying protocol, which is based on the Zigbee protocol. The devices can work in a peer-to-peer network or in a coordinator network. In the Xbee coordinator mode the Master will perform active scans to determine what devices are present on a particular channel. Since the EtherLuxes could be offline during this active scans, the Xbee coordinator mode was not used. All work on this project is with the Xbee set in peer-to-peer mode but because data is unicast transmissions, and there is the existence of a base station, it acts more like a coordinated network, where data packets are sent to one destination device only.

The microcontroller used is a PIC 16f690 8Mhz, which has 18 I/Os, 4Kx14 of program memory (each instruction is 14 bits) and a 10 bit ADC and has an onboard reference voltage. A higher frequency could have been achieved with a separate oscillator but to decrease the number components only the internal max frequency of 8MHz was used. This particular model can operate at 8Mhz at 2V, and also includes a built in UART module, which is a function of the system clock. Each instruction on the PIC takes 4 clock cycles to complete. The PIC is not pipelined so the effective instructions per second are 2 Million.

The four major components are wired up as shown in the Illustration 2. The PIC microcontroller interfaces with the Xbee transceiver, the LCD display and controls the power charging system. The UART is set to 19200bps, but in actuality due to limitations of using the PIC's internal PLL of 8Mhz the PIC can only achieves a rate of 19230bps or 0.16% error in frequency. The reason for this error is due to limitations in PIC baud rate generator. The baud rate for PIC UART is calculated as follows [6]:

$$DesiredBaudRate = \frac{F_{OSC}}{16\left(\left[SPBRGH : SPBRG\right]+1\right)} \qquad \text{Equation 1}$$

Where SPBRGH and SPBRG (Baud Rate Generator) are a register pair, of an integer value, that determine the period of the free running baud rate timer and Fosc is the clock frequency (8MHz). To determine the value of the baud rate generator the equation can be re-arranged:

$$SPBRGH : SPBRG = Round\left(\frac{F_{OSC}}{16DesiredBaudRate} - 1\right) = Round\left(\frac{8MHz}{16 * 19.2kbps} - 1\right) = 25$$

As a result the frequency of the UART and error is, based on Equation 1:

$$DesiredBaudRate = \frac{F_{OSC}}{16([SPBRGH : SPBRG] + 1)} = \frac{8MHz}{16 * (25 + 1)} = 19230.77bps$$

$$Error = \frac{19230.77 - 19200}{19200} = 0.16\%$$

The error in frequency between the PIC and the Xbee can be resolved with software but at the expense of some packets being resent, which consumes extra power.

The PIC controls the LCD using 4 data pins and a clock pin. The data pins are *Reset_L* (Active Low), *ChipSelect_L* (Active Low), *Data_or_Command*, *SDIN* (Serial data in) and the clock is called *SCLK* (Serial clock). For the Xbee the PIC connects to both the *Dout* and *Din* of the Xbee and the Sleep pin. The PIC uses two extra pins to drive a couple of MOSFETs for power control, see Chapter 3, and one analog input pin to measure the Vdd rail's voltage.

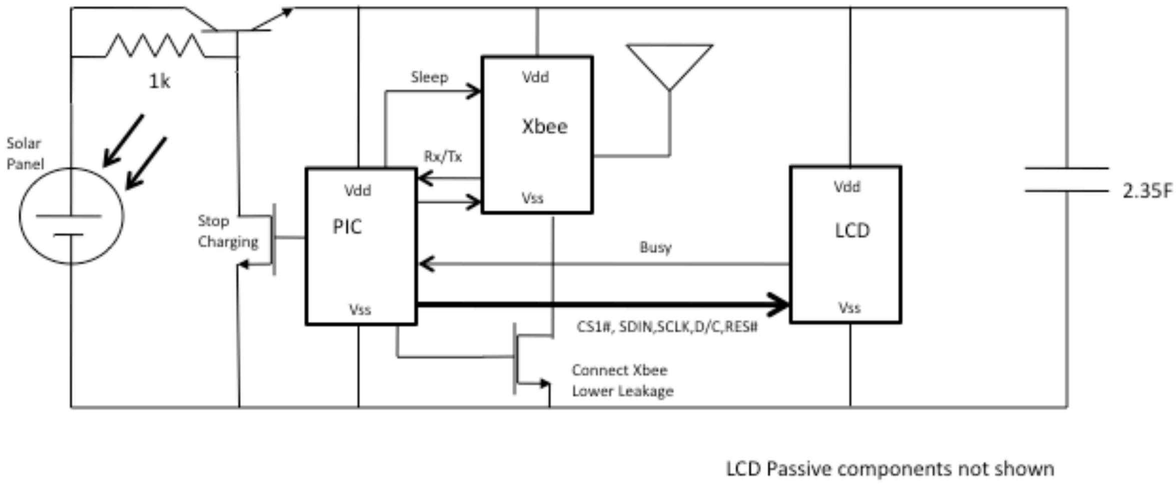


Illustration 2: Circuit schematic of all hardware components.

CONTROL SOFTWARE

The State diagram for the EtherLux is shown in Figure 2. The EtherLux initializes, setting up the frequency the ADC and the UART, and then moves into *InitialState* where the system first enters a sleep loop where a register called *SleepVal* sets the number of times the system will call the 1.057s PIC low power sleep state. The sleep state time is implemented with a watchdog timer on the PIC that is generated by 32768 periods of 31KHz on board clock. During each cycle after coming out of sleep the voltage is measured. After the sleep loop is over the voltage is checked to see if it is high enough (above 3.06V) to move onto the next state *StartFrame*, otherwise *InitialState* is rerun. In *StartFrame* the Xbee is turned on the LCD is reset and 13.2ms is given after closing the footer MOSFET on the Xbee in order to initialize [7]. At this point the Xbee enables its UART so that it can communicate with the Xbee.

The system now moves to *StartFrameCall*, where the PIC sends out a packet to the Xbee that will be transmitted to the base station to tell the base station that it is ready to receive data. The PIC sends this packet to the Xbee requesting an acknowledgement from the base station when received. If no acknowledgement occurs the part returns to the *InitialState* (According to the Xbee spec the Xbee itself will attempt to send the data three times). If there is a UART error (due to the clocks being off between the Xbee and the PIC) or the Xbee is unable to establish a clear channel, the *StartFrameCall* state will be run again, for a maximum of three times after which point the system will return to the *InitialState*.

If the base station responds and no errors occur the next state is the *ReceiveFrame*, which based on a particular data command will move to the *GraphicsFrame* or the *SleepFrame*.

In the *SleepFrame*, the EtherLux will change value of *SleepVal*, allowing the part to spend more time in sleep and build up more charge. The base station can decide this because it is sent knowledge of the systems voltage earlier. If there is an error, the sleep value will not be updated, otherwise the value is updated and in either case the PIC moves to the *Sleeptran* state where it will send a packet to the base station to tell the base station that either the sleep value was properly updated or not, and then the system moves to the *InitialState*.

In the *GraphicsFrame* state there an extra byte to determine what packet worth of data is being set. Since the display requires 1056 bytes and each packet is only capable of 100 bytes of data, 11 packets are necessary to send the entire image to the system. This packet number is used to convert to a particular address on the LCD has to be immediately written to the LCD. This is necessary because the PIC does not have enough memory to store the 100 bytes of data. The data is then written into the LED's buffer and when the whole packet is finished the PIC determines if a UART error occurs. The PIC then moves to the *GraphicCont* state where the status of the last packet is communicated to the base station and part moves to the *ReceiveFrame* State unless all 11 packets were received successfully. If the packet was not successfully received the status of the packet to the base station will say so and it is up to the base station to resend the missing packet.

If all 11 packets were received successfully the PIC moves to the *UpdateDisplay* state, which turns off the Xbee, to conserve power since the display update has significant power draw for 100ms, and then turns on the LCD requests it to update waits till it finishes and then moves back to the *InitialState*.

Now if while the data is being received or a transmit is occurring and the voltage drops to a point where the Xbee is no longer functioning, this will always end in causing a UART error and the system will go back into the *InitialState*. As long as the voltage stays above 2.8 volts the LCD will retain the portion of the packets that were received properly. It is now up to the base station to realize that no proper acknowledgement of the data occurred by the wireless system and therefore the particular packet needs to be resent when the system comes back online. So once the system has enough charge built up it can receive the failed packet and continue until the whole image is received. This way an image can be sent over time if the system is in very low light conditions where one packet at a time is all the system can handle.

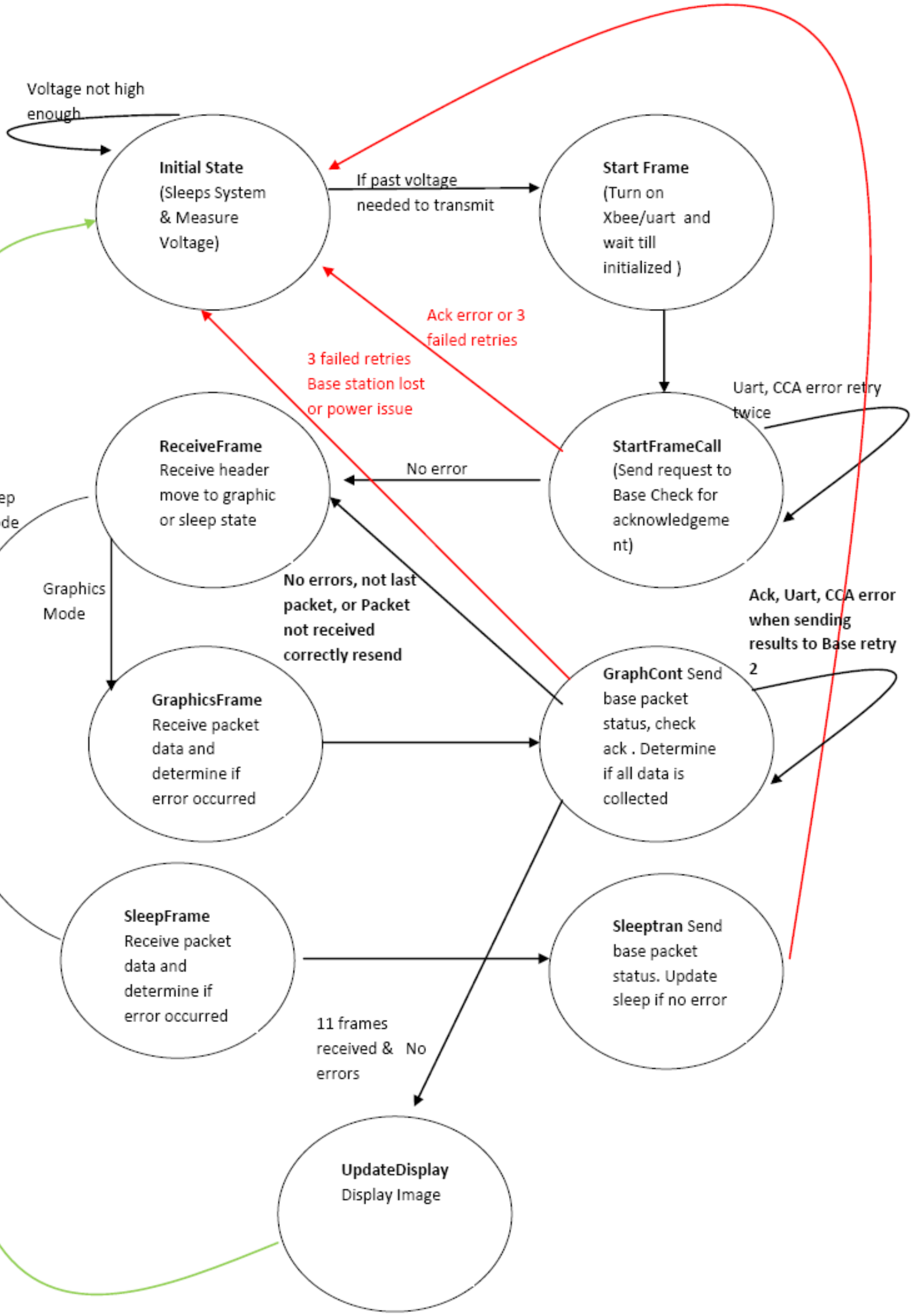

Voltage not high enough
Initial State
(Sleeps System & Measure Voltage)
If past voltage needed to transmit
Start Frame
(Turn on Xbee/uart and wait till initialized)
Ack error or 3 failed retries
3 failed retries
Base station lost or power issue
Uart, CCA error retry twice
ReceiveFrame
Receive header move to graphic or sleep state
No error
StartFrameCall
(Send request to Base Check for acknowledgement)
Sleep Mode
Graphics Mode
No errors, not last packet, or Packet not received correctly resend
Ack, Uart, CCA error when sending results to Base retry 2
GraphicsFrame
Receive packet data and determine if error occurred
GraphCont Send base packet status, check ack . Determine if all data is collected
SleepFrame
Receive packet data and determine if error occurred
Sleeptran Send base packet status. Update sleep if no error
11 frames received & No errors
UpdateDisplay
Display Image

Figure 2: State Diagram for the PIC.

Chapter 3: *Circuit Design*

OVERVIEW

The components selected can all operate together at the same Voltage, 2.8-3.4, which greatly simplifies the design because only a single supply is needed to control everything; this is of course at the cost inefficiency with respect to energy usage. According to the specification, the PIC elsewhere actually start operating at about 2.0V which means it will be able to control both the LCD and Xbee while the system is still charging, which is essential to the way the entire system works. The solar panel charges up the capacitors to generate the power plane voltage (Vdd). In the range of 1-3.5V the current is relatively constant, as is seen in Figure 3, which plots current versus voltage under low light conditions. Low light in this Figure is defined as approximately 350 Lumens, based on the output of a 20Watt 120V halogen bulb [8]. Full sunlight measurements were also taken on the solar panel seen in Figure 5 and 6. Note that at full sunlight the solar panel acts as a relatively constant current source over the voltages used in the EtherLux. These measurements were made by connecting the solar panel directly to various known resistance values to determine the solar panel's current voltage relationship.

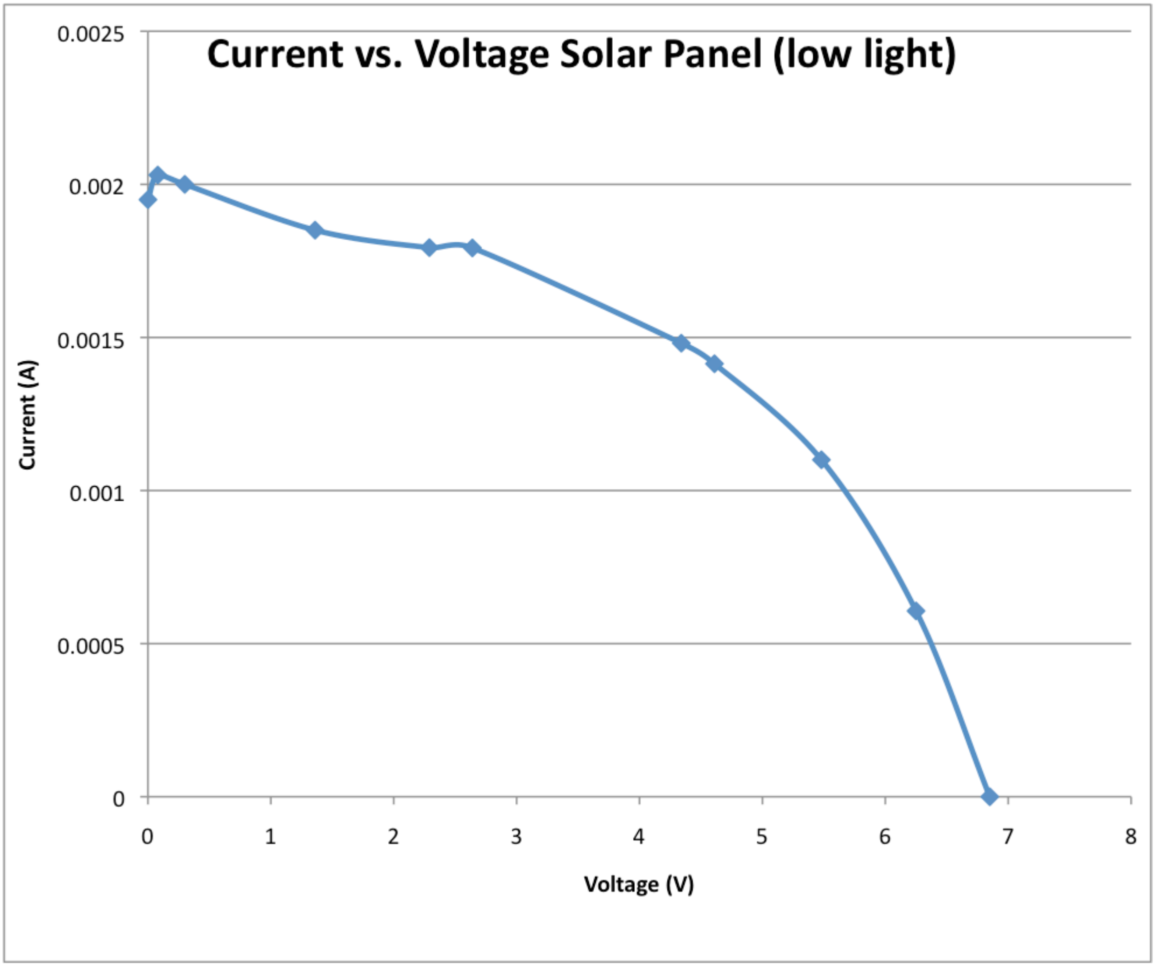


Figure 3: Current vs. Voltage for Solar Panel in Low light.

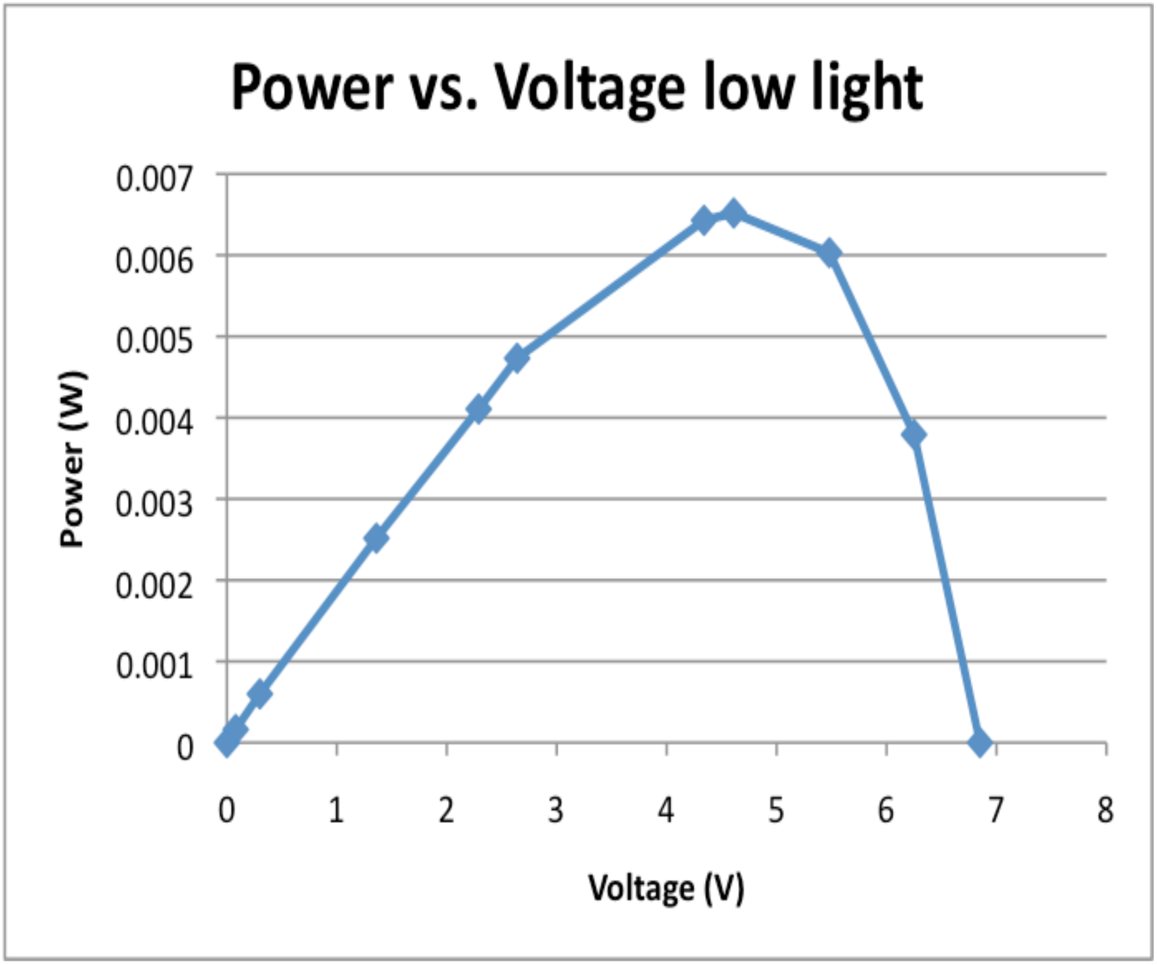


Figure 4: Power vs. Voltage for Solar Panel in low light.

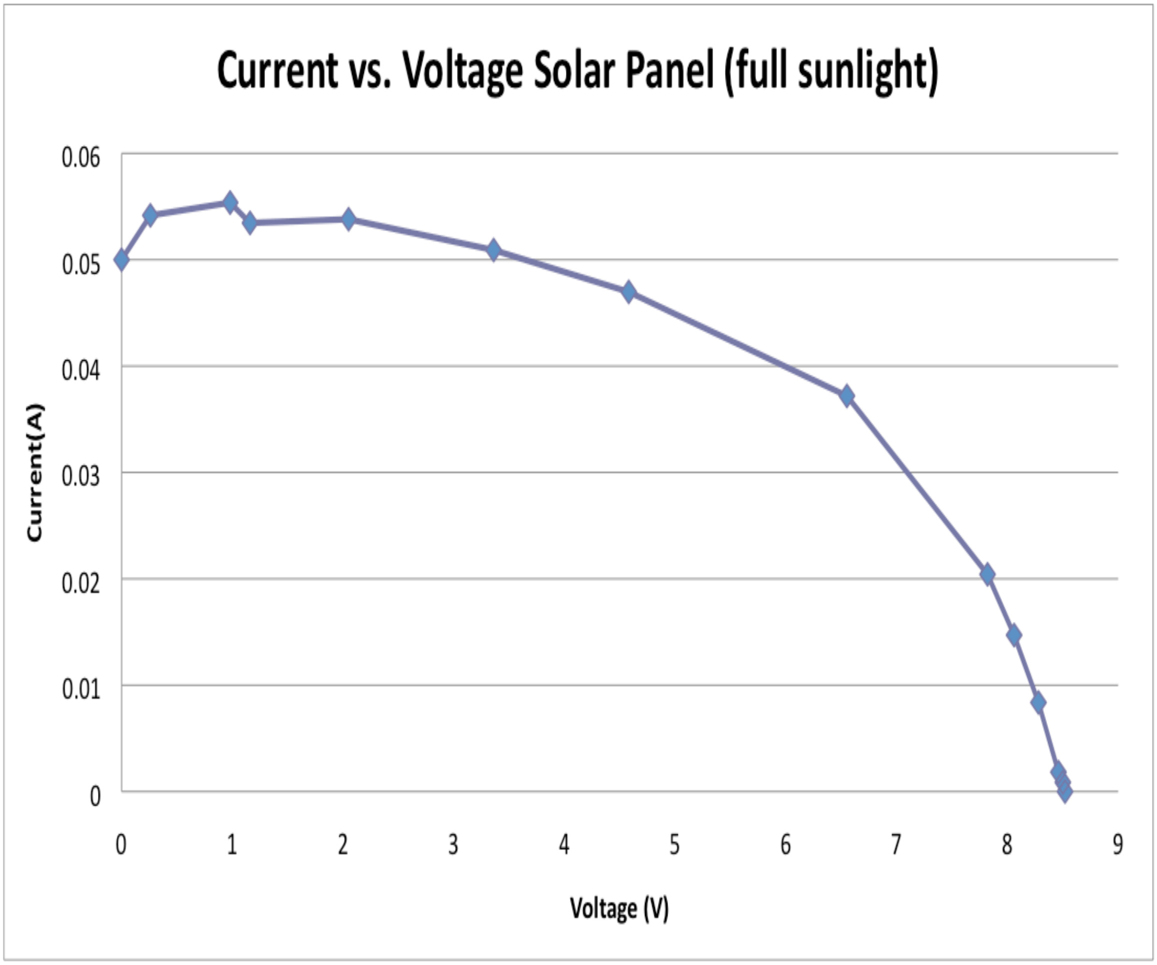


Figure 5: Current vs. Voltage for Solar Panel in full sunlight.

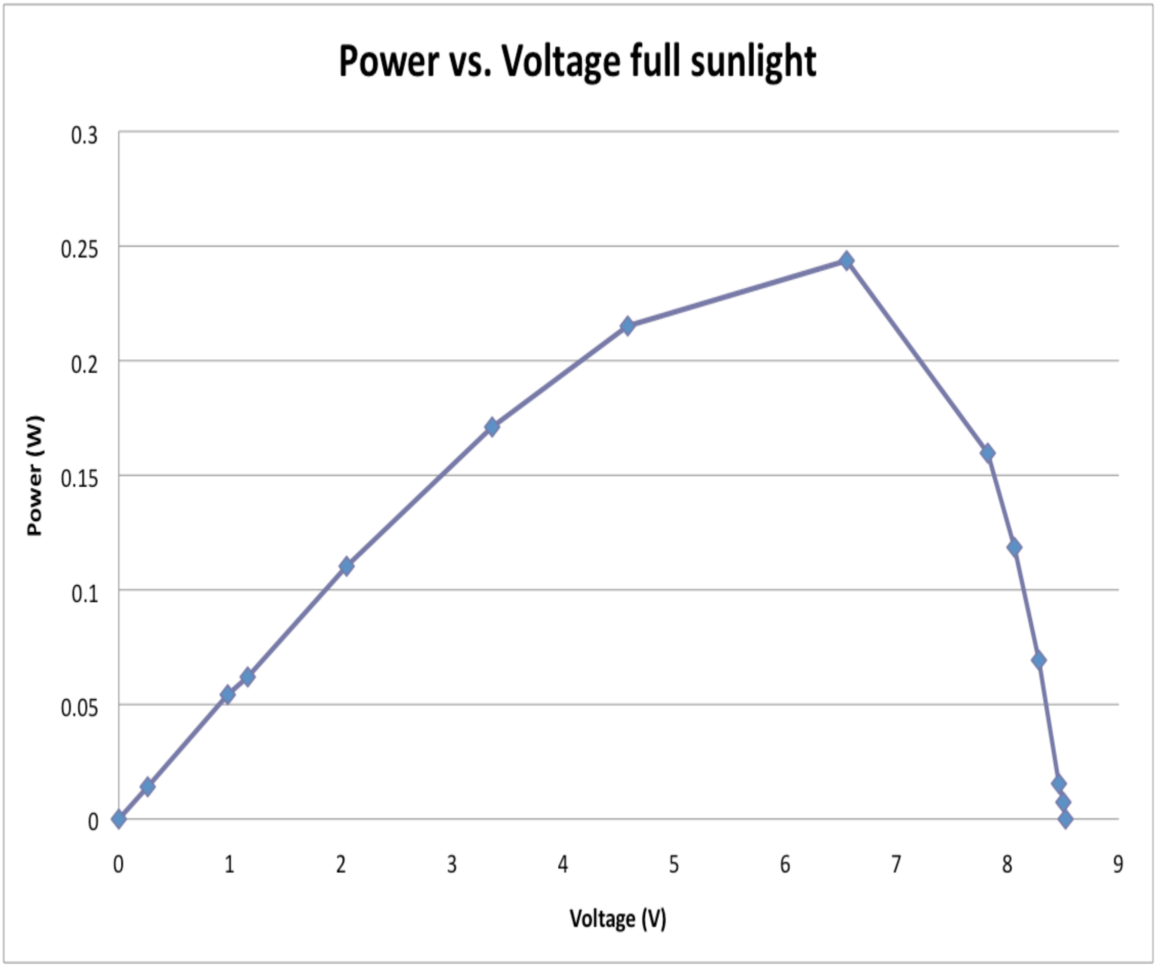


Figure 6: Power vs. Voltage for Solar Panel in full sunlight.

Also worth noting is that a solar cell has a maximum power point where the solar panel delivers the best voltage current product, unfortunately the maximum power point is outside of the operating voltage of the system in both low and full lighting conditions. Though one good note is that as the system charges closer to the 3.06V minimum voltage, the solar panel is putting out more power and becoming more efficient.

Overvoltage Protection

If the solar cell is modeled as a nearly constant current source the voltage ramp that is seen on the capacitors and the entire Vdd plane can be expressed as:

$$Q = CV \Rightarrow I = C\frac{dV}{dt} \Rightarrow \frac{dV}{dt} = \frac{I}{C} \qquad \text{Equation 2}$$

To regulate the voltage of the supply, the onboard 10-bit ADC of the PIC is used to determine the current voltage. The current voltage can then be used to decide if capacitors should continue charging or be cutoff from the solar panel when the voltage is the maximum operating voltage of the components. The PIC uses an internal 0.6V voltage reference to compare the voltage against. The equation to represent the Vdd voltage is [6]:

$$Vdd = 0.6V * \frac{2^{10} - 1}{VP6Count} \qquad \text{Equation 3}$$

VP6Count is the 10 bit value that is the result of the sensing. From the equation Vdd is inversely proportional to VP6Count and in order to have a reliable Vdd measurement the variations in the 0.6V reference need to be accounted for. Unfortunately these variations are different from PIC to PIC so a calibration method is needed. A voltmeter and some experimental PIC code were used to get an accurate reading of the voltage.

One of the drawbacks of using the PIC to measure the voltage is that the whole procedure uses power that would otherwise be stored for receiving and displaying data. To counter this the PIC can be put into a low power sleep state for most of the time and then at specific time intervals wake up and make a voltage measurement. As a result it needs to be determined as to how long the PIC can sleep without allowing the voltage to climb above the specific maximum voltage. Using equation 2 the most the voltage can rise (assuming we are operating at max current of the solar panel) in a time period Δt is:

$$V = \frac{I}{C}\Delta t$$ Equation 4

Using the maximum current that the solar panel can deliver at 3V, which is 50mA and the capacitance of 2.35F, the voltage rises at a rate of 0.02V/s. Therefore even long sleep times will not result in a large increase in the voltage. After some experimentation ~1 second of sleep time per measurement yielded relatively good voltage control, and a very minimal current draw. The actual sleep time is achieved through setting a watchdog timer for approximately 1.057s and putting the PIC to sleep. When the watchdog timer fills up, an interrupt is sent that awakes the PIC and moves to the next instruction. Though real testing in full sunshine did show that voltage did climb higher than 20mV as calculated (Figure 7), which suggests that the solar cells are capable of higher current output in optimal conditions. A further expansion on the algorithm for calculating voltage is to adaptively change time between measurements as the voltage increases so that regulation is more precise.

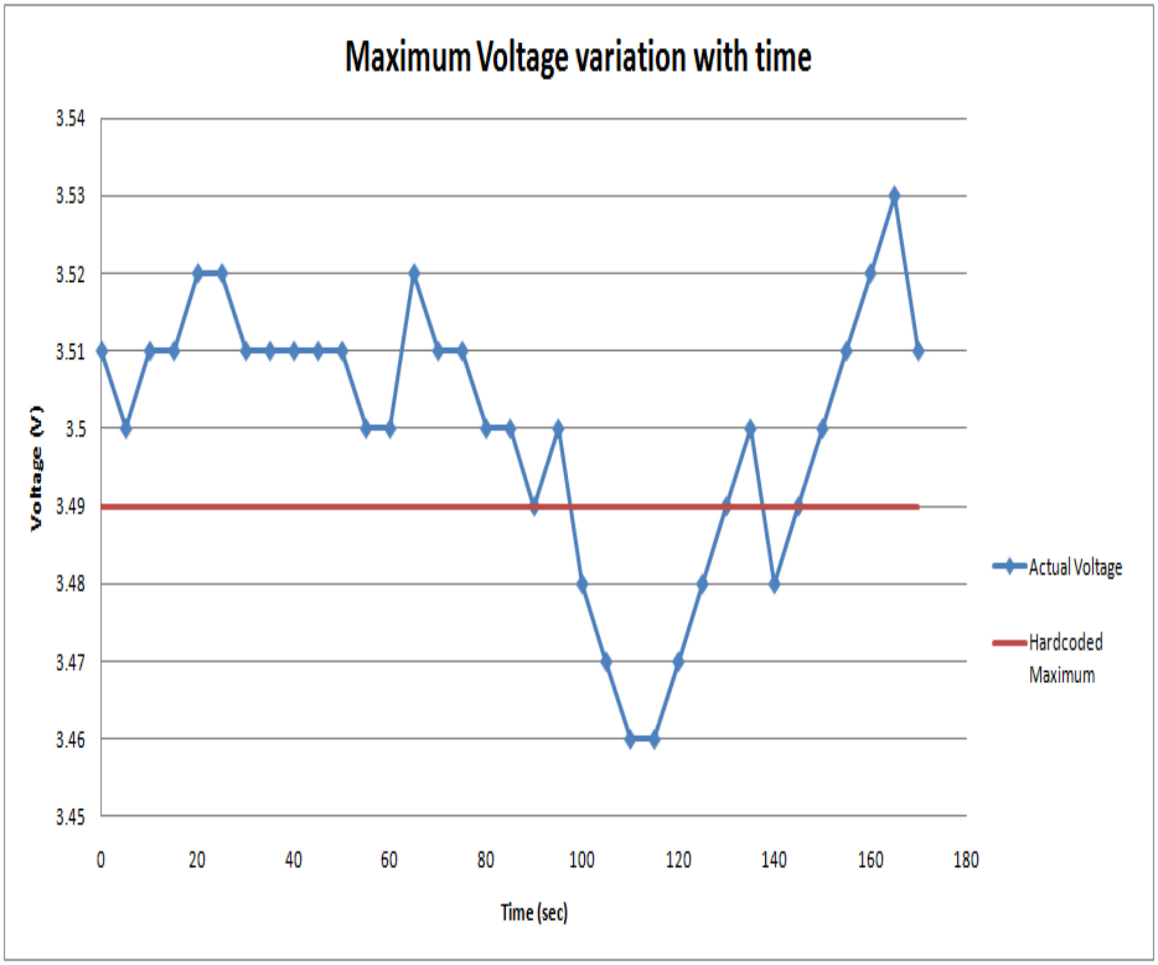


Figure 7: Observed Voltage variations at maximum Voltage.

The circuit that actually cuts off the solar panels from the Vdd plane is composed of two transistors and a resistor. Solar panels as stated before deliver current in the presence of light, but when there is no light to drive them they act as a resistor and will slowly dissipate any stored charge. The solution to this problem is to put a diode between the solar panel and the Vdd plane. When the solar panel is delivering power the diode will be forward biased and the capacitors will charge up, when the light is removed the diode is reversed bias and has a very large resistance value. As a result the undesirable dissipation of charge is decreased by orders of magnitude, and also because the target Vdd is no larger than 3.49 Volts there is no threat of going into the breakdown or avalanche reverse diode region [9].

In the actual circuit this diode is created with an npn transistor by connecting the collector and the base, through a 1K resistor together to the solar panel. So when the solar panel is driving power the diode connected npn transistor will see a diode drop from the base to the emitter, which is 400mV, and the capacitors will charge. When the solar panel is off the transistor is also off because the Vbe is below the threshold. To make this controllable the base of the npn is also tied to a NMOSFET to ground. This NMOSFET's gate is driven by the PIC itself through an I/O pin (RB4). When the PIC senses that the voltage is at the maximum value or just slightly higher it will drive this I/O pin high which will turn the NFET on, and pull down the base of the npn transistor. This will cause the npn to turn off and the charging of the capacitors will cease. There is a current draw from the collector on the Solar Panel to ground while this is going on, but it is in an external current path, so as no effect on the EtherLux. When the voltage drops below the voltage limit the PIC de-asserts the I/O pin and the npn transistor turns on once again charging the device.

During initial testing of the charging circuit the Xbee exhibited a very significant leakage current, when below 2.0V. At low lighting conditions this leakage would consume all the power being generated by the solar panel stalling the systems charging. The exact reasoning for this leakage is not fully understood but it believed to be due to a parasitic current path between the Xbee and the PIC on the UART signals. The solution to this problem, at the expense of some voltage headroom, is to put a footer nmos transistor on the Vss pin on the Xbee. The gate of this nmos is tied to an I/O pin on the PIC (RB6), and is low while the system is charging and any other time that the Xbee is not needed. This solved the leakage issues but did raise the Voltage minimum needed to get all the parts working together. The nmos appears to have a 200mV drain to source potential, which means that the Xbee has about 200mV less Voltage headroom than other components. As a result the max and minimum voltage had to be raised by 200mV. Also every time the Xbee is turned on from being completely off it needs to be initialized which takes about 13.2ms, which ends up using more energy than it would if it was brought directly out of its sleep state. But the decrease in the leakage current greatly outweighs this power increase from power up, because the system can hold charge longer, and as will be explained later, the current draw of the system when not transmitting or receiving is about 5.5uA, whereas the XBee in either of its doze or hibernate states is <50uA to <10uA, 9 to 1.8 times more current.

The highest external frequency of any signal is 1MHz, which means that in the designing phase there is no need for high signal traces and standard wiring can be used. All the components in the prototype were laid out on a breadboard, but could easily be placed on a cheap custom designed PCB.

Chapter 4: *Experiments and Building Device*

Indoors the EtherLux works at ranges up to 40 feet even in the presence of multiple 2.4GHz 802.11g wireless systems. And the best outdoors line-of-sight distance achieved was 76 feet. The reason that this distance value is significantly shorter than the 300 ft number quoted in the Xbee specification is due to the fact that the Xbee module purchased for this project uses an integrated on chip antenna which is not capable of the long distance ranges that the external antenna versions can achieve.

The device was left in darkness originally at 3.03V (which is below the voltage at which the Xbee can transmit) for 7 hours to see how much current was being consumed in the low power state. After 7 hours the voltage had dropped to 2.97V or by 60mV. This results in an average current draw of 5.53uA in the low power state. With a longer sleep state time the current draw can be decreased further, at the expense of over-voltage protection without the use of adaptive sleep timing.

Using the low light conditions of the solar panel as was observed in Figure 3 the charging was both modeled and tested in the actual circuit (In this test the transmit was turned off, it was only to test how long it takes to get to the maximum voltage). For the modeling, a 3rd order polynomial was fit to model the current as a function of voltage (based on the data in Figure 3) and then using Equation 4 and approximation steps of 5 seconds, the charging voltage was calculated. The results are in Figure 8, which show that the voltage of 3.46 Volts is achieved in 10500 seconds or 2 hours 55 minutes. In actual testing it took 3 hours 7 minutes to charge up to 3.46 Volts, the voltage was not constantly measured during charging due to the fact that the multimeter could have a significant impact on charging time. This is because the multimeter will draw some finite current, and with the small current being output by the solar cell, may represent a sizeable fraction of the current. There are some minor modeling issues between the two, but in low light conditions (350 Lumens) the EtherLux should be able to fully charge up in ~3 hours. But as mentioned earlier the leakage current is incredibly low such that as long as the device gets some exposure each day it should never drop below 2.6 Volts, which greatly shortens charging time.

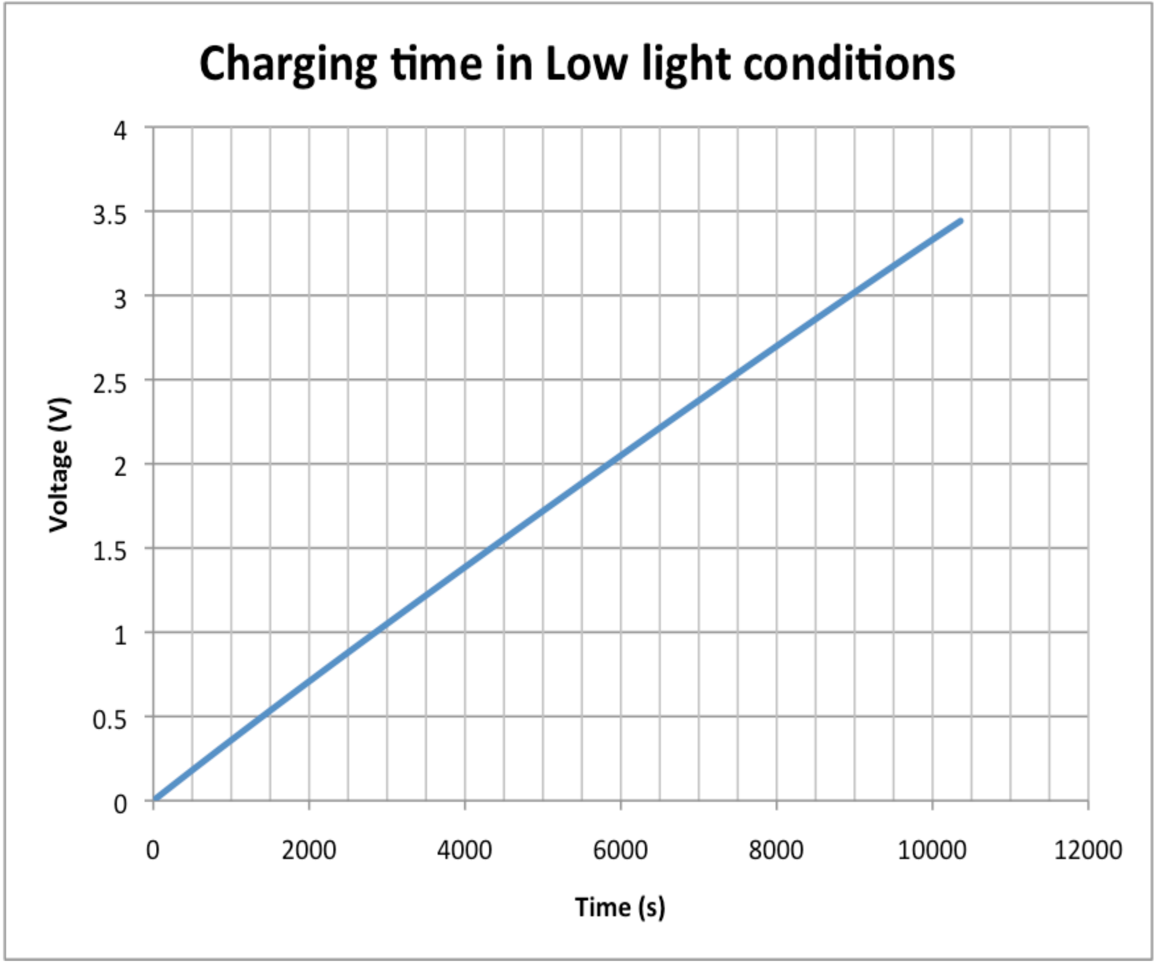


Figure 8: Modeled charging time in low light conditions.

Using Figure 5, a 3rd order polynomial was used to simulate the charging time of the EtherLux in full sunlight. The results are seen in Figure 9, which shows that in 155 seconds (2.6 minutes) the system is charged up to the maximum voltage of 3.49V. In actual testing in full sunlight the charging time was 190s or 35 seconds slower than the simulation. This may be due in part to the sunlight varying slightly due to a few wispy clouds passing by overhead (modulating the current), and due to the 3rd order polynomial, which underestimated currents at lower voltages and over estimated at higher voltages.

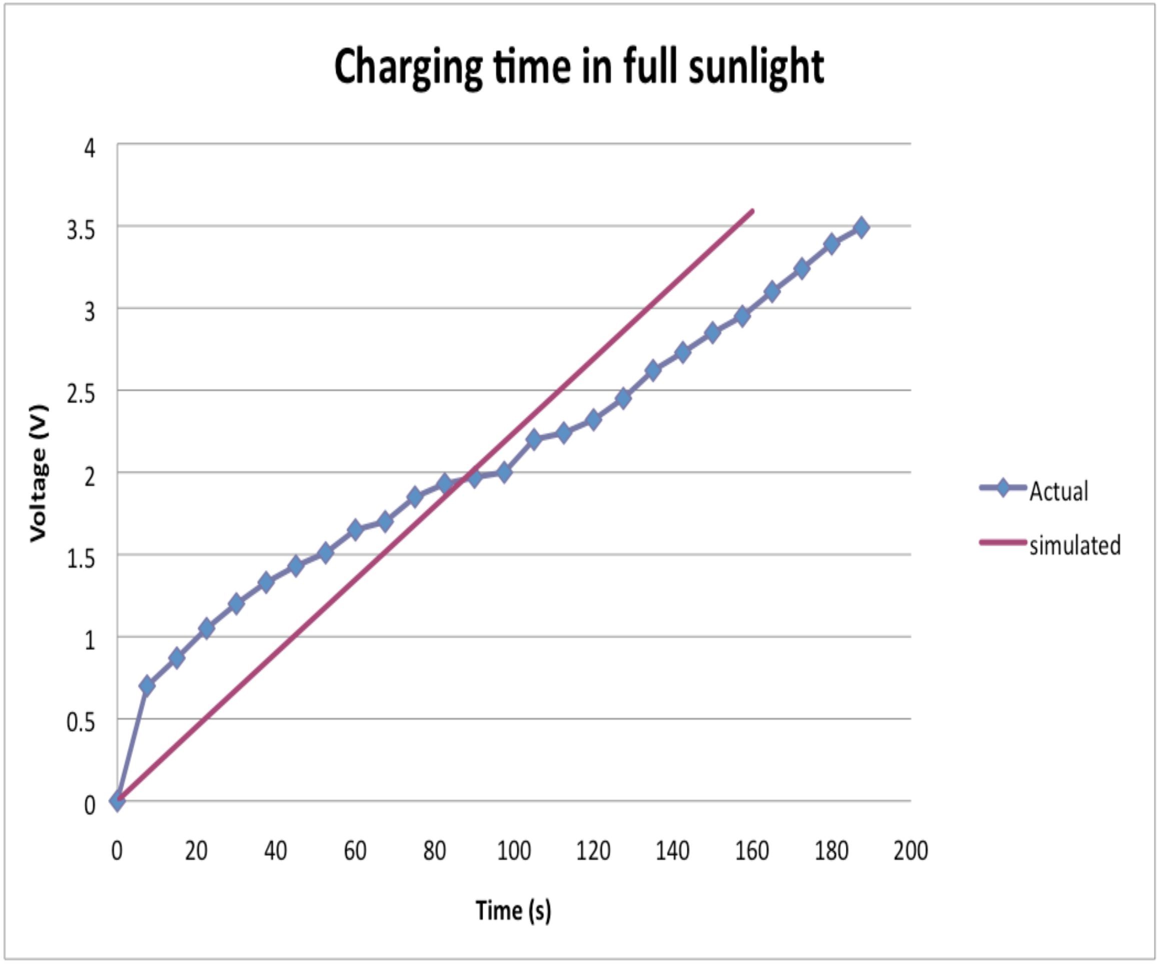


Figure 9: Modeled charging time in full sunlight with actual results.

The system was set about 4 feet from a window during the morning to watch the average refresh times. During the time when the system was seeing somewhere in the range of 75% full sunlight refreshes were taking 44s, with the system spending 16 seconds in the *InitialState* before deciding to come online or not. As the morning progressed by 4 hours, which at this point the sunlight had decreased to 20%, approximate refresh times went up to 26m 57s. At this point the system would keep going to longer and longer refresh times until it can no longer charge at a rate above the low power sleep state current of 5.53uA.

The EtherLux was set at two distances indoors, 20 and 40ft, and sent images over the course of several hours to determine the success rate of sending packets from the base station to the EtherLux.

At 20 feet indoors the base station sent 648 packets to the EtherLux. Of these 648 packets 615 were successfully received or a 94.9% success rate. Of the packets that failed 1.71% were acknowledgement fails, which means that the base station never responded and acknowledged the packet. This might be due to the voltage dropping on the Xbee causing its transmitter power to decrease. 0.78% of the fails were checksum errors, which are due to the UART frequency mismatch as well as 1.71% due to the UART timing out while receiving (Frequency mismatch). This leads to a total error from frequency mismatch of 2.49%, which is significantly higher than the 0.16% calculated in Equation 1. There is possibly some other issues, voltage drooping for instance, that must be exacerbating the frequency mismatch. Finally 0.16% of the fails were due to the UART resetting itself, which only occurs when more than two bytes have been received and not moved out of the queue in the PIC. There isn't a good explanation for this failure because the code was profiled to guarantee that all bytes from the UART would be out of queue way before another byte was received.

The system was also placed 40ft away from the base station inside behind two walls to see if this would increase the failure rate. The base station sent 293 total packets, of these 276 where received successfully for a success rate of 94.2%. The 40ft values were only marginally worse than the 20ft numbers. 1.7% of the failing packets were due to acknowledgement errors. There was 2.72% UART errors plus 1.02% UART timeout errors for a total percentage error due to frequency mismatch of 3.74%. Finally there was 0.34% error rate because EtherLux was unable to establish that the wireless channel was clear to send data on. This error was also interesting because there must have been some wireless product using the same frequency to transmit, (unlikely since only one packet failed in this manner) or some spurious frequency noise from the starting of a car for instance.

Bill of Materials

- 1 Kent Display $17.38
- 2 4.7F 2.3V Capacitors $6.24
- 2 NMOSFETs BS170 $0.98
- 1 PIC 16F690 $2.44
- 1 Xbee $22.95
- 4 PowerFilm solar cells MP3-25 $11.80
- 1 npn transistor 2N3904 $0.50
- Misc Ceramic capacitors for OLED $6.00

Total for Wireless display system: $68.29

Extras needed for this prototype and for the base station:

- 1 prototype board for OLED $15.00
- 1 PIC programmer $34.99
- 1 Extra Xbee base station $22.95
- 1 USB controller for Xbee $24.95

Total for extras: $97.89

Chapter 5: *Relation to Prior Work*

GOOGLE PROJECT

Some of the original ideas for the project were based on a Google project called Radish that was wireless solar display system for displaying room assignments and other related calendar information. Little was known about the design and implementation other than the use of an Xbee and LCD. About 3/4 of the way through this project a link to Google Radish design page was found and it was determined that systems have some very similar features and software implementations [10]. Without much information about the specifics of their system, but since the EtherLux screen is smaller it can achieve a higher refresh rate than their system, but the EtherLux is unable to guarantee one update per day in most lighting conditions.

Chapter 6: *Conclusion*

SUMMARY OF KEY CONTRIBUTIONS

In this project a working low powered solar wireless display system was created and tested and shown to work in various lighting conditions.

FUTURE WORK

One future task would be to have a full PCB design to for the circuit. There are significant power algorithm updates and modifications that could be made to increase power efficiency. The device charges up to 3.4 volts but is unable to operate once the voltage is below 3V, which is waste of a significant amount charge that could still be utilized using some sort of charge pump system. Compression of the wireless data is another area of future work since this can decrease the amount of time the Xbee is on. The EtherLux also has the potential to integrate other sensors to create a more adaptive design. There are many other possible paths to explore based on the EtherLux design.

Appendix

ADDITIONAL PACKET INFORMATION

Some of the extras of the packets are worth noting as well, since there is extra information sent to the base station by the wireless system for decision making by the base station and some debug information. The packet information explained here is the data sent through the UART to the Xbee (and vice versa), as opposed to internal Xbee packet, which uses a different internal structure to communicate between Xbees according to the specifications. When the wireless system sends a packet to the base station it includes a byte that conveys the packet number times 2 plus the status bit. For instance if the 7 packet was received successfully this byte is 0x0F or when the system first comes up it request frame number 1 with a success of zero or the byte 0x02. It then includes the last measured Vp6Count (explained in Chapter 3), which can be used to determine the system voltage. Next byte is the Rssi value of the last received packet, which is, received signal strength indicator in -dBm. Finally the next included byte is the Status byte which indicates a ACK error a UART error, CCA (Clear channel assessment) error a Rx timeout (this occurs because the Xbee will ignore malformed packets due to UART error and the PIC won't know this so it needs a timeout) and finally a UART reset (which occurs when there are some low power issues)

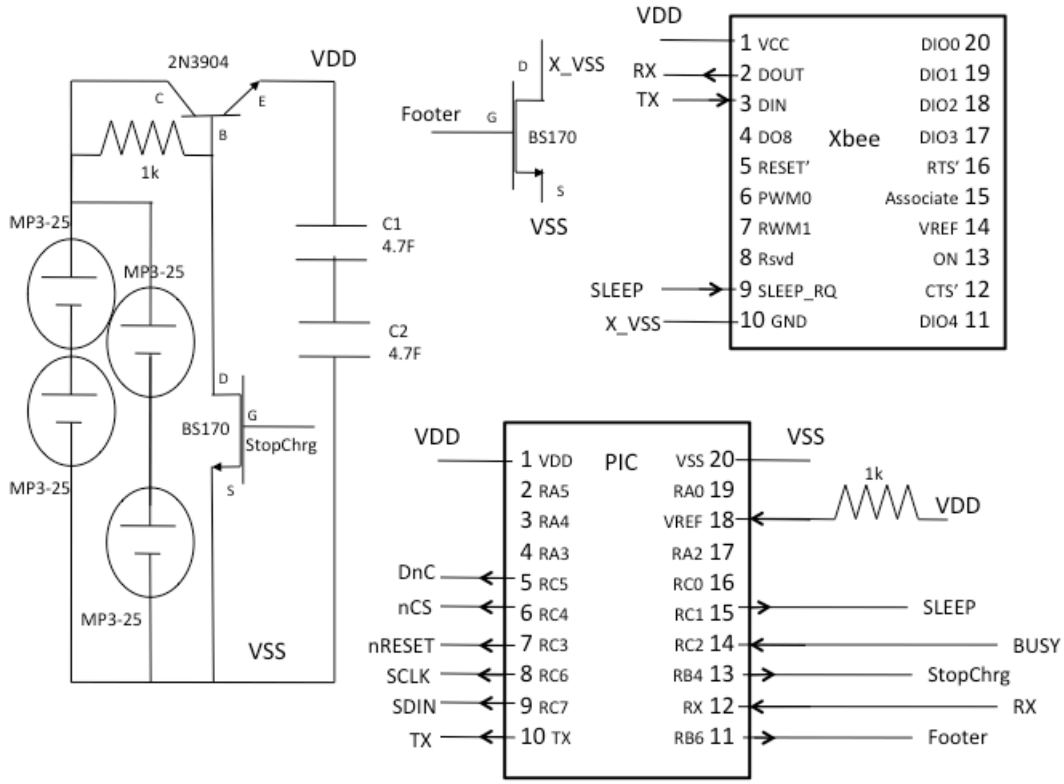
When the base station sends data to the wireless display it includes a byte that has the packet number times 4 and added with the command value 0-3, only 0 (graphics) and 2 (Sleep) are supported. If the part if on packet 11 in the *graphicsFrame* mode it will also include a *sleepVal* byte, so that the base station can update this without another request, and a temp byte which the base station has to send to wireless system to allow the OLED to know the approximate temperature so it can refresh properly.

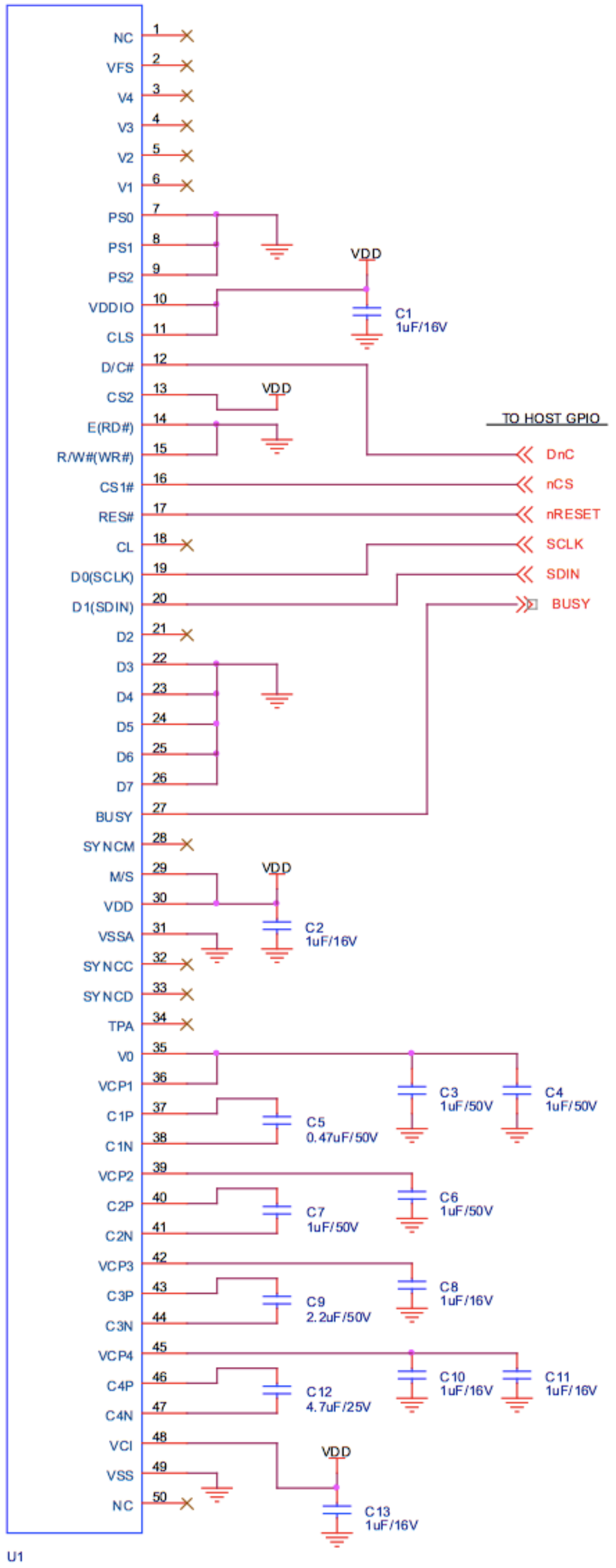
See the reference on the Xbee [7], for further information on the packet structure.

FULL ASSEMBLY CODE

The full assembly code can be downloaded at the following link (provided by the author): http://docs.google.com/Doc?docid=0Af9X2S6wi4I7ZGd4ZGttNWhfMzdja3h2ZDdneg&hl=en

FULL SCHEMATIC




NC
VFS
V4
V3
V2
V1
PS0
PS1
PS2
VDDIO
CLS
D/C#
CS2
E(RD#)
R/W#(WR#)
CS1#
RES#
CL
D0(SCLK)
D1(SDIN)
D2
D3
D4
D5
D6
D7
BUSY
SYNCM
M/S
VDD
VSSA
SYNCC
SYNCD
TPA
V0
VCP1
C1P
C1N
VCP2
C2P
C2N
VCP3
C3P
C3N
VCP4
C4P
C4N
VCI
VSS
NC
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
VDD
C1
1uF/16V
VDD
TO HOST GPIO
DnC
nCS
nRESET
SCLK
SDIN
BUSY
VDD
C2
1uF/16V
C3
1uF/50V
C4
1uF/50V
C5
0.47uF/50V
C6
1uF/50V
C7
1uF/50V
C8
1uF/16V
C9
2.2uF/50V
C10
1uF/16V
C11
1uF/16V
C12
4.7uF/25V
VDD
C13
1uF/16V
U1

References

[1] PowerFilm Inc. 2009. "PowerFilm" Retrieved Nov 20, 2009, from http://www.powerfilmsolar.com/

[2] Choudhury, Debasish. May 2009. "European record: CIS thin-film solar cell efficiency" Retrieved Nov 20, 2009, from http://globalsolartechnology.com/index.php?option=com_content&task=view&id=3038

[3] Amon, Amelia. May 2009. "The ephemeralization of energy production: Photovoltaics using fabric" Retrieved Nov 20, 2009, from http://fabricarchitecturemag.com/articles/0509_f1_photovoltaics.html

[4] IEEE. Sept 2009. "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)" Retrieved Nov 20, 2009, from http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf

[5] Kent Displays Inc. 2009 "132x64 Cholestric Display Module with LCD Controller" Retrieved Nov1, 2009, from http://www.kentdisplays.com/services/resources/datasheets/25112c_132x64_Datasheet.pdf

[6] Microchip Technology. 2005 "PIC16F685/687/689/690 Data Sheet" Retrieved Nov 1, 2009, from http://ww1.microchip.com/downloads/en/devicedoc/41262A.pdf

[7] Digi International Inc. 2007. "XBee Product Manual v1.xAx-802.15.4 Protocol" Retrieved Nov 1, 2009, from http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Manual.pdf

[8] GE Lighting. 2009. "L34715 – Q20T2.5/12V/CL" Retrieved Nov 20, 2009, from http://genet.gelighting.com/LightProducts/Dispatcher?REQUEST=CONSUMERSPECPAGE&PRODUCTCODE=34715&TABID=2&BreadCrumbValues=Lamps_Halogen_Single-Ended_T3&ModelSelectionFilter=

[9] Fairchild Semiconductor. 2001. "NPN General Purpose Amplifier" Retrieved Nov 20, 2009, from http://www.fairchildsemi.com/ds/2N/2N3904.pdf

[10] Chau, Austin. May 2008. "Radish- Indoor Solar-Powered Calendar Display" Retrieved Nov 1, 2009, from http://code.google.com/apis/gdata/articles/radish.html

[11] Parhi, Keshab K. VLSI Digital Signal Processing Systems: Design and Implementation. Hoboken:Wiley, 1999.

[12] Razavi, Behzad Design of Analog CMOS Integrated Circuits. New York:McGraw-Hill, 2000.

Vita

Andrew Edward Hocker attended Cornell University as an Undergraduate and obtained a Bachelor of Science in Electrical and Computer Engineering graduating Cum Laude. He has since graduating worked at Advanced Micro Devices in Austin, Texas working in the Debug and Fault isolation, making high performance microprocessors. He has a passion for learning and has tinkered with electronics and electrical devices all his life.

Permanent address: 5800 Brodie Lane Apt 837, Austin, TX 78745
e-mail: ahocker@gmail.com

This report was typed by Andrew Edward Hocker