FEBRUARY 2022

M.Sc. in Electrical and Electronics Engineering

UĞUR POLAT

REPUBLIC OF TURKEY
GAZİANTEP UNIVERSITY
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

DEVELOPMENT OF EMBEDDED SYSTEM FOR ENERGY ANALYZER

M.Sc. THESIS
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

BY
UĞUR POLAT
FEBRUARY 2022

DEVELOPMENT OF EMBEDDED SYSTEM FOR ENERGY ANALYZER

M.Sc. Thesis
in
Electrical and Electronics Engineering
Gaziantep University

Supervisor

Assist. Prof. Dr. Nurdal WATSUJİ

by
Uğur POLAT
February 2022

©2022[Uğur POLAT]

DEVELOPMENT OF EMBEDDED SYSTEM FOR ENERGY ANALYZER

submitted by **Uğur POLAT** in partial fulfillment of the requirements for the degree of Master of Science in **Electrical and Electronics Engineering, Gaziantep University** is approved by,

Prof. Dr. Mehmet İshak YÜCE
Director of the Graduate School of Natural and Applied Sciences…………............

Prof. Dr. Ergün ERÇELEBİ
Head of the Department of Electrical and Electronics Engineering…………............

Assist. Prof. Dr. Nurdal WATSUJİ
Supervisor, Electrical and Electronics Engineering
Gaziantep University …………............

Graduation Date: 03 February 2022

Examining Committee Members:

Assist. Prof. Dr. Nurdal WATSUJİ
Thesis Supervisor, Electrical and Electronics Engineering
Gaziantep University …………............

Prof. Dr. Arif NACAROĞLU
Electrical and Electronics Engineering
Gaziantep University …………............

Assist. Prof. Dr. Noor Baha ALDİN
Electrical and Electronics Engineering
Hasan Kalyoncu University …………............

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Uğur POLAT

ABSTRACT

DEVELOPMENT OF EMBEDDED SYSTEM FOR ENERGY ANALYZER

POLAT, Uğur
MSc. in Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Nurdal WATSUJİ

February, 2022

146 pages

The increase in energy consumption and the gradual decrease in limited energy resources raise fundamental questions such as how much energy is produced, what is its efficiency, how it distributes to the end users. Not only on an individual basis, but also on a global basis research and development are carried out on this subject. Also, required equipment usage in energy generation, distribution and utilization is increasing every day, too. In the common systems for monitoring, need to install individual measuring devices e.g., Voltmeter, Oscilloscope, etc. separately because of the harmonic currents and voltages in the grids. Our design yields an industrial analyzer that sense the current and voltage at the input then designed circuit transmits signals proportional levels to the MCU which will be analyzed based on some standards such as IEEE Std 1459TM-2010 [1]. An A/D converter, converts the voltage and current signals to the digital representation at regular intervals. MCU periodically samples (f_sampling=1600 Hz, N=32, A/D resolution 12-bit) then analyze the voltage and current signals at regular intervals of time with the *0,4 % error rate. Then, it stores the data of electrical variables which are analyzed. e.g., frequency, true RMS volts, true RMS amps, cos(φ) and even harmonic values, including energy variables. For the project ARM based STM-32F405 and LCD-TFT screen which part number is HY-32D are one of the most important components. The features of this device completely provide our requirements with isolation circuits, fuses and another safety equipment. As a summary, it is the goal of the thesis to develop a compact analyzer device that analyzes the variable parameters in a grid by eliminating the require of multiple measuring devices. Thanks to the LCD, with the compact design we are able to see all data only on a screen and it’s completely mobilized.

*The error rate depends on the f_sampling, N, and A/D resolution.

Key Words: Embedded Systems, Arm Cortex, Energy Analyzer, Single Phase Analyzer, STM-32f429, Digital Signal Processing, Real-time Operating System.

ÖZET

GÖMÜLÜ SİSTEM KULLANARAK BİR ENERJİ ANALİZÖRÜ GELİŞTİRİLMESİ

POLAT, Uğur
Yüksek Lisans Tezi, Elektrik Elektronik Mühendisliği
Danışman: Dr. Öğrt. Üyesi Nurdal WATSUJİ
Şubat, 2022
146 sayfa

Enerji tüketiminin artması ve sınırlı enerji kaynaklarının giderek azalması; ne kadar enerji üretildiğinin, verimliliğin, son kullanıcılara nasıl dağıtıldığının sorularını gündeme getiriyor. Bu konuda sadece bireysel bazda değil, global bazda da araştırma ve geliştirmeler mevcuttur. Ayrıca enerji üretimi, dağıtımı ve kullanımında gerekli ekipman kullanımı da her geçen gün artmaktadır. Güç sistemlerinde şebekedeki dengesiz harmonik akımlar ve gerilimleri gözlemek için; Voltmetre, Ampermetre, Osiloskop Wattmetre vb. ayrı ayrı ölçüm cihazlarının kurulumu gereklidir. Yapmış olduğumuz çalışmamızda, şebekeden gelen akımı ve voltajı algılayan özel bir devre MCU'ya orantılı seviyelerde sinyaller iletir. IEEE Std 1459TM-2010 [1] gibi bazı standartlara göre analizler yapılır. Bir A/D dönüştürücü, voltaj ve akım sinyallerini düzenli aralıklarla ayrık zamanlı sinyale dönüştürür. MCU periyodik olarak örnekler (f_sampling=1600 Hz, N=32, A/D çözünürlüğü 12-bit) daha sonra voltaj ve akım sinyallerini düzenli aralıklarla yaklaşık *0,4 % hata oranıyla analiz eder. Analiz edilen frekans, RMS voltları, RMS amperleri, cos(φ), harmonikler ve enerji değerleri ölçülür. Projede STM-32F405 MCU ve HY-32D LCD-TFT ekran en önemli bileşenlerden biridir. PCB'miz için izolasyon devreleri, sigortalar ve diğer güvenlik ekipmanları ile gereksinimlerimizi tamamen karşılamaktadır. Özet olarak, birden fazla ölçüm cihazı gereksinimini ortadan kaldıran bir şebekedeki değişken parametreleri analiz eden tek bir analizör geliştirmek tezin amacıdır. LCD sayesinde tüm verileri sadece bir ekranda gösterecek mobilize ve kompakt yapıda olacaktır.

*Hata oranı f_örnekleme, N ve A/D çözünürlüğüne bağlıdır.

Anahtar Kelimeler: Gömülü Sistem, Arm Cortex, Enerji Analizörü, Tek Fazlı Analizör, STM-32f429, Sayısal Sinyal İşleme, Gerçek Zamanlı İşletim Sistemi

“Dedicated to my darling wife Kübra, daughter Eslem
Âmine and science of electronics…”

ACKNOWLEDGEMENTS

I would like to thanks my supervisor, Assist. Prof. Dr. Nurdal WATSUJİ and my motivator Mr. F. Gökçegöz and Mr. U. Şentürk for their guidance and support throughout the study. I am thankful for their encouragement and motivation.

I would like to express my gratitude and gratitude to my teachers who have supported me throughout my studies, to my darling wife, my family and friends who have always been there for me.

TABLE OF CONTENTS

Page

ABSTRACT V

ÖZET VI

ACKNOWLEDGEMENTS VIII

TABLE OF CONTENTS IX

LIST OF TABLES XII

LIST OF FIGURES XIII

LIST OF SYMBOLS XVIII

LIST OF ABBREVIATIONS XIX

CHAPTER I: INTRODUCTION 1

1.1. Literature Summary 1

1.1.1 Effective Value in AC Systems 2

1.1.2 Phase Difference Between AC Voltage and AC Current 4

1.1.3 Power Calculation in AC Systems 5

1.1.4 Power Factor (Cos φ) in Power Systems 7

1.1.5 Harmonic Distortion 9

1.2 Purpose of the Thesis 13

1.3 Conclusion and Key Findings 14

CHAPTER II: PHASE LOCKED LOOP (PLL) DESIGN 16

2.1 Phase Locked Loop 16

2.2 Phase Locked Loop (PLL) Methods 19

2.2.1 Notch Filtered Single Phase PLL 19

2.2.2 Orthogonal Signal Generator 26

2.3 Discretizing the Phase Locked Loop Model 35

2.3.1 Separation of the Second Order Low Pass Filter 36
2.3.2 Discretization of the PI Controller 39
2.3.3 Discrete Time Model of Classical PLL Structure 43
2.3.4 Software Implementation of the Digital PLL Model with PSIM..... 45
2.4 Hardware Implementation of Software PLL 47
CHAPTER III: HARDWARE DESIGN FOR INDUSTRIAL ANALYZER..... 50
3.1 Main Supply Circuit 50
3.2 Voltage Detection Circuit 51
3.3 Current Detection Circuit 53
3.4 Voltage and Current Zero-cross Detection Circuit 55
3.5 Battery Back-up Circuit 56
3.6 Microcontroller Unit (MCU) 57
3.7 Human Hardware Interface Design 62
3.8 Communication Hardware 66
3.9 Designed PCB 68
CHAPTER IV: SOFTWARE DESIGN FOR INDUSTRIAL ANALYZER 70
4.1 Message Queue Based Operating System 70
4.2 TFT-LCD Graphical Library 73
4.3 Effective Value Calculation of Voltage and Current in Discrete Time ... 75
4.4 Harmonics and Fast Fourier Transform (FFT) Analysis 79
4.5 The Calculation of Phase Angle [cos($\boldsymbol{\varphi}$)] 82
4.6 The Calculation of Power Factor (PF) 85
4.7 Power and Energy Calculation 88
CHAPTER V: HARDWARE REALIZATION AND MEASUREMENT RESULTS 91
5.1 Working Screens 91
5.2 The Effect of Sampling Frequency on the Measurements....................... 98

5.3 The Effect of Analog Digital Converter Resolution on Measurements... 98
5.4 The Effect of Total Number of Samples on Measurements 100
5.5 Other Factors That Effective on The Results.. 101
CHAPTER VI: MODBUS COMMUNICATION 103
6.1 Modbus Communication 103
CHAPTER VII: CONCLUSION AND FUTURE WORKS 106
REFERENCES 108
APPENDIX 111
Appendix A: Simulating the PLL for Varying Conditions for Notch Filter 111
Appendix B: Simulating the PLL for Varying Conditions for OSG Filter 114
Appendix C: Schematic of Human Hardware Interface HY32D 117
Appendix D: Schematic of HC-06 Bluetooth Device 118
Appendix E: Real-time Operating System Message Queue Service Codes........ 119
    E.1 Service Function of the System.. 119
    E.2 Message Sending Function to the System....................................... 119
    E.3 Message Requesting Function from the System 119
Appendix F: $\mathbf{Cos\varphi}$ Calculating Function by the MCU...................................... 121
Appendix G: Power Factor Calculating Function by the MCU 122
Appendix H: MODBUS Communication Address Mapping:............................. 123
Cirriculum Vitae 124

LIST OF TABLES

Page

Table 1.1 Electrical device classification according to EN 61000 - 3 - 2 standard 10
Table 1.2 Harmonic limits according to EN 61000 - 3 - 2 standards................. 12
Table 1.3 Voltage distortion limits according to IEEE 519 - 1992 standard 12
Table 2.1 PLL and adaptive notch filter coefficients 32
Table 5.1 Experiment results with the different sampling frequency 98
Table 5.2 Experiment results with the different resolutions of ADC................. 99
Table 5.3 Experiment results with the different samples of ADC 101
Table 6.1 Query 105
Table 6.2 Response... 105

LIST OF FIGURES

Page

Figure 1.1 Equivalent system block diagram for an energy analyzer 2
Figure 1.2 Value of the grid voltage in sine form 3
Figure 1.3 Sinusoidal signals that has the phase difference 4
Figure 1.4 Sinusoidal signals that has phase difference 5
Figure 1.5 Power Triangle, Relationship between Power factors 6
Figure 1.6 Instantaneous voltage, current and power values 6
Figure 1.7 A current signal with the 5th harmonic in it 9
Figure 1.8 Flow chart to apply standard EN 61000 - 3 -2.................................. 11
Figure 2.1 Phase Locked Loop Block Diagram 17
Figure 2.2 Single Phase PLL With Notch Filter 19
Figure 2.3 The response of the notch filter 21
Figure 2.4 Filter Interrupt flow chart 23
Figure 2.5 PLL Response to Varying Grid Conditions...................................... 23
Figure 2.6 PLL Response to Varying Grid Conditions...................................... 24
Figure 2.7 PLL Response to Varying Grid Conditions...................................... 24
Figure 2.8 PLL Response to Varying Grid Conditions...................................... 25
Figure 2.9 PLL Response to Varying Grid Conditions...................................... 25
Figure 2.10 OSG Based Single Phase PLL 26
Figure 2.11 Second Order Generalized Integrator for Orthogonal Signal Generation 26
Figure 2.12 Extraction of the Fifth Harmonic Using the SOGI 28
Figure 2.13 Interrupt Filter flow chart 29
Figure 2.14 Phase Jump of 90° 29
Figure 2.15 Phase Jump of 90° 30
Figure 2.16 Frequency drift at mid-point highlights the need for adaptive notch filter... 30
Figure 2.17 Amplitude change (Voltage Sags and Dips).................................... 31
Figure 2.18 Amplitude change with harmonics (Voltage Sags and Dips) 31

Figure 2.19 The signal $x(t)$ for f = 50 Hz and T = 0.2 s. 32
Figure 2.20 Under the multiple disturbances; (a) the signal, $x(t)$, (b) reference-filter output signal, and (c) FFT of the filter output. 33
Figure 2.21 Single-phase PLL application with OSG.. 33
Figure 2.22 Conventional OSG; a) SOGI, b) Park, c) Derivative, d) APF PLL. 33
Figure 2.23 Responses of the estimated angle error for 1 rad step change in PLL input phase 34
Figure 2.24 Responses of the estimated angle error for 50 rad/s step change in PLL input frequency 34
Figure 2.25 Responses of the estimated angle error for 50 rad/s step change in PLL input frequency 35
Figure 2.26 Classic PLL PSIM model 35
Figure 2.27 Response of MATLAB for Discrete-time transfer function 37
Figure 2.28 Response of MATLAB for Continuous-time transfer function 37
Figure 2.29 Programming block diagram for a second order low pass filter 38
Figure 2.30 Calculation of filter coefficients with PSIM................................... 39
Figure 2.31 General structure of the PI controller 40
Figure 2.32 Programming block diagram for PI controller................................ 42
Figure 2.33 Calculation of KP and Ki coefficients with Simulink environment 43
Figure 2.34 Discrete time block diagram of the classical PLL model 43
Figure 2.35 Response of digital PLL to sinusoidal input signal 44
Figure 2.36 Software implementation of digital PLL 45
Figure 2.37 Software PLL's response to sinusoidal input signal 45
Figure 2.38 Sampled Signal from the grid voltage (V_{line}).................................. 47
Figure 2.39 PLL output signal that locked to the grid signal............................. 48
Figure 2.40 MATLAB Simulink Demo PLL Program Outputs.......................... 49
Figure 3.1 Sophisticated Triac Driver Optoisolator 230 VAC to 12 VDC Power Supply Circuit 50
Figure 3.2 5 VDC circuit for TFT-LCD... 50
Figure 3.3 3V3 DC circuit for electronic components...................................... 51
Figure 3.4 Voltage detection circuit.. 51
Figure 3.5 Voltage offset circuit 53
Figure 3.6 Current detection circuit with hall effect sensor............................... 53
Figure 3.7 Current output circuit.. 54

Figure 3.8 Characteristic Performance data of ACS711KLCA-25A, Vcc=3,3 V 55
Figure 3.9 Zero-cross detection circuit 56
Figure 3.10 Battery backup circuit 56
Figure 3.11 Central processing unit (CPU) input and outputs 57
Figure 3.12 STM32F405VGT6 microchip designed by STM. 59
Figure 3.13 General features of the STM32F405VGT6 chip by STM. 60
Figure 3.14 ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32 61
Figure 3.15 Serial communication debugger interface circuit 61
Figure 3.16 External Oscillator circuit 62
Figure 3.17 Human Hardware Interface TFT-LCD Front view 63
Figure 3.18 Human Hardware Interface TFT-LCD Rear view 63
Figure 3.19 Header wiring diagram for connection between TFT-LCD and MCU 64
Figure 3.20 LED Indicators 65
Figure 3.21 Human PCB Interface buttons 65
Figure 3.22: RS-485 Interface circuit with the MCU 66
Figure 3.23 Bluetooth Serial Interface with the MCU 66
Figure 3.24 Preferred Bluetooth Module (HC-06) 67
Figure 3.25 USB Full-speed Interface Circuit 67
Figure 3.26 Designed PCB, Board Layer Stack Front view v.0.2 68
Figure 3.27 Designed PCB, Board Layer Stack Rear view v.0.2 68
Figure 3.28 Designed PCB, 3D Front view v.0.2 69
Figure 3.29 Designed PCB, 3D Front view v.0.2 69
Figure 4.1 Message Queue Structure 70
Figure 4.2 AppWizard by SEGGER and GUI Builder by NXP 74
Figure 4.3 emWin Bitmap Converter tool by SEGER. 74
Figure 4.4 STM-32F405 DMA Structure 76
Figure 4.5 CPU and DMA1 request an access to SRAM1 76
Figure 4.6 Peripheral-to-memory transfer states 77
Figure 4.7 Memory-to-peripheral transfer states 77
Figure 4.8 Timing diagram for sampling 78
Figure 4.9 Effective values of the Current Harmonics 82

Figure 4.10 Voltage and current signal to be calculated phase difference.......... 83
Figure 4.11 MATLAB output of Cos φ calculation for a harmonic-free system 84
Figure 4.12 Voltage and harmonic current signal to be calculated phase difference 85
Figure 4.13 MATLAB output of cosφ calculation for a harmonic system 85
Figure 4.14 Power factor calculation on MATLAB for a harmonic system....... 87
Figure 4.15 State of powers according to the sign of the phase angle 89
Figure 4.16 Energy flow direction according to the state of forces 88
Figure 5.1 Voltage, Current, frequency, Cos φ and PF analyze screen without the load 91
Figure 5.2 Active, Reactive and Apparent Power Analyze Screen without the load 92
Figure 5.3 Imported Active, Inductive and Reactive Energy Analyze Screen without the load... 92
Figure 5.4 Exported Active, Inductive and Reactive Energy Analyze Screen without the load... 93
Figure 5.5 Voltage Harmonical Spectrum Graphical Analyze Screen without the load 93
Figure 5.6 Voltage Harmonic Ratios Analyzed Values without the load 94
Figure 5.7 Current Harmonical Spectrum Graphical Analyze Screen without the load 94
Figure 5.8 Current Harmonic Ratios Analyzed Values without the load............ 94
Figure 5.9 Total Harmonic Distortion on the Voltage and Current Analyze Screen without the load... 95
Figure 5.10 Voltage, Current, frequency, Cos φ and PF analyze screen on the load 95
Figure 5.11 Active, Reactive and Apparent Power Analyze Screen on the load. 95
Figure 5.12 Imported Active, Inductive and Reactive Energy Analyze Screen on the load 96
Figure 5.13 Exported Active, Inductive and Reactive Energy Analyze Screen on the load 96
Figure 5.14 Voltage Harmonical Spectrum Graphical Analyze Screen on the load 96
Figure 5.15 Voltage Harmonic Ratios Analyzed Values on the load 97

Figure 5.16 Current Harmonical Spectrum Graphical Analyze Screen on the load 97
Figure 5.17 Current Harmonic Ratios Analyzed Values on the load 97
Figure 5.18 Total Harmonic Distortions on the Voltage and Current Analyze Screen on the load 98
Figure 5.19 Voltage and Current Graphs when the ADC has 6-bits resolution... 99
Figure 5.20 Voltage and Current Graphs when the ADC has 8-bits resolution... 99
Figure 5.21 Voltage and Current Graphs when the ADC has 10-bits resolution100
Figure 5.22 Voltage and Current Graphs when the ADC has 12-bits resolution100

LIST OF SYMBOLS

φ	Phase Difference between Voltage and Current
f	Frequency
Ω	Ohm
π	Pi
ω	Angular Velocity
f_c	Cutting Frequency
f_s	Sampling Frequency
T	Periodic Time
T_s	Sampling Periodic Time
ζ	Damping Ratio
K_v	Voltage Distortion Rate
K_i	Current Distortion Rate
P	Active Power
Q	Reactive Power
S	Apparent Power
THD	Total Harmonic Distortion
THD_I	Total Harmonic Distortion in Current
THD_V	Total Harmonic Distortion in Voltage
V_{rms}	RMS Voltage
V_{ref}	Reference Voltage

LIST OF ABBREVIATIONS

A/D	Analog to Digital Converter
AC	Alternate Current
ADC	Analog to Digital Converter
AGND	Analog Ground
APF	All-pass-filter
BAM	Batch Acquisition Mode
BPF	Band-Pass Filter
CLK	Clock
CMOS	Complementary Metal Oxide Semiconductor
CONVST	Conversation Start
CRC	Cyclic redundancy checky
D/A	Digital to Analog Converter
DAC	Digital to Analog Converter
DC	Direct Current
DGND	Digital Ground
DMA	Direct- memory Address
DSP	Digital Signal Processor
DVMs	Digital Volt-meters
FET	Field Effect Transistor
FIFO	First IN First OUT
FFT	Fast Fourier Transform
FPU	Floating-point Unit
FS	Full- Scale
GUI	Graphical User Interface
IC	Integrated Circuit
ISR	Interrupt Service Routine
LCD	Liquid Crystal Display

LPF	Low-Pass Filter
LSB	Least Significant Bit
MPU	Memory Protection Unit
MSB	Most Significant Bit
MSPS	Mega-Samples Per Second
NZ	Nyquist Zone
OSG	Orthogonal signal generator
P	Active Power- Magnitude of Complex Power
PCB	Printed Circuit Board
PD	Phase Detect
PF	Power Factor
PCM	Pulse Code Modulation
PGA	Programmable Gain Amplifier
PLL	Phase Lock Loop
PWM	Pulse Width Modulation
Q	Reactive Power- Magnitude of Complex Power
RAM	Random Access Memory
REFIN	Reference Input
REFOUT	Reference Output
RD	Read
RMS	Root Mean Square
RNG	Random number generator
RTC	Real-time Clock
RTD	Resistive Temperature Detector
RTOS	Real-Time Operating System
S	Apparent Power- Magnitude of Complex Power
SRAM	Static Random-Access Memory
SAR	Successive Approximation
SFDR	Spurious-Free Dynamic Range
SHA	Sample and Hold Amplifier
SNR	Signal to Noise Ratio
SOGI	Second-order generalized integrator
SPI	Serial Peripheral Interface

T/H	Track and Hold
TF	Transfer Function
TFT	Thin Film Transistor
THD	Total Harmonic Distortion
USART	Universal Synchronous Asynchronous Receiver Transmitter
VCO	Voltage-Controlled Oscillator
WE	Windowing Effect
WRT	Write

CHAPTER I

INTRODUCTION

1.1. Literature Summary

The measurement in electrical systems is generally a numerical expression of an electrical quantity. There are lots of data in the electrical networks. In alternative current systems, generally voltage, current, frequency, phase difference, power factor, active power, reactive power and energy values are numerically measured values. In the common systems for monitoring power circuits, we require to install individual measuring devices e.g., Voltmeter, Ammeter, Oscilloscope, Wattmeter etc. separately because of the increasing harmonic currents and voltages in the grids. Our design yields an industrial analyzer that sense the current and voltage at the input terminals then designed special circuit transmits signals proportional levels to the MCU which will be analyzed based on some standards such as IEEE Std 1459TM-2010 [1]. A 12-bit A/D converter, converts the voltage and current signals to the digital representation at regular intervals. MCU periodically samples (f_sampling=1600 Hz, N=32, A/D resolution 12-bit) then analyze the voltage and current signals at regular intervals of time with the approximately 0,4 % error rate. (The error rate depends on the f_sampling, N, and A/D resolution.) After that, stores the data of electrical variables which is analyzed. e.g., these electrical variables may be frequency, true RMS volts, true RMS amps, cos(φ) and even harmonic values, including energy variables. For the project ARM Cortex family as STM-32F405 and LCD-TFT screen which part number is HY-32D are one of the most important components. The features of this device completely provide our requirements with isolation circuits, fuses and another safety equipment for our PCB. As a summary, it is an objective of the thesis to develop an industrial analyzer which a single analyzer device that analyzes the variable parameters in a grid by eliminating the require of multiple measuring devices. Thanks to the LCD, we are able to see all data only in a screen and it’s completely mobilized. You can see block diagram of the analyzer in Figure (1.1).

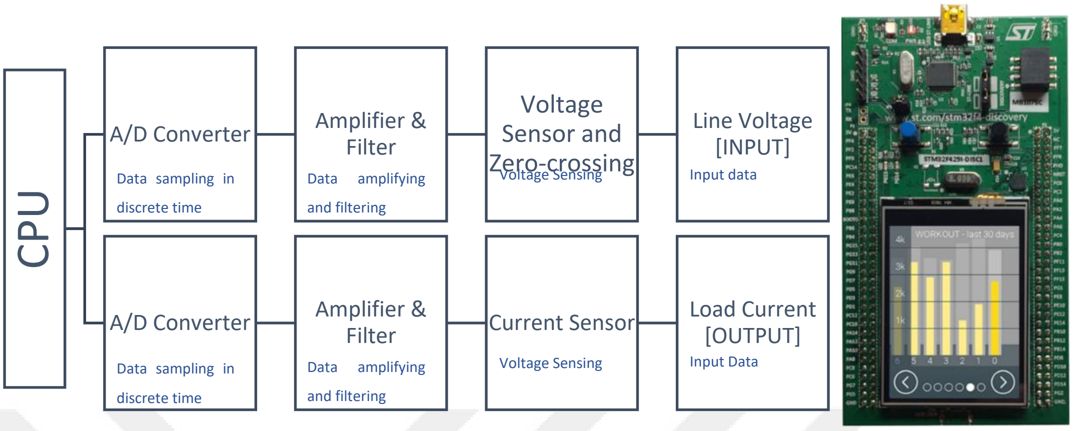


Figure 1.1 Equivalent system block diagram for an energy analyzer

1.1.1 Effective Value in AC Systems

Voltage and current magnitudes change according to time and have a certain amplitude and frequency. For this reason, it is usually expressed with its effective values. Effective value is a type of value that enables the expression of equivalent equivalents of magnitudes in alternating current systems in direct current systems. It can be expressed as the equivalent of AC voltage required to obtain the amount of heat energy generated in a resistance under a DC voltage [2].

The effective value can be represented as the RMS (Root Mean Square) value consisting of the effective value or the initials of its English equivalent. The active power expended on a load is obtained by equation (1.2) in alternating and direct current systems [2]. According to this equation, the effective value can be expressed as the square root of the mean of the square of the voltage (or current), the integral over a period. In digital systems, the sum of the squares of instantaneous samples of the signal is averaged. The square root of this value is calculated and therefore the effective value is found.

$$P = \frac{V^2}{R} \tag{1. 1}$$

$$\frac{V_{dc}^2}{R} = \frac{1}{T}\cdot\int_0^T \frac{v(t)^2}{R}\cdot dt \tag{1. 2}$$

$$V_{rms} = \sqrt{\frac{1}{T}\cdot\int_0^T v(t)^2\cdot dt} \tag{1. 3}$$

The effective (effective) value of the grid voltage in sine form can be written as in (1.5) equation.

$$v(t) = v_m\cdot\sin(\omega t) \tag{1. 4}$$

$$v_{rms} = \sqrt{\frac{1}{T}\cdot\int_0^T \{v_m\cdot\sin(wt)\}^2\cdot dt} \tag{1. 5}$$

- The wave in red is said to *lead* the wave in green by θ.
- The wave in green *sin(ωt)* is said to *lag* the wave in red by θ.
- The units of θ and *ωt* must be consistent when computing the sine function.

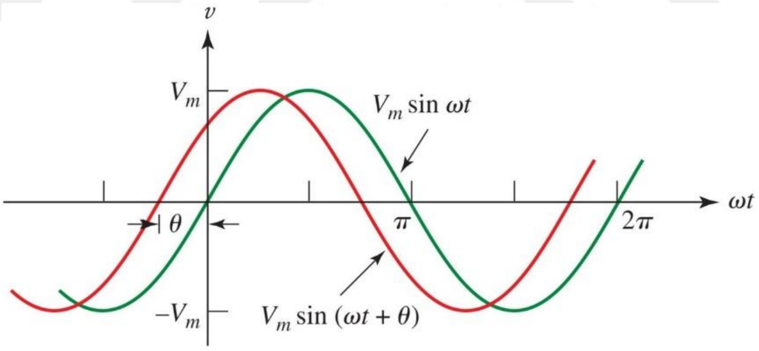


Figure 1.2 Value of the grid voltage in sine form

The effective value of the network voltage in the sine form in the range of [0, 2π] can be mathematically calculated by equation (1.6) where v_m as the peak value.

$$v_{rms} = \sqrt{\frac{1}{T}\cdot\int_0^T [v_m\cdot\sin(\omega t)]^2\cdot dt} \tag{1. 6}$$

$$v_{rms} = \frac{v_m}{\sqrt{2}} \tag{1. 7}$$

1.1.2 Phase Difference Between AC Voltage and AC Current

Inductance and capacitors in AC systems cause phase difference between voltage and current [3]. If we take the V1 signal as reference in Figure (1.3), the V2 signal is in reverse phase according to the V1 signal, and the V3 signal is in forward phase according to the V1 signal. Even if the amplitudes are not equal, the frequencies must be equal in order to be able to talk about the phase difference. If there is a DC component on the signal, this component can be reset and the phase difference can be checked [3].

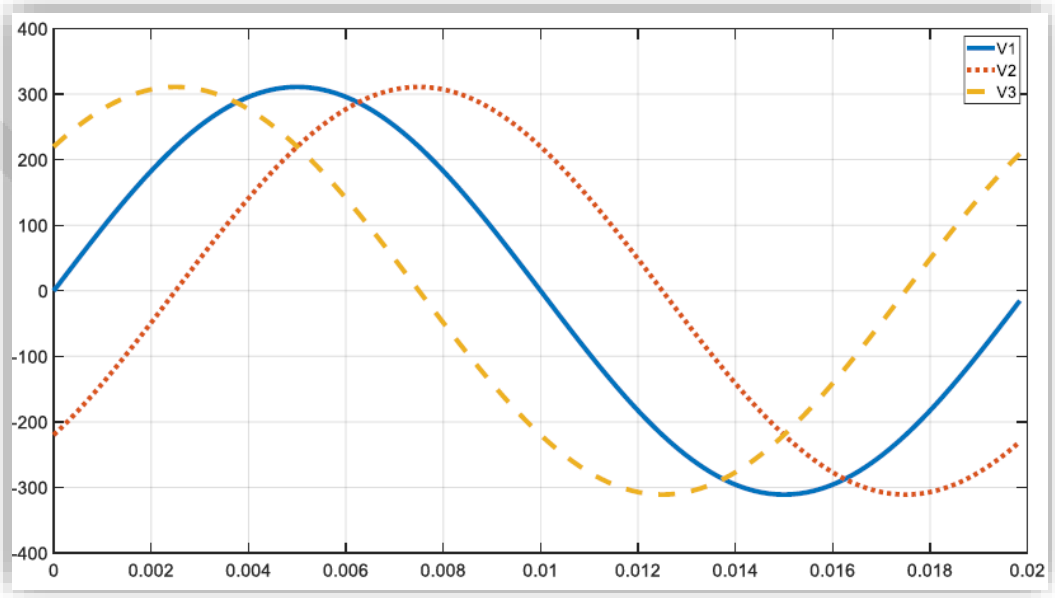


Figure 1.3 Sinusoidal signals that has the phase difference

If the load is pure ohmic, there is no phase difference between voltage and current [3]. In case of inductive load, the current is in phase back from the voltage, in case of capacitive load the voltage is in phase back from the current.

This phase difference between voltage and current is denoted by "φ" and is called the phase angle. Since the phase angle close to zero is the most ideal situation for the network, the reactive power in the systems is compensated and the phase difference between the current drawn from the network and the voltage is tried to be reset. In this case, the reactive power required by the system is supplied from the compensation panel installed in the operation.

Current and voltage signals with a phase difference of 57.3 ° is shown in Figure (1.4).

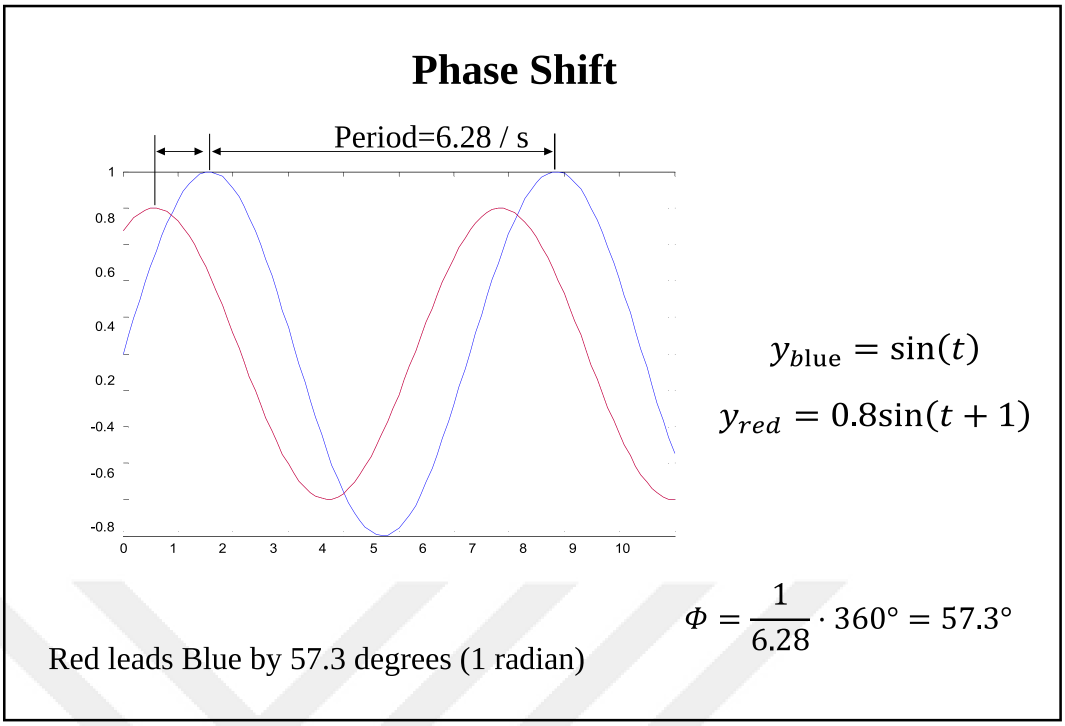


Figure 1.4 Sinusoidal signals that has phase difference

1.1.3 Power Calculation in AC Systems

Active power (P) in AC systems is defined as useful and usable power. Reactive power (Q) is defined as the power that does not work actively but is withdrawn from the network and then transferred back to the network. Apparent power (S) is the total resultant power formed by active and reactive power components. If the system has a pure ohmic load, there will be no phase difference between the current and voltage, so there will be no reactive power. Therefore, the active power value will be equal to the apparent power value. If there is a reactive power in the system, the apparent power will be equal to the combination of active and reactive powers.

$$S^2 = P^2 + Q^2 \tag{1. 8}$$

$$P = S \cdot \cos\varphi \tag{1. 9}$$

$$Q = S \cdot \sin\varphi \tag{1. 10}$$

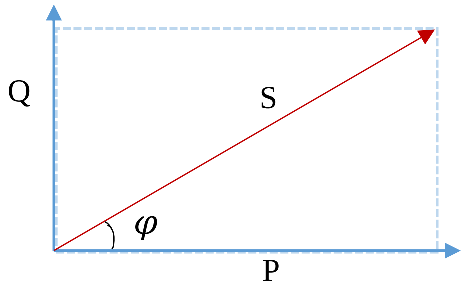


Figure 1.5 Power Triangle, Relationship between Power factors

Instantaneous power value is obtained by multiplying the instantaneous values of the voltage and current components [3]. Instantaneous power value changes with twice the system frequency.

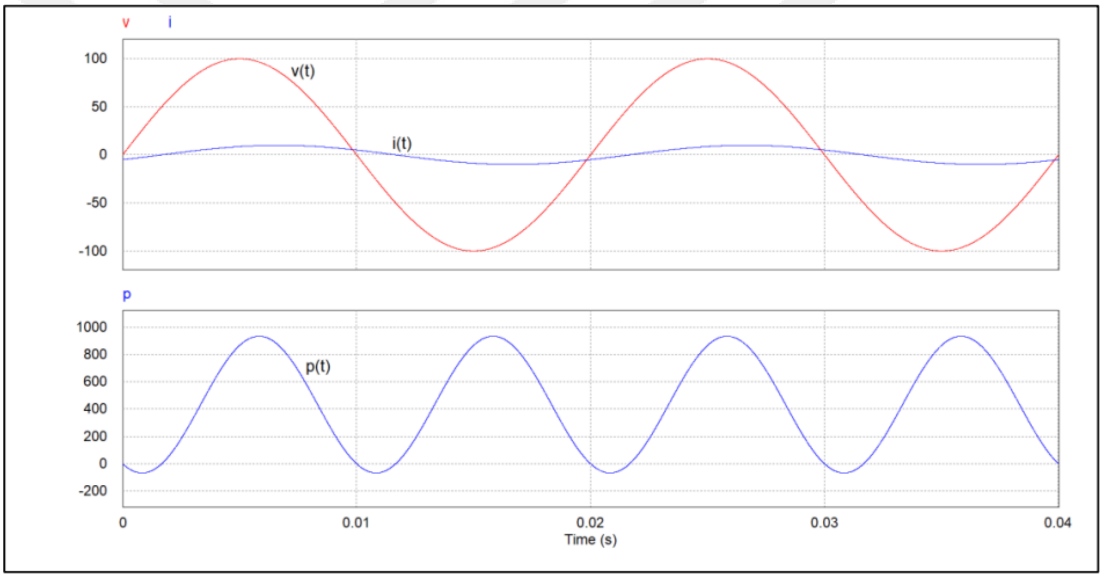


Figure 1.6 Instantaneous voltage, current and power values

In electrical systems, instantaneous power is expressed by equation (1.10), where v (t) and i (t) are instantaneous values of voltage and current.

$$P(t) = v(t) \cdot i(t) \tag{1. 11}$$

Devices that measure electrical power measure the average value of power [3]. In this case, if the instantaneous power expression is integrated over a period, the expression (1.13) can be written for the average power value (active power).

$$v(t) = v_m \cdot \sin(\omega t) \tag{1. 12}$$

$$i(t) = i_m \cdot \sin(\omega t - \varphi) \quad (1.\ 13)$$

$$P = \frac{1}{2\pi} \cdot \int_0^{2\pi} [v_m \cdot I_m \cdot \sin(\omega t) \cdot \sin(\omega t - \varphi)] \cdot dwt \quad (1.\ 14)$$

$$P = v_{rms} \cdot I_{rms} \cdot \cos(\varphi) \quad (1.\ 15)$$

The reactive power value can be calculated by the product of the voltage component and the 90° phase-shifted state of the current component or by the product of the current component and the 90 ° phase-shifted state of the voltage component. If the 90° phase shifted state of the current signal is expressed as I_q, the expression (1.17) can be written for the reactive power.

$$v(t) = v_m \cdot \sin(\omega t) \quad (1.\ 16)$$

$$I_q(t) = i_m \cdot \sin\left(\omega t - \varphi + \frac{\pi}{2}\right) = i_m \cdot \cos(\omega t - \varphi) \quad (1.\ 17)$$

$$Q = \frac{1}{2\pi} \cdot \int_0^{2\pi} [v_m \cdot I_m \cdot \sin(\omega t) \cdot \cos(\omega t - \varphi)] \cdot d\omega t \quad (1.\ 18)$$

$$Q = v_{rms} \cdot I_{rms} \cdot \sin(\varphi) \quad (1.\ 19)$$

As a summary power equation is follows,

Active Power → $P = v_{rms} \cdot I_{rms} \cdot \cos(\varphi)$ (Watt)

Reactive Power → $Q = v_{rms} \cdot I_{rms} \cdot \sin(\varphi)$ (Var)

Apparent Power → $S = v_{rms} \cdot I_{rms}$ (VA)

In digital systems, it is possible to calculate the phase difference and power values based on the instantaneous values of voltage and current. In MATLAB environment, instantaneous values of current and voltage and power values can be calculated by using averaging command.

1.1.4 Power Factor (Cos φ) in Power Systems

In power systems, the ratio of active power to apparent power is defined as the power factor [2]. If the network voltage and current are in sinusoidal form, that is, if the system does not contain a non-linear load, cos φ is equal to the power factor (PF).

$$P = v_{rms} \cdot I_{rms} \cdot \cos(\varphi) \quad (1.\ 20)$$

$$Q = v_{rms} \cdot I_{rms} \cdot \sin(\varphi) \tag{1. 21}$$

$$S = \sqrt{P^2 + Q^2} \tag{1. 22}$$

$$\cos(\varphi) = PF = \frac{P}{S} \tag{1. 23}$$

In addition to that, if there are non-linear loads such as rectifiers, uninterruptible power supplies, electronic ballasts in power systems, current and voltage signal shapes move away from the sinusoidal form and harmonics occur in the system.

In a harmonic system, power expressions such as the effective value of the voltage fundamental component v_1 and the effective value of the current fundamental component i_1 can be written as follows [5].

$$S_1 = \sqrt{{P_1}^2 + {Q_1}^2} \tag{1. 24}$$

$$P_1 = v_1 \cdot I_1 \cdot \cos(\varphi) \tag{1. 25}$$

$$Q_1 = v_1 \cdot I_1 \cdot \sin(\varphi) \tag{1. 26}$$

$$PF = \frac{v_1}{v_{rms}} \cdot \frac{i_1}{i_{rm_s}} \cdot \cos\varphi \tag{1. 27}$$

$$PF = K_{v\_distortion} \cdot K_{i\_distortion} \cdot \cos(\varphi) \tag{1. 28}$$

In equation (1.28), $K_{distortion}$ coefficients are defined as distortion factors for current and voltage, and $\cos\varphi$ is defined as the displacement factor. If it is assumed that there is no harmonic in the network voltage, the $K_{v\_distortion}$ coefficient will have no effect and the power factor value will be as in equation (1.40) [5].

$$PF = 1 \cdot K_{i\_distortion} \cdot \cos(\varphi) = K_{i\_distortion} \cdot \cos(\varphi) \tag{1. 29}$$

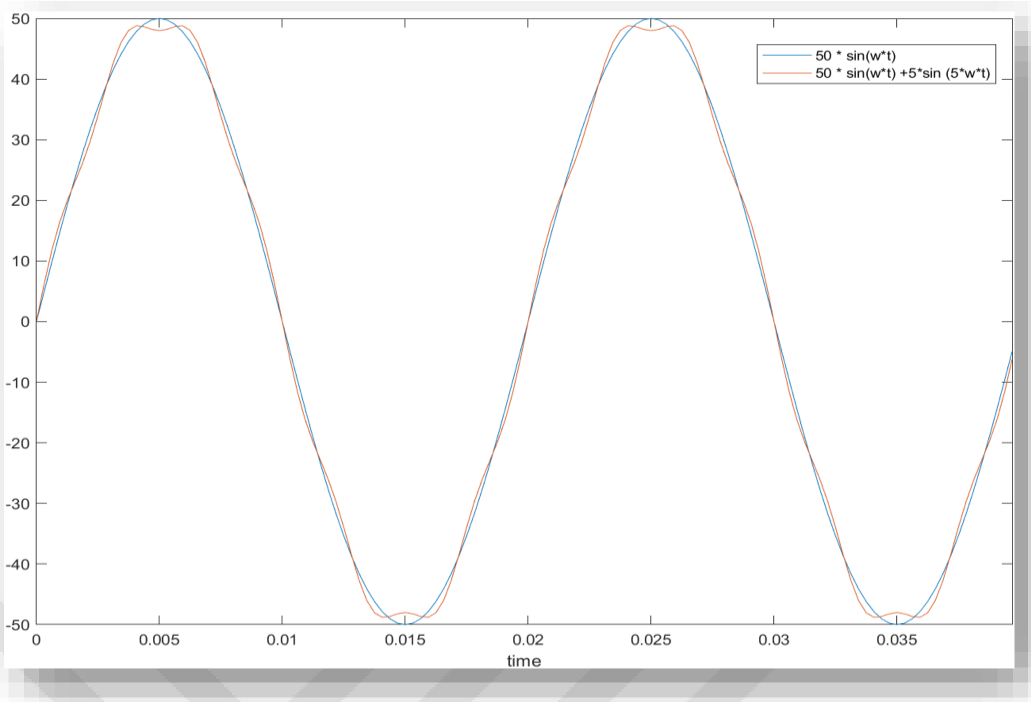


Figure 1.7 A current signal with the 5th harmonic in it

1.1.5 Harmonic Distortion

In electrical power systems containing harmonics, the effective values of current and voltage can be expressed in the following equations.

$$v_{rms} = \sqrt{{v_1}^2 + {v_2}^2 + {v_3}^2 + \cdots + {v_n}^2} \tag{1. 30}$$

$$v_{rms} = \sqrt{v_1^2 + \sum_{h=2}^{n} v_h^2} \tag{1. 31}$$

$$i_{rms} = \sqrt{{i_1}^2 + {i_2}^2 + {i_3}^2 + \cdots + {i_n}^2} \tag{1. 32}$$

$$i_{rms} = \sqrt{i_1^2 + \sum_{h=2}^{n} i_h^2} \tag{1. 33}$$

Total harmonic distortion value can be defined as the ratio of harmonic components to fundamental components. This value can be written separately for current and voltage.

$$THD_v = \frac{\sqrt{v_2^2+v_3^2+\cdots+v_n^2}}{v_1} \tag{1. 34}$$

$$THD_i = \frac{\sqrt{i_2^2+i_3^2+\cdots+i_n^2}}{i_1} \qquad (1.\ 35)$$

International standards such as IEEE 519 and IEC 61000 regarding energy quality have been put forward. The EN 61000 - 3 - 2 standard offered by IEC is implemented in our country as TS EN 61000 - 3 - 2: 2015, which was updated in 2015. Electrical devices are classified according to these standards and voltage and current harmonic limits are specified. Classification of devices below 16A per phase according to this standard is shown in Table (1.1). In addition, the flow chart for determining the classes in the EN 61000-3-2 standard is included in Figure (1.11). In addition, harmonic limits of these classes are given in Table (1.2), and voltage distortion limits according to IEEE 519 - 1992 standard are given in Table 1.3. [4]

Table 1.1 Electrical device classification according to EN 61000 - 3 - 2 standard

Class	Definition
A	Balanced three-phase loaded devices, professional devices in the range of 75 - 1000W, electrical devices not defined in B / C / D classes
B	Non-professional portable electrical devices that consume 75W and above, non-professional welding devices
C	Lighting devices below 1000W
D	Computers, monitors, televisions and radio equipment in the 75W-600W range

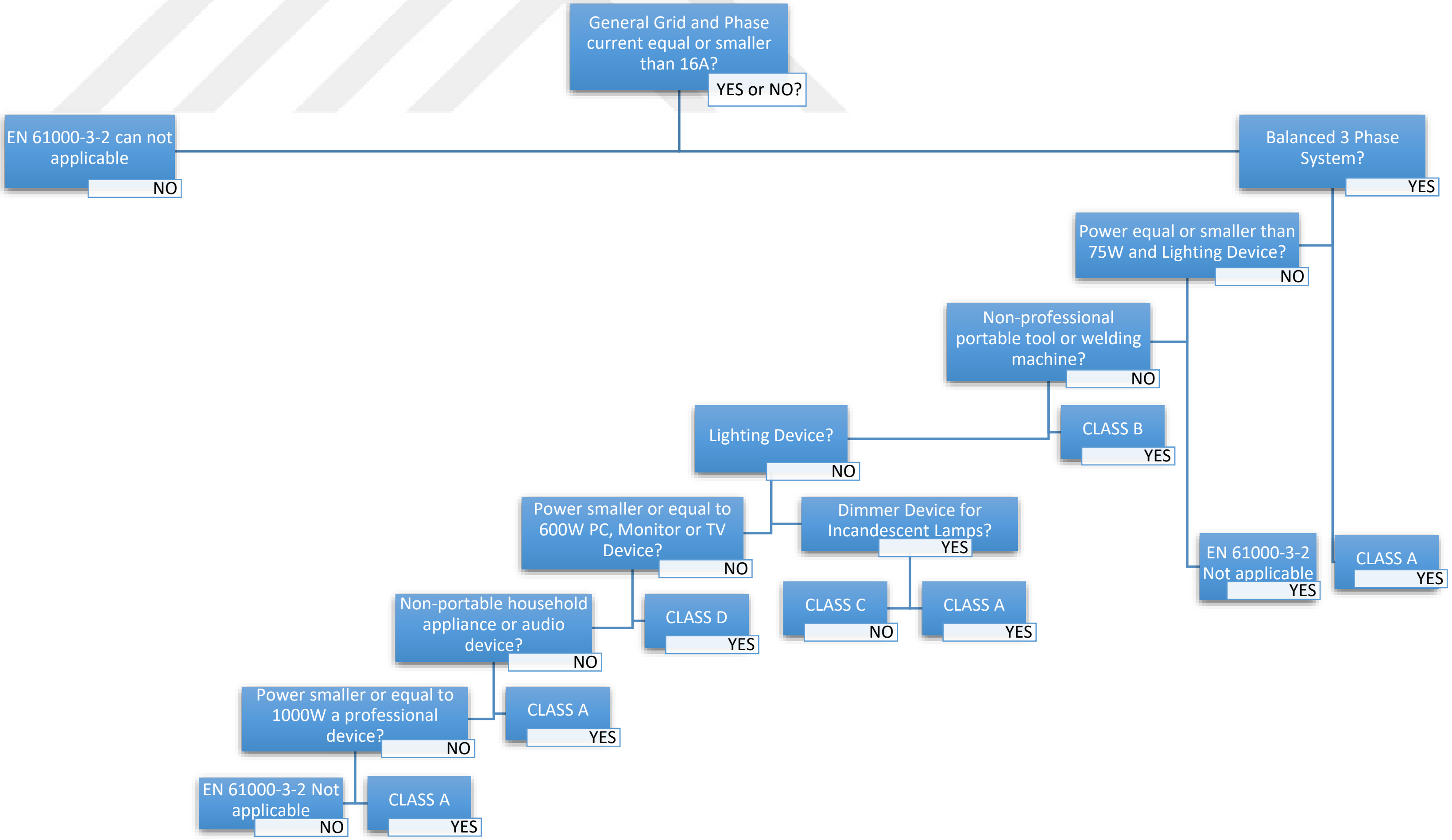

General Grid and Phase current equal or smaller than 16A?
YES or NO?
EN 61000-3-2 can not applicable
NO
Balanced 3 Phase System?
YES
Power equal or smaller than 75W and Lighting Device?
NO
Non-professional portable tool or welding machine?
NO
CLASS B
YES
Lighting Device?
NO
Power smaller or equal to 600W PC, Monitor or TV Device?
NO
Dimmer Device for Incandescent Lamps?
YES
EN 61000-3-2 Not applicable
YES
CLASS A
YES
Non-portable household appliance or audio device?
NO
CLASS D
YES
CLASS C
NO
CLASS A
YES
Power smaller or equal to 1000W a professional device?
NO
CLASS A
YES
EN 61000-3-2 Not applicable
NO
CLASS A
YES

Figure 1.8 Flow chart to apply standard EN 61000 - 3 -2

Table 1.2 Harmonic limits according to EN 61000 - 3 - 2 standards

Harmonic No.	**Class A (A)**	**Class B (A)**	**Class C (I_n/I_1) (%)**	**Class D (mA/W)**
Odd numbered Harmonics				
3	2,30	3,45	30	3,4
5	1,13	1,71	10	1,9
7	0,77	1,155	7	1,0
9	0,40	0,60	5	0,5
11	0,33	0,495	3	0,35
13	0,21	0,315	3	0,3
15	0,15	0,220	3	0,25
17≤h≤39	0,13*(15/h)	0,19*(15/h)	3	3,85/n
Even numbered Harmonics				
2	1,08	1,62	2	-
4	0,43	0,645	-	-
6	0,30	0,45	-	-
8	0,23	0,34	-	-
10≤h≤40	0,23*(8/h)	0,345*(8/h)	-	-

Table 1.3 Voltage distortion limits according to IEEE 519 - 1992 standard

V	**V_n/V (%)**	**THD (%)**
V ≤ 1kV	5,0	8,0
1kV < V ≤ 69kV	3,0	5,0
69kV < V ≤ 161kV	1,5	2,5
161kV < V	1,0	1,5

Apart from these standards, there are some other standards related to energy quality [5].

- ✓ EN 61000 - 3 - 3: It determines the voltage fluctuation limits in LV systems.
- ✓ EN 61000 - 3 - 4: They define harmonic limits for devices larger than 16A per phase.
- ✓ EN 50006: It determines the disruption limits caused by household and similar electrical devices containing electronic elements in the network.
- ✓ TS 9882: It determines the disruption limits caused by household and similar electrical devices in the network.
- ✓ VDE 0838: Home Appliances,
- ✓ VDE 0160: Converters,
- ✓ VDE 0712: Sets harmonic limits for fluorescent lamps and ballasts.

1.2 Purpose of the Thesis

The monitoring of electrical power is important to ensure that this energy resource is effectively generated, distributed and utilized. Utilities need to measure power coming out of a generation station or going into power station. In addition, to minimize power transmission losses, it's important to minimize the phase relationship between the current and voltage waveforms of the power being transmitted. In industrial control applications, it's important to be able to continuously monitor the current and phase of the power into a machine which may vary with the machine load. Traditional systems for monitoring power circuits require the installation of individual measuring devices to measure a specific power system parameter; for example, Watts, VARs, Amps, or Volts. These devices typically comprise discrete analog transducers which convert AC voltage and current signals from a power system into DC output signals proportional to the true power on the system. For example, typical utility revenue kilowatt hour meters measure power in an analog fashion. Where a data acquisition system must measure numerous circuits, requiring separate measuring devices for each circuit can add greatly to the overall cost of the system. Also, where a number of circuits are being remotely monitored for computer processing and display, the individual devices must be connected to a data acquisition device and suitably processed to interface with the computer. Further, multiple measuring devices greatly increase the overall error of the system. Also, it is a complex system for measuring various parameters in multiple power circuits and the cost of purchasing installing and maintaining the system may be increased. Accordingly, it is a principal objective of the thesis to provide a simplified system for measuring various parameters in multiple power circuits to reduce the cost of purchasing, installing and maintaining the system. More specifically, it is an objective of the thesis to provide a system in which a single measuring device measures the fundamental parameters in a power system so that subsequent derived parameters can be computed in a controller data processing computer, thereby eliminating the necessity of multiple measuring devices. Additionally, the thesis aims to provide a power analyzer which reduces the cost of maintenance of the overall system and to provide a power analyzer which has very high precision through self-calibration and a non-synchronous measurement technique. It is one more objective of the thesis is to provide a microcontroller-controlled power analyzer that can continuously and rapidly monitor a plurality of circuits. With the thesis, low-cost power and energy analyzer will be developed. The power analyzer that will be developed in the thesis will include a microprocessor coupled to the circuit for performing additional calculations on the electrical variables to thereby derive electrical parameters relating to the performance of the power circuits. These parameters may include Watts, Watt-hours, VARs, Power Factor, etc. After the calculations whole results will be monitoring on the screen and also there will be a real time plotting of the spectrum. [36]

1.3 Conclusion and Key Findings

It’s proved that various factors directly affect the measurement quality. Temperature changes of electronic components used are one in all the important factors. thanks to the optimal working environment of the components are different, so as to attenuate the negative impact of this case, the materials used is preferred with higher sensitivity and lower temperature coefficient. Also, the field effect sensor utilized in the current input is tormented by the external magnetic field. To avoid negative effect of this case, it's going to be possible to produce magnetic isolation of the area where the sensor is located on the hardware or standard transformer like rated 100A/4mA is also accustomed get far better results with the high precision. However, a current transformer requires much space. Besides, sensitivity of the operational amplifiers utilized in voltage and current inputs is critical. A low precision operational amplifier will cause irrelevant results, especially when making measurements too close to zero. To avoid this case, choosing a high quality and high precision operational amplifier will influence positively affect the results. Additionally, chosen MCU is vital. If a microcontroller with high performance and decimal processing capability isn't preferred, it'll influence negatively the results in order that it'll cause loss of sensitivity during the mathematical operations.

Also, the reference voltage for the calculations is critical. So, the battery voltage will also affect negatively the results.

The sampling frequency of the selected microcontroller to be used in the designed digital measurement system, the resolution of the analog-digital converter, the sampling rate and the number of ADC channels that can operate simultaneously are sufficient, calculation can be performed without losing time in sampling.

In addition, if the temperature dependence of the sensors and electronic materials used is calibrated with an external temperature measurement, an error-free measurement can be obtained.

CHAPTER II

PHASE LOCKED LOOP (PLL) DESIGN

The phase angle of use may be an important piece of information on the operation of power grids such as PV inverters. A locked phase loop is a closed system in which the internal oscillator is controlled to keep track of the time and the external time signal phase using the feedback loop. PLL is the only servo system that controls its output signal phase and specifies phase error between the output phase and the reference phase is small. Lock quality directly affects the operation of the control loop on applications that are tied to the grid. Since line measurement, voltage imbalance, line immersion, phase loss and frequency variations are common conditions for mechanical and electrical interactions, PLL needs to be able to reject these error sources and maintain a clean phase lock on the grid. [6]. In this section, various software loop lock options will be analyzed and the simulation results will be presented.

2.1 Phase Locked Loop

Typically, a phase loop is a closed loop control system with an internal oscillator so that it can lock out the external signal section using the feedback structure. It can be simply expressed as a servo control system that minimizes the phase difference between the reference signal and the output signal. In devices connected to the network, there may be situations such as frequency change, absence of phase or imbalance [6]. In such cases, a phase locked loop design should be designed to minimize these errors for the stability of measurement systems. As seen in Figure (2.1), a simple phase locked loop (PLL) consists of a phase detector, a PI controller and a voltage-controlled oscillator unit at its output.
A functional diagram of a PLL is showing in figure, which consists of a phase detect (PD), a loop filter (LPF), and a voltage-controlled oscillator (VCO).

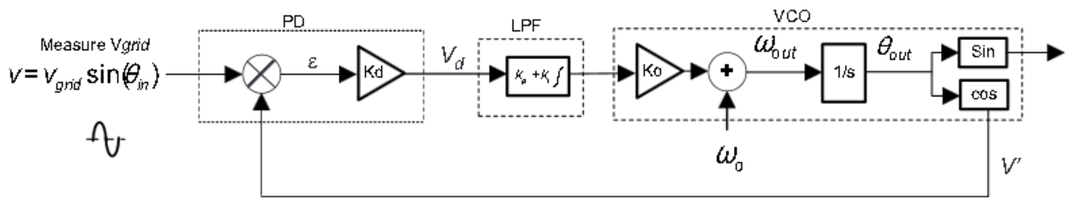


Figure 2.1 Phase Locked Loop Block Diagram

The measured grid voltage can be written in terms the grid frequency (w_{grid}) as follows,

$$v = v_{grid} \cdot \sin(\theta_{in}) - v_{grid} \cdot \sin\left(\omega_{grid} t + \theta_{grid}\right) \tag{2. 1}$$

Now, assuming that VCO produces sine waves near the sinusoid grid, the VCO output can be labeled,

$$v' = \cos(\theta_{out}) - \cos(\omega_{PLL} t + \theta_{PLL}) \tag{2. 2}$$

The purpose of the phase detection block is to compare the input sine with the locked sine from the VCO and to generate the error signal in line with the angle error. In this case, the phase acquisition block doubles the VCO output and the estimated input value to be obtained:

$$v_d = \frac{K_d \cdot v_{grid}}{2}\left[\sin\left(\left(\omega_{grid} - \omega_{PLL}\right)t + \left(\theta_{grid} - \theta_{PLL}\right)\right) + \sin\left(\left(\omega_{grid} - \omega_{PLL}\right)t + \left(\theta_{grid} + \theta_{PLL}\right)\right)\right] \tag{2. 3}$$

From Equation (2.3), it is clear that the output of the PD block contains information about the lock error. However, the lock error information found in PD is linear, and has twice as much variation as grid frequency. To use this error lock information PLL angle, part of the double frequency grid must be removed. In the meantime, ignoring duplicate part of the grid frequency, a lock error is provided by:

$$v_d = \frac{K_d \cdot v_{grid}}{2} \sin\left(\left(\omega_{grid} - \omega_{PLL}\right)t + \left(\theta_{grid} - \theta_{PLL}\right)\right) \tag{2. 4}$$

For steady state operation, the $\omega_{grid} - \omega_{PLL}$ term can be ignored, for small values of theta sin(θ) ~ θ. Hence, a linearized error is given as:

$$err = \frac{v_{grid}(\theta_{grid} - \theta_{PLL})}{2} \tag{2. 5}$$

This error is an input loop filter, which is nothing but a PI controller, which is used to reduce the lock error in the stabilization mode to zero. Small signal analysis is performed using network theory, in which the response loop is broken to obtain an open loop transfer number and then a closed loop transmission function:

$$Closed\ Loop\ TF = Open\ Loop\ TF\ /\ (1 + Open\ Loop\ TF) \quad (2.\ 6)$$

Thus, in response to the line the PLL transfer function can be written as follows:

Closed loop section TF:

$$H_0(s) = \frac{\theta_{out}(s)}{\theta_{in}(s)} - \frac{LF(s)}{s+LF(s)} - \frac{v_{grid}\cdot\left(k_p s+\frac{k_p}{Ti}\right)}{s^2+v_{grid}\cdot k_p s+v_{grid}\cdot\frac{k_p}{T_i}} \quad (2.\ 7)$$

Closed loop error transfer function:

$$E_0(s) = \frac{v_d(s)}{\theta_{in}(s)} - 1 - H_o(s) - \frac{s}{s+LF(s)} - \frac{s^2}{s^2+k_p s+\frac{k_P}{T_i}} \quad (2.\ 8)$$

Comparing the closed loop transfer function with the standard second order system transfer function, provided:

$$H(s) = \frac{2\cdot\zeta\cdot\omega_n s+\omega_n^2}{s^2+2\cdot\zeta\cdot\omega_n s+\omega_n^2} \quad (2.\ 9)$$

The natural frequency and reduction of the PLL rate in line is given by:

$$\omega_n = \sqrt{\frac{v_{grid}\cdot k_p}{T_i}} \quad (2.\ 10)$$

$$\zeta = \sqrt{\frac{v_{grid}\cdot T_i\cdot k_p}{4}} \quad (2.\ 11)$$

Note to PLL, PI serves two purposes:

- To filter out high frequency that is at twice the frequency of the carrier and grid
- Manage PLL response to track changes in grid shapes, for example, section jumps, size sources, and more. Since the loop filter has a low-pass filter feature, it can be used to filter a portion of the high frequency that was previously ignored. If the network frequency / lock signal frequency is high, the lower PI transition features are good enough to cancel twice the frequency of the network company frequency. However, in grid-connected systems as the grid frequency is very low (50Hz-60Hz), the roll off provided by the PI is not sufficiently satisfactory and introduces a high frequency feature at the loop filter output, which affects PLL performance. From the discussion above, it is clear that the LPF feature of the PI controller cannot be used to eliminate part of a double frequency to grid from the output phase in the case of grid-connected

applications. Therefore, other methods should be used that guide the PD block. In this application report, the two PLL modes that make up the PD exit line, are displayed:

- One uses a notch filter to filter out a double part of a grid frequency on a PD output
- The other uses an orthogonal signal generating method to use the PLL process for a fixed PLL framework in one PLL phase.

2.2 Phase Locked Loop (PLL) Methods

2.2.1 Notch Filtered Single Phase PLL

In this PLL method, a notch filter can be used to exit the phase detection block, which doubles the part of the grid frequency very well. The flexible notch filter can also be used for selection by selecting the correct frequency in the event of a grid frequency variation. Section (2.1) illustrates the process of selecting PI coefficients, their digital usage and mapping. The design of the flexible notch filter is displayed with the automatic coefficient calculation method, and in line is displayed using embed code. A single phase, phase locked loop structure with a notch filter is as follows.

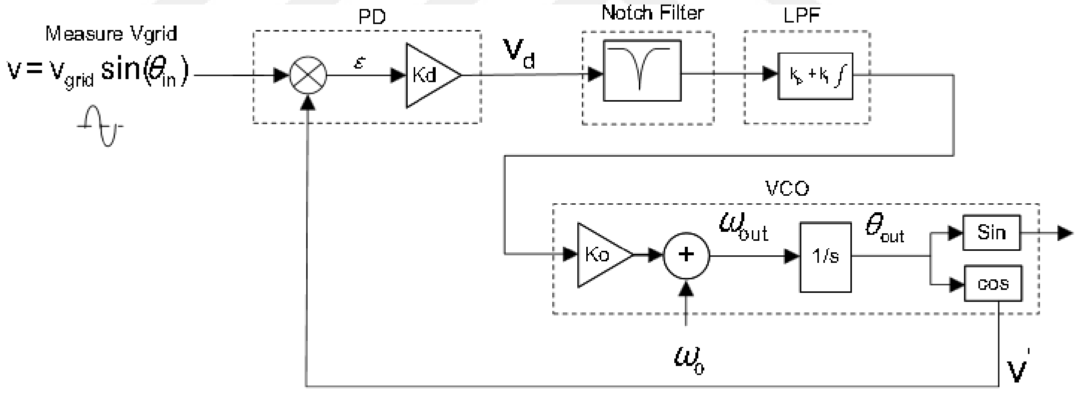


Figure 2.2 Single Phase PLL With Notch Filter

As discussed in Section (2.1), with the addition of a notch filter, PI tuning can only be performed based on the PLL variable response. Section (2.1) illustrates the digital use of the PI controller and the selection of coefficients for the use of the PI control.

2.2.1.1 Discrete Implementation of PI Controller

The loop filter or the PI is implemented as a digital controller with like this

$$y_{lf}[n] - y_{lf}[n-1] \cdot A_1 + y_{notch}[n] \cdot B_0 + y_{notch}[n-1] \cdot B_1 \quad (2.\ 12)$$

Using z-transform, equation (2.12) can be re-written as:

$$\frac{y_{lf}(z)}{y_{notch}(z)} = \frac{B_0+B_1 \cdot z^{-1}}{1-z^{-1}} \tag{2. 13}$$

It is well known the PI controller in Laplace transform is given by:

$$\frac{y_{lf}(s)}{y_{notch}(s)} = k_p + \frac{k_i}{s} \tag{2. 14}$$

Using bi-linear transformation, replace $s = \frac{2}{T} \cdot \left(\frac{z-1}{z+1}\right)$, where T = *Sampling Time*.

$$\frac{y_{lf}(z)}{y_{notch}(z)} = \frac{-\left(\frac{2 \cdot k_p+k_i \cdot T}{2}\right)-\left(\frac{2 \cdot k_P-K_i \cdot T}{2}\right) \cdot z^{-1}}{1-z^{-1}} \tag{2. 15}$$

Equation (2.13) and Equation (2.15) can be compared to a map of the equity and completeness of PI control in a digital domain. The next challenge is to choose the right amount of benefit and total profit. Step response to standard second order calculations:

$$H(s) = \frac{\omega_n^2}{s^2+2 \cdot \zeta \cdot \omega_n s+\omega_n^2} \tag{2. 16}$$

is given as:

$$y(t) = 1 - ce^{-\sigma t} \cdot \sin(\omega_d \cdot t + \varphi) \tag{2. 17}$$

Ignoring the LHP zero from Equation (2.17). The settling time is given as the time it takes for the response to settle between an error band, say this error is ∂, then:

$$1 - \partial - 1 - ce^{-\sigma t_s} \rightarrow \partial - ce^{-\sigma t_s} \rightarrow t_s - \frac{1}{\sigma} \cdot \ln\left(\frac{c}{\sigma}\right) \tag{2. 18}$$

$$\text{Where, } \sigma = \zeta \cdot \omega_n \text{ and } c = \frac{\omega_n}{\omega_d} \text{ and } \omega_d = \sqrt{1-\zeta^2} \cdot \omega_n \tag{2. 19}$$

Using settling time as *30 milliseconds*, and the error band as *5%* and damping ratio to be *0.7*, the natural frequency is obtained to be 158,6859. Back substituting $k_p = 222{,}1603$ and $k_i = 25181{,}22$. Back substituting these values into the digital loop filter coefficients:

$$B_0 = \left(\frac{2 \cdot k_p+k_i \cdot T}{2}\right) \text{ and } B_1 = \left(\frac{2 \cdot k_p-k_i \cdot T}{2}\right) \tag{2. 20}$$

For 50 kHz run rate of the PLL, B_0 = 223,4194 and B_1 = -220,901.

2.2.1.2 Adaptive Notch Filter Design

The notch filter used in the PLL shown in Figure (2.1) requires a double reduction of the grid frequency. The grid frequency, although stable, can vary somewhat, and with the increase in renewable content major variations are possible. Therefore, in order to accurately write a grid frequency, a compatible notch filter is used. The standard notch filter number is “s-domain” as shown in Equation (2.21):

$$H_{n\_f}(s) = \frac{s^2+2\cdot\zeta_2\cdot\omega_n s+\omega_n^2}{s^2+2\cdot\zeta_1\cdot\omega_n s+\omega_n^2} \tag{2. 21}$$

where $\zeta_2 \ll \zeta_1$ for notch action to occur.

Discretizing Equation (2.21) using zero order hold, , $s = \frac{(z-1)}{T}$, the equation is reduced to:

$$H_{n\_f}(z) = \frac{z^2+(2\cdot\zeta_2\cdot\omega_n\cdot T-2)\cdot z+\left(-2\cdot\zeta_2\cdot\omega_n\cdot T+2\omega_n^2\cdot T^2+1\right)}{z^2+(2\cdot\zeta_1\cdot\omega_n\cdot T-2)\cdot z+\left(-2\cdot\zeta_1\cdot\omega_n\cdot T+2\omega_n^2\cdot T^2+1\right)} - \frac{B_0+B_1\cdot z^{-1}+B_2\cdot z^{-2}}{A_0+A_1\cdot z^{-1}+A_2\cdot z^{-2}} \tag{2. 22}$$

In the Equation (2.22) the maps are in digital format with two poses zero and the notch filter coefficients can be changed variable as the grid frequency varies by calling the posterior system measuring the coefficients based on the grid rating scale. For example, taking $\zeta_2 = 0{,}00001$ and $\zeta_1 = 0{,}1$ ($\zeta_2 \ll \zeta_1$), the notch response as shown in Figure (2.2) of the 50 Hz and 60 Hz grid, where the coefficient is calculated based on the grid. The notch filter response is shown below in Figure (2.2).

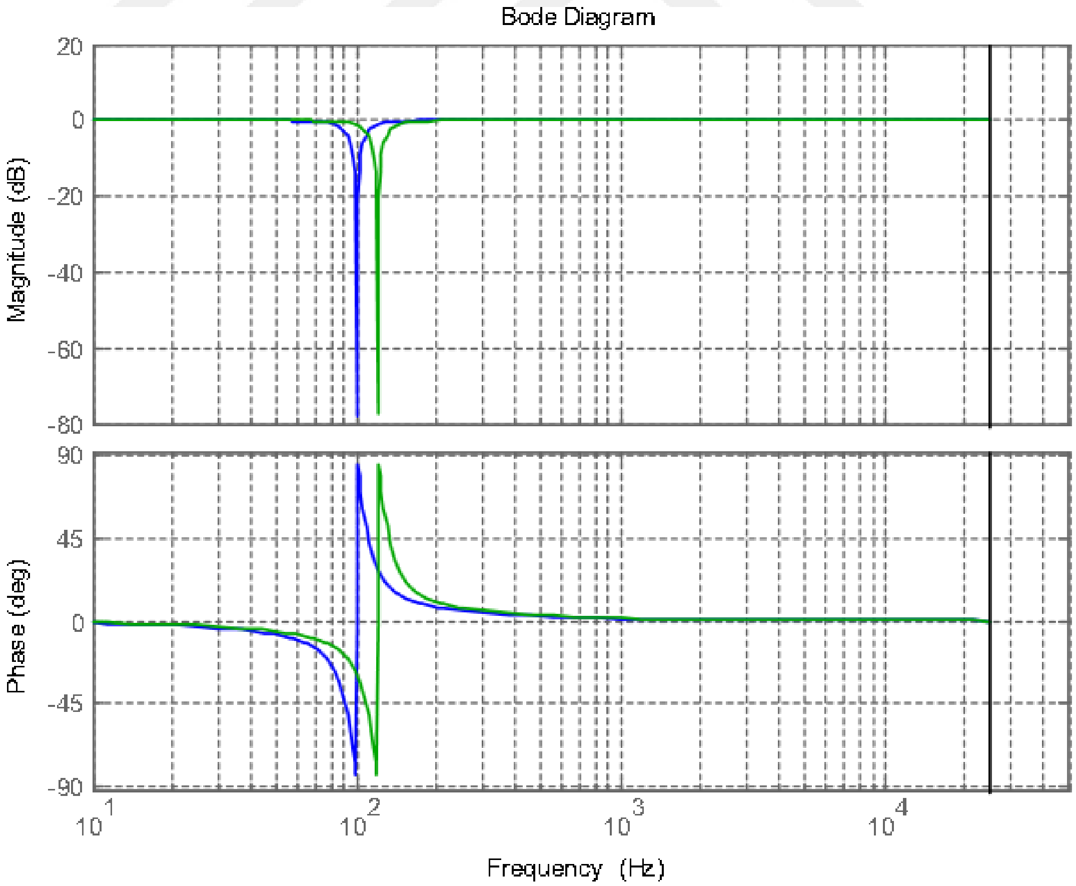


Figure 2.3 The response of the notch filter

2.2.1.3 Sine and Cosine Generation

The PLL uses sine and cosine calculation, these calculations can consume large number of cycles in a typical microcontroller. To avoid this issue, the sine and cosine value is generated in this module by applying the principle of integration.

$$y(t + \Delta t) = y(t) + \frac{dy(t)}{dt} \cdot \Delta t \tag{2. 23}$$

For sine and cosine signal, this reduces to:

$$sin(t + \Delta t) = sin(t) + \frac{dsin(t)}{dt} \cdot \Delta t - sin(t) + cos(t) \cdot \Delta t \tag{2. 24}$$

$$cos(t + \Delta t) = cos(t) + \frac{dcos(t)}{dt} \cdot \Delta t - cos(t) + sin(t) \cdot \Delta t \tag{2. 25}$$

2.2.1.4 Simulating the Phase Locked Loop for Varying Conditions

It is important to mimic the PLL behavior with different grid shapes. Fixed-point processors are used, at a lower cost to multiple grids connected by converters. IQ Mathis is an easy way to look at fixed points numbers with a decimal point. The C2000IQ math library provides built-in functions that can make it easy to manage a decimal point by an editor. However, coding in a fixed location can have additional issues of flexibility and accuracy; therefore, it is better to mimic the behavior of fixed-point processors in the simulation environment. Therefore, MATLAB® is used to mimic and point a Q-point where the algorithm needs to work. Below, the MATLAB text uses a fixed-point tool that checks the PLL algorithm with a variable grid. You can check in the section ***Appendix A*** more details about the codes.

General block diagram of interrupt service routine (ISR) for notch filter is shown below:

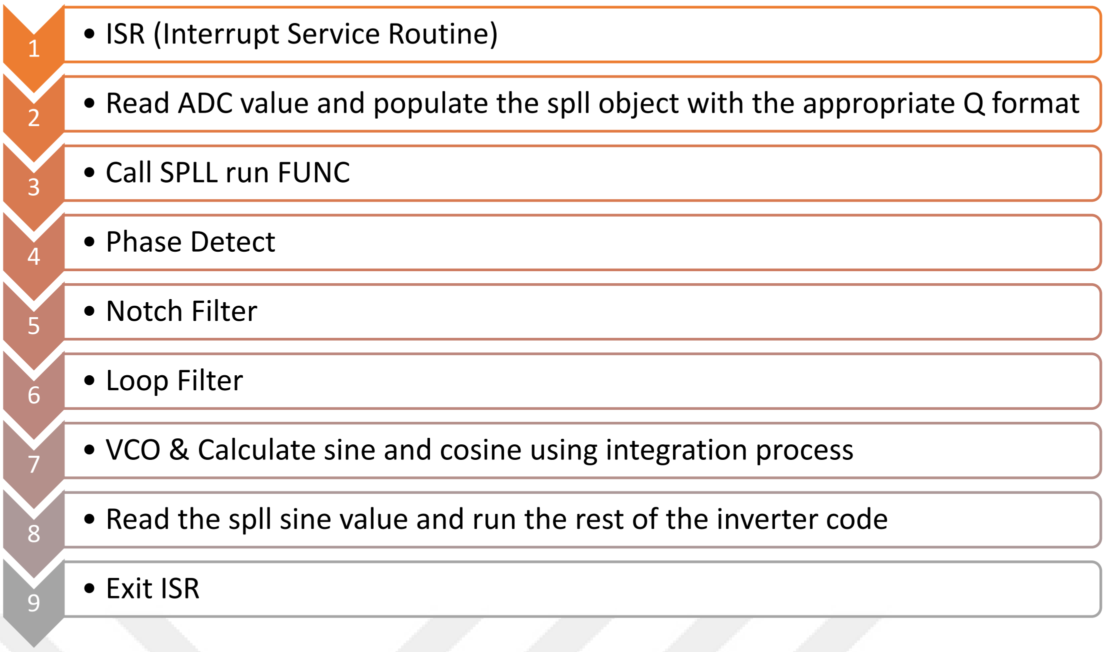

1
• ISR (Interrupt Service Routine)
2
• Read ADC value and populate the spll object with the appropriate Q format
3
• Call SPLL run FUNC
4
• Phase Detect
5
• Notch Filter
6
• Loop Filter
7
• VCO & Calculate sine and cosine using integration process
8
• Read the spll sine value and run the rest of the inverter code
9
• Exit ISR

Figure 2.4 Filter Interrupt flow chart

Figure (2.5), (2.6), (2.7), (2.8), (2.9) shows the results of the varying grid condition on the PLL simulation.

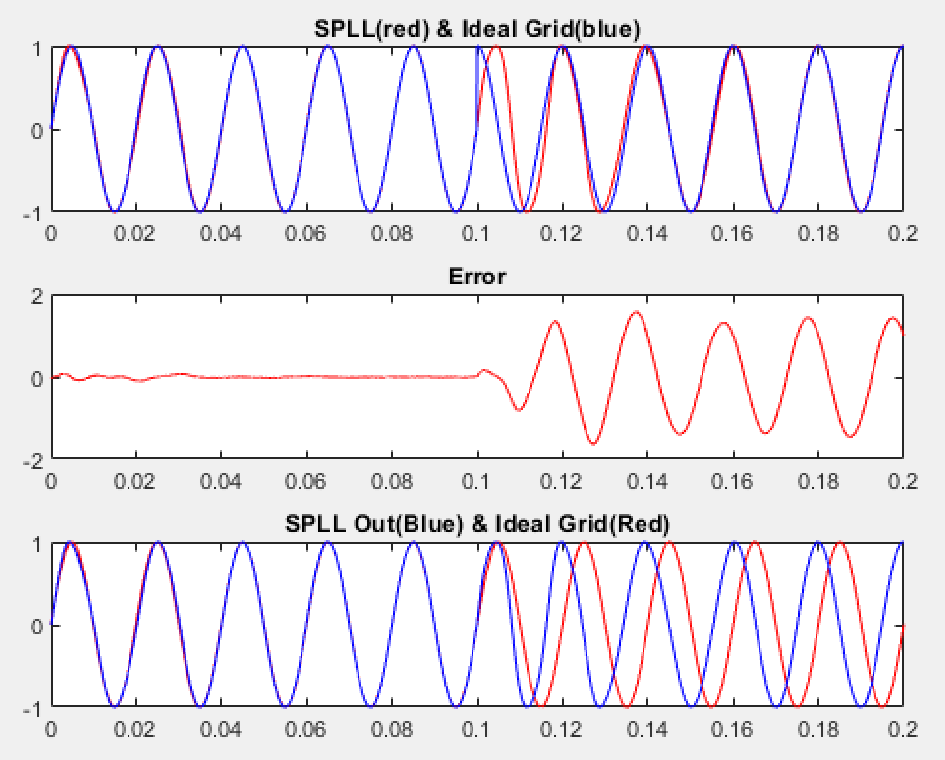


Figure 2.5 PLL Response to Varying Grid Conditions

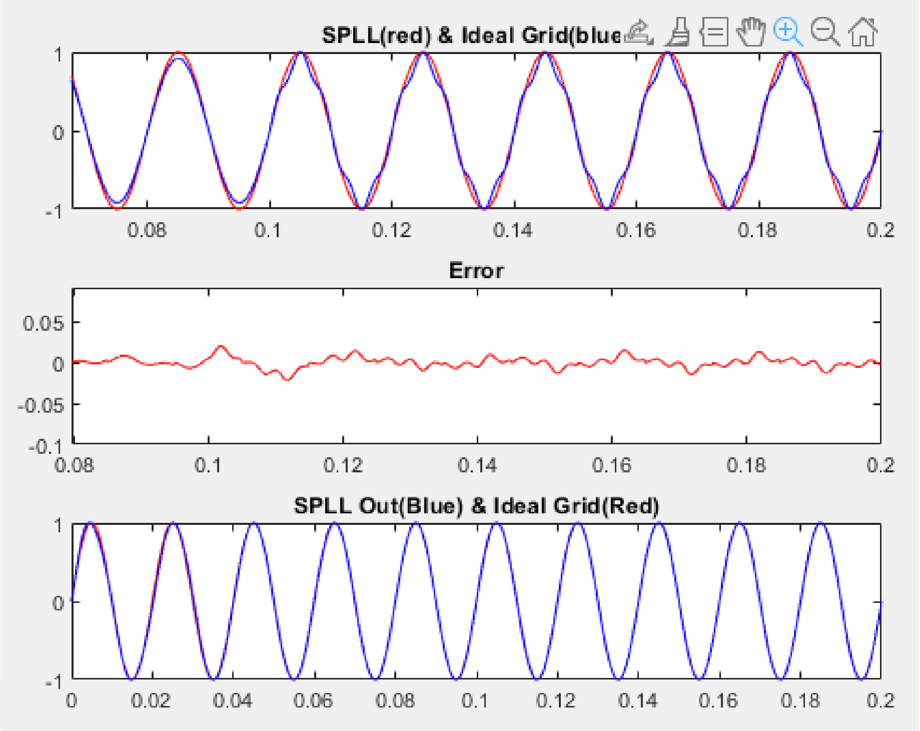


Figure 2.6 PLL Response to Varying Grid Conditions

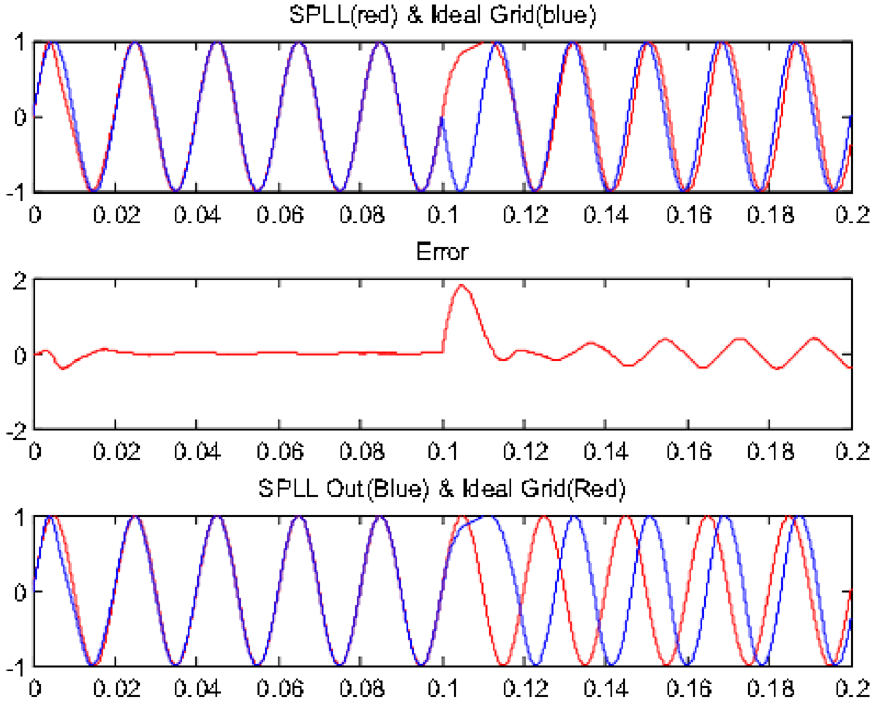


Figure 2.7 PLL Response to Varying Grid Conditions

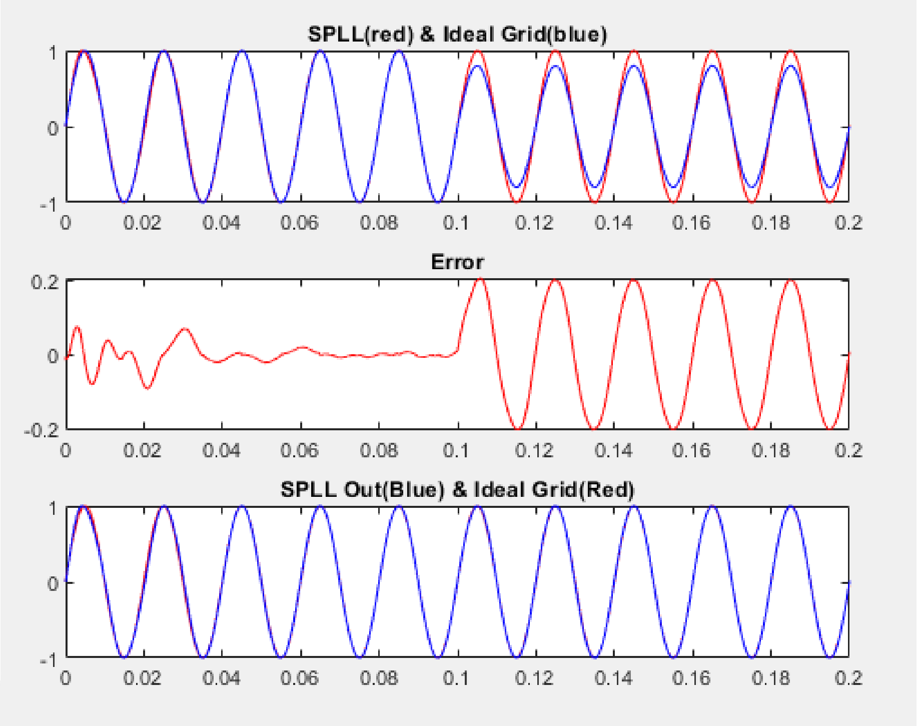


Figure 2.8 PLL Response to Varying Grid Conditions

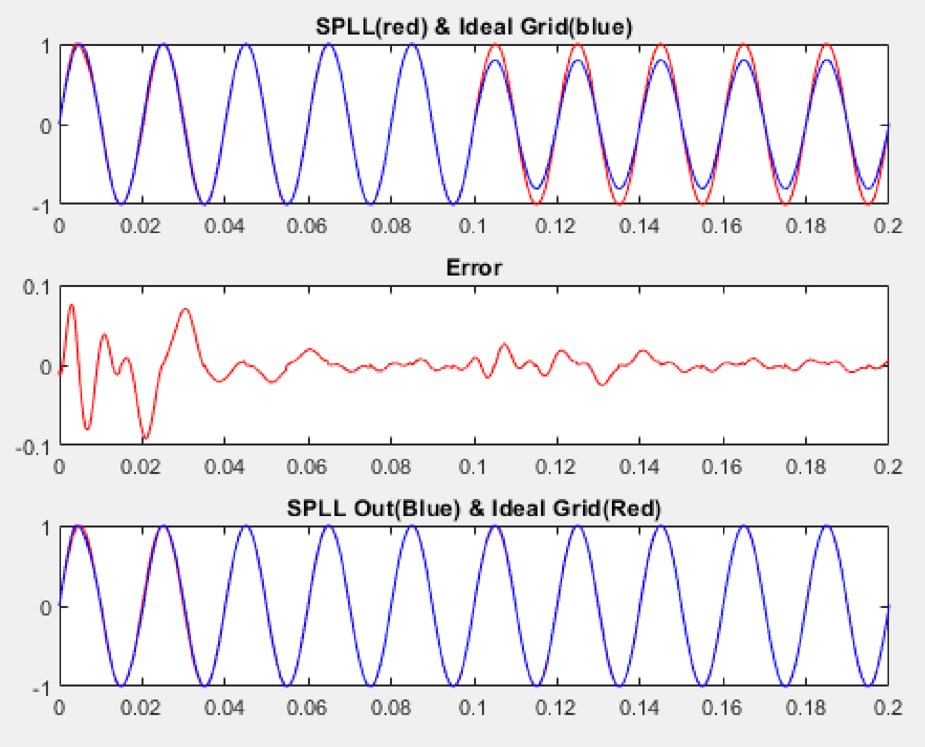


Figure 2.9 PLL Response to Varying Grid Conditions

2.2.2 Orthogonal Signal Generator

As discussed earlier, the design of the PLL single-phase grid software is deceptive due to duplicate part of the existing grid frequency in the output acquisition phase. A notch filter was previously used to complete the component and satisfactory results were obtained. Alternatively, to create a PD output line, use an orthogonal signal generating system and use a park conversion. The synchronized reference framework for PLL is then used for a single-phase application. A working diagram of such a PLL is shown in Figure (2.10), which contains a PD that includes an orthogonal signal generator as well as a park conversion, LPF and VCO. [6]

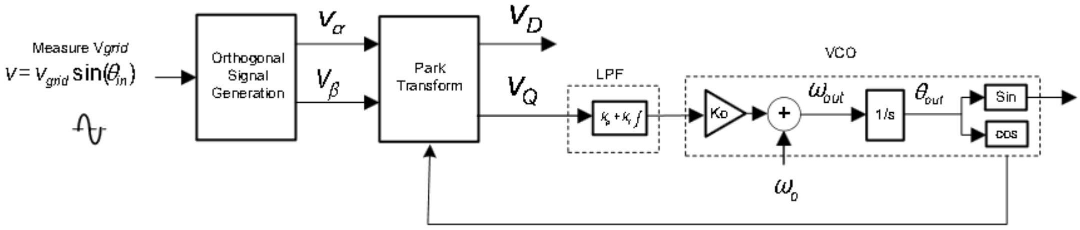


Figure 2.10 OSG Based Single Phase PLL

The orthogonal part from the input voltage signal can be produced in various ways such as transport delays, Hilbert transform, and so on. The most widely discussed approach is to use a second-order link as proposed in ‘A New Single Phase PLL Structure Based on Second Order Generalized Integrator’, Mihai Ciobotaru, et al, PESC’06. This method is useful as it can be used for tuning by selecting an orthogonal signal generator to reject other frequencies other than the grid frequency. [6]

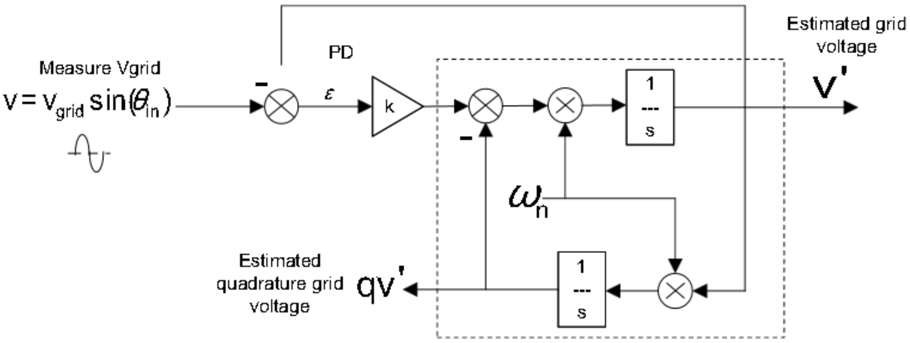


Figure 2.11 Second Order Generalized Integrator for Orthogonal Signal Generation

The second order generalized integrator closed-loop transfer function can be written as:

$$H_d(s) = \frac{v'}{v}(s) = \frac{k \cdot \omega_n \cdot s}{s^2 + k \cdot \omega_n \cdot s + \omega_n^2} \tag{2. 26}$$

$$H_q(s) = \frac{qv'}{v}(s) = \frac{k \cdot \omega_n^2}{s^2 + k \cdot \omega_n \cdot s + \omega_n^2} \tag{2. 27}$$

As we discussed before, the grid frequency can change, therefore, this orthogonal signal generator must be able to tune its coefficients in case of grid frequency change. To achieve this, trapezoidal approximation is used to get the discrete transfer function as follows:

$$H_d(z) = \frac{k \cdot \omega_n \cdot \frac{2}{T_S} \cdot \frac{z-1}{z+1}}{\left(\frac{2}{T_S} \frac{z-1}{\cdot z+1}\right)^2 + k\omega_n \cdot \frac{2}{T_S} \cdot \frac{z-1}{z+1} + \omega_n^2} - \frac{(2 \cdot k \cdot \omega_n \cdot T_S) \cdot (z^2-1)}{4 \cdot (z-1)^2 + (2 \cdot k \cdot \omega_n \cdot T_S) \cdot (z^2-1) + (\omega_n \cdot T_S)^2 \cdot (z+1)^2} \tag{2. 28}$$

Now, using $x = 2 \cdot k \cdot \omega_n \cdot T_s$ and $y = (\omega_n \cdot T_s)^2$:

$$H_d(z) = \frac{\frac{x}{x+y+4} + \left(\frac{-x}{x+y+4}\right) \cdot z^{-2}}{1 - \left(\frac{2(4-y)}{x+y+4}\right) \cdot z^{-1} - \left(\frac{x-y-4}{x+y+4}\right) \cdot z^{-2}} - \frac{b_0 + b_2 \cdot z^{-2}}{1 - a_1 \cdot z^{-1} - a_2 \cdot z^{-2}} \tag{2. 29}$$

Similarly,

$$H_q(z) = \frac{\frac{k \cdot y}{x+y+4} + 2 \cdot \left(\frac{k \cdot y}{x+y+4}\right) \cdot z^{-1} + \left(\frac{k \cdot y}{x+y+4}\right) \cdot z^{-2}}{1 - \left(\frac{2(4-y)}{x+y+4}\right) \cdot z^{-1} - \left(\frac{x-y-4}{x+y+4}\right) \cdot z^{-2}} - \frac{q \cdot b_0 + qb_1 \cdot z^{-1} + qb_2 \cdot z^{-2}}{1 - a_1 \cdot z^{-1} - a_2 \cdot z^{-2}} \tag{2. 30}$$

Once the orthogonal signal is generated, the park switch is used to locate the Q and D segments in the rotating reference frame. This is then applied to the loop filter that controls the VCO PLL. The configuration of the loop filter is similar to that described in the notch filter in Section (2.2.1.1). Additionally, the coefficients of the orthogonal signal generator can be adjusted for grid frequency and sample time (ISR frequency). The only difference is k, which determines the frequency choice of the second system connector. The second standard integration order presented can also be adjusted to remove part of the harmonic frequency from the grid monitoring application, if required. A lower k value should be selected for this purpose; However, a low k has the effect of reducing the response. Figure (2.12) shows the fifth harmonic discharge using SOGI. The implementation of this follows the implementation of SOGI, which will be discussed next; However, the details are left out.

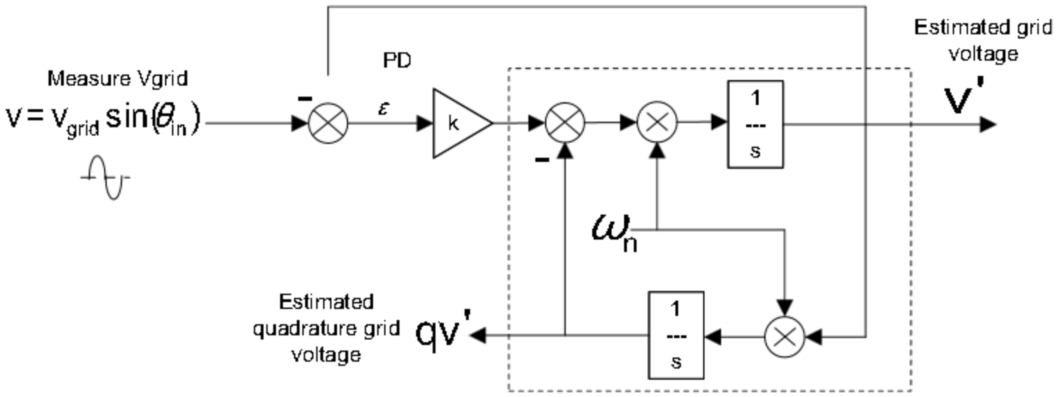


Figure 2.12 Extraction of the Fifth Harmonic Using the SOGI

Additionally, the RMS voltage of the grid can also be estimated using (2.31):

$$v_{rms} = \frac{1}{\sqrt{2}} \cdot \sqrt{v'^2 + qv'^2} \qquad (2.\,31)$$

2.2.2.1 Simulating the PLL for Varying Conditions

It is important to mimic the PLL behavior with different grid shapes. Fixed-point processors are used, at a lower cost to multiple grids connected by converters. IQ Mathis is an easy way to look at fixed points numbers with a decimal point. The C2000 IQ math library provides built-in functions that can make it easy to manage a decimal point by an editor. First, MATLAB® is used to mimic and identify Q-point where the algorithm needs to be operational. Below, the MATLAB text uses a fixed-point tool that checks the PLL algorithm with a variable grid. You can check in the section ***Appendix B*** for more details about the codes.

General block diagram of interrupt service routine (ISR) for notch filter is shown below:

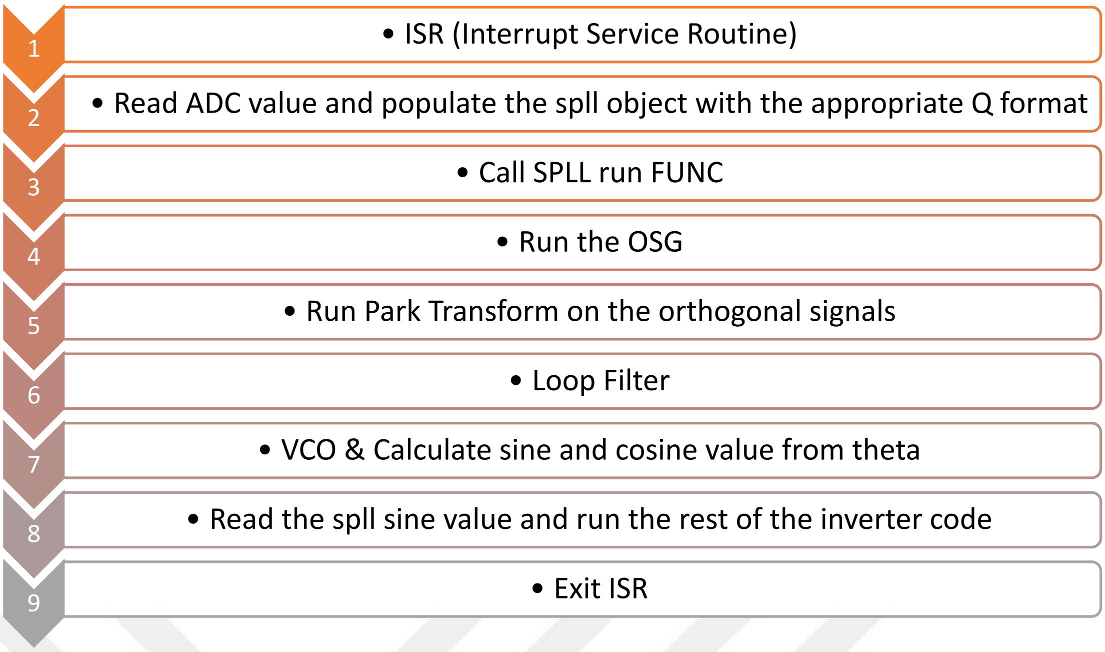

1
• ISR (Interrupt Service Routine)
2
• Read ADC value and populate the spll object with the appropriate Q format
3
• Call SPLL run FUNC
4
• Run the OSG
5
• Run Park Transform on the orthogonal signals
6
• Loop Filter
7
• VCO & Calculate sine and cosine value from theta
8
• Read the spll sine value and run the rest of the inverter code
9
• Exit ISR

Figure 2.13 Interrupt Filter flow chart

Figure (2.14), (2.15), (2.16), (2.17), (2.18) shows the results of the varying grid condition on the PLL simulation.

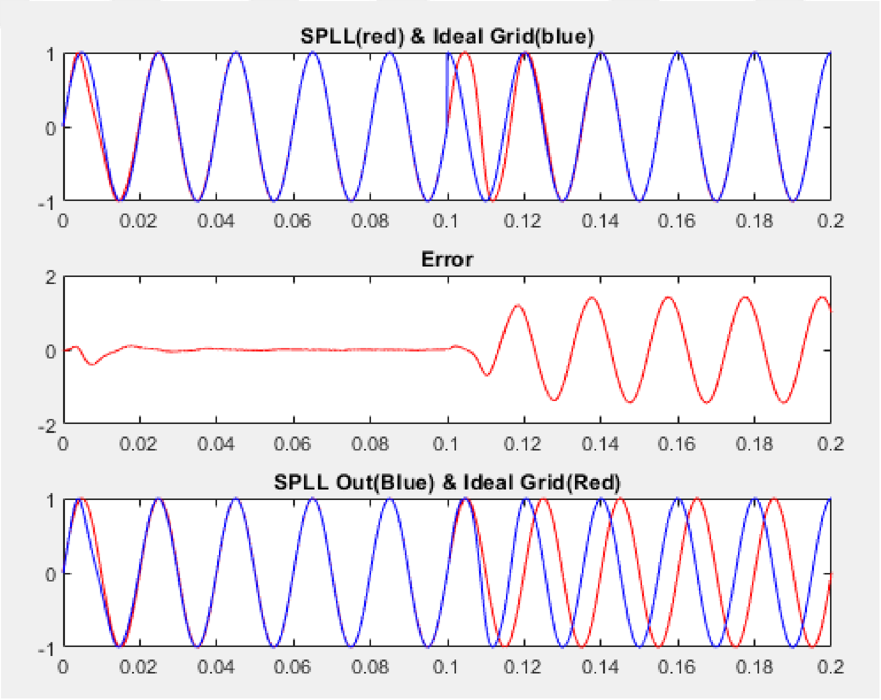


Figure 2.14 Phase Jump of 90°

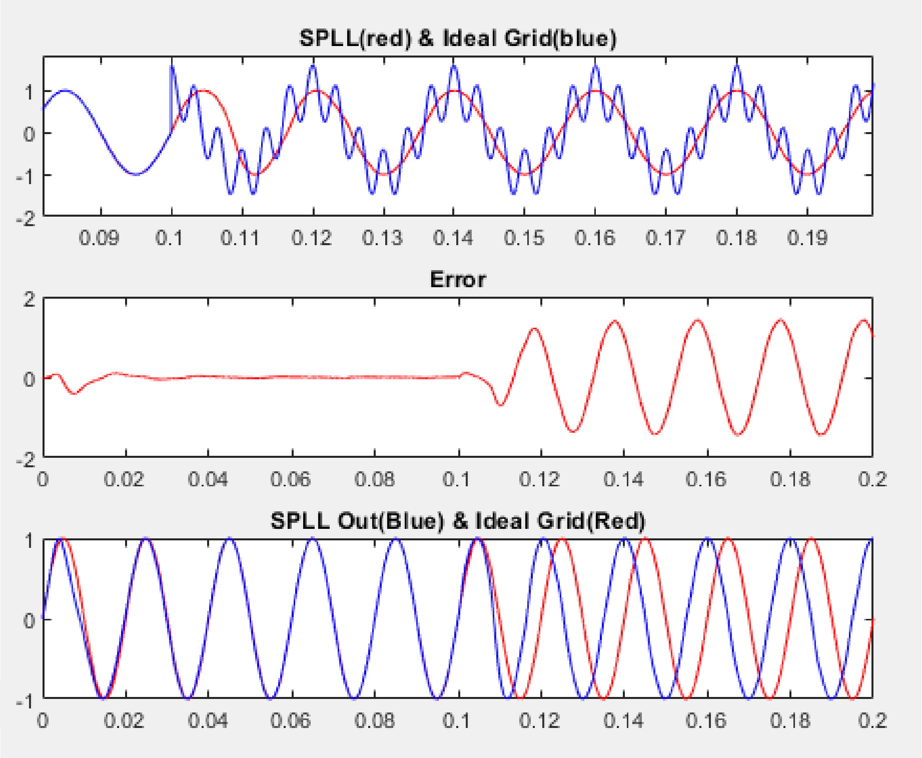


Figure 2.15 Phase Jump of 90°

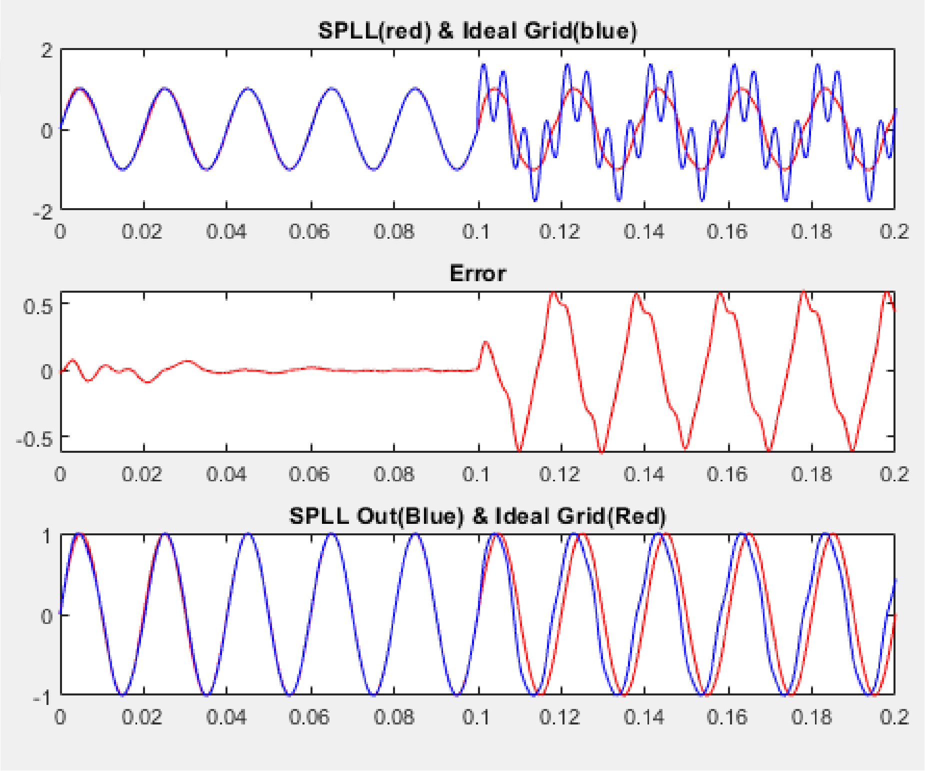


Figure 2.16 Frequency drift at mid-point highlights the need for adaptive notch filter

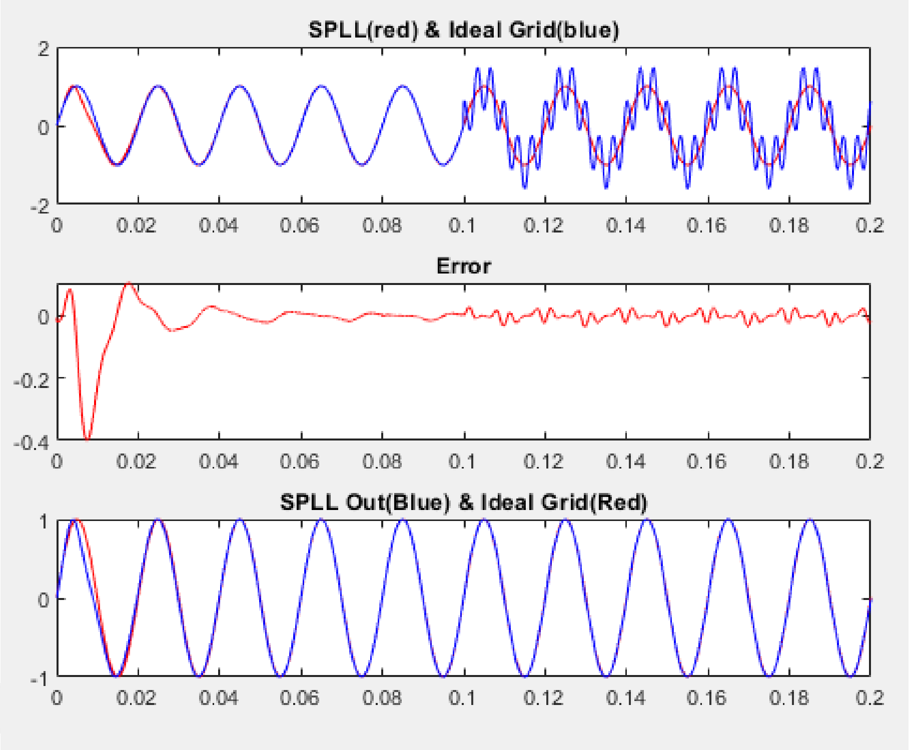


Figure 2.17 Amplitude change (Voltage Sags and Dips)

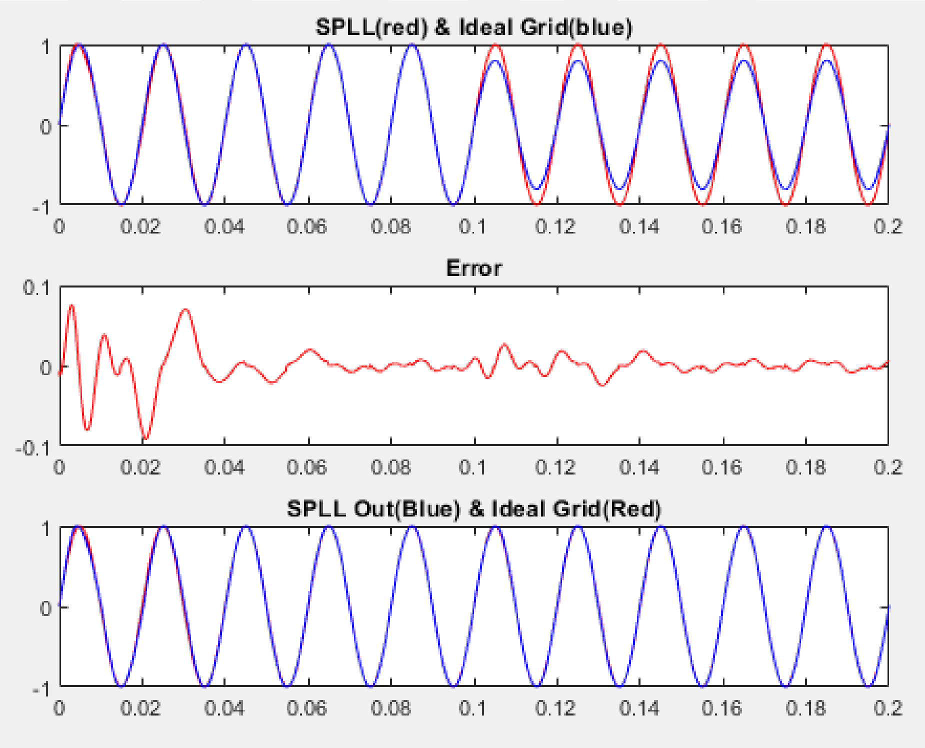


Figure 2.18 Amplitude change with harmonics (Voltage Sags and Dips)

Table 2.1 PLL and adaptive notch filter coefficients

PLL		**Adaptive Filter**	
Coefficient	Value	**Coefficient**	Value
K_i	1800	K	0,0085
K_p	60	V	0,125
K_t	1,0	T_s	50 µs
T_s	50 µs		

N = 1000 samples are taken for the FFT analysis and in the DFT analysis, IEC 61000-4-7 standards (which suggest using the windows with the period of $T = 10/f$ at the 5 Hz resolution for f = 50 Hz) are considered. Signals are windowed with a rectangular window and interharmonic group method is used to calculate the interharmonic with FFT and DFT.

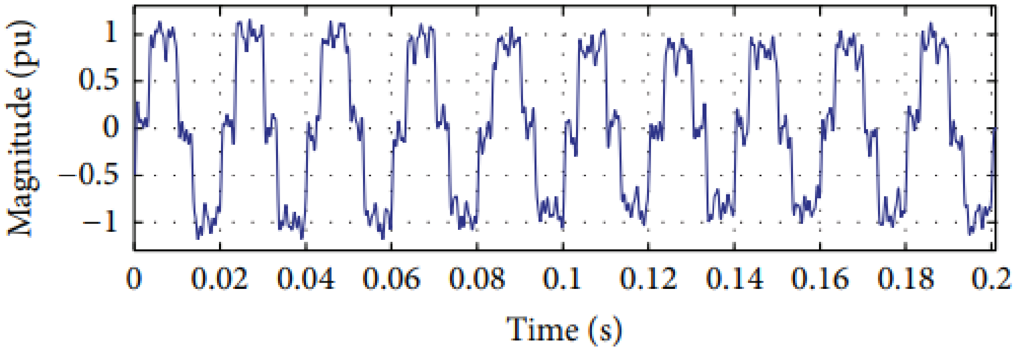
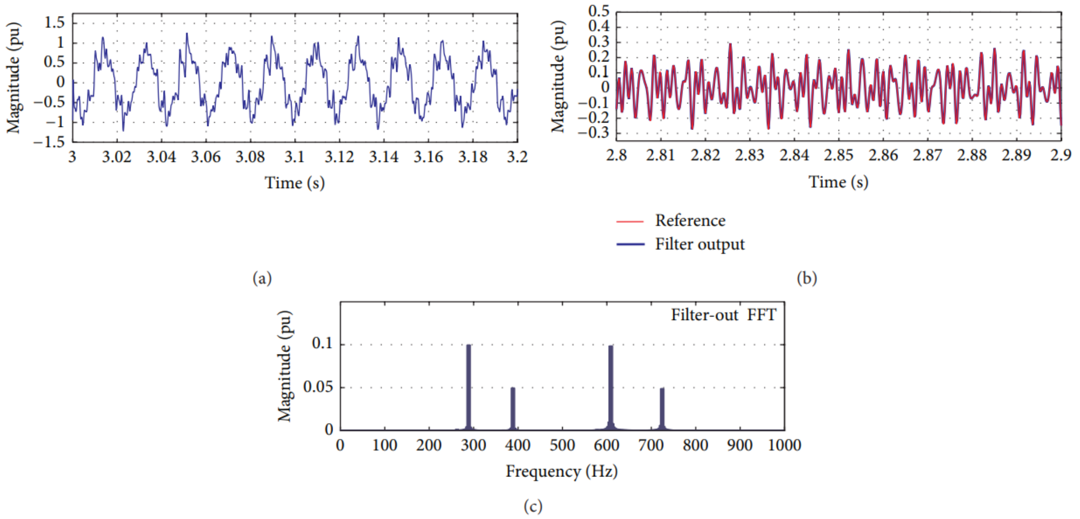


Figure 2.19 The signal $x(t)$ for f = 50 Hz and T = 0.2 s.



(a) (b)

(c)

Figure 2.20 Under the multiple disturbances; (a) the signal, $x(t)$, (b) reference-filter output signal, and (c) FFT of the filter output.

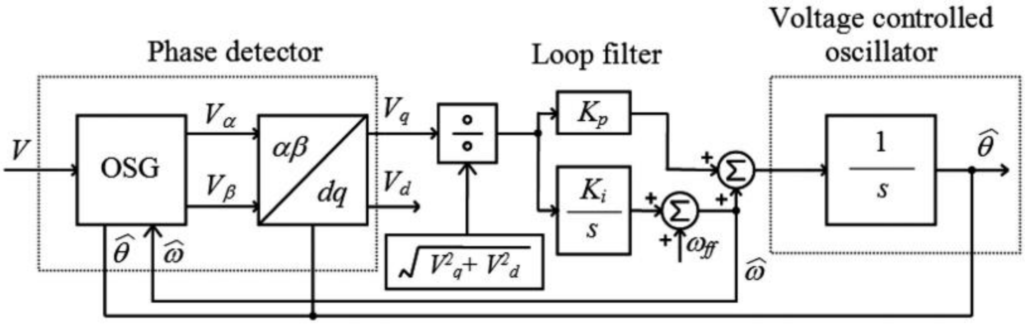


Figure 2.21 Single-phase PLL application with OSG

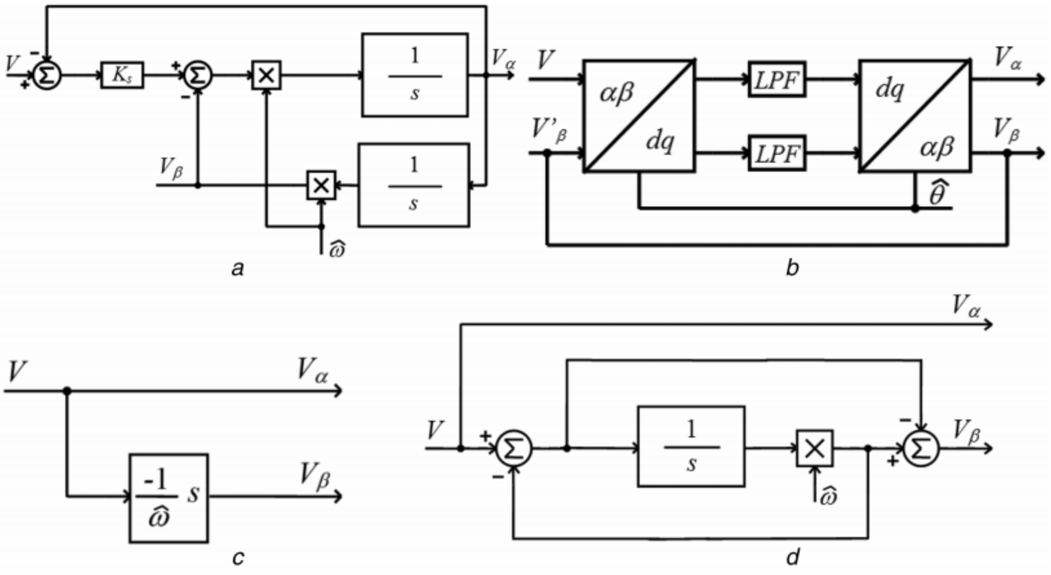


Figure 2.22 Conventional OSG; a) SOGI, b) Park, c) Derivative, d) APF PLL

Responses in different cases shown below.

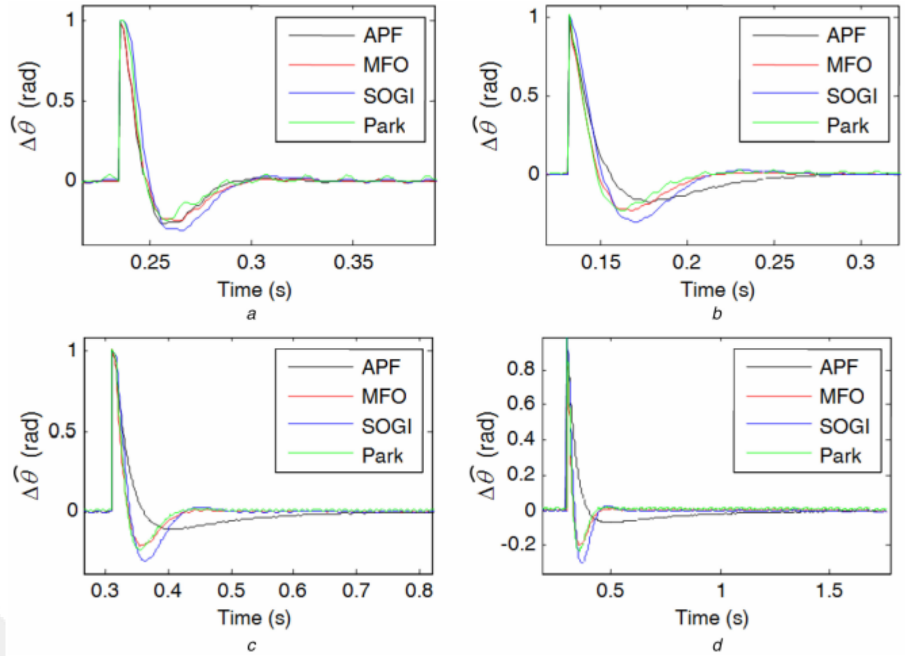


Figure 2.23 Responses of the estimated angle error for 1 rad step change in PLL input phase for APF, Park, MFO and SOGI OSG filters, for PLL with the following disturbance attenuation at 2ω / (a) Atten@2ω = −20 dB, (b) Atten@2ω = −25 dB, (c) Atten@2ω = −30 dB, (d) Atten@2ω = −35 dB

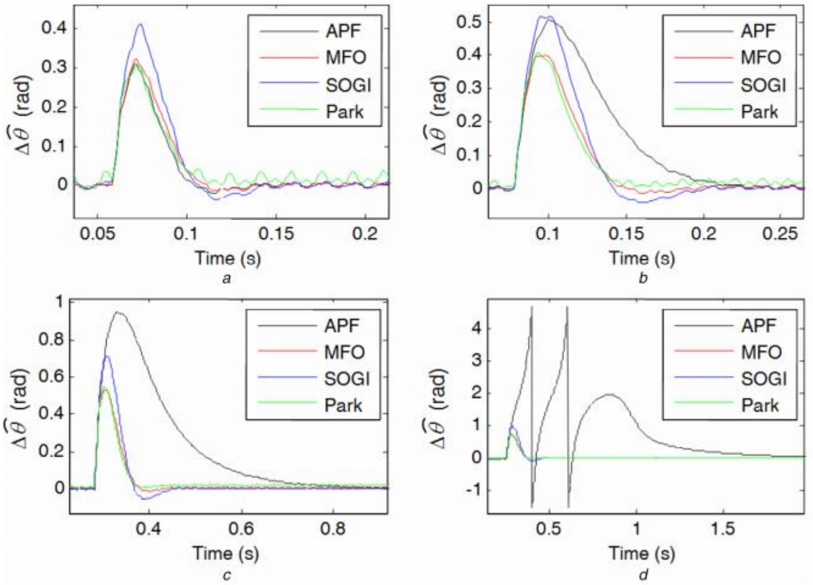


Figure 2.24 Responses of the estimated angle error for 50 rad/s step change in PLL input frequency for APF, Park, MFO and SOGI OSG filters, for PLL with the following disturbance attenuation at 2ω / (a) Atten@2ω = −20 dB, (b) Atten@2ω = −25 dB, (c) Atten@2ω = −30 dB, (d) Atten@2ω = −35 Db

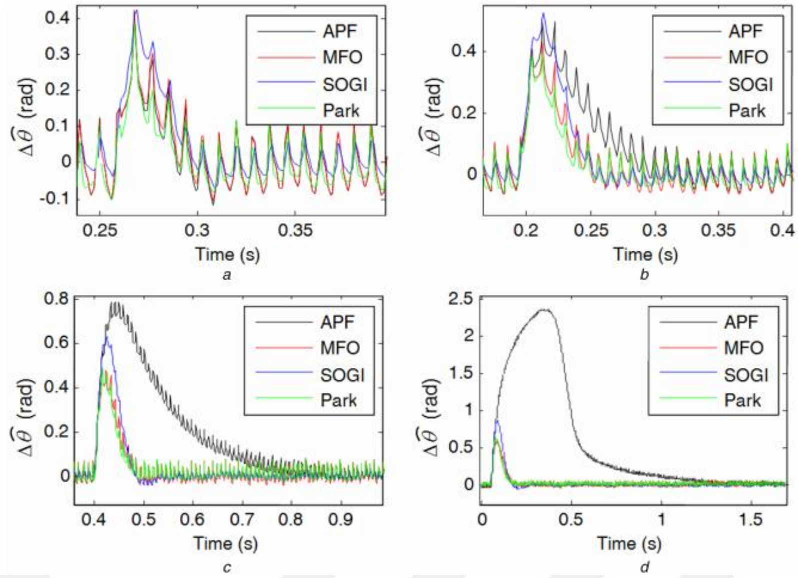


Figure 2.25 Responses of the estimated angle error for 50 rad/s step change in PLL input frequency for APF, Park, MFO and SOGI OSG filters, for square-wave input and for PLL with the following disturbance attenuation at 2ω / (a) Atten@2ω = −20 dB, (b) Atten@2ω = −25 dB, (c) Atten@2ω = −30 dB, (d) Atten@2ω = −35 Db

2.3 Discretizing the Phase Locked Loop Model

The second order low pass filter and PI controller part in the classical phase locked loop structure discussed in this section will be discretized and this structure will be made to be implemented on the real hardware software.

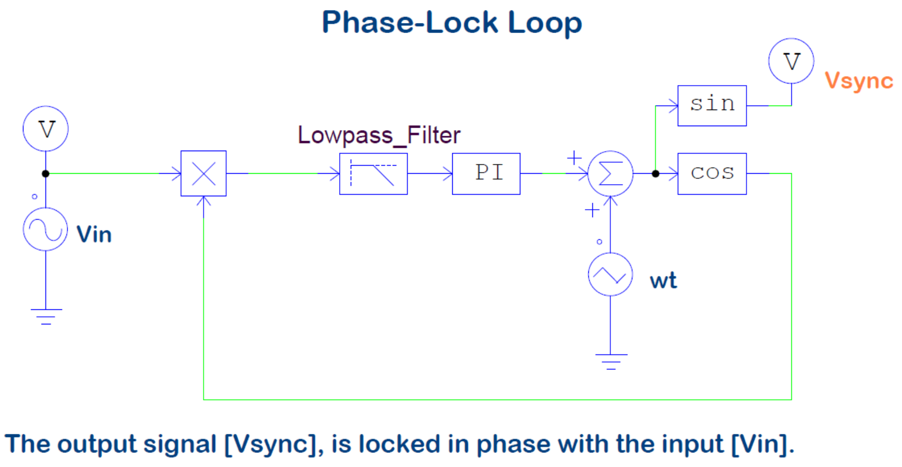

Phase-Lock Loop
V
Vin
Lowpass_Filter
PI
+
+
wt
sin
cos
V
Vsync
The output signal [Vsync], is locked in phase with the input [Vin].

Figure 2.26 Classic PLL PSIM model

2.3.1 Separation of the Second Order Low Pass Filter

In the s domain of the second-order low-pass filter, the transfer function can be written as in equation (2.32).

k →refers to Gain

ξ →refers to Damping Ratio

f_c →refers to Cut-off Frequency $(\omega_c = 2\pi f)$

$$G(s) = \frac{k \cdot \omega_c^2}{s^2 + 2\xi w_c \cdot s + w_c^2} \tag{2. 32}$$

In order to transform the transfer function in the continuous time s domain into discrete time, it must first be transformed into a z domain. There are various methods for this. For this transformation, the bilinear transformation (Tustin Transformation) method will be used. This transformation can also be done in the MATLAB environment.

As an example, let's assume that the following values are given like this;

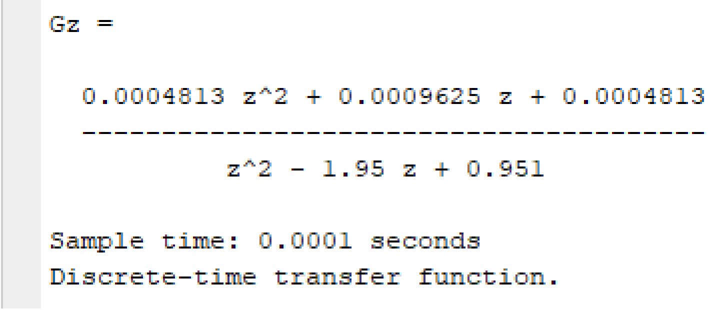
$k = 2$

$\xi = 0{,}8$

$f_c = 50$

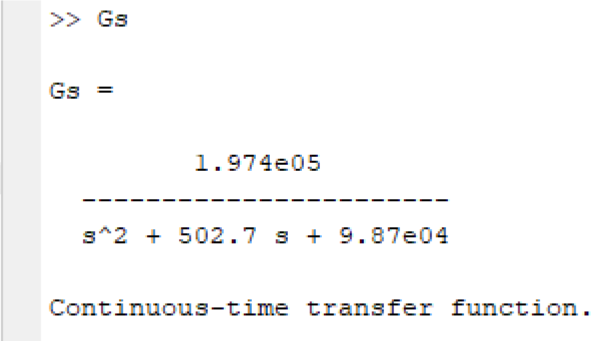
The second-order low-pass filter with these parameters can be converted to z-domain with a sampling time of 100µs in MATLAB environment as follows.

```
clear all
clc
k = 2;
ksi = 0.8;
fc = 50;
Ts = 100e-5;
wc = 2 * pi * fc;
numerator = k * wc^2;
denominator = [1 2*ksi*wc wc^2];
Gs = tf(numerator, denominator);
Gz = c2d(Gs, Ts, 'tustin')
```



```
Gz =

  0.0004813 z^2 + 0.0009625 z + 0.0004813
  ---------------------------------------
           z^2 - 1.95 z + 0.951

Sample time: 0.0001 seconds
Discrete-time transfer function.
```

Figure 2.27 Response of MATLAB for Discrete-time transfer function



```
>> Gs

Gs =

        1.974e05
  -----------------------
  s^2 + 502.7 s + 9.87e04

Continuous-time transfer function.
```

Figure 2.28 Response of MATLAB for Continuous-time transfer function

$$G(z) = \frac{az^2+bz+c}{z^2+dz+e} \tag{2. 33}$$

Discrete time transfer function can be obtained as in Eq. (2.33). In general, the discrete time transfer function of second-order low-pass filters can be written as in Eq. (2.34) or in Eq. (2.35) if the numerator and denominator are multiplied by (z^{-2}).

$$G(z) = \frac{b_0z^2+b_1z+b_2}{z^2+a_1z+a_2} \tag{2. 34}$$

$$G(z) = \frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}} \tag{2. 35}$$

In order to operate the discrete time transfer function on the microcontroller, it must be programmable. The direct programming method can be used for this. Discrete time transfer function can generally be written as in Equation (2.36).

$$D(z) = \frac{h_0z^m+h_1z^{m-1}+\cdots+h_{m-1}z^{-1}+h_m}{a_0z^n+\cdots+a_n} \tag{2. 36}$$

If the numerator and denominator are multiplied by $(z^{-n}).X(z)$ where n is the highest degree of the denominator

$$\frac{E_2(z)}{E_1(z)} = D(z) = \frac{b_0 z^m + b_1 z^{m-1} + \cdots + b_{m-1} z^{-1} + b_m}{a_0 z^n + \cdots + a_n} \cdot \left\{\frac{z^{-n}}{z^{-n}} \cdot \frac{X(z)}{X(z)}\right\} \tag{2. 37}$$

$$E_1(z) = [a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}].X(z) \tag{2. 38}$$

$$E_1(z) = [b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}].X(z).z^{-(n-m)} \tag{2. 39}$$

Here, E_1 (z) can be expressed as the input of the system and E_2 (z) as the output of the system. The general discrete time transfer function of the second order low pass filter can also be written and programmed in the same way.

$$G(z) = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2} \tag{2. 40}$$

$$\frac{E_2(z)}{E_1(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_m}{1 + a_1 z^{-1} + a_2 z^{-2}} \cdot \left\{\frac{X(z)}{X(z)}\right\} \tag{2. 41}$$

Using Equation (2.41), the filter output can be written with E_2 (z) (2.42) as below.

$$E_2(z) = [b_0 + b_1 z^{-1} + b_2 z^{-2}].X(z) \tag{2. 42}$$

If the state variable for the filter is X (z), it can be written with equation (2.43).

$$E_1(z) = [1 + a_1 z^{-1} + a_2 z^{-2}].X(z) \tag{2. 43}$$

$$X(z) = E_1(z) - [a_1 z^{-1} + a_2 z^{-2}].X(z) \tag{2. 44}$$

Finally, if we write the difference equations for the filter,

$$E_2(k) = b_0.X(k) + b_1.X(k-1) + b_2.X(k-2) \tag{2. 45}$$

$$X(k) = E_1(k) - a_1.X(k-1) - a_2.X(k-2) \tag{2. 46}$$

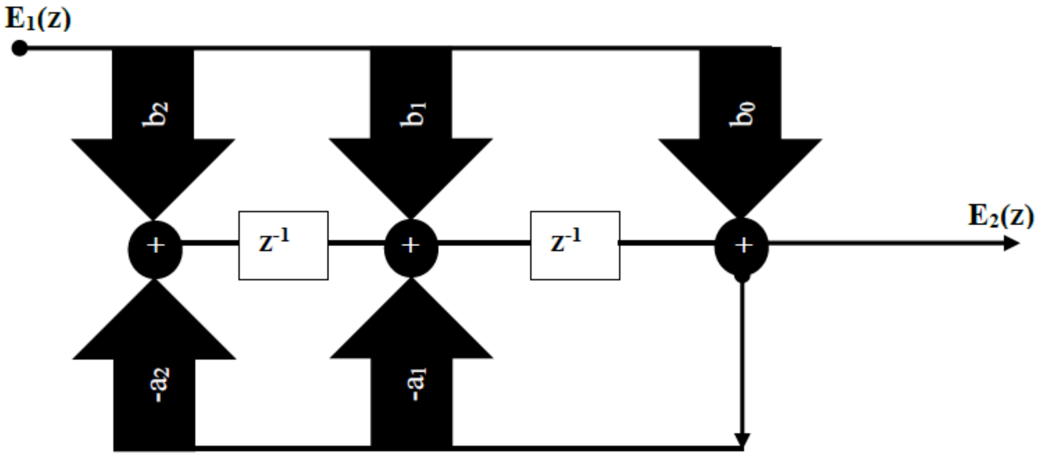
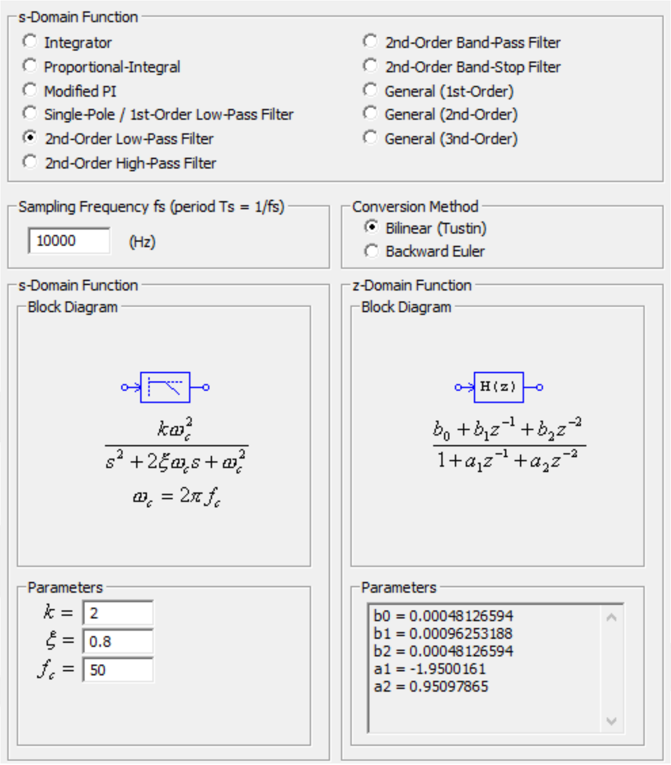


Figure 2.29 Programming block diagram for a second order low pass filter

The discrete time transfer function coefficients of the second order low pass filters will also be calculated according to the values of gain, damping ratio and cutoff frequency. These coefficient calculations were made with the tool in the Simulink environment.



s-Domain Function

○ Integrator
○ Proportional-Integral
○ Modified PI
○ Single-Pole / 1st-Order Low-Pass Filter
● 2nd-Order Low-Pass Filter
○ 2nd-Order High-Pass Filter
○ 2nd-Order Band-Pass Filter
○ 2nd-Order Band-Stop Filter
○ General (1st-Order)
○ General (2nd-Order)
○ General (3nd-Order)

Sampling Frequency fs (period Ts = 1/fs)
10000 (Hz)

Conversion Method
● Bilinear (Tustin)
○ Backward Euler

s-Domain Function
Block Diagram

$$\frac{k\omega_c^2}{s^2+2\xi\omega_c s+\omega_c^2}$$

$$\omega_c = 2\pi f_c$$

Parameters
k = 2
ξ = 0.8
f_c = 50

z-Domain Function
Block Diagram

$$\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}}$$

Parameters
b0 = 0.00048126594
b1 = 0.00096253188
b2 = 0.00048126594
a1 = -1.9500161
a2 = 0.95097865

Figure 2.30 Calculation of filter coefficients with PSIM

2.3.2 Discretization of the PI Controller

The transfer function in the s domain of the PI controller can be written as in Equation (2.47).

k = Gain

T = PI Controller Time Constant

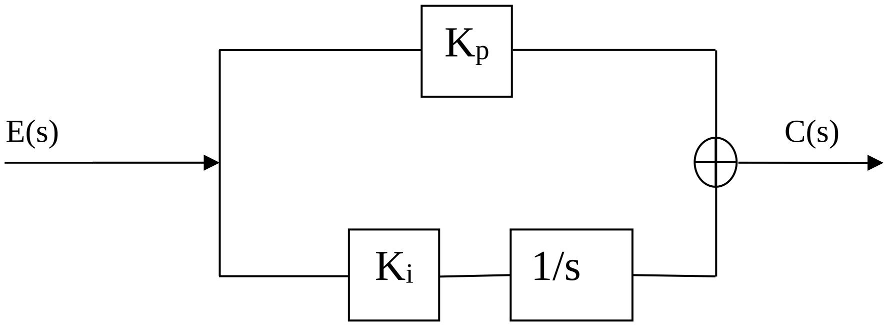


Figure 2.31 General structure of the PI controller

$$G(s) = K_p + \frac{K_i}{s} \tag{2. 47}$$

$$K_p = k \tag{2. 48}$$

$$K_i = \frac{k}{T} \tag{2. 49}$$

if we assume the values like this the transfer function of the PI controller in continuous time can be written as follows.

$$G(s) = k + \frac{k}{s.T} = k.\frac{1+s.T}{s.T} \tag{2. 50}$$

In order to discretize the transfer function in the continuous time s domain, it must first be transformed into z domain. For this transformation, the bilinear transformation (Tustin transformation) method will be used as in the second order low pass filter. This transformation can also be done in the MATLAB environment. For example, let's assume that;

$k = 2$ and

$T = 1\ ms$

A PI controller with these parameters can be converted to z domain with sampling time of $T_s = 100\ \mu s$ in MATLAB environment as follows.

```
clear all
clc
k = 2;
T = 0.001;
numerator = [k *T k];
denominator = [T 0];
G_s= tf(numerator, denominator);
T_s= 100e-6;
G_z= c2d (G_s, T_s, "tustin")
```

$$G(z) = \frac{2{,}1.z-1{,}9}{z-1} \tag{2. 51}$$

Discrete time transfer function can be obtained as in Eq. (2.51). In general, the discrete time transfer function of the PI controller can be written as in Eq. (2.52) or if the numerator and denominator of the rightmost expression is multiplied by z in Eq. (2.53).

$$G(z) = K_p + K_i \cdot \frac{Ts}{2} \cdot \frac{1+z^{-1}}{1-z^{-1}} \tag{2. 52}$$

$$G(z) = K_p + K_i \cdot \frac{Ts}{2} \cdot \frac{z+1}{z-1} \tag{2. 53}$$

In order to run the obtained discrete time transfer function on the microcontroller, it must be programmable by direct programming method as in the low pass filter.

$$G(z) = K_p + K_i \cdot \frac{Ts}{2} \cdot \frac{1+z^{-1}}{1-z^{-1}} \tag{2. 54}$$

$$K_i \cdot \frac{Ts}{2} = K_x \tag{2. 55}$$

If the expression G (z) is rearranged,

$$G(z) = \frac{K_p + K_x + (K_x - K_{p)}\ z^{-1}}{1-z^{-1}} \tag{2. 56}$$

Let’s assume that,

$$K_p + K_x = b_0 \tag{2. 57}$$

$$K_x - K_p = b_1 \tag{2. 58}$$

so that the Equation (2.58) become as Equation (2.59).

$$G(z) = \frac{b_0 + b_1 \cdot z^{-1}}{1-z^{-1}} \tag{2. 59}$$

We can obtain the difference equations by multiplying the numerator and denominator of the expression $G(z)$ obtained in its simplest form by $X(z)$.

$$\frac{E_2(z)}{E_1(z)} = \frac{b_1 \cdot z^{-1} + b_0}{1 - z^{-1}} \cdot \frac{X(z)}{X(z)} \tag{2. 60}$$

Using Equation (2.59), the PI controller output can be written with $E_2(z)$ Equation (2.61).

$$E_2(z) = b_0 \cdot X(z) + b_1 \cdot z^{-1} \cdot X(z) \tag{2. 61}$$

If the state variable of the PI controller is $X(z)$, it can be written with Equation (2.63).

$$E_1(z) = X(z) - z^{-1} \cdot X(z) \tag{2. 62}$$

$$X(z) = E_1(z) + z^{-1} \cdot X(z) \tag{2. 63}$$

Finally, if we write the difference equations for the PI controller

$$E_2(k) = b_0 \cdot X(k) + b_1 \cdot X(k-1) \tag{2. 64}$$

$$X(k) = E_1(k) + X(k-1) \tag{2. 65}$$

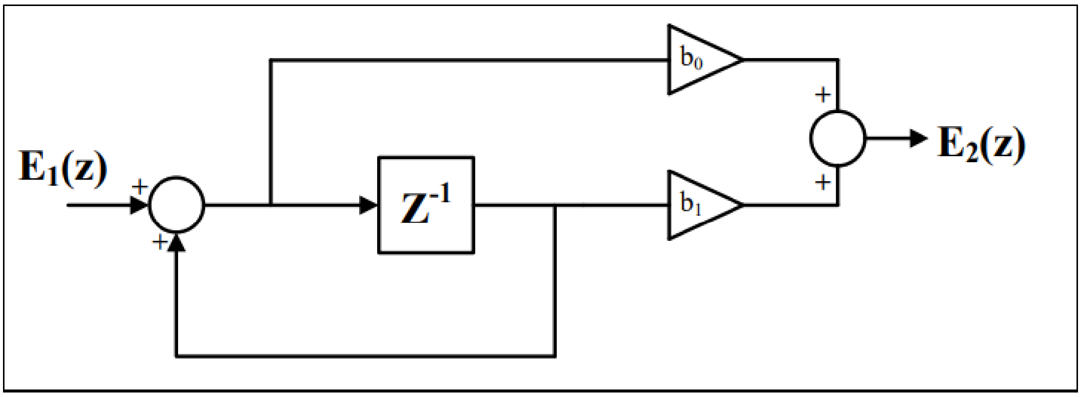
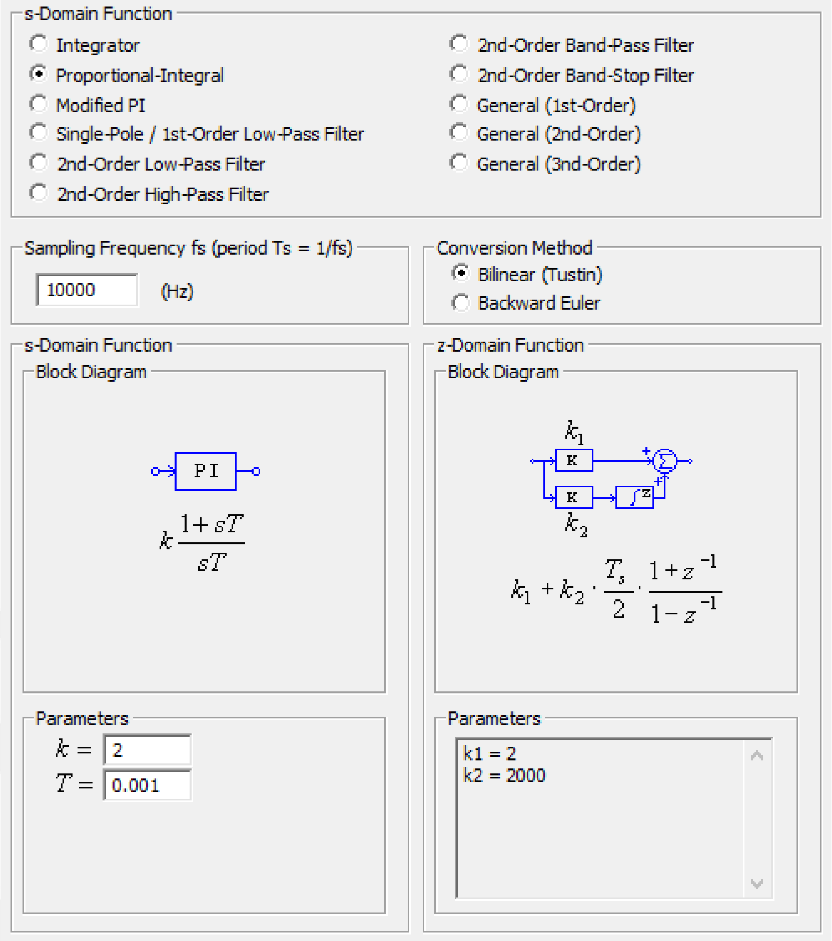


Figure 2.32 Programming block diagram for PI controller

The discrete time transfer function coefficients of PI controllers will also be calculated according to the gain and time constant values. These coefficient calculations were made with the tool in the PSIM environment.

k_1 refers to K_P and k_2 refers to K_P/T.



s-Domain Function

○ Integrator
● Proportional-Integral
○ Modified PI
○ Single-Pole / 1st-Order Low-Pass Filter
○ 2nd-Order Low-Pass Filter
○ 2nd-Order High-Pass Filter
○ 2nd-Order Band-Pass Filter
○ 2nd-Order Band-Stop Filter
○ General (1st-Order)
○ General (2nd-Order)
○ General (3nd-Order)

Sampling Frequency fs (period Ts = 1/fs)

10000 (Hz)

Conversion Method

● Bilinear (Tustin)
○ Backward Euler

s-Domain Function

Block Diagram

$$k\frac{1+sT}{sT}$$

Parameters

k = 2
T = 0.001

z-Domain Function

Block Diagram

$$k_1 + k_2 \cdot \frac{T_s}{2} \cdot \frac{1+z^{-1}}{1-z^{-1}}$$

Parameters

k1 = 2
k2 = 2000

Figure 2.33 Calculation of K_P and K_i coefficients with Simulink environment

2.3.3 Discrete Time Model of Classical PLL Structure

The discrete time block diagram of the classical PLL model is as follows.

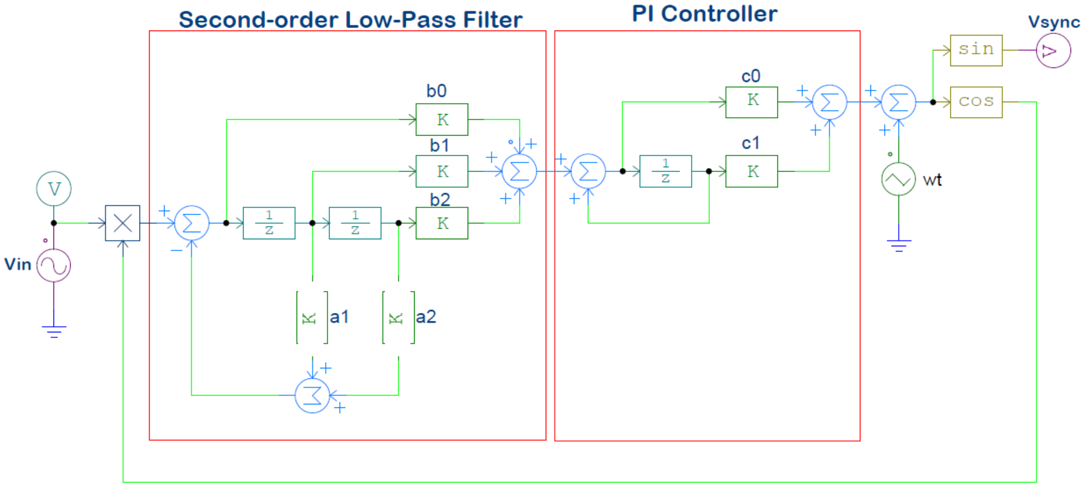


Figure 2.34 Discrete time block diagram of the classical PLL model

The synchronization signal produced by the classical PLL model, which we obtained the discrete time model, according to the sinusoidal input is obtained as follows. Thanks to the synchronization signal obtained by the PLL model, synchronization with the network will be ensured in case of disturbances in the network and frequency shifts. As an example of the discrete time block diagram of the PLL model obtained in Figure (2.34), it is seen in Figure (2.35) that it is properly locked into phase at the output against the sinusoidal input signal with phase difference. Filter coefficients for the sample are as follows.

Second order low pass filter coefficients;

b_0 = 0,00048126594

b_1 = 0,00096253188

b_2 = 0,00048126594

a_1 = -1,9500161

a_2 = 0,95097865

PI controller coefficients;

b_0= 10,4

b_1= -9,96

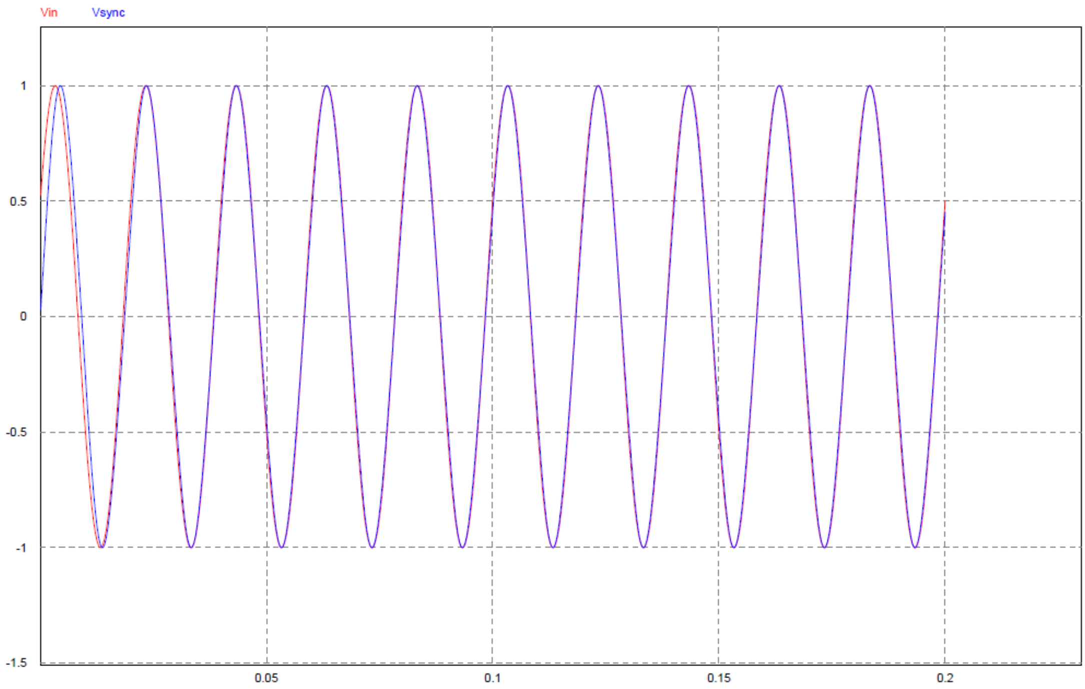


Figure 2.35 Response of digital PLL to sinusoidal input signal

2.3.4 Software Implementation of the Digital PLL Model with PSIM

There is a block on the PSIM simulation program that can be written with C / C ++ code and used in simulations. Before testing the digital PLL model obtained in Figure (2.36) on real hardware, we can implement it with this block. For this block, the input signal is the measured network voltage and the output signal is the generated synchronization signal. The output signal of the model in Figure (2.37) and the codes run on the C block are as follows.

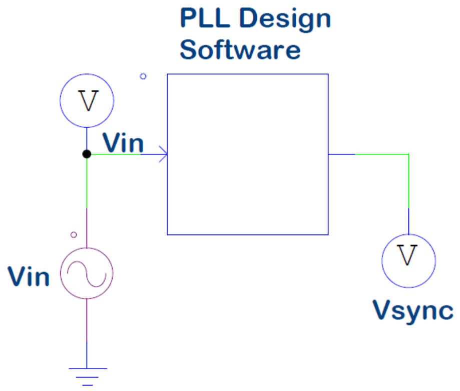


Figure 2.36 Software implementation of digital PLL

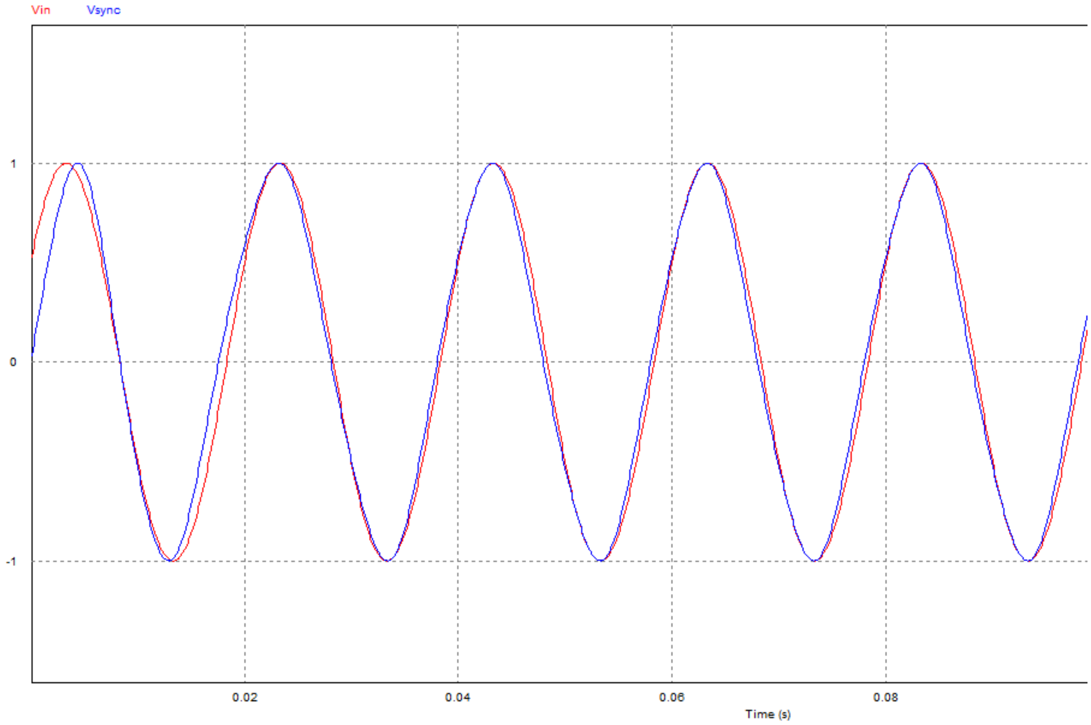


Figure2.37 Software PLL's response to sinusoidal input signal

```
 ---------------------// Low-pass Filter Coefficients and related variables//-----------------
double b0 = 0.00048126594;
double b1 = 0.00096253188;
double b2 = 0.00048126594;
double a1 = -1.9500161;
double a2 = 0.95097865;
static double Xk = 0;                   // 2nd order LPF static variable
static double Xk_1 = 0;     // 2nd order LPF static variable previous value
static double Xk_2 = 0;     // 2nd order LPF static variable second previous value
---------------------// PI controller Coefficients and related variables//-----------------
double T_s = 0.0001;
double k1 = 2;
double k2 = 10000;
double kx = k2 * (Ts/2);
double c0 = kx + k1;
double c1 = kx - k1;

static double Xk_PI = 0;    // static variable
static double Xk_1_PI = 0;              // static variable previous value

-----------------------------------// other variables//-------------------------------

static double Tri_Index = 1;
static double Cos_Out = 0;
static double Pre_Out = 0;
double input_lpf, output_lpf, input_pi, output_pi, Triangle_Out;

input_lpf = in[0] * Cos_Out;
Xk = input_lpf - a1 * X_k_1 - a2 * X_k_2;
output_lpf = X_k * b0 + Xk_1 * b1 + X_k_2 * b2;
X_k_2 = X_k_1;
X_k_1 = X_k;

input_pi = output_lpf;
X_k_PI = input_pi + X_k_1_PI;
output_pi = c0 * X_k_PI + c1 * X_k_1_PI;
X_k_1_PI = X_k_PI;

Triangle_Out = Tri_Index * 1.8;
Pre_Out = output_pi + Triangle_Out;

if(++Tri_Index > 200) {
Tri_Index = 1; }

Cos_Out = cos(Pre_Out * M_PI / 180);
out[0] = sin(Pre_Out * M_PI / 180);
```

2.4 Hardware Implementation of Software PLL

The software of the PLL model was also tested on the designed hardware. In this section, only the waveform and PLL synchronization signal of samples taken from the mains voltage are included. Sampling time is determined as 156.25 μs since 128 samples will be taken in a period. This value is calculated by considering the network period as 20ms and its frequency as 50Hz. Small fluctuations in the network frequency are calculated at certain intervals with the synchronization signal generated by the PLL, and the sampling time is dynamically adjusted to receive 128 samples in a period. In this way, all measurements to be made are calculated more accurately and reliably.

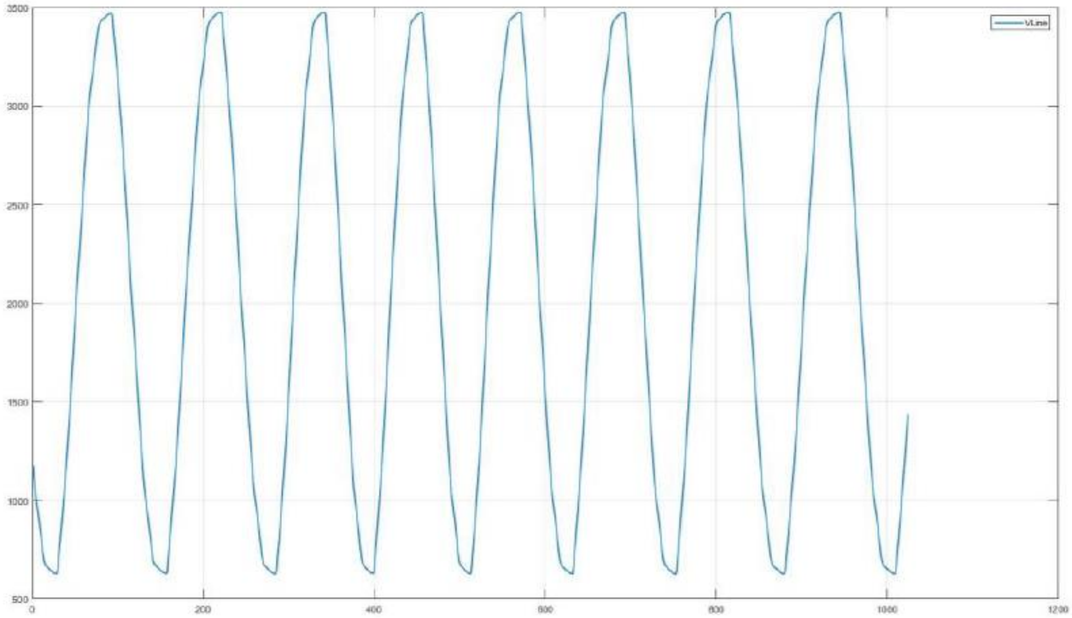


Figure 2.38 Sampled Signal from the grid voltage (***V**_{line}*)

The samples taken by the microcontroller with the serial port on the designed hardware were transferred to the MATLAB environment and the signals were plotted.

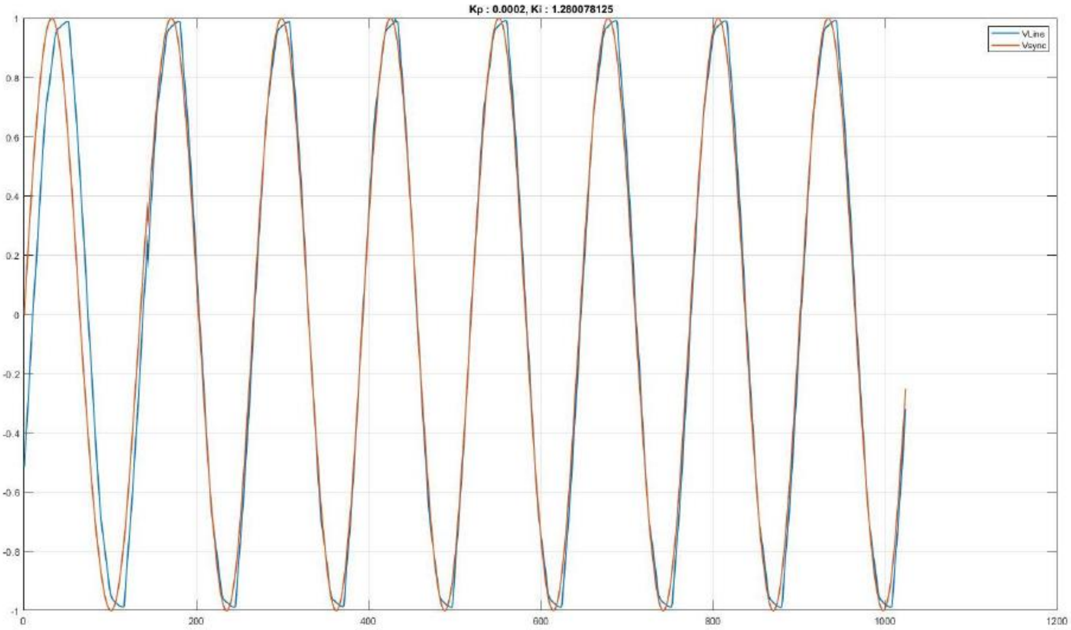


Figure 2.39 PLL output signal that locked to the grid signal (Blue=V_{line}, Red=V_{sync})

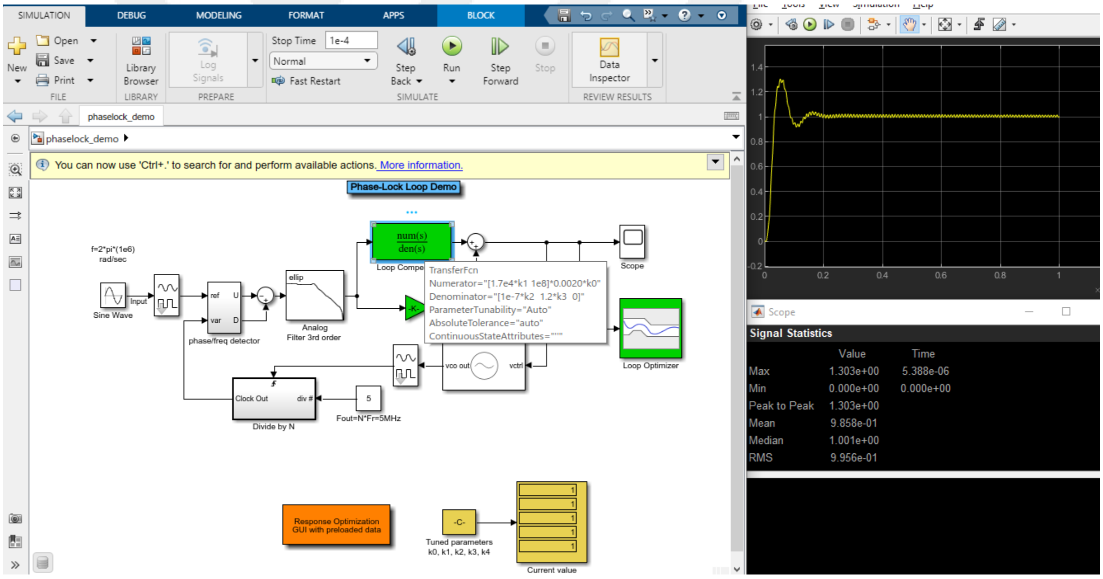
The part running the PLL algorithm in the software is as follows. This part is regularly called in the ISR where the samples were taken.

```
stPLL.wt = (stPLL.wtIndex * 360) / 128.0;
stPLL.LPF_Input = stPLL.Vin * stPLL.CosOut;
stPLL.LPF_Xk = stPLL.LPF_Input - (LPF_A1 * stPLL.LPF_Xk1) - (LPF_A2
* stPLL.LPF_Xk2);

stPLL.LPF_Output = (stPLL.LPF_Xk * LPF_B0) + (stPLL.LPF_Xk1 *
LPF_B1) + (stPLL.LPF_Xk2 * LPF_B2);

stPLL.PI_Input = stPLL.LPF_Output;
stPLL.PI_Xk = stPLL.PI_Input + stPLL.PI_Xk1;
stPLL.PI_Output = (stPLL.K_PI_x * stPLL.PI_Xk) + (stPLL.K_PI_y *
stPLL.PI_Xk1);

stPLL.Theta = stPLL.PI_Output + stPLL.wt;
stPLL.CosOut = arm_cos_f32(Deg2Rad(stPLL.Theta));
stPLL.Vsync = arm_sin_f32(Deg2Rad(stPLL.Theta));
stPLL.Vsign = signum(stPLL.Vsync);
```

In this section, various PLL models used in single phase system are presented, discrete time modeling and simulation results are given. In addition, an example is provided on the designed hardware. There are many studies on this subject in the literature. Second order generalized integrator based (SOGI) PLL structures, which can be used especially in both single phase and three phase systems, are very useful and can produce a very sensitive output without being affected by distortions in the network.

In recent years, it has become possible to achieve excellent results on digital platforms with the increase in the performance of microcontrollers and digital signal processors (DSP) and the addition of units that can perform decimal and trigonometric processing independently of the central processing unit (CPU).

For OSG simulating in MATLAB, it requires approximately: 868.451 seconds, in the contrast Notch simulating requires approximately: 264.621 seconds. According to these results a combined filter will get better results.



Stop Time 1e-4

Normal

You can now use 'Ctrl+.' to search for and perform available actions. More information.

TransferFcn
Numerator="[1.7e4*k1 1e8]*0.0020*k0"
Denominator="[1e-7*k2 1.2*k3 0]"
ParameterTunability="Auto"
AbsoluteTolerance="auto"
ContinuousStateAttributes="''"

Signal Statistics

	Value	Time
Max	1.303e+00	5.388e-06
Min	0.000e+00	0.000e+00
Peak to Peak	1.303e+00	
Mean	9.858e-01	
Median	1.001e+00	
RMS	9.956e-01	

Figure 2.40 MATLAB Simulink Demo PLL Program Outputs

CHAPTER III

HARDWARE DESIGN FOR INDUSTRIAL ANALYZER

3.1 Main Supply Circuit

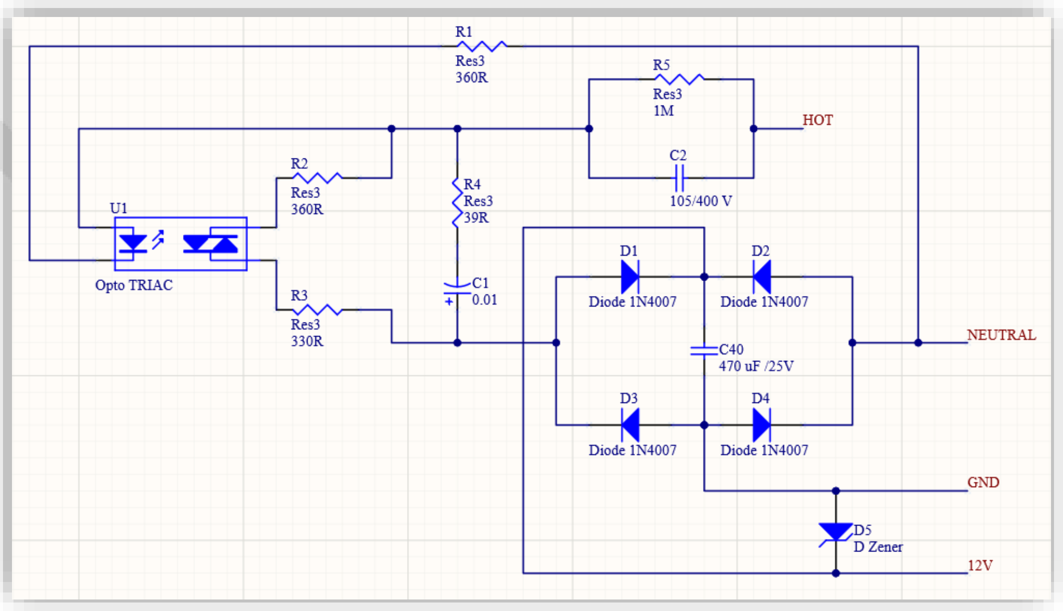


Figure 3.1 Sophisticated Triac Driver Optoisolator 230 VAC to 12 VDC Power Supply Circuit

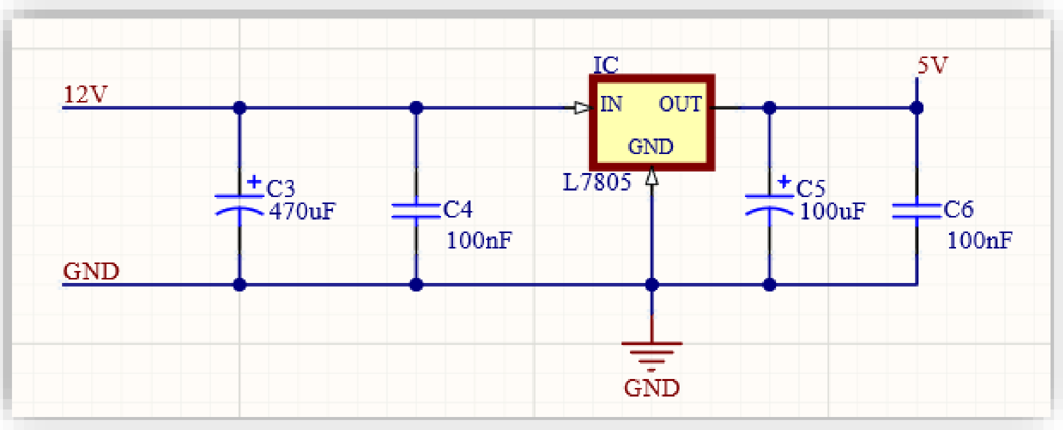


Figure 3.2 5 VDC circuit for TFT-LCD

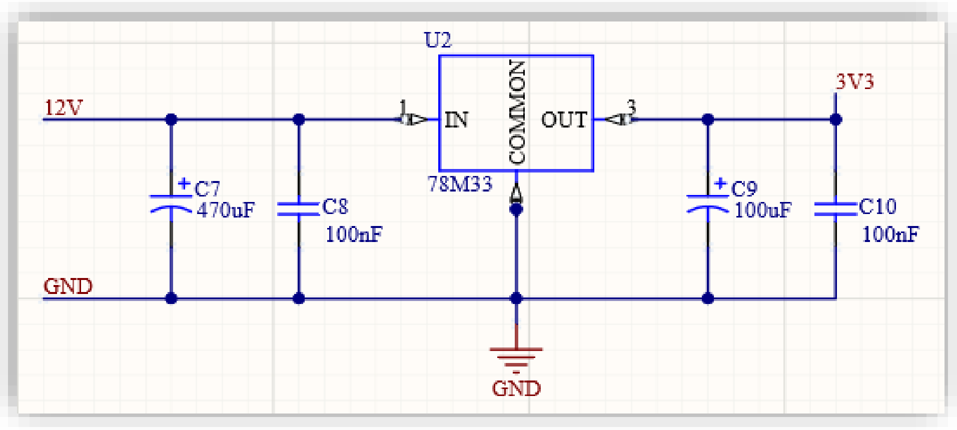


Figure 3.3 3V3 DC circuit for electronic components

The energy requirement of the equipment to be designed is reduced gradually to 12V DC voltage with the converter circuit in the system with 230V AC voltage. All digital elements on the circuit (Microcontroller, OPAMP, Communication Peripherals etc.) except the TFT LCD screen, operate with 3.3V DC voltage. Only TFT LCD screen operates with 5V DC voltage. 12V DC voltage taken from the converter circuit is converted into 5V DC and 3.3V DC voltage by linear voltage regulators step by step. LM7805 is used for 5V conversion and 78M33 linear regulator is used for 3.3V conversion. In addition, diodes have been added to the input sections for linear current control.

3.2 Voltage Detection Circuit

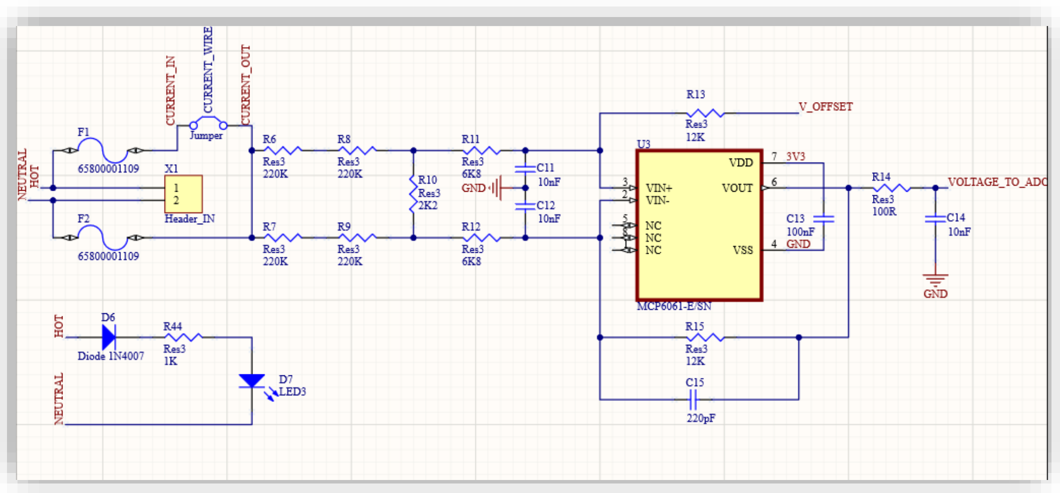


Figure 3.4 Voltage detection circuit

In order to use any current transformer in the input circuit, a connection has been made using jumper. If desired, an external current transformer can be connected. In the voltage measurement circuit, the network voltage level was decreased with voltage dividers, then the voltage was increased with the differential input opamp circuit and the analog digital converter part of the microcontroller was entered. For the voltage divider part, the voltage falling on the R10 resistor can be calculated by Equation (3.1).

$$v_{R10} = v_{in} \cdot \frac{R_{10}}{R_6+R_7+R_8+R_9} \tag{3. 1}$$

The gain of the differential input opamp circuit can be calculated by Equation (3.2) if the resistances of R8 - R9 and R10 - R11 are equal.

$$Gain = \frac{R13}{R11} = \frac{R13}{R15} = \frac{12k}{6{,}8k} = 1{,}7647 \tag{3. 2}$$

The time constant of the RC circuit on the output side of the opamp circuit can be calculated with Equation (3.3).

$$\tau = R \cdot C = 100 \cdot 10 \cdot 10^{-9} = 1\mu s \tag{3. 3}$$

A single supply, high precision opamp coded as MCP6061 from Microchip company was used as amplifier. The opamp output is shifted to half (1.65V) of the supply voltage, since both the positive and negative alternans of the grid voltage must be measured. In this way, the voltage sampled with an analog digital converter is converted into a sinusoidal DC voltage oscillating above 1.65V. In this circuit, the supply voltage is divided into two with two equivalent resistors and a voltage is applied to the voltages to be shifted by a follower opamp circuit.

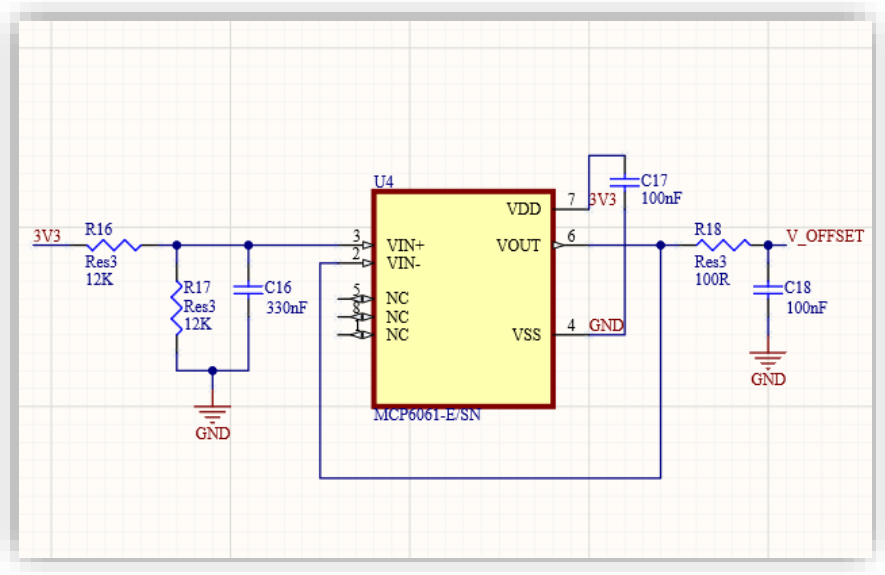


Figure 3.5 Voltage offset circuit

3.3 Current Detection Circuit

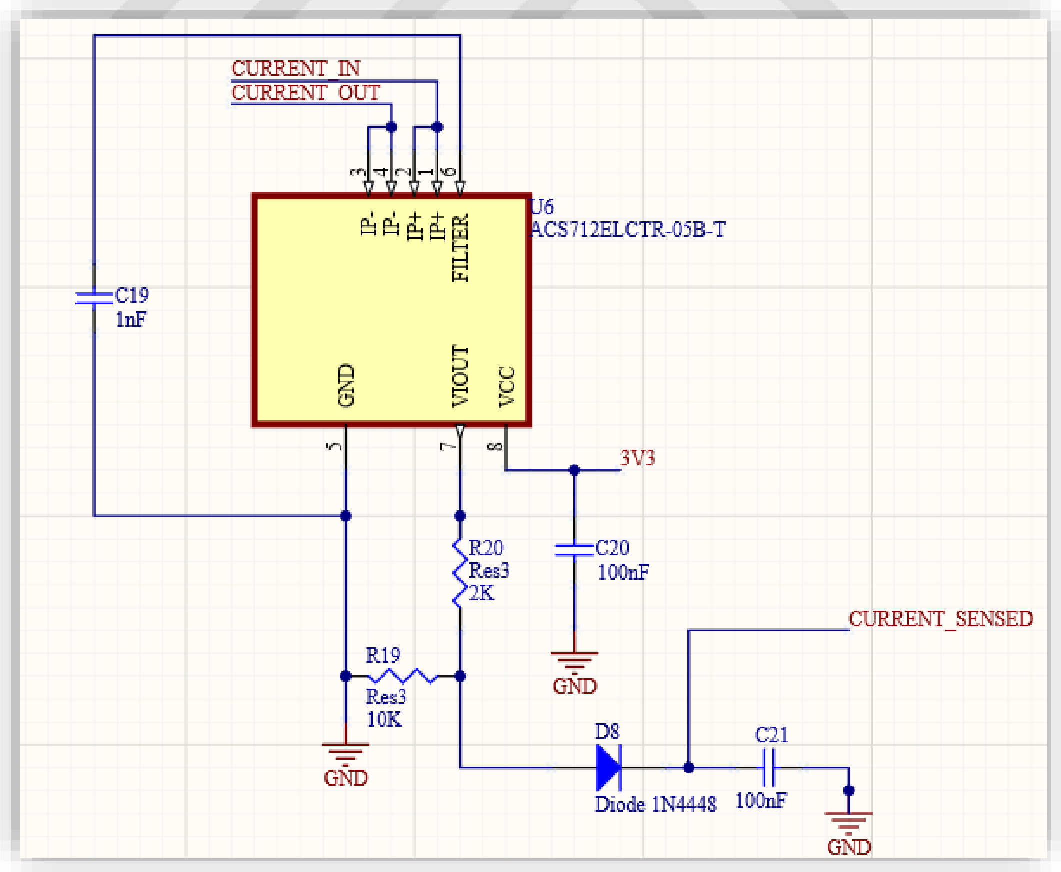


Figure 3.6 Current detection circuit with hall effect sensor

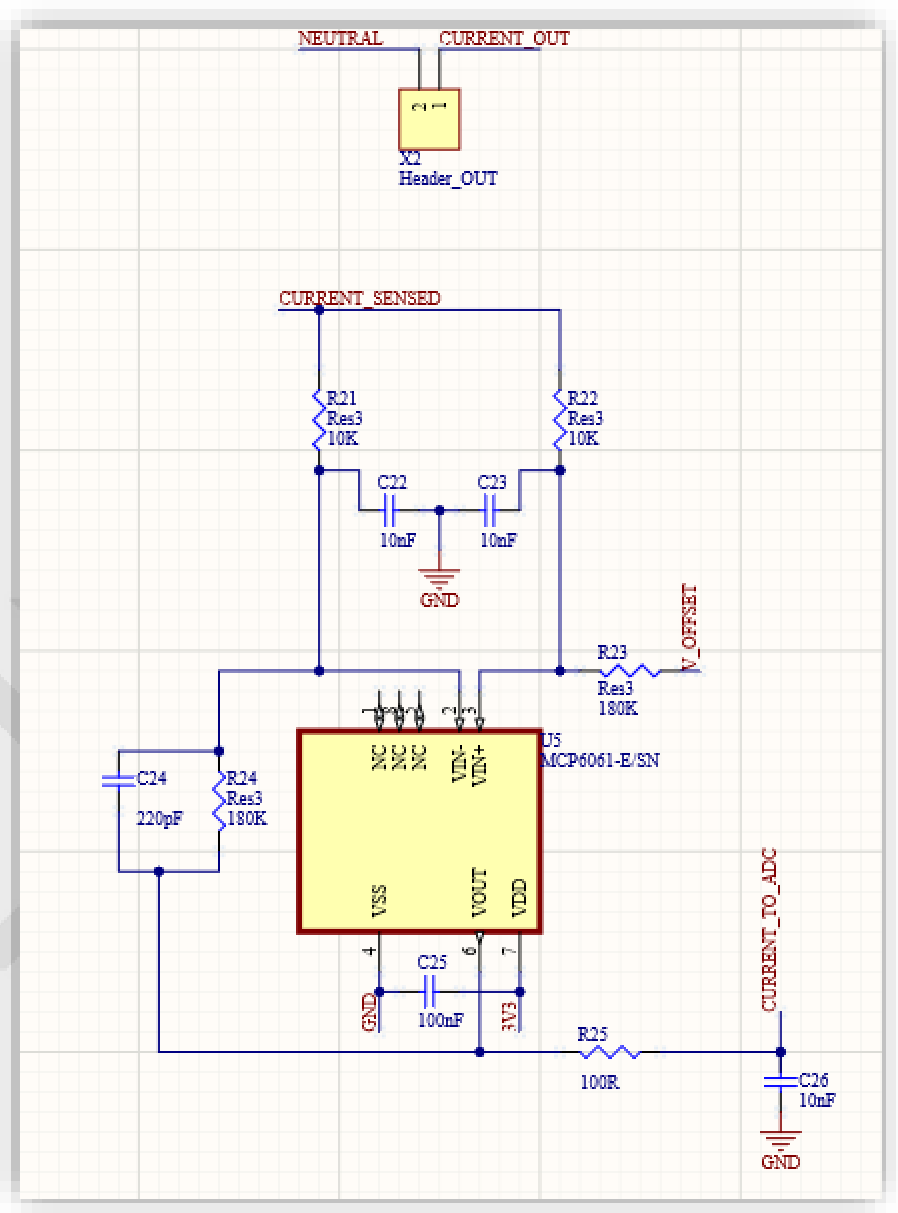
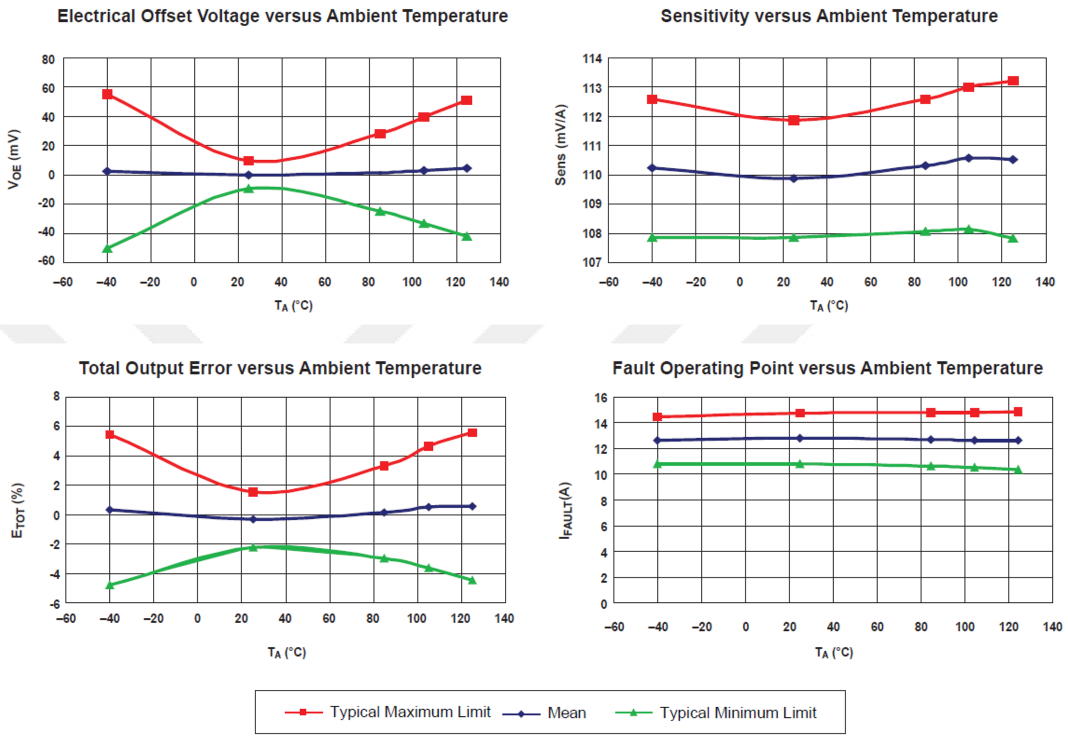


Figure 3.7 Current output circuit

Current measurement can be made on the hardware in two ways. First, it is the current measurement input where the field effect current sensor connected to the current input is used, while the measurement can be made over the jumper by adding an external current transformer if desired. Field effect sensor was used for measurement experiments. In addition, with the differential opamp in the current input circuit, since there is no need for boosting and shifting, the sensor output was directly connected to the opamp output and measurements were made.

As the field effect current sensor, a model of Allegro company, coded ACS711KLCA, capable of measuring in the range of ± 12.5A with a single supply of 3.3V and operating in the range of -40 / + 125 ° C was used. The output of the sensor is taken directly offset to half of the supply voltage. At the sensor output, 110mV voltage is produced per 1A.



Electrical Offset Voltage versus Ambient Temperature

Sensitivity versus Ambient Temperature

Total Output Error versus Ambient Temperature

Fault Operating Point versus Ambient Temperature

Figure 3.8 Characteristic Performance data of ACS711KLCA-25A, Vcc=3,3 V

3.4 Voltage and Current Zero-cross Detection Circuit

A comparator circuit has been added on the hardware to capture zero-cross from voltage and current output signals. LM393 integrated circuit is used as comparator. The corresponding signals were entered into the positive inputs of the comparators, and the voltage obtained from the shifting circuit was entered into the negative inputs. When the voltage level is higher than the shifting voltage the output is positive and when it is low the output is zero. Thus, the frequency of the input signal is detected by the square wave obtained from the output by connecting to the external interrupt inputs of the microcontroller. In the software, only the zero-crossing signal of the voltage was captured by external interrupts and frequency measurement was performed.

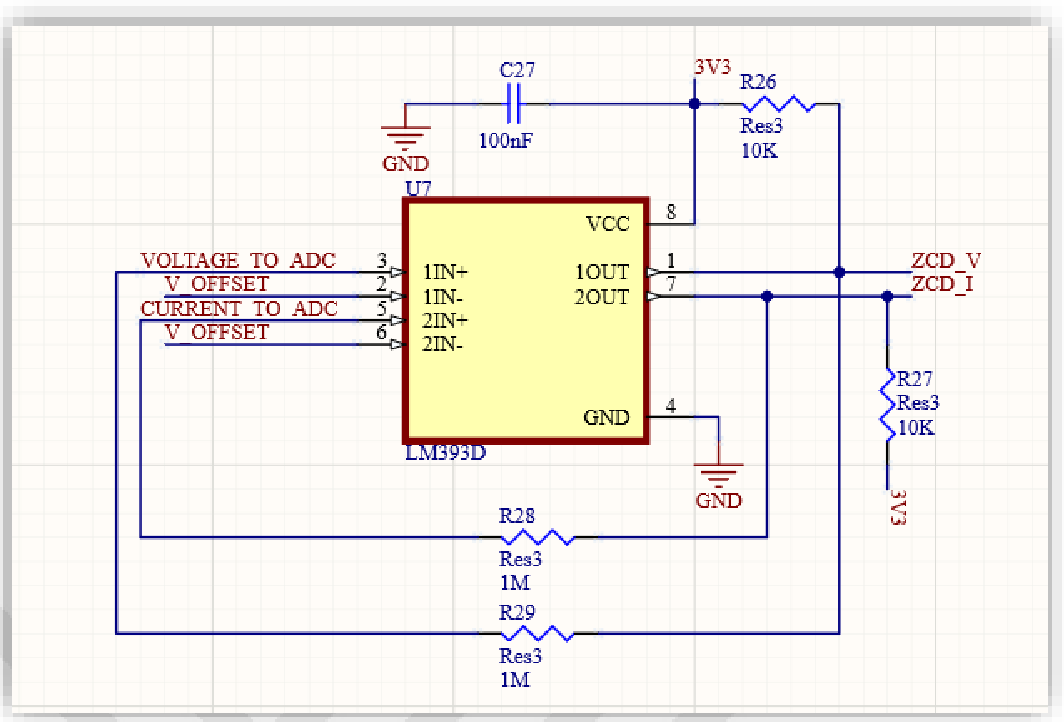


Figure 3.9 Zero-cross detection circuit

3.5 Battery Back-up Circuit

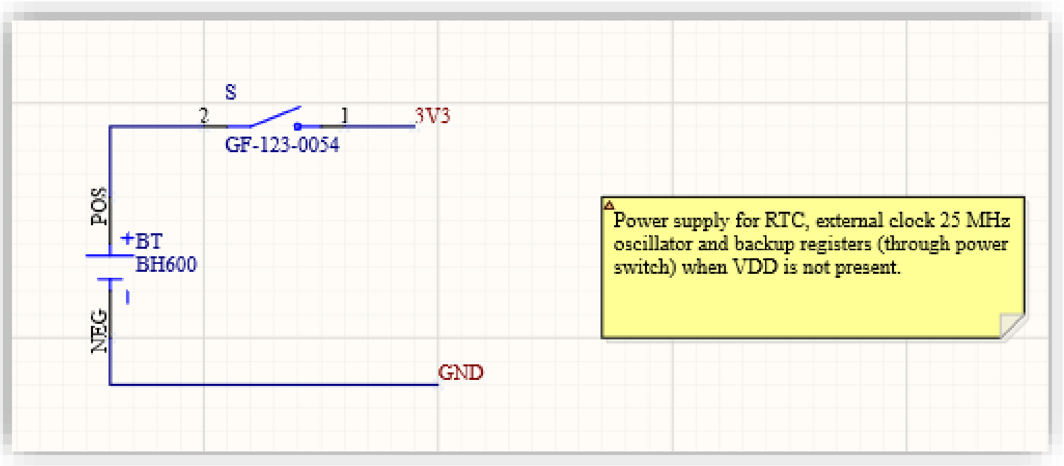

S
2
1
3V3
GF-123-0054
POS
+BT
BH600
NEG
GND
Power supply for RTC, external clock 25 MHz oscillator and backup registers (through power switch) when VDD is not present.

Figure 3.10 Battery backup circuit

A backup battery system has been added to meet the energy needs of the processor in any case.

3.6 Microcontroller Unit (MCU)

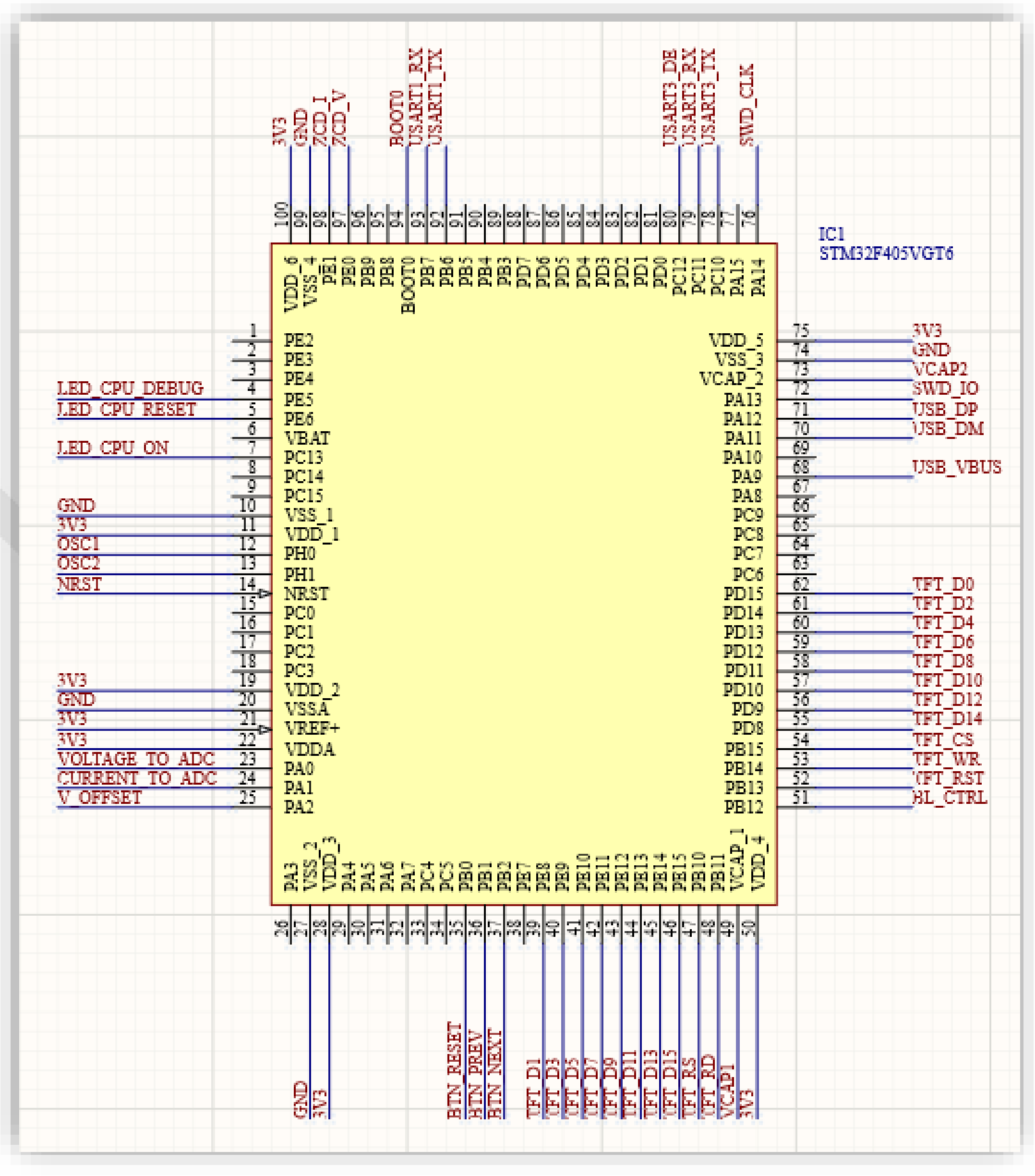


Figure 3.11 Central processing unit (CPU) input and outputs

As the microcontroller on the hardware, STM32F405VG with a 32bit core, decimal processing unit (FPU) and a speed of 168MHz, which is produced by ST Microelectronics, is ARM Cortex M4 based. Generally, its features are as follows. [16]

- Core: Arm® 32-bit Cortex®-M4 CPU with FPU [16]
- ART Accelerator allowing 0-wait state execution from Flash memory [16]
- Frequency up to 168 MHz [16]
- Memory protection unit [16]

- 210 DMIPS/ 1.25 DMIPS/MHz (Dhrystone 2.1) [16]
- DSP instructions [16]
- 1 Mbyte of Flash memory [16]
- LCD parallel interface, 8080/6800 modes [16]
- Low-power operation [16]
- 3×12-bit, 2.4 MSPS A/D converters: up to 24 channels and 7.2 MSPS in triple interleaved mode [16]
- 2×12-bit D/A converters [16]
- General-purpose DMA: 16-stream DMA controller with FIFOs and burst support [16]
- Up to 17 timers: up to twelve 16-bit and two 32- bit timers up to 168 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input [16]
- Debug mode
  - Serial wire debug (SWD) & JTAG interfaces [16]
  - Cortex-M4 Embedded Trace Macrocell™ [16]
- I/O ports with interrupt capability
- Up to 15 communication interfaces
  - Up to 3 × I2C interfaces (SMBus/PMBus) [16]
  - Up to 4 USARTs/2 UARTs (10.5 Mbit/s, ISO 7816 interface, LIN, IrDA, modem control) [16]
  - Up to 3 SPIs (42 Mbits/s), 2 with muxed full-duplex I2S to achieve audio class accuracy via internal audio PLL or external clock [16]
  - 2 × CAN interfaces (2.0B Active) – SDIO interface [16]
- Advanced connectivity
  - USB 2.0 full-speed device/host/OTG controller with on-chip PHY [16]
  - USB 2.0 high-speed/full-speed device/host/OTG controller with dedicated DMA, on-chip full-speed PHY and ULPI [16]
  - 10/100 Ethernet MAC with dedicated DMA: supports IEEE 1588v2 hardware, MII/RMII [16]

General features of the MCU are shown below in figure (3.13).

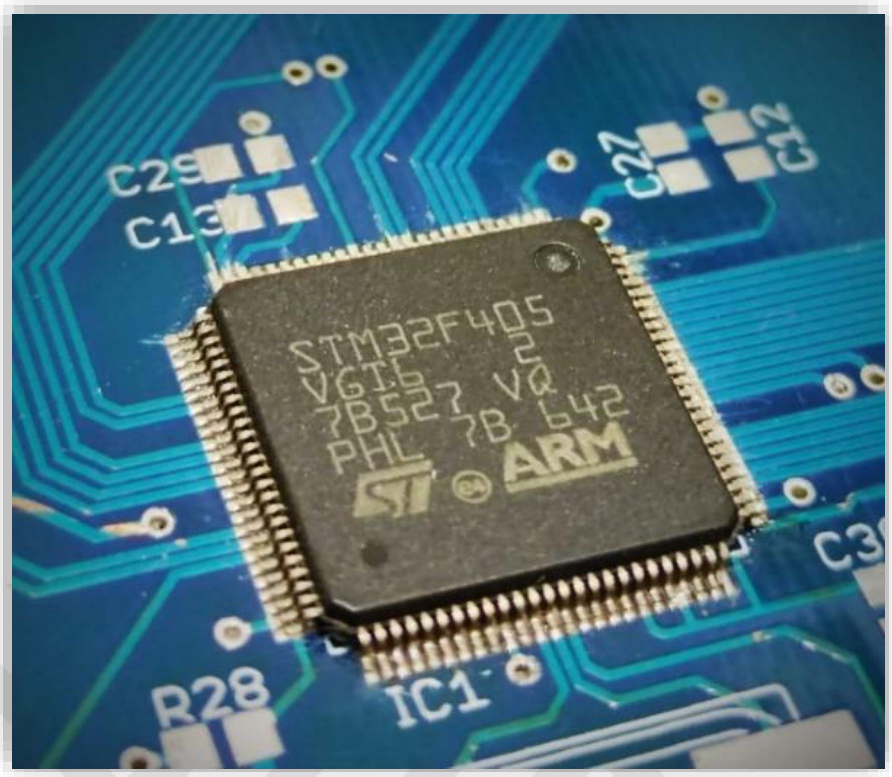


Figure 3.12 STM32F405VGT6 microchip designed by STM.

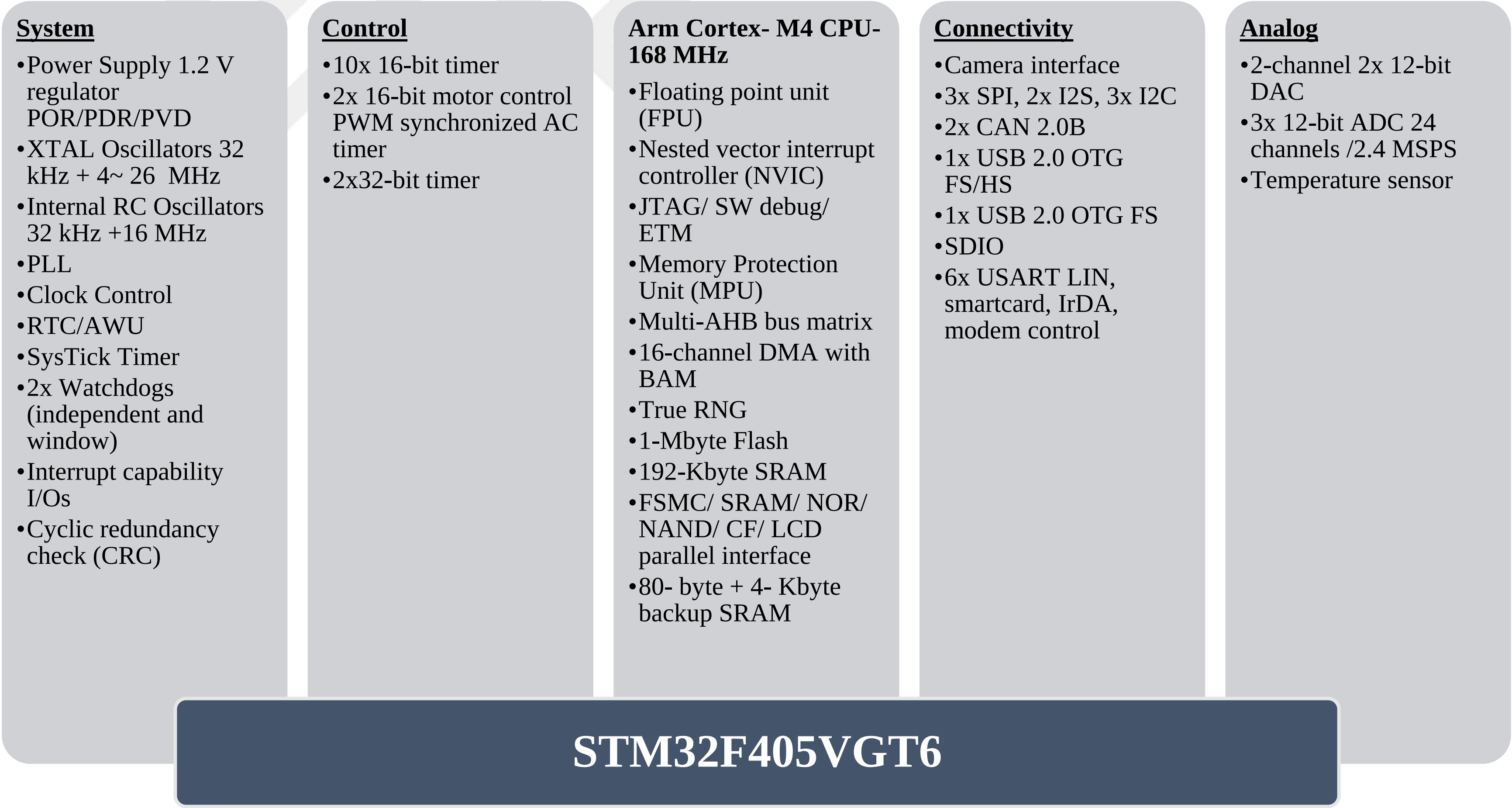

System
•Power Supply 1.2 V regulator POR/PDR/PVD
•XTAL Oscillators 32 kHz + 4~ 26 MHz
•Internal RC Oscillators 32 kHz +16 MHz
•PLL
•Clock Control
•RTC/AWU
•SysTick Timer
•2x Watchdogs (independent and window)
•Interrupt capability I/Os
•Cyclic redundancy check (CRC)
Control
•10x 16-bit timer
•2x 16-bit motor control PWM synchronized AC timer
•2x32-bit timer
Arm Cortex- M4 CPU- 168 MHz
•Floating point unit (FPU)
•Nested vector interrupt controller (NVIC)
•JTAG/ SW debug/ ETM
•Memory Protection Unit (MPU)
•Multi-AHB bus matrix
•16-channel DMA with BAM
•True RNG
•1-Mbyte Flash
•192-Kbyte SRAM
•FSMC/ SRAM/ NOR/ NAND/ CF/ LCD parallel interface
•80- byte + 4- Kbyte backup SRAM
Connectivity
•Camera interface
•3x SPI, 2x I2S, 3x I2C
•2x CAN 2.0B
•1x USB 2.0 OTG FS/HS
•1x USB 2.0 OTG FS
•SDIO
•6x USART LIN, smartcard, IrDA, modem control
Analog
•2-channel 2x 12-bit DAC
•3x 12-bit ADC 24 channels /2.4 MSPS
•Temperature sensor
STM32F405VGT6

Figure 3.13 General features of the STM32F405VGT6 chip by STM.

The ST-LINK/V2 is an in-circuit debugger and programmer for the STM8 and STM32 microcontrollers. The single-wire interface module (SWIM) and JTAG/serial wire debugging (SWD) interfaces are used to communicate with any STM8 or STM32 microcontroller located on an application board. In addition to providing the same functionalities as the ST-LINK/V2, the ST-LINK/V2-ISOL features digital isolation between the PC and the target application board. It also withstands voltages of up to 1000 v_{rms}. [36]

Figure 3.14 ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32

STM32F405VG with LQFP100 sheath structure shown in Figure 63 operates with 3.3V voltage. In addition, programming and debugging can be done via SWD or JTAG interface with ST Microelectronics' ST-LINK programming and debug tool.

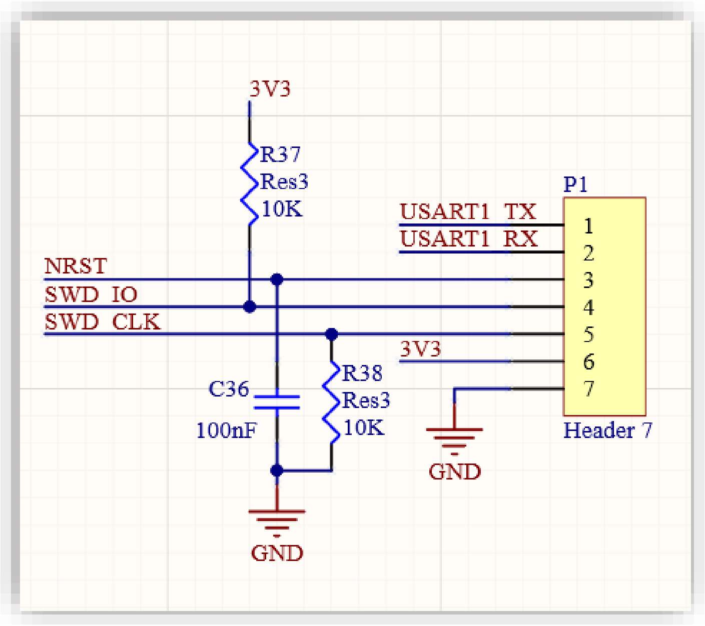


Figure 3.15 Serial communication debugger interface circuit

Data, clock and reset pins of the SWD interface are taken out through the Header 7 connector in order to be able to connect with the microcontroller via the ST-LINK programming tool. In addition, two USART connections on the same connector have been added in order to transfer and monitor various data of the microcontroller through the serial port channel during operation.

In addition to them, the clock signals required for the operation of the microcontroller have been provided with an external crystal to have higher accuracy despite the presence of a built-in RC oscillator. In figure (3.16) an external crystal frequency is chosen as 25MHz and this frequency has been increased up to 144MHz in the microcontroller. Although the maximum operating frequency is 168 MHz, the operating frequency is limited to 144 MHz in order for the TFT LCD display to work properly.

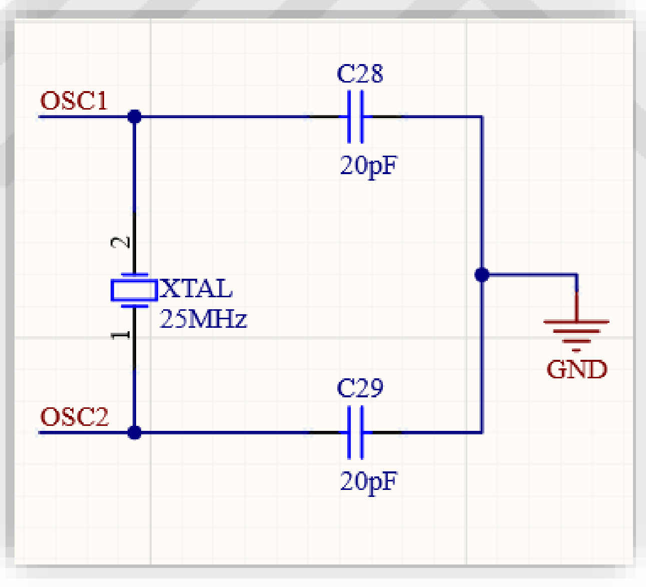


Figure 3.16 External Oscillator circuit

3.7 Human Hardware Interface Design

A TFT LCD has been added to show the measurement data on the hardware, three LEDs to indicate the operating conditions and three buttons to allow the user to switch between the operating screen pages and reset.

As a TFT LCD screen, an ILI9328 internal driver module of ILITEK company, with a screen size of 3.2” and a resolution of 320x240 pixels, was preferred. The presence of internal SRAM on the driver allows the microcontroller to be driven with the FSMC memory controller.

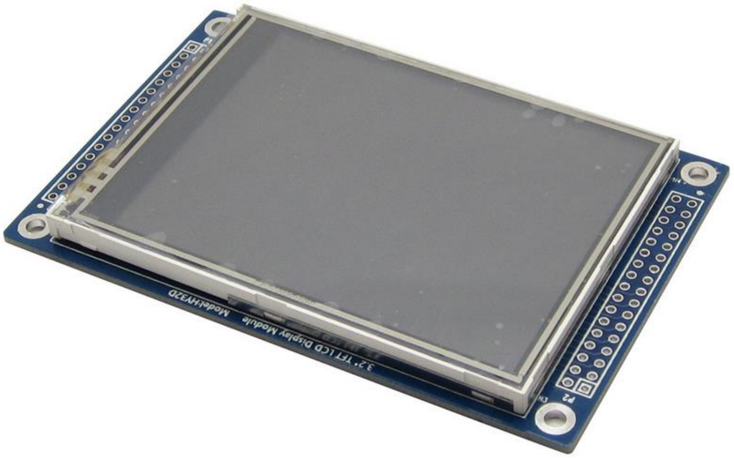


Figure 3.17 Human Hardware Interface TFT-LCD Front view

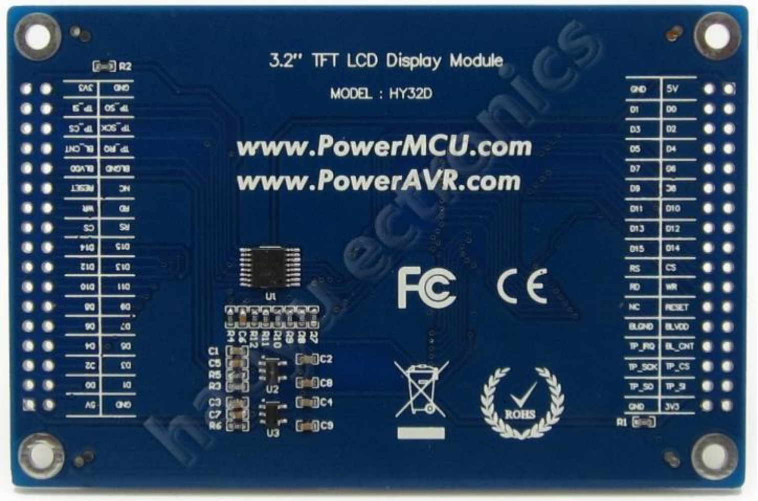


Figure 3.18 Human Hardware Interface TFT-LCD Rear view

The wiring diagram of the TFT LCD module used with the microcontroller is as in Figure (3.13). The TFT LCD screen has been driven in 16-bit RGB565 format. Read, write, select or reset signals are automatically generated by the FSMC memory control unit in the microcontroller. The brightness of the backlight, which is fed with 5V DC voltage, can be adjusted by PWM signals applied to the control pin. The schematic of HY32D shown in ***Appendix C***.

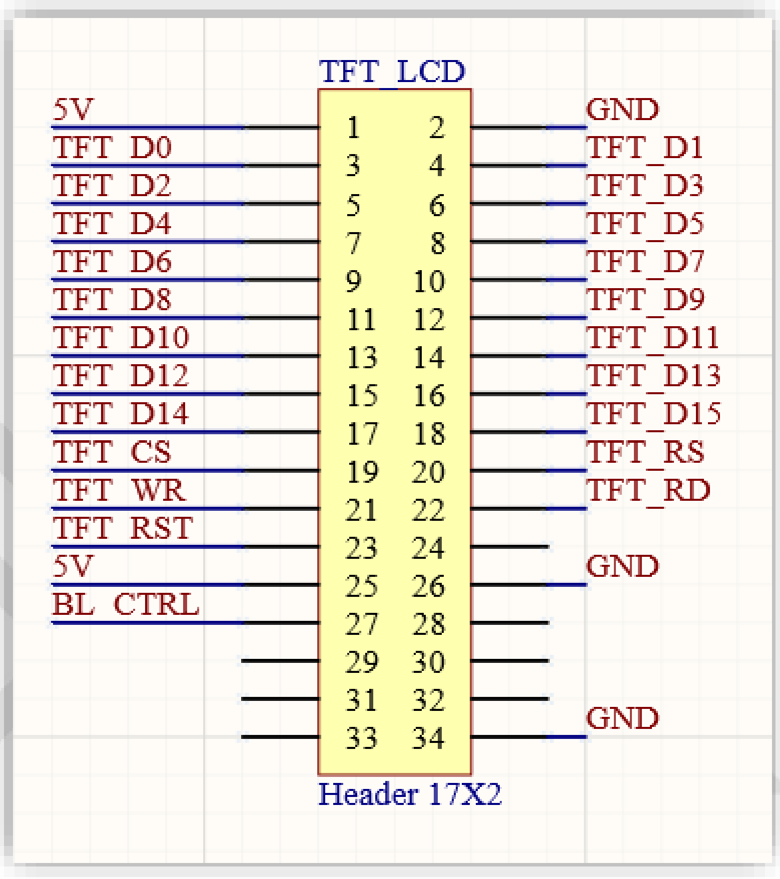


Figure 3.19 Header wiring diagram for connection between TFT-LCD and MCU

The connection of the LED diodes used to show the operating states is as follows. In Figure (3.20), LED1 shows the power status of the device. It turns on as soon as energy comes to the CPU input. LED2 lights when the central processing unit (CPU) is processing a message in the main task function. Depending on the burning state of this LED, the intensity of the microcontroller's working state can be observed. LED3 shows the communication status. It actively turns on when there is a query on the device via Modbus.

In Figure (3.21), The first two of the buttons are effectively used to switch between working screens. The first button is used to go to the previous pages, and the second button to switch to the next working screen. The last button is designed for the reset CPU. If it needs to be reset this can be used for that purpose.

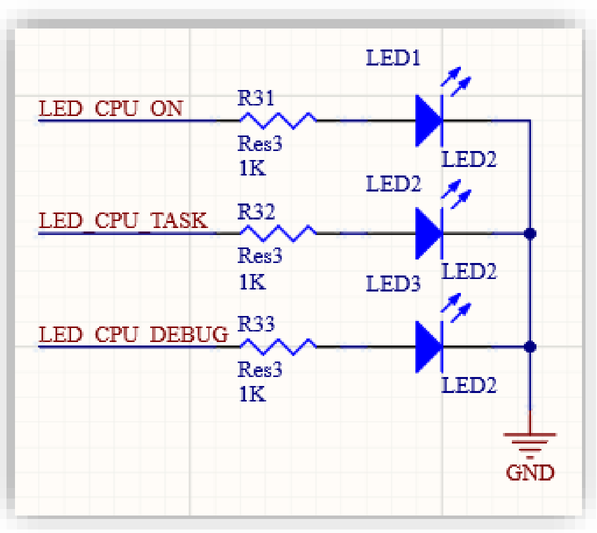
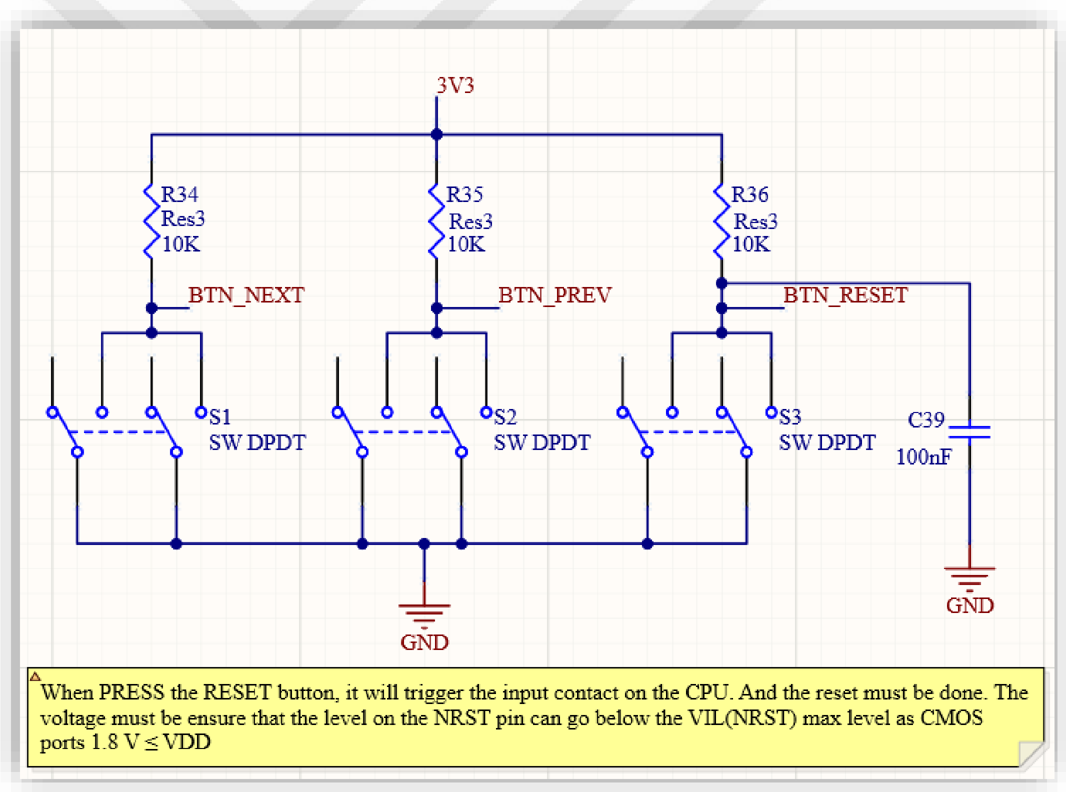


Figure 3.20 LED Indicators



When PRESS the RESET button, it will trigger the input contact on the CPU. And the reset must be done. The voltage must be ensure that the level on the NRST pin can go below the VIL(NRST) max level as CMOS ports 1.8 V ≤ VDD

Figure 3.21 Human PCB Interface buttons

3.8 Communication Hardware

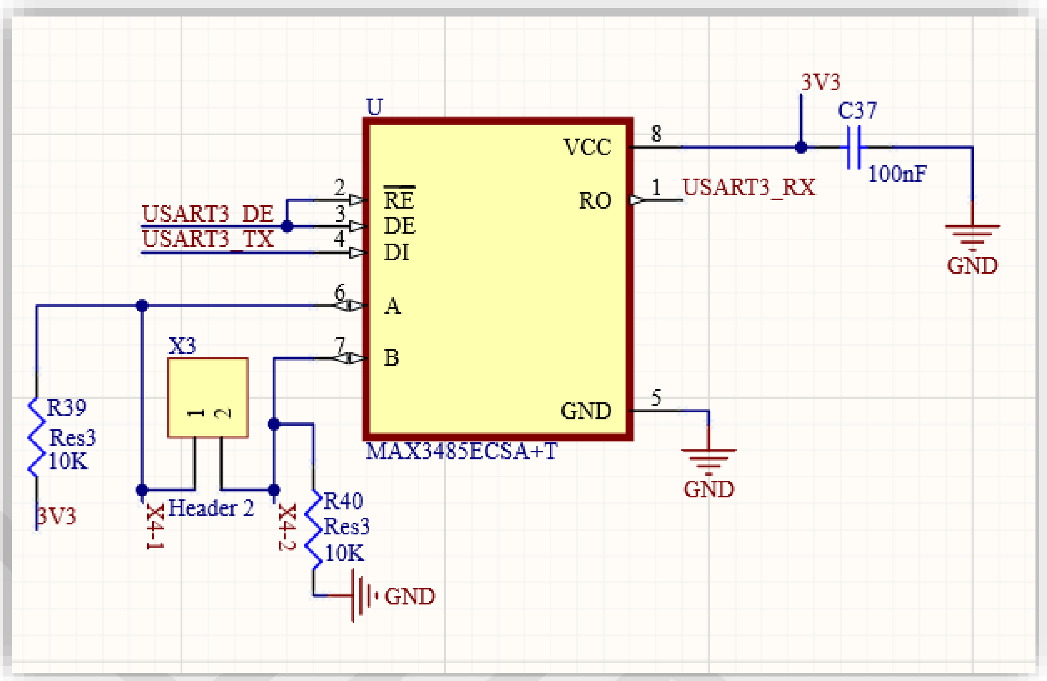


Figure 3.22: RS-485 Interface circuit with the MCU

Both RS485 connection Figure (3.22) and Bluetooth connection Figure (3.23) have been added on the hardware to enable the microcontroller to communicate. The USART3 unit of the microcontroller is used for both communications. If desired, serial communication can be provided via RS485 or Bluetooth. Wireless data transfer was realized over the virtual serial port that will be provided with the connection made over Bluetooth.

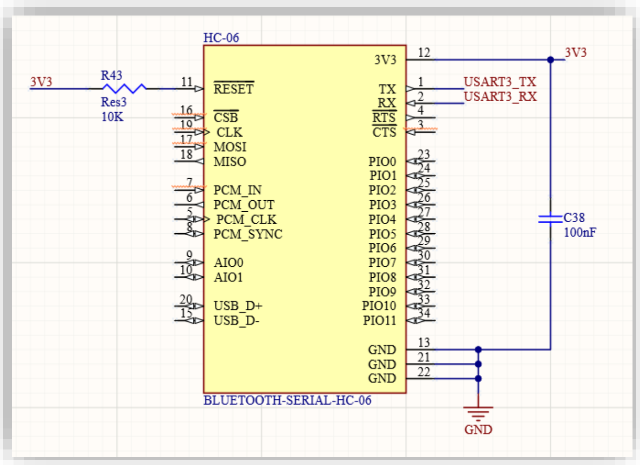


Figure 3.23 Bluetooth Serial Interface with the MCU

The Bluetooth module used is a satellite mode Bluetooth module with HC-06 code. It provides a wireless data transfer with USART with the serial port protocol (SPP) it hosts. Since it can operate in satellite mode, the party that is the main unit must initiate the connection. After the connection is made positive, data transfer will be possible over the virtual serial ports that will be formed.

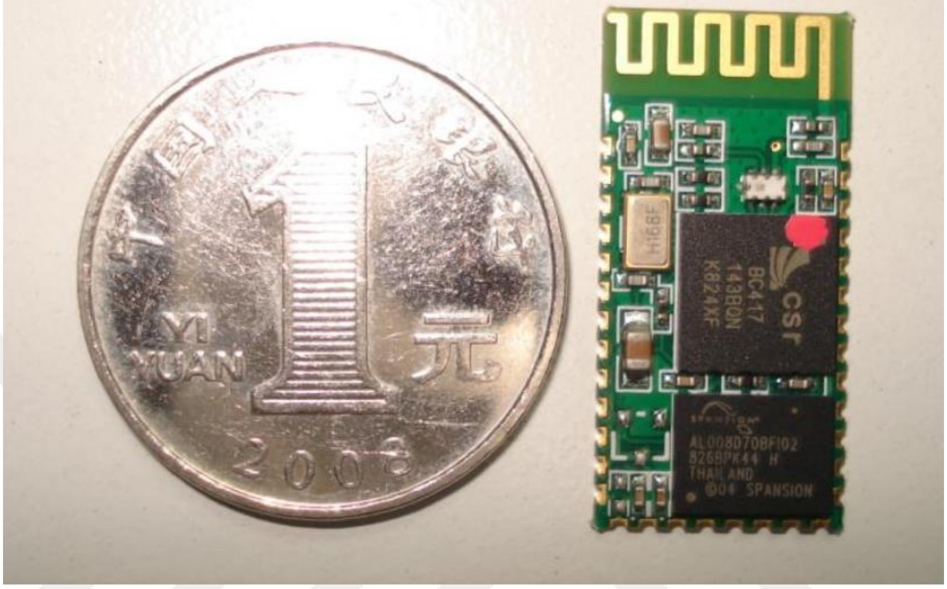


Figure 3.24 Preferred Bluetooth Module (HC-06)

In addition to RS485 and Bluetooth, a USB 2.0 mini-USB connection port has been added on the hardware. If desired, communication with the microcontroller can be provided through this port.

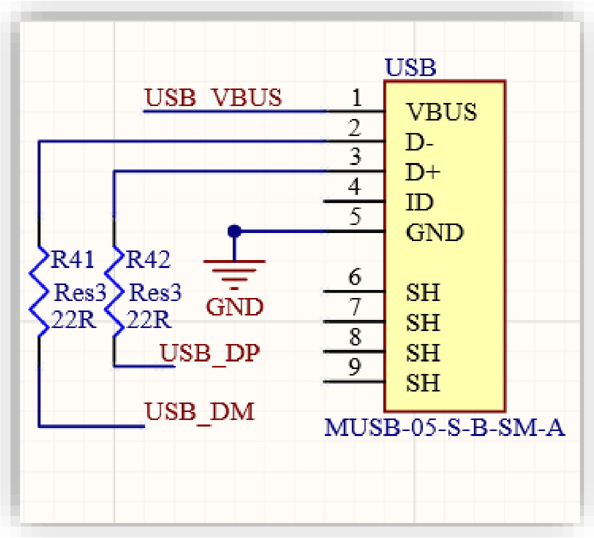


Figure 3.25 USB Full-speed Interface Circuit

3.9 Designed PCB

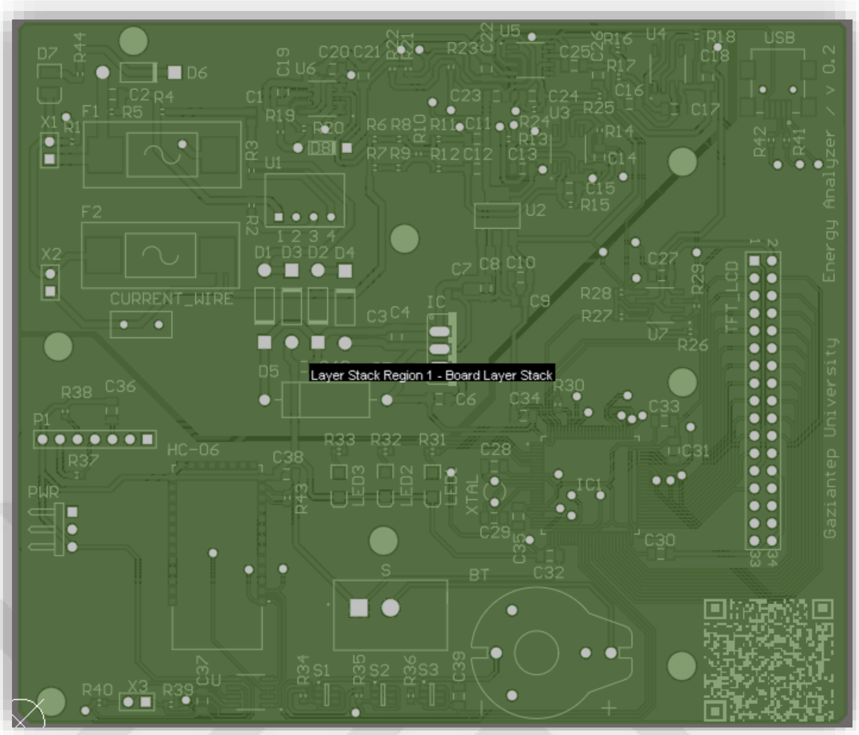

Layer Stack Region 1 - Board Layer Stack
Gaziantep University Energy Analyzer / v 0.2
USB
CURRENT_WIRE
HC-06
TFT_LCD
PWR

Figure 3.26 Designed PCB, Board Layer Stack Front view v.0.2

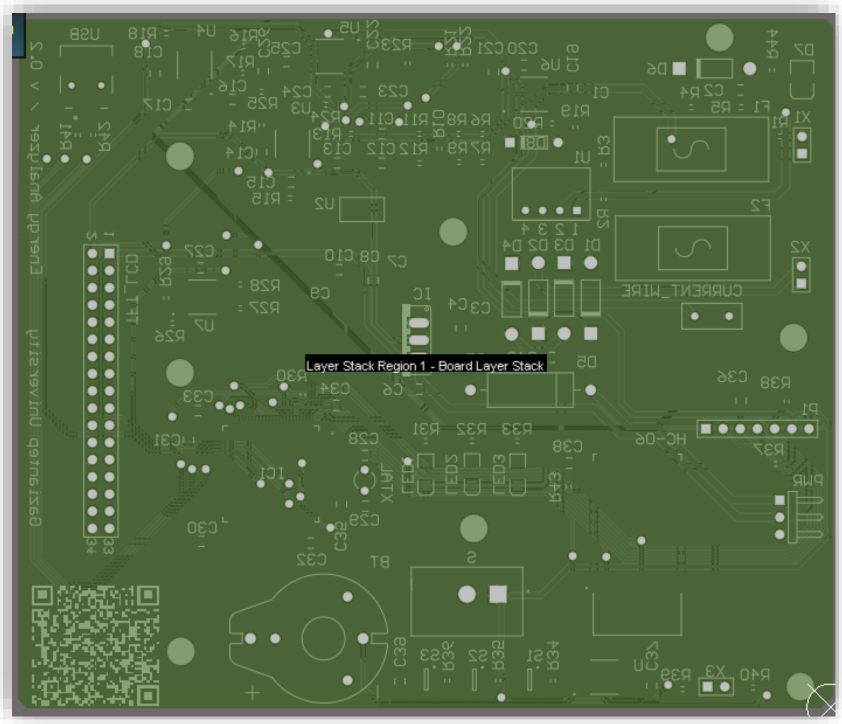


Figure 3.27 Designed PCB, Board Layer Stack Rear view v.0.2

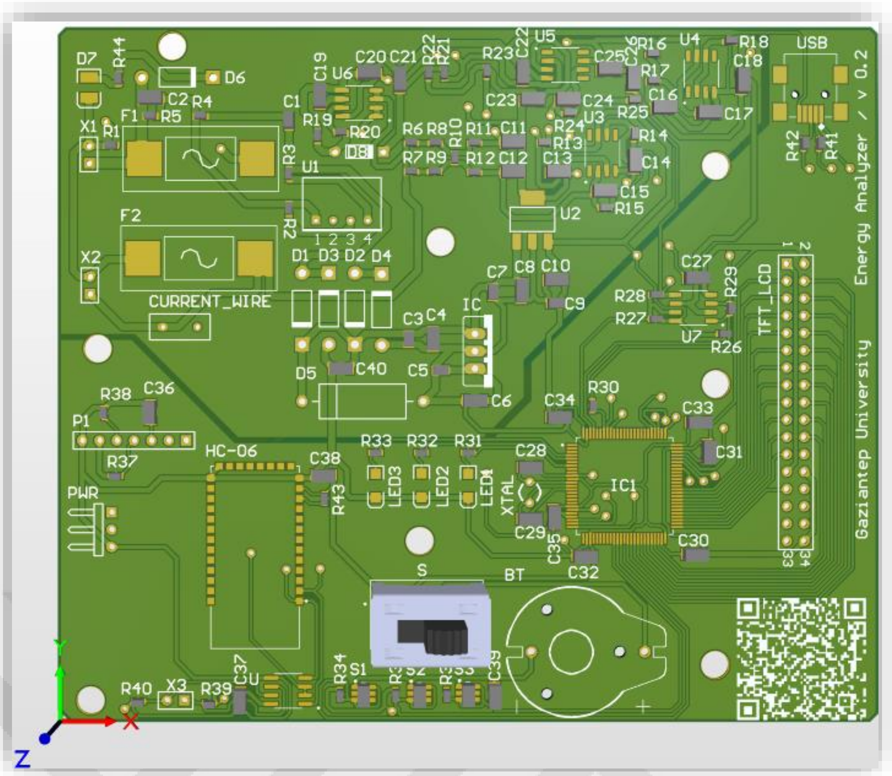


Figure 3.28 Designed PCB, 3D Front view v.0.2

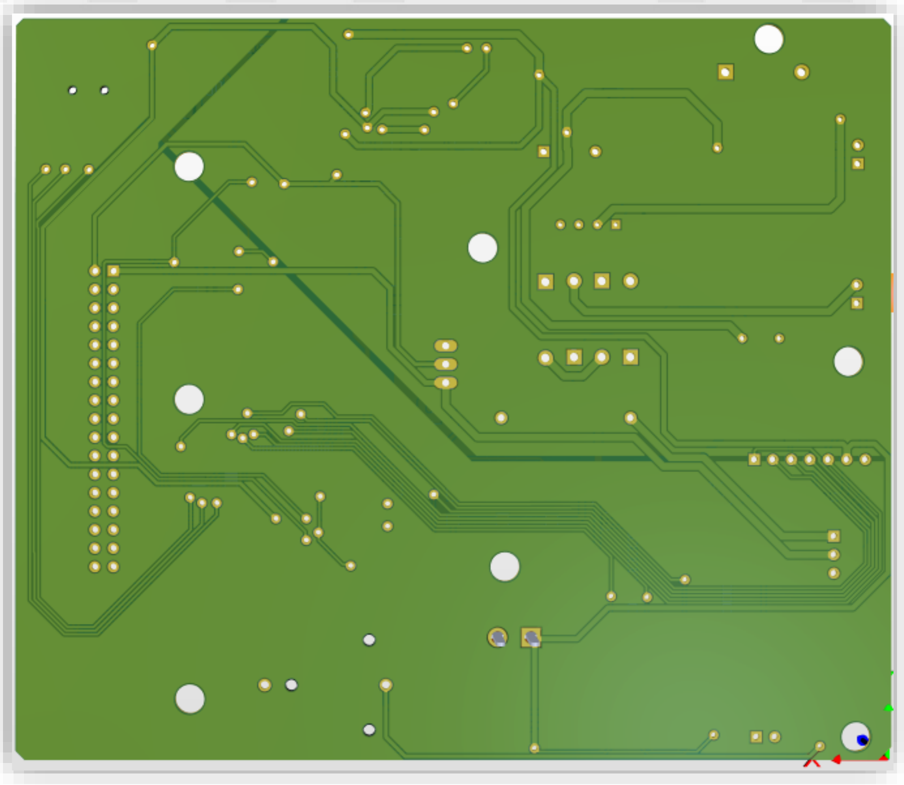


Figure 3.29 Designed PCB, 3D Front view v.0.2

CHAPTER IV

SOFTWARE DESIGN FOR INDUSTRIAL ANALYZER

4.1 Message Queue Based Operating System

As we will remember from the queue data structures, it is a linear data storage structure. We can make this subject concrete with a simple example. We wanted to withdraw money from a cash machine and when we went to the front of the cash dispenser, we saw that there was a long line in front of the cash dispenser. How does this queue or queue work? The tail has a head so there is a person pulling money from the cash machine at that moment, and he's probably the first to enter the queue. Those who come after him are lined up behind him, and after the head of the queue has done his job at the cash machine, he leaves the queue, and the person behind him goes to the queue. This process is repeated continuously until there is no one left in the queue. Queuing usage in RTOS is similar. In this example, people in the queue are data or messages in RTOS. They do certain operations by reading these data in order of tasks and tasks can be blocked when the queue is empty. The beauty of using the tail also starts here. When a data comes to the queue, the desired task can be activated.

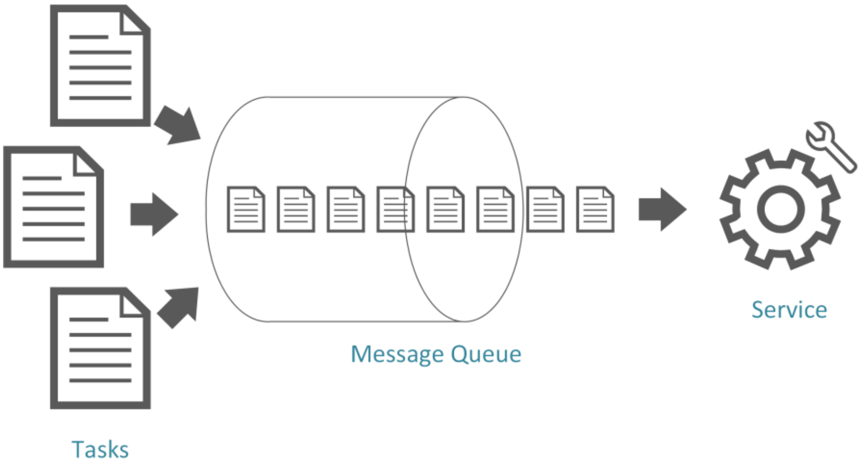


Figure 4.1 Message Queue Structure

A queue can hold fixed size variables (8 bits, 16 bits, etc.). The maximum number of these variables in the queue is called "length". These two parameters are defined when the queue is first created. Queues often use a FIFO (First in First Out) buffer. In other words, the first data written to the queue is read first. When writing data to the queue, it is usually appended at the end, when data is read from the beginning of the queue.

The queue is an object in their own right. It can be accessed from any task or ISR (Interrupt Service Routine). More than one task can write data to the queue and read data from the same queue. Typically, it is more common to write data to multiple flood queues than to read data from multiple flood queues. Tasks can be blocked with the queue. If there is no data in the queue, the task can be blocked until the data is in the queue. This blocking can be done until the data comes to the queue or optionally for a certain period of time. In other words, if the task waits for data to come to the queue for a certain period of time, and if there is no data within this period, it can continue its operations from where it left off. If there is more than one task that reads data from the queue, these tasks can be blocked at the same time, but the highest priority task gets the data in the queue first. If the priorities of the tasks waiting for data from the queue are equal, the longest waiting task will read the data from the queue.

A task can also be blocked while writing data to the queue. If the queue is full, the task that writes data to the queue can be blocked until the data is deleted from the queue. Again, in this case, more than one task can be blocked while writing data to the queue. The highest priority task writes data to the queue when space is free. If the priorities are equal, the task that waits for the longest time to write data to the queue writes the data to the queue.

The message structure to be defined in the software contains three pieces of data. These data are variables that hold the data about which task functions the messages concern, namely the target function, which message it contains and the message it carries. In the main loop, a system has been established that receives the messages transmitted to the message queue structure and transmits them to the relevant function. This mechanism receives, processes and deletes from the queue if there are any messages accumulated in the queue. Queuing structure has been established to keep maximum 32 messages in memory. If it is insufficient, it can be enlarged, but it should not be forgotten that it means taking the system away from real time.

A message structure within this structure is as follows.

```
typedef struct
{
eTasks_tTargetTask;
uint16_tTargetTaskMessageID;
uint16_tTargetTaskMessageData;
}stMessage_t;
```

The function that receives and processes the messages in the queue is as follows.

```
for(;;)
 {
      if(stMessageQueue.Counter)
{
      KernelMessagePop();
      DoProcess();
 }
}
```

The body of the main task functions that receive and process when a message is received is as follows.

```
void TaskMain(stMessage_t *stMessage)
{
      switch(stMessage->TargetTaskMessageID)
  {
      case msgX:
      break;
 case msgY:
  break;
 case msgZ:
  break;
 case msgT:
  break;
 }
}
```

Two task functions that fulfill the main operations and the parts related to the measurement are defined in the software. At any time of operation, a message can be sent to the desired task function. For example, when any of the buttons on the hardware is pressed, sending a message to the main task function (Main Task) can be done as follows. (All functions related to the structure established in the software are included in ***Appendix E***.)

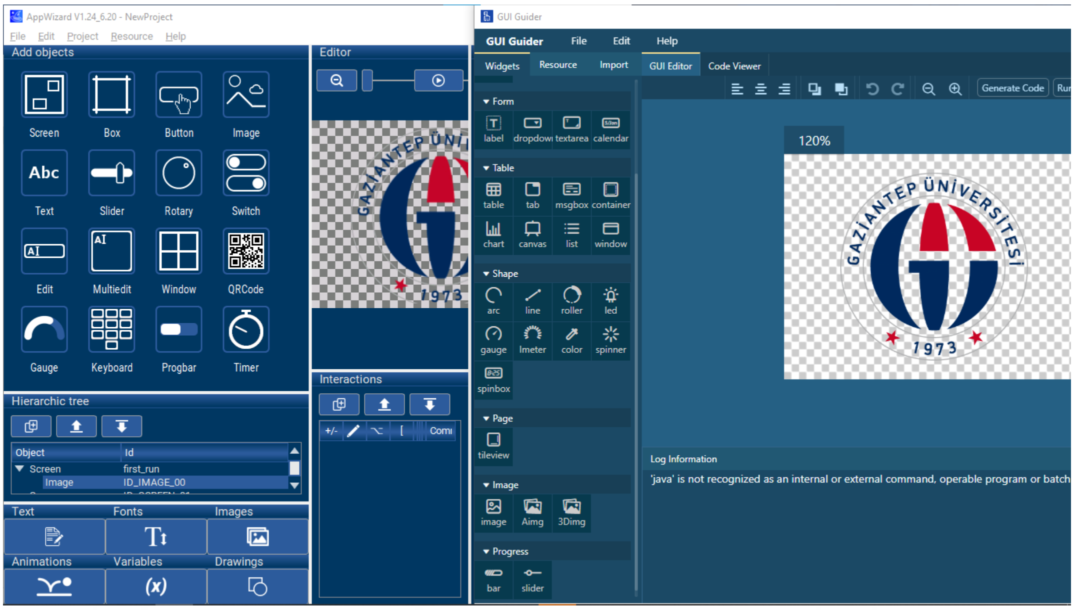
```
KernelMessage_Push(eTaskMain, msgButtonPush, stButton->ButtonNo);
```

Such structures have advantages over real-time operating system (RTOS) structures. There is a constant transition between task functions in RTOS systems. During this transition, the variables belonging to each task function must be stored in its own memory areas. After each transition, the data in these memory areas are read and operations continue accordingly. Therefore, this is an important factor that consumes processing time. In addition, RTOS systems are structures that take a significant place in the memory of the microcontroller. For this reason, they do not work well on low performance platforms.

Within the scope of this thesis, there is no continuous transition between functions in the message queue-based mini operating system. Only when a message is sent to a task function, the function is received. Therefore, there is no performance loss. It does not take up a significant amount of memory of the microcontroller. Therefore, it can be used in any system. It provides great convenience and flexibility during software development. It also improves readability in software. Therefore, the establishment of such a structure is stipulated within the scope of the thesis.

4.2 TFT-LCD Graphical Library

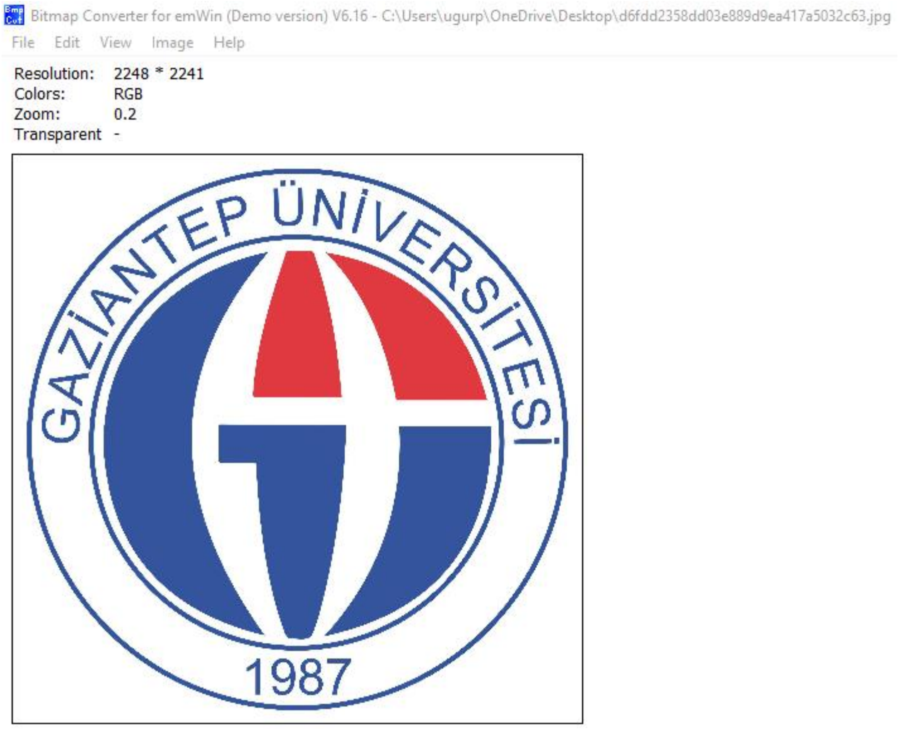
Nowadays, the use of LCD screens has increased due to the widespread use of graphic displays and the rapid increase of microcontrollers. Being aware of this situation, manufacturers also offer various graphic libraries to their users free of charge. In the software prepared within the scope of this thesis, emWin graphic library, which is offered free of charge by ST Microelectronics, was used. The emWin library is a very comprehensive tool produced by SEGGER. It can be used on many different microcontrollers. It also supports many different TFT LCD drivers, communication interfaces and picture formats. It also supports touch applications. In addition, the driverless TFT LCD displays have a structure that can be used very efficiently, whether using a real-time operating system or not. Therefore, companies such as ST and NXP offer this library free of charge. They have designer programs but general structure and graphics are similar. However, the library is not offered as open source. There are compiled library files for various platforms (Cortex M0, Cortex M3, Cortex M4, Cortex M7 etc.). It has been included in projects developed using the C / C ++ language and made available.



'java' is not recognized as an internal or external command, operable program or batch

Figure 4.2 AppWizard by SEGGER and GUI Builder by NXP

In addition to these, the emWin tool also includes a Windows program called GUI Builder to design ready-made windows and generate C / C ++ codes, a font file conversion tool to use fonts installed on Windows in projects, and converter programs that can convert image files to C / C ++ sequences. In these little tools, emWin has made the graphics library very useful and common.


Bitmap Converter for emWin (Demo version) V6.16 - C:\Users\ugurp\OneDrive\Desktop\d6fdd2358dd03e889d9ea417a5032c63.jpg
File Edit View Image Help

Resolution: 2248 * 2241
Colors: RGB
Zoom: 0.2
Transparent -

Figure 4.3 emWin Bitmap Converter tool by SEGER.

4.3 Effective Value Calculation of Voltage and Current in Discrete Time

In the first part, the Equation (4.1) regarding the effective value calculation in continuous time was obtained. In this section, numerically effective value calculation will be made by the microcontroller with the Equation (4.2).

$$V_{rms} = \sqrt{\frac{1}{T} \cdot \int_0^T v(t)^2 \cdot dt} \tag{4. 1}$$

$$V_{rms} = \sqrt{\frac{1}{N} \cdot \sum_{i=0}^{N} v_i^2} \tag{4. 2}$$

We can calculate the effective value by taking the square root of the average of the sum of the squares of the samples taken from the voltage signal, as can be understood from the expression in Equation (4.2).

The DMA is an AMBA advanced high-performance bus (AHB) module that features three AHB ports: a slave port for DMA programming and two master ports (peripheral and memory ports) that allow the DMA to initiate data transfers between different slave modules. The DMA allows data transfers to take place in the background, without the intervention of the Cortex-Mx processor. During this operation, the main processor can execute other tasks and it is only interrupted when a whole data block is available for processing. Large amounts of data can be transferred with no major impact on the system performance. The DMA is mainly used to implement central data buffer storage (usually in the system SRAM) for different peripheral modules. This solution is less expensive in terms of silicon and power consumption compared to a distributed solution where each peripheral needs to implement its own local data storage. Thanks to the Direct Memory Address (DMA) inside the microcontroller, the samples taken in order are designed to generate interrupts after they are written to the specified addresses. The analog to digital converter unit (ADC) was periodically triggered by the timer (TIMER) unit, and the sampling process was performed. Following the sample taken from the voltage channel, the current channel is sampled and then the results are transferred to the relevant variables by the DMA unit. Thus, it provides periodic sampling from voltage and current. The sampling period is set to take 128 samples per period. It is 156.25 μs for 50Hz network frequency. However, as it is explained in the second section, since this time is adjusted synchronously to the PLL output, in case of a frequency change, this time is adjusted dynamically. [17]

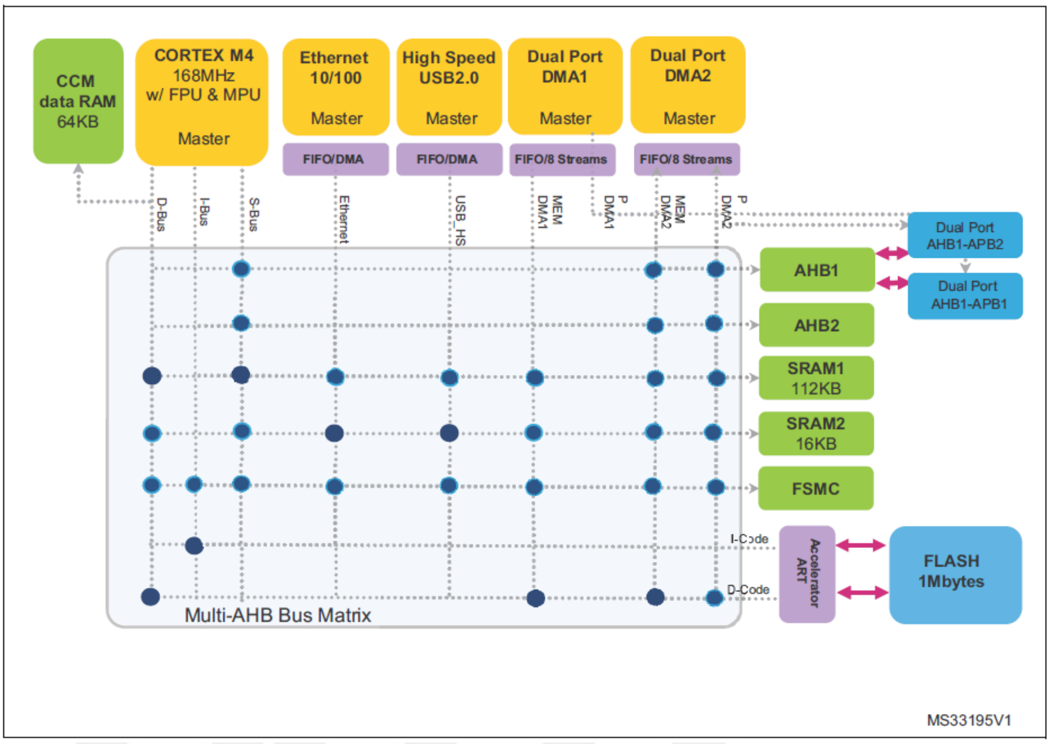


Figure 4.4 STM-32F405 DMA Structure

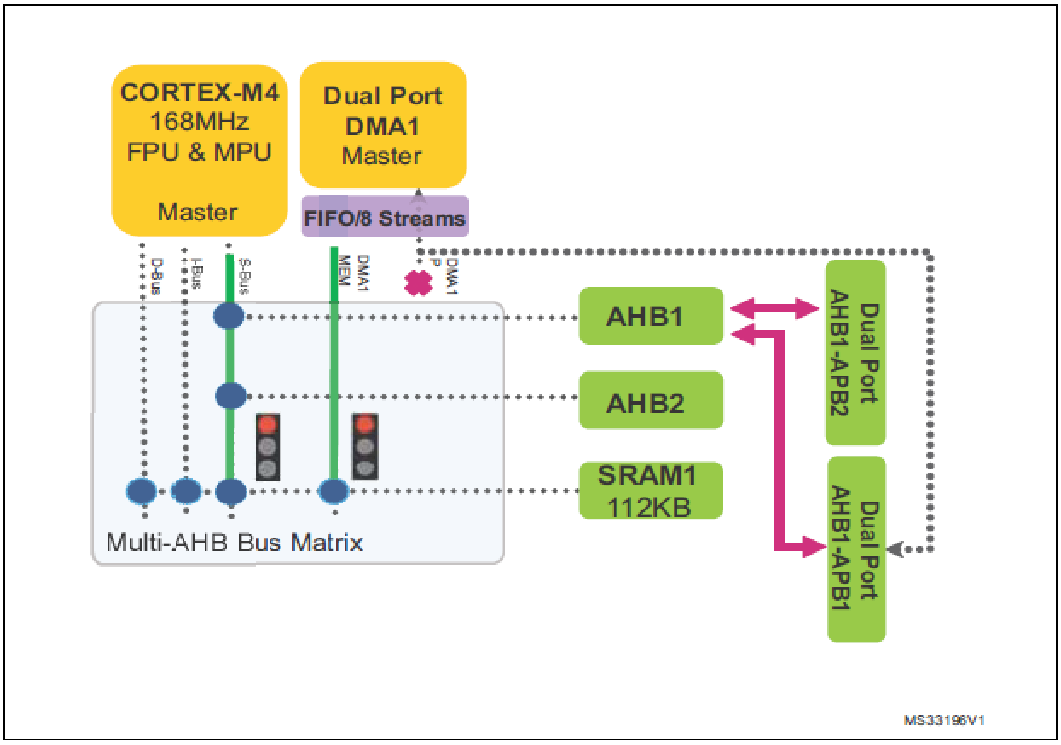


Figure 4.5 CPU and DMA1 request an access to SRAM1

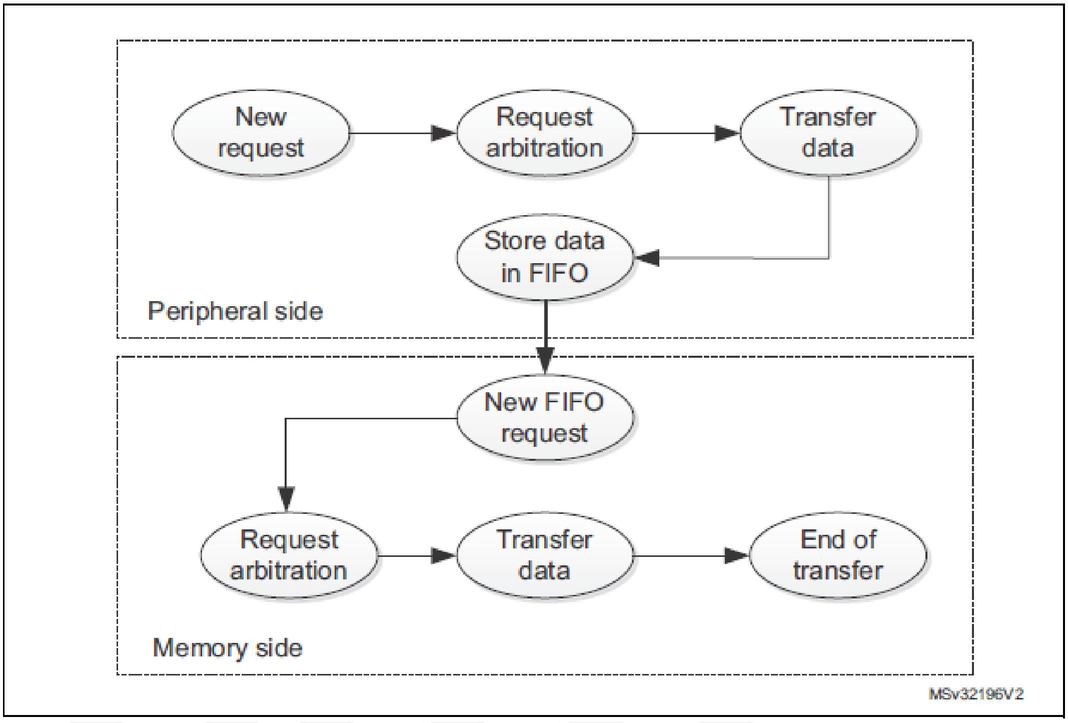


Figure 4.6 Peripheral-to-memory transfer states

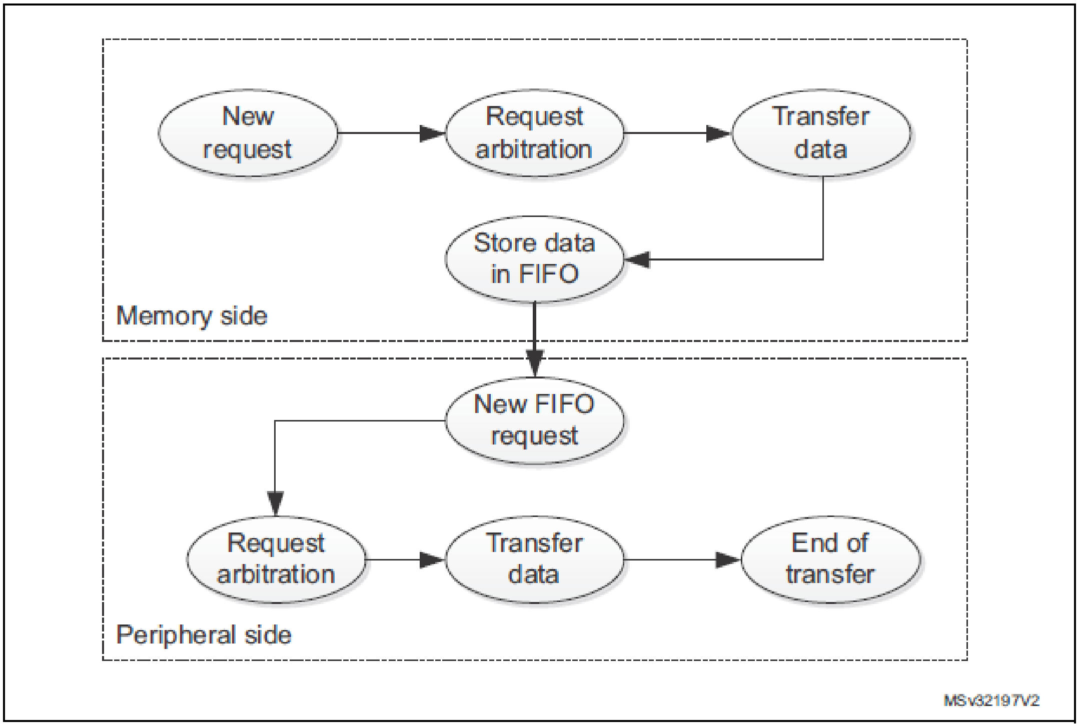


Figure 4.7 Memory-to-peripheral transfer states

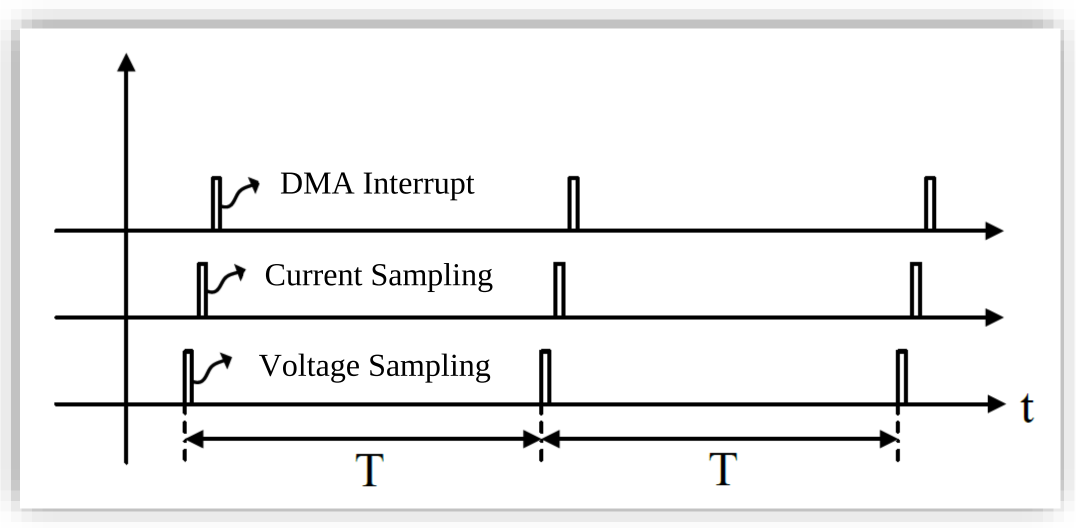


Figure 4.8 Timing diagram for sampling

Below is the cut sub-function for transferring raw values to the sequences where samples are stored in DMA interrupt. When the number of samples to be taken was completed, a message was sent to the task function related to measurement that samples were taken.

```
void DMA2_Stream0_IRQHandler(void)
 {
 if(DMA_GetITStatus(DMA2_Stream0, DMA_IT_TCIF0))
       {
DMA_ClearITPendingBit(DMA2_Stream0, DMA_IT_TCIF0);

stTaskMeas.SampleBuffer[0][stTaskMeas.SampleIndex] =
InstantRawValues[0];
stTaskMeas.SampleBuffer[1][stTaskMeas.SampleIndex] =
InstantRawValues[1];

if(++stTaskMeas.SampleIndex == TOTAL_SAMPLE_NUM)
        {
stTaskMeas.SampleIndex = 0;
KernelMessagePush(eTaskMeas, msgSamplesCollected, 0);
       }
       }
 }
```

When the sampling is completed, the series in which the samples are kept are copied, as new samples will continue to be added to the original series. First of all, since the samples are taken offset, this shift amount must be subtracted from the sample values. This process was applied in two strings containing both voltage and current samples.

```
arm_mean_q15((int16_t *)stTaskMeas.CopyBuffer[0], TOTAL_SAMPLE_NUM,
(int16_t *)&stTaskMeas.AverageVal[0]);
arm_mean_q15((int16_t *)stTaskMeas.CopyBuffer[1], TOTAL_SAMPLE_NUM,
(int16_t *)&stTaskMeas.AverageVal[1]);

for(int i = 0; i < TOTAL_SAMPLE_NUM; i++)
{
 stTaskMeas.CopyBuffer[0][i] -= stTaskMeas.AverageVal[0];
 stTaskMeas.CopyBuffer[1][i] -= stTaskMeas.AverageVal[1];
}
```

As the next step, the sum of the squares of the samples was transferred to the relevant variables, and then the raw effective value was obtained by taking the square root. After this step, the values obtained will be passed through the average filter and the results with a smoother transition will be shown on the graphic screen.

```
arm_power_q15(stTaskMeas.CopyBuffer[0],TOTAL_SAMPLE_NUM,
&TotalSquare[0]);
arm_power_q15(stTaskMeas.CopyBuffer[1],TOTAL_SAMPLE_NUM,
&TotalSquare[1]);
v = (uint16_t)sqrt((double)TotalSquare[0] / (TOTAL_SAMPLE_NUM));
i = (uint16_t)sqrt((double)TotalSquare[1] / (TOTAL_SAMPLE_NUM));
```

4.4 Harmonics and Fast Fourier Transform (FFT) Analysis

Non-linear loads cause harmonic currents to be drawn from the network. Both network voltage and current can be harmonic [33]. Sine and cosine components at different frequencies in a harmonic system can be calculated with the help of Fourier analysis. According to Fourier analysis, a periodic signal can be expressed as the sum of sine and cosine components whose frequencies are exactly multiple of each other.

$$i(t) = i_0 + a_1 \cdot \cos(\omega t) + a_2 \cdot \cos(2\omega t) + \cdots + a_n \cdot \cos(n \cdot \omega t) + b_1 \cdot \sin(\omega t) + b_2 \cdot \sin(2\omega t) + \cdots + bn \cdot \sin(n \cdot \omega t) \tag{4. 3}$$

$$i(t) = i_0 + \sum_{n=1}^{\infty} [A_n \cdot \cos(n \cdot \omega t) + B_n \cdot \sin(n \cdot \omega t)] \tag{4. 4}$$

In a harmonic system, if there is no dc component, the value of becomes zero [33]. If the signal has a single function symmetry Equation (4.5), the cosine components will not be found [33].

$$i(\omega t) = -i(-\omega t) \tag{4. 5}$$

If the signal has a double function symmetry Equation (4.6), there will be no sine components [33].

$$i(\omega t) = i(-\omega t) \tag{4. 6}$$

If the signal has half-wave symmetry Equation (4.7), there will be no even numbered components [33].

$$i(\omega t) = -i(\omega t + \pi) \tag{4. 7}$$

In Fourier analysis, the DC component Equation (4.8), A_n and B_n coefficients can also be calculated using Equation (4.9) and Equation (4.10) [33].

$$i_0 = \frac{1}{T} \cdot \int_0^T i(\omega t) \cdot d\omega t \tag{4. 8}$$

$$A_n = \frac{2}{T} \cdot \int_0^T i(\omega t) \cdot cos(n \cdot \omega t) \cdot d\omega t \tag{4. 9}$$

$$B_n = \frac{2}{T} \cdot \int_0^T i(\omega t) \cdot sin(n \cdot \omega t) \cdot d\omega t \tag{4. 10}$$

Using the coefficients A_n and B_n, the effective value of the relevant harmonic component and the phase difference of the harmonic component can be found as follows [34].

$$a_n \cdot cos(n \cdot \omega t) + b_n \cdot sin(n \cdot \omega t) = I_n \cdot sin(n \cdot \omega t + \varphi_n) \tag{4. 11}$$

$$I_n = \sqrt{A_n^2 + B_n^2} \tag{4. 12}$$

$$\varphi_n = \arctan \frac{a_n}{b_n} \tag{4. 13}$$

The effective values of harmonics can be calculated by Equation (4.12) and phase angles of harmonics can be calculated by Equation (4.13). Effective value including all harmonics can be calculated with Equation (4.14).

$$I = \sqrt{I_1^2 + I_2^2 + I_3^2 + \cdots + I_n^2} \tag{4. 14}$$

With the help of Fourier analysis, we can examine current harmonics on MATLAB as follows.

```
clear all
clc

f = 50;
T = 1/f;
w = 2*pi*f;
Fs = 6400;
Ts = 1/Fs;
L = Fs/f;
t = (0:L-1)*Ts;

N = 35
I_rms = 0;
i = 25*sqrt(2)*sin(w*t)+4*sqrt(2)*sin(4*w*t)+2*sqrt(2)*sin(10*w*t);

An = zeros(1,N);
Bn = zeros(1,N);
Cn = zeros(1,N);
for j=1:length(i)
for n = 1 : N
 An(n)  = An(n) + (2/T)* Ts * i(j) * cos(n*w*(Ts*(j-1)));
 Bn(n)  = Bn(n) + (2/T)* Ts * i(j) * sin(n*w*(Ts*(j-1)));
 Cn(n)  = sqrt( An(n)^2 + Bn(n)^2 )/sqrt(2);
 end
I_rms = I_rms + i(j)^2;
 end
I_rms = sqrt(I_rms / L);
I1_rms = Cn(1);
THDI = sqrt( I_rms^2 - I1_rms^2 ) / I1_rms;

subplot(2,1,1);
bar(Cn); grid; axis tight; ylabel('Harmonic Analyze');
xlabel('Harmonic No');
subplot(2,1,2);
plot(t,i); grid; axis tight; xlabel('time');
```

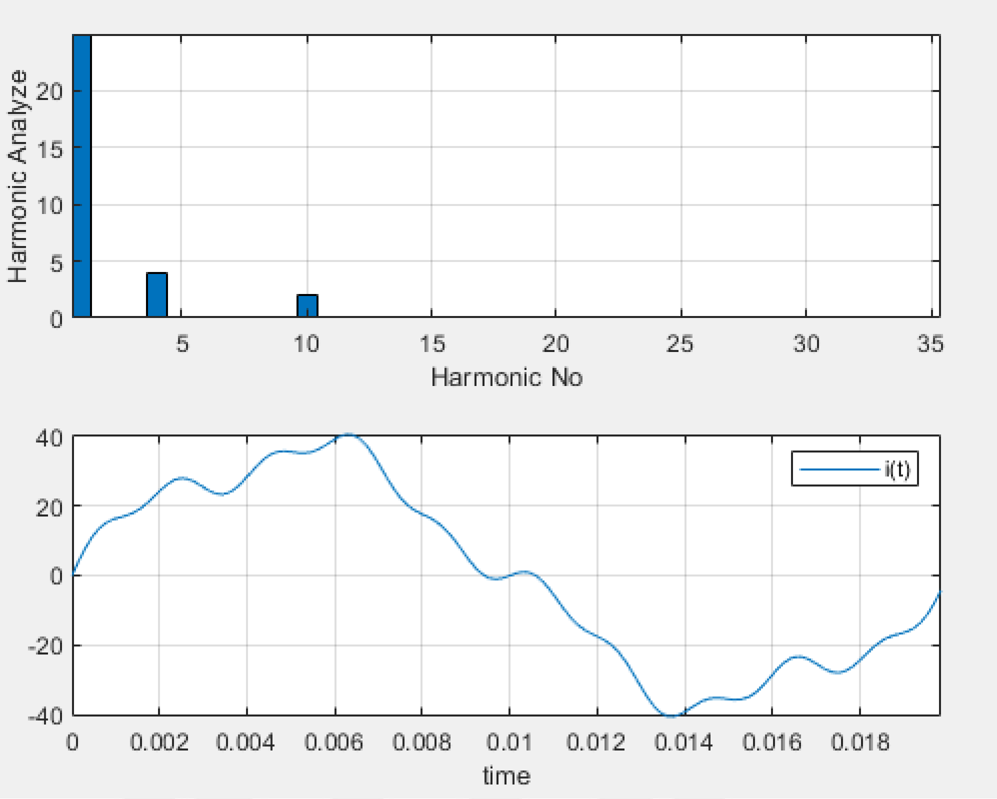


Figure 4.9 Effective values of the Current Harmonics

4.5 The Calculation of Phase Angle [cos(φ)]

After calculating the effective values of voltage and current, the next step will be cos(φ) calculation. For this, first the phase angles of the main harmonic component of the voltage signal and the main harmonic component of the current signal must be calculated [34].

$$\varphi = \varphi_v - \varphi_i \tag{4. 15}$$

The values of φ_v and φ_i can be calculated with the help of A_n and $B_n$coefficients obtained with the help of Fourier analysis as explained in the previous section.

$$\varphi_{1\_v} = arctan\frac{a_{1\_v}}{b_{1\_v}} \tag{4. 16}$$

$$\varphi_{1\_i} = arctan\frac{a_{1\_i}}{b_{1\_i}} \tag{4. 17}$$

In this case;

$$\cos(\varphi) = \cos(\varphi_1) = \cos\,(\varphi_{1\_v} - \varphi_{1\_i}) \tag{4. 18}$$

will be calculated as in Equation (4.18).

The $\cos(\varphi)$ value of a voltage and current signal in Figure (4.10), without harmonics and with a phase difference of 20° can be calculated as follows in MATLAB environment.

```
clear all
clc
f = 50;
T = 1 / f;
Fs = 6400;
Ts = 1/Fs;
Ls = Fs/f;
t = (0:Ls-1)*Ts;
w = 2*pi*f;
N = 1;
v = 120 * sin(w*t);
i = 15 * sin(w*t - (pi/9));

y1=v;
subplot(1,1,1);
plot(t,v); grid; axis tight; xlabel('time');
hold on
y2=i;
plot(t,i); grid; axis tight; xlabel('time');
hold off
```

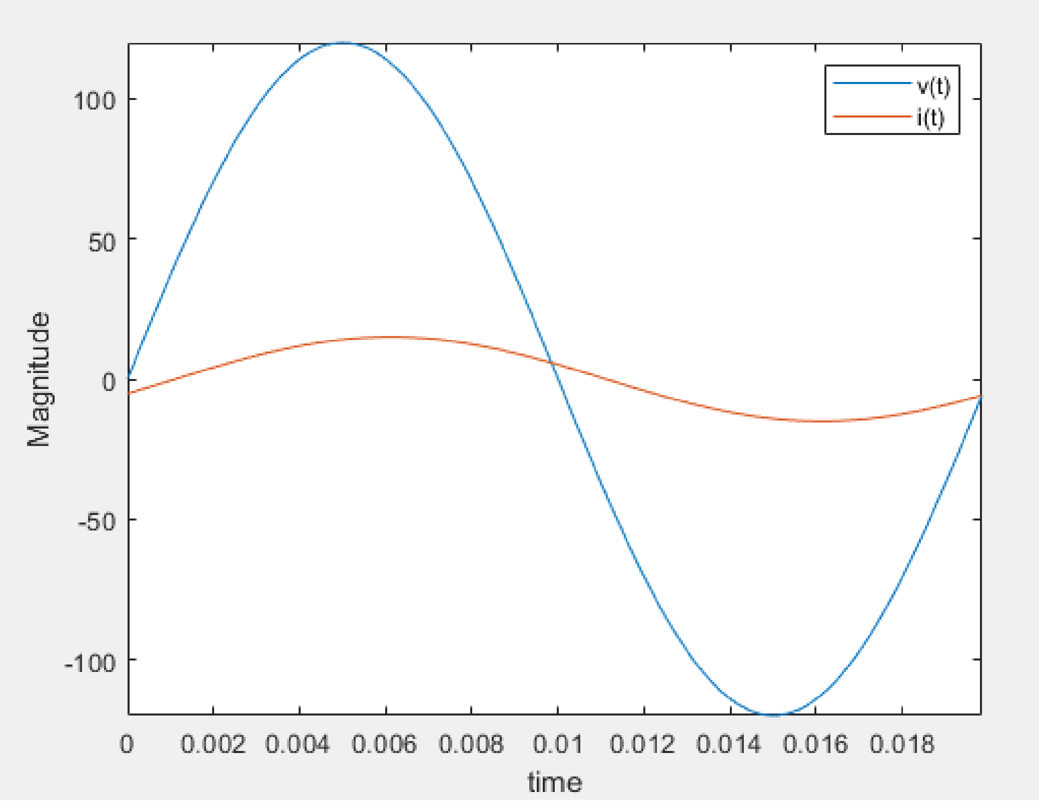
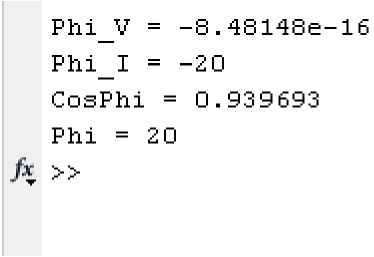


Figure 4.10 Voltage and current signal to be calculated phase difference

```
clear all
clc
f = 50;
T = 1 / f;
Fs = 6400;
Ts = 1/Fs;
Ls = Fs/f;
t = (0:Ls-1)*Ts;
w = 2*pi*f;
N = 1;
v = 120 * sin(w*t);
i = 15 * sin(w*t - (pi/9));
An = zeros(1,N);
Bn = zeros(1,N);
for j=1:length(v)
for n = 1 : N
An(n) = An(n) + (2/T)* Ts * v(j) * cos(n*w*(Ts*(j-1)));
Bn(n) = Bn(n) + (2/T)* Ts * v(j) * sin(n*w*(Ts*(j-1)));
end
end
Phi_V = atand(An(1)/Bn(1));
An = zeros(1,N);
Bn = zeros(1,N);
for j=1:length(i)
for n = 1 : N
An(n) = An(n) + (2/T)* Ts * i(j) * cos(n*w*(Ts*(j-1)));
Bn(n) = Bn(n) + (2/T)* Ts * i(j) * sin(n*w*(Ts*(j-1)));
end
end

Phi_I = atand(An(1)/Bn(1));
Phi = Phi_V - Phi_I;
CosPhi = cosd(Phi);

fprintf('Phi_V = %g\n', Phi_V);
fprintf('Phi_I = %g\n', Phi_I);
fprintf('CosPhi = %g\n',CosPhi);
fprintf('Phi = %g\n',Phi);
```



```
Phi_V = -8.48148e-16
Phi_I = -20
CosPhi = 0.939693
Phi = 20
>>
```

Figure 4.11 MATLAB output of Cos φ calculation for a harmonic-free system

For a voltage and current signal whose current signal is harmonic and whose fundamental component is 30° phase different Figure (4.12), cos(φ) calculation will be as follows.

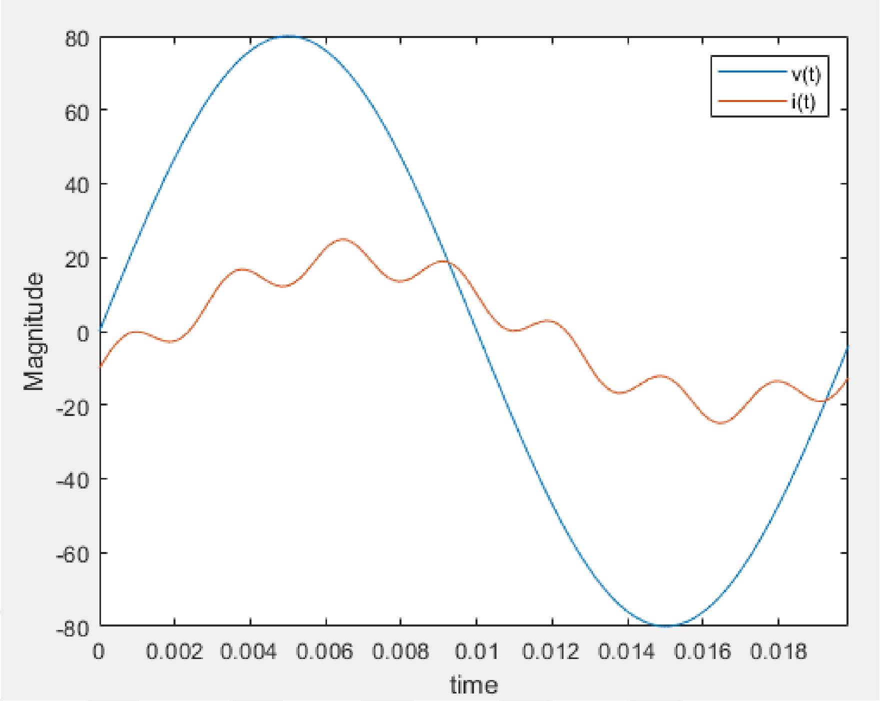
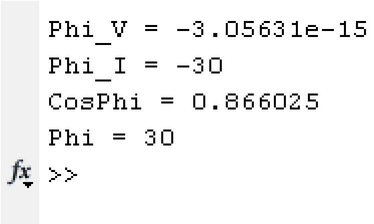


Figure 4.12 Voltage and harmonic current signal to be calculated phase difference

```
v = 80 * sin(w*t);
i = 20 * sin(w*t - (pi/6)) + 5 * sin(7*w*t);
```



```
Phi_V = -3.05631e-15
Phi_I = -30
CosPhi = 0.866025
Phi = 30
>>
```

Figure 4.13 MATLAB output of $\cos(\varphi)$ calculation for a harmonic system

The part that performs the above operations in the software is given in ***Appendix F***.

4.6 The Calculation of Power Factor (PF)

After calculating the phase angle, the next step will be to calculate the power factor. Since $\mathrm{v_{rms}} = \mathrm{v_{1\_rms}}$ and $\mathrm{i_{rms}} = \mathrm{i_{1\_rms}}$ expressions are valid in a system without harmonics, $\cos(\varphi)$ and power factor value are equal.

$$PF = \frac{v_{1\_rms}}{v_{rms}} \cdot \frac{i_{1\_rms}}{i_{rms}} \cdot \cos(\varphi) \qquad (4.\ 19)$$

$$PF = K_v \cdot K_i \cdot \cos(\varphi) \tag{4. 20}$$

In a network whose voltage is generally accepted as non-harmonic, the power factor can be defined by Equation (4.21).

$$PF = \frac{i_{1\_rms}}{i_{rms}} \cdot \cos(\varphi) \tag{4. 21}$$

The total harmonic distortion (THD) value for the current in a harmonic system is defined by the Equation (4.22).

$$THD_I = \frac{\sqrt{I_2^2+I_3^2+I_4^2 \dots +I_n^2}}{I_1} \tag{4. 22}$$

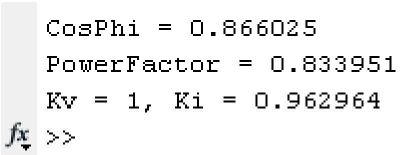
If the network voltage is non-harmonic, it can be written as Equation (4.23) depending on the value of the power factor (THD_I).

$$PF = \frac{1}{\sqrt{1+THD_I{}^2}} \cdot \cos(\varphi) \tag{4. 23}$$

In the software, voltage and current harmonics will be calculated with the help of Fourier analysis and the power factor will be calculated using the obtained THD value and $\cos(\varphi)$. These steps can be performed on MATLAB as follows.

```
clear all
clc
f = 50;
T = 1 / f;
Fs = 6400;
Ts = 1/Fs;
Ls = Fs/f;
t = (0:Ls-1)*Ts;
w = 2*pi*f;
N = 10;

V_rms = 0;
I_rms = 0;

v = 75 * sin(w*t);
i = 25 * sin(w*t - (pi/6)) + 7 * sin(5*w*t);

An = zeros(1,N);
Bn = zeros(1,N);
Cn = zeros(1,N);
```

```
for j=1:length(v)
 for n = 1 : N
An(n) = An(n) + (2/T)* Ts * v(j) * cos(n*w*(Ts*(j-1)));
Bn(n) = Bn(n) + (2/T)* Ts * v(j) * sin(n*w*(Ts*(j-1)));
Cn(n) = sqrt( An(n)^2 + Bn(n)^2 )/sqrt(2);
 end

V_rms = V_rms + v(j)^2;
 end

Phi_V = atand(An(1)/Bn(1));
V_rms = sqrt(V_rms / length(i));
Kv = Cn(1)/V_rms;

An = zeros(1,N);
Bn = zeros(1,N);
Cn = zeros(1,N);

for j=1:length(i)
 for n = 1 : N
An(n) = An(n) + (2/T)* Ts * i(j) * cos(n*w*(Ts*(j-1)));
Bn(n) = Bn(n) + (2/T)* Ts * i(j) * sin(n*w*(Ts*(j-1)));
Cn(n) = sqrt( An(n)^2 + Bn(n)^2 )/sqrt(2);
 end
I_rms = I_rms + i(j)^2;
 end

Phi_I = atand(An(1)/Bn(1));
I_rms = sqrt(I_rms / length(i));
Ki = Cn(1)/I_rms;

Phi = Phi_V - Phi_I;
CosPhi = cosd(Phi);
PF = Kv * Ki * CosPhi;

fprintf('CosPhi = %g\n', CosPhi);
fprintf('PowerFactor = %g\n', PF);
fprintf('Kv = %g, Ki = %g\n', Kv, Ki);
```



```
CosPhi = 0.866025
PowerFactor = 0.833951
Kv = 1, Ki = 0.962964
>>
```

Figure 4.14 Power factor calculation on MATLAB for a harmonic system

The part that performs the above operations in the software is given in ***Appendix G***.

4.7 Power and Energy Calculation

After completing the effective values of voltage and current values, cos(φ) and power factor calculations, the next step is to make power and energy calculations. Apparent power, active power and reactive power are calculated in the software as follows.

```
S = stTaskMeas.Voltage * stTaskMeas.Current;
P = S * stTaskMeas.CosPhi;
Q = S * stTaskMeas.SinPhi;
```

In the energy calculation, the direction of the energy flow can be from the network to the load or from the load to the network, depending on the positive and negative phase angle. If the phase angle is positive, the energy flow is defined as from the network to the load (Import), and if it is negative, it is defined as the direction of production (Export). In addition, it will be taken into account as inductive or capacitive reactive power, depending on whether the reactive power is positive or negative along with the energy flow direction.

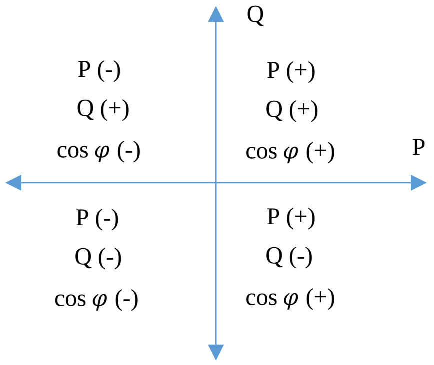


Figure 4.15 State of powers according to the sign of the phase angle

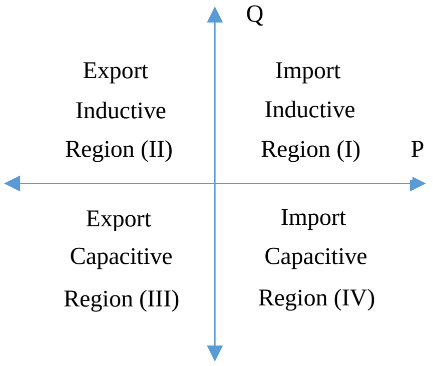


Figure 4.16 State of powers according to the sign of the phase angle

- In the first region, active power and reactive power are positive since $\cos(\varphi)$ and $\sin(\varphi)$ are positive.
- In the second region, active power is negative and reactive power is positive since $\cos(\varphi)$ is negative and $\sin(\varphi)$ is positive.
- In the third region, active power and reactive power is negative since $\cos(\varphi)$ and $\sin(\varphi)$ is negative.
- In the fourth region, active power is positive and reactive power is negative since $\cos(\varphi)$ is positive and $\sin(\varphi)$ is negative.

The right side of the Q-axis is the import direction for the energy flow direction, and the left side is the export direction. In terms of reactive power, the above part of the P-axis is inductive and the below part is capacitive.

The electrical energy value is generally expressed as in Equation (4.24) and Equation (4.25). Its unit is watt-second. In energy systems, it is generally used as watt-hour (Wh) or kilowatt-hour (kWh). In the software, energy values are obtained by collecting cumulatively according to the state of power values and calculated according to kWh unit.

$$E_p = P \cdot t \tag{4. 24}$$

$$E_q = Q \cdot t \tag{4. 25}$$

```
if(stTaskMeas.CosPhi >= 0)
{
stTaskMeas.PowerDirection = IMPORT;
stTaskMeas.stEnergiesImport.ActiveEnergy += ((0.16 * P) / 3600);

if(Q >= 0)
 {
stTaskMeas.stEnergiesImport.InductiveReactiveEnergy += ((0.16 * Q)
/ 3600);
 }
else
      {
stTaskMeas.stEnergiesImport.CapacitiveReactiveEnergy += ((0.16 * -
Q) / 3600);
      }
}
else
 {
stTaskMeas.PowerDirection = EXPORT;

stTaskMeas.stEnergiesExport.ActiveEnergy += ((0.16 * -P) / 3600);

if(Q >= 0)
  {
stTaskMeas.stEnergiesExport.InductiveReactiveEnergy += ((0.16 * Q)
/ 3600);
 }
else
  {
stTaskMeas.stEnergiesExport.CapacitiveReactiveEnergy += ((0.16 * -
Q) / 3600);
}
```

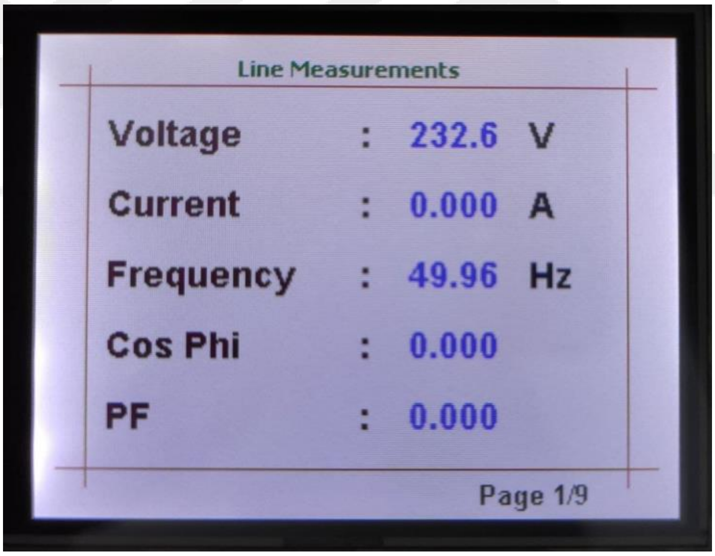
Since the sampling process in the software is based on 8 period sampling as 128 samples per period, the calculations are made over 1024 samples. Therefore, since 1024 samples were taken at 160ms intervals, the energy values were calculated by taking the calculated power values during this time as a reference.

CHAPTER V

HARDWARE REALIZATION AND MEASUREMENT RESULTS

5.1 Working Screens

9 operating screens are designed for the graphic display on the hardware. Measurement results can be followed through these screens. Apart from this, it will also be possible to read the measurement results via Modbus communication. Address maps for Modbus communication are available in ***Appendix H***.



Line Measurements

Voltage : 232.6 V

Current : 0.000 A

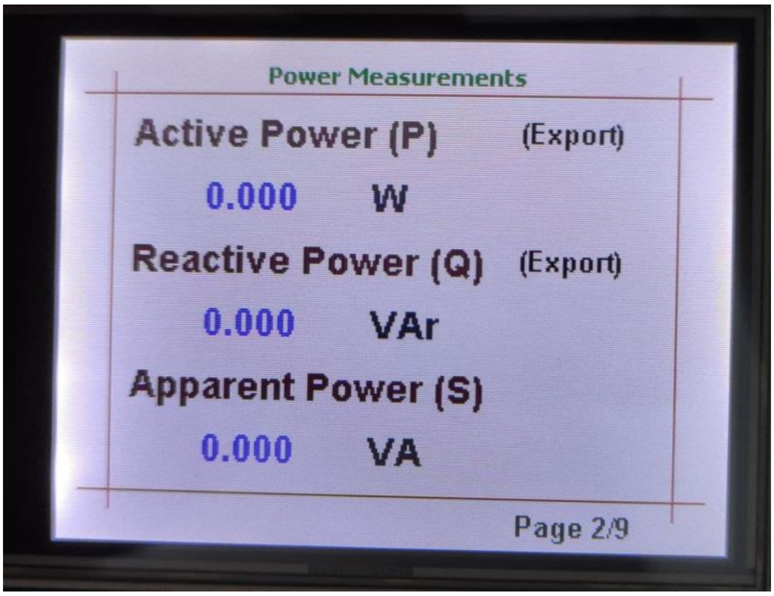
Frequency : 49.96 Hz

Cos Phi : 0.000

PF : 0.000

Page 1/9

Figure 5.1 Voltage, Current, frequency, Cos φ and PF analyze screen without the load



Power Measurements

Active Power (P) (Export)

0.000 W

Reactive Power (Q) (Export)

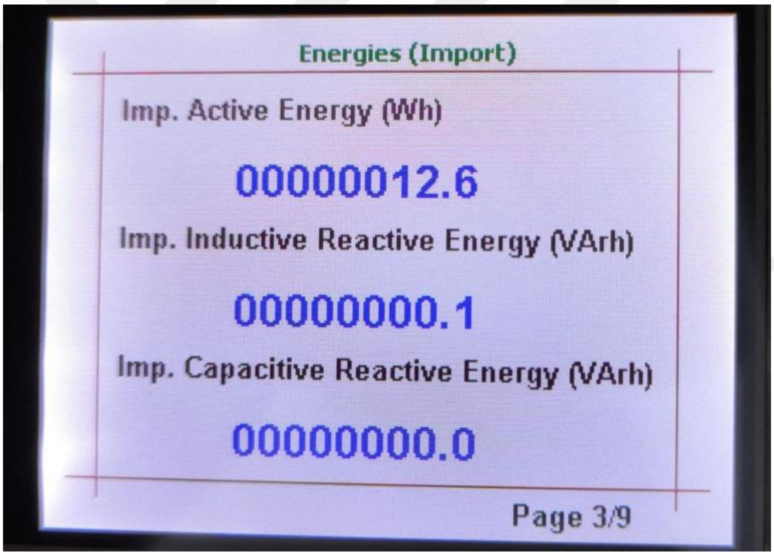
0.000 VAr

Apparent Power (S)

0.000 VA

Page 2/9

Figure 5.2 Active, Reactive and Apparent Power Analyze Screen without the load



Energies (Import)

Imp. Active Energy (Wh)

00000012.6

Imp. Inductive Reactive Energy (VArh)

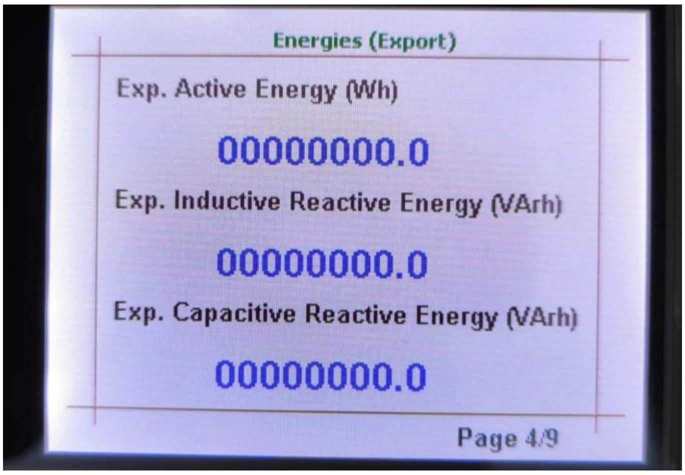
00000000.1

Imp. Capacitive Reactive Energy (VArh)

00000000.0

Page 3/9

Figure 5.3 Imported Active, Inductive and Reactive Energy Analyze Screen without the load


Energies (Export)

Exp. Active Energy (Wh)

00000000.0

Exp. Inductive Reactive Energy (VArh)

00000000.0

Exp. Capacitive Reactive Energy (VArh)

00000000.0

Page 4/9

Figure 5.4 Exported Active, Inductive and Reactive Energy Analyze Screen without the load

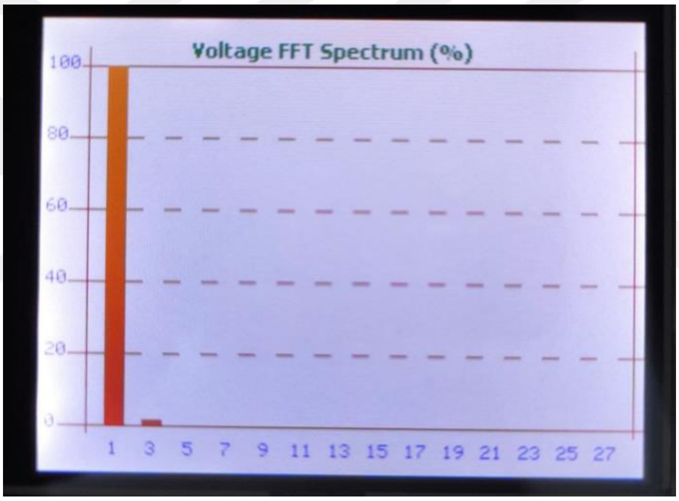
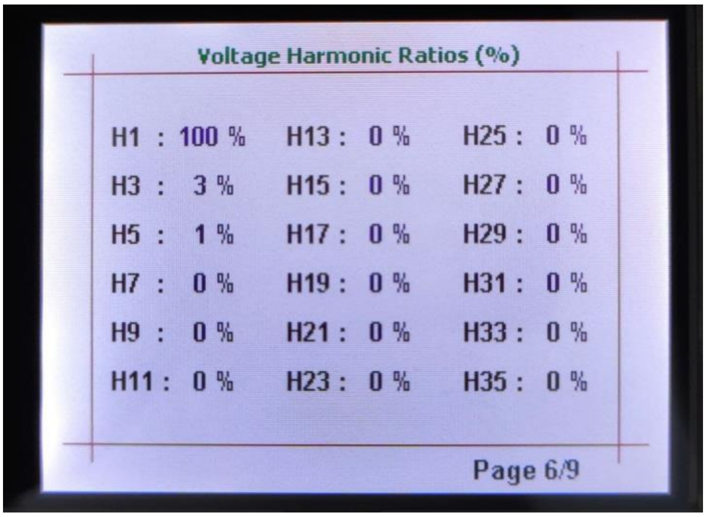


Figure 5.5 Voltage Harmonical Spectrum Graphical Analyze Screen without the load



Voltage Harmonic Ratios (%)

H1 : 100 % H13 : 0 % H25 : 0 %
H3 : 3 % H15 : 0 % H27 : 0 %
H5 : 1 % H17 : 0 % H29 : 0 %
H7 : 0 % H19 : 0 % H31 : 0 %
H9 : 0 % H21 : 0 % H33 : 0 %
H11 : 0 % H23 : 0 % H35 : 0 %

Page 6/9

Figure 5.6 Voltage Harmonic Ratios Analyzed Values without the load

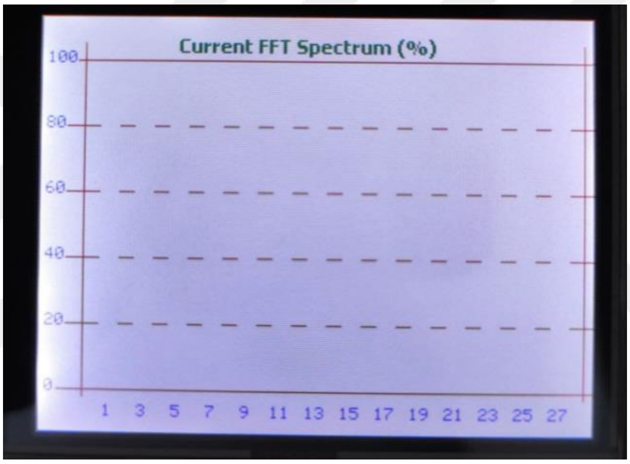
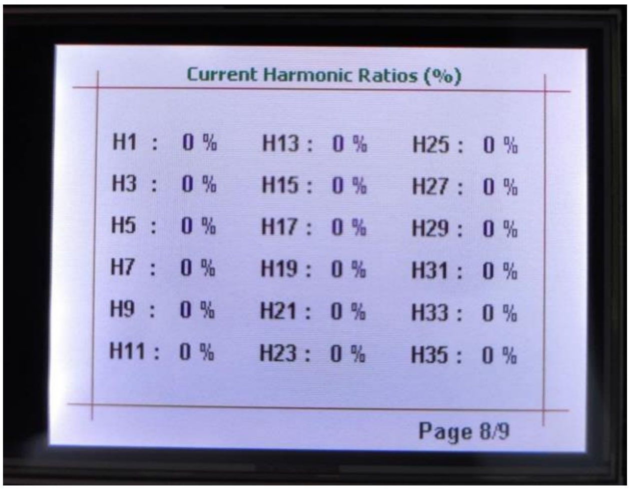


Figure 5.7 Current Harmonical Spectrum Graphical Analyze Screen without the load

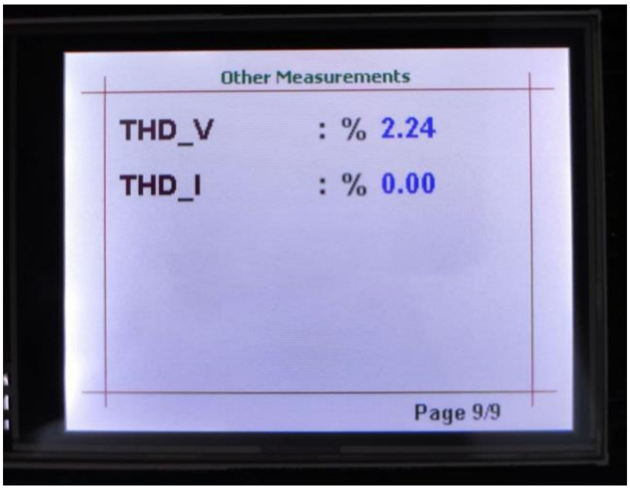


Current Harmonic Ratios (%)

H1 : 0 % H13 : 0 % H25 : 0 %
H3 : 0 % H15 : 0 % H27 : 0 %
H5 : 0 % H17 : 0 % H29 : 0 %
H7 : 0 % H19 : 0 % H31 : 0 %
H9 : 0 % H21 : 0 % H33 : 0 %
H11 : 0 % H23 : 0 % H35 : 0 %

Page 8/9

Figure 5.8 Current Harmonic Ratios Analyzed Values without the load

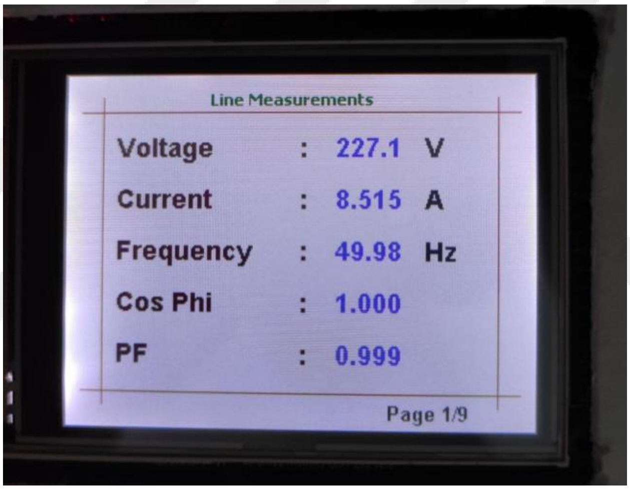

Other Measurements

THD_V : % 2.24

THD_I : % 0.00

Page 9/9

Figure 5.9 Total Harmonic Distortion on the Voltage and Current Analyze Screen without the load


Line Measurements

Voltage : 227.1 V

Current : 8.515 A

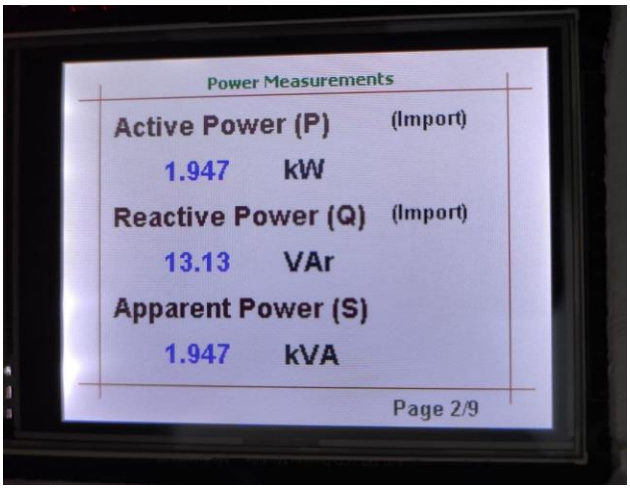
Frequency : 49.98 Hz

Cos Phi : 1.000

PF : 0.999

Page 1/9

Figure 5.10 Voltage, Current, frequency, Cos φ and PF analyze screen on the load


Power Measurements

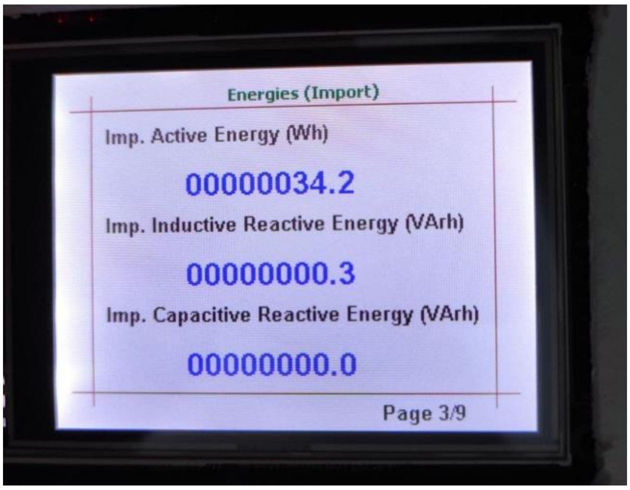
Active Power (P) (Import)
1.947 kW

Reactive Power (Q) (Import)
13.13 VAr

Apparent Power (S)
1.947 kVA

Page 2/9

Figure 5.11 Active, Reactive and Apparent Power Analyze Screen on the load



Energies (Import)

Imp. Active Energy (Wh)

00000034.2

Imp. Inductive Reactive Energy (VArh)

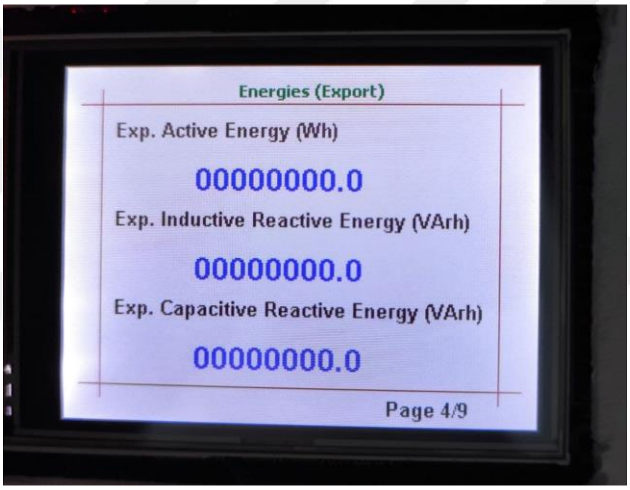
00000000.3

Imp. Capacitive Reactive Energy (VArh)

00000000.0

Page 3/9

Figure 5.12 Imported Active, Inductive and Reactive Energy Analyze Screen on the load



Energies (Export)

Exp. Active Energy (Wh)

00000000.0

Exp. Inductive Reactive Energy (VArh)

00000000.0

Exp. Capacitive Reactive Energy (VArh)

00000000.0

Page 4/9

Figure 5.13 Exported Active, Inductive and Reactive Energy Analyze Screen on the load

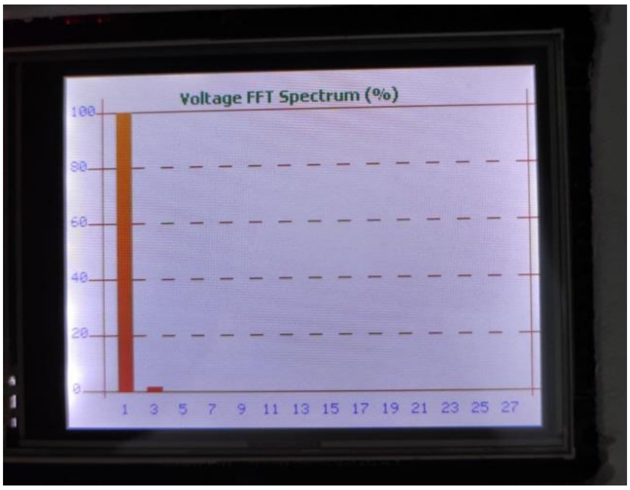
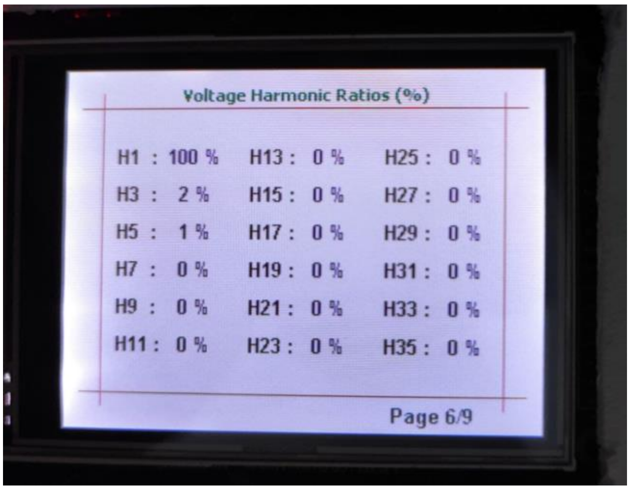


Figure 5.14 Voltage Harmonical Spectrum Graphical Analyze Screen on the load



Voltage Harmonic Ratios (%)

H1 : 100 %	H13 : 0 %	H25 : 0 %
H3 : 2 %	H15 : 0 %	H27 : 0 %
H5 : 1 %	H17 : 0 %	H29 : 0 %
H7 : 0 %	H19 : 0 %	H31 : 0 %
H9 : 0 %	H21 : 0 %	H33 : 0 %
H11 : 0 %	H23 : 0 %	H35 : 0 %

Page 6/9

Figure 5.15 Voltage Harmonic Ratios Analyzed Values on the load

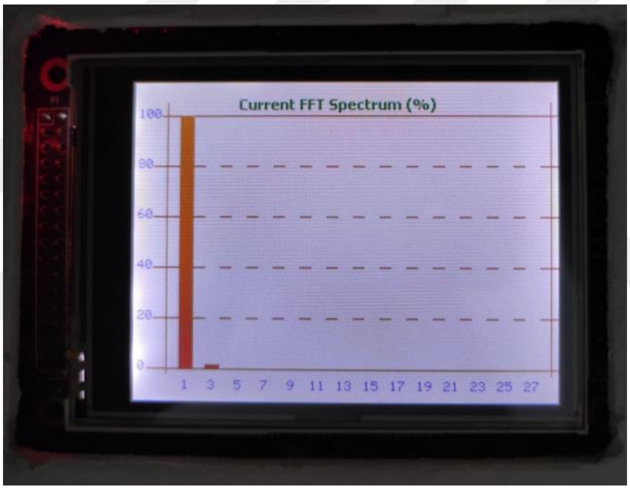
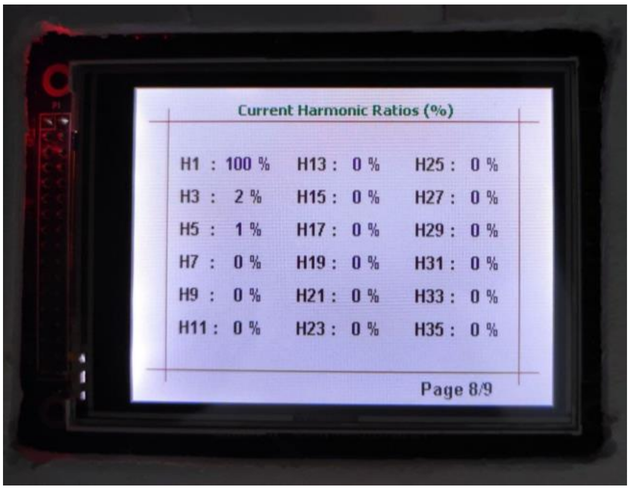


Figure 5.16 Current Harmonical Spectrum Graphical Analyze Screen on the load

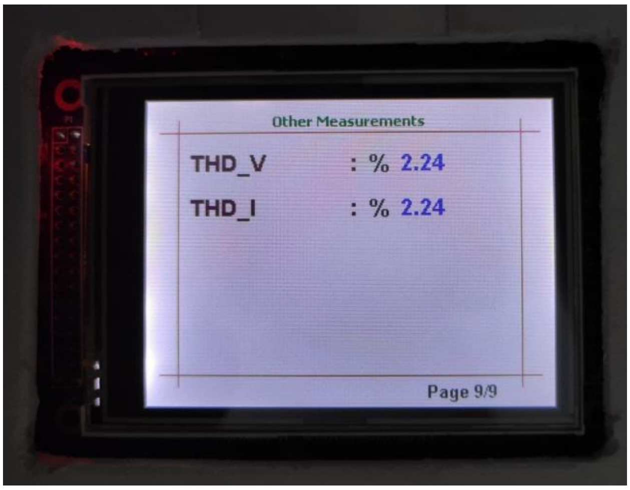


Current Harmonic Ratios (%)

H1 : 100 %	H13 : 0 %	H25 : 0 %
H3 : 2 %	H15 : 0 %	H27 : 0 %
H5 : 1 %	H17 : 0 %	H29 : 0 %
H7 : 0 %	H19 : 0 %	H31 : 0 %
H9 : 0 %	H21 : 0 %	H33 : 0 %
H11 : 0 %	H23 : 0 %	H35 : 0 %

Page 8/9

Figure 5.17 Current Harmonic Ratios Analyzed Values on the load



Other Measurements

THD_V : % 2.24

THD_I : % 2.24

Page 9/9

Figure 5.18 Total Harmonic Distortions on the Voltage and Current Analyze Screen on the load

5.2 The Effect of Sampling Frequency on the Measurements

In this section, sampling at different sampling frequencies was used and a comparison was made between the measurement results and the reference measuring instrument. ***V **** is the value read from the multimeter, ***V*** is the value measured by us on the graphic display.

Table 5.1 Experiment results with the different sampling frequency

F_s	N	V*(V)	V(V)	I(A)	Voltage Error (%)
12,8 kHz	256	227,5	226,25	8,605	0,549
6,4 kHz	128	225,3	224,22	8,683	0,479
3,2 kHz	64	226,3	225,38	8,638	0,406
1,6 kHz	32	226,8	225,74	8,624	0,467

5.3 The Effect of Analog Digital Converter Resolution on Measurements

In this section, the measurement results of various resolutions are compared by changing the resolution of the microcontroller to analog digital converter. ***V **** is the value read from the multimeter and ***V*** indicates the values measured by us on the graphic display.

Table 5.2 Experiment results with the different resolutions of ADC

Resolution	V*(V)	V(V)	Error Rate (%)
12-bits	221,12	220,40	0,325
10-bits	224,56	222,15	1,073
8-bits	223,59	216,43	3,202
6-bits	227,47	210,65	7,39

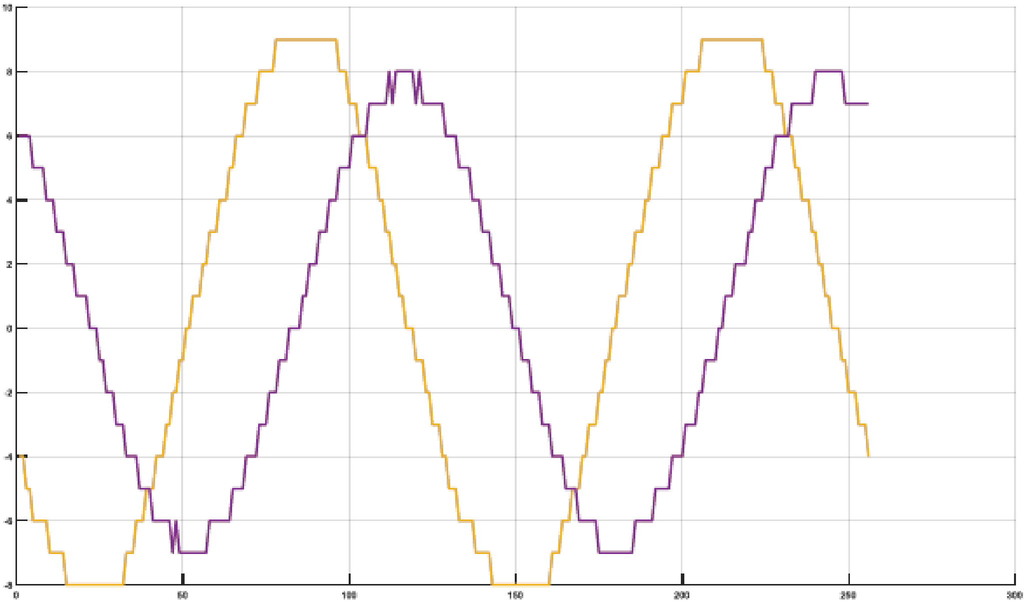


Figure 5.19 Voltage and Current Graphs when the ADC has 6-bits resolution

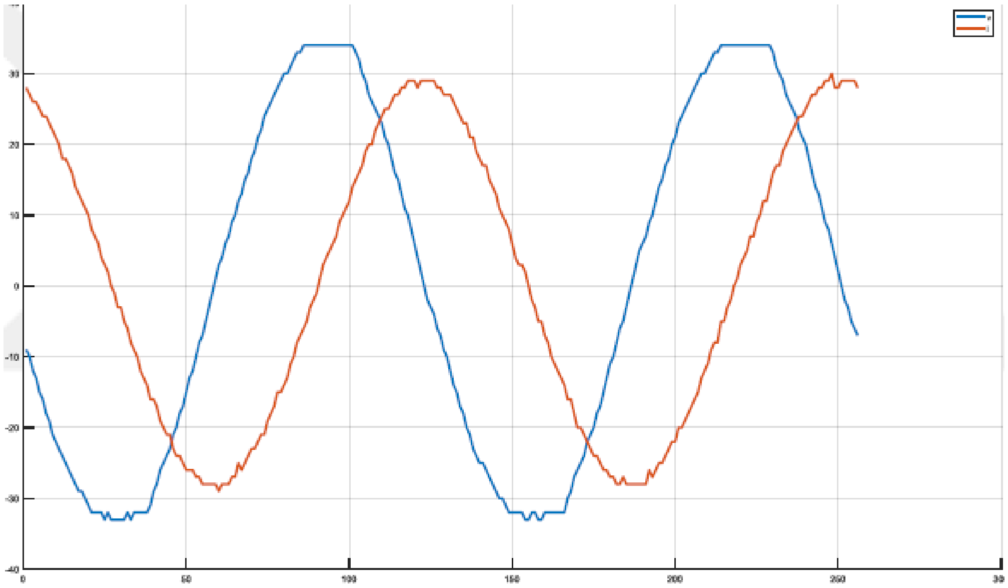


Figure 5.20 Voltage and Current Graphs when the ADC has 8-bits resolution

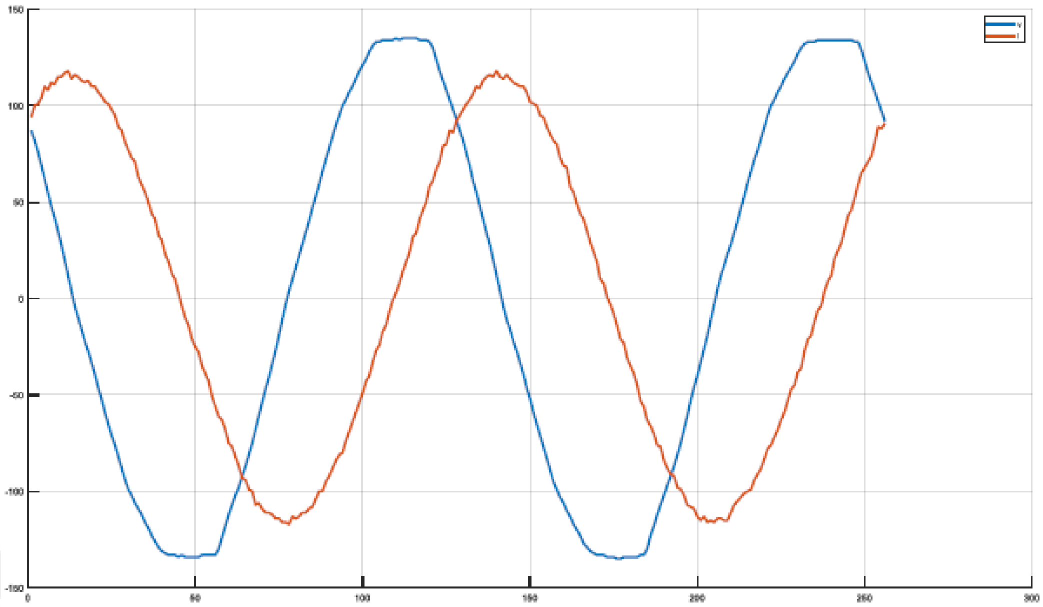


Figure 5.21 Voltage and Current Graphs when the ADC has 10-bits resolution

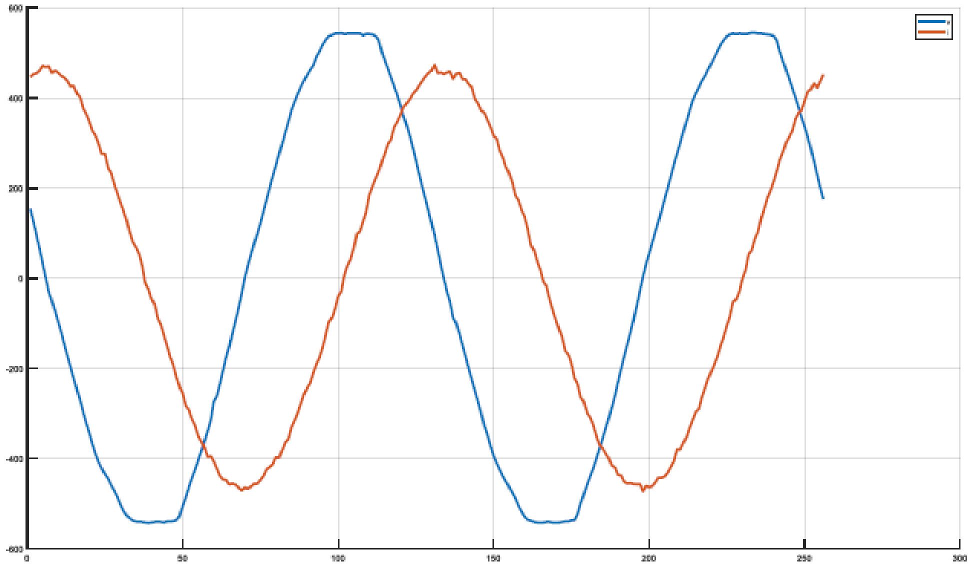


Figure 5.22 Voltage and Current Graphs when the ADC has 12-bits resolution

5.4 The Effect of Total Number of Samples on Measurements

In this section, the total number of samples taken was kept at various values and the calculated measurement results were compared. The number of samples taken in a period was kept as 128, and the measurement period was changed to 1, 2, 3, 4, 5, 6, 7, 8 and different values were obtained in the total number of samples.

Table 5.3 Experiment results with the different samples of ADC

Number of Samples (N)	Period (T)	V*(V)	V(V)	Error Rate (%)
128	1	227,33	226,68	0,285
256	2	227,69	226,61	0,474
384	3	227,27	226,14	0,497
512	4	227,29	225,83	0,642
640	5	226,92	224,61	1,017
768	6	227,18	225,87	0,576
896	7	226,74	225,53	0,533
1024	8	226,42	225,44	0,432

5.5 Other Factors That Effective on The Results

It’s proved that various factors directly affect the measurement quality. Temperature changes of electronic components used are one in all the important factors. thanks to the optimal working environment of the components are different, so as to attenuate the negative impact of this case, the materials used is preferred with higher sensitivity and lower temperature coefficient. Also, the field effect sensor utilized in the current input is tormented by the external magnetic field. To avoid negative effect of this case, it's going to be possible to produce magnetic isolation of the area where the sensor is located on the hardware or standard transformer like rated 250/5A is also accustomed get far better results with the high precision. However, a current transformer requires much space. Besides, sensitivity of the operational amplifiers utilized in voltage and current inputs is critical. a low precision operational amplifier will cause irrelevant results, especially when making measurements too close to zero. To avoid this case, choosing a high quality and high precision operational amplifier will influence positively affect the results. Additionally, chosen MCU is vital. If a microcontroller with high performance and decimal processing capability isn't preferred, it'll influence negatively the results in order that it'll cause loss of sensitivity during the mathematical operations.

In addition to that, low-battery voltage also will affect negatively the calculations. For the remainder, the second revolution of PCB also generated and everyone necessary revisions are upgraded. thanks to its open source, you'll be able to modify the PCB software as you desire. The software is sort of the identical but the screenshotted PCB can be modified.

CHAPTER VI

MODBUS COMMUNICATION

6.1 Modbus Communication

There are a number of agreements designed for RS-232 or RS-485 communications that are still very much in place in automated systems. Modbus is probably the most well-known and widely accepted, and is described here as a typical example. The Modbus serial communications protocol is a standard designed to integrate PLCs, computers, terminals, sensors and actuators. Modbus is a master / slave system which means that one device, the master node, controls all serial functions by selecting slave resources. Modbus supports one main device and up to 247 slave devices. Each device is assigned a unique node address. There are two Modbus variants: ASCII and RTU. ASCII mode uses 'printable' message format. ASCII messages start with a colon and end with a cart return. [24]

RTU mode uses binary so it is not 'printable'. Eight-character characters are sent as a continuous explosion and the end of the message is defined by 3.5 times silent periods. RTU mode messages use half of the letters of the same ASCII message. Only a professional who starts work. The master is usually the PC in charge or HMI device because most Modicon PLCs are addictive and cannot execute Modbus function (new Quantum PLCs can function as Modbus masters). Usually, the householder will read or write letters to the slave. In each case, the slave will return a reply to the message. With the learning function, the response will handle the requested data. For a writing task, feedback is used to ensure acceptance of the writing instruction. A special case is the ‘broadcast’ function where the writing function can be directed to all slaves. In this case, no reply is coming. The 8-bit address field is the first part of the message (1 RTU byte, or 2 ASCII characters). This field indicates the local address of the slave who should respond to the message; all the slaves get the message but only the slave in question will act on it. The job code field tells the slave written what work to do.

Modbus performance codes are specifically designed to communicate with PLC in Modbus industrial communication system. Two error checkers are added at the end of each message: ASCII mode uses longitudinal multiplication test (LRC), and RTU mode uses 16-bit CRC test. In the examples in Table 6.1 and Table 6.2, the home PC starts a three-frame study application starting with # 1.08 from call address 06. The initial catch register is 40108 but '4 'has been released in a series of messages. and the entire register address is 'one bit' (0108 becomes 0107, 0107 is entered 006B in hexadecimal). The answer repeats the address and code of operation, but includes the values read in the drive.

There are four types of types in Modbus communication.

- Coil
- Discrete Login
- Holding Registration
- Login Registration

Parameter types are in bit or 16-bit formats. Bit parameter type and only readable type parameter is called "Discrete Input", and the type that can be read and written is called "Coil". Parameters of 16bit type and readable only type are named as "Input Register", and those that can be both read and written are called "Holding Register" [24]. The parameter list for the designed hardware is given in ***Appendix H***. All parameters are defined in read-only Input Register type. The basic data type is defined as 16-bit, 32-bit parameters, two 16-bit parameters in sequence.

Table 6.1 Query

Field Name	RTU (hex)	ASCII Characters
Header	None	: (colon)
Slave Address	06	0 6
Function	03	0 3
Starting address High	00	0 0
Starting address Low	6B	6 B
No. of registers High	00	0 0
No. of registers Low	03	0 3
Error check	CRC(2-bytes)	LRC (2 chars)
Trailer	None	CRLF
Total Bytes	8	17

CRLF: Carriage Return Line Feed

Table 6.2 Response

Field Name	RTU (hex)	ASCII Characters
Header	None	: (colon)
Slave Address	06	0 6
Function	03	0 3
Byte count	06	0 6
Data High	02	0 0
Data Low	2B	2 B
Data High	00	0 0
Data Low	00	0 0
Data High	00	0 0
Data Low	63	6 3
Error check	CRC(2-bytes)	LRC (2 chars)
Trailer	None	None
Total Bytes	11	23

CHAPTER VII

CONCLUSION AND FUTURE WORKS

Within the scope of this thesis, an embedded system has been developed so as to make numerical measurements in electrical grid systems and therefore the factors which will affect the measurement are investigated by making experiments on various systems supported IEEE 1459-2010 Standards [1].

It’s been proved that some factors directly affect the measurements. The first thing is temperature changes of electronic components. In the operation progress, the active components switching sustainedly and during these operations they are warming. A good ventilating by the fan and heatsink with the thermal grease will prepare the equipment working on properly under hard environments. And also, the optimal working environment of the components are different, so as to attenuate the negative impact of this case, the materials used is preferred with higher sensitivity and lower temperature coefficient. As the second, the field effect sensor utilized in the current input is tormented by the external magnetic field. To avoid negative effect of this case, it's going to be possible to produce magnetic isolation of the area where the sensor is located on the hardware. The third one is a compact current transformer that is rated 100A/4mA is also accustomed get far better results with the high precision. However, a current transformer requires much space. But there are connection input terminals on the last PCB, so it can be connected to system externally.

Besides, sensitivity of the operational amplifiers utilized in voltage and current inputs is critical. A low precision operational amplifier will cause irrelevant results, especially when making measurements too close to zero. To avoid this case, choosing a high quality and high precision operational amplifier will influence positively affect the results.

Additionally, chosen MCU is vital. If a microcontroller with high performance and decimal processing capability isn't preferred, it'll influence negatively the results in order that it'll cause loss of sensitivity during the mathematical operations. For the remainder, the second revolution of PCB also generated and everyone necessary revisions are upgraded. thanks to its open source, you'll be able to modify the PCB software as you desire. The software is sort of the identical but the screenshotted PCB is modified.

The sampling frequency of the selected microcontroller to be used in the designed digital measurement system, the resolution of the analog-digital converter, the sampling rate and the number of ADC channels that can operate simultaneously are sufficient, calculation can be performed without losing time in sampling.
In addition, if the temperature dependence of the sensors and electronic materials used is calibrated with an external temperature measurement, an error-free measurement can be obtained.

Another important factor is main supply voltage. Due to calculation reference voltage and component working voltage is important, even small changes affect the measurements. So, with the low DC voltage or high voltage has a critical role on the measurements.

The last thing is measurements occurring on the grid, and it’s always rippling. So, the measurement results may change suddenly. With the taking more sample, it’ll be able to get much better results.

REFERENCES

[1] IEEE-Std 1459-2010, (2010). IEEE Standard Definitions for the Measurement of Electric Power Quantities Under Sinusoidal, Nonsinusoidal, Balanced, or unbalanced Conditions.

[2] Uğur, A., (2013). *Elektrik Elektronik Devrelerinin Analizi*. Alfa.

[3] Uğur, A., (2007). *Elektrik Elektronik Mühendisliğinin Temelleri -Alternatif Akım Devreleri*. 3rd edition, Alfa.

[4] Wikipedia., (2021). IEC 61000-3-2 Harmonic current limits for class A, B, C, D. Intl.: https://en.wikipedia.org/wiki/IEC_61000-3-2#Harmonic_current_limits_for_class_A,B,C_and_D, 15.01.2022.

[5] Wikipedia, (2021). IEC 61000-3-2 Electromagnetic compatibility. Intl.: https://en.wikipedia.org/w/index.php?title=IEC_61000-3-2&action=history, 15.01.2022.

[6] Texas Instruments, (2013-rev.2017). Software Phase Locked Loop Design Using C2000 Microcontrollers for Single Phase Grid Connected Inverter. Application Report.

[7] Kevin, S., Michel, A. Phase Noise of Integer-N and Fractional-N PLL Synthesizers. Intl.: https://www.analog.com/en/technical-articles/phase-noise-of-integer-n-and-fractional-n-pll-synthesizers.html, 15.01.2022.

[8] Ayhan, Ö., Mehmet, T., (2014). PLL Based Digital Adaptive Filter for Detecting Interharmonics. Intl.: https://www.researchgate.net/publication/275068708_PLL_Based_Digital_Adaptive_Filter_for_Detecting_Interharmonics, 15.01.2022.

[9] Djordje, S., Nikola, G., Marco, R., Saša, M., (2018). Novel orthogonal signal generator for single phase PLL applications.

[10] Francisco, D., (2008). Robust Phase Locked Loops Optimized for DSP implementation in Power Quality Applications, IECON 2008, 3052-3057.

[11] Texas Instruments, TMS320F28030, TMS320F28031, TMS320F28032, TMS320F28033, TMS320F28034, TMS320F28035 Piccolo Microcontrollers Data Manual (SPRS584).

[12] Marco, L., Pedro, R., Remus, T., (2011). Grid Converters for Photovoltaic and Wind Power Systems. John Wiley & Sons Ltd.

[13] Pedro, R., J., Pou, Joan, B., Ignacio, C., (2007). *Double Synchronous Reference Frame PLL for Power Converters Control*. **22(2)**.

[14] Alexander, C., K., Sadiku, M., N., O., 3rd Edition. *Fundamentals of Electric Circuits*. Mc. Graw Hill.

[15] Blooming, T., M., Carnovale, D., (2006). *Application of IEEE STD 519-1992 Harmonic Limits. Annual Pulp and Paper Industry Technical Conference*, Appleton, WI.

[16] ST Microelectronics, (2020). STM32F405xx STM32F407xx Microcontroller Datasheet.

[17] ST Microelectronics, (2016). Using the STM32F2, STM32F4 and STM32F7 Series DMA controller Datasheet. AN4031.

[18] Ayhan, Ö., Mehmet, T., (2014). PLL Based Digital Adaptive Filter for Detecting Interharmonics. Hindawi Publishing Corporation Mathematical Problems in Engineering. Article ID 501781

[19] Ferudun G., Faruk, B., (2019). Tek Fazlı Bir Enerji Analizörünün Gerçekleştirilmesi. Master Thesis.

[20] James, N., Susan, R., 9th Edition. *NILSSON RIEDEL, Electric Circuits*. Chapter 9.

[21] Stojić, D., Georgijević, N., Rivera, M., Milić, S., (2018). Novel ortogonal signal generator for single phase PLL applications. *IET Power Electronics*, **11(3)**, 427-433.

[22] Andrew, M., (1990). Electric Power Measuring System. United States Patent Office, US4980634A.

[23] William, E., Andrew, M., Charles, D., (1991). Power Line Measurement System. United States Patent Office, US5027285A.

[24] MODICON, Inc., (1996). Modbus Protocol Reference Guide.

[25] Haoyu Electronics. 3,2" Touch Screen TFT LCD with 16 bit parallel interface. Int.l: https://www.hotmcu.com/32-touch-screen-tft-lcd-with-16-bit-parallel-interface-p-36.html , 15.01.2022.

[26] Charles, A., Matthew, S., (2013). *Fundamentals of Electric Circuits. 5th Edition*, Mc. Graw Hill.

[27] Thomas, B., Daniel, C., (2006). Application of IEEE STD 519-1992 Harmonic Limits. Annual Pulp and Paper Industry Technical Conference, Appleton, WI, June 18- 23 2006.

[28] Texas Instruments, (2017). Software Phase Locked Loop Design Using C2000 Microcontrollers for Single Phase Grid Connected Inverter. Application Report, (2013-rev.2017).

[29] LSIS Co. Ltd., (2016). Electric power measuring system. JUSTIA Patents, Patent: 9863986, July 13, 2016.

[30] P., Wattanayingcharoen, A., Detchrat, Sakreya, C., (2012). *Developing Harmonic Power Analyzer based on IEEE 1459-2010 Standard. Proceedings of the International Multiconference of Engineers and Computer Scientists 2012* **(2),** IMECS 2012, Hong Kong, March 14-16, 2012.

[31] George, T., Joel, H., Frank, G., (2005). *Thomas' Calculus. 11th Edition*, TURKISH language edition published by Beta Basim Yayim Dağitim A.S., (2009), ISBN 978 - 605 - 377 - 068 – 8.

[32] Bill, D., (2009). *The Control Techniques Drives and Controls Handbook*. 2nd Edition. The Institution of Engineering and Technology, ISBN 978-1-84919-101-2 (PDF).

[33] Hacı, B., (2014). Güç Elektroniği. Power Electronics Lecture Notes, Yıldız Technical University, İSTANBUL.

[34] Ümit, P., Uğur, A., (2004). *Statik Reaktif Güç Kompanzasyonu Uygulaması ve Matlab Simulasyonu*. Sakarya University.

[35] ST Microelectronics, (2021). ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32 Datasheet.

[36] Uğur, P., Ergün, E., (2021). ARM Based Development of Embedded System for an Energy and Harmonic Analyzer, Gaziantep University, *Manchester Journal of Artificial Intelligence & Applied Sciences*, **2(1).** 2021, ISSN: 2634-1034.

APPENDIX

Appendix A: Simulating the PLL for Varying Conditions for Notch Filter

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PLL Simulating for notch
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Select numeric type, let's choose Q21
T=numerictype('WordLength',32,'FractionLength',21);
%Specify math attributes to the fimath object
F=fimath('RoundMode','floor','OverflowMode','wrap');
F.ProductMode='SpecifyPrecision';
F.ProductWordLength=32;
F.ProductFractionLength=21;
F.SumMode='SpecifyPrecision';
F.SumWordLength=32;
F.SumFractionLength=21;
%specify fipref object, to display warning in cases of overflow and
%underflow
P=fipref;
P.LoggingMode='on';
P.NumericTypeDisplay='none';
P.FimathDisplay='none';
%PLL Modelling starts from here
Fs=50000; %Sampling frequency = 50Khz
GridFreq=50; %Nominal Grid Frequency in Hz
Tfinal=0.2; %Time the simulation is run for = 0.5 seconds
Ts=1/Fs; %Sampling Time = 1/Fs
t=0:Ts:Tfinal; %Simulation Time vector
wn=2*pi*GridFreq; %Nominal Grid Frequency in radians
%generate input signal and create a fi object of it
%input wave with a phase jump at the mid point of simulation
% CASE 1 : Phase Jump at the Mid Point
L=length(t);
for n=1:floor(L)
u(n)=sin(2*pi*GridFreq*Ts*n);
end
for n=1:floor(L)
u1(n)=sin(2*pi*GridFreq*Ts*n);
end
for n=floor(L/2):L
u(n)=sin(2*pi*GridFreq*Ts*n+pi/2);
end
%CASE 2 : Harmonics
 L=length(t);
 for n=1:floor(L)
 u(n)=0.9*sin(2*pi*GridFreq*Ts*n)+0.1*sin(2*pi*5*GridFreq*Ts*n);
 end
 for n=1:floor(L)
 u1(n)=sin(2*pi*GridFreq*Ts*n);
 end
%CASE 3 : Frequency Shift
 L=length(t);
 for n=1:floor(L)
 u(n)=sin(2*pi*GridFreq*Ts*n);
 end
 for n=1:floor(L)
 u1(n)=sin(2*pi*GridFreq*Ts*n);
 end
 for n=floor(L/2):L
 u(n)=sin(2*pi*GridFreq*1.1*Ts*n);
 end
%CASE 4: Amplitude Variations
 L=length(t);
 for n=1:floor(L)
 u(n)=sin(2*pi*GridFreq*Ts*n);
 end
 for n=1:floor(L)
 u1(n)=sin(2*pi*GridFreq*Ts*n);
 end
 for n=floor(L/2):L
 u(n)=0.8*sin(2*pi*GridFreq*Ts*n);
 end;
u=fi(u,T,F);
u1=fi(u1,T,F);
%declare arrays used by the PLL process
Upd=fi([0,0,0],T,F);
ynotch=fi([0,0,0],T,F);
ynotch_buff=fi([0,0,0],T,F);
ylf=fi([0,0],T,F);
SinGen=fi([0,0],T,F);
Plot_Var=fi([0,0],T,F);
Mysin=fi([0,0],T,F);
Mycos=fi([fi(1.0,T,F),fi(1.0,T,F)],T,F);
theta=fi([0,0],T,F);
werror=fi([0,0],T,F);
%notch filter design
c1=0.1;
c2=0.00001;
X=2*c2*wn*2*Ts;
Y=2*c1*wn*2*Ts;
Z=wn*2*wn*2*Ts*Ts;
B_notch=[1 (X-2) (-X+Z+1)];
A_notch=[1 (Y-2) (-Y+Z+1)];
B_notch=fi(B_notch,T,F);
A_notch=fi(A_notch,T,F);
% simulate the PLL process
for n=2:Tfinal/Ts % No of iteration of the PLL process in the
simulation time
% Phase Detect
Upd(1)= u(n)*Mycos(2);
%Notch Filter
ynotch(1)=-A_notch(2)*ynotch(2)-
A_notch(3)*ynotch(3)+B_notch(1)*Upd(1)+B_notch(2)*Upd(2)+B_notch(3)*
Upd(3);

%update the Upd array for future sample
Upd(3)=Upd(2);
Upd(2)=Upd(1);

% PI Loop Filter
%ts=30ms, damping ration = 0.7
% we get natural frequency = 110, Kp=166.6 and Ki=27755.55
% B0=166.877556 & B1=-166.322444
```

```
ylf(1)= fi(1.0,T,F)*ylf(2)+fi(166.877556,T,F)*ynotch(1)+fi(-
166.322444,T,F)*ynotch(2);
%update Ynotch for future use
ynotch(3)=ynotch(2);
ynotch(2)=ynotch(1);
ynotch_buff(n+1)=ynotch(1);
ylf(1)=min([ylf(1) fi(200.0,T,F)]);
ylf(2)=ylf(1);
wo=fi(wn,T,F)+ylf(1);
werror(n+1)=(wo-wn)*fi(0.00318309886,T,F);
%integration process
Mysin(1)=Mysin(2)+wo*fi(Ts,T,F)*(Mycos(2));
Mycos(1)=Mycos(2)-wo*fi(Ts,T,F)*(Mysin(2));
%limit the oscillator integrators
Mysin(1)=max([Mysin(1) fi(-1.0,T,F)]);
Mysin(1)=min([Mysin(1) fi(1.0,T,F)]);
Mycos(1)=max([Mycos(1) fi(-1.0,T,F)]);
Mycos(1)=min([Mycos(1) fi(1.0,T,F)]);
Mysin(2)=Mysin(1);
Mycos(2)=Mycos(1);
%update the output phase
theta(1)=theta(2)+wo*Ts;
%output phase reset condition
if(Mysin(1)>0 && Mysin(2) <=0)
theta(1)=-fi(pi,T,F);
end
SinGen(n+1)=Mycos(1);
Plot_Var(n+1)=Mysin(1);
end
% CASE 1 : Phase Jump at the Mid Point
error=Plot_Var-u;
%CASE 2 : Harmonics
%error=Plot_Var-u1;
%CASE 3: Frequency Variations
%error=Plot_Var-u;
%CASE 4: Amplitude Variations
%error=Plot_Var-u1;
figure;
subplot(3,1,1),plot(t,Plot_Var,'r',t,u,'b'),title('SPLL(red) & Ideal
Grid(blue)');
subplot(3,1,2),plot(t,error,'r'),title('Error');
subplot(3,1,3),plot(t,u1,'r',t,Plot_Var,'b'),title('SPLL Out(Blue) &
Ideal Grid(Red)');
```

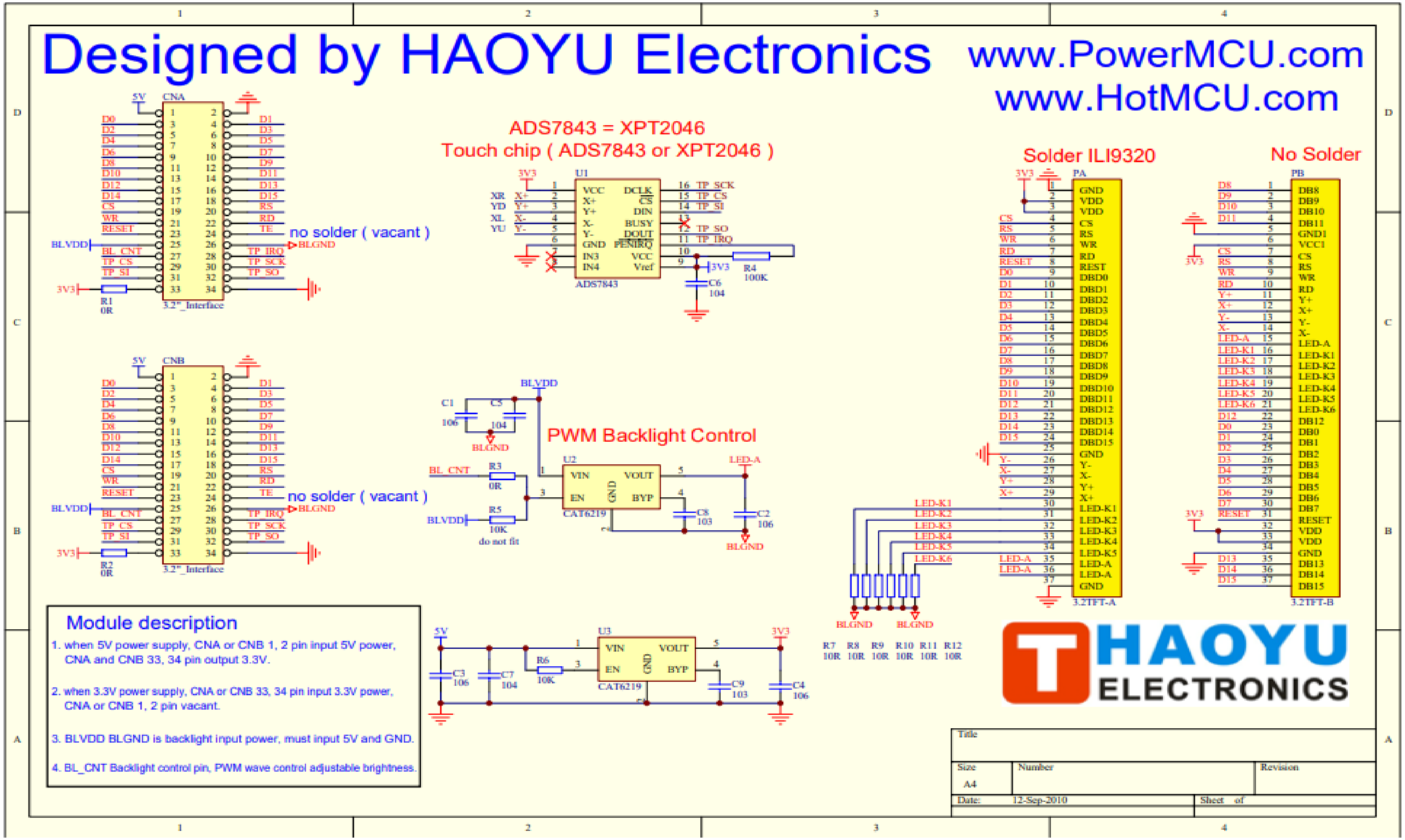
Appendix B: Simulating the PLL for Varying Conditions for OSG Filter

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PLL Simulating for OSG
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;
% define the math type being used on the controller using objects
from the fixed-point toolbox in MATLAB
%Select numeric type, let’s choose Q23
T=numerictype('WordLength',32,'FractionLength',23);
%Specify math attributes to the fimath object
F=fimath('RoundMode','floor','OverflowMode','wrap');
F.ProductMode='SpecifyPrecision';
F.ProductWordLength=32;
F.ProductFractionLength=23;
F.SumMode='SpecifyPrecision';
F.SumWordLength=32;
F.SumFractionLength=23;
%specify fipref object, to display warning in cases of overflow and
%underflow
P=fipref;
P.LoggingMode='on';
P.NumericTypeDisplay='none';
P.FimathDisplay='none';

%PLL Modelling starts from here
Fs=50000;        %Sampling frequency = 50Khz
GridFreq=50;     %Nominal Grid Frequency in Hz
Tfinal=0.2;      %Time the simulation is run for = 0.5 seconds
Ts=1/Fs;         %SamplingTime= 1/Fs
t=0:Ts:Tfinal;   %Simulation Time vector
wn=2*pi*GridFreq;    %Nominal Grid Frequency in radians
%declare arrays used by the PLL process
err=fi([0,0,0,0,0],T,F);
ylf=fi([0,0,0,0,0],T,F);
Mysin=fi([0,0,0,0,0],T,F);
Mycos=fi([1,1,1,1,1],T,F);
theta=fi([0,0,0,0,0],T,F);
dc_err=fi([0,0,0,0,0],T,F);
wo=fi(0,T,F);

% used for plotting
Plot_Var=fi([0,0,0,0],T,F);
Plot_theta=fi([0,0,0,0],T,F);
Plot_osgu=fi([0,0,0,0],T,F);
Plot_osgqu=fi([0,0,0,0],T,F);
Plot_D=fi([0,0,0,0],T,F);
Plot_Q=fi([0,0,0,0],T,F);
Plot_dc_err=fi([0,0,0,0,0],T,F);

%orthogonal signal generator
```

```
%using trapezoidal approximation
osg_k=0.5;
osg_x=2*osg_k*wn*Ts;
osg_y=(wn*wn*Ts*Ts);
osg_b0=osg_x/(osg_x+osg_y+4);
osg_b2=-1*osg_b0;
osg_a1=(2*(4-osg_y))/(osg_x+osg_y+4);
osg_a2=(osg_x-osg_y-4)/(osg_x+osg_y+4);

osg_qb0=(osg_k*osg_y)/(osg_x+osg_y+4);
osg_qb1=2*osg_qb0;
osg_qb2=osg_qb0;
osg_k=fi(osg_k,T,F);
osg_x=fi(osg_x,T,F);
osg_y=fi(osg_y,T,F);
osg_b0=fi(osg_b0,T,F);
osg_b2=fi(osg_b2,T,F);
osg_a1=fi(osg_a1,T,F);
osg_a2=fi(osg_a2,T,F);
osg_qb0=fi(osg_qb0,T,F);
osg_qb1=fi(osg_qb1,T,F);
osg_qb2=fi(osg_qb2,T,F);

osg_u=fi([0,0,0,0,0,0],T,F);
osg_qu=fi([0,0,0,0,0,0],T,F);

u_Q=fi([0,0,0],T,F);
u_D=fi([0,0,0],T,F);

%generate input signal
% CASE1 : Phase Jump at the Mid Point
L=length(t);
for n=1:floor(L)
u(n)=sin(2*pi*GridFreq*Ts*n);
end
for n=1:floor(L)
u1(n)=sin(2*pi*GridFreq*Ts*n);
end
for n=floor(L/2):L
u(n)=sin(2*pi*GridFreq*Ts*n+pi/2);
end
u=fi(u,T,F);
% simulate the PLL process
for n=3:Tfinal/Ts   % No of iteration of the PLL process in the
simulation time
%Orthogonal Signal Generator
osg_u(1)=(osg_b0*(u(n)-u(n-2)))+osg_a1*osg_u(2)+osg_a2*osg_u(3);
osg_u(3)=osg_u(2);
osg_u(2)=osg_u(1);

osg_qu(1)=(osg_qb0*u(n)+osg_qb1*u(n-1)+osg_qb2*u(n-
2))+osg_a1*osg_qu(2)+osg_a2*osg_qu(3);
osg_qu(3)=osg_qu(2);
osg_qu(2)=osg_qu(1);

%park transform from alpha beta to d-q axis
u_Q(1)=Mycos(2)*osg_u(1)+Mysin(2)*osg_qu(1);
u_D(1)=-Mysin(2)*osg_u(1)+Mycos(2)*osg_qu(1);

%Loop Filter
```

```
ylf(1)=fi(1,T,F)*ylf(2)+fi(166.877556,T,F)*u_Q(1)+fi(-
166.322444,T,F)*u_Q(2);
u_Q(2)=u_Q(1);
u_D(2)=u_D(1);

%Limit LF according to its Q? size pipeline
ylf(1)=max([ylf(1) fi(-128,T,F)]);
ylf(1)=min([ylf(1) fi(128,T,F)]);
ylf(2)=ylf(1);
%update output frequency
wo=GridFreq+ylf(1);

%update the output phase
theta(1)=theta(2)+wo*fi(Ts,T,F);

if(theta(1)>fi(1.0,T,F))theta(1)=fi(0,T,F);
end

theta(2)=theta(1);
Mysin(1)=sin(theta(1)*fi(2*pi,T,F));
Mycos(1)=cos(theta(1)*fi(2*pi,T,F));
Mysin(2)=Mysin(1);
Mycos(2)=Mycos(1);

Plot_theta(n+1)=theta(1);
Plot_osgu(n+1)=osg_u(1);
Plot_osgqu(n+1)=osg_qu(1);
Plot_Var(n+1)=Mysin(1);
Plot_D(n+1)=u_D(1);
Plot_Q(n+1)=u_Q(1);
end
% CASE1 : Phase Jump at the Mid Point
error=Plot_Var-u;

%CASE2 : Harmonics
%error=Plot_Var-u1;

%CASE3: Frequency Variations
%error=Plot_Var-u;

%CASE4: Amplitude Variations
%error=Plot_Var-u1;
subplot(3,1,1),plot(t,Plot_Var,'r',t,u,'b'),title('SPLL(red)&
IdealGrid(blue)');
subplot(3,1,2),plot(t,error,'r'),title('Error');
subplot(3,1,3),plot(t,u1,'r',t,Plot_Var,'b'),title('SPLLOut(Blue)&
IdealGrid(Red)');
```

Appendix C: Schematic of Human Hardware Interface HY32D



Module description

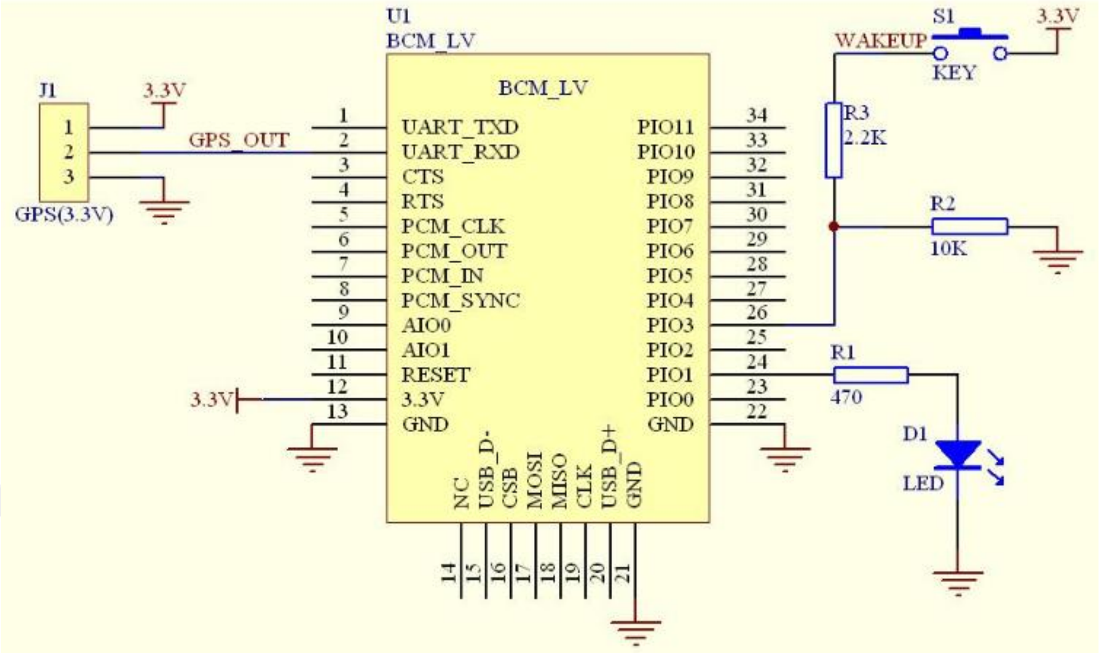
1. when 5V power supply, CNA or CNB 1, 2 pin input 5V power, CNA and CNB 33, 34 pin output 3.3V.

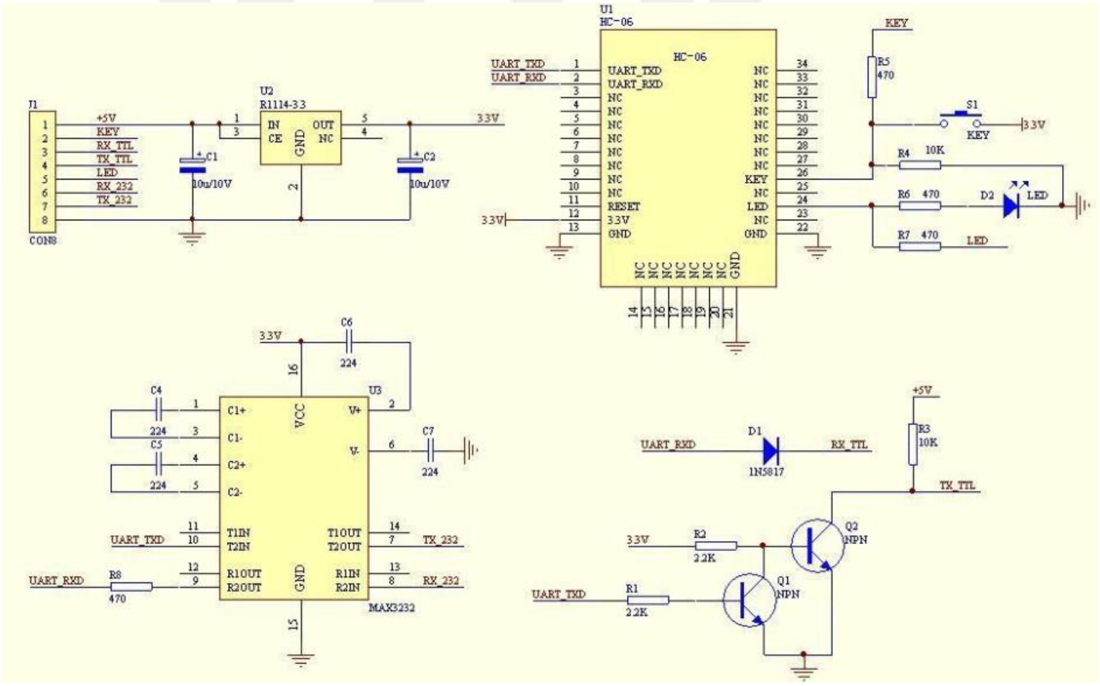
2. when 3.3V power supply, CNA or CNB 33, 34 pin input 3.3V power, CNA or CNB 1, 2 pin vacant.

3. BLVDD BLGND is backlight input power, must input 5V and GND.

4. BL_CNT Backlight control pin, PWM wave control adjustable brightness.

Appendix D: Schematic of HC-06 Bluetooth Device





Appendix E: Real-time Operating System Message Queue Service Codes

E.1 Service Function of the System

```
void KernelRun(void)
{
 for(;;)
 {
if(stMessageQueue.Counter)
  {
mRunLedOn();
KernelMessagePop();
DoProcess();
mRunLedOff();
  }
 }
}
```

E.2 Message Sending Function to the System

```
void KernelMessagePush(eTasks_t TargetTask, uint16_t
TargetTaskMessageID, uint16_t TargetTaskMessageData)
 {
stMessageQueue.Objects[stMessageQueue.WriteIndex].TargetTask =
TargetTask;

stMessageQueue.Objects[stMessageQueue.WriteIndex].TargetTaskMessage
ID = TargetTaskMessageID;
stMessageQueue.Objects[stMessageQueue.WriteIndex].TargetTaskMessage
Data = TargetTaskMessageData;
if(++stMessageQueue.WriteIndex == MESSAGE_QUEUE_SIZE)
 {
stMessageQueue.WriteIndex = 0;
 }
if(++stMessageQueue.Counter == MESSAGE_QUEUE_SIZE)
 {
stMessageQueue.Counter = 0;
 }
}
```

E.3 Message Requesting Function from the System

```
void KernelMessagePop(void)
 {
stMessageQueue.CurrentObject.TargetTask =
stMessageQueue.Objects[stMessageQueue.ReadIndex].TargetTask;
stMessageQueue.CurrentObject.TargetTaskMessageID =
stMessageQueue.Objects[stMessageQueue.ReadIndex].TargetTaskMessageID
;
stMessageQueue.CurrentObject.TargetTaskMessageData =
stMessageQueue.Objects[stMessageQueue.ReadIndex].TargetTaskMessageDa
ta;
  if(++stMessageQueue.ReadIndex == MESSAGE_QUEUE_SIZE)
 {
  stMessageQueue.ReadIndex = 0;
 }
  --stMessageQueue.Counter;
}
```

Appendix F: Cos(φ) Calculating Function by the MCU

```
for (i = 0; i < SAMPLE_MAX; i++)
 {

for (j = 0; j < N; j++)
{
An[j] += (2/FFT_T)* Ts * (float)stTaskMeas.CopyBuffer[0][i] *
arm_cos_f32((j+1)*FFT_w*(FFT_Ts*i));
Bn[j] += (2/FFT_T)* Ts * (float)stTaskMeas.CopyBuffer[0][i] *
arm_sin_f32((j+1)*FFT_w*(FFT_Ts*i));
    }
 }
Phi_V = (float32_t) atan(An[0]/Bn[0]);
ClearFFTCoeff();

for (i = 0; i < SAMPLE_MAX; i++) {
for (j = 0; j < N; j++) {
An[j] += (2/FFT_T)* Ts * (float)stTaskMeas.CopyBuffer[0][i] *
arm_cos_f32((j+1)*FFT_w*(FFT_Ts*i));
Bn[j] += (2/FFT_T)* Ts * (float)stTaskMeas.CopyBuffer[0][i] *
arm_sin_f32((j+1)*FFT_w*(FFT_Ts*i));
}
 }

Phi_I = (float32_t) atan(An[0]/Bn[0]);
Phi = Phi_V - Phi_I;
ClearFFTCoeff();
stTaskMeas.stLineParamsRaw[stTaskMeas.RawIndex].CosPhi =
arm_cos_f32(Phi);
```

Appendix G: Power Factor Calculating Function by the MCU

```
for (s = 0; s < N; s++)
{
arm_sqrt_f32(((An[s]*An[s] + Bn[s]*Bn[s])/2) , &Cn[s]);
Temp += Cn[s] * Cn[s];
}

arm_sqrt_f32(Temp, &Temp);
Kv = Cn[0]/Temp;
for (m = 0; m < N; m++)
{
arm_sqrt_f32(((An[m]*An[m] + Bn[m]*Bn[m])/2) , &Cn[m]);
Temp += Cn[m] * Cn[m];
}

arm_sqrt_f32(Temp, &Temp);
Ki = Cn[0]/Temp;

stTaskMeas.stLineParamsRaw[stTaskMeas.RawIndex].PowerFactor = Kv *
Ki * stTaskMeas.stLineParamsRaw[stTaskMeas.RawIndex].CosPhi;
```

Appendix H: MODBUS Communication Address Mapping:

Address (Decimal)	Address Read / Write	Type:	Definition:	Gain:
0	R	U16	Voltage	10
1	R	U16	Current	1000
2	R	S16	Cos($\boldsymbol{\varphi}$)	1000
3	R	S16	Power Factor	1000
4	R	U16	Frequency	100
5	R	U16	THD$_V$	100
6	R	U16	THD$_I$	100
7	R	S16	Active Power	1000
8	R	S16	Reactive Power	1000
9	R	U16	Apparent Power	1000
10	R	U32	Active Power (Import)	10
12	R	U32	Inductive Reactive Power (Import)	10
14	R	U32	Capacitive Reactive Power (Import)	10
16	R	U32	Active Power (Export)	10
18	R	U32	Inductive Reactive Power (Export)	10
20	R	U32	Capacitive Reactive Power (Export)	10

Curriculum Vitae

PERSONAL INFORMATION

Name and Surname : Uğur POLAT

EDUCATION

Degree	Graduate School	Year
Master of Science	Gaziantep University, Gaziantep	2022
Bachelor of Science	Kahramanmaraş Sütçü İmam University, Kahramanmaraş	2019
High School	Opet Anatolian High School, Gaziantep	2013

PUBLICATIONS

Uğur, P., Ergün, E., (2021). ARM Based Development of Embedded System for an Energy and Harmonic Analyzer, Gaziantep University, Manchester Journal of Artificial Intelligence & Applied Sciences, **2(1). 2021**, ISSN: 2634-1034.