

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'Hamed BOUGARA – Boumerdes



Institute of Electrical and Electronic Engineering
Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of

MASTER

In Electronics

Option: Computer Engineering

Title:

**Low-Cost M-Class Phasor Measurement
Unit with Real-Time Monitoring System**

Presented by:

- **Bellabaci Ahmed Nazim**
- **Chaoui Abdenour**

Supervisor:

Dr Metidji Brahim

Abstract

On This project we intended to implement a prototype for a low-cost based M (Measurement) Class Phasor Measurement Unit (PMU) oriented for research and education use cases. The Analog Device ADE7880 is used with a 32-bit MCU to calculate the phasors in addition to frequency and power of a three phase system. The Pulse per Second (PPS) signal from a Sim808 Global Positioning System (GPS) module is used to generate the sampling pulses and synchronize the sampling process done by the Analog to Digital Converters (ADC) of the ADE7880. The reference time is also obtained from the GPS with which the phasors are time stamped and sent via Global Packet Radio Service (GPRS) and displayed in our Synchro Phasor Visualization Dashboard. The data collected from PMUs is stored and processed for visualization and future use.

Acknowledgment

We express our sincere gratitude to our supervisor, Dr. Metidji Brahim for his guidance and equipment support for the implementation of the subject matter of this report.

We also want to thank Dr. Kheldoun Aissa for his valuable suggestions and his consistent encouragements throughout the work.

Last but not least, our warmest gratitude goes to Wameedh Scientific Club which always provided the most suitable working environment without which this thesis would not be in its present form.

Contents

1	Introduction	2
1.1	PMU Fundamentals	3
1.1.1	Phasor	3
1.1.2	Synchro phasor	3
1.2	PMU	4
1.3	Short comings of standard PMUs	5
1.4	Our PMU	5
1.5	Standardization	5
2	Hardware design and implementation	7
2.1	Signal Acquisition	7
2.2	ADC and Data Processing	8
2.2.1	ADE 7880	8
2.2.2	Microcontroller	12
2.3	Signal Conditioning	18
2.3.1	Current Measurement	19
2.3.2	Voltage Measurement	20
2.4	Data Transmission	22
2.4.1	Sim808 interface	22
2.4.2	Sim808 C library	28
2.5	GPS synchronization signal	30
2.6	Program flowchart	31
2.7	PCB Design	32
2.7.1	Proteus ISIS	32
2.7.2	Circuit Schematic Design	33
2.7.3	PCB Layout	36
3	Real time monitoring system	37
3.1	Data collection	38
3.2	Data verification	38
3.3	Data storage	39
3.3.1	Architecture	39
3.4	Data processing	39
3.5	Synchrophasor data visualization	40

4	Experimental Results and Discussion	43
4.1	Standard Requirements	43
4.2	Results & Evaluation	44
5	Conclusion	46
	Appendices	49
A	ADE Library	50
B	SIM808 Library	52
C	PCB Design	54
D	Secondary Components Schematics	56
E	Experiment Test Code	59
F	Experiment Test Results	61

List of Figures

1.1	Phasor representation of an alternating quantity	3
1.2	Phasor Comparison of Two Different Buses at Remote Location . .	4
1.3	Phasor Comparison with Reference Phase angle	4
2.1	PMU Schematic Diagram	7
2.2	PMU installation and connection	8
2.3	ADE7880 Functional Block Diagram	9
2.4	ADE7880 current differential analog input	9
2.5	SPI Read Operation of a 32-Bit Register	11
2.6	SPI Write Operation of a 32-Bit Register	12
2.7	STM32 ST-LINK Utility	13
2.8	mikroC PRO for ARM IDE	13
2.9	ZERO PLUS logic Analyzer	17
2.10	Current Transformer	19
2.11	YUANXING PCB Mount Current Transformers	20
2.12	CT APPLICATION METHOD	20
2.13	YUANXING PCB mount Voltage Transformers	21
2.14	Voltage Transformer Application Method	21
2.15	SIM808 Top and Bottom View	22
2.16	SIM808 Serial Interface	23
2.17	SIM808 Call initiation	24
2.18	SIM808 GPRS UDP connection initiation	25
2.19	IEEE C37.118-2 standard message format	26
2.20	27
2.21	Map location of decoded NMEA coordiantes	28
2.22	PPS synchronized data sampling	30
2.23	PMU STM32f103 Program Flowchart	32
2.24	PMU Power Supply Proteus Circuit Diagram	33
2.25	Sim808 Schematic Diagram	34
2.26	ADE7880 Functional Circuit	34
2.27	STM32f103C8 Schematic Diagram	35
2.28	Conditioning Circuit Proteus Circuit Diagram	35
2.29	PMU Proteus PCB Layout	36
2.30	PMU PCB	36
3.1	Real time monitoring system software structure	37
3.2	UDP server	38
3.3	ERD of the database	39

3.4	Real time display webpage	40
3.5	Line chart display	41
3.6	Phasor diagram display	41
3.7	phase angle display	41
3.8	power factor display	42
3.9	power values display	42
4.1	Graphical representation of the TVE. Based on IEEE C37.118 . . .	44
4.2	PMU Test Implementation	45
D.1	Proteus Schematic Diagram	56
D.2	Datasheet Schematic Diagram	57
D.3	Proteus Schematic Diagram	57
D.4	Datasheet Schematic Diagram	57
D.5	Molex Sim Card Holder Pin Definition	58
D.6	Proteus Schematic Diagram	58

List of Tables

2.1	Identification ATcommands Overview	23
2.2	GPRS related ATcommands Overview	25
2.3	GPS related ATcommands Overview	26
4.1	PMU Test Results	45
F.1	61

CS Coding Scheme

CSD Circuit Switch Data

CT Current Transformer

ERD Entity Relationship Diagram

ESD Electrostatic discharge

GSM Global Standard for Mobile Communication

GUI Graphical User Interface

IAP Line A Positive

IAN Line A Negative

IBP Line B Positive

IBN Line B Negative

ICP Line C Positive

ICN Line C Negative

IDE Integrated Development Environment

IMEI International Mobile Equipment Identity

INP Line Neutral Positive

INN Line Neutral Negative

IOT Internet of things

ISR Interrupt Service Routine

MCU μ C Microcontroller unit

MISO Master in Slave Out

MOSI Master Out Slave In

MSB Most Significant Bit

SIM Subscriber Identification Module

SPI Serial Peripheral Interface

UART Universal Asynchronous Receiver-Transmitter

UDP User Datagram Protocol

UI User Interface

UTC Universal Coordinated Time

Chapter 1

Introduction

As the world continues to move towards a smarter grid day by day, it has become a necessity to incorporate real-time monitoring of the grid where an instantaneous evaluation can be made available.

Phasors are considered to be the most efficient parameters for the representation of the instantaneous state of the Electrical Grid, that's why PMUs play a key role in enhancing power systems stability and reliability. The phasors are time tagged and transmitted from PMUs located at remote substations or generating stations to a central repository called Phasor Data Concentrators (PDC) where they can be time-aligned, compared and stored. Although the phasor of a substation may not mean much at that substation itself, the comparison of these phasors data from different locations at a central location provides the real-time angular differences and system separation between the different parts of the grid.

This paper highlights the method for the design and implementation of a low-cost PMU with its Synchro phasor Visualization Dashboard. The first Chapter brings some Fundamental definitions, shortcomings of standard PMUs and the difference we wanted to make with our work. Chapter two goes through the hardware design and implementation whereas Chapter three presents our Synchro Phasor Visualization Dashboard. On Chapter four we discuss our results, as we performed test to prove the compliance of our work with the IEEE Std. C37.118-2014. Finally in the last section we make a main conclusion and future work we consider to do, especially to develop a P (Protection) Class PMU as well as prediction and anomaly detection algorithms with the data gathered during measurements.

1.1 PMU Fundamentals

1.1.1 Phasor

A Phasor is a complex number that represents both the magnitude and phase angle of the sine waves found in electricity [1].

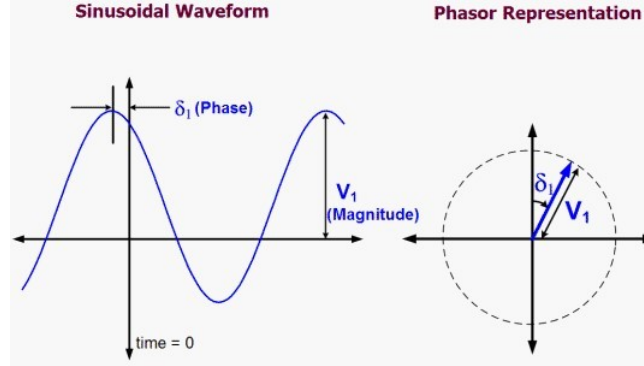


Figure 1.1: Phasor representation of an alternating quantity

Phasor measurements that occur at the same time are called "Synchrophasors". A Synchrophasor is the metered value whereas the PMU is the metering device.

1.1.2 Synchro phasor

A Synchrophasor is a phasor that is time stamped to an extremely accurate time reference like GPS signals. Phasors are sampled at high speed, typically between 30 and 120 samples per second. The IEEE Std. C37.118.1a-2014 defines the synchrophasor representation (X) of a periodic signal as: $X(t) = X_m \cos(\omega t + \theta)$.

$$X = \frac{X_m}{\sqrt{2}} e^{j\phi} \quad (1.1)$$

where the magnitude of the synchrophasor ($X = |X|X_m/\sqrt{2}$) is the RMS value of $x(t)$, and $\theta = \text{angle}(X)$ represents the momentary phase angle relative to a cosine function at the nominal system frequency synchronized to UTC [1].

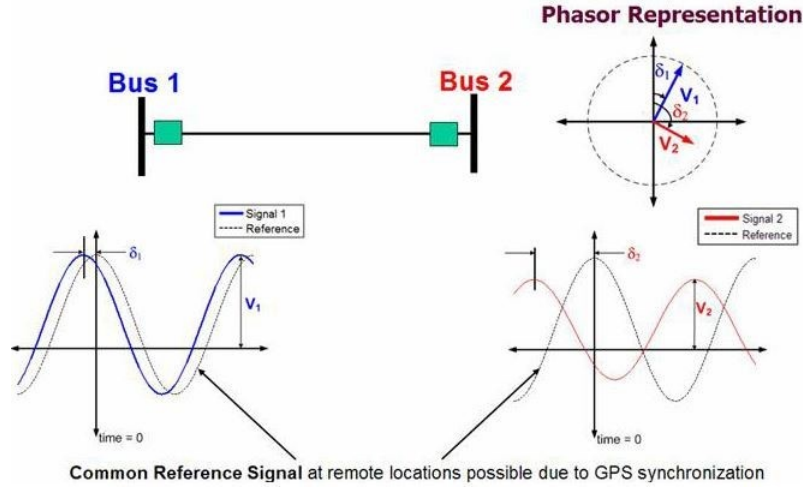


Figure 1.2: Phasor Comparison of Two Different Buses at Remote Location

The phase angle differences between two sets of phasor measurements (i.e. $\delta_1 - \delta_2$) is independent of the reference. Typically, one of the phasor measurements is chosen as the “reference” and the difference between all the other phase angle measurements (also known as the absolute phase angle) and this common “reference” angle is computed and referred to as the relative phase angles with respect to the chosen reference (see figure below) [2].

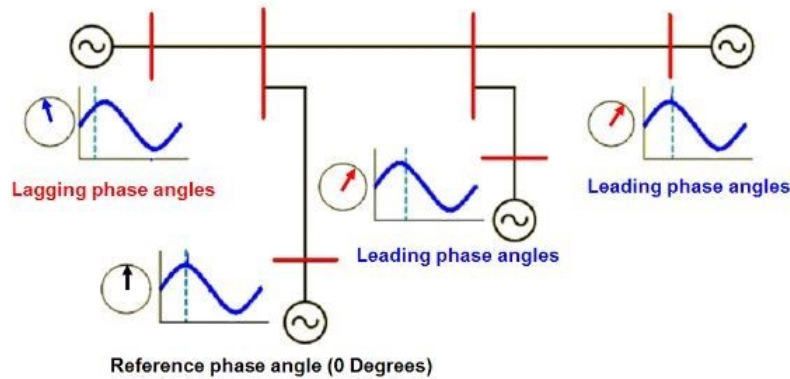


Figure 1.3: Phasor Comparison with Reference Phase angle

By synchronizing the sampling processes for different signals, which may be hundreds of miles apart, it is possible to put their phasors on the same phasor diagram.

1.2 PMU

A PMU is a device used to estimate the magnitude and phase angle of an electrical Phasor quantity like voltage or current as well as frequency in the electricity grid using a common time source for synchronization. Time synchronization is usually provided by GPS and allows synchronized real-time measurements of multiple

remote measurement points on the grid. PMUs measure voltages and currents at principal intersecting locations (critical substations) on a power grid and can output accurately time-stamped voltage and current phasors to monitor voltage and current change across the network when something happens (e.g. line outage, generation trip, or load change) [3].

1.3 Short comings of standard PMUs

Due to the cost of early PMU devices, PMU technology has historically been limited to transmission system applications where the business case justified expensive phasor analysis equipment. Commercial PMUs are already available in the market, but they often come with a very high price tag and strict copyright limitations. Costs for such units vary between US \$6000 and US \$15000 depending on the specification. The schematics of the PMUs are not openly available as their business policies go against it, the algorithm or methodology of how a PMU actually operates, how it calculates the phasors of the sampled voltage and current signal are also guarded by copyright laws. These PMUs do not allow to be used for educational or academic research purposes.

1.4 Our PMU

In this spirit we aimed to work on a platform which can be reconfigured to suit the requirement of the end users allowing control and visibility over the entire function of the device. Building a low cost PMU might have proven a costly endeavor in the past but recent advances and wide availability of low cost high performance MCUs have given rise to many possibilities which can be exploited to build the desired PMU. There has recently been a growing attempt to manufacture a cost effective open source PMU for research and academic purpose. we wanted our work to be oriented on research and education use cases but also power system model validation, focusing on how the data is used for the display with an efficient UI (User Interface). We designed our PMU following IEEE Std. C37.118 standard to qualify it as a reliable measurement device, but also permit it to be compatible with other end user tools, independent of the platform and operating system.

1.5 Standardization

In order to allow the power industry to store or transmit time-tagged data and to guarantee inter compatibility across networks the IEEE published standard C37.118 in 2005 which was updated in 2011 in two parts; IEEE Std. C37.118.1 [4] gives details of the estimation of the synchro phasor and certification requirements, and IEEE Std. C37.118.2 [4] gives details of how the data is to be represented and transferred. This standard introduced two types of PMUs to ensure that the device is compatible with existing PDCs and visualization softwares:

M-class: primarily for steady-state measurements, similar to the requirements presented on the IEEE C37.118-2005.

P-class: dedicated to protection functionality, with fast response without filtering, having more relaxed performance but intended to capture the dynamic behavior [1].

Chapter 2

Hardware design and implementation

This section describes the implementation of our Phasor measurement unit. The block diagram of such a unit is shown in Figure 2.1 .

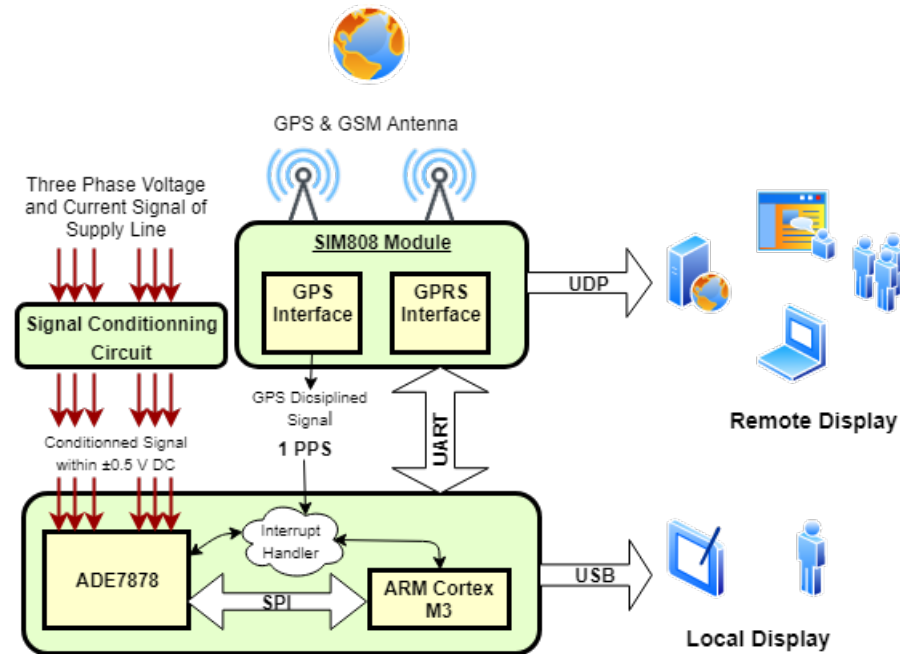


Figure 2.1: PMU Schematic Diagram

2.1 Signal Acquisition

For the calculation of a phasor, the data (i.e. the sampled voltage & current signals) must be acquired.

when the PMU is tested in real-world scenarios a means of getting the signals from the transmission lines is necessary, which is accomplished using a Potential

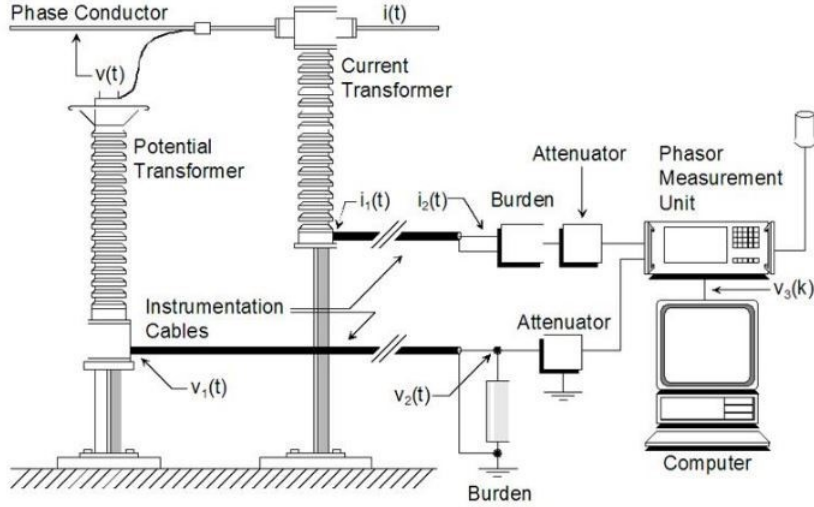


Figure 2.2: PMU installation and connection

Transformer (PT) and a Current Transformer (CT) in the substations. This signal is further stepped down Using the voltage & current sensors as described in Chapter 2.3. Typically installation and interconnection of a PMU at a substation or at a generation plant where electrical engineers install “shunts” in all Current Transformer (CT) secondary circuits that are to be measured. Potential Transformer (PT) connections will not require the installation of any additional equipment other than terminal blocks and fuses. They will have to run wires from the CT shunts and the PTs to either an interface cabinet or directly to the input connections of the PMU [5] (Figure 2.2).

2.2 ADC and Data Processing

2.2.1 ADE 7880

The ADE7880 is a high accuracy, 3-phase electrical energy measurement IC with serial interface, it incorporates a set of 24 bit analog-to-digital converters (ADCs), a digital integrator, reference circuitry, and all of the signal processing required to perform RMS calculations as well as the RMS of harmonics on the phase and neutral currents and on the phase voltages, together with the active, reactive and apparent powers, and the power factor and harmonic distortion on each harmonic for all phases. Total harmonic distortion (THD) is computed for all currents and voltages. A fixed function digital signal processor (DSP) executes all this signal processing. The DSP program is stored in the internal ROM memory. The ADE7880 contains waveform sample registers that allow access to all ADC outputs; The devices also incorporate power quality measurements, voltage period measurement, and angles between phase voltages and currents [6].

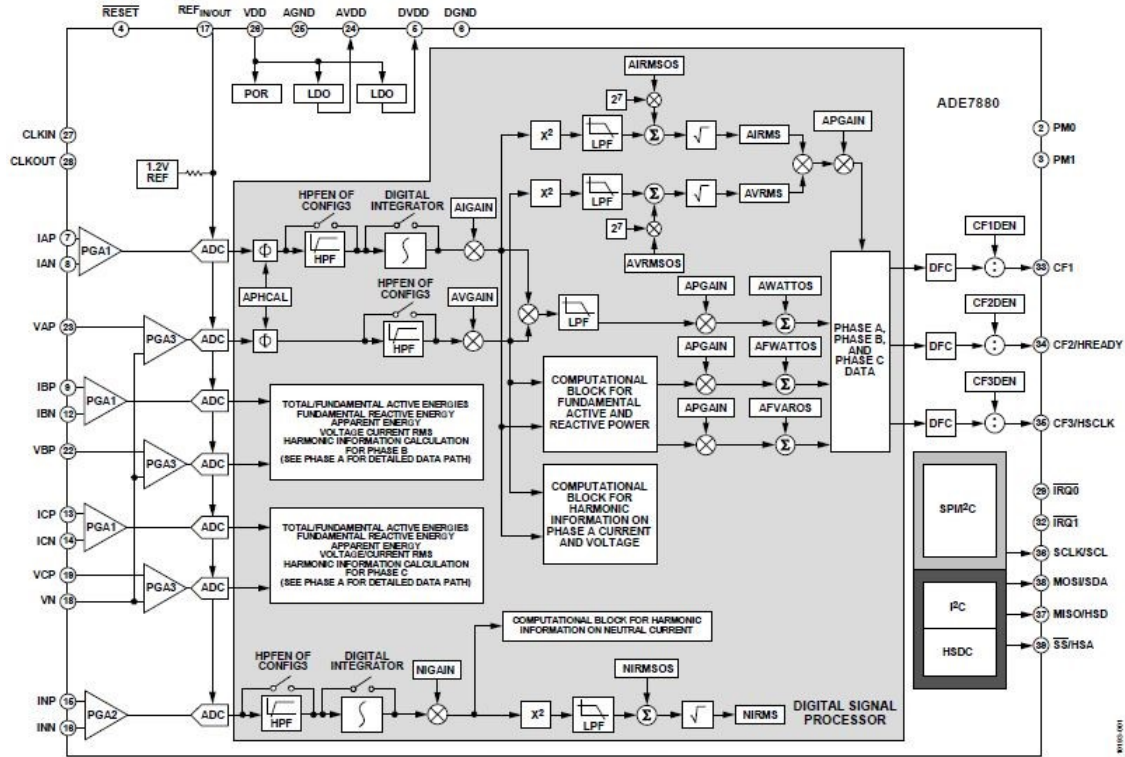


Figure 1. ADE7880 Functional Block Diagram

Figure 2.3: ADE7880 Functional Block Diagram

THEORY OF OPERATION

Analog Inputs

The ADE7880 has seven analog inputs forming current and voltage channels. The current channels consist of four pairs of fully differential voltage inputs: IAP and IAN, IBP and IBN, ICP and ICN, and INP and INN. These voltage input pairs have a maximum differential signal of 0.5 V. The maximum signal level on analog inputs for the IxP/IxN pair is also 0.5 V with respect to AGND. Figure 2.5 presents a schematic of the input for the current channels and their relation to the maximum common-mode voltage [6].

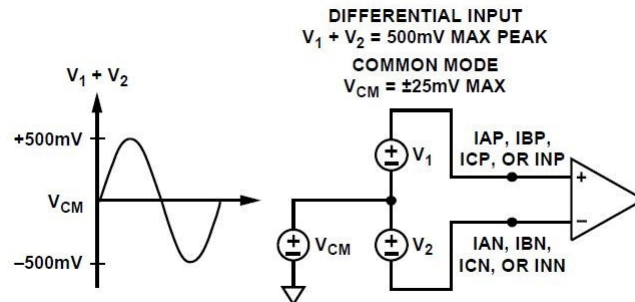


Figure 2.4: ADE7880 current differential analog input

Current and Voltage Sampling

The ADC outputs are signed twos complement 24-bit words and are available at a rate of 8 kSPS. With the specified full-scale analog input signal of 0.5 V, the ADC produces its maximum output code value. The ADC output swings between 5,326,737 (0xAE86F) and +5,326,737 (0x514791). The serial ports of the ADE7880 work on 32-, 16-, or 8-bit words. When the 24-bit signed data registers are read from the ADE7880, they are transmitted sign extended to 32 bits[6].

Method of frequency estimation

The ADE7880 provides the period measurement of the line in the voltage channel. Bits [1:0] (PERSEL[1:0]) in the MMODE[7:0] register select the phase voltage used for this measurement. The period register is a 16-bit unsigned register and is updated every line period. The period measurement has a resolution of 3.90625 s/LSB (256 kHz clock), which represents 0.0195% (50 Hz/256 kHz) when the line frequency is 50 Hz and 0.0234% (60 Hz/256 kHz) when the line frequency is 60 Hz [6].

Digital Signal Processor

The ADE7880 contain a fixed function DSP that computes all powers and rms values. It contains program memory ROM and data memory RAM. The program used for the power and rms computations is stored in the program memory ROM and the processor executes it at a rate of 8 kHz.

At power-up or after a hardware or software reset, the DSP is in idle mode. No instruction is executed. All the registers located in the data memory RAM are initialized at 0, their default values, and they can be read/written without any restriction. The run register, used to start and stop the DSP, is cleared to 0x0000. The run register needs to be written with 0x0001 for the DSP to start code execution [6].

Serial Interfaces

The ADE7880 have three serial port interfaces: one fully licensed I2C interface, one serial peripheral interface (SPI), and one high speed data capture port (HSDC). As the SPI pins are multiplexed with some of the pins of the I2C and HSDC ports, the ADE78xx accepts two configurations: one using the SPI port only and one using the I2C port in conjunction with the HSDC port, in our case we went with the SPI interface.

After reset, the HSDC port is always disabled. We choose between the I2C and SPI ports by manipulating the SS/HSA pin after power-up or after a hardware reset. To configure the serial port for SPI the SS/HAS pin is toggled high to low three times, If the SS/HSA pin is kept high the I2C port is selected [7].

SPI-Compatible Interface

The SPI of the ADE7880 is always a slave of the communication and consists of four pins (with dual functions): SCLK/SCL, MOSI/SDA, MISO/HSD, and SS/HSA. The functions used in the SPI-compatible interface are SCLK, MOSI, MISO, and SS. The serial clock for a data transfer is applied at the SCLK logic input. All data transfer operations synchronize to the serial clock. Data shifts into the ADE7880 at the MOSI logic input on the falling edge of SCLK and the ADE7880 samples it on the rising edge of SCLK. Data shifts out of the ADE7880 at the MISO logic output on a falling edge of SCLK and can be sampled by the master device on the raising edge of SCLK. The most significant bit of the word is shifted in and out first. The maximum serial clock frequency supported by this interface is 2.5 MHz. The SS logic input is the chip select input. This input is used when multiple devices share the serial bus.

SPI Read Operation

The read operation using the SPI interface of the ADE7880 initiate when the master sets the SS /HSA pin low and begins sending one byte, representing the address of the ADE7880, on the MOSI line. The master sets data on the MOSI. The most significant seven bits of the address byte can have any value, Bit 0 (read/write) of the address byte must be 1 for a read operation. Next, the master sends the 16-bit address of the register that is read. After the master receives the last bit, it sets the SS and SCLK lines high and the communication ends. See Figure 2.6 for details of the SPI read operation [6].

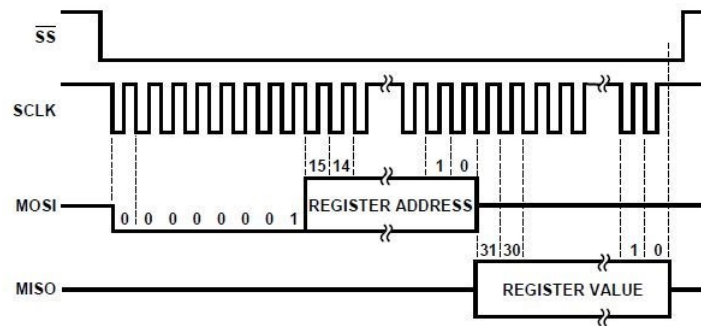


Figure 2.5: SPI Read Operation of a 32-Bit Register

SPI Write Operation

The write operation using the SPI interface of the ADE7880 initiates when the master sets data on the MOSI. The most significant seven bits of the address byte can have any value. 1. Bit 0 (read/ write) of the address byte must be 0 for a write operation. Next, the master sends both the 16-bit address of the register that is written and the 32-, 16-, or 8-bit value of that register. After the last bit is transmitted, the master sets the SS and SCLK lines high at the end of the SCLK cycle and the communication ends. See Figure 2.7 for details of the SPI write operation [6].

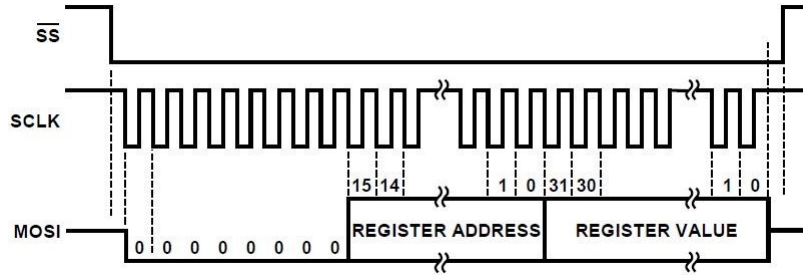


Figure 2.6: SPI Write Operation of a 32-Bit Register

2.2.2 Microcontroller

The STM32F103xx medium-density performance line family incorporates the high performance ARM® Cortex-M3 32-bit RISC core operating at a 72 MHz frequency, high speed embedded memories (Flash memory up to 128 Kbytes and SRAM up to 20 Kbytes), and an extensive range of enhanced I/Os and peripherals. All devices offer two 12-bit ADCs, three general purpose 16-bit timers plus one PWM timer, as well as standard and advanced communication interfaces: up to two I2Cs and SPIs, three USARTs, an USB and a CAN. These features make the STM32F103xx medium-density performance line MCU family suitable for a wide range of applications such as motor drives, application control, medical and handheld equipment, PC and gaming peripherals, GPS platforms, industrial applications, PLCs, inverters, printers, scanners, alarm systems, video intercoms, and HVACs [7].

Flash Debug

The ARM Serial wire debug port (SWJ-DP) Interface is embedded, and is a combined JTAG and serial wire debug port that enables either a serial wire debug or a JTAG probe to be connected to the target. The JTAG TMS and TCK pins are shared with SWDIO and SWCLK, respectively, and a specific sequence on the TMS pin is used to switch between JTAG-DP and SW-DP [ST] [7]. In our case, we use the Serial Wire debug port to load hex files to the STM32F103 using STM32 ST-LINK Utility software (Figure 2.8).

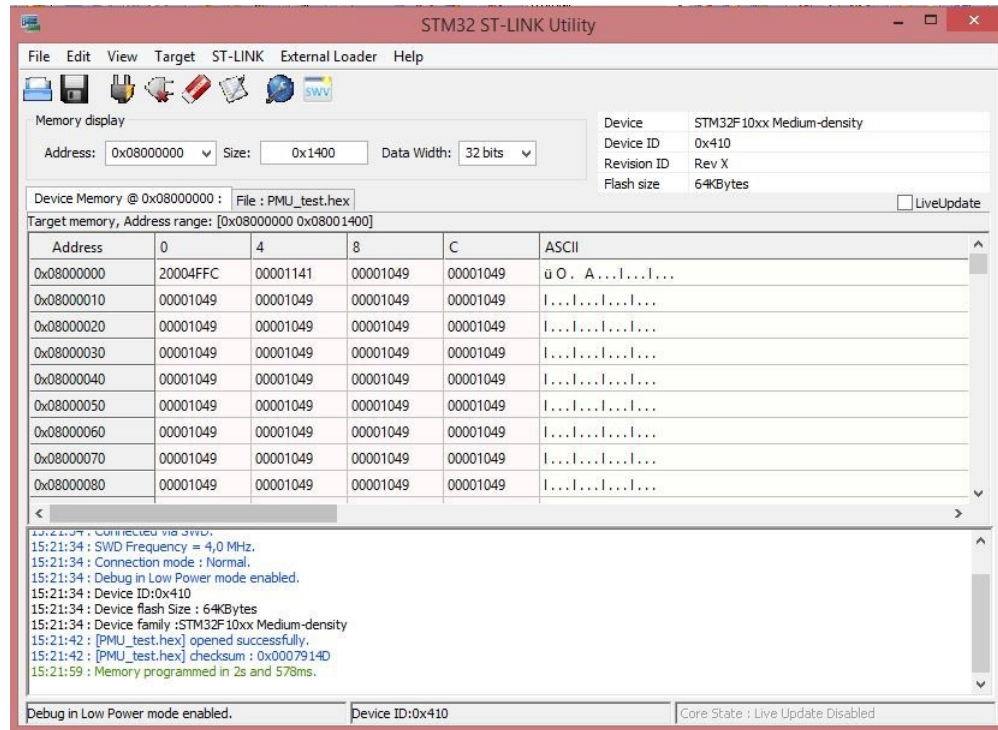


Figure 2.7: STM32 ST-LINK Utility

Software Environment

MikroC PRO for ARM is a full-featured ANSI C compiler for ARM Cortex-M3, M4 and M7 devices. It is the best solution for developing code for ARM devices. It features intuitive IDE, powerful compiler with advanced optimizations, lots of hardware and software libraries, and additional tools. Compiler comes with comprehensive Help file and lots of ready-to-use examples [8].

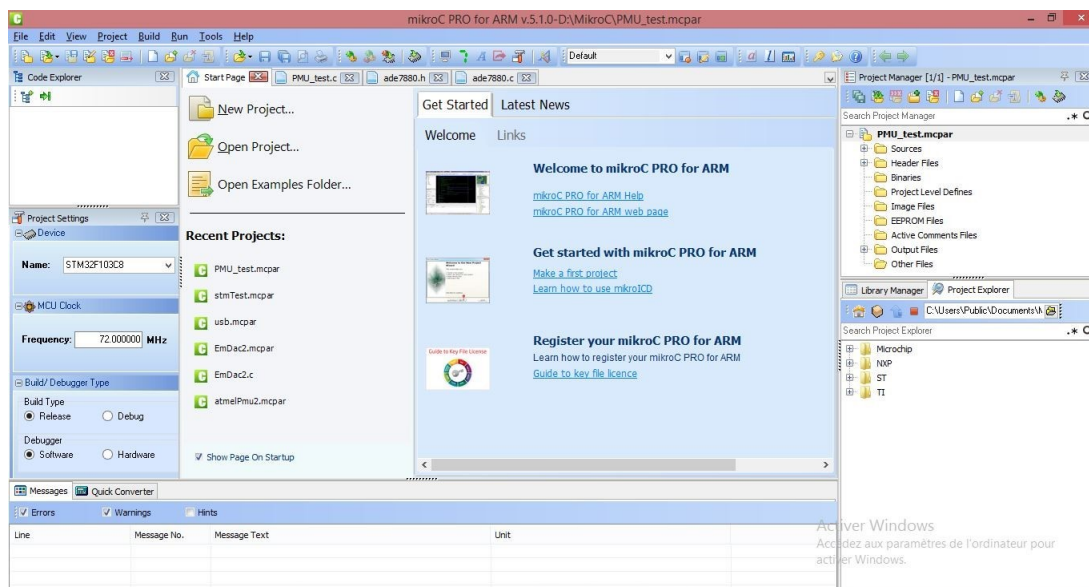


Figure 2.8: mikroC PRO for ARM IDE

ADE7880 library

To standardize and implement our work in an efficient way, we wrote the code for a special ADE7880 library (which is also compatible with all ADE78xx family). The header file contains registers address and other constant definitions as well as the prototypes for the functions we created. Because the registers of the ADE's DSP have multiple length (8, 16, 24, and 32), we had to create read/write functions with different SPI read/write length. We made our library available on Github.

```
#define RUNREG 0xE228

//Phase rms voltage and current
#define AIRMS 0x43C0
#define AVRMS 0x43C1
#define BIRMS 0x43C2
#define BVRMS 0x43C3
#define CIRMS 0x43C4
#define CVRMS 0x43C5
#define NIRMS 0x43C6

//interrupt status regs
#define STATUS0 0xE502
#define STATUS1 0xE503

//interrupt enable regs
#define MASK0 0xE50A
#define MASK1 0xE50B

//Instantaneous value of Phase currents
#define IAWV 0xE50C
#define IBWV 0xE50D
#define ICWV 0xE50E
#define INWV 0xE50F

//Instantaneous value of Phase voltages
#define VAWV 0xE510
#define VBWV 0xE511
#define VCWV 0xE512

//Instantaneous value of active power
#define AWATT 0xE513
```

```

#define BWATT 0xE514
#define CWATT 0xE515

//Instantaneous value of active power
#define AVA 0xE519
#define BVA 0xE51A
#define CVA 0xE51B

#define CHECKSUM 0xE51F
#define CONFIG 0xE618
#define CONFIG2 0xEC01
#define CONFIG3 0xEA00
#define MMODE 0xE700
#define CFMODE 0xE610

//RMS Offset
#define AIRMSOS 0x438F
#define AVRMSOS 0x4390
#define BIRMSOS 0x4391
#define BVRMSOS 0x4392
#define CIRMSOS 0x4393
#define CVRMSOS 0x4394
#define NIRMSOS 0x4395

//phase angles
#define ANGLE0 0xE601
#define ANGLE1 0xE602
#define ANGLE2 0xE603

unsigned char ADE_Read8(unsigned int reg);
unsigned int ADE_Read16(unsigned int reg);
unsigned long ADE_Read24(unsigned int reg);
unsigned long ADE_Read32(unsigned int reg);

void ADE_Write8(unsigned int reg, unsigned char dat);
void ADE_Write16(unsigned int reg, unsigned int dat);
void ADE_Write24(unsigned int reg, unsigned long dat);
void ADE_Write32(unsigned int reg, unsigned long dat);

```

```

void ADE_Init();
void ADE_SpiEnable();
void ADE_SpiMode(char mode);
unsigned long getVRMS(char phase);
unsigned long getIRMS(char phase);

unsigned long getWatt(char phase);
unsigned long getVA(char phase);
unsigned int getPhaseShift(char phase);

```

Testing the Ade Library

We first initialize the SPI peripheral in our Master device (STM32F103) according to the ADE7880 datasheet specification. 1.25 Mhz Spi speed , Clock idle state is high , 8 bit data transfer on second edge transition , MSB first.

```

void SpiSetup(){

    GPIO_Digital_Output(&GPIOB.BASE, _GPIO_PINMASK_3| _GPIO_PINMASK_5);
    GPIO_Digital_Input(&GPIOB.BASE, _GPIO_PINMASK_4);
    GPIO_Digital_Output(&GPIOA.BASE, _GPIO_PINMASK_15); // Set CS# pin as Output
    GPIOA_ODRbits.ODR15=1;
    AFIOEN_bit = 1;
    SWJ_CFG1_bit = 1;
    SPI1_Init_Advanced(_SPI_FPCLK_DIV32, _SPI_MASTER | _SPI_8_BIT | _SPI_CLK_IDLE_HI(
    _SPI_FIRST_CLK_EDGE_TRANSITION | _SPI_MSB_FIRST | _SPI_SS_ENABLE | _SPI_SSM_DIS(
    _SPI_SSI_0 , &GPIO_MODULE_SPI1_PB345);
    AFIO_MAPR.SPI1_REMAP=1;
}

```

To enable the SPI interface on the Analog Device, we execute three SPI write operations to a location in the address space that is not allocated to a specific ADE78xx register (for example 0xEBFF, where eight bit writes can be executed). These writes allow the SS/HSA pin to toggle three times. We then write 0x00 to the CONFIG2 register to lock the interface into SPI mode.

```

void ADE_SpiEnable(){
    char d=0;
    //toggle cs 3 times by writing 0x0 to a reserved reg address
    for(d=0;d<3;d++)ADE_Write8(0xEBFF,0x00);
    //lock spi
    ADE_Write8(CONFIG2,0x00);
}

```

Next step is to initialize the ADE according to our needs, this implies configuration registers as well as offset and gain registers. When finished we enable write protection into the ADE and start the DSP by writing 0x0001 the RUN register.

```
void ADE_Init(){

    ADE_Write16(CFMODE,0x00A0);
    //Write protection
    ADE_Write8(0xE7FE,0xAD);
    ADE_Write8(0xE7E3,0x80);
    //start dsp
    ADE_Write16(RUNREG,0x0001);
}
```

To test the correctness of our library we try perform a read operation on the content of the CHECKSUM register at run time and check we get the default value specified on the datasheet (0x33666787).

To debug and get a close look on our SPI implementation we used the ZERO PLUS logic Analyzer with LAP-C_V3.13.05 software interface.



Figure 2.9: ZERO PLUS logic Analyzer

Measurement functions

Our Library contains functions to get the calculated data from the appropriate registers. In addition to the parameters mentioned on the IEEE standard (RMS values and frequency) we also integrated further parameters that are also calculated by the ade7880 such as active, reactive and apparent power as well as phase shifts.

```
unsigned long getVRMS(char phase){

    if (phase==0)return ADE_Read32(AVRMS);
    else if (phase==1)return ADE_Read32(BVRMS);
    else return ADE_Read32(CVRMS);
}
```

```

unsigned long getIRMS(char phase){

if (phase==0)return ADE_Read32(AIRMS);
else if (phase==1)return ADE_Read32(BIRMS);
else return ADE_Read32(CIRMS);
}

unsigned long getWatt(char phase){

if (phase==0)return ADE_Read32(AWATT);
else if (phase==1)return ADE_Read32(BWATT);
else return ADE_Read32(CWATT);
}

unsigned long getVA(char phase){

if (phase==0)return ADE_Read32(AVA);
else if (phase==1)return ADE_Read32(BVA);
else return ADE_Read32(CVA);
}

unsigned int getPhaseShift(char phase){

if (phase==0)return ADE_Read16(ANGLE0);
else if (phase==1)return ADE_Read16(ANGLE1);
else return ADE_Read16(ANGLE2);
}

```

The ADE also contains register for THD and harmonics calculation, where special registers needs to be configured to get the appropriate component from the select phase. Appendix 1 contains the rest of our library code.

2.3 Signal Conditioning

The three phase voltages need to be measured by the ADE for estimation of the Phasors. However, the problem is the ADE's ADC being able to only take in the signals in the range of 0.5 V DC as input. To address this issue a suitable voltage signal conditioning circuit needed to be designed which can convert the 240 volt Phase voltages. For this purpose, suitable Current and voltage sensors has been designed. Current and voltage sensors must be conditioned to vary from the differential input of the ADE7880.

2.3.1 Current Measurement

Current Transformers

CTs are useful when measuring AC, transients or switching mode DC, since it senses the changing magnetic field produced by the AC oscillation. The CTs use the power line as the primary of a transformer, with 1 to a few turns that, its flowing current (I in Figure 2.16), induces an alternating magnetic field in the core (B in Figure 2.16), producing an alternating current in the secondary. As the induced current is the result of the relationship between the primary's number of turns (single turn), and secondary's turns (N), the output current $1/N \cdot I$

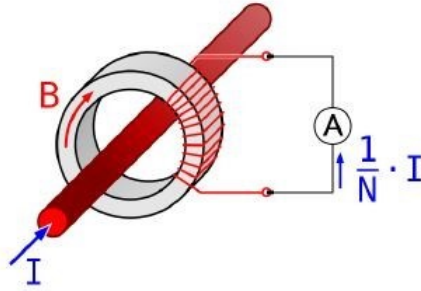


Figure 2.10: Current Transformer

$$V_{out} = I_{out} \times R_{burden} \quad (2.1)$$

where I_{out} is the output current at both ends of the secondary in a closed circuit, N is the number of turns of the secondary, V_{out} is the current flowing through the power line. Placing a low value burden resistor in parallel with the load and closing the secondary winding circuit will convert the given current to a voltage signal that can be calculated, knowing the desired output voltage, by ohms law.

YUANXING CT

The YUANXING TA series current transformers are designed for applications where AC current signals must be transformed accurately into an owner AC current or voltage signal appropriate for micro-processor based circuits. The TA2X11 series of PCB Mounted Miniature current transformers Are designed specifically for integration into products which require exceptionally accurate primary signal transformation while exposed to Harsh environmental operating conditions [9].



Figure 2.11: YUANXING PCB Mount Current Transformers

Application Method

Application circuit is as Figure 2.18, the secondary directly connect the sampling resistor in parallel to get the required voltage value.

Following Equation 2, where V_{out} is the voltage drop over the burden resistor I_{out} is the current flowing through the secondary winding R_{burden} is the value in ohms of the resistor closing the secondary circuit. In our case R_{burden} must be chosen so that V_{out} vary from 0.5 V. $I_{out}=10mA$, $V_{out}=0.5V$ and thus $R_{burden}=50\text{ ohm}$, the closest value in our possession is 100 ohm, we will just have to divide our software gain by two.

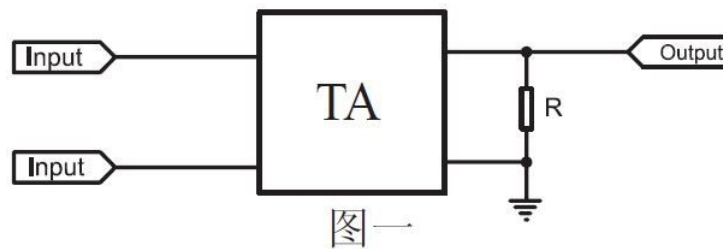


Figure 2.12: CT APPLICATION METHOD

2.3.2 Voltage Measurement

TV31 Voltage/ Current Transformer

When measuring voltage, it is necessary to construct an attenuation network of resistors in a voltage divider implementation, to accommodate the voltage level to the required input and limit the current flow.

YUANXING Voltage/current Transformer

The Yuanxing TV31 voltage/ current transformer is designed for applications where the primary AC signal must be transformed accurately into a lower secondary AC signal appropriate for micro-processor based circuit designs. These PCB mounted transformers are designed for harsh operating environments [10].

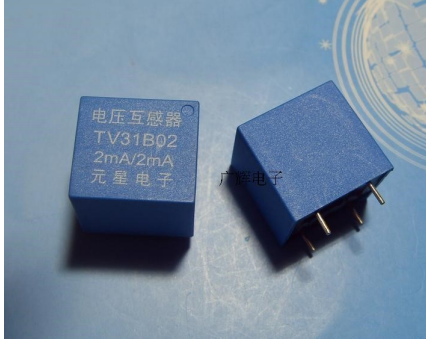


Figure 2.13: YUANXING PCB mount Voltage Transformers

Application Method

The TV31 voltage/ current transformer is in fact a 1:1 current transformer. An input voltage cannot be directly applied to the primary winding. A limiting resistor – R' – must be used in series with the primary input winding. The transformation of the input voltage signal is at the milli-ampere (mA) electric current level (the recommended operating condition is 2mA/2mA) [10]. In our case when the input voltage is 240V full scale, R' should be sized to be 120kohms, thus limiting the TV31 primary current to 2mA full scale.

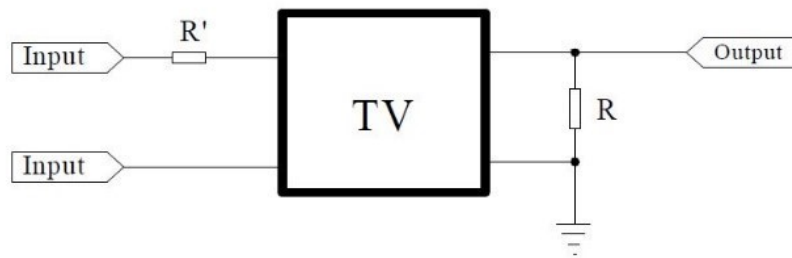


Figure 2.14: Voltage Transformer Application Method

R is put in parallel with the load to close the secondary winding circuit and obtains the voltage signal through a resistance – R – across the secondary output, knowing the desired output voltage, by ohms law.

$$V_{out} = I_{out} \times R$$

$I_{out} = 2mA, V_{out} = 0.5V$, thus $R = 250ohm$ As done earlier, we choose the closest resistor value in our possession (120 ohm) and modify our software gain.

2.4 Data Transmission

Sim808 GPS/GPRS module



Figure 2.15: SIM808 Top and Bottom View

Designed for global market, the SIM808 module is integrated with a high frequency GSM/GPRS engine, a GPS engine and a BT engine. The GSM/GPRS engine is a quad-band GSM/GPRS module that works on frequencies GMS 850 MHz, EGSM 900 MHz, DCS 1800 MHz, and PCS 1900 MHz. SIM808 features GPRS multi-slot class 12/ class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4. The GPS solution offers the best in- class acquisition and tracing sensitivity, Time-To-First-Fix (TTFF) and accuracy. With a tiny configuration of 24x24x2.6mm, the SIM808 can meet almost all the space requirements in user applications, such as M2M, smart phone, PDA, tracker and other mobile devices [11].

2.4.1 Sim808 interface

Interface settings

Between user Application and the Module, standardized UART interface is used for the communication, and default values for the interface settings as following: 115200bps, 8 bit data, no parity, 1 bit stop, no data stream control [12].

AT commands

AT commands are instructions used to control a modem. AT is the abbreviation of Attention. Every command line starts with "AT" or "at". That's why modem commands are called AT Commands. Many of the commands that are used to control wired dial-up modems, such as ATD (Dial), ATA (Answer), ATH (Hook control) and ATO (Return to online data state), are also supported by GSM/GPRS modems and mobile phones. Besides this common AT command set, GSM/GPRS modems and mobile phones support an AT command set that is specific to the GPS technology [13].

Test and initialization

To test our module we use the external UART interface of our breakout board (Figure 2.22) with a USB to serial (FTDI) using the Serial monitor of the Arduino

software. We set the baud rate do 115200 bps and write basic at commands to sim module.

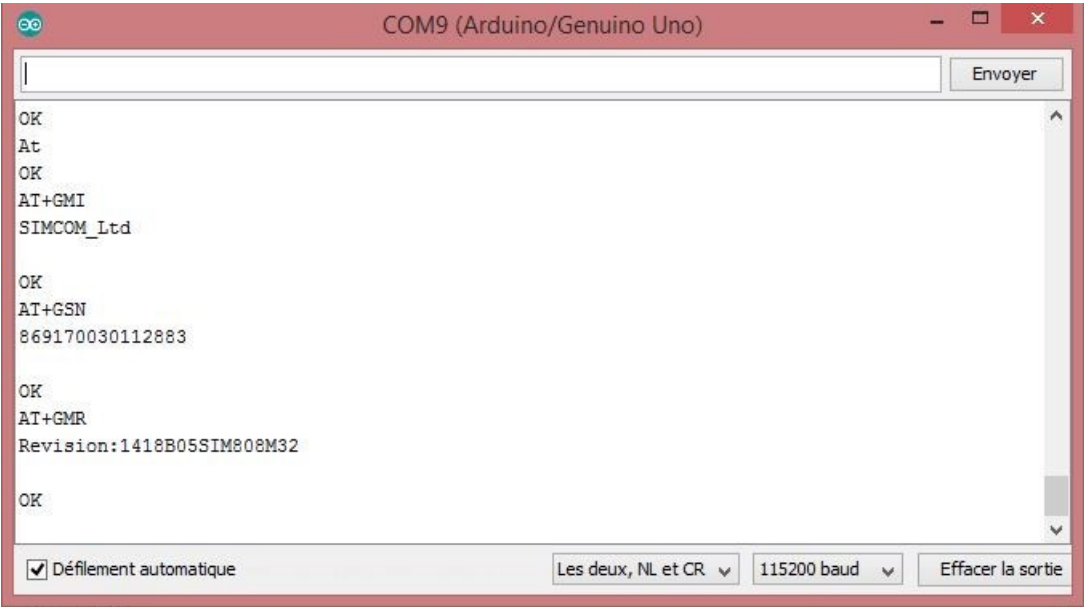


Figure 2.16: SIM808 Serial Interface

Sending At to the module gives OK on the serial, we tested further commands to ensure the correctness of the interface.

AT+GMI	Request manufacturer identification
AT+GMM	Request TA model identification
AT+GMR	Request TA revision identification of software release
AT+GOI	Request global object identification
AT+GSN	Request TA serial number identification

Table 2.1: Identification ATcommands Overview

GSM Test

After we ensured our module interface is working, we moved to the GSM related commands. We first check that our "ooredoo" sim card is well positioned and recognized by the module with the AT+CPIN command which give us a response of READY on success. For GSM we first test the call availability with the ATD command.

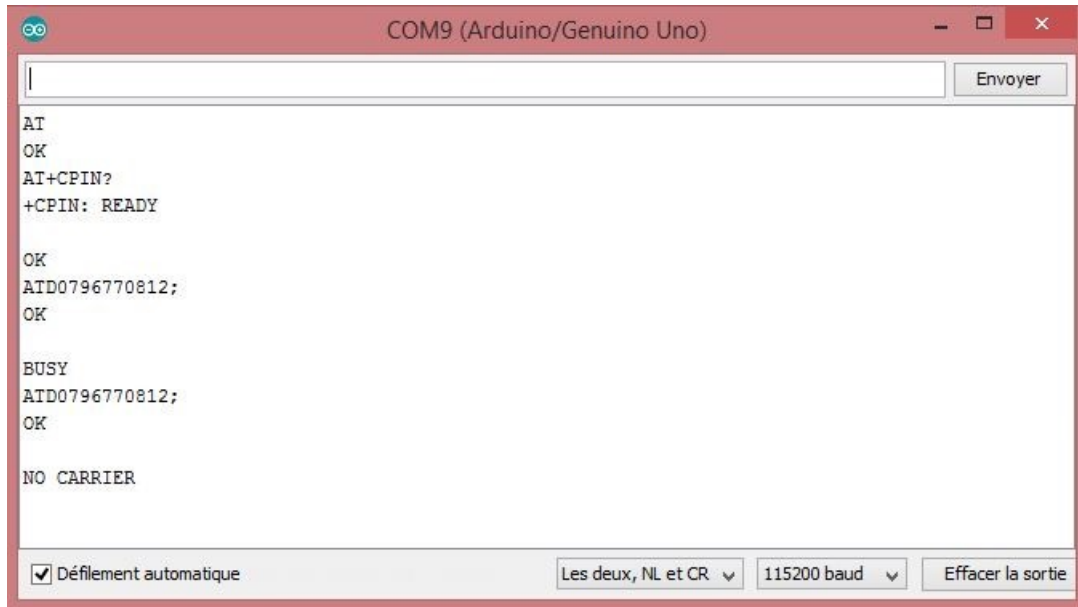


Figure 2.17: SIM808 Call initiation

As we can see in (Figure 2.23) we can dial a number with the ATD commands as we get OK as a response , BUSY if the number we are dialing hangout and NO CARRIER if the number is out of reach (or the GSM antenna is not placed).

GPRS configuration

To connect through GPRS to our server, we need to configure our sim module with the commands shown in Table 2.2.

AT command	Response	Description
AT	OK	test command. reply is OK
AT+CGATT?	+CGAT: n	checks if GPRS is attached ? n=1 if attached
AT+CIPMUX=n	OK	use 0 for single connection or use 1 for multiple connection
AT+CSTT="apn","username","pass"	OK	Set APN, username and password
AT+CIIR	OK	Brings up wireless connection
AT+CIFSR	ip address	Get local IP Address if connected
AT+CSTT="TYPE","domain","port"	Connected	Establishes a connection with a server. Type can be UDP or TCP
AT+CIPSEND	>	Sends data when the connection is established
AT+CIPCLOSE	OK	Closes the connection
AT+CIPSHUT	SHUT OK	Resets IP session if any

Table 2.2: GPRS related ATcommands Overview

We then configured our modem as follows: APN name, username and password can be found on phones configuration panels, in our case with our "ooredoo" sim card, the APN name is "internet" whereas the username and password are both "OOREDOO".

We intend to send our data through UDP, "5.189.189.207" is the address of our server and 5012 is the open distant port number. Figure 2.24 shows the GPRS configuration we tested through our serial monitor.

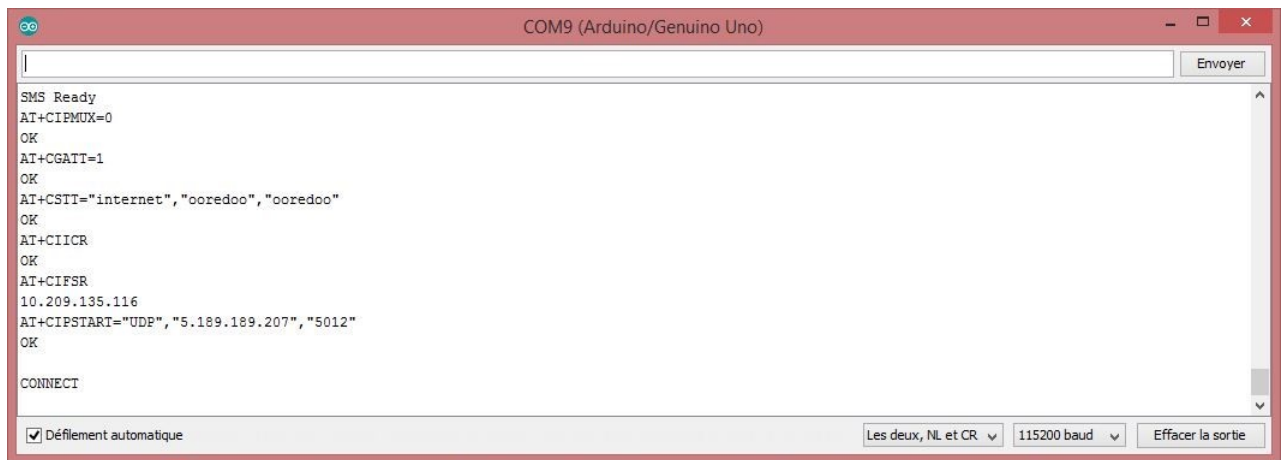


Figure 2.18: SIM808 GPRS UDP connection initiation

The device can connect to our server using GPRS without additional infrastructure costs. Transmissions of synchrophasor measurements starts immediately after

power up. Using a GPRS connection instead of wired Ethernet links removes the need for a dedicated networking infrastructure. We chose to send data though UDP instead of TCP to decrease the latency with which data is sent.

Data Frame

Our phasor measurement unit can directly communicate using the IEEE C37.118 protocol and be integrated with standard IEEE C37.118 compliant phasor data concentrators (PDCs) to reuse existing equipment. IEEE C37.118-2 specified a standard format for different types of the messages as depicted in Figure 2.25. Each message begins with identification and synchronization word (SYNC), followed by FRAMESIZE (total Bytes inside message), IDCODE (ID of the synchro phasor data source), SOC (Second Of Century count since epoch midnight 01.01.1970), FRACSEC (Fraction of Second and time quality), DATA (Depends on message type) and CHK (Cyclic Redundancy Check (CRC)). The content and structure of DATA field is different for different types of Messages [14].

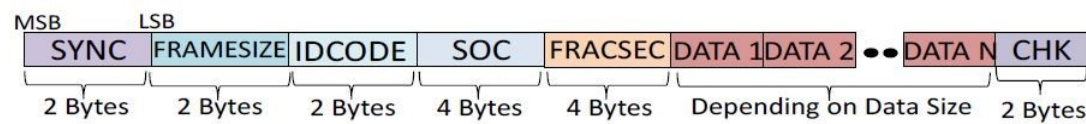


Figure 2.19: IEEE C37.118-2 standard message format

Each time data is updated, we modify the data frame to be sent to the server.

GPS commands

Table 3 shows GPS AT commands overview

Command	Description
AT+CGPSWR	GPS power control
AT+CGPSRST	GPS mode reset(hot/warm/cold)
AT+CGPSSTATUS	GPS current GPS status
AT+CGPSOUT	GPS NMEA data output control
AT+CGPSINF	GET current GPS location info

Table 2.3: GPS related ATcommands Overview

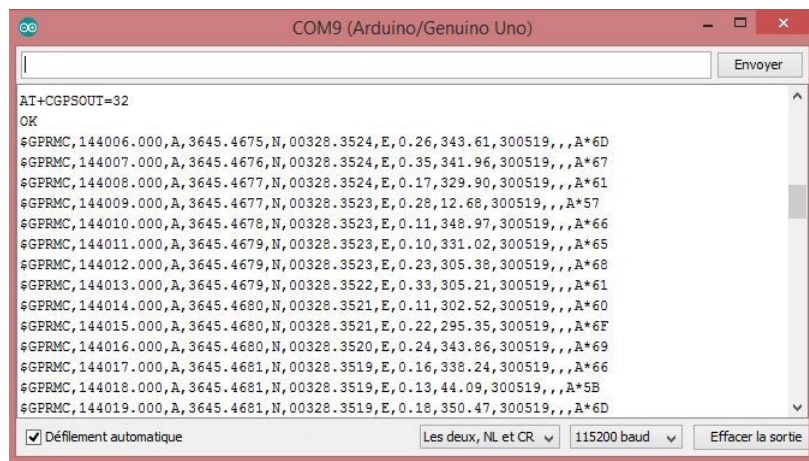
AT+CGPSPWR=1 turn GPS signal on , the following commands is used to reset and AT+CGPSSTATUS gives position status , if the signal is good enough , the module output a fixed location response.

NMEA Data

The National Marine Electronics Association (NMEA) has developed a specification that defines the interface between various pieces of marine electronic equipment. The standard permits marine electronics to send information to computers and to other marine equipment, GPS receiver communication is defined within this specification. Most computer programs that provide real time position information understand and expect data to be in NMEA format. This data includes the complete PVT (position, velocity, time) solution computed by the GPS receiver [15].

NMEA sentences

NMEA consists of sentences, the first word of which, called a data type, defines the interpretation of the rest of the sentence. Each Data type would have its own unique interpretation and is defined in the NMEA standard. RMC : NMEA has its own version of essential GPS PVT (position, velocity, time) data. It is called RMC, The Recommended Minimum [15]. AT+CGPSOUT=32, (2 to the power of 5) , returns the GPS NMEA sentence of GPRMC. Executing gps commands gave us the following result:



```
COM9 (Arduino/Genuino Uno)
AT+CGPSOUT=32
OK
$GPRMC,144006.000,A,3645.4675,N,00328.3524,E,0.26,343.61,300519,,A*6D
$GPRMC,144007.000,A,3645.4676,N,00328.3524,E,0.35,341.96,300519,,A*67
$GPRMC,144008.000,A,3645.4677,N,00328.3524,E,0.17,329.90,300519,,A*61
$GPRMC,144009.000,A,3645.4677,N,00328.3523,E,0.28,12.68,300519,,A*57
$GPRMC,144010.000,A,3645.4678,N,00328.3523,E,0.11,348.97,300519,,A*66
$GPRMC,144011.000,A,3645.4679,N,00328.3523,E,0.10,331.02,300519,,A*65
$GPRMC,144012.000,A,3645.4679,N,00328.3523,E,0.23,305.38,300519,,A*68
$GPRMC,144013.000,A,3645.4679,N,00328.3522,E,0.33,305.21,300519,,A*61
$GPRMC,144014.000,A,3645.4680,N,00328.3521,E,0.11,302.52,300519,,A*60
$GPRMC,144015.000,A,3645.4680,N,00328.3521,E,0.22,295.35,300519,,A*6F
$GPRMC,144016.000,A,3645.4680,N,00328.3520,E,0.24,343.86,300519,,A*69
$GPRMC,144017.000,A,3645.4681,N,00328.3519,E,0.16,338.24,300519,,A*66
$GPRMC,144018.000,A,3645.4681,N,00328.3519,E,0.13,44.09,300519,,A*5B
$GPRMC,144019.000,A,3645.4681,N,00328.3519,E,0.18,350.47,300519,,A*6D
☒ Défilement automatique
Les deux, NL et CR 115200 baud Effacer la sortie
```

Figure 2.20:

Where :

RMC	Recommended Minimum sentence C
144006	Fix taken at 14:40:06 UTC(time in hhhmmss format)
A	Status A=active or V=Void.
3645.4675,N	Latitude 36 deg 45.4675' N
0328.3524,E	Longitude 3 deg 28.3524' E
0.26	Speed over the ground in knots
343.61	Track angle in degrees True
300519	Date \ 30th of May 2019
6A	The checksum data , always begins with

Checking coordinates using an online NMEA decoder[16] to compare the coordinates we got from our module with our current position.

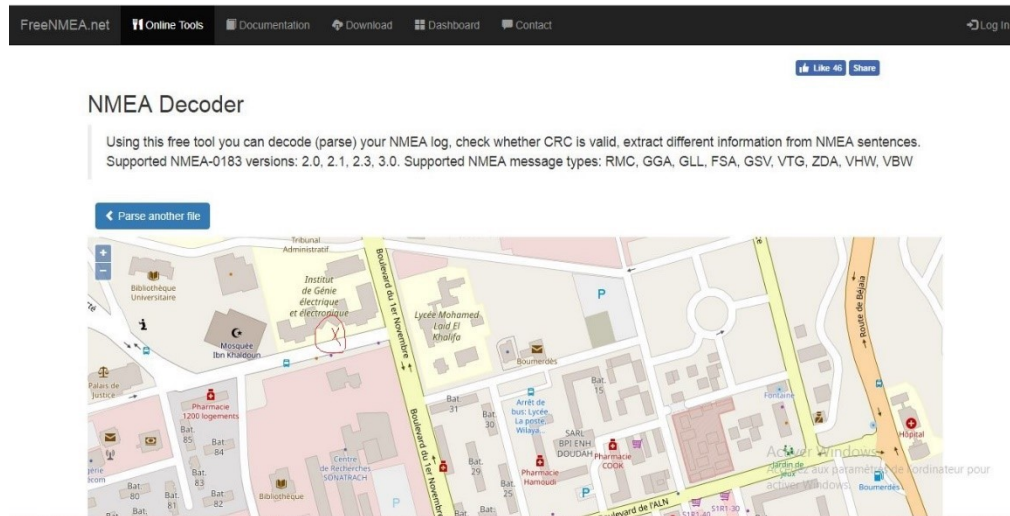


Figure 2.21: Map location of decoded NMEA coordiantes

With and external GPS antenna the coordinates we got are about 10 meters away from the exact location.

2.4.2 Sim808 C library

As done with the ADE7880, to optimize our coding we designed a dedicated library to configure and communicate with the Sim808 module. The library contains functions that are necessary for the UART communication as well as GPS and GPRS functions. We made the library available in Github.

```
#define CR  0x0D
#define LF  0x0A

void Sim808Power();
void GPS_Setup();
void Get_NMEA();
void Get_IP();
void GPRS_Config();
void Send_UDP();
void AT_Write(char *CMD);
char response_success();
void reset_buff();
```

The AT write functions is the main function used to communicate with the module, it simply writes the command as a string argument to the UART interface and adds the two end characters (NL and CR) necessary for the communication.

```
void AT_Write(char *CMD)
{
```



```

    UART1_Write_Text(CMD);
    UART1_Write(0x0D);
    UART1_Write(0x0A);
}

```

The response function checks UART receive buffer when data is ready (set by buffer full interrupt), returns true if the buffer contains the expected commands response and false if it doesn't.

```

char response_success()
{
    char result;
    while(!data_ready);
    if((strstr(rx_buff,"OK")) || (strstr(rx_buff,"CONNECT")) || (strstr(rx_buff,"re
        result = 1;
    }else result = 0;
    reset_buff();
    return result;
}

```

Each time a command needs to be executed on the SIM module, we write the commands using the AT_Write function and wait for response success to return true, the Figure 2.31 show how this is used.

```

void GPS_Setup(){

    memset(rx_buff,0,64);
    UART3_Write_Text("GPS Power On");
    UART3_Write_Text("Please wait...");
    AT_Write("AT+CGSPWR=1");
    //change the working mode to 1
    while(!response_success());

    UART3_Write_Text("GPS Cold Reset");
    UART3_Write_Text("Please wait...");
    AT_Write("AT+CGPSRST=0");
    //change the working mode to 1
    while(!response_success());
    UART3_Write_Text("GPS On !");
}

```

Same methods has been used for the other library functions (GPRS_Config, Get_Ip and Get_NMea). Rest of the code is presented on Appendix B.

2.5 GPS synchronization signal

Recent developments in the PMU has been made possible only because of easy available of the synchronizing pulses which are derived from the GPS modules. The GPS satellites have multiple number of Atomic clocks on board, which gives them the capability of accurately tracking time. Thanks to the low cost GPS modules being available now a days, anyone with a GPS module with a low price tag, can access this time source accurate to only a few microseconds. The GPS module for a PMU serves three purposes such as

- After a successful fix with at least three satellites, the GPS module provides a Pulse Per Second (PPS) signal, which is given to the MCU to generate the pulses to trigger sampling on the ADC.

- It also provides UTC, reference time received from the Satellites to the MCU according to which the calculated phasors are time Stamped and can be synchronized irrespective of their origin and the time delay which may incur between their transmission and reception at the Server Unit.

- Since a GPS module can also report the Geographical coordinates i.e. latitudes and Longitudes of itself, this data can be transmitted to the server where the location of the PMU is mapped to the map of the grid, and it can show a clear picture of the health status of the grid in the geographical area.

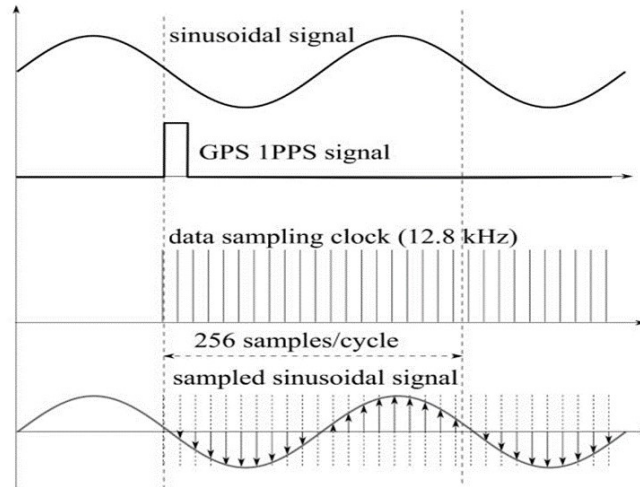


Figure 2.22: PPS synchronized data sampling

Our primary objective is to take 1 PPS signal (generated from GPS unit) and generate predefined set of equidistant pulses per second. According to which the sampling is initiated and the phasors are calculated and time stamped.

External Interrupt

As discussed previously, the ADE sampling initiation will be interrupt driven by the PPS signal, we thus connect the PPS output from the SIM module to the PB0 pin of the STM32 which will be used as a rising edge External Interrupt .

The PMU calculation and sampling update will be wrote on an ISR that maps to the vector address of our pin.

```

//DATA UPDATE ISR
void PPS_ISR() iv IVT_INT_EXTIO ics ICS_AUTO {

    EXTI_PR.B0=1; // clear flag

    pmu[0]=getVRMS(2);
    pmu[1]=getIRMS(2);
    pmu[2]=getVRMS(1);
    pmu[3]=getIRMS(1);
    pmu[4]=getVRMS(0);
    pmu[5]=getIRMS(0);
    pmu[6]=getVA(2);
    pmu[7]=getWatt(2);
    pmu[8]=getVA(1);
    pmu[9]=getWatt(1);
    pmu[10]=getVA(0);
    pmu[11]=getWatt(0);
    pmu[12]=getPhaseShift(2);
    pmu[13]=getPhaseShift(1);
    pmu[14]=getPhaseShift(0);
    pmu[15]=getPeriode(2);
    pmu[16]=getPeriode(1);
    pmu[17]=getPeriode(0);

}

```

2.6 Program flowchart

Figure 2.34 show the flowchart of the program ran on the ARM Cortex M3.

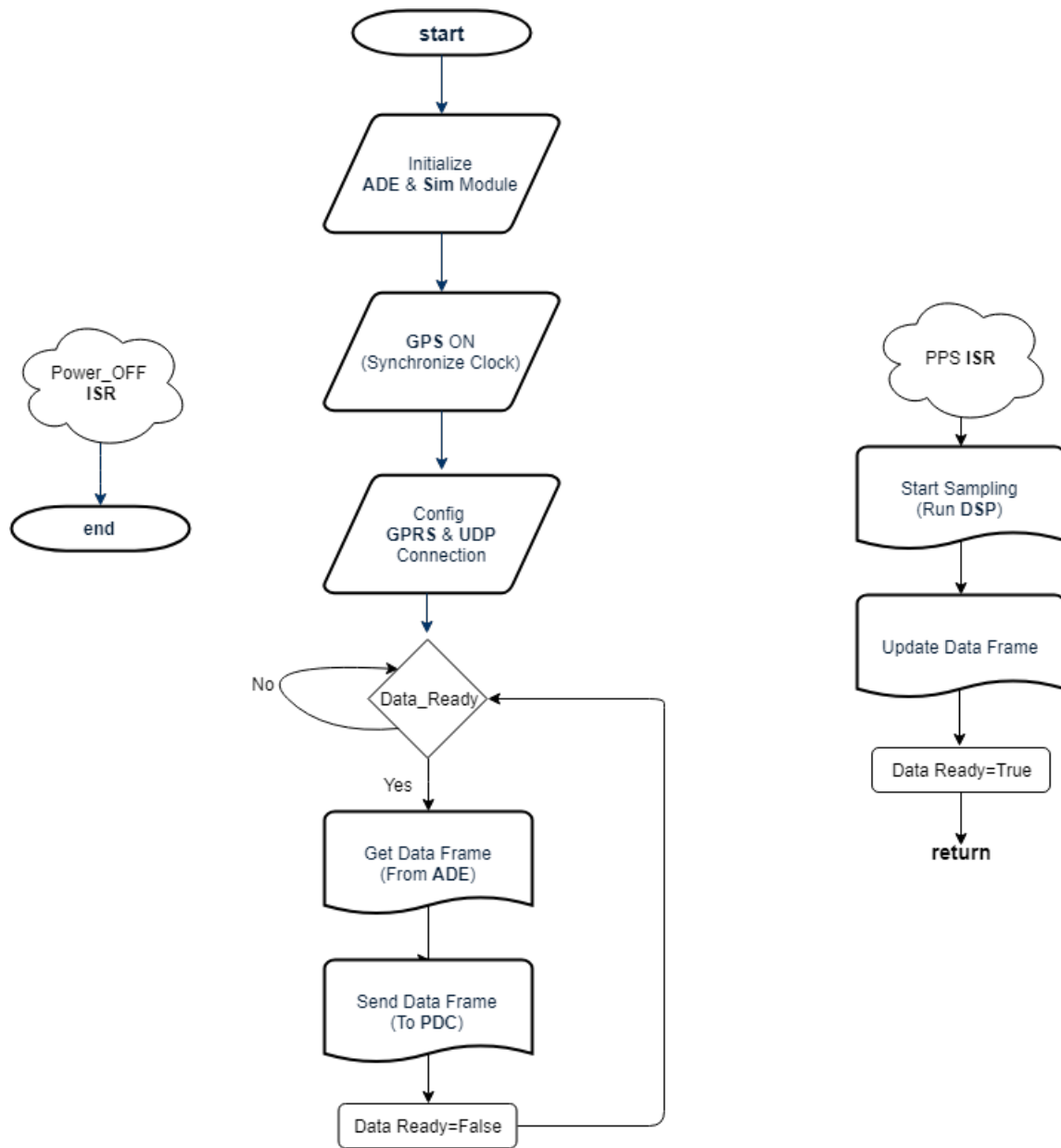


Figure 2.23: PMU STM32f103 Program Flowchart

2.7 PCB Design

2.7.1 Proteus ISIS

Proteus ISIS is a circuit designing and simulation software. ARES is used for designing PCB Layouts. First we make our required circuit on Proteus, test it, and then make the desired PCB layout in ARES.

2.7.2 Circuit Schematic Design

This section shows the main circuit and schematic diagrams used for our PCB design.

Power circuit

The power supply of the SIM808 range from 3.4V to 4.4V. The recommended voltage is 4.0V, the transmitting burst will cause voltage drop that's why we need a high current stable supply.

LM2576

The LM2576 series of regulators are integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving 3-A load with excellent line and load regulation. These devices are available in fixed adjustable output voltage [17]. To supply Both the ADE7880 and the STM32f103 we use the AMS1117 voltage regulator to step down the 12v input to 3.3v (Figure 2.35).

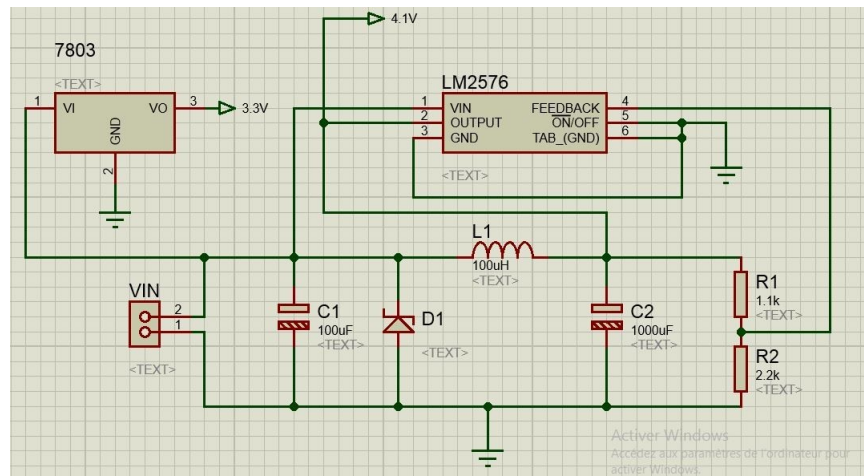


Figure 2.24: PMU Power Supply Proteus Circuit Diagram

Sim808 Schematic Diagram

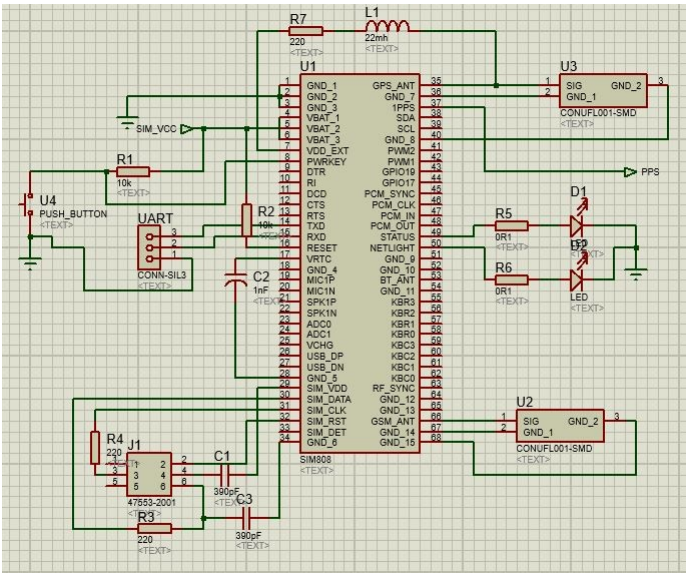


Figure 2.25: Sim808 Schematic Diagram

ADE7880 Functional Circuit

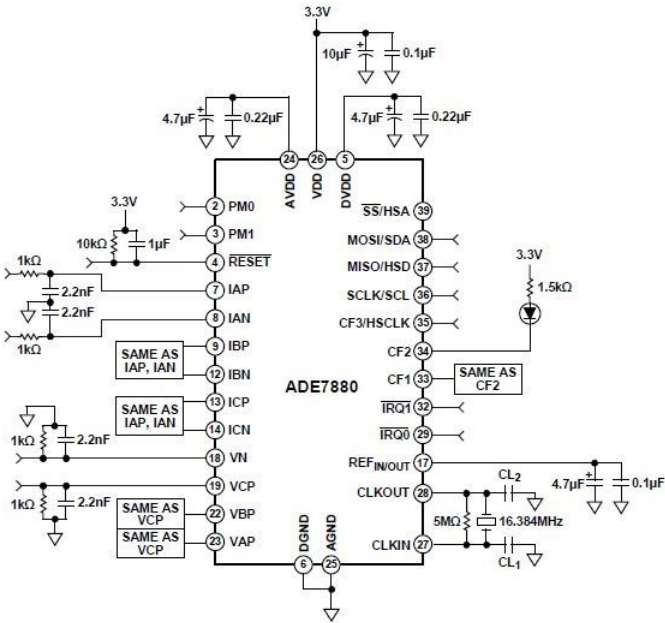


Figure 2.26: ADE7880 Functional Circuit

STM32f103C8 Schematic Diagram

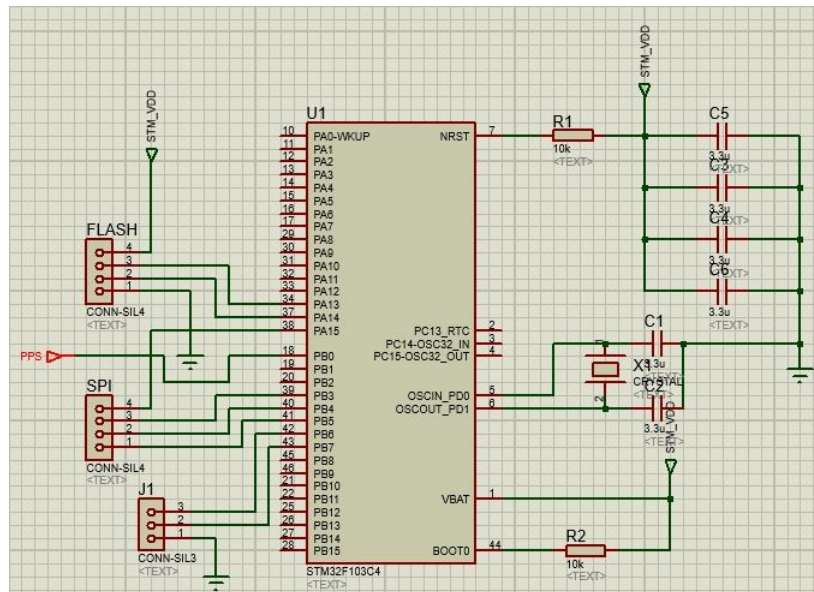


Figure 2.27: STM32f103C8 Schematic Diagram

Current & Voltage Sensors Proteus Schematic Diagram

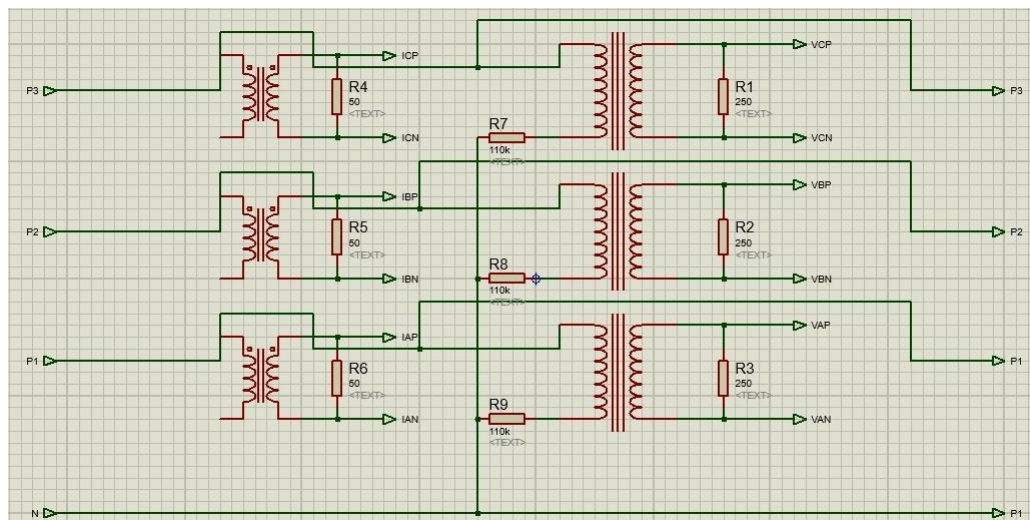


Figure 2.28: Conditioning Circuit Proteus Circuit Diagram

2.7.3 PCB Layout

Proteus PCB layout

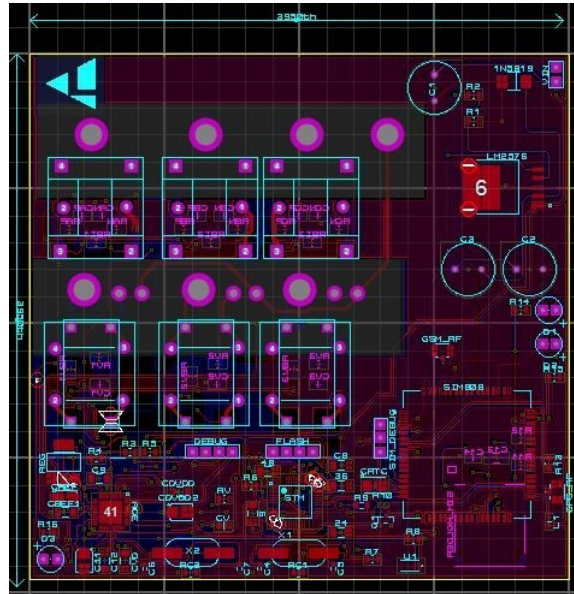


Figure 2.29: PMU Proteus PCB Layout

Printed Circuit Board



Figure 2.30: PMU PCB

Chapter 3

Real time monitoring system

The use of PMUs and synchrophasors allow the implementation of smart grid monitoring systems, this technologies enable the supervision of key grid metrics to develop a better understanding of the network and identify deteriorating system conditions. This section describes the software implementation of a data storage and visualization. The structure of such software is shown in figure 3.1 This structure can be used for any IOT application.

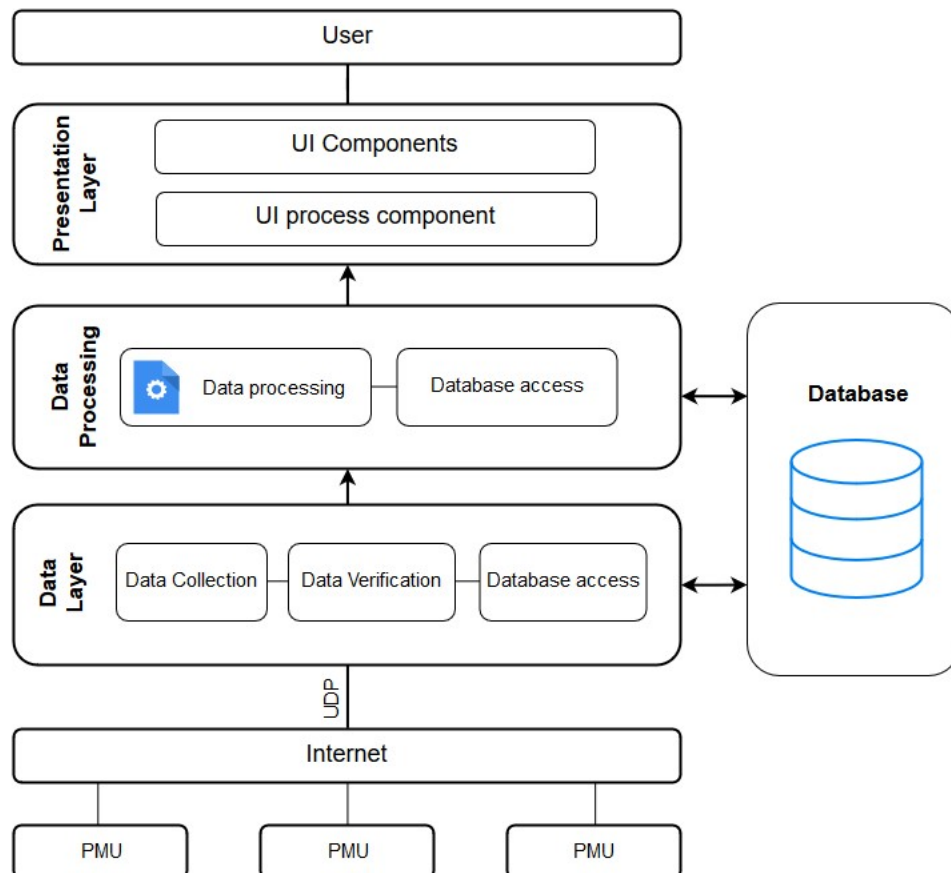


Figure 3.1: Real time monitoring system software structure

As shown in the figure 3.1 we form a node in the system where data from a number of PMUs is aggregated. The collected data is verified, stored, and processed depending on application requirements to retrieve for future use. The data can be accessed immediately from any web enabled device.

3.1 Data collection

In order to receive data from PMUs, a UDP server is created to receive datagrams coming from clients(PMUs). To achieve such set-up a UDP socket is set to wait for data. The received datagram is then processed on the server then it waits for another datagram arrival and repeats the process.

The socket is programmed using python socket interface, used at lower level to access basic socket support in the underlying operating system. The figure 3.2 shows the steps to implement a UDP socket server. First, the creation of a socket object that communicates with Internet Protocol v4 addresses, then binding it to a reserved port on the server IP address, finally, the reception of data.

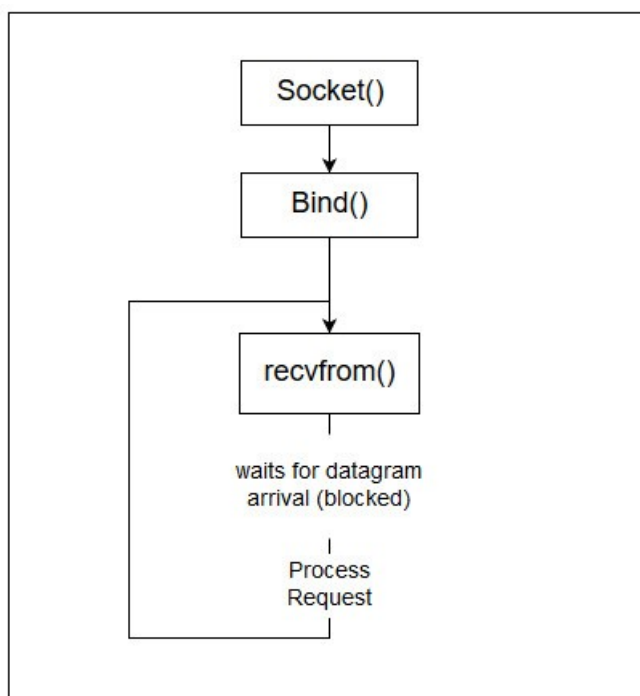


Figure 3.2: UDP server

3.2 Data verification

basic data verification and checking is performed on the data collected from PMUs. It includes checking the length of the frame, device ID authentication and performing a data integrity check namely cyclic redundancy. Data is then stored directly whereas corrupted data is flagged out before storage.

3.3 Data storage

The stream of grid metrics measurements originated from PMUs can generate an enormous amount of data. The server needs to handle this data and store it the most convenient way possible, taking in consideration speed, reliability and the fact that the data has a defined structure and is going to be used for visualization and future analysis.

SQL Database

SQL database follows a relational data model to store the data. In this model, data is stored in rows and columns in tabular form. Related tables can be interlinked together[18]. We are going to use MySQL relational database management system which. performance, facilitate interaction and ensure robustness.

3.3.1 Architecture

Knowing that this project is aimed for research and development purposes, the architecture design is focused on two main pillars which are simplicity, and Scalability for future system growth. The Architecture consist of two tables, the first one is filled with all the connected PMUs' information such as location (latitude and longitude) and IMEI number used for data authentication, these data are stored the first time a PMU is connected to the database. The second table stores the real time data generated from the PMUs after assigning an Id and a timestamp tag. Figure 3.3 shows an ERD for the database used for this project.

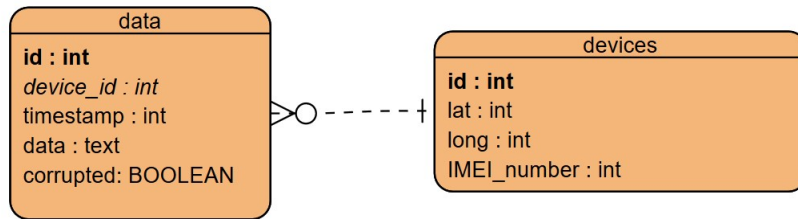


Figure 3.3: ERD of the database

3.4 Data processing

Data processing occurs according to application and user requirements, in this case visualization. After checking these requirements, the software request the required data from the database, it is then segmented and export it to a convenient format. This part is is challenging because the software has to have the ability to take this amount of data and process it very fast to achieve a real-time display with minimal delay.

3.5 Synchrophasor data visualization

Efficient monitoring system requires the proper visualization and support tools. these tools must represent the key measured metrics in real time and should be up- dated dynamically with the minimum delay. Figure 3.4 shows the display webpage.

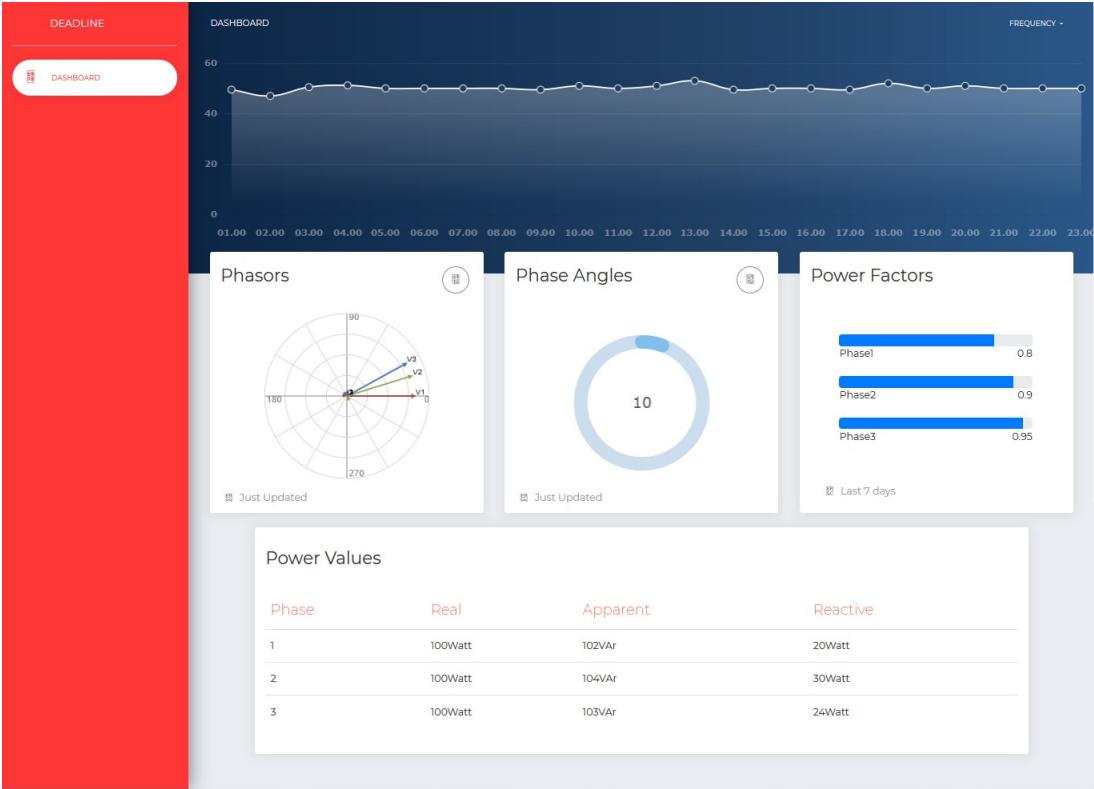


Figure 3.4: Real time display webpage

The most important measured values are displayed on the dashboard using different UI components, to facilitate the the understanding of the system state, and the interaction with the different UI elements. Next we are going to present these UI element.

Line chart

On this chart three of the main measured values are represented, The RMS value of current and voltage as well as the frequency which should always be kept close to a nominal value (50Hz in Algeria) within allowable tolerance. Figure 3.5 shows the chart displaying RMS current. These value are updated dynamically and the user can switch between them anytime.



Figure 3.5: Line chart display

Phasor diagram

This chart gives a visual representation to the phase and magnitude of the voltage and current of each phase. This chart is also updated dynamically.

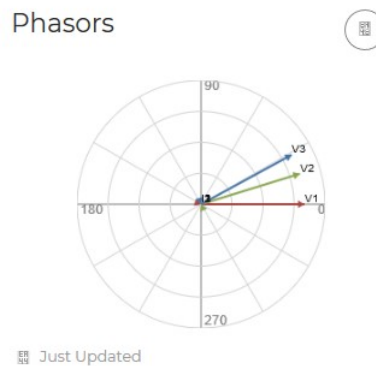


Figure 3.6: Phasor diagram display

Radial Gauge chart

One other important value to monitor is the phase angle. It is represented on radial gauge chart and is also updated dynamically.



Figure 3.7: phase angle display

Linear Gauge charts

On this chart, we represent the power factor in order to get an overall picture of energy usage and planning efficiency measures.

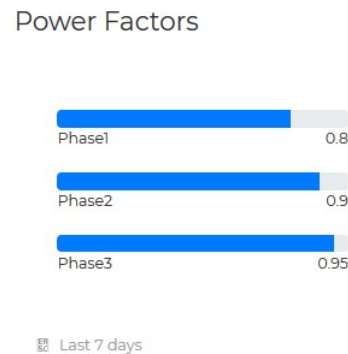


Figure 3.8: power factor display

Table

The last UI element is a table display of the three phase’s measured power values.

Power Values			
Phase	Real	Apparent	Reactive
1	100Watt	102VAr	20Watt
2	100Watt	104VAr	30Watt
3	100Watt	103VAr	24Watt

Figure 3.9: power values display

In the future when there are more than one PMU sending data, A PMU can be set as reference and compared to the others to extract more useful information and achieve a wide area monitoring system.

Chapter 4

Experimental Results and Discussion

This section describes the experimental setup and the results obtained from the developed Laboratory prototype PMU.

4.1 Standard Requirements

To meet the requirements of the IEEE. C37.118 standard our PMU shall be capable of receiving time from a reliable and accurate source and keep the total vector error (TVE) and the frequency error (FE) within the required limits.

Frequency Error Calculation

The measure of error between the theoretical frequency and the measured frequency for the given instant of time.

$$|f_{true} - f_{measured}| = |\Delta f_{true} - \Delta f_{measured}| \quad (4.1)$$

Total Vector Error Calculation

The TVE was the quality indicator introduced in the IEEE. C37.118-2005, and since then has been included in the subsequent version of the standard. TVE is defined as the percentage magnitude of a vector difference between the measured and actual phasors treated as vectors (Figure 4.). TVE is defined in Equation 4.2.

$$TVE(n) = \sqrt{\frac{(\hat{X}_r(n) - X_r(n))^2 + (\hat{X}_i(n) - X_i(n))^2}{(X_r(n))^2 + (X_i(n))^2}} \quad (4.2)$$

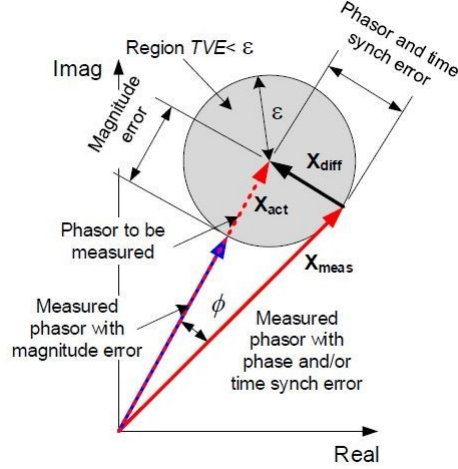


Figure 4.1: Graphical representation of the TVE. Based on IEEE C37.118

Where $\hat{X}_r(n)$ and $\hat{X}_i(n)$ are the sequences of estimates given by the unit under test, and $X_r(n)$ and $X_i(n)$ are the sequences of theoretical values of the input signal at the instants of time (n) assigned by the unit to those values [1].

4.2 Results & Evaluation

Data Sampling GPS Synchronization

When the SIM808 GPS location is fixed, the “PPS” output on pad 37 toggles once per second once the device starts receiving the GPS data and thus we can use this signal to synchronize our system clock to the GPS clock.

Measurement Test Conditions

To calculate the measurement efficiency of our implementation, we connected a variable voltage transformer from a 240 V 50 Hz outlet to our circuit and measured frequency and phasors for different through voltage in the range of 0 to 240V with a purely resistive load, we then calculated the average error for both frequency (FE) and total vector error TVE. Figure 4.2 shows our experimental setup

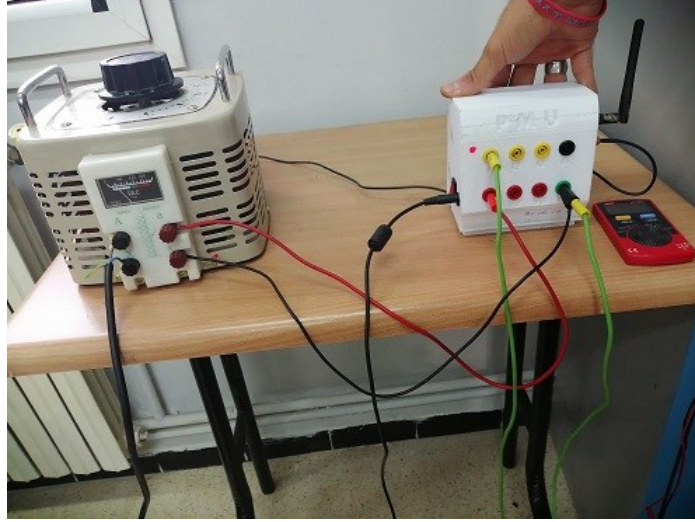


Figure 4.2: PMU Test Implementation

Test Results

Table 4.1 shows the measurement results.

Test Conditions F nominal = 50 Hz Purely Resistive Load From 0 to 100% rated magnitude	Average FE = 0.002 Hz		Max FE = 0.005 Hz
	Voltage	Current	
	Average TVE = 0.833 %	Average TVE = 0.667 %	Max TVE = 1%

Table 4.1: PMU Test Results

As we the reported error is well within the IEEE Standard compliance range of 1 % of the actual TVE and the 0.005 Hz FE allowed. Test code and test results are shown in Appendix E and F respectively.

Chapter 5

Conclusion

PMUs are going to be the basic building blocks for monitoring the Smart Grid of the future. With the increase in the number of active PMUs in the Electric Grids day by day, the Real-Time monitoring of the health of the grid is going to be a reality sooner than expected. Since a lot of manufacturers are going to build their own versions of PMU, the much needed IEEE Standard C37.118.x.2014 is definitely a welcome guidance to make the different PMUs compatible with each other and with the PDCs.

The goal of developing a low cost PMU was not to compete with other manufacturers who provide commercial PMUs, but to facilitate the research in the academics and the development organizations which may incorporate the data from our PMUs to design and simulate the various projects related to a smart Electrical Grids. The objective was to simplify the hardware implementation process of Phasor measurements such that a PMU can be built using the many economic hardware software computing platforms available now a days. We can say that our goal concerning that regard was achieved, as we succeeded in the realization of a very low-cost prototype that is well within the IEEE Standard compliance of PMUs.

Further work As for future work, we would like to improve our PMU to the P class and enhance its performance user control over the grid. We would also like to integrate a prediction algorithm as well as an anomaly detection algorithm using data measured during the process.

Bibliography

- [1] IEEE Power Energy Society, IEEE Std C37.118.1-2011.IEEE Standard for Synchrophasor Measurements for Power Systems, vol. 2011, no. December. 2011.
- [2] H. Lehpamer, Introduction to Power utility communications. Artech House, 2016.
- [3] New technology can improve electric power system efficiency and reliability- Today in Energy - U.S. Energy Information Administration (EIA).www.eia.gov.
- [4] IEEE Power and Energy Society, IEEE Std C37.118.2-2011 -IEEE Standard for Synchrophasor Data Transfer for Power Systems, vol. 2011, no. December. 2011.
- [5] Phasor Measurement (Estimation) Units,Dr. Anurag K.Srivastava,Assistant Professor, The School of Electrical Engineering and Computer ScienceDirector, Smart Grid Demonstration and Research Investigation Lab.
- [6] ANALOG DEVICES, "Polyphase Multifunction Energy Metering IC with Harmonic Monitoring" , ADE7880 datasheet [Rev c].
- [7] STMicroelectronics, "Medium-density performance line ARM®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces", STM32F103x8 [August 2015] .
- [8] Texas Instrument, Accessed March 2019, Available at : <https://www.ti.com/devnet/docs/catalog>.
- [9] Current Transformers for Measurement YUANXING Electronics Co. Ltd. datasheet.
- [10] TV31 PCB Mounted Voltage/Current Transformers YUANXING Electronics Co., LTD datasheet.
- [11] SIM808_ Hardware Design_V1.00 Copyright c Shangai SIMCom Wireless Solutions Ltd. 2014.
- [12] SIMCOM,SIM800 Series_AT Command Manual_V1.09.
- [13] www.developershome.com, Accessed April 2019 , Available at: <https://www.developershome.com/sms/atCommandsIntro.asp>.

- [14] IEEE C37.118-2 Synchrophasor Communication Framework. Overview, Rafi-ullah Khan, Kieran McLaughlin, David Lavery and Sakir Sezer, Queen's University Belfast, Belfast, U.K.
- [15] www.gpsinformation.org, Accessed May 2019, Available at: <https://www.gpsinformation.org/dale/nmea.htm>.
- [16] <https://rl.se/gprmc>.
- [17] LM2576xx Series SIMPLE SWITCHER® 3-A Step-Down Voltage Regulator Texas Instrument datasheet.
- [18] Rautmare, Sharvari, and D. M. Bhalerao. "MySQL and NoSQL Database Comparison for IoT Application." 2016 IEEE International Conference on Advances in Computer Applications (ICACA), 2016.

Appendices

Appendix A

ADE Library

NB: we made the library which is compatible with all the ADE78xx family available on GitHub at: <https://github.com/NazimBLADE7880>

```
3  unsigned char ADE_Read8(unsigned int reg){
4
5      unsigned char b0,r1,r0;
6      r0=(unsigned char)(reg & 0xFF);
7      r1=(unsigned char)(reg >> 8);
8      GPIOA_ODRbits.ODR15=0;
9      Delay_us(10);
10     SPI_Write(0x01);
11     SPI_Write(r1);
12     SPI_Write(r0);
13     b0=SPI_Read(0x00);
14     GPIOA_ODRbits.ODR15=1;
15     Delay_us(10);
16     return b0;
17 }
18 unsigned int ADE_Read16(unsigned int reg){
19
20     unsigned char b0,r1,r0;
21     r0=(unsigned char)(reg & 0xFF);
22     r1=(unsigned char)(reg >> 8);
23     GPIOA_ODRbits.ODR15=0;
24     Delay_us(10);
25     SPI_Write(0x01);
26     SPI_Write(r1);
27     SPI_Write(r0);
28     b1=SPI_Read(0x00);
29     b0=SPI_Read(0x00);
30     GPIOA_ODRbits.ODR15=1;
31     Delay_us(10);
32
33     return (unsigned int)b1<<8 | (unsigned int)b0;
34 }
35
36 unsigned long ADE_Read32(unsigned int reg){
37
38     unsigned char b3,b2,b1,b0;
39     char r3,r2,r1,r0;
40
41     r0=(unsigned char)(reg & 0xFF);
42     r1=(unsigned char)(reg >> 8);
43     GPIOA_ODRbits.ODR15=0;
44     Delay_us(10);
45
46     SPI_Write(0x01);
47     SPI_Write(r3);
48     SPI_Write(r2);
49     SPI_Write(r1);
50     SPI_Write(r0);
51     b3=SPI_Read(0x00);
52     b2=SPI_Read(0x00);
53     b1=SPI_Read(0x00);
54     b0=SPI_Read(0x00);
55
56     GPIOA_ODRbits.ODR15=1;
57     Delay_us(10);
58
59     return ((unsigned long)b3<<24 | (unsigned long)b2<<16 | (unsigned long)b1<<8 | (unsigned long)b0);
60 }
```

```

void ADE_Write32(unsigned int reg, unsigned long dat){
    char b0,b1,b2,b3;
    char r1,r0;

    b0=(unsigned char)(dat & 0xFF);
    b1=(unsigned char)(dat >> 8);
    b2=(unsigned char)(dat >> 16);
    b3=(unsigned char)(dat >> 24);

    r0=(unsigned char)(reg & 0xFF);
    r1=(unsigned char)(reg >> 8);

    ADE_SpiMode(0);

    GPIOA_ODRbits.ODR15=0;
    Delay_us(10);

    SPI_Write(0x00);
    SPI_Write(r1);
    SPI_Write(r0);
    SPI_Write(b3);
    SPI_Write(b2);
    SPI_Write(b1);
    SPI_Write(b0);

    GPIOA_ODRbits.ODR15=1;
    Delay_us(10);
}

79 void ADE_Write8(unsigned int reg ,unsigned char dat){
80
81     char r1,r0;
82     r0=(unsigned char)(reg & 0xFF);
83     r1=(unsigned char)(reg >> 8);
84
85     GPIOA_ODRbits.ODR15=0;
86     Delay_us(10);
87     SPI_Write(0x00);
88     SPI_Write(r1);
89     SPI_Write(r0);
90     SPI_Write(dat);
91     GPIOA_ODRbits.ODR15=1;
92     Delay_us(10);
93 }

```

```

void ADE_Write24(unsigned int reg, unsigned long dat){

    char b0,b1,b2;
    char r1,r0;

    b0=(unsigned char)(dat & 0xFF);
    b1=(unsigned char)(dat >> 8);
    b2=(unsigned char)(dat >> 16);

    r0=(unsigned char)(reg & 0xFF);
    r1=(unsigned char)(reg >> 8);

    ADE_SpiMode(0);
    GPIOA_ODRbits.ODR15=0;
    Delay_us(10);

    SPI_Write(0x00);
    r0=(unsigned char)(reg & 0xFF);
    r1=(unsigned char)(reg >> 8);
    SPI_Write(b2);
    SPI_Write(b1);
    SPI_Write(b0);

    GPIOA_ODRbits.ODR15=1;
    Delay_us(10);
}

```

Appendix B

SIM808 Library

```
void GPRS_Config(){

    char ap_cnt = 0;
    memset(rx_buff,0,64);
    UART3_Write_Text("Setting STA mode");
    UART3_Write_Text("Please wait...");
    AT_Write("AT+CHMODE=1");
    while(!response_success());
    UART3_Write_Text("Setting connection mode");
    UART3_Write_Text("Please wait...");
    AT_Write("AT+CIPMUX=0");
    while(!response_success());

    UART3_Write_Text("Setting GPRS connection");
    UART3_Write_Text("Please wait...");
    AT_Write("AT+CGATT=1");
    while(!response_success());

    while(1)
    {
        UART3_Write_Text("Connect to ISP");
        UART3_Write_Text("Please wait...");
        UART1_Write_Text("AT+CSSTT=");
        UART1_Write("");
        UART1_Write_Text(APN);
        UART1_Write("");
        UART1_Write(',');
        UART1_Write("");
        UART1_Write_Text(APN_user);
        UART1_Write("");
        UART1_Write(',');
        UART1_Write("");
    }

    UART1_Write("");
    UART1_Write(' ');

    UART3_Write_Text("Setting GPRS connection");
    UART3_Write_Text("Please wait...");
    AT_Write("AT+CIIICR");
    while(!response_success());

    UART3_Write_Text("Connected start sending");
}

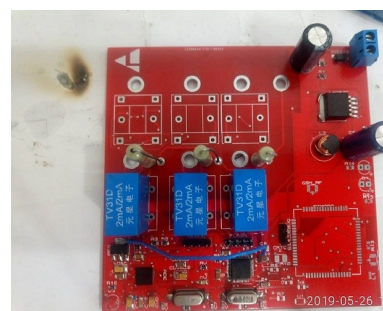
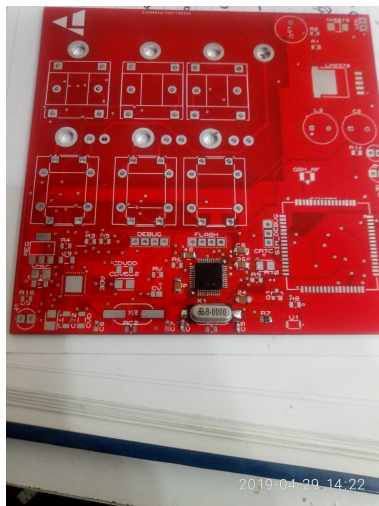
void Send_UDP(){

    UART3_Write_Text("Configure UDP connection\n\r");
    UART3_Write_Text("Please wait...\n\r");
    UART1_Write_Text("AT+CIPSTART=");
    UART1_Write("UDP");
    UART1_Write(',');
    UART1_Write("");
    UART3_Write_Text("PDC_IP");
    UART1_Write("");
    UART1_Write(',');
    UART1_Write("");
    UART3_Write_Text("UDP_port");
    UART1_Write("");
    UART1_Write(CR);
    UART1_Write(LF);
    while(!response_success());
}

void Get_IP()
{
    char response_cnt, conn_flg;
    UART3_Write_Text("Getting IP Address\n\r");
    UART3_Write_Text("Please wait...\n\r");
    AT_Write("AT+CIFSR");
    while(!data_ready);
    reset_buff();
    while(!data_ready);
    //IP address is stored in rx_buff
    UART3_Write_Text(rx_buff);
    reset_buff();
}
```


Appendix C

PCB Design



Appendix D

Secondary Components

Schematics

GPS External Antenna Circuit

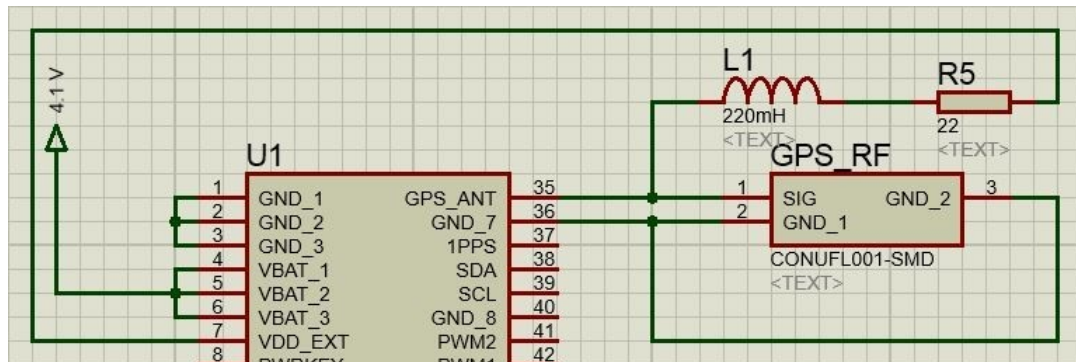


Figure D.1: Proteus Schematic Diagram

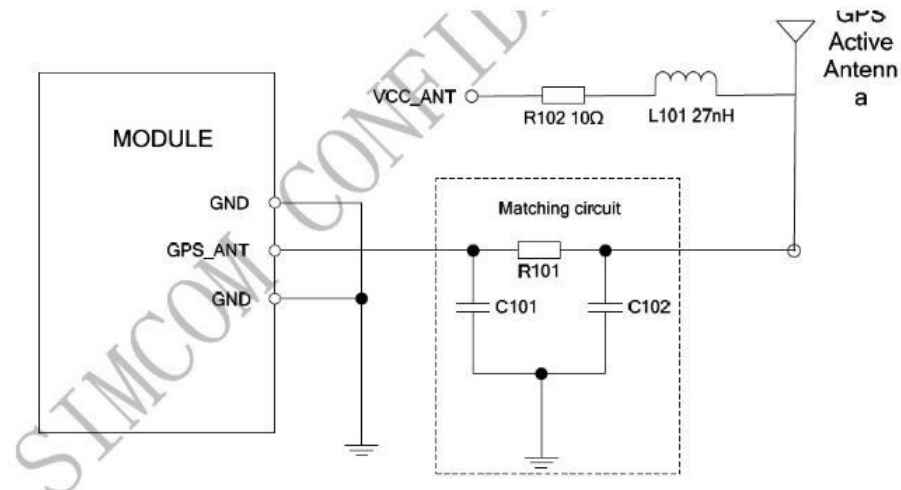


Figure D.2: Datasheet Schematic Diagram

SIM Holder GSM Antenna Circuit

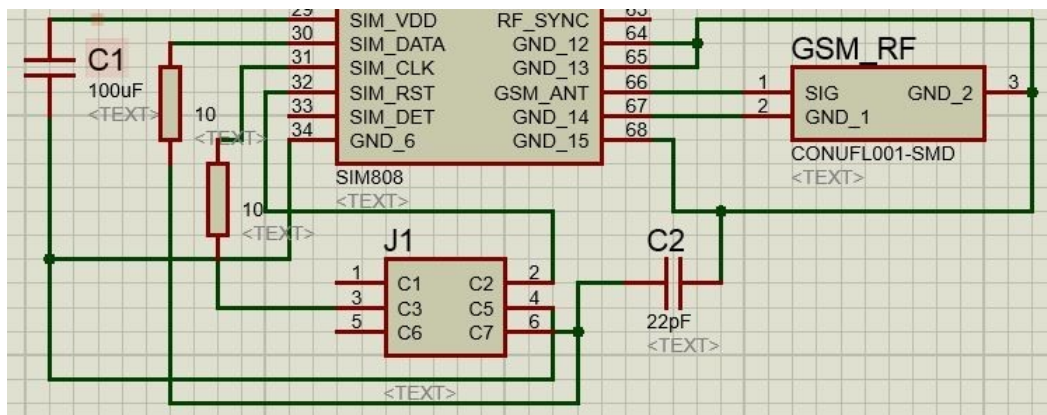


Figure D.3: Proteus Schematic Diagram

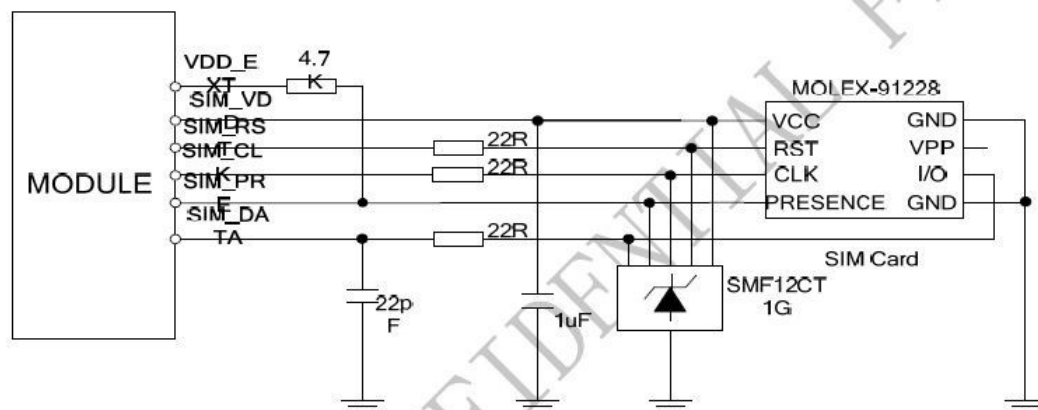


Figure D.4: Datasheet Schematic Diagram

Pin name	Signal	Description
C1	SIM_VDD	SIM card power supply
C2	SIM_RST	SIM card reset
C3	SIM_CLK	SIM card clock
C3	GND	Connect to GND
C5	GND	Connect to GND
C6	VPP	Not connect
C7	SIM_DATA	SIM card data I/O
C8	SIM_PRE	Detect SIM card presence

Figure D.5: Molex Sim Card Holder Pin Definition

SIM808 Power Key

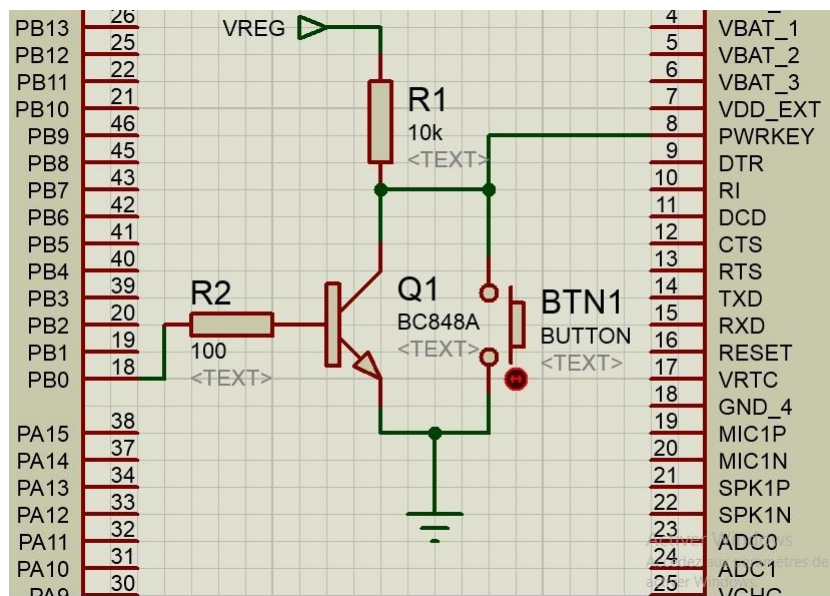


Figure D.6: Proteus Schematic Diagram

Appendix E

Experiment Test Code

```
2  #include "ade7880.h"
3  char take='b',buff='a',take1,buff1;
4  char val1[8],val2[4];
5  void ReadVI();
6  void SpiSetup();
7  char v=0;
8  char txt[17];
9  char txt3[7];
10 unsigned long checksum=0;
11 unsigned int angle=0;
12 unsigned int status=0;
13
14 void main() {
15
16     GPIO_Digital_Output(&GPIOC_BASE, _GPIO_PINMASK_13);
17     SpiSetup();
18     UART1_Init_Advanced(19200, _UART_8_BIT_DATA, _UART_NOPARITY, _UART_ONE_STOPBIT, &_GPIO_MODULE_USART1_PB67);
19     Delay_ms(400);
20     ADE_Init();
21     ADE_SpiEnable();
22
23     while(1){
24
25         checksum=ADE_Read32(CHECKSUM);
26         LongLongUnsignedToHex(checksum, txt);
27         UART1_Write_Text("checksum: ");
28         UART1_Write_Text(txt);
29         UART1_Write('\n');
30         UART1_Write('\r');
```

```
32     checksum=getIRMS(0);
33     LongLongUnsignedToHex(checksum, txt);
34     UART1_Write_Text("current: ");
35     UART1_Write_Text(txt);
36     UART1_Write('\n');
37     UART1_Write('\r');
38     checksum=getVRMS(0);
39     LongLongUnsignedToHex(checksum, txt);
40     UART1_Write_Text("voltage: ");
41     UART1_Write_Text(txt);
42     UART1_Write('\n');
43     UART1_Write('\r');
44     checksum=getWatt(0);
45     LongLongUnsignedToHex(checksum, txt);
46     UART1_Write_Text("active power: ");
47     UART1_Write_Text(txt);
48     UART1_Write('\n');
49     UART1_Write('\r');
50     checksum=getVA(0);
51     LongLongUnsignedToHex(checksum, txt);
52     UART1_Write_Text("apparent power: ");
53     UART1_Write_Text(txt);
54     UART1_Write('\n');
55     UART1_Write('\r');
56     angle=getPhaseShift(0);
57     IntToStr(angle, txt3);
58     UART1_Write_Text("phase: ");
59     UART1_Write_Text(txt3);
60     UART1_Write('\n');
61     UART1_Write('\r');
62     Delay_ms(1000);
63 }
64 }
```

Appendix F

Experiment Test Results

Voltage (V)		Current (A)		Phase shift (DEG)		frequency (Hz)	
Real	Measured	Real	Measured	Real	Measured	Real	Measured
2.50	2.50	0.100	0.102	0.000	0.425	50.000	50.019
5.00	5.02	0.200	0.203	0.000	0.733	50.000	50.052
10.0	10.2	0.700	0.705	0.000	0.514	50.000	50.022
20.0	20.3	1.200	1.207	0.000	0.844	50.000	50.013
40.0	40.4	1.700	1.708	0.000	0.861	50.000	50.009
50.0	50.5	2.300	2.312	0.000	0.941	50.000	50.011
70.0	70.6	2.800	2.851	0.000	0.954	50.000	50.043
100.0	101.4	3.600	3.705	0.000	1.564	50.000	50.004
110.0	111.4	3.900	4.035	0.000	1.236	50.000	49.984
140.0	141.8	4.100	4.241	0.000	1.487	50.000	49.988
180.0	181.9	5.600	5.780	0.000	1.654	50.000	50.010
200.0	202.0	6.200	6.389	0.000	1.981	50.000	50.019
210.0	212.1	6.500	6.692	0.000	1.954	50.000	50.013
220.0	222.3	6.900	7.101	0.000	2.481	50.000	50.041
240.0	242.5	7.300	7.505	0.000	2.654	50.000	50.003

Table F.1: