ECE 3334 Group 4
Week 4 Presentation

Solar Panel Powered Microgrid and Monitoring Station

Kenneth Cody

Computer Engineer

Justin Price

Computer Engineer

Juan Torres

Electrical Engineer

Jared Tulio

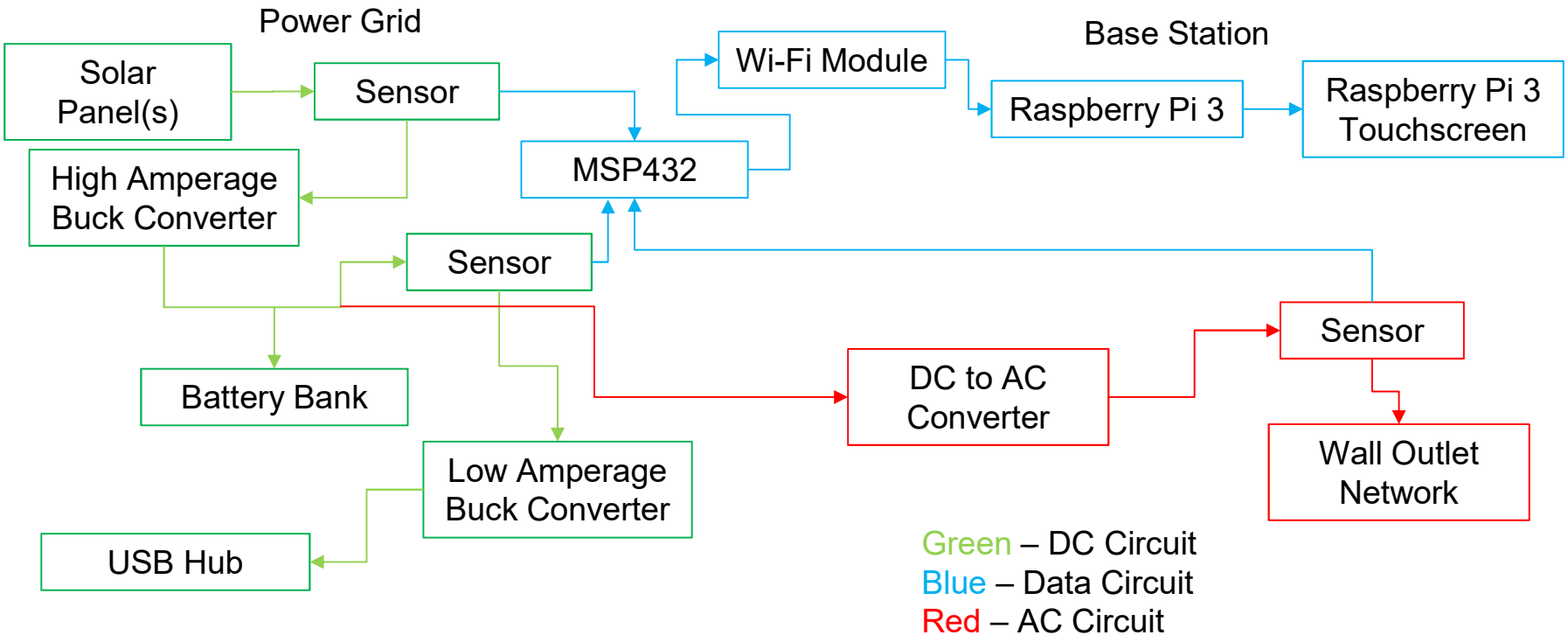
Electrical Engineer – Team Lead

February 14, 2018

System Description

- The microgrid will use a solar panel to charge a battery while the different sections of the grid are monitored using sensors.
- System data is set via Xbee Wi-Fi Modules to a base station built using a Raspberry Pi where data about the system is displayed on a touchscreen.

Hardware Flowchart



Jared Tulio

Division of Labor

Base Station Software and UI: Kenneth

Micro-grid Sensors and Communication: Justin

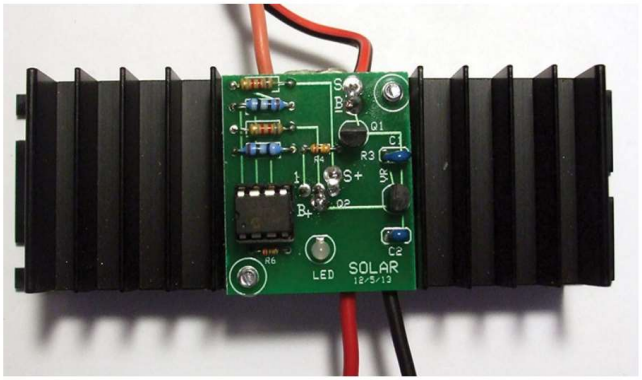
Battery and Power: Juan

Power Conversion and Grid Construction: Jared

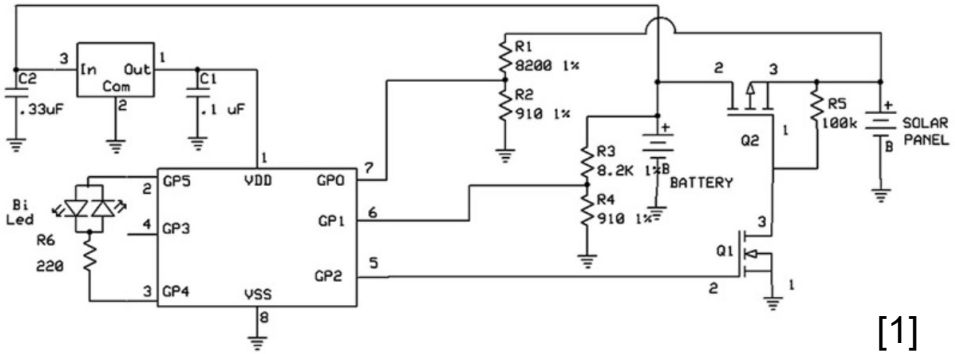
Deliverables for Previous Week

- Work on ZigBee - Complete
- Work on Kivy Application Code – Complete
- Research Power Controller System – Complete
- Research Ac to DC Converters - Complete

Charge Controller



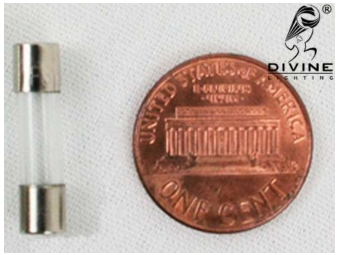
Features:
-Charges lead-acid batteries at 14.25V
-Handles up to 20A
-Disconnects panel from battery when not charging



[1]

Fuse

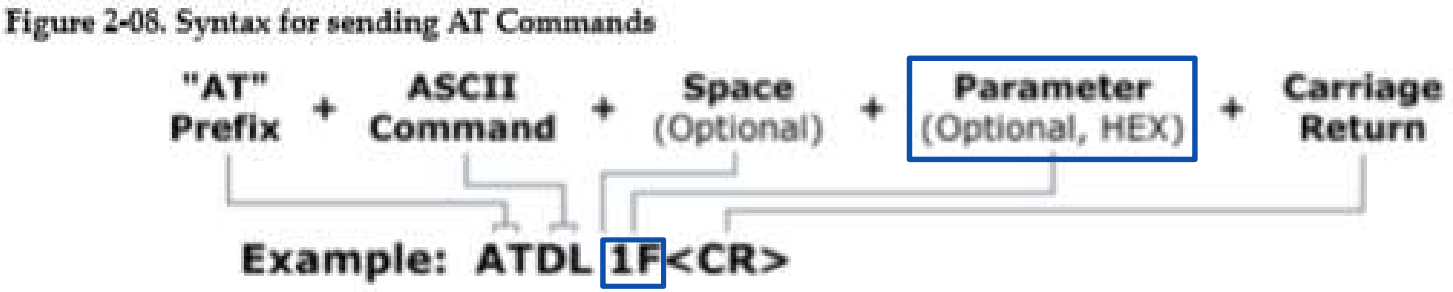
Divine Lighting GMA 10A Fast-Blow Fuse 10 Amp 250v



- Diameter : 5mm (~3/16 in)
- Length : 20mm (~3/4 in)
- Rating : 10A 250V
- Body Material : Glass
- Fuse Type : Fast Blow (Fast Burn)

[2]

ZigBee AT Command Structure



Command Structure [3]

- Same for AT and API command types
- Read or write depends on Parameter field (blue square)

Relevant ASCII commands

- WR – write to non-volatile memory
- CH – set/ read the channel number
- DH- set/read upper 32 bits of destination address
- DL – set/read lower 32 bits of destination address
- EE – enable/disable AES encryption
- KY – set AES encryption key

Use of Zigbee Modules

- Used in the default network configuration: Transparent mode, 64 bit address, unicast message type [1]
- Will change the default channel and network address
- Will change the destination address on the weather station's module to communicate between MSP's
- All MSP modules will always send to the weather station
- Will not use the API command type

How Solar Power Works

- The sun shines on the solar panels generating DC electricity
- The DC electricity is fed into a solar inverter that converts it to 240V 50Hz AC electricity.
- The 230V AC electricity is used to power appliances.
- Surplus electricity is fed back into the main grid.

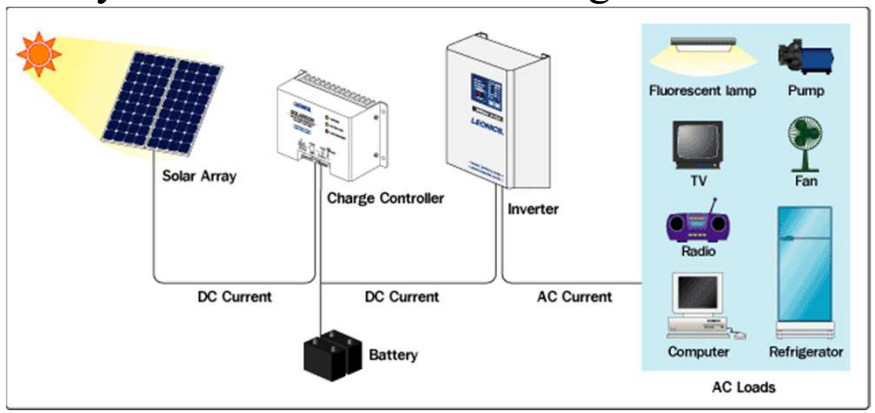


Figure 1: Block Diagram Of Solar Grid[4]

DC to AC Inverter

- The focus of a DC to AC inverter or DAC is to transform DC power to AC power similar to that of a wall outlet
- There are two forms of a DAC's output: a modified sine wave and a pure sine wave.

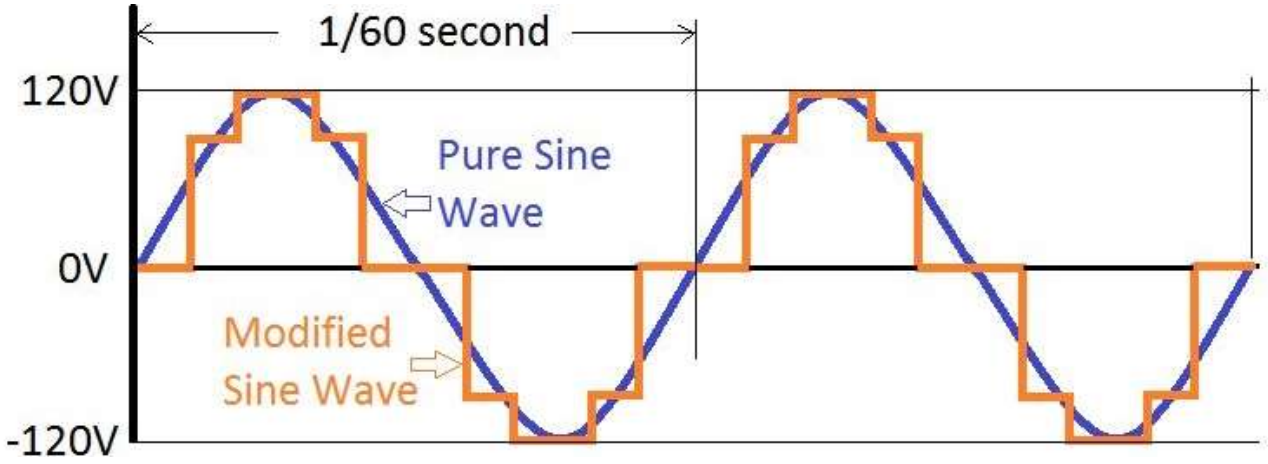


Figure 2: Pure Sine Wave vs Modified Sine Wave [5]

Convert DC to AC

- In order to make a solar inverter a the CD4047 Monostable/Astable Multivibrator and MOSFET IRFZ44 were utilized.
- The CD4047 generates a wave of 50Hz is a simple DC to AC inverter mostly only to be used for lighting devices for the wave is a square wave.

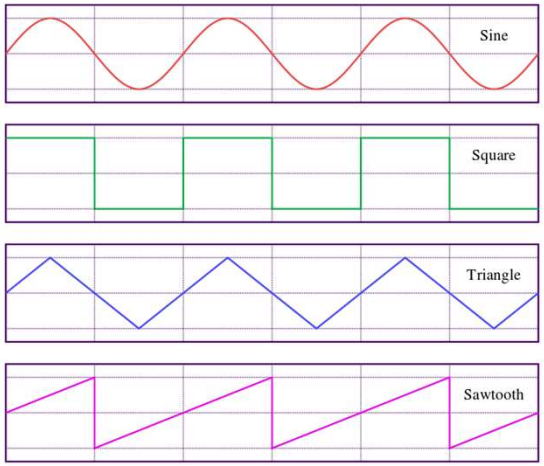


Figure 3: Wave Types [6]

Convert DC to AC

- The MOSFET IRFZ44 provides high current to drive step-up transformer, so power is available in addition to the high voltage transformer. The power MOSFETs are connected in Push Pull configuration The MOSFETs will switch according to the pulse from CD4047. Thus an AC voltage is transferred to the transformer; it is stepped up to 230V. The transformer used here is an ordinary step down transformer which is connected in inverted manner. This circuit uses 12V input (12V battery) to out 220V 50HZ.

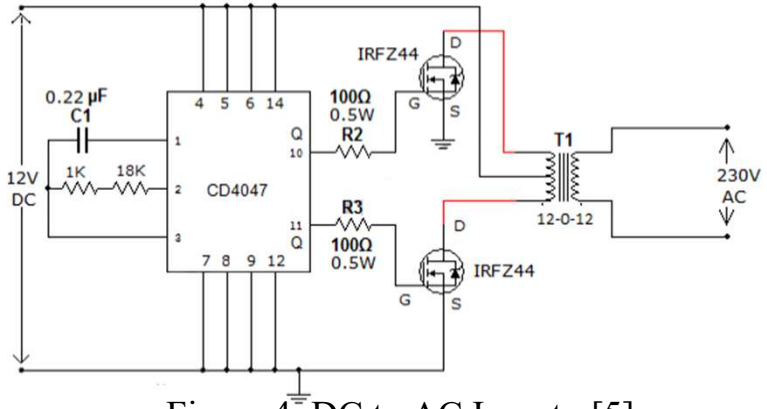


Figure 4: DC to AC Inverter[5]

Information on the CD4047

CD4047BM/CD4047BC Low Power Monostable/Astable Multivibrator

General Description

CD4047B is capable of operating in either the monostable or astable mode. It requires an external capacitor (between pins 1 and 3) and an external resistor (between pins 2 and 3) to determine the output pulse width in the monostable mode, and the output frequency in the astable mode.

Astable operation is enabled by a high level on the astable input or low level on the $\overline{\text{astable}}$ input. The output frequency (at 50% duty cycle) at Q and $\overline{Q}$ outputs is determined by the timing components. A frequency twice that of Q is available at the Oscillator Output; a 50% duty cycle is not guaranteed.

Monostable operation is obtained when the device is triggered by low-to-high transition at + trigger input or high-to-low transition at − trigger input. The device can be retriggered by applying a simultaneous low-to-high transition to both the + trigger and retrigger inputs.

A high level on Reset input resets the outputs Q to low, $\overline{Q}$ to high.

Features

■ Wide supply voltage range	3.0V to 15V
■ High noise immunity	0.45 V_{DD} (typ.)
■ Low power TTL compatibility	Fan out of 2 driving 74L or 1 driving 74LS

SPECIAL FEATURES

- Low power consumption: special CMOS oscillator configuration
- Monostable (one-shot) or astable (free-running) operation
- True and complemented buffered outputs

MONOSTABLE MULTIVIBRATOR FEATURES

- Positive- or negative-edge trigger
- Output pulse width independent of trigger pulse duration
- Retriggerable option for pulse width expansion
- Long pulse widths possible using small RC components by means of external counter provision
- Fast recovery time essentially independent of pulse width
- Pulse-width accuracy maintained at duty cycles approaching 100%

ASTABLE MULTIVIBRATOR FEATURES

- Free-running or gatable operating modes
- 50% duty cycle
- Oscillator output available
- Good astable frequency stability

typical frequency deviation	± 2% + 0.03%/°C @ 100 kHz ± 0.5% + 0.015%/°C @ 10 kHz (circuits trimmed to frequency V_{DD} = 10V ± 10%)

Applications

- Frequency discriminators
- Timing circuits
- Time-delay applications
- Envelope detection
- Frequency multiplication
- Frequency division

Figure 5: Description [6]

Block and Connection Diagrams

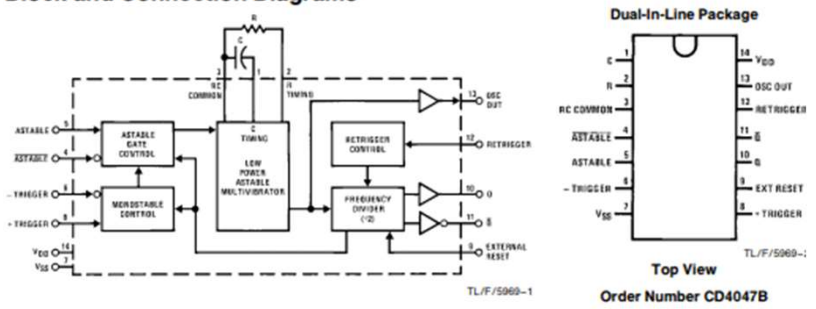


Figure 6: CD4047 Pinout[5]

Base Station: Raspberry Pi 3 Software

```
import kivy
kivy.require('1.10.1')
from kivy.
from kivy.app import App
from kivy.lang import Builder
from kivy.core.window import Window
from kivy.core.text import Label as CoreLabel
from kivy.uix.button import Button
from kivy.graphics import Rectangle, RenderContext, Color

class application(App)
    def build(self)
        # imports system image and sets as background of upper box
        wimg = Image(source='Sys.png',x=240,y=400)
        sysImage=Texture.create(wimg)

        # Sets up the data display for the bottom box
        sysData = CoreLabel()
        sysData.text = defaultData
```

Base Station: Raspberry Pi 3 Software

```
# Sets up layout and main widgets
layout = BoxLayout(orientation='vertical')
layout.add_widget(sysImage,index=1)
layout.add_widget(sysData,index=0)

# Example buttons that overlay the system image and respont to user touches by changing the data displayed in the bottom box
btn1=Button(background_color=(0,0,0,0), pos(100,340), size(25,25), on_press=self.getData1)
btn2=Button(background_color=(0,0,0,0), pos(270,300), size(25,25), on_press=self.getData2)
btn3=Button(background_color=(0,0,0,0), pos(400,440), size(25,25), on_press=self.getData3)
btn4=Button(background_color=(0,0,0,0), pos(600,400), size(25,25), on_press=self.getData4)
# Resets to showing the overview of the data
resetBtn=Button(text='RESET',pos(750,430),size(50,50), on_press=self.getDefault)
# Adds buttons to the layout
layout.add_widget(btn1)
layout.add_widget(btn2)
layout.add_widget(btn3)
layout.add_widget(btn4)
layout.add_widget(resetBtn)
```

Base Station: Raspberry Pi 3 Software

```
# defines what happens when a button is pressed
def getData1:
    sysData.text=Data1
    sysData.text.fontsize=12
    sysData.text.alignment='center'
    sysData.text.split_str='!'

def getData2:
    sysData.text=Data2
    sysData.text.fontsize=12
    sysData.text.alignment='center'
    sysData.text.split_str='!'

def getData3:
    sysData.text=Data3
    sysData.text.fontsize=12
    sysData.text.alignment='center'
    sysData.text.split_str='!'

def getData4:
    sysData.text=Data4
    sysData.text.fontsize=12
    sysData.text.alignment='center'
    sysData.text.split_str='!'

def getDefault:
    sysData.text=defaultData
    sysData.text.fontsize=12
    sysData.text.alignment='center'
    sysData.text.split_str='!'
```

This code is for testing, the actual data will be received from the Zigbee Wi-Fi over UART to the Raspberry Pi and then parsed so that the correct data is stored in the variables that will be displayed when the user presses a button on the screen.

Total Budget

Lab 1 - Group 2	*Running Total*			*Total Estimate*			**Start Date**	2/1/2018
Direct Labor:							**Today**	2/22/2018
Category or individual:	*Rate/Hr*	*Hrs*		*Rate/Hr*	*Hrs*			
Kenneth	18	32	$576.00	18	215	$3,870.00	**End Date**	5/5/2018
Jared	18	26	$468.00	18	215	$3,870.00	**Days Past**	21
Justin	18	32	$576.00	18	215	$3,870.00	**Total Days**	93
Juan	18	26	$468.00	18	215	$3,870.00	**Days Left**	72
DL Subtotal (DL)		Subtotal:	$2,088.00		Subtotal:	$15,480.00		
Labor Overhead	*rate:*	100%	$2,088.00	*rate:*	100%	$15,480.00		
Total Direct Labor (TDL)			$4,176.00			$30,960.00		
Contract Labor:								
Lab 1 Help	$15	0	$0.00	$15	0	$0.00		
Lab 2 Help	$18	0	$0.00	$18	0	$0.00		
Lab 3 Help	$18	0	$0.00	$18	10	$180.00		
Lab 4 Help	$18	0	$0.00	$18	15	$270.00		
Lab 5 Help	$25	0	$0.00	$25	10	$250.00		
Tutors	$40	0	$0.00	$40	15	$600.00		
Lab Assistants	$40	0	$0.00	$40	15	$600.00		
Woodcock	$100	0	$0.00	$100	5	$500.00		
Prof. Ray	$200	0	$0.00	$200	20	$4,000.00		
Total Contract Labor (TCL)			$0.00			$6,400.00		
Direct Material Costs:			$0.00			$650.00		
(from Material Cost worksheet)								
Total Direct Material Costs: (TDM)			$0.00			$650.00		
Equipment Rental Costs:	Value	Rental Rate		Value	Rental Rate		Date begin	Date end (or today)
Oscilloscope	$5,300.00	0.20%	$222.60	$5,300.00	0.20%	$985.80	2/1/2018	5/5/2018
Function Generator	$500.00	0.20%	$21.00	$500.00	0.20%	$93.00	2/1/2018	5/5/2018
DMM	$958.00	0.20%	$40.24	$958.00	0.20%	$178.19	2/1/2018	5/5/2018
Power Supply	$1,700.00	0.20%	$71.40	$1,700.00	0.20%	$316.20	2/1/2018	5/5/2018
Soldering Station	$100	0.20%	$4.20	$100.00	0.20%	$18.60	2/1/2018	5/5/2018
Total Rental Costs: (TRM)			$355.24			$1,573.19		
Total TDL+TCL+TDM+TRM			$4,531.24			$39,583.19		
Business overhead		100%	$4,531.24		100%	$39,583.19		
Total Cost:		**Current**	$9,062.47		**Estimate**	$79,166.38		

Total Budget Cont'd

Name	Cost	Quantity	Website	Notes	Purchase Date	Total		TOTAL
ESP8266 WiFi Module	$6.95	2	SparkFun	NA		$13.90		$538.38
100W-Solar Panel - Grape Solar	$97.00	1	Home Depot	NA		$97.00		
35Ah 12V Lead Acid Battery	$64.99	1	Amazon	NA		$64.99		
7Ah 12V Lead Acid Battery	$28.49	1	Amazon	Pack of 2		$28.49		
Buck Converter - High Amperage	$26.00	2	Amazon	NA		$52.00		
Buck Converter - Low Amperage	$8.00	4	Amazon	NA		$32.00		
Rasberry Pi 3	$35.00	1	Allied Electronics	NA		$35.00		
Rasberry Pi Touch Screen	$70.00	1	Allied Electronics	NA		$70.00		
Rechargeable Battery	$15.00	1	Amazon	NA		$15.00		
Current Sensor	$20.00	3	Mouser	NA		$60.00		
Power Inverter	$70.00	1	Amazon	NA		$70.00		

Gantt Chart

Group 2	% done	Justin	Jared	Juan	Kenneth	
Wifi Network	**12.5**					
Router Setup	50	4				4
Setup Wifi Module	0					0
Implementation	0					0
Troubleshoot	0					0
Working Sensors	**0**					
Research	0	4				4
Configure with controller	0					0
Integration	0					0
Troubleshooting	0					0
DC Power Grid	**14.286**					
Research	100		4	2		6
Order Parts	0					0
Construct Schematic	0					0
Test Schematic	0					0
Construct Actual Circuit	0					0
Troubleshoot	0					0
Construct Finalized Circuit	0					0
AC Power Grid	**0**					
Research	0					0
Order Parts	0					0
Construct Schematic	0					0
Test Schematic	0					0
Construct Actual Circuit	0					0
Troubleshoot	0					0
Construct Finalized Circuit	0					0
Base Station	**9.0909**					
Research	50		2			
Order Parts	0					
Construct Schematic	0					
Test Schematic	0					
Construct Actual Circuit	0					
Troubleshoot	0					
Construct Finalized Circuit	0					
Research Code	50				4	
Implement Preliminary Solutions	0					
Troubleshoot	0					
Integrate with hardware	0					
Compile Entire Project	**0**					
Troubleshoot all software	0					0
Compile all circuits together	0					0
Troubleshoot and resolve all issues	0					0
		Total	**Total**	**Total**	**Total**	
		8	6	2	4	

Deliverables for Next Week

- Continue developing Kivy application – Kenneth
- Build test charge controller – Jared
- Create and test ADC code - Justin

References

1. http://www.nutsvolts.com/index.php?/magazine/article/March2014_Newton
2. https://www.amazon.com/Divine-Lighting-GMA-10A-Fast-Blow/dp/B076DPPRKS
3. https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf
4. http://www.uni-kl.de/elektronik-lager/418072
5. https://www.energymatters.com.au/components/micro-string-central-inverters
6. http://www.theorycircuit.com/pv-solar-inverter-circuit-diagram/
7. http://www.efxkits.us/12v-dc-to-120v-ac-inverter-circuit/

Questions?