Phasor Measurement Unit

Gaurav Shilpakar

Texas Tech University

December 2019

Acknowledgment

It is a very humbling experience to be working on a project under the guidance of Dr. Stephen Bayne. I am very pleased to have been a part of this project. I would love to extend my sincere gratitude towards everybody who was a part of this project and to those who guided us throughout the process. I want to thank our instructor, Dr. Bayne, for giving us an opportunity to excel in our class learned knowledge on an extravagant physical project.

Abstract

The Phasor Measurement Unit (PMU) is a device used to estimate the magnitude and phase of an electrical oscillating wave, AC source in this case. In this project, it is used to measure the voltage, current, and phase of the AC source. Multiple numbers of such units are used in a power grid to monitor any outages or faults.

This paper deals with the front-end development works that went into creating a Phasor Measurement System. This site is mainly composed of three major subsystems, data inquiry, data processing, and data output. For this aspect technologies of LabVIEW, HTML, CSS as well as the assistance of AJAX, PHP and JavaScript have been utilized.

Table of Contents

List of Figures

Introduction

The objective of this project is to build a two-phase measurement unit. Such a unit is used to measure the voltage, frequency, phase and current of an AC source. The data obtained from such a unit is used during the events of power outage or failures by analyzing any faults seen in through the PMU. The system consists of four major subsystems, rectifier, phase-locked loop, data collector and user interface.

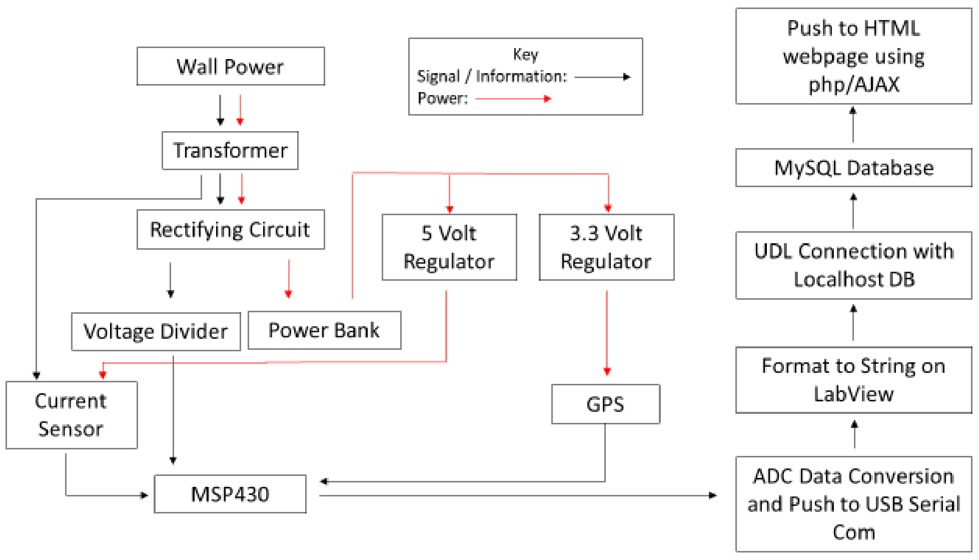


Figure 1 System Block Diagram

This paper deals with the user interface portion of the project. It involves data acquisition from the microcontroller, analyzation, and result calculation plus representation on LabVIEW and a Webpage.

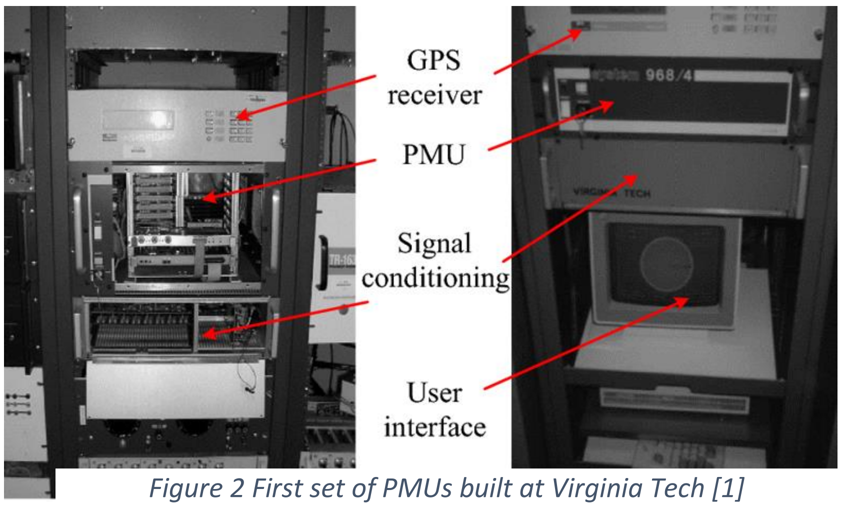


Figure 2 First set of PMUs built at Virginia Tech [1]

Technologies Used

LabVIEW

LabVIEW is a graphical programming structure that is used to visualize elements of an application involving hardware configuration, data measurement and debugging. This application is used as a communication hub between the microcontroller and the MySQL database. It acts as the data acquisition subsystem for the PMU.

MySQL Database

MySQL is an open-source database management system. It is used by major tech companies including Facebook, YouTube, Google, PayPal to keep their data managed and accessible upon request. This application stores all the data acquired from the microcontroller with the help of LabVIEW. The data is separated based on the information i.e. if it is data involving voltage, current, phase or frequency. Then such collection of data is pushed to a Webpage.

HTML

HyperText Markup Language is a computer text processing for documents that are designed to be displayed on the internet. HTML is used to form the fundamental base of the webpage which is accessed by a user for PMU data study.

PHP

Hypertext Preprocessor is another text processing mainly created for web development. It can be easily embedded in an HTML Document to perform actions like normal programming language functions. This platform is used to create the connection between the SQL Database and the webpage. It is also used to perform relevant calculations about the pure sinusoid and phase which is obtained from the PMU data.

AJAX

Asynchronous JavaScript and XML is a programming concept which is used to send and receive data from a web server asynchronously (in the background) without interfering with the present data. Such an approach updates parts of a web page, without reloading the whole page. This technique is used to obtain the real-time capabilities of the PMU by refreshing the contents at the sampling rate.

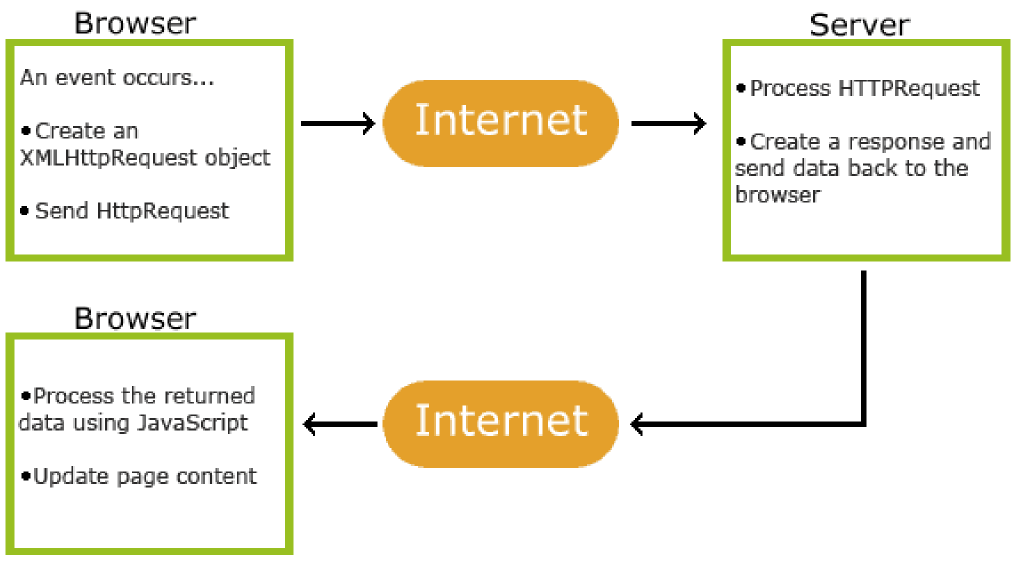

Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

Figure 3 How AJAX Works? [2]

LabVIEW Programming

With an initial condition stated that the microcontroller is connected to the computer and the LabVIEW program is running, the program itself follows the given flowchart.

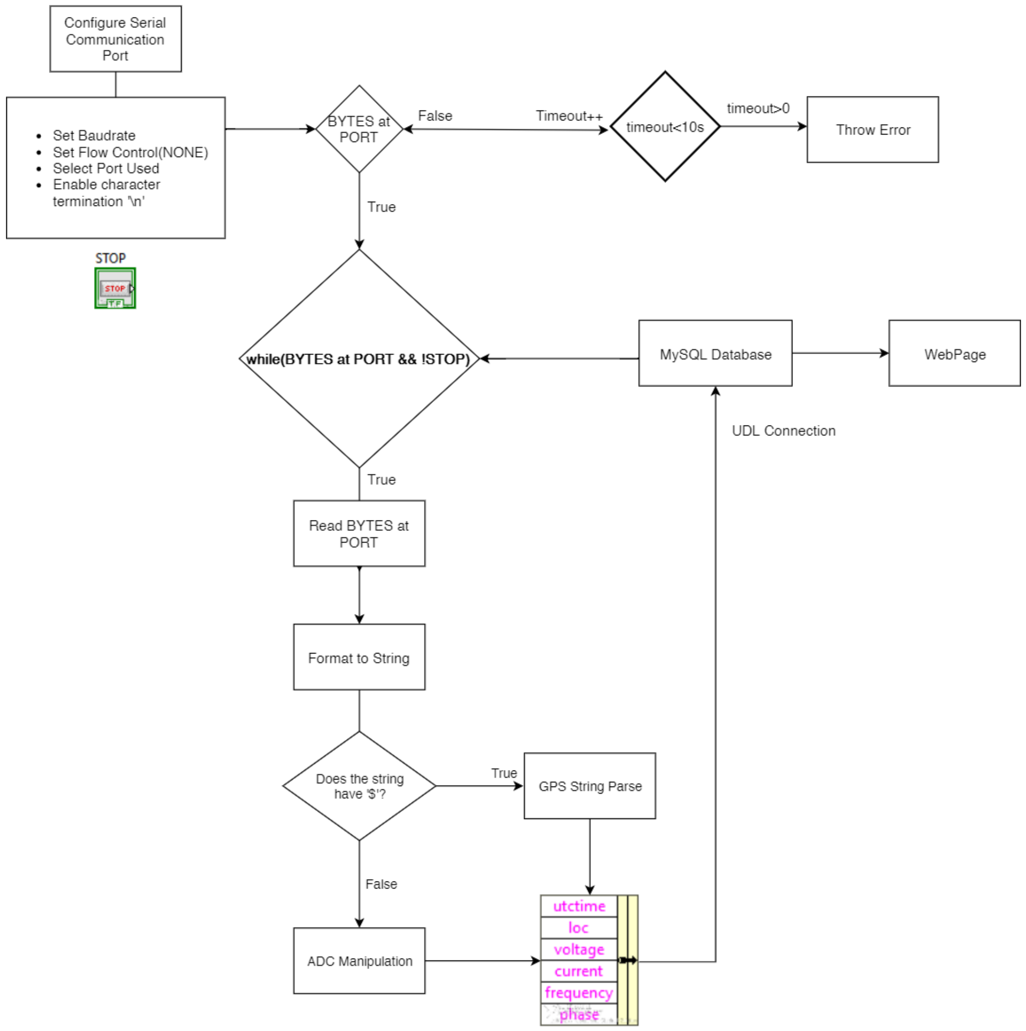
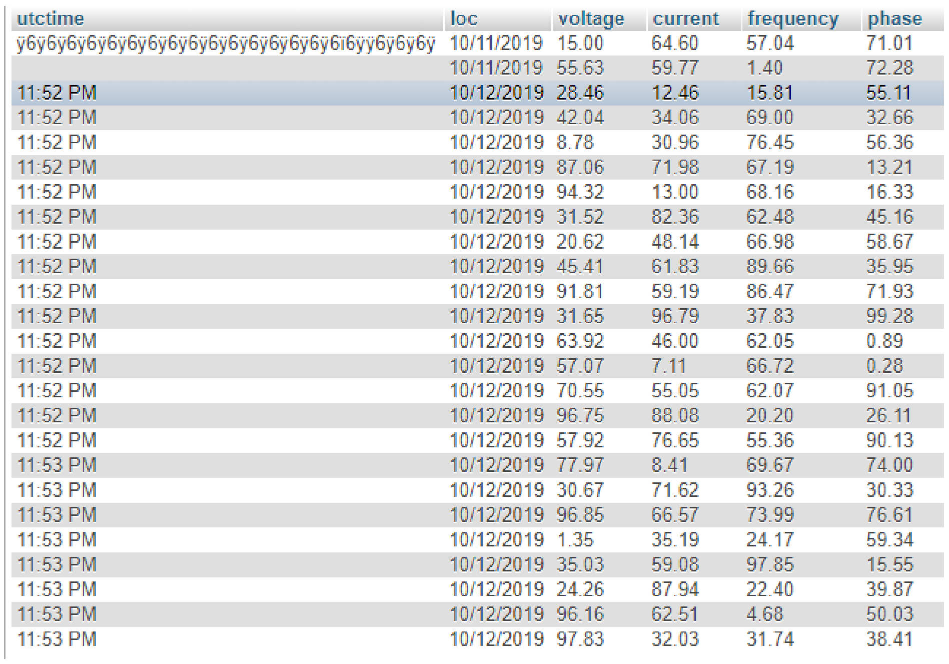


Figure 4 LabVIEW Flowchart

First, the serial communication port on the computer is configured using the VISA Instrument Drivers, native to LabVIEW, to recognize the device(microcontroller). It sets the port that is being used, sets the baud rate to highest possible of 460,800bps, and then sets the flow control to NONE, which is to prevent a fast sender from overwhelming a slow receiver. Then, it checks if there is any data being transmitted from the uC to the computer by ‘BYTES at PORT.’ If ‘false,’ a timeout is incremented until 10s. The ‘BYTES at PORT' is continuously checked during this period. If the timeout exceeds 10s, an exception is thrown, and the program stops.

If the BYTES at PORT is 'true' as well as an interrupt STOP button hasn't been pressed, the program continues. It reads the bytes of data being transferred to the USB Serial Port and then formats each block of data separated by a '\r' delimiter into a string. The respective data is sent into the respective data bins. Necessary type conversions of string to double are made within LabVIEW to form a waveform chart. After that, the bundled data is pushed to a locally hosted MySQL Database. The block sends relevant information to the webpage for the user interface and continues to the loop. The STOP button acts as an interrupt which can be pressed at any time during the process to halt the whole system. The program exits itself only when the STOP has been pressed or the timeout has passed 10s.

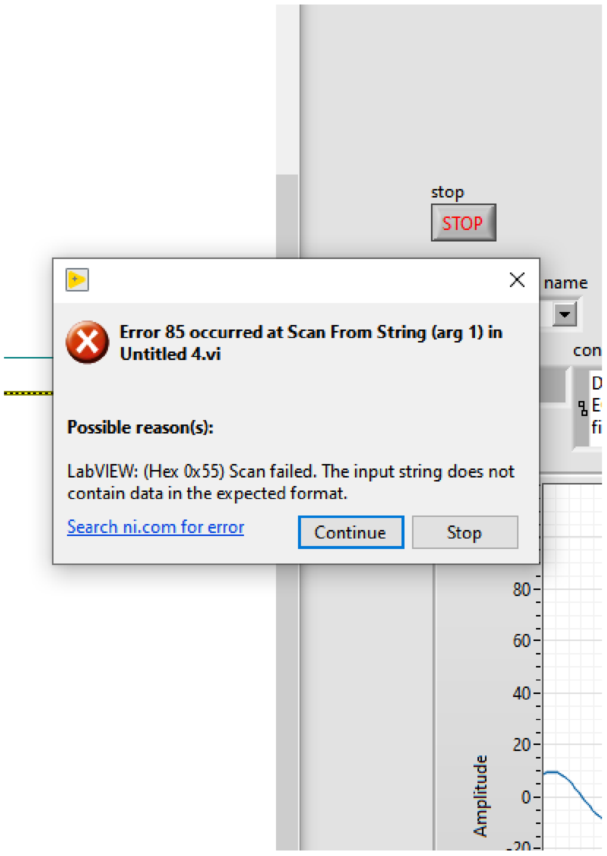
Initially, the MSP430 was used in UART (Universal Asynchronous Receiver-Transmitter) mode for data transfer. The uC was set at 115200bps baud rate which resulted in numerous data failures. The TI Datasheet claims an RX/TX error rate of about 2% whilst using the microcontroller in UART Mode [3].



utctime	loc	voltage	current	frequency	phase
ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ÿ6ï6ÿÿ6ÿ6ÿ6ÿ	10/11/2019	15.00	64.60	57.04	71.01
	10/11/2019	55.63	59.77	1.40	72.28
11:52 PM	10/12/2019	28.46	12.46	15.81	55.11
11:52 PM	10/12/2019	42.04	34.06	69.00	32.66
11:52 PM	10/12/2019	8.78	30.96	76.45	56.36
11:52 PM	10/12/2019	87.06	71.98	67.19	13.21
11:52 PM	10/12/2019	94.32	13.00	68.16	16.33
11:52 PM	10/12/2019	31.52	82.36	62.48	45.16
11:52 PM	10/12/2019	20.62	48.14	66.98	58.67
11:52 PM	10/12/2019	45.41	61.83	89.66	35.95
11:52 PM	10/12/2019	91.81	59.19	86.47	71.93
11:52 PM	10/12/2019	31.65	96.79	37.83	99.28
11:52 PM	10/12/2019	63.92	46.00	62.05	0.89
11:52 PM	10/12/2019	57.07	7.11	66.72	0.28
11:52 PM	10/12/2019	70.55	55.05	62.07	91.05
11:52 PM	10/12/2019	96.75	88.08	20.20	26.11
11:52 PM	10/12/2019	57.92	76.65	55.36	90.13
11:53 PM	10/12/2019	77.97	8.41	69.67	74.00
11:53 PM	10/12/2019	30.67	71.62	93.26	30.33
11:53 PM	10/12/2019	96.85	66.57	73.99	76.61
11:53 PM	10/12/2019	1.35	35.19	24.17	59.34
11:53 PM	10/12/2019	35.03	59.08	97.85	15.55
11:53 PM	10/12/2019	24.26	87.94	22.40	39.87
11:53 PM	10/12/2019	96.16	62.51	4.68	50.03
11:53 PM	10/12/2019	97.83	32.03	31.74	38.41

Figure 5 Errors Caused due to UART Protocol

Instead, it was configured to run via USB Communication Protocol. This gave us USB 2.0 speeds for data transfer. Also, no issues were noticed in SQL Database about the data packets being transferred.


Error 85 occurred at Scan From String (arg 1) in Untitled 4.vi

Possible reason(s):

LabVIEW: (Hex 0x55) Scan failed. The input string does not contain data in the expected format.

Figure 6 LabVIEW String Parsing Issues

After the possible data failures in transmitting the data from the uC to the computer through the USB Bus was solved, another issue persisted. The received string of data LabVIEW was not parsed correcting in LabVIEW. LabVIEW was not able to recognize the string being sent to the computer and consistently threw an exception. To solve this issue the initial string parsing block for LabVIEW was redone which resulted in a flowchart as shown in fig 7. For the correction, a character checker loop was added after reading the bytes at the port. The bytes once formatted into string entered this loop. It checked if the input string consisted of the char identifier '$' or had numbers from '0-9.' The former denoted the string data received from the GPS Module while the later denoted the ADC Data about the sinusoid. After this detection, the respective string manipulation was done in their respective blocks of GPS or ADC Parsing. If none of the mentioned identifiers were detected, the received data was deemed as garbage value and then continued to read the newer lines of string. This implementation of the character checker loop fixed all the string issues either caused due to the USB Buffer or LabVIEW's Vis.

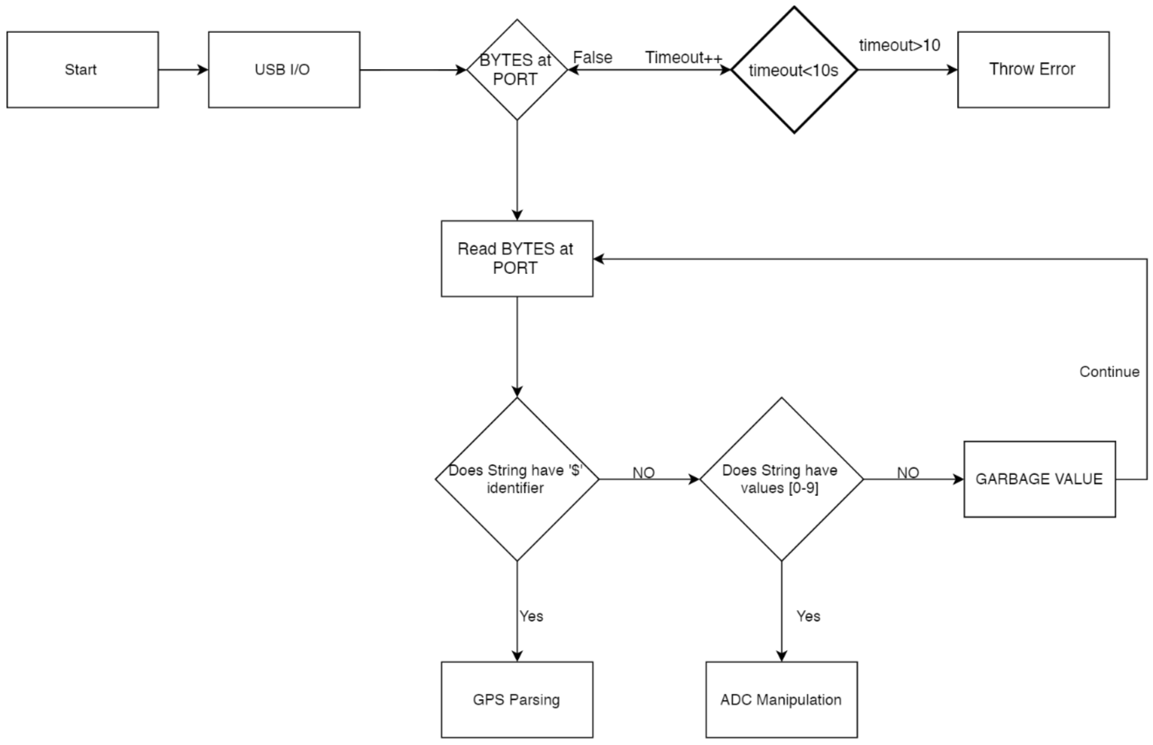


Figure 7 Corrected String Parsing

GPS String Parsing

Once the formatted line of string was identified as GPS data, it entered the GPS Parsing Block. In this block, the received string was multiplexed into six separate blocks: '$' identifier, Latitude, N/S Indicator, Longitude, E/W Indicator and UTC Time.

The $ sign indicated the character checker block mentioned above in string parsing issues. The latitude data was divided by 100 which was added to the remainder after it was divided by 60 to get the actual latitude in the form of 'XXX.YYY.' This was then concatenated with the N/S Indicator to get the latitude string. A similar process was repeated for the longitude data. The UTC Time Block was just divided into three blocks: HH, MM, SS and concatenated with a colon ':' in between them. All these separately concatenated blocks were pushed to the MySQL Database for further processing.

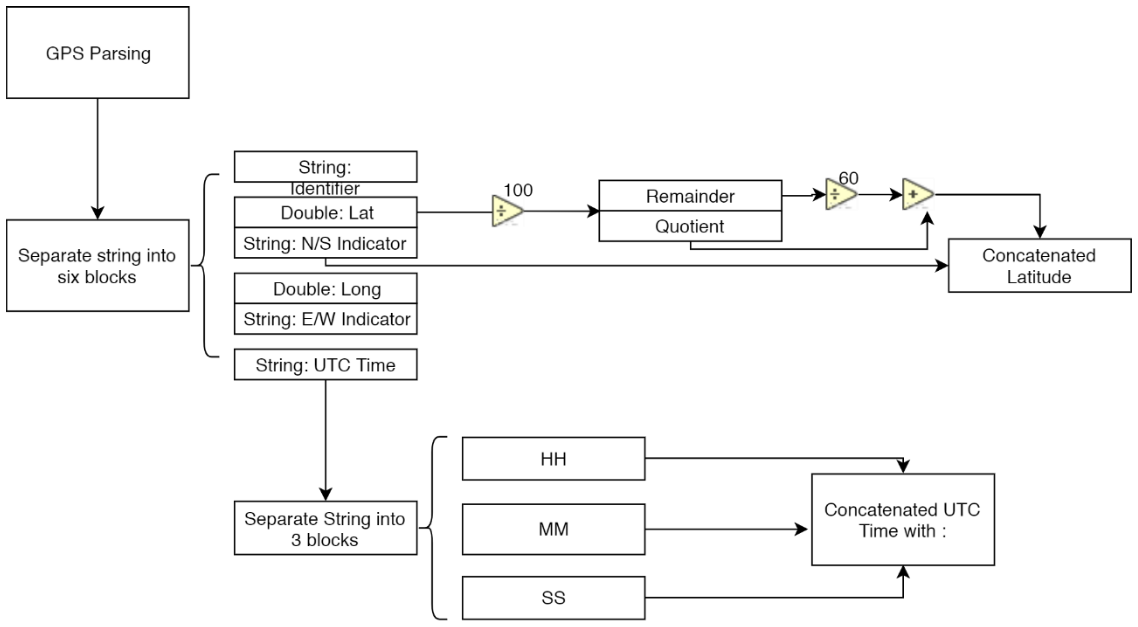


Figure 8 GPS String Parsing Flowchart

Example of GPS Data Manipulation from the uC

"$GNGLL,3333.50657,N,10151.35547,W,013513"

$GNGLL	= Identifier			
3333.50657,N	= Latitude	=	33 deg 33.5067 mins	= 33.5585 deg N
10151.35547,W	= Longitude	=	101deg 51.35547 mins	= 101.85592 deg W
013513	= UTC Time	=	01:35:13	

ADC Data Manipulation

Once the received data was identified having being started by any number from 0-9, it was sent to the ADC Manipulation Block. The Microcontroller collected the instantaneous voltage of the rectified sine wave through two separate ADC Pins as explained in Pranesh’s report. ADC1 and ADC2 collected data about the positive and negative half of the rectified wave respectively. An example of the data being received is shown. These instantaneous data points were initially combined using the given formula as derived by Samuel.

$$Instantaneous\ Voltage = 16.35(\left(\frac{ADC_1 * (92k+29k)}{29k} + 320mV\right) - \left(\frac{ADC_2 * (92k+29k)}{29k} + 320mV\right))$$

0.213	0
0.806	0
1.388	0
1.889	0
2.229	0
2.348	0
2.333	0
2.114	0
1.543	0
0.73	0.001
0	0
0	0.219
0	0.821
0	1.643
0	2.043
0	2.334
0	2.324
0	2.004
0	1.466
0	0.873
0	0.247
0	0
0.38	0
0.971	0
1.769	0
2.244	0
2.364	0
2.349	0
2.065	0
1.681	0
1.144	0
0.567	0

Figure 9 32 Samples of ADC Data

This formula posed issues in calculating the RMS voltage of the sine wave once it was decided that 2 different transformers were going to be used. So a newer formula was derived as:

$$\boldsymbol{V_{instantaneous}} = (\boldsymbol{ADC1} - \boldsymbol{ADC2}) * \boldsymbol{73.108}$$

The use of the newer formula resulted in the waveforms as shown in fig 10 which closely represents a sinusoid. Finally, the necessary calculations were done in house inside LabVIEW to find the RMS voltage, frequency, current and the phase. The RMS voltage and frequency were measured using the Tone Measurement module in LabVIEW. This resulted in the calculated RMS and frequency to hover around 120V and 60Hz as expected. The current drawn was calculated by diving the RMS voltage by 10k which represented the resistor being used. This resulted in the maximum current drawn to hover around 12mA. The necessary calculations for the phasor were done separately as discussed below.

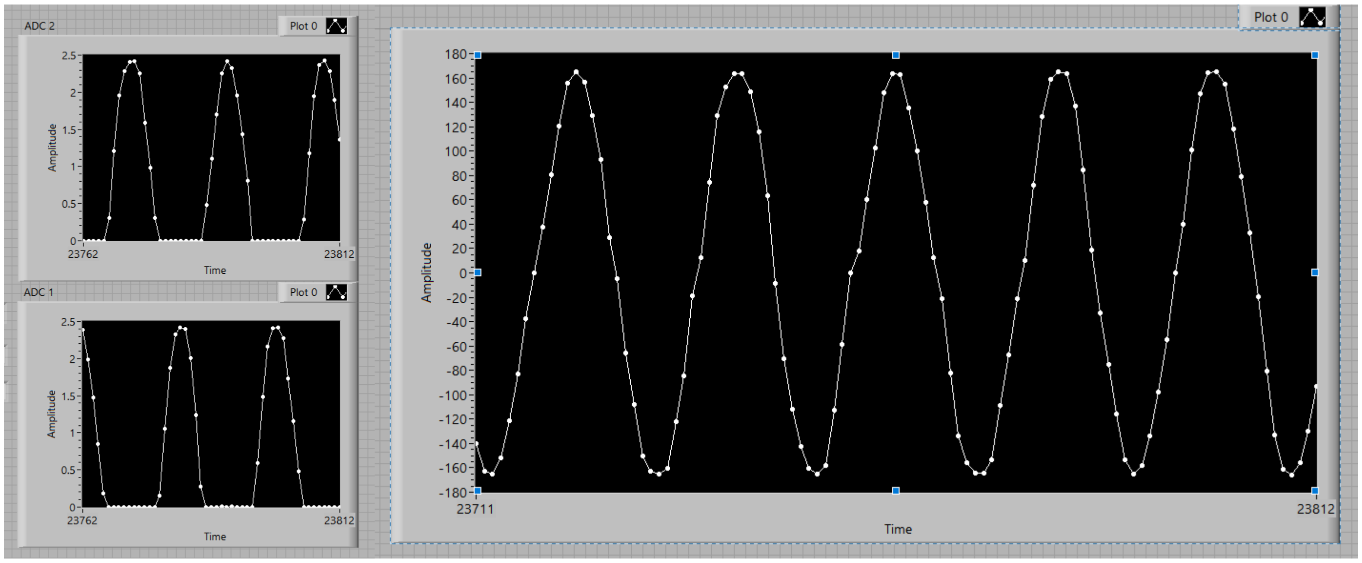
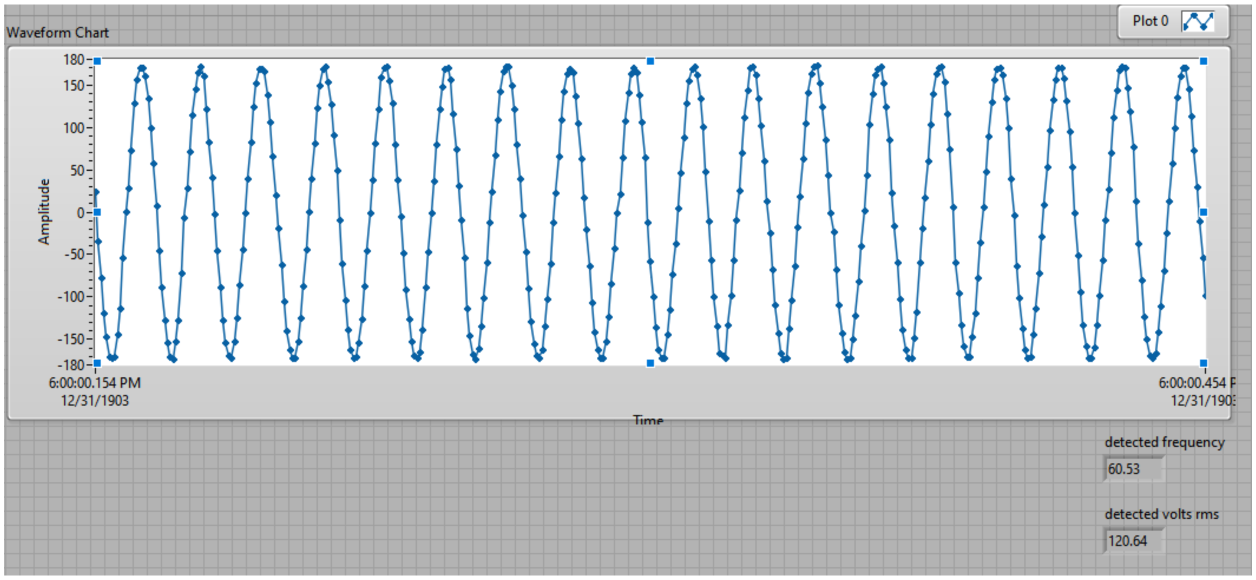


Figure 10 Final Waveform



detected frequency
60.53

detected volts rms
120.64

Figure 11 Results of Tone Measurement

Calculation

Phasor

The phasor diagram has the formula denoted below which were all collected in LabVIEW

. $V = A \sin(2\pi ft+phi)$

- $A = Vrms$
- $phi = phase$
- $f = frequency$

The phasor consists of both real and imaginary parts. Since LabVIEW didn't have a built-in phasor module, the following components for the real and imaginary axes were calculated.

$A_{real} = A*\cos(p)$ where, $p = 2\pi ft+phi$

$A_{imaginary} = A*\sin(p)$

These real and imaginary quantities were calculated separated and then bundled into a data cluster. This was inputted into a draw graph module which was set up to create XY-axis ranging from -130 to 130 on both the axes. The collective data resulted in a rotating phasor based on its frequency and phase.

Thus, these quantities were used to plot the phasor.

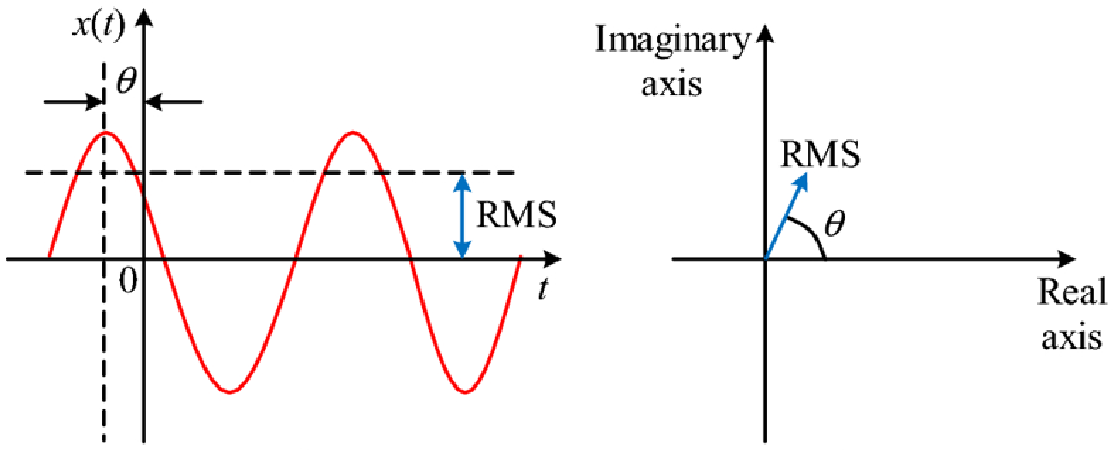
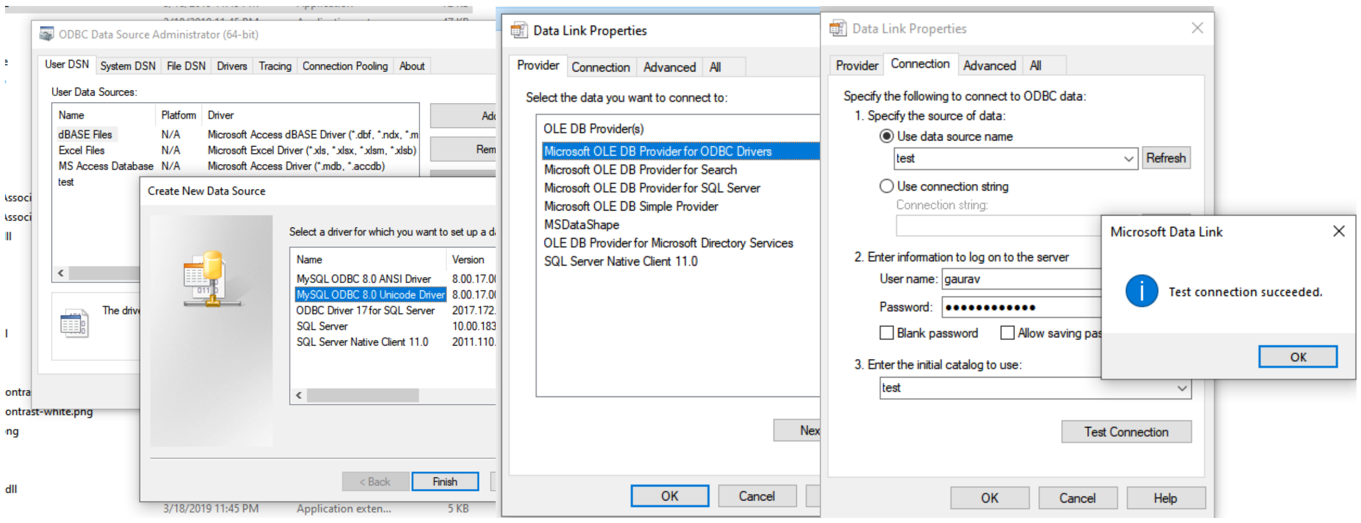


Figure 12 Sinusoid and Phasor

From figure 14, Θ = phase = arctan (Ay/Ax)

MySQL Database Link

One of the assigned tasks for the project was to store all the relevant data to a SQL Database. So, a modular table was created for the two PMUs. Each included columns for data about Maximum Voltage, Maximum Current, Frequency, Phase, and Time. For LabVIEW to push the collected data to the SQL Table, a Universal Data Link file was necessary. This UDL File allows third-party applications, like Labview to push data to a preexisting table without ay issues. The following steps were followed as shown in fig 13 for this process.


ODBC Data Source Administrator (64-bit)

User Data Sources:

Name	Platform	Driver
dBASE Files	N/A	Microsoft Access dBASE Driver (*.dbf, *.ndx, *.m
Excel Files	N/A	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
MS Access Database	N/A	Microsoft Access Driver (*.mdb, *.accdb)
test		

Create New Data Source

Select a driver for which you want to set up a d

Name	Version
MySQL ODBC 8.0 ANSI Driver	8.00.17.0
MySQL ODBC 8.0 Unicode Driver	8.00.17.0
ODBC Driver 17 for SQL Server	2017.172.
SQL Server	10.00.183
SQL Server Native Client 11.0	2011.110.

Data Link Properties

Select the data you want to connect to:

OLE DB Provider(s)
- Microsoft OLE DB Provider for ODBC Drivers
- Microsoft OLE DB Provider for Search
- Microsoft OLE DB Provider for SQL Server
- Microsoft OLE DB Simple Provider
- MSDataShape
- OLE DB Provider for Microsoft Directory Services
- SQL Server Native Client 11.0

Data Link Properties

Specify the following to connect to ODBC data:

1. Specify the source of data:
   - [✓ Use data source name] test
   - Use connection string
2. Enter information to log on to the server
   - User name: gaurav
   - Password: ••••••••••••
3. Enter the initial catalog to use: test

Microsoft Data Link

Test connection succeeded.

Figure 13 Stepwise process for UDL Connection [5]

Finally, DB Tools present as one of LabVIEW Modules were utilized to access and push the previously clustered data to the SQL Table.

Plot

Initially, attempts were made to plot the sinusoid and phasor diagram on the website itself. The server-side page ‘grid.php’ was used to create a canvas for the sinusoid. Axes were set for time and voltage so concurrent; voltage, frequency, and phase could be plotted. Also, the plot received real-time capabilities through the implementation of AJAX, which continuously loaded new data from the SQL Database without halting any other processes on the page. But later, it was realized that this process was very taxing on the computer and there were time constraints as well. So, the plots were limited to LabVIEW.

Pseudocode

```
if (page loaded) {
      wait for an interval of (0.75ms) {
            load (new data)->canvas
      }
```

The delay of 3ms was obtained to accommodate a sample rate of 22 samples per cycle.

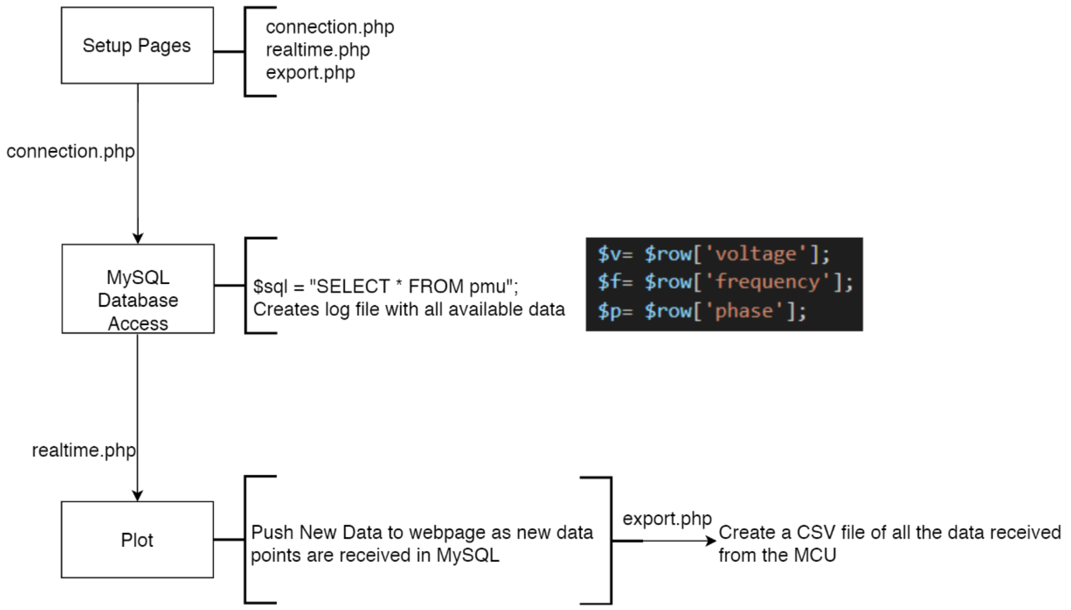
1 cycle = 22*60 samples

1320 samples = 1 second

1 second = 1/1320 second = 0.75mS

Webpage Hosting

The front-end block of the project intended for user interface with the acquired data is divided into five sub-blocks.



```
$sql = "SELECT * FROM pmu";
Creates log file with all available data
```

```
$v= $row['voltage'];
$f= $row['frequency'];
$p= $row['phase'];
```

Push New Data to webpage as new data points are received in MySQL

Figure 14 Webpage Block Diagram

Setup Pages

First, all the client-side page was loaded. This page involved the whole GUI of the interface for a user to study the data being fed to the servers.

In the background process, the server-side pages were load as per the need. These constituted functions intended to perform necessary functions and calculations to present the data in a readable format. The server-side pages included the following:

1. Connection.php
2. Realtime.php
3. Export.php

These server heavy pages work in conjunction with the neat front pages for easy data access and study.

MySQL Database Access

In order to display the necessary data from the PMU resting at an SQL Server, a connection was required. For this, the connection.php page comes into play. It contains necessary information about the server, username, password and the database being used for the project. This access page was used to retrieve available data from the SQL Database and feed it to any other scripts required. For example, export.php made use of this connection to collect all the data and form an easily accessible CSV log. Also, realtime.php spits the data into a readable table.

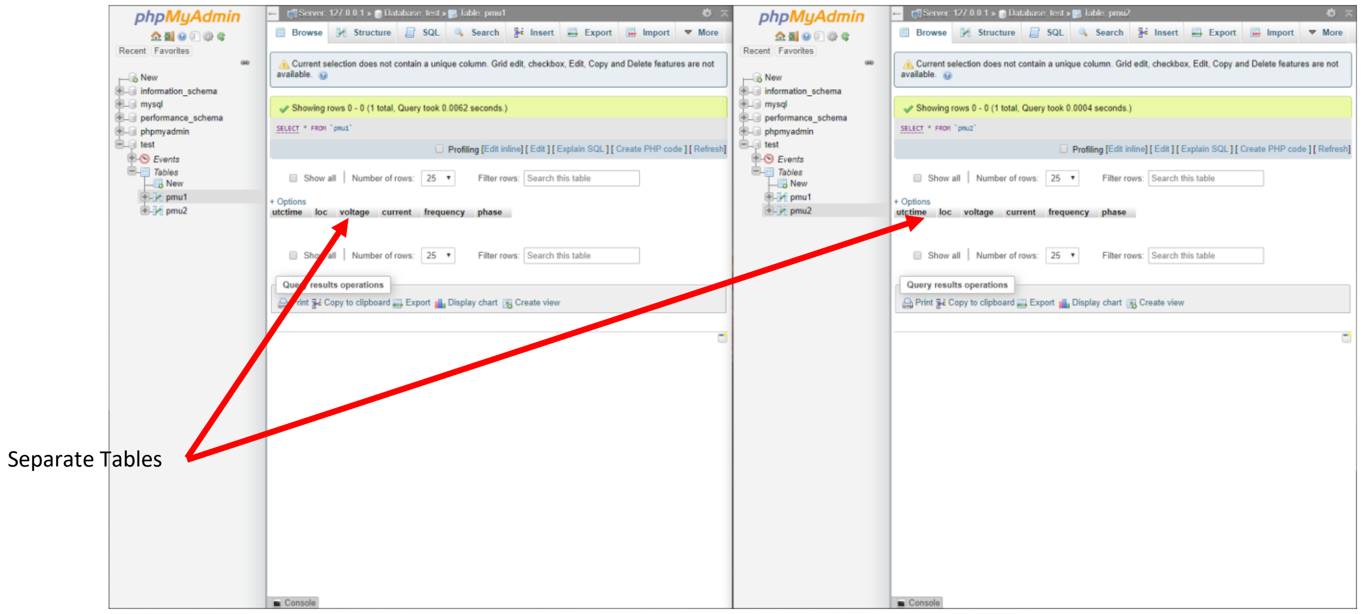
Pseudocode

```
$sql = "SELECT * FROM pmu"; //selects data from the DB

//if there is data at the DB
if($sql>0) {
//set the given variables to the necessary rows of data
            $v= $row['voltage'];
            $f= $row['frequency'];
            $p= $row['phase'];
      }
```

Realtime pushing of data into the webpage through an AJAX script that runs on the homepage at its set time period.

Final SQL Table



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0062 seconds.)

```
SELECT * FROM `pmu1`
```

utctime	loc	voltage	current	frequency	phase

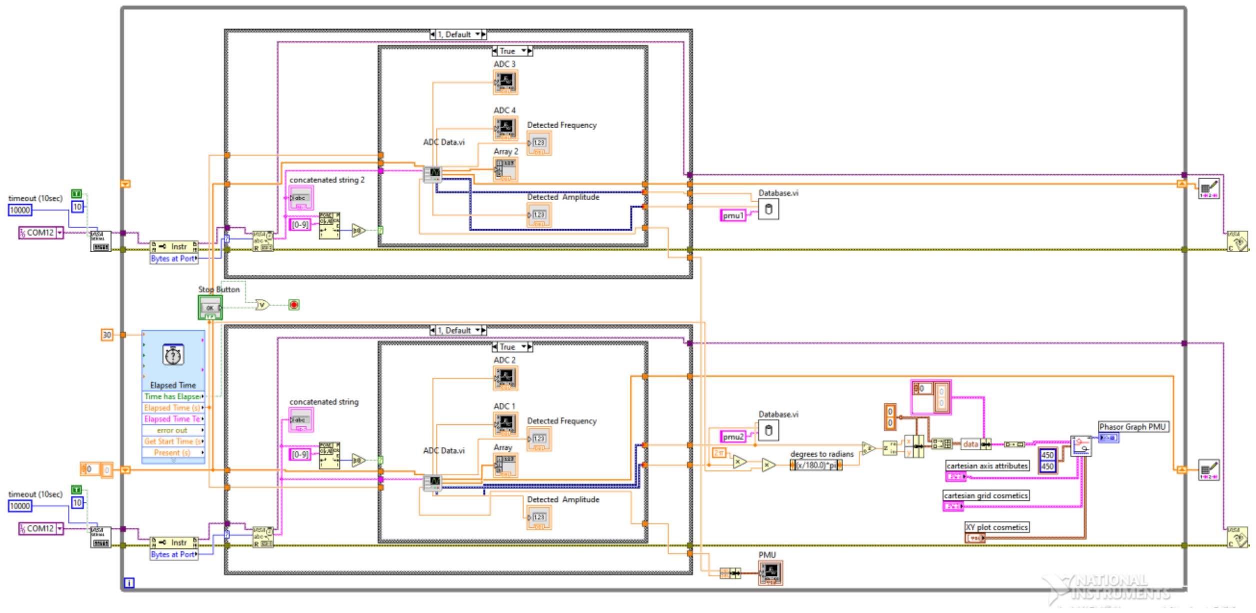
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

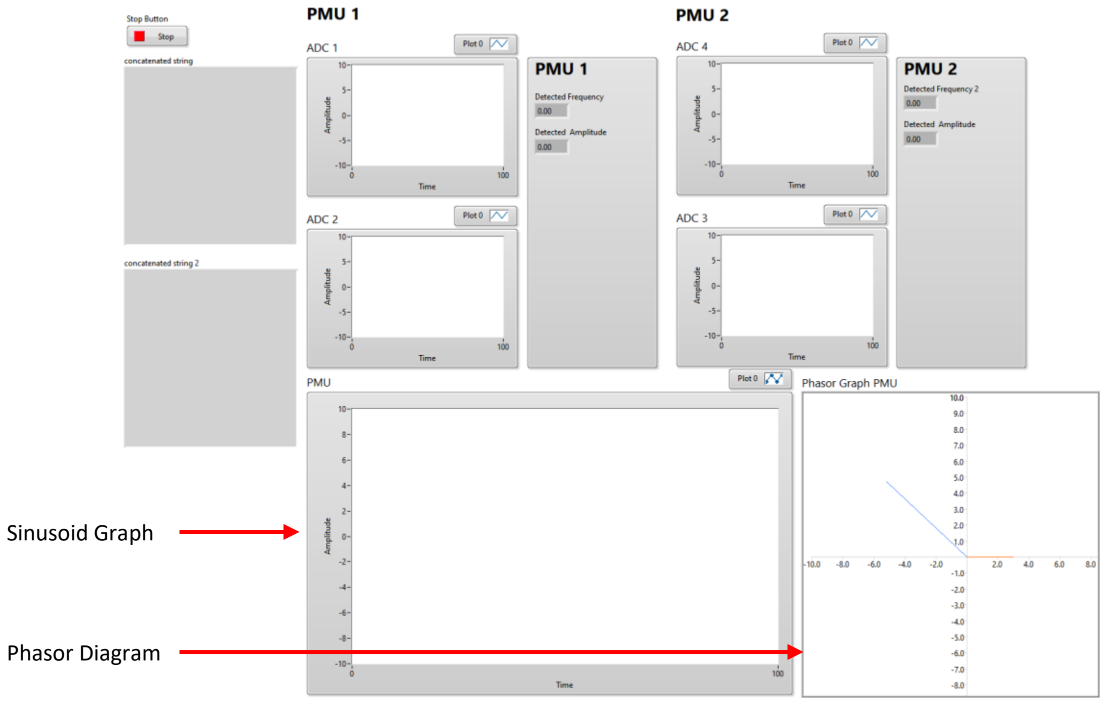
```
SELECT * FROM `pmu2`
```

utctime	loc	voltage	current	frequency	phase

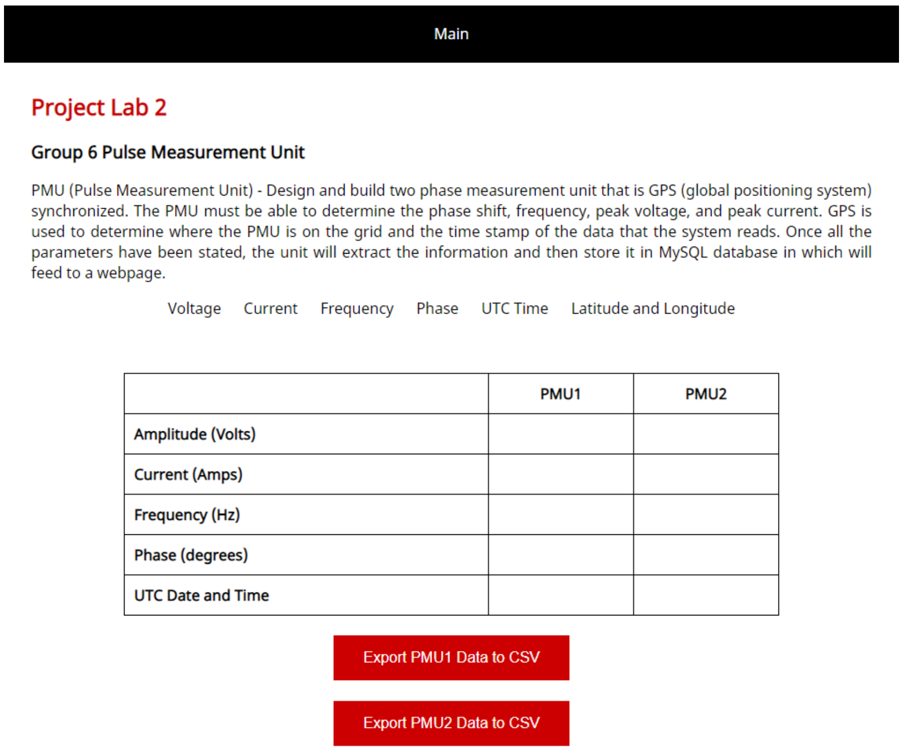
Final LabVIEW VI



Final LabVIEW Front Panel



Final Web Page UI



Main

Project Lab 2

Group 6 Pulse Measurement Unit

PMU (Pulse Measurement Unit) - Design and build two phase measurement unit that is GPS (global positioning system) synchronized. The PMU must be able to determine the phase shift, frequency, peak voltage, and peak current. GPS is used to determine where the PMU is on the grid and the time stamp of the data that the system reads. Once all the parameters have been stated, the unit will extract the information and then store it in MySQL database in which will feed to a webpage.

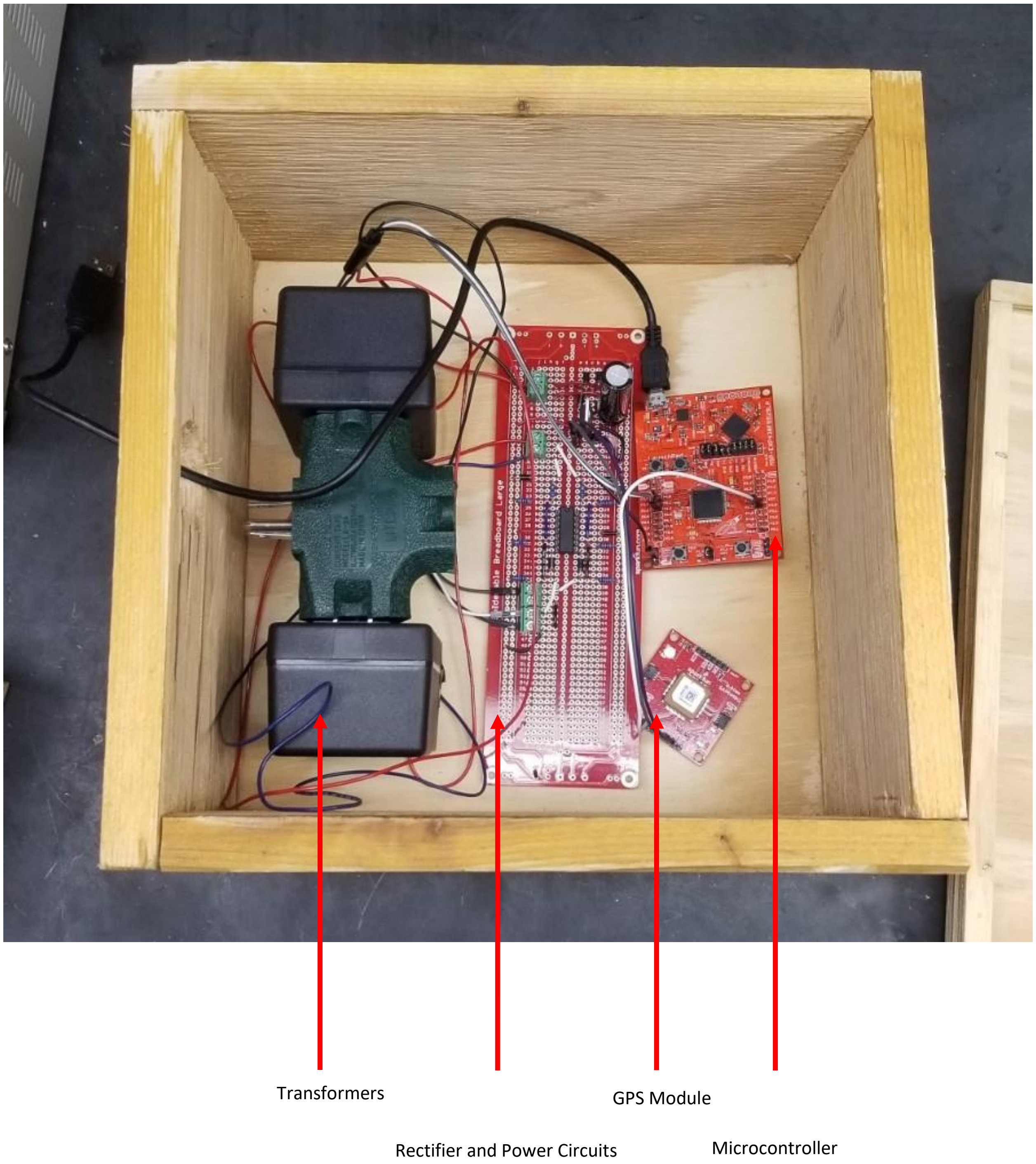
Voltage Current Frequency Phase UTC Time Latitude and Longitude

	PMU1	PMU2
Amplitude (Volts)		
Current (Amps)		
Frequency (Hz)		
Phase (degrees)		
UTC Date and Time		

Export PMU1 Data to CSV

Export PMU2 Data to CSV

Final System Diagram



Issues Faced

Individual works were completed ahead of time. Then during integration, some issues were faced. Since, the phasor block wasn’t a National Instruments proprietary function, efficiency issues were faced when the diagram was plotted. Instead of a smooth gain in phase, the plotted line skipped frames to complete 2pi rotations. Buffer issues were faced while transferring data from the microcontroller to the computer. The uC was holding huge data in its buffer. This was because it couldn’t push the data at the speed it was being collected. This resulted in slower plots of the data compromising the real-time factor of the project’s intention.

Conclusion

Thus, a budgeted Phasor Measurement Unit was built. Information about the AC wave was accumulated correctly to the most content. Voltage, frequency and current data were collected brilliantly. The manipulation of the acquired data for graphical plots was primarily viewed as effective. But, data transfer and representation consumed a huge amount of hardware resources than expected. It maximized the CPU usage causing slower load times and inefficient browsing. So, for further development more research on data transfer and collection needs to be made. Time-syncing aspects of the project for more effective data comparison should be improved upon. Access to the data in an improved environment must also be made. Thus, such data representation of the PMU would help to study failures and outages.

Safety and Ethics

As coveted members of IEEE, certain steps were taken to assure ethical practices in our engineering approach. The safety of our peers, teachers and ourselves was taken of the highest priority. Ethical design practices were used to make sure no one was harmed during the project duration. The factors that might put any party in danger were promptly made aware of. Honesty was maintained in stating claims of the collected data. Constructive criticism from fellow peers and instructors about the technical work to correct all errors made were accepted. Proper acknowledgment of others' contributions to the project was also performed. Attempts to assist peers in their respective projects through the knowledge acquired during the engineering process were also made.

The Phasor Measurement Unit project mainly dealt with an AC Source at mains voltage. Thus, there were steps taken to ensure the safety of the crew members and the customer themselves. A high voltage sign was put up at all times whilst working on the project. It was made sure that all the devices had correctly powered down before leaving the lab. A fuse was also implemented to make sure a dangerous amount of current wasn't drawn.

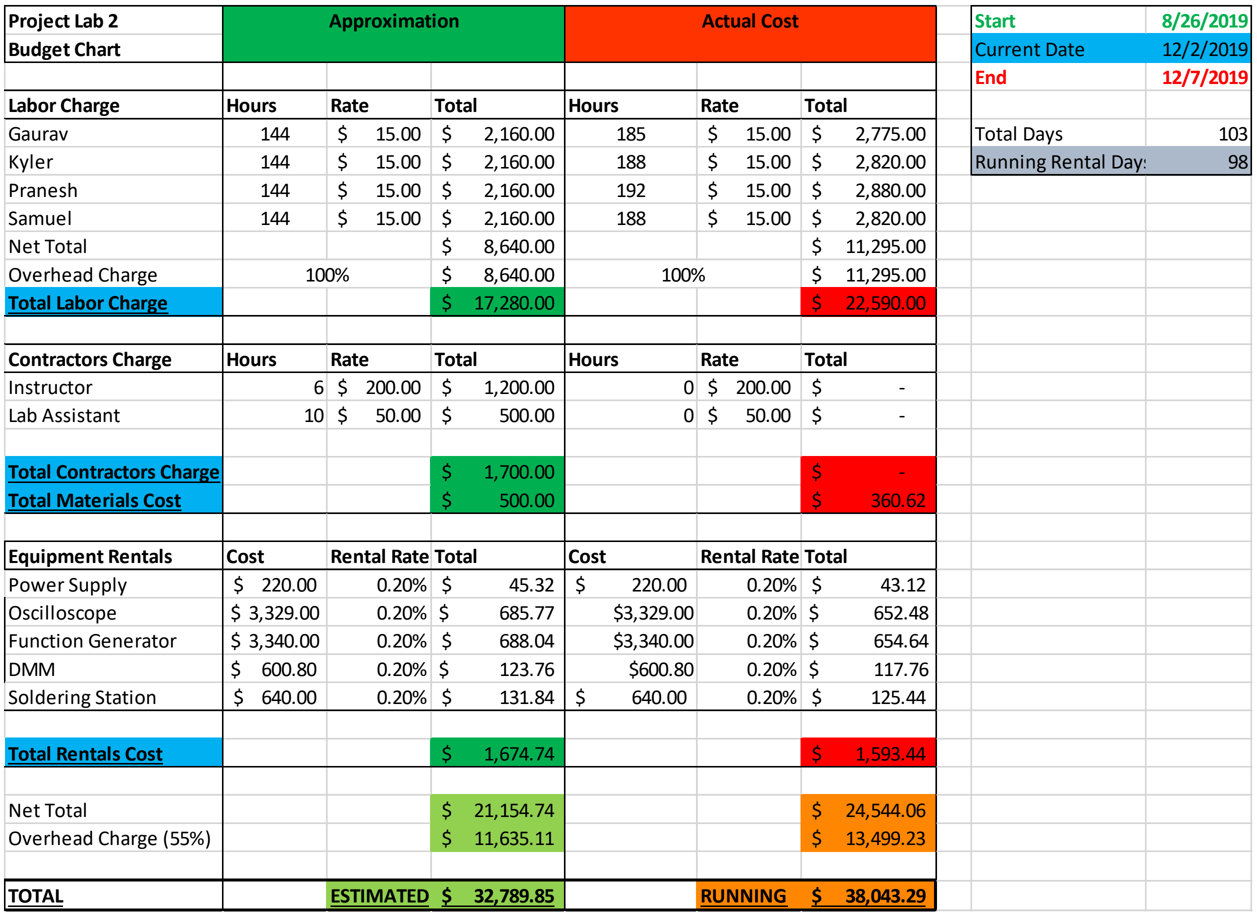
Appendix-References

1. First PMU Built at Virginia Tech, "https://link.springer.com/article/10.1007/s40565-018-0423-3"
2. AJAX Introduction, "https://www.w3schools.com/xml/ajax_intro.asp"
3. MSP430F5529 Datasheet, "http://www.ti.com/lit/ug/slau533d/slau533d.pdf"
4. Sinewave in HTML, "https://gist.github.com/gkhays/e264009c0832c73d5345847e673a64ab"
5. UDLConnection,

   "https://www.micromediaint.com/download/FAQ/Connecting%20ALERT%20to%20a%2

   0MySQL%20DB.pdf"

Appendix-Budget

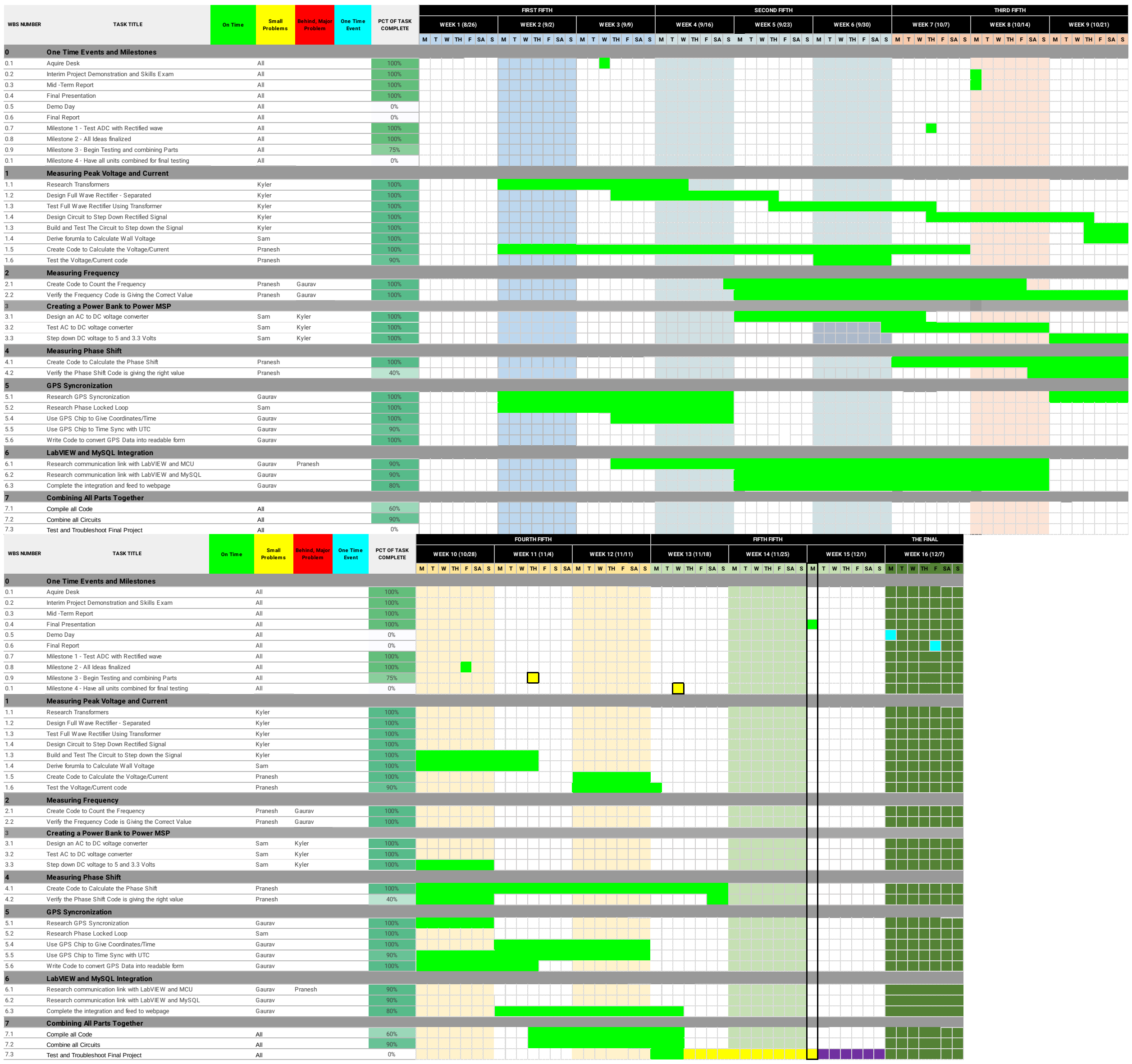


Project Lab 2	Approximation			Actual Cost		
Budget Chart						
Labor Charge	**Hours**	**Rate**	**Total**	**Hours**	**Rate**	**Total**
Gaurav	144	$ 15.00	$ 2,160.00	185	$ 15.00	$ 2,775.00
Kyler	144	$ 15.00	$ 2,160.00	188	$ 15.00	$ 2,820.00
Pranesh	144	$ 15.00	$ 2,160.00	192	$ 15.00	$ 2,880.00
Samuel	144	$ 15.00	$ 2,160.00	188	$ 15.00	$ 2,820.00
Net Total			$ 8,640.00			$ 11,295.00
Overhead Charge	100%		$ 8,640.00	100%		$ 11,295.00
Total Labor Charge			$ 17,280.00			$ 22,590.00
Contractors Charge	**Hours**	**Rate**	**Total**	**Hours**	**Rate**	**Total**
Instructor	6	$ 200.00	$ 1,200.00	0	$ 200.00	$ -
Lab Assistant	10	$ 50.00	$ 500.00	0	$ 50.00	$ -
Total Contractors Charge			$ 1,700.00			$ -
Total Materials Cost			$ 500.00			$ 360.62
Equipment Rentals	**Cost**	**Rental Rate**	**Total**	**Cost**	**Rental Rate**	**Total**
Power Supply	$ 220.00	0.20%	$ 45.32	$ 220.00	0.20%	$ 43.12
Oscilloscope	$ 3,329.00	0.20%	$ 685.77	$3,329.00	0.20%	$ 652.48
Function Generator	$ 3,340.00	0.20%	$ 688.04	$3,340.00	0.20%	$ 654.64
DMM	$ 600.80	0.20%	$ 123.76	$600.80	0.20%	$ 117.76
Soldering Station	$ 640.00	0.20%	$ 131.84	$ 640.00	0.20%	$ 125.44
Total Rentals Cost			$ 1,674.74			$ 1,593.44
Net Total			$ 21,154.74			$ 24,544.06
Overhead Charge (55%)			$ 11,635.11			$ 13,499.23
TOTAL		**ESTIMATED**	**$ 32,789.85**		**RUNNING**	**$ 38,043.29**

Start	**8/26/2019**
Current Date	12/2/2019
End	**12/7/2019**
Total Days	103
Running Rental Day:	98

The budget sheet indicates the total labor charge, materials and rental costs for the project. The materials cost for the front-end development portion included the microcontroller, GPS Module, and Web Hosting services. The actual cost of the project was a little higher than the budget approximated at the start of the semester. This was mostly due to the number of labor hours allocated vs the hours that were actually put in.

Appendix-Gantt Chart



WBS NUMBER	TASK TITLE	On Time	Small Problems	Behind, Major Problem	One Time Event	PCT OF TASK COMPLETE
0	**One Time Events and Milestones**					
0.1	Aquire Desk		All			100%
0.2	Interim Project Demonstration and Skills Exam		All			100%
0.3	Mid -Term Report		All			100%
0.4	Final Presentation		All			100%
0.5	Demo Day		All			0%
0.6	Final Report		All			0%
0.7	Milestone 1 - Test ADC with Rectified wave		All			100%
0.8	Milestone 2 - All Ideas finalized		All			100%
0.9	Milestone 3 - Begin Testing and combining Parts		All			75%
0.1	Milestone 4 - Have all units combined for final testing		All			0%
1	**Measuring Peak Voltage and Current**					
1.1	Research Transformers		Kyler			100%
1.2	Design Full Wave Rectifier - Separated		Kyler			100%
1.3	Test Full Wave Rectifier Using Transformer		Kyler			100%
1.4	Design Circuit to Step Down Rectified Signal		Kyler			100%
1.3	Build and Test The Circuit to Step down the Signal		Kyler			100%
1.4	Derive forumla to Calculate Wall Voltage		Sam			100%
1.5	Create Code to Calculate the Voltage/Current		Pranesh			100%
1.6	Test the Voltage/Current code		Pranesh			90%
2	**Measuring Frequency**					
2.1	Create Code to Count the Frequency		Pranesh	Gaurav		100%
2.2	Verify the Frequency Code is Giving the Correct Value		Pranesh	Gaurav		100%
3	**Creating a Power Bank to Power MSP**					
3.1	Design an AC to DC voltage converter		Sam	Kyler		100%
3.2	Test AC to DC voltage converter		Sam	Kyler		100%
3.3	Step down DC voltage to 5 and 3.3 Volts		Sam	Kyler		100%
4	**Measuring Phase Shift**					
4.1	Create Code to Calculate the Phase Shift		Pranesh			100%
4.2	Verify the Phase Shift Code is giving the right value		Pranesh			40%
5	**GPS Syncronization**					
5.1	Research GPS Syncronization		Gaurav			100%
5.2	Research Phase Locked Loop		Sam			100%
5.4	Use GPS Chip to Give Coordinates/Time		Gaurav			100%
5.5	Use GPS Chip to Time Sync with UTC		Gaurav			90%
5.6	Write Code to convert GPS Data into readable form		Gaurav			100%
6	**LabVIEW and MySQL Integration**					
6.1	Research communication link with LabVIEW and MCU		Gaurav	Pranesh		90%
6.2	Research communication link with LabVIEW and MySQL		Gaurav			90%
6.3	Complete the integration and feed to webpage		Gaurav			80%
7	**Combining All Parts Together**					
7.1	Compile all Code		All			60%
7.2	Combine all Circuits		All			90%
7.3	Test and Troubleshoot Final Project		All			0%

The Gantt chart clearly shows that most of the individual components were duly completed on time. There were a few issues highlighted by the yellow which denotes the occurrence of errors during Data Communication as highlighted before in the paper. The integration of the hardware and software modules took longer than expected which is shown at the end of the Gantt.