WIRELESS MODBUS
FOR SWITCHGEAR APPLICATIONS

by

MUHAMMAD NUR IMAN BIN ZAHARI

Final Report

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfilment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Wireless MODBUS for Switchgear Applications

by

Muhammad Nur Iman Bin Zahari

A project dissertation submitted to the
Electrical and Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL AND ELECTRONICS ENGINEERING)

Approved by,

____
(Ir. Mohd. Faris Bin Abdullah)
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

DECEMBER 2010

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

MUHAMMAD NUR IMAN BIN ZAHARI

ABSTRACT

In the power industry, switchgears are normal to be used at the distribution side. However, problems can arise when the time comes to operate and maintain the switchgear. Switchgear can be in high voltage (HV), medium voltage (MV) and low voltage (LV) category. Regardless of their voltage classifications, for most MV and LV switchgear, switching would be done at the front of the circuit breakers. Potential explosion hazards at the time of switching may pose danger particularly to precious human capital. Although SCADA systems are widely available, many or majority of substations and are not equipped with SCADA. Considerations on the space capacity, cost of control panels, servers and the appropriate wiring made it realized only in big and critical systems governing a wide area. Thus, the purpose of the project is to save human lives on majority substations by creating an interface between the switchgears and a portable controlling and data monitoring system i.e. pocket pc, laptop that is entirely cheap with addition to mass deployment.

ACKNOWLEDGEMENT

First and foremost, my utmost appreciation to Allah the Almighty, for with his blessings and guidance, I am able to successfully complete my Final Year Project (FYP) I and II without any harm and sharp pitfalls. I would like to express my earnest gratitude to my supervisor Ir. Mohd. Faris Abdullah for his guidance and ideas in assisting me completing the project.

Special acknowledgements are due to:

i. Bruce Durdle, B.Eng. (Electrical), M.Eng; MIPENZ ; MIET; Chartered Engineer (UK). Academic Staff, Waikato Institute of Technology, New Zealand.

ii. Abdul Wahid Jaafar, B.Eng (Electrical), Senior Executive, Facilities Engineering Department, PETRONAS Carigali, KLCC.

With both is and had worked in the industrial world and acquired the nits and bits of electrical and electronics hardware and standards which is what is required to make my project successful. Sorry if I had disturbed while you guys are working.

Cordial appreciation is given to Universiti Teknologi PETRONAS (UTP) and Electrical and Electronics Engineering Department for equipping students with facilities and guidelines in doing the FYP.

Finally, my acknowledgement is considered incomplete without thanking my roommate, all my fellow colleagues, friends and family who have been giving me great support and encouragement for me to complete this project.

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MCC	Motor Control Center
LVS	Low Voltage Switchgear
LV	Low Voltage
MV	Medium Voltage
HV	High Voltage
MB	Modbus
RTU	Remote Terminal Unit
ASCII	American Standard Code for Information Interchange
MCB	Miniature Circuit Breaker
MCCB	Moulded Case Circuit Breaker
PIC	Programmable Interface Controller
CPU	Central Processing Unit
MCU	Microcontroller Unit
SCADA	Supervisory Control and Data Acquisition
HMI	Human/Machine Interface
RF	Radio Frequency
CRC	Cyclic Redundancy Check
LRC	Longitudinal Redundancy Check

CHAPTER 1
INTRODUCTION

1.1 Background of Study

The study involves distribution part of power system in control, instrumentations and communications systems. Studies conducted mainly revolve around power distribution, protection, control, communication protocols, data acquisition, microcontrollers, power electronics, serial interfaces, transceivers, C programming, Cicode programming, current and voltage sensing and HMI.

1.2 Problem Statement

i. In any substation, switching of circuit breakers at the front of it manually could produce explosive hazard to the operators.
ii. Full SCADA system is costly and often would not be even considered in the design process of small/medium sized substations.
iii. Current remote switching in small substations does not support wireless switching – it must be in the line of sight (LOS) of the operator.
iv. Remote switching is isolated and wiring of remote control has to be done one by one
v. Control of circuit breaker is not portable and is fixed at one position.

1.2.1 Problem Identification

The problem identified when the author was doing internship in the design department of an oil and gas company. High voltage machines and equipments can incur not only electrical hazards but also explosive hazards.

According to Malaysia Department of Occupational Safety and Health, taking a look in the utility sector from 2007 till 2009, the number of reported accidents increases from 65 in 2007 to 106 in 2009. By introducing low cost and affordable SCADA in the majority of small substations in the country, the accidents of switching circuit breaker and working in front of switchgears can be reduced, thus minimizing casualties.

1.2.2 Significance of Project

The aim of this project is novelty. If the project succeeds, all substations can be equipped with a device that can control and monitor their switchgears which are affordable. This is then can lead to a lower casualties caused by equipments failure accidents.

1.3 Objectives

The ultimate objective of this project is to produce a working prototype of a wireless MODBUS connecting a portable computer i.e. laptop/pocket pc to circuit breakers/contactors on switchgears.

1.4 Scope of Study

The scope of this project would involve the followings:

i. Power systems engineering

The practical side of power distribution, MV switchgear, circuit breakers, transducers and power electronics.

ii. Control systems/instrumentation engineering

This is inclusive of the HMI, SCADA, inputs and outputs (current and voltage sensing) and methods of controlling variables and data (data protocols).

iii. Communication systems engineering

Data are transmitted over wireless transceivers. Proper knowledge upon the bandwidth, frequency to be used, modulation method, error detection and data methods is vital for the communication.

iv. Computer systems

To link all of the above, a proper computer or microcontroller hardware and software design is needed. The hardware requirement (memory capacity, speed, analog to digital capability, samplings and other functions) must be properly considered so as not to affect the capability of the whole system itself.

1.4.1 The Relevancy of the Project

Although it is a mix instrumentation and control, this project is in line with the Power and Energy cluster. In reality, everything is a mix of something but the main thing require in innovation is the main spotlight. Among all, the many benefit of the project includes:

i. Greatly reduce the need for control cables
ii. Provide easier commissioning and maintenance of switchgears
iii. Reduce cost of a switchgear – cost of miles of copper and ex-rated cables, which is expensive
iv. Less wiring means less maintenance
v. Wireless supervision of switchgears, trip or running
vi. A cheap alternative for an n+1 system
vii. Less faults because of less wires
viii. Wireless monitoring of electrical elements in the field (switchgears, pumps, motors, etc)
ix. Faster fabrication and commissioning of switchgears related components

1.4.2 Feasibility of the Project within the Scope and Time Frame

Given the amount of time of about 5 months to complete the designing and planning (FYP 1) and about another 5 month for the process of fabrication, construction of the prototype and further improvement on the prototype, the author budgeted that it can really be done.

Even though the FYPs do not carry that much of a mark, the amount of time that needed to be allocated to the project is quite enormous. While attending classes, doing assignments, tutorials, lab sessions, quizzes, tests, exams and lots more, the project’s time is slipped between those sessions.

CHAPTER 2
LITERATURE REVIEW AND THEORY

2.1 Classifications of Switchgear

Switchgears voltage level are in three forms; HV, MV and LV. All of this is in the distribution side of electrical power system [2]. Power distribution is only one part of delivering electrical power. It is a process after transmission and the customer.

Figure 1: A typical generation, transmission and distribution in electrical power system

Switchgear is required to enable power sources to be connected to and disconnected from a voltage distribution system [1]. This switching is necessary for normal operational purposes and also for the rapid and automatic disconnection of any circuit that becomes faulty.

In addition, switchgears allow any circuit to be isolated from the live system and for that circuit to be made safe so that work can be carried out to the equipment connected to it.

2.1.1 Types of Switchgear

2.1.1.1 Circuit Breakers

Figure 2: Types of LVS[1]

They are used to transformers, section breakers on switchboard sections, inter-connectors between switchboards and auxiliary generators. Circuit breakers can be of the air-break, vacuum and gas-insulated types.

2.1.1.2 Moulded Case and Miniature Circuit Breakers

They are special class of lightweight compact circuit breakers for mounting onto or behind panels [1]. They are designed for hand operation, but have built-in protective tripping arrangements. These are typical in LVS.

2.1.2 Switchgear Operations - LVS

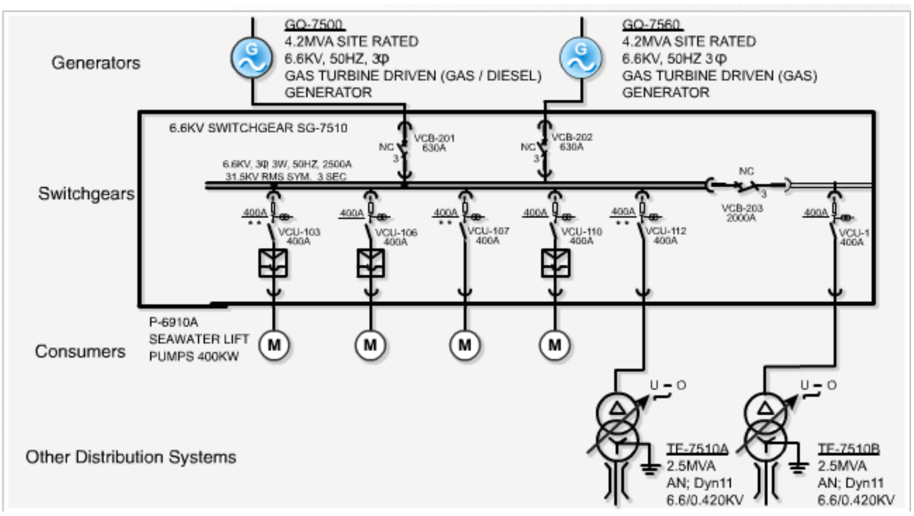


Figure 3: A single line diagram of a LVS system[1]

An LV switchboard is usually supplied from one or two step down transformers fed from the HV system. LV operating voltages are normally 415V. On the diagram, it shows an onshore LV system with two transformer incomers, a bus section breaker, a heavy feeder and two group distributing sections; left and right, also called motor control sections or MCC.

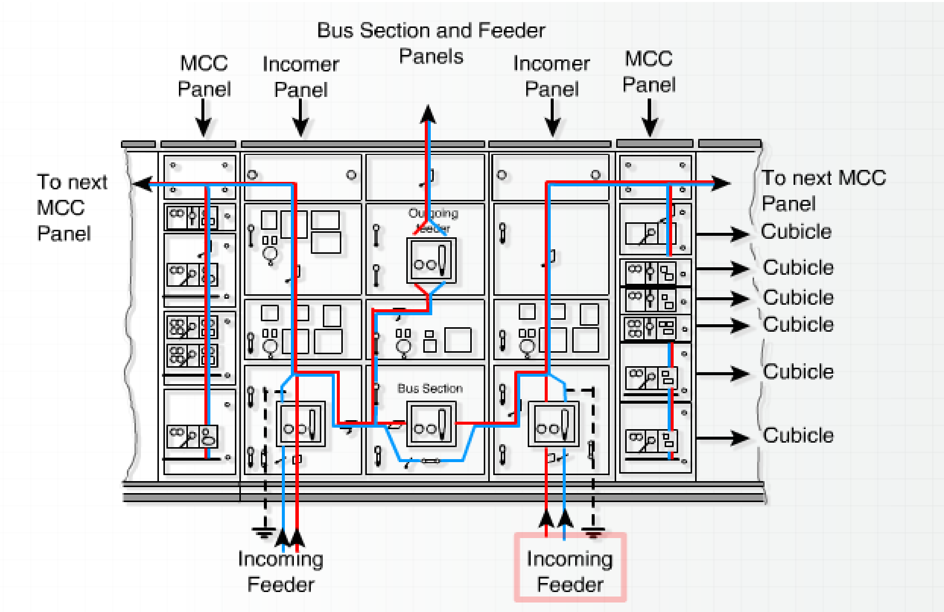


Figure 4: Basic in and out of a LVS cubicle[1]

The circuit breakers usually form the centre section with power being passed to the left and to the right by the busbars. Heavy current feeders and the larger interconnectors feeding power to and receiving power from other LV switchboards sometimes require circuit breaker protection. They are then brought into the centre section.

The centre three panels contain:

i. Cubicles for incoming feeders
ii. Bus sections
iii. Heavy feeder circuit breakers
iv. Associated protective relays
v. Control switches
vi. Indication equipment mounted in front of each panel

MCC panels can be seen on either side of the centre panels. Additional MCC panels are added to house the feeder cubicles necessary to meet the requirements of the systems concerned.

Each MCC panel contains a number of motor control contactor cubicles and fused switch cubicles mounted one above the other to control the outgoing circuits. The fused switch cubicles control those circuits not associated with motors such as sub-distribution boards or welding sockets.

A large switchboard may include as many 30 or more MCC panels. The arrangement of the busbars and circuit connections is shown diagrammatically in color. The main busbars are shown in red for the phases and blue for neutral. Power is supplied to each outgoing feeder cubicles by a set of dropping busbars housed in a vertical enclosure at the rear of each MCC panel.

2.2 MODBUS Protocol

Figure 5: Modbus logos, old and new (left to right)

The MODBUS protocol or simply MB has been used since the 1979's. It was developed by Modicon for the use of Programmable Logic Controllers for industrial communication. MB has since become the reference and standard for any industrial processes system and control[6].

2.2.1 Benefits

Although MB is quite old (more than 30 years old now) or can be said a legacy protocol, but because of its extensive coverage of the concept of communication and covers all the basic and some advanced functions, the people in the industry finds it unnecessary to equip their equipment with newer protocols which also functions the same but they think that they did not need the extra functions. They just need something that can simply work to control their processes.

Among the benefits that people found it useful about MB is that:

i. Openly published and royalty-free[6]
ii. Easy industrial network to deploy[6]
iii. Its protocols are easy to be integrated between vendors that uses different hardwares

2.2.2 The Protocol

Protocols are laws. However, in communication systems, it is more like rules that the system must abide. Laws when are not followed will invite legal action and so on. But rules are not that strict as a law. In communication system, a set of rules are called protocols. And in one system, it should have not more than one protocol. This is as it would create problem later on regarding hardware and synchronization.

In MODBUS, the protocol follows that of a Master – Slave protocol[3]. A master slave type system has one node (the master node) that issues explicit commands to one of the “slave” nodes and processes responses.

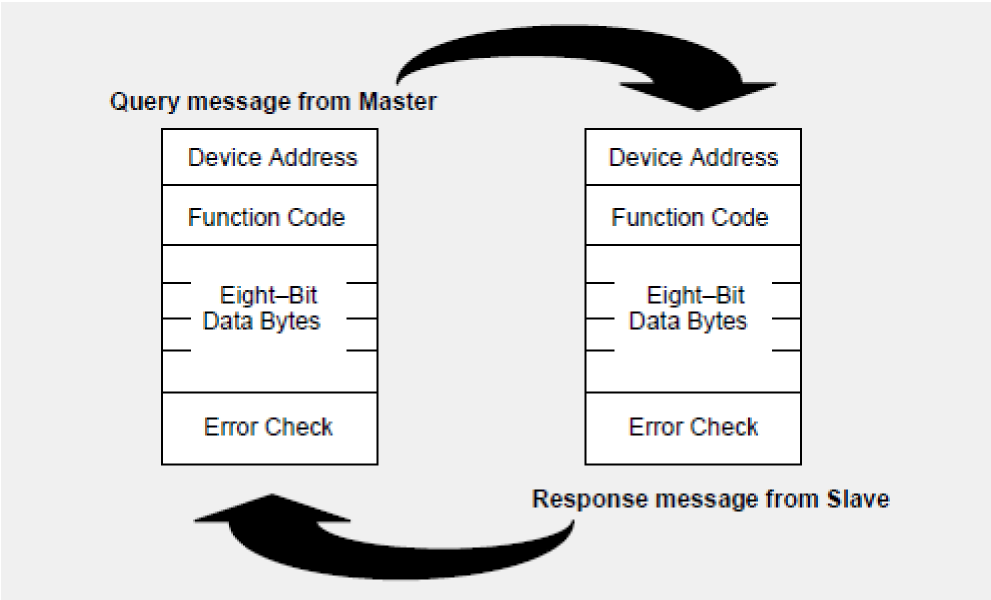


Figure 6: Master to slave protocol layers

Slave nodes will not typically transmit data without a request from the master node, and do not communicate with other slaves[3].

At the physical level, MODBUS over Serial Line systems may use different physical interfaces (RS485, RS232). TIA/EIA-485 (RS485) Two-Wire interface is the most common[2].

The followings shows a general overview of the MODBUS serial layer compared to the basic OSI model.

Layer	ISO/OSI Model	
7	Application	MODBUS Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	MODBUS Serial Line Protocol
1	Physical	EIA/TIA-485 (or EIA/TIA-232)

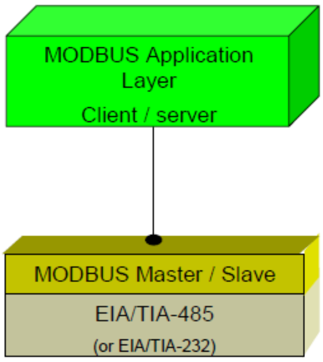


Figure 7: MODBUS protocols and ISO/OSI model[2]

2.2.3 Modes

Users who intend to use serial transmission in Modbus may choose two types of modes that is ASCII or RTU[3]. The mode defines the bit contents of message fields transmitted serially on the line. It determines how information is packed into the message fields and decoded[2].

Special attention must be paid that the transmission mode must be the same for all devices. However, the default setup must be the RTU[3].

2.2.3.1 RTU

RTU or Remote Terminal Unit is currently the most widely used serial transmission mode in the industry. This is because, this mode has greater character density which allows better data throughput than ASCII mode for the same baud rate.

For each byte in RTU mode is [4]:

Coding System: 8-bit binary

Bits per Byte : 1 start bit

8 data bits, least significant bit sent first

1 bit for parity completion

1 stop bit

Even parity is required for RTU. But to make sure compatibility with various hardwares, it is recommended to support No parity mode.

The characters are transmitted serially like below:

With Parity Checking

Start	1	2	3	4	5	6	7	8	Par	Stop

Figure 8: Bit sequence in RTU mode

Each character or byte is sent in this order (left to right) where the Least Significant Bit (LSB) is first bit and Most Significant Bit (MSB) is the last.

There are Even, Odd or No Parity checking. If No Parity is implemented, an additional stop bit is transmitted to fill out the character frame to a full 11-bit asynchronous character:

Without Parity Checking

Start	1	2	3	4	5	6	7	8	Stop	Stop

Figure 9: Bit sequence in RTU mode (specific case of No Parity) [4]

In all, the RTU summarizes as the frame description underneath:

Slave Address	Function Code	Data	CRC	
1 byte	1 byte	0 up to 252 byte(s)	2 bytes	
			CRC Low	CRC Hi

Figure 10: RTU message frame [4]

CRC is a frame checking field that must be included in the frame. This is to avoid any missing, invalid and wrong data [4]. For more about CRC, please refer Appendix IV.

2.2.3.2 ASCII

American Standard Code for Information Interchange or ASCII is used when RTU mode cannot be used which is normally caused by timers management problems. This mode is less efficient than RTU because each byte that it carries needs two more characters [3].

For an example, the byte 0x5B is encoded as 0x35 and 0x42 (0x35 = “5”, and 0x42 = “B” in ASCII.

For each byte in ASCII mode is [4]:

Coding System: Hexadecimal, ASCII characters 0-9, A-F
One hexadecimal character contains 4-bits of data within each ASCII character of the message

Bits per Byte : 1 start bit
7 data bits, least significant bit sent first
1 bit for parity completion
1 stop bit

Like RTU, it is recommended to go for No Parity mode for maximum compatibility with other products.

The bit sequence in ASCII is the same as RTU with the exception of bit 8 [3]. This means ASCII has the start bit, 1-7 bit, parity and stop bits.

Start	Address	Function	Data	LRC	End
1 char :	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

Figure 11: ASCII message frame [4]

Apart from that, ASCII mode uses LRC (Longitudinal Redundancy Check) instead of CRC [4]. For more about LRC, please refer Appendix V.

2.2.4 MODBUS Functions

Modbus supports a variety of basic functions (switch coil on/off, read memory) and also some utility functions (diagnostic etc). The list of functions is as below:

Table 1: Modbus functions [4]

No.	Functions	No.	Functions
1.	01 Read Coil Status	13	Program Controller
2.	02 Read Input Status	14	14 Poll Controller
3.	03 Read Holding Registers	15	15 Force Multiple Coils
4.	04 Read Input Registers	16	16 Preset Multiple Registers
5.	05 Force Single Coil	17	17 Report Slave ID
6.	06 Preset Single Register	18	18 Program 884/M84
7.	07 Read Exception Status	19	19 Reset Comm. Link
8.	08 Diagnostics	20	20 Read General Reference
9.	09 Program 484	21	21 Write General Reference
10.	10 Poll 484	22	22 Mask Write 4X Register
11.	11 Fetch Communication Event Counter	23	23 Read/Write 4X Registers
12.	12 Fetch Communication Event Log	24	24 Read FIFO Queue

Although there are 24 functions and much more sub-functions regarding this, only a few will be selected to be used in this project. This is to avoid energy and time be used on complex but unnecessary functions for the small scope of this project. The functions that will be used is described in the results and discussion part of this report.

2.3 Microchip PIC Microcontroller

The selection of this microcontroller was done even though there is a wide range of microcontrollers is because the author has experience using that particular chip.

2.3.1 Preface to Microcontrollers

Microcontrollers are among the many indigenous inventions for the past 4 decades. The use of microcontrollers became more apparent in the last two decades when it is embedded into small electronic devices and appliances such as handphones, amplifiers, household appliances and much more.

Before microcontrollers were invented, there is the microprocessor. Microprocessors are a big electronic component specialized in crunching raw data and algorithms. Other essential components such as the memory, input and output devices, graphics card and so on are added separately. This makes the whole system be bigger and also it consumes more power [5].

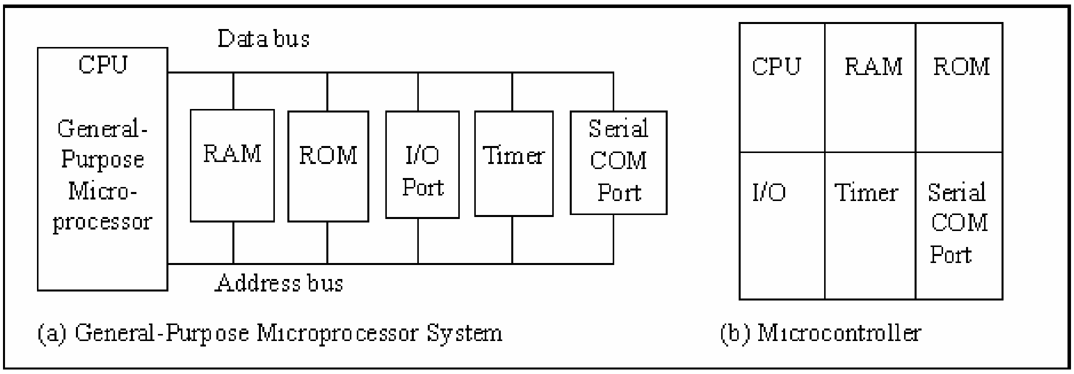


Figure 12: Comparison of a microprocessor system and a microcontroller

2.3.2 PIC 16F877A

Besides being cheap, this model of microcontroller from Microchip can provide all of the basic to medium level of capacity to support any small to medium sized systems.

Figure 13: Size of different microcontrollers compared to a dime

Table 2: Features of PIC 16F877A [5]

No.	Functions	No.	Functions
1.	40 pin DIP package	7.	10 bit multichannel ADC
2.	8K word (14bit) program memory (13 bit PIC)	8.	Built-in USART
3.	368 bytes of RAM space	9.	Built-in Synchronous Serial Port with SPI and I2C
4.	256 bytes of EEPROM space	10.	5 I/O Ports (A,B,C,D,E)
5.	8/16 bit timers	11.	35 Instruction sets
6.	2 capture and compare PWM modules		

2.3.2.1 Physical Pin Out of 16F877A

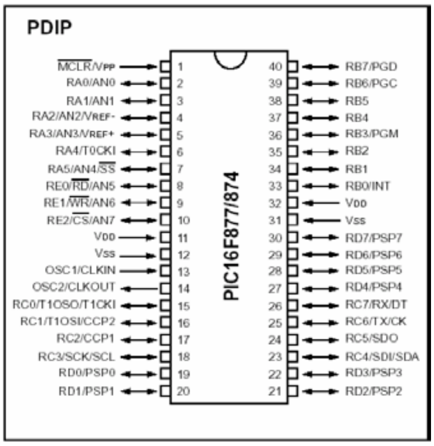


Figure 14: The pin out of PIC 16F877A [5]

For more information of the pins and detailed data for the functions of this microcontroller, please refer to the datasheet.

2.4 Wireless RF Transceiver

The wireless transmission could be done in a wide range of mediums. Among that are considered are RF, infrared, microwave, GSM, Bluetooth, Wi-FI, Zigbee and so on. However, the Wireless RF Transceiver 431-478 Mhz GFSK Data Transfer modules from Sure Electronics is used because of its simplicity and robustness.

The transceiver uses a band between 431-487 Mhz which is good when there are obstructions between the transmitter and receiver. Normally, high frequency RF are used in range of 2.4Ghz or so, but the range is limited to only a few meters. With this cheap and low power transceiver from Sure Electronics, the range can be up to 1000m.

Figure 15: Wireless RF Transceiver 431-478 Mhz GFSK [12]

2.4.1 Features

The half-duplex wireless data transceiver is integrated with an ultra speed MCU and a powerful RF chip. Because of its price, slim size, wide power supply range and ultra long transmission distance it is chosen as the communication medium for the project.

Among other features of the transceiver is:

i. Ultra long transmission distance: 800-1000 meters@1200bps
ii. Working frequency: 431-478Mhz (1kHz step)
iii. Over 100 channels
iv. GFSK modulation
v. Cyclic interleaving error correction
vi. Configurable RFID
vii. Three interfaces: UART/TTL, RS485, RS232
viii. Built-in watchdog for long term operation

One of the innovative features about the device is that, it is configurable. The configuration software is included to accommodate the device. Even if it is not configured, the transceivers can be used as it is.

2.4.1.1 RF Magic

RF Magic is the name of the software to configure the settings of the transceivers. Before configuring the item, it is good to know what parameter settings are supported by the module.

Table 3: The supported settings

Settings	**Options**	**Default**
Series Rate	1200, 2400, 4800, 9600bps	9600bps
Series Parity	Disable, Even, Odd	Disable
RS485/RS232	RS485, RS232	RS485
RFID Disable	Disable, Enable	Disable
RFID Index	0-65535 (16 bits)	12345
RF Frequency	431MHz-478MHz (1k step, accuracy ±100Hz)	434MHz
Airborne Rate	1200, 2400, 4800, 9600bps	9600bps
Frequency Deviation	5.4, 10.8, 21.6, 43.2, 86.4kHz	21.6kHz
RF Power	1-10 (10 indicates 20mW)	10 (20mW)

Proper wirings are needed to be done to the module for the software be able to read/write the configurations into its chip.

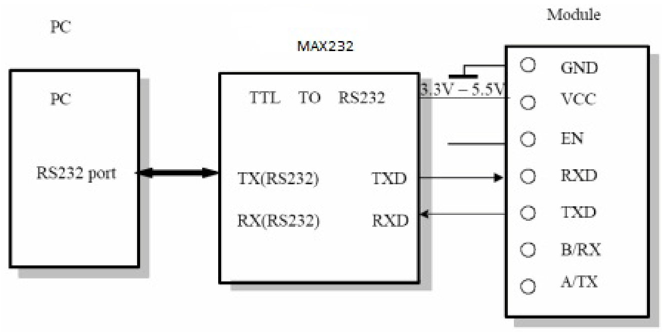


Figure 16: Hardware connection for RS232 [12]

After the wirings like the above has been done, only then the software can be run. Settings can then be picked from the drop down button. After proper adjustments are made, the write button is pressed to save the setting to the chip on the module.

Settings picked must be checked and double checked at all times when making adjustments. Without knowing and having any knowledge of the data transfer would make the synchronization between the module and connection interface fail to communicate properly.

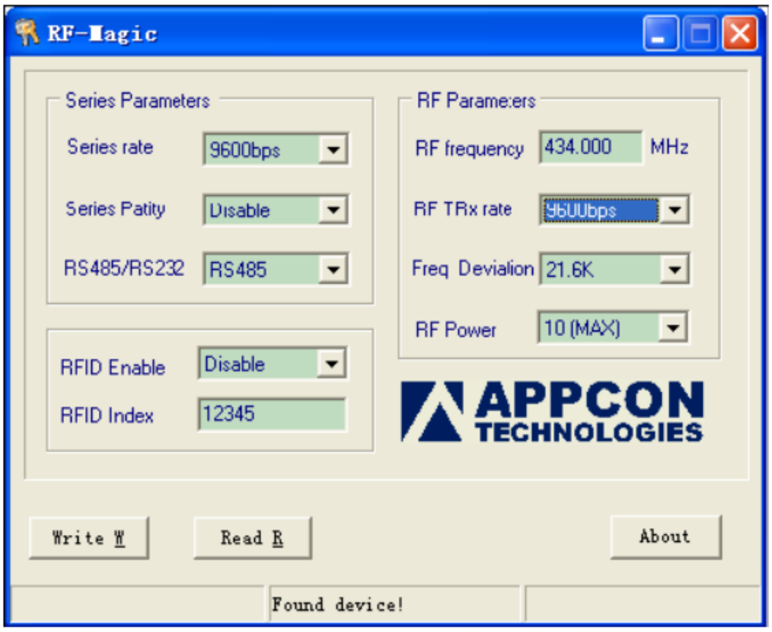


Figure 17: Programming the wireless module

2.4.2 Pin Definition

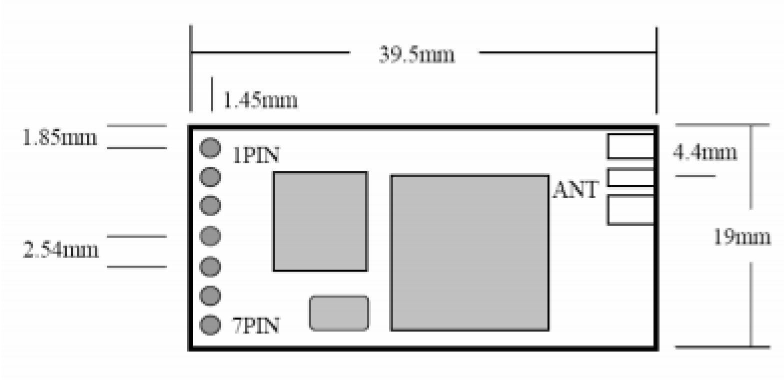


Figure 18: Physical layout of the module

For transmitting and receiving data properly, the right pins have to be connected to the right inputs.

Table 4: The wireless module pin out

Pins	Definition	Description
1	GND	0V
2	VCC	3.3V – 5.5V
3	EN	POWER ENABLE (>1.6V) or SUSPENDED ENABLE (<0.5V Hibernation)
4	RXD	UART input, TTL
5	TXD	UART output, TTL
6	B/RX	RS485- or RS232 RX
7	A/TX	RS485+ or RS232 TX

For the purpose of this project, pins number 4 and 5 will be used.

2.5 Serial Interface

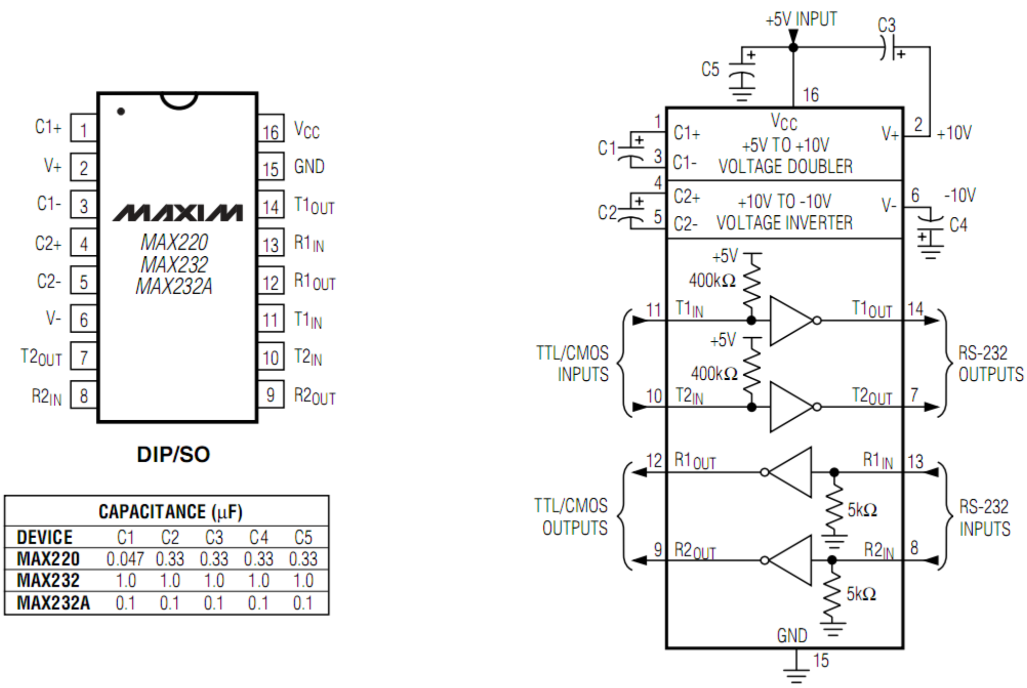
Serial interface are common to be used in data transfers for computers. Often, it is used to connect printers, handhelds and other peripheral devices. In this project, the serial interface will be used to connect between the computer and the wireless transceiver. The serial interface has been used 48 years since its first introduction.

It is connected from ports on a computer to other devices. Normally, designers would pick the 9 pins serial port or DB9 instead of its larger predecessor of DB16 and DB25.

2.5.1 Maxim MAX232

MAX232 is the chipset that is used to establish the serial connection. This chip converts the direct connection from a serial port to TTL, signal that can be understood by the microcontrollers. The incoming connection from the serial port is quite high, thus the chipset is used to regulate the 25V voltage to the source and also downsize 25V into 5V used by the PIC.

There are other type of chipset that can be used. For the basic connection, let it be the MAX232. Changing to the newer type of Ethernet connection, RS485 is not worth it because we are not using the extra function of networking in this project. Maybe, for later improvements, RS485 will be considered.

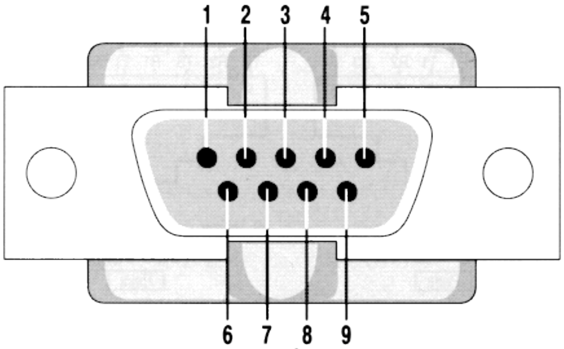


CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 19: MAX 232 and how connections should be made

2.5.2 RS232 Port (DB9)

DB9 is considered to be used in this project simply since there are simply 3 connections that are needed (despite the fact that it has 9 pins) to establish a communication.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Figure 20: DB-9 pin

For a simple operation, only pin 2,3 and 5 are needed to be wired. This DB9 port is connected from a computer to its MAX232 counterpart and then the PIC.

2.5.3 USB-RS232 Converter

Figure 21: A USB to RS232 converter

In modern laptops, RS232 are already phased out. A lot of application now uses USB instead of RS232. This is since USB is faster. As of that, a converter got to be used to convert USB ports to RS232 port. It is quite simple as we have to do no more than installing the driver and plug in the socket.

2.6 Printed Circuit Board (PCB)

After the hardware and software have been done, proper presentation of the electronics is needed to be done. For such purposes, a PCB design and fabrication would be nice. It is to avoid wear and tear to the electronics, to fit in a suitable housing and so on.

Among the exotic and necessary electronic components to supplement the microcontroller and the whole projects are listed as below:

2.7 Hardware Protection

For protecting the circuit from dangerous high voltages from connected loads, it is advisable to put an inhibitor to physically separate parts of high voltage and low voltage of the circuit.

2.7.1 Optocoupler

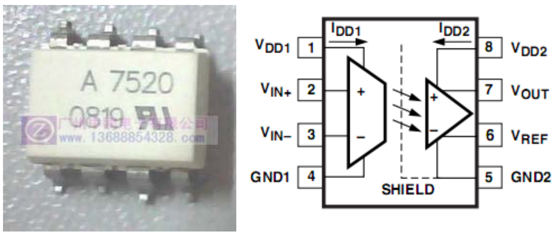


Figure 22: HCPL-7520 and its pin out [10]

The component in mind would be the HCPL-7520. It is safe and stable. It can withstand input voltage up to 3150V. Unfortunately, it is quite expensive at RM31 for one unit. In this project, 8 are required for 4 devices. Which if used, the isolation function will use up to half of the cost of the project budget.

Because of that, cheap low cost optoisolator which has been used for the industry has been identified to be used in this project. It is selected based on availability (widely used and available in any electronics shop) and cost, which is 30 times cheaper than the HCPL-7520.

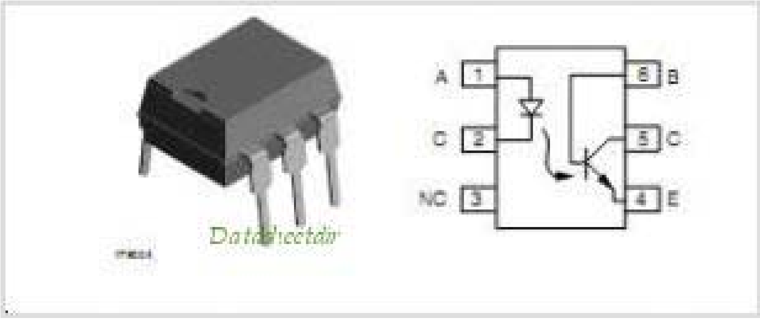


Figure 23: 4N35 pin outs [17]

The 4N35 optoisolator although is quite hard to be used since of its simplicity, is an oxymoron. It is 2 pins less than the HCPL-7520, and because of that, proper and special consideration got to be maintained in designing it to be used for 5V and 20mA voltage and current measurement/sensing.

2.7.2 Voltage Regulator

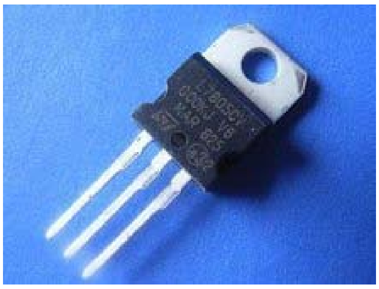


Figure 24: A L7805 voltage regulator [18]

This component would be used at the entry of the whole circuit. Clean voltage can be brought in without surges or spikes which can damage the sensitive PIC. A L7805 voltage regulator is used for this purpose.

2.8 EEP ROM

The EEP ROM is for temporary use. This thing came in handy when there is blackout. Important and essential data needed to troubleshoot later can be obtained from the EEP ROM.

Actually, the main function of this memory is to handle immediate data. Data from field devices are gathered into one central or intermediate microcontroller unit. The microcontroller unit will then queue which data to be sent to HMI.

2.9 Citect SCADA

This HMI software will be the platform of user input and output of the system. From the GUI, operators can see what is the current status of the switchgears like voltage and current reading, switch position, status of the whole system and so on. Apart from that, control such as on/off can be done.

CitectSCADA has been in the market for quite a long time now. It is easy to be used and does not require much programming. This would save time to construct new HMI software from scratch.

CHAPTER 3
METHODOLOGY

3.1 Procedure Identification

The project starts with literature review and after proper hardware and software design, procurement of electronic components assembly is conducted. After the suitable components have been bought and assembled, the software will be programmed and debugged. This process will be continued for several times until the system can work. For the project activity flow, it will follow the following flowchart.

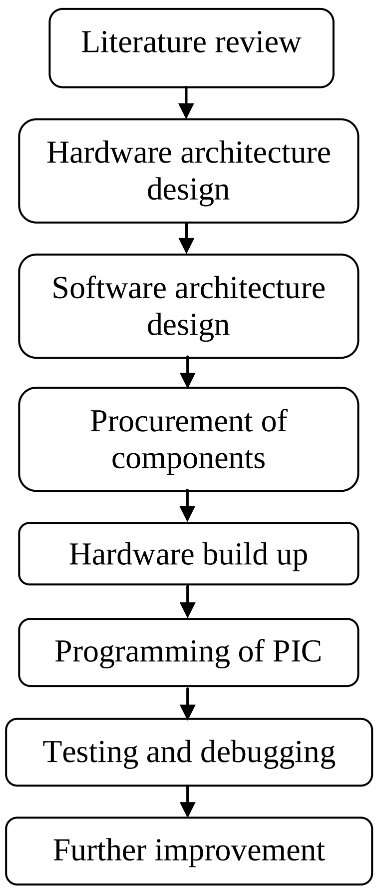


Figure 25: Flowchart of the project

3.2 Project Work

The whole project started from January till December with a total of 41 weeks. The literature review goes hand in hand with the hardware and architecture design as design changes based on requirement on the hardware and software part. For more details please refer Chapter 4: Results and Discussion.

Table 5: The project's works

Tasks \ Week	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41
Hardware Architecture Design		■	■	■	■	■															
Software Architecture Design				■	■	■	■	■													
Procurement of Components					■	■	■	■													
Hardware Build Up								■	■	■	■	■	■								
Programming of PIC									■	■	■	■	■	■							
Testing and Debugging														■	■	■	■	■			
Further Improvement																		■	■	■	■

3.3 Tools and Equipments Required

The tools that are required for this project will be categorised into two that is:

3.3.1 Hardware

Table 6: List of hardwares required

No.	Name	No.	Name
1.	Microchip PIC 16F877A	7.	PIC programmer
2.	Wireless transceivers	8.	Relays
3.	Crystal oscillators (20Mhz)	9.	Voltage regulators
4.	MAX232	10.	Switchgear cubicles (optional)
5.	USB-RS232 Converter	11.	Housing (for the electronics part)
6.	Power Adapter		

3.3.2 Software

Table 7: List of softwares needed

No.	Name	No.	Name
1.	Citect SCADA	4.	PICkit2
2.	Eagle PCB	5.	Modbus View
3.	CCS PICC		

Other supplementary tools will be added later on as the project progresses.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Hardware Architecture Design

The project progressed steadily until completion. The proposed hardware architecture also seems to be working like it should be. Voltage and current sensing instruments worked as planned.

4.1.1 General System Overview

The whole system of this project can be depicted as the figure below.

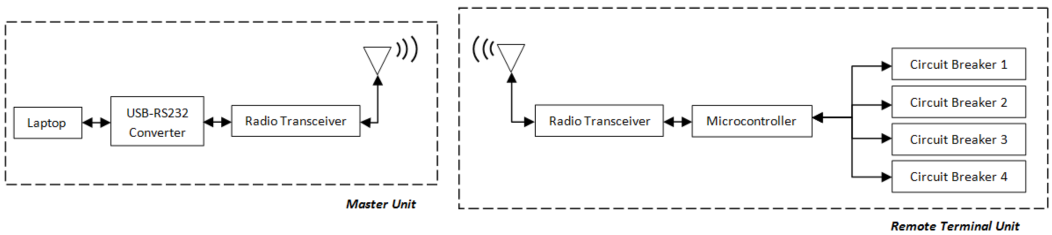


Figure 26: Project overview

The whole system is categorized into two that is the Master Unit or MU and the Remote Terminal Unit (RTU). RTU should be at the field and MU is to be anywhere as long as it is in the 1000m range.

4.1.2 *Detailed System Overview*

In the detailed system overview, the whole system is configured into two main housings; the portable transmitter and the main board (the part which is sensing and controlling the field devices, i.e. circuit breakers, motors etc).

4.1.2.1 Master Unit

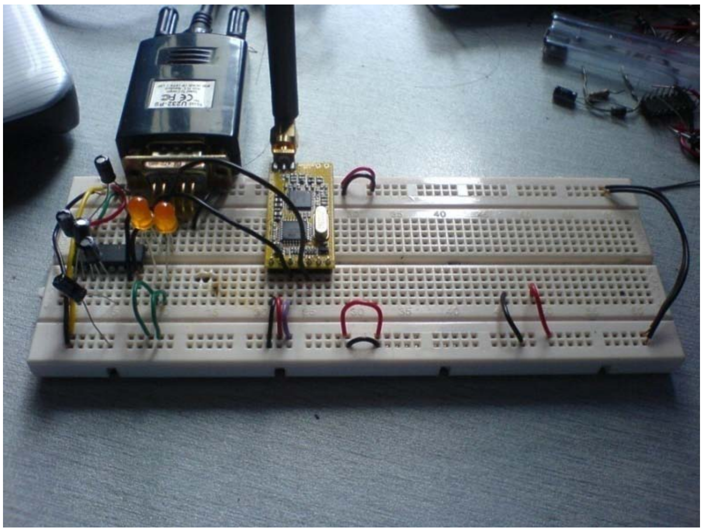


Figure 27: The portable transceiver

The MU works just like an adapter. It serves as a portable transmitter ready to be plug in into any computer.

There are three main important components in the portable transmitter, which are the MAX 232, wireless transceiver module and the Serial to USB Converter.

1. USB to Serial Converter

   The converter is quite important in this project. Without it, the portable transmitter cannot communicate properly in the right language with a computer.

   Recent computers are not readily equipped with a RS232 port. RS232 which is invented more than 30 years ago has become outdated and been replaced by the newer RS485 which uses the LAN interface.

   The converter can accommodate the widely used USB interface and convert it to simple RS232 signals to go to the MAX 232 chip.

2. MAX 232

This chip is the heart of the transmission of the system; from the MMI or any computer to the main board. The function of the chip is to regulate the voltage or in simpler terms, to step up and step down high voltage (up to 20V) from the serial port to 5V.

From this lower voltage, it is fed to the wireless transceiver. Likewise, low voltage from the wireless transceiver is stepped up to 20V to be recognized by the RS232 port at the computer. It is a conversion of RS232 to TTL.

3. Wireless Transceiver

This module, elaborated in detail in chapter one of the report, serves to transmit the data gotten from MAX 232 to the PIC wherever it is – as long it is in the range of the other wireless transceiver.

The other wireless transceiver, connected to the main board internally, would receive the incoming data and transmit back outgoing data back to the portable transmitter and to the MMI.

4.1.2.2 Remote Terminal Unit

The remote terminal unit or RTU is the brain at the field equipments. The RTU contains the necessary terminals to be connected to the circuit breakers at the switchgears.

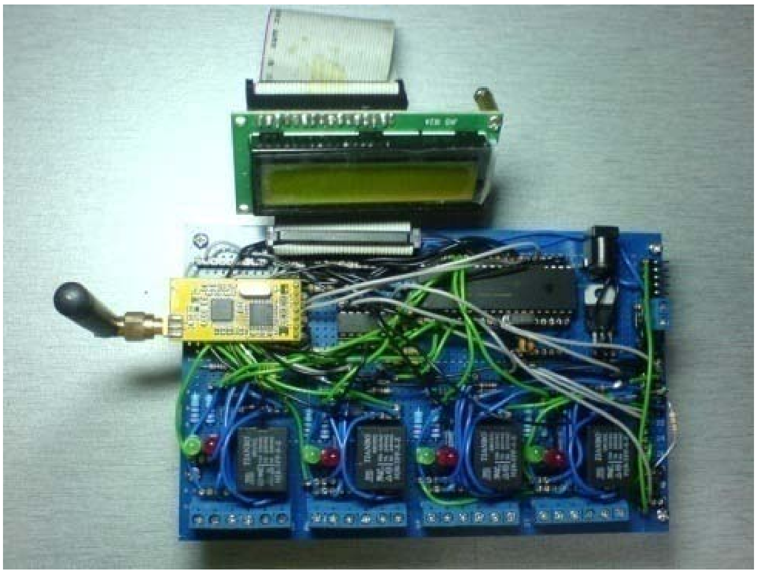


Figure 28: The RTU

It is much more complex than the portable transmitter because this is the fail or pass gateway to the system. The circuit contains:

a. Display (16x2 LCD display)
b. 16 I/O terminals for that can accommodate 4 devices
c. Wireless transceiver module
d. Microcontroller
e. Switching devices (relays)
f. Power electronics
g. Isolation/protection circuit
h. Voltage and current sensing circuits
i. ICSP

Each 9 small circuits are combined into one circuit working in harmony with each other. Every detail of the component is explained like below:

1. Display

The display is part of the output of the system. It displays what the system is doing at any given time. It indicates whether the circuit is receiving data, transmitting data or in standby waiting for command.

Without the LCD, the operator could not possibly know whether the system is working or not.

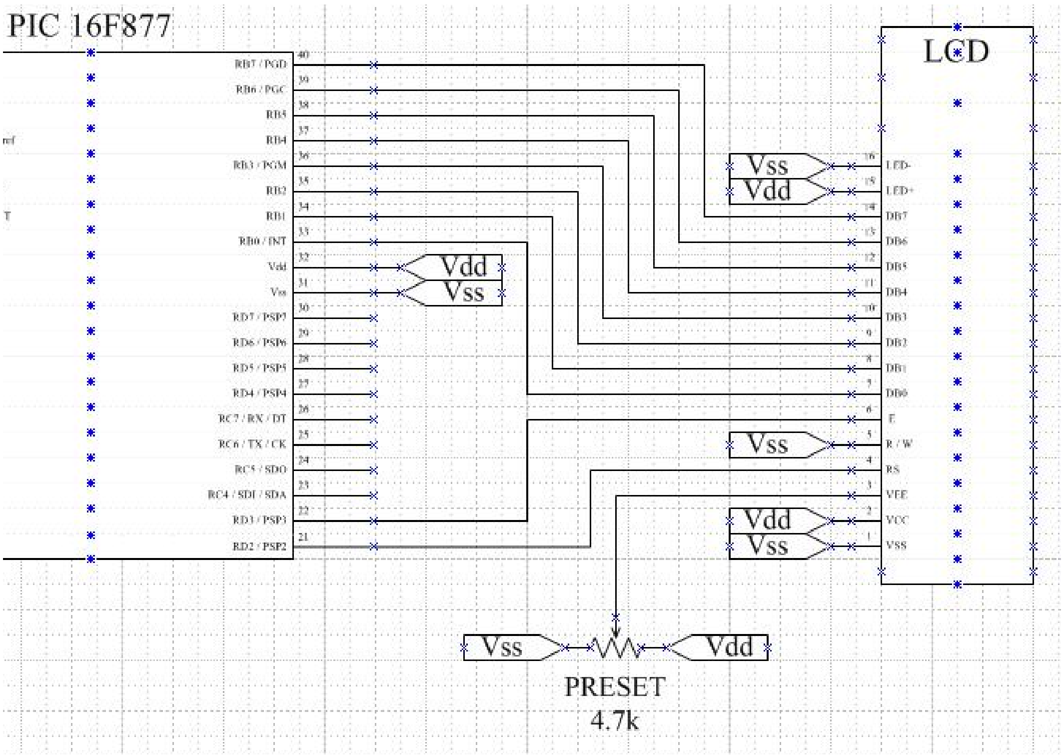


Figure 29: Connections to the 16x2 LCD

2. I/O Terminal

The interconnection between the one switchgear to the main board is done with this terminal. The terminal has 4 ports which is for different purposes resembling:

a. Voltage measurement
b. Current measurement
c. Control pins

There are thousands different type of terminals that can be chosen for this purpose, but with keeping cost and availability (for maintenance) in mind, the following terminal is used.

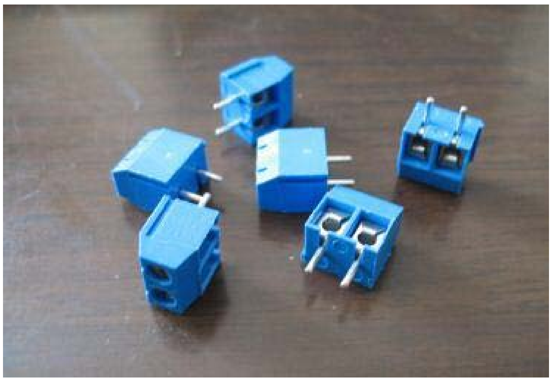


Figure 30: Terminals to outside devices

The terminal has 2 contacts and can be connected with each other to resemble say 8 contacts. With the price of 80 cents apiece, it only cost about RM12.80 for 16 contacts for the whole system.

3. Wireless transceiver module

The module is connected to pin C6 and C7 of the microcontroller which is the pin receiving data and transmitting data. It should not be accompanied with another MAX 232 chip because the proper conversion has been done at the portable transmitter side.

4. Microcontroller

One microcontroller is used in the main board system to control the whole system. It is the brain that does all the incoming data processing and giving command to transistor and switches for a successful operation.

The microcontroller is very sensitive and susceptible to small voltage difference and sparks that can burn its memory and pins. In this project, about 3 microcontrollers have burned because at that time, the isolation/protection circuit is not done yet.

Furthermore, while inserting and pulling components in and out of the breadboard while the device is still running produces arching which damages the pins and memory.

In order to avoid further damage to the PIC, ICSP circuit has been introduced to the main board.

At the end of this subsection, the whole view of the detailed schematic of the main board system will be represented and elaborated further.

5. Power Electronics

The purpose of power electronics introduced in the board is for power harmonics and spikes. The power electronics components also serve the purpose of amplifying and controlling the switches.

One example for is avoiding back emf in the switching device, diodes are used to supply the spike back to the source. This is important because if the spike is not handled, it will reset the whole main board. Experience of experimenting the circuit with high starting current device like a motor can cause a voltage dip so serious which can also reset the PIC. In some instances, spikes can destroy the PIC

The power electronics used in the circuit are:

a. Diodes
b. NPN Transistors

Different configuration of these components can yield different results and functions for different intentions.

6. Switching Devices (Relays)

The switching in the main board is done by relays. The relay is very tough and difficult to be damaged. Moreover, it is really cheap and reliable. At RM 1 per unit it is really affordable for the purpose of this project.

Figure 31: 5V relay

The specification for the relay is as follows:
Maximum voltage: 240VAC or 28VDC
Maximum current: 7Aac or 10Adc
Contact excitation voltage: 5V

The connection for the relay would be similar to the following:

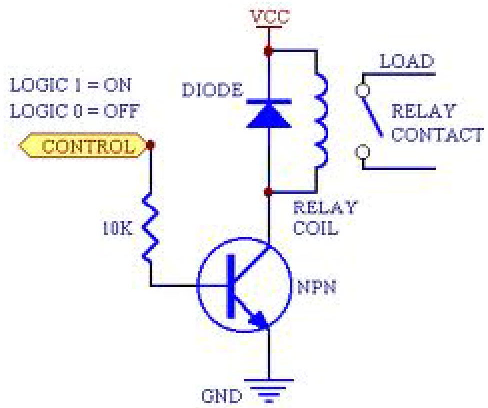


Figure 32: Transistor connection for relay

7. Voltage and current sensing circuits

Actually, both the voltage and current measuring circuits can be identified in different sections, but because of it is in the same category of measurements, it is categorized in one subsection.

a. Voltage measurement

The voltage measurement for the project is done through a series of step:

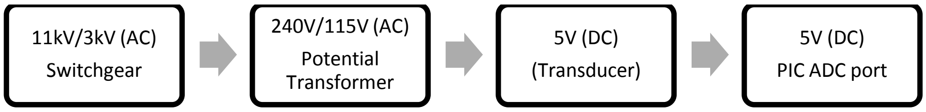


Figure 33: Implementation for real case voltage measuring

The PT (potential transformer) and transducer is not the concern of the main board as it is a conversion outside of the main board electronics itself. It is a matter of implementation at the site.

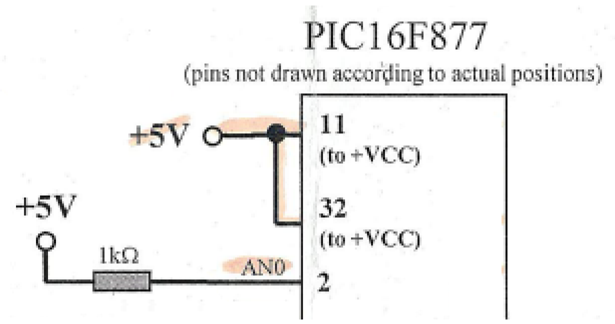


Figure 34: Voltage measurement to ADC pin

A simple 1k ohm resistor needed to be put to in series with the voltage measurement pin. The resistor serves as a current limiting resistor not to damage the pin.

b. Current measurement

It gets a bit complicated in measuring the correct current flowing in the switchgear. It follows the following procedure:

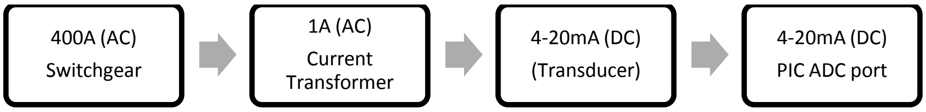


Figure 35: Implementation for real case current measurement

To detect the current properly, a special type of resistor is introduced in the circuit. The current sensing resistor of 0.2 ohm uses very small resistance to detect the small current flowing in the voltage induced line. By using V = IR, the value of current, I = V/R can be determined.

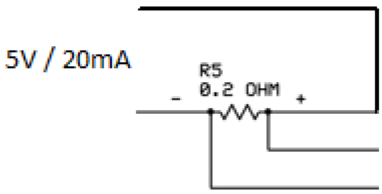


Figure 36: Current measurement to ADC pins

The 2 wires coming out of the resistor will be later connected to optoisolator giving only one output, which will be connected to a port in the PIC.

For both the measurements, the ADC is done by using 10 bits conversion resulting a 2^{10} = 1024 sampling accuracy or in simpler words, 1/1024 = 0.001. It can detect up to a change of three decimal points in the voltage or current fluctuations.

8. Isolation/protection circuits

When dealing with high voltage, it is a standard in industry electronics to bring in some kind of isolation between the high voltage of the outside system and the low voltage of the inner electronics system.

Protection in the main board is achieved by two components:

a. Voltage regulator

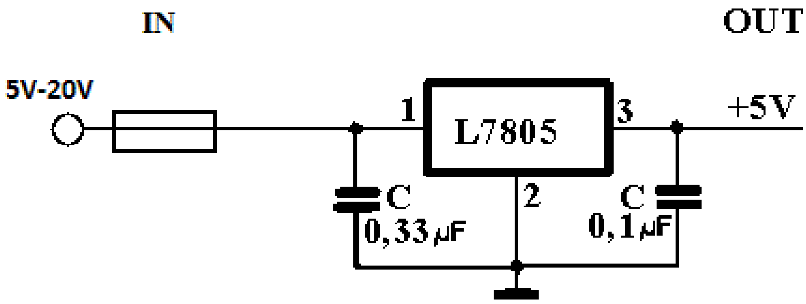


Figure 37: Connections for the L7805

b. Optoisolator

The voltage regulator will limit the incoming supply powering the electronics of the main board of to 5V. Hence, the electronics is insusceptible to voltage spikes and unstable voltage.

There are two mediums of connecting the electronics of the main board to the outside world; the supply and I/O terminals.

For the I/O terminals, optoisolators are used.

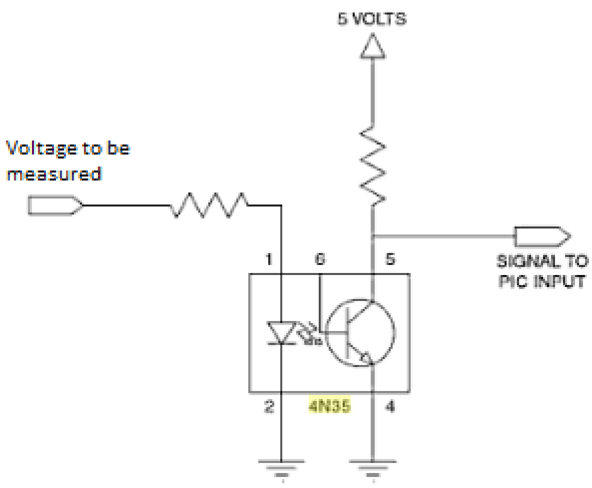


Figure 38: Connection for the 4N35

From the diagram, there is no physical connection between the voltage to be measured and the signal voltage to PIC input. The electronics are isolated from the dangerous voltage from the outside.

The transistor at the PIC side is regulated by the LED's brightness while using the safe internal 5V voltage supply.

9. ICSP

In the troubleshooting period of the project, the PIC would be reprogrammed thousands of time. This would involve plugging in and out the PIC thousands of times at the bread board. The process would not only introduce the PIC to dangerous voltage arcing, what's more the pins can be bended and damaged. As such, ICSP circuit is introduced to the main board.

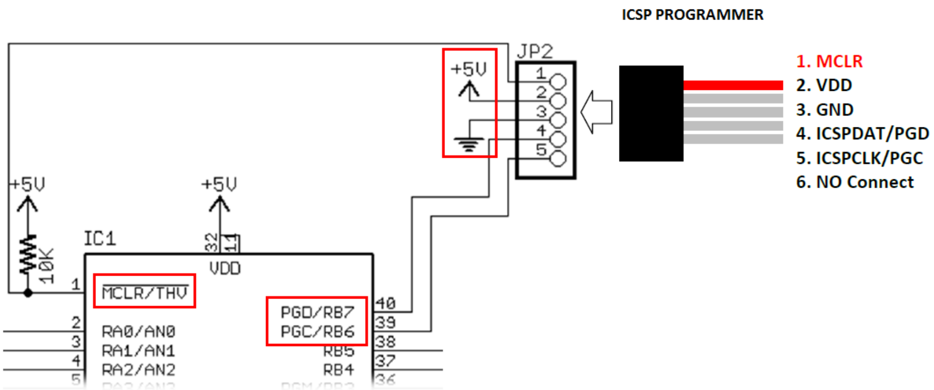


Figure 39: ICSP Connections

By using ICSP or In-Circuit Serial Programming, the PIC can stay and never be moved at all in the whole troubleshooting and testing period. This is because, an ICSP programmer module can just be plugged in the PIC and the programming and debugging is done simultaneously.

ICSP not only save debugging time, but also save cost by eliminating the probability of the PIC getting damaged.

4.2 Software System Overview

The programming was added to three this time. In the previous report, the author forgot to put very important software criteria needed for this project to work, which is the Modbus protocol functions.

4.2.1 Modbus Protocol Functions

The Modbus functions are made very wide so that it can be used in hardwares and equipments in more industries not regarding merely to industrial plants and any other rigid stuffs. It has been made so that it could be used anywhere and whatever it is necessary to accomplish the job.

As such, the functions are broad and there are some functions that will be deemed relevant and pertinent to the purpose of the project itself. But regarding switchgears, based on the Modbus Protocol Reference Guide, the author has configured the protocols to adapt to the Wireless Modbus especially for switchgear applications.

Table 8: Modbus function code assimilated to the project

No.	Tasks	Function in Modbus	Modbus Description	Priority
1	On and OFF of one device	05 (0x05)	Force Single Coil	High
2	Emergency shutdown	05 (0x05), in broadcast mode	Force Single Coil	High
3	Shutdown of multiple coils controlling one device (same department)	15 (0x0F)	Force Multiple Coils	High
4	Status, on/off	01 (0x01)	Read Coil Status	High
5	Status on input before on/off (readiness)	04 (0x04)	Read Input Registers	Medium
6	System diagnostics	08 (0x08)	Diagnostics (serial line only)	Medium
7	History	11 (0x0B)	Fetch Communication Event Counter	Medium
8	Electric meter (read values of current, voltage)	02 (0x02)	Read Input Status	Medium
9	Log (event log)	12 (0x0C)	Fetch Communication Event Log	Medium
10	Block write capabilities (line of authority)	22 (0x16)	Mask Write 4X Registers	Low
11	Read queue	24 (0x18)	Read FIFO Queue	Low

In this switchgear case, the device might be feeders, MCC, motors etc. In the table, the priority was given in the case of which to work for first. If it is high, it means that without this function, the device could not work. Or it simply means, this is the body of the project and this is the main functions that are used.

If it is medium, it means that with this function, it would be a great add-on to the system. This would help on maintaining and provides more information to the person handling it.

Lastly, the low priority is that the system could run perfectly even without these functions. These functions are extra functions which would or would not be useful but whatever the reason is, this project could survive or not have a negative impact even if this function did not exist. However, all of the functions are assembled together; from function 01 to 24

4.2.2 CitectSCADA

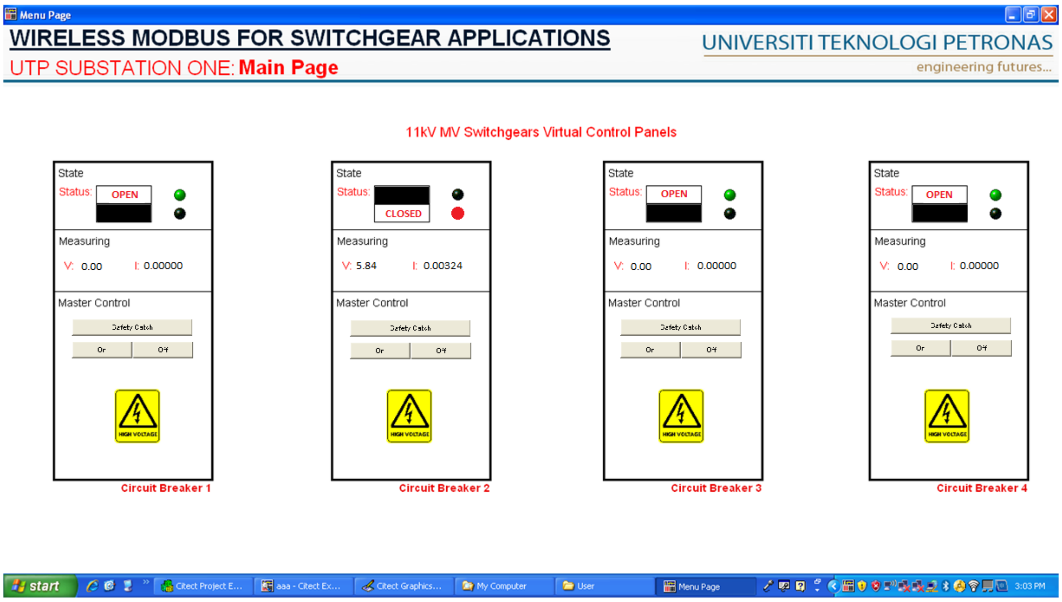


Figure 40: SCADA in operation

The software for the HMI is done using Object Oriented Programming (OOP), thus there is no source code to show. The software is working beautifully with the functions on/off and switch status available.

4.2.3 Microcontroller Programming

The microcontroller programming for the Master Control Hardware has been done. Please refer Appendix VI.

CHAPTER 5
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project managed to complete its primary objective that is to save human live. By able to switch a switchgear a distance away, outside of dangers zone, precious human capital can be saved.

All of the designs have been completed. There is some accuracy error in the voltage and especially current measurement. More money is needed for precision electronic components that can withstand slight temperature increase or decrease, electromagnetic noise and harmonics in the circuit. A burr-brown IC in the likes of the INA 11 can be used for this purpose.

In the final design, cheap optoisolator as recommended in the report is omitted because of its inversely proportional output to input. Furthermore, the accuracy would severely be damaged or distorted because of the use of the component. A HCPL-7520 is really recommended for all the 8 inputs to the ADC in the microcontroller for the purpose of isolation and accurate measuring. It is packed with operational amplifiers, optoisolators and other necessary components to make the output linear and steady.

5.2 Recommendations

In the improvement recommendation, if the project is to be continued by any other person or parties, proper isolation of incoming voltages in the terminal section should be sought after first. This is to ensure proper accuracy and isolation. Currently, the circuit is not protected.

Secondly, the laptop for the SCADA should be replaced by a Personal Digital Assistant (PDA) such as a PocketPC or preferably the more popular iPhone. Appropriate SCADA software should be selected for this work. For this, proper interface medium between the transceiver for the laptop should have 2 wireless modules; one from the transceiver to the main board and the other one from the transceiver to the PDA.

Thirdly, the RS232 should be upgraded to the more high speed and more modern RS485 UTP (Unshielded Twisted Pair). For this to happen, the MAX 485 should be properly configured to be implemented with the PIC 16f877a.

If the RS485 is implemented, comes the fourth recommendation, to setup a server for more main boards to be connected to a central server. This server should be the laptop itself to save cost. When this is done, more nodes/slaves can be connected to each other. This means that more substations can be controlled by the laptop. At this time, the project will get more complicated.

Finally, the range of the communication from the main board and transceiver should be increased to about 10km or so. Imagine in UTP, there are more than 2 dozen substations in the academic complex. There are lots of academic block from end to end and this covers a large area. Future wireless communication should be about 10km or so. If this is not continued, then proper programming of the MODBUS slave is needed. This is to add for hop-in features. Hop-in means that data will hop from an address to another till it get to its destination. Thus, the range will be reduced to the last nearest node, which should be less than more than half.

REFERENCES

[1] Elearning Skillgroup (SKG) 13 Elearning. 2008, *Module 7: Distribution System*, PETRONAS Carigali.

[2] MODBUS.ORG. 28th December 2006, "*Modbus Application Protocol Specification V1.1b*"

[3] MODBUS.ORG. 20th December 2006, "*Modbus Over Serial Line: Specification and Implementation Guide V1.02*"

[4] MODICON, Inc., Industrial Automation Systems. June 1996, "*Modicon Modbus Protocol Reference Guide, Rev. 7*"

[5] Richard H. Barnett, Sarah Cox, Larry O'Cull. 2003, *Embedded C Programming and the Microchip PIC 1st Edition*, Delmar Cengage Learning

[6] Wikipedia, February 2010 <http://en.wikipedia.org/wiki/Modbus>

[7] William Stallings. 2004, *Data and Computer Communications, 7th Edition*, Prentice-Hall

[8] F. H. Ramírez Leyva, J. P. Acevedo Cuellar, R. G. Maldonado Basilio, E. Espinosa Justo, 2004, "Wireless System for Electrical Networks Testing Based on MODBUS Protocol", *14th International Conference on Electronics, Communications and Computers (CONIELECOMP'04)*

[9] Microchip. 2003, *"PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers"*

[10] Agilent Technologies. 2003, *"Agilent HCPL-7520 Isolated Linear Sensing IC Data Sheet"*

[11] Ohmite. 2003, *"10 Series Axial Wire Element Current Sense"*

[12] Sure Electronics. 2009, *"Wireless RF Transceiver 431-478 Mhz GFSK Data Transfer User Guide"*

[13] Maxim. 2006, *"+5V-Powered, Multichannel RS-232 Drivers/Receivers Rev. 15"*

[14] Cliff Jao and Xi Guo. 2008, *"Power Box: The Safe Power AC Meter"*, <http://courses.cit.cornell.edu/ee476/FinalProjects/s2008/cj72_xg37/cj72_xg37/index.html>, Electrical Engineering Department, Cornell University

[15] Malaysia Department of Safety and Health (DOSH), *Annual Report 2009*

[16] Suruhanjaya Tenaga Malaysia, *"Electricity Supply Industry in Malaysia 2008"*

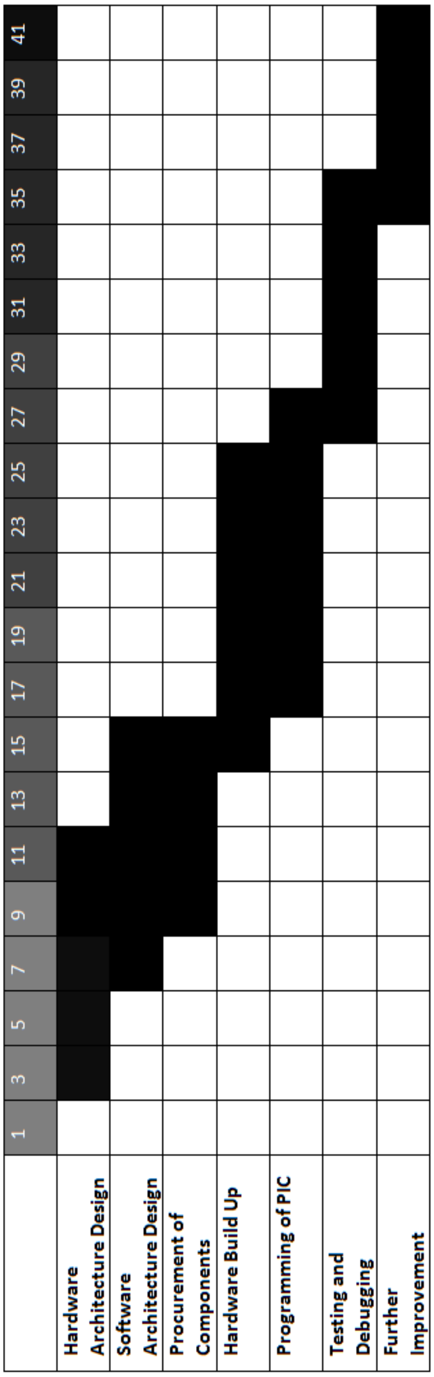
[17] Motorola Semiconductor, *"4N35 – 6 Pin DIP Optoisolator"*

[18] ST Microelectronics, "L7805 Series"

APPENDICES

Appendix I

Gantt Chart



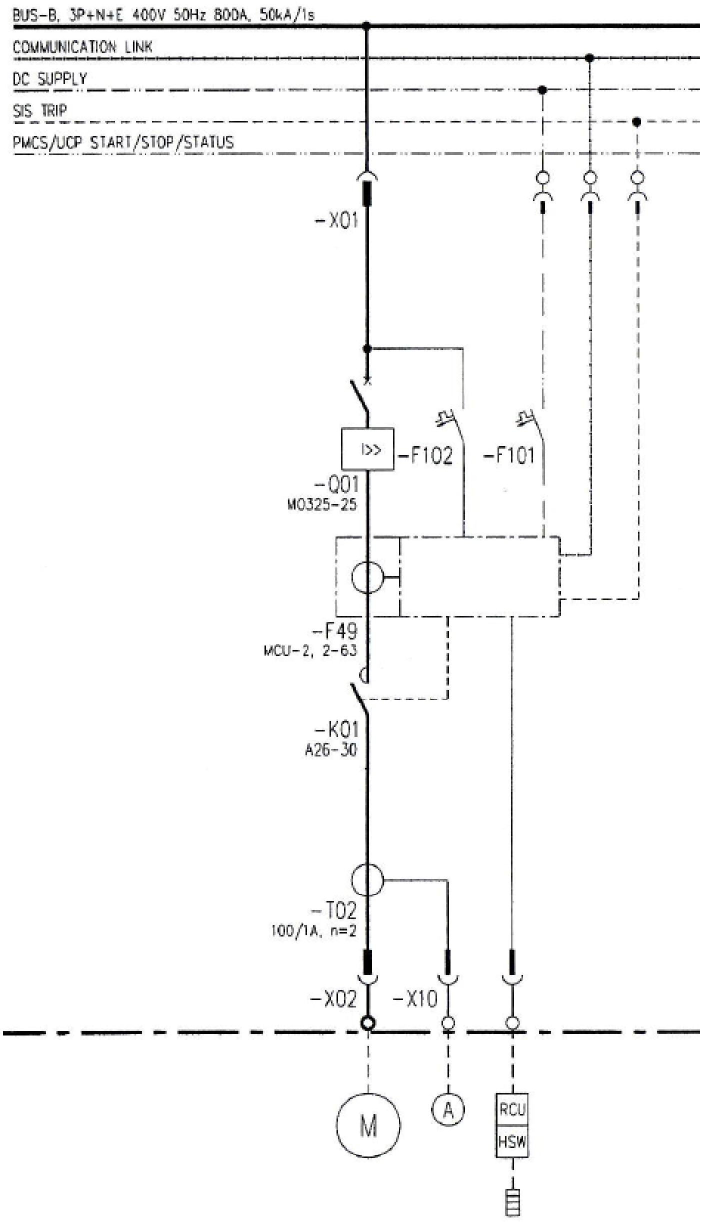
Appendix II

LV Switchgear

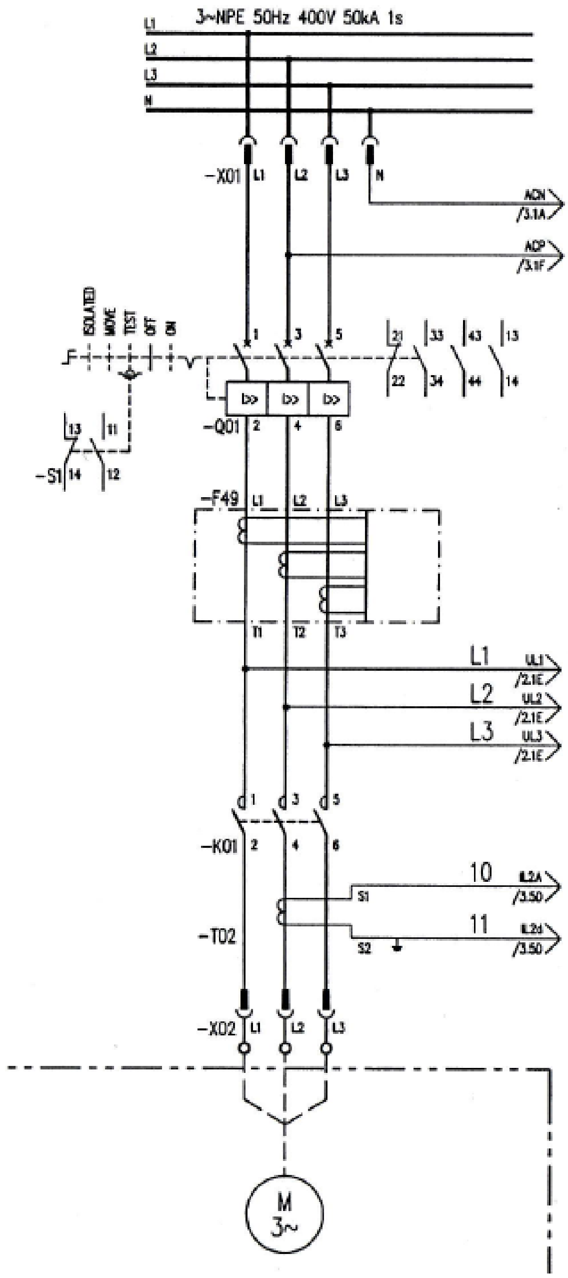
The numbers and spaces of only LVS for a Central Processing Platform of an oil and gas project offshore Sarawak.

Appendix III

Motor Wiring Diagram



The wirings required for one motor panel for a motor. There are 300 motors in a typical oil platform utilizing the need for LVS.



The detailed view of the electrical parts which consists of wires for relays, current transformers, contactors, overcurrent protection and so on for one motor. There are lots of wire regarding this. It is a typical about 7 wires in this diagram, not inclusive of the thermal protection for the motor and so on. So for one platform, there will be about 2100 individual wires doing their own function. If a minimal of 100m per wire, then there will be 2,100,000m or 2,100km for the total of the motor loads.

Appendix IV
CRC Generation

The Cyclical Redundancy Check (CRC) field is two bytes, containing a 16–bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The CRC is started by first preloading a 16–bit register to all 1's. Then a process begins of applying successive 8–bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit, do not apply to the CRC.

During generation of the CRC, each 8–bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8–bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, is the CRC value. A procedure for generating a CRC is:

1. Load a 16–bit register with FFFF hex (all 1's). Call this the CRC register.

2. Exclusive OR the first 8–bit byte of the message with the low–order byte of the 16–bit CRC register, putting the result in the CRC register.
3. Shift the CRC register one bit to the right (toward the LSB), zero–filling the MSB. Extract and examine the LSB.
4. (If the LSB was 0): Repeat Step 3 (another shift). (If the LSB was 1): Exclusive OR the CRC register with the polynomial value A001 hex (1010 0000 0000 0001).
5. Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8–bit byte will have been processed.
6. Repeat Steps 2 through 5 for the next 8–bit byte of the message. Continue doing this until all bytes have been processed.
7. The final contents of the CRC register is the CRC value.
8. When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

Example

An example of a C language function performing CRC generation is shown on the following pages. All of the possible CRC values are preloaded into two arrays, which are simply indexed as the function increments through the message buffer. One array contains all of the 256 possible CRC values for the high byte of the 16–bit CRC field, and the other array contains all of the values for the low byte.

Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

The function takes two arguments:

unsigned char *puchMsg ; A pointer to the message buffer containing binary data to
be used for generating the CRC

unsigned short usDataLen ; The quantity of bytes in the message buffer.

The function returns the CRC as a type unsigned short.

CRC Generation Function

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ;                 /* message to calculate CRC upon */
unsigned short usDataLen ;               /* quantity of bytes in message */
{
unsigned char uchCRCHi = 0xFF ;   /* high byte of CRC initialized */
unsigned char uchCRCLo = 0xFF ;   /* low byte of CRC initialized */
unsigned uIndex ;                        /* will index into CRC lookup table */
while (usDataLen—)                       /* pass through message buffer */
{
uIndex = uchCRCHi ^ *puchMsgg++ ; /* calculate the CRC */
uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex} ;
uchCRCLo = auchCRCLo[uIndex] ;
}
return (uchCRCHi << 8 | uchCRCLo) ;
}
```

Appendix V
LRC Generation

The Longitudinal Redundancy Check (LRC) field is one byte, containing an 8–bit binary value. The LRC value is calculated by the transmitting device, which appends the LRC to the message. The receiving device recalculates an LRC during receipt of the message, and compares the calculated value to the actual value it received in the LRC field. If the two values are not equal, an error results. The LRC is calculated by adding together successive 8–bit bytes in the message, discarding any carries, and then two's complementing the result. The LRC is an 8–bit field, therefore each new addition of a character that would result in a value higher than 255 decimal simply 'rolls over' the field's value through zero. Because there is no ninth bit, the carry is discarded automatically. A procedure for generating an LRC is:

1. Add all bytes in the message, excluding the starting 'colon' and ending CRLF. Add them into an 8–bit field, so that carries will be discarded.
2. Subtract the final field value from FF hex (all 1's), to produce the ones–complement.
3. Add 1 to produce the twos–complement.

Example

An example of a C language function performing LRC generation is shown below.

The function takes two arguments:

unsigned char *auchMsg ; A pointer to the message buffer containing binary data to be used for generating the LRC

unsigned short usDataLen ; The quantity of bytes in the message buffer.

The function returns the LRC as a type unsigned char.

LRC Generation Function

```
static unsigned char LRC(auchMsg, usDataLen)
unsigned char *auchMsg ;                        /* message to calculate LRC upon */
unsigned short usDataLen ;                      /* quantity of bytes in message */
{
unsigned char uchLRC = 0 ;                      /* LRC char initialized */
while (usDataLen—)                              /* pass through message buffer */
uchLRC += *auchMsg++ ;                          /* add buffer byte without carry */
return ((unsigned char)(–((char)uchLRC))) ; /* return twos complement */
}
```

Appendix VI

Modbus Slave RTU Coding

```
#define USE_WITH_PC 1

#include <16f877a.h>
#device adc = 10
#device *=16
#fuses HS, NOWDT, NOLVP, NOBROWNOUT, NOPROTECT, PUT
#use delay(clock=20M)
#include <lcd.c>

#define MODBUS_TYPE MODBUS_TYPE_SLAVE
#define MODBUS_SERIAL_RX_BUFFER_SIZE 64
#define MODBUS_SERIAL_BAUD 9600

#ifndef USE_WITH_PC
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_EXT
#define MODBUS_SERIAL_TX_PIN PIN_B1   // Data transmit pin
#define MODBUS_SERIAL_RX_PIN PIN_B0   // Data receive pin
//The following should be defined for RS485 communication
//#define MODBUS_SERIAL_ENABLE_PIN   0   // Controls DE pin for RS485
//#define MODBUS_SERIAL_RX_ENABLE    0   // Controls RE pin for RS485
#else
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA
#endif

unsigned int16 adcvalue1, adcvalue2, adcvalue3, adcvalue4, adcvalue5, adcvalue6, adcvalue7, adcvalue8;

#include "my_modbus.c"
#define MODBUS_ADDRESS 0x01

/*This function may come in handy for you since MODBUS uses MSB first.*/
int8 swap_bits(int8 c)
{
   return ((c&1)?128:0)|((c&2)?64:0)|((c&4)?32:0)|((c&8)?16:0)|((c&16)?8:0)
          |((c&32)?4:0)|((c&64)?2:0)|((c&128)?1:0);
}

void main()
{
   int8 coils = 0b00000000;
   int8 inputs = 0b00000000;
   int16 hold_regs[] = {0x8800,0x7700,0x6600,0x5500,0x4400,0x3300,0x2200,0x1100};
   int16 input_regs[] = {0,1,2,3,4,5,6,7};//0x1100,0x2200,0x3300,0x4400,0x5500,0x6600,0x7700,0x8800
   int16 event_count = 0;

   setup_adc_ports( ALL_ANALOG );
   setup_adc(ADC_CLOCK_INTERNAL);        // Use internal ADC clock.

   lcd_init();
   delay_ms(50);
   lcd_putc("\fWireless MODBUS");
   lcd_putc("\nStandby..");
   delay_ms(50);

   modbus_init();

   while(TRUE)
   {
      inputs = input_c();
      output_b(coils);

            set_adc_channel(0);
            delay_us(50);  // Delay for sampling cap to charge
            adcvalue1 = read_adc(); // Get ADC reading
```

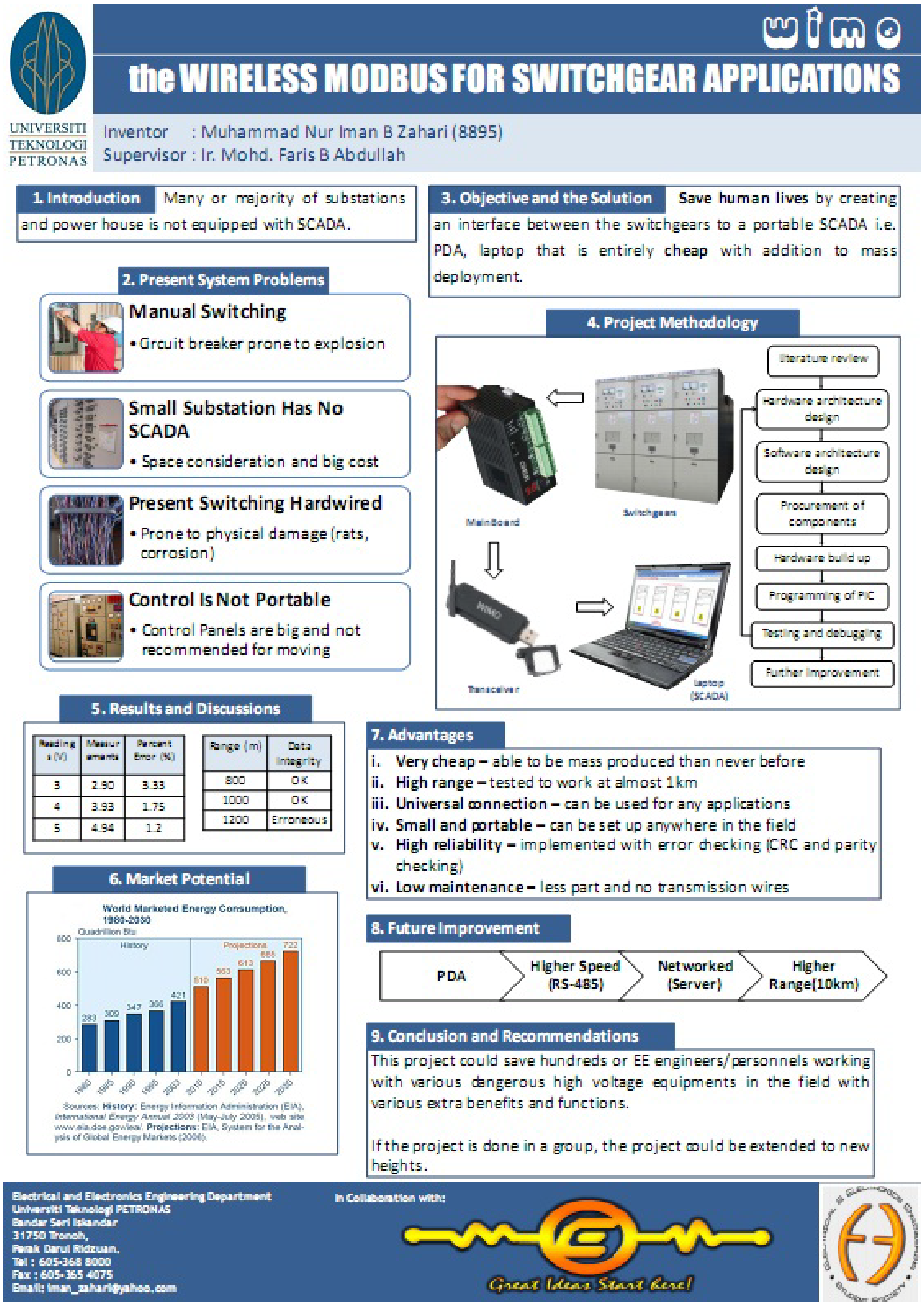
```
            set_adc_channel(1);
            delay_us(50);
            adcvalue2 = read_adc();

            set_adc_channel(2);
            delay_us(50);
            adcvalue3 = read_adc();

            set_adc_channel(3);
            delay_us(50);
            adcvalue4 = read_adc();

            set_adc_channel(4);
            delay_us(50);
            adcvalue5 = read_adc();

            set_adc_channel(5);
            delay_us(50);
            adcvalue6 = read_adc();

            set_adc_channel(6);
            delay_us(50);
            adcvalue7 = read_adc();

            set_adc_channel(7);
            delay_us(50);
            adcvalue8 = read_adc();

//          delay_us(50);  // Preset delay, repeat every 10ms
//          voltan1 = 0.9358717434869739 * 5.000 * adcValue1 / 1024.000;
//          voltan2 = 0.9358717434869739 * 5.000 * adcValue2 / 1024.000;

      //adcvalue1 = "%02X%02X",make8(adcvalue1,0),make8(adcvalue1,1);
      adcvalue1 = make16(make8(adcvalue1,0),make8(adcvalue1,1));
      adcvalue2 = make16(make8(adcvalue2,0),make8(adcvalue2,1));
      adcvalue3 = make16(make8(adcvalue3,0),make8(adcvalue3,1));
      adcvalue4 = make16(make8(adcvalue4,0),make8(adcvalue4,1));
      adcvalue5 = make16(make8(adcvalue5,0),make8(adcvalue5,1));
      adcvalue6 = make16(make8(adcvalue6,0),make8(adcvalue6,1));
      adcvalue7 = make16(make8(adcvalue7,0),make8(adcvalue7,1));
      adcvalue8 = make16(make8(adcvalue8,0),make8(adcvalue8,1));

      input_regs[0] = adcvalue1;
      input_regs[1] = adcvalue2;
      input_regs[2] = adcvalue3;
      input_regs[3] = adcvalue4;
      input_regs[4] = adcvalue5;
      input_regs[5] = adcvalue6;
      input_regs[6] = adcvalue7;
      input_regs[7] = adcvalue8;

      while(!modbus_kbhit());

      lcd_putc("\nCommand Received");

      //check address against our address, 0 is broadcast
      if((modbus_rx.address == MODBUS_ADDRESS) || modbus_rx.address == 0)
      {
        switch(modbus_rx.func)
        {
          case FUNC_READ_COILS:    //read coils
          case FUNC_READ_DISCRETE_INPUT:    //read inputs
            if(modbus_rx.data[0] || modbus_rx.data[2] ||
              modbus_rx.data[1] >= 8 || modbus_rx.data[3]+modbus_rx.data[1] > 8)
              modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
            else
            {
              int8 data;
```

```
            if(modbus_rx.func == FUNC_READ_COILS)
              data = coils>>(modbus_rx.data[1]);      //move to the starting coil
            else
              data = inputs>>(modbus_rx.data[1]);      //move to the starting input

            data = data & (0xFF>>(8-modbus_rx.data[3]));  //0 out values after quantity

            if(modbus_rx.func == FUNC_READ_COILS)
              modbus_read_discrete_input_rsp(MODBUS_ADDRESS, 0x01, &data);
            else
              modbus_read_discrete_input_rsp(MODBUS_ADDRESS, 0x01, &data);

            event_count++;
          }
          break;
        case FUNC_READ_HOLDING_REGISTERS:
        case FUNC_READ_INPUT_REGISTERS:
          if(modbus_rx.data[0] || modbus_rx.data[2] ||
            modbus_rx.data[1] >= 8 || modbus_rx.data[3]+modbus_rx.data[1] > 8)
            modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
          else
          {
            if(modbus_rx.func == FUNC_READ_HOLDING_REGISTERS)

modbus_read_holding_registers_rsp(MODBUS_ADDRESS,(modbus_rx.data[3]*2),hold_regs+modbus_rx.data[1]);
            else

modbus_read_input_registers_rsp(MODBUS_ADDRESS,(modbus_rx.data[3]*2),input_regs+modbus_rx.data[1]);

            event_count++;
          }
          break;
        case FUNC_WRITE_SINGLE_COIL:      //write coil
          if(modbus_rx.data[0] || modbus_rx.data[3] || modbus_rx.data[1] > 8)
            modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
          else if(modbus_rx.data[2] != 0xFF && modbus_rx.data[2] != 0x00)
            modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_VALUE);
          else
          {
            //coils are stored msb->lsb so we must use 7-address
            if(modbus_rx.data[2] == 0xFF)
              bit_set(coils,7-modbus_rx.data[1]);
            else
              bit_clear(coils,7-modbus_rx.data[1]);


modbus_write_single_coil_rsp(MODBUS_ADDRESS,modbus_rx.data[1],((int16)(modbus_rx.data[2]))<<8);

            event_count++;
          }
          break;
        case FUNC_WRITE_SINGLE_REGISTER:
          if(modbus_rx.data[0] || modbus_rx.data[1] >= 8)
            modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
          else
          {
            //the registers are stored in little endian format
            hold_regs[modbus_rx.data[1]] = make16(modbus_rx.data[3],modbus_rx.data[2]);

            modbus_write_single_register_rsp(MODBUS_ADDRESS,
                      make16(modbus_rx.data[0],modbus_rx.data[1]),
                      make16(modbus_rx.data[2],modbus_rx.data[3]));
          }
          break;
        case FUNC_WRITE_MULTIPLE_COILS:
          if(modbus_rx.data[0] || modbus_rx.data[2] ||
            modbus_rx.data[1] >= 8 || modbus_rx.data[3]+modbus_rx.data[1] > 8)
            modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
          else
```

```
            {
              int i,j;

              modbus_rx.data[5] = swap_bits(modbus_rx.data[5]);

              for(i=modbus_rx.data[1],j=0; i < modbus_rx.data[1]+modbus_rx.data[3]; ++i,++j)
              {
                if(bit_test(modbus_rx.data[5],j))
                  bit_set(coils,7-i);
                else
                  bit_clear(coils,7-i);
              }

              modbus_write_multiple_coils_rsp(MODBUS_ADDRESS,
                          make16(modbus_rx.data[0],modbus_rx.data[1]),
                          make16(modbus_rx.data[2],modbus_rx.data[3]));

              event_count++;
            }
            break;
          case FUNC_WRITE_MULTIPLE_REGISTERS:
            if(modbus_rx.data[0] || modbus_rx.data[2] ||
              modbus_rx.data[1] >= 8 || modbus_rx.data[3]+modbus_rx.data[1] > 8)
              modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
            else
            {
              int i,j;

              for(i=0,j=5; i < modbus_rx.data[4]/2; ++i,j+=2)
                hold_regs[i] = make16(modbus_rx.data[j+1],modbus_rx.data[j]);

              modbus_write_multiple_registers_rsp(MODBUS_ADDRESS,
                          make16(modbus_rx.data[0],modbus_rx.data[1]),
                          make16(modbus_rx.data[2],modbus_rx.data[3]));

              event_count++;
            }
            break;
          default:    //We don't support the function, so return exception
            modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_FUNCTION);
        }

        delay_ms(100);
        lcd_putc("\nStandby..       ");
      }
  }
}
```

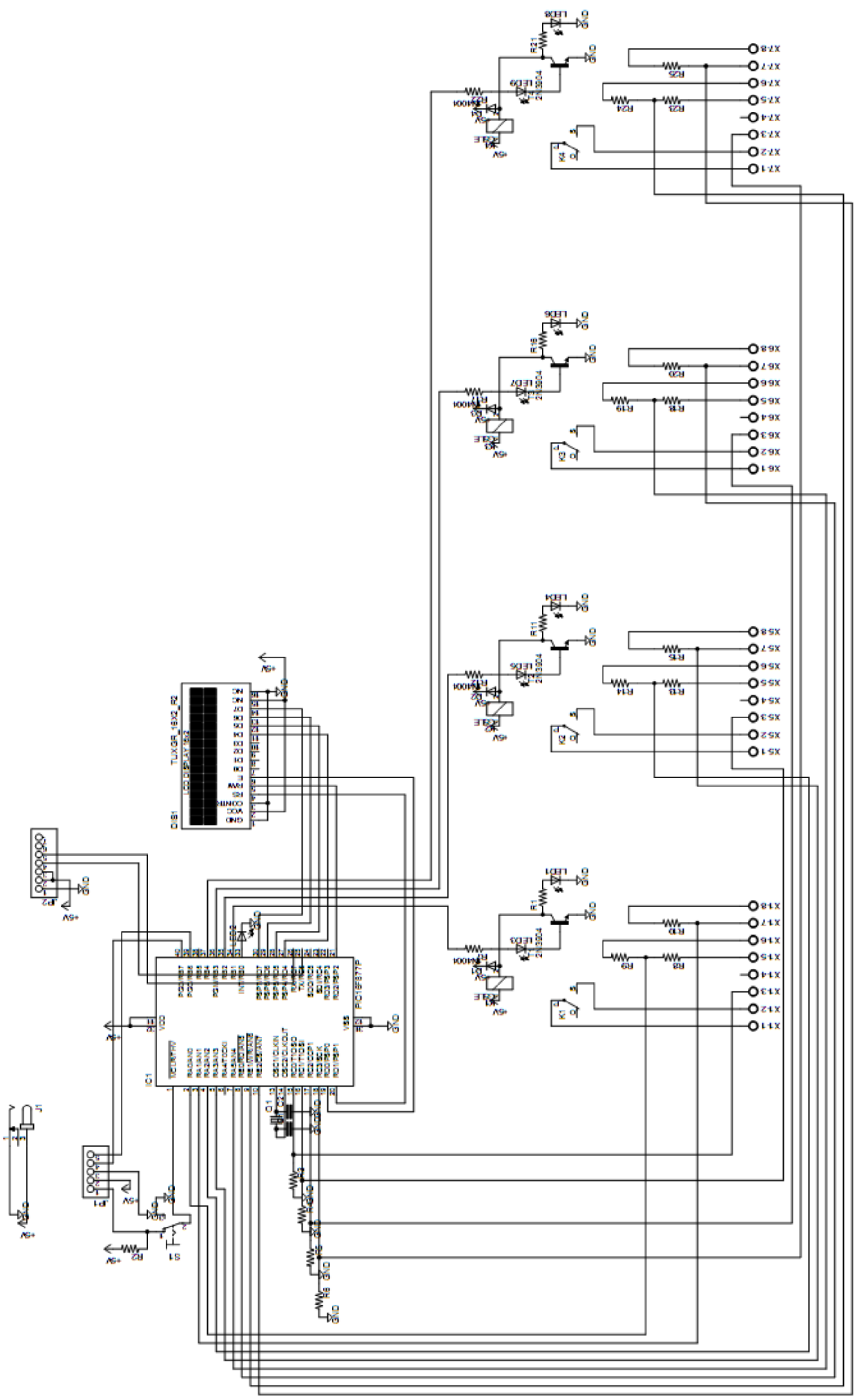
Appendix VII

Poster Presentation



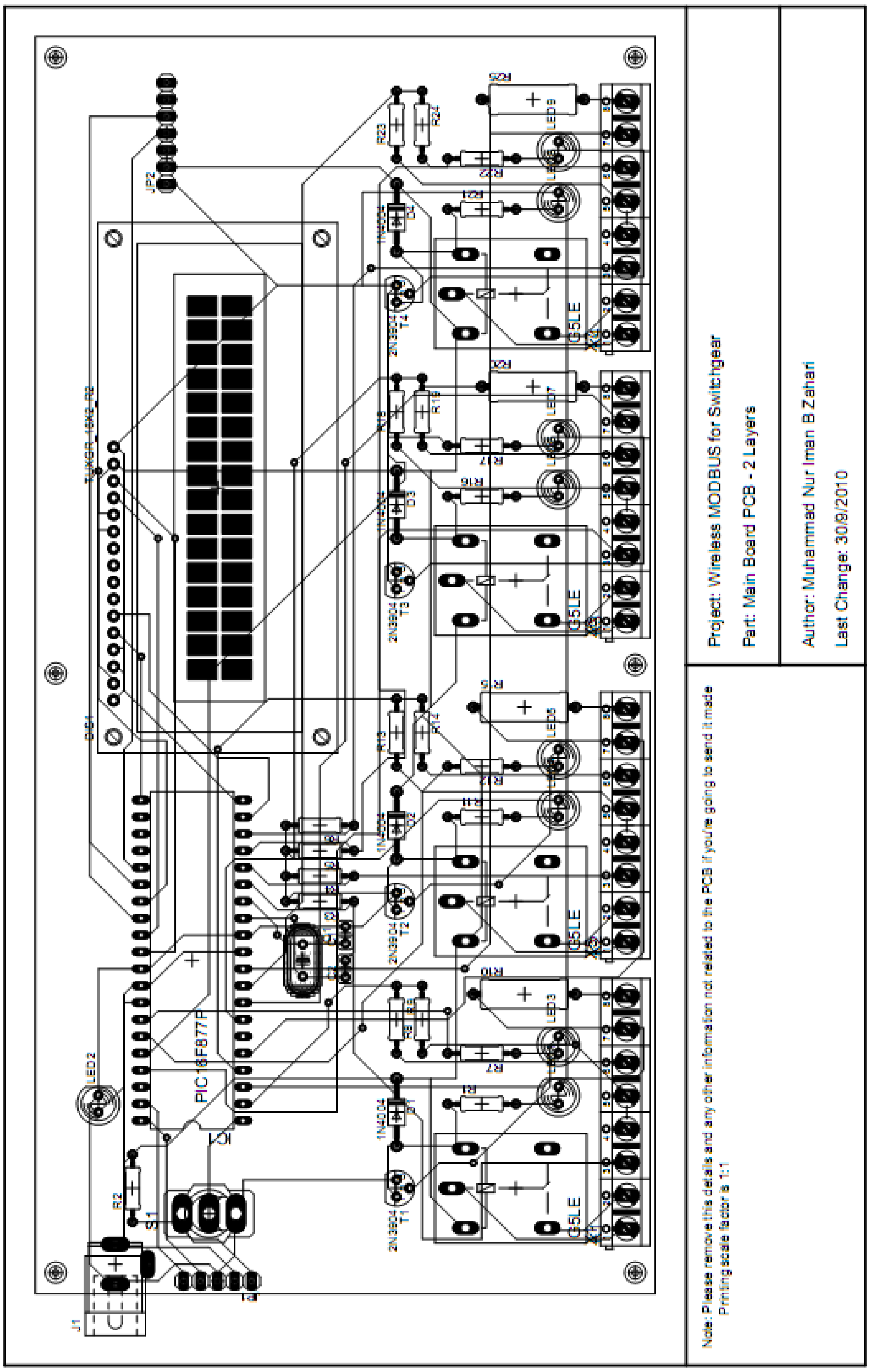
Appendix VIII

Main Board Schematics



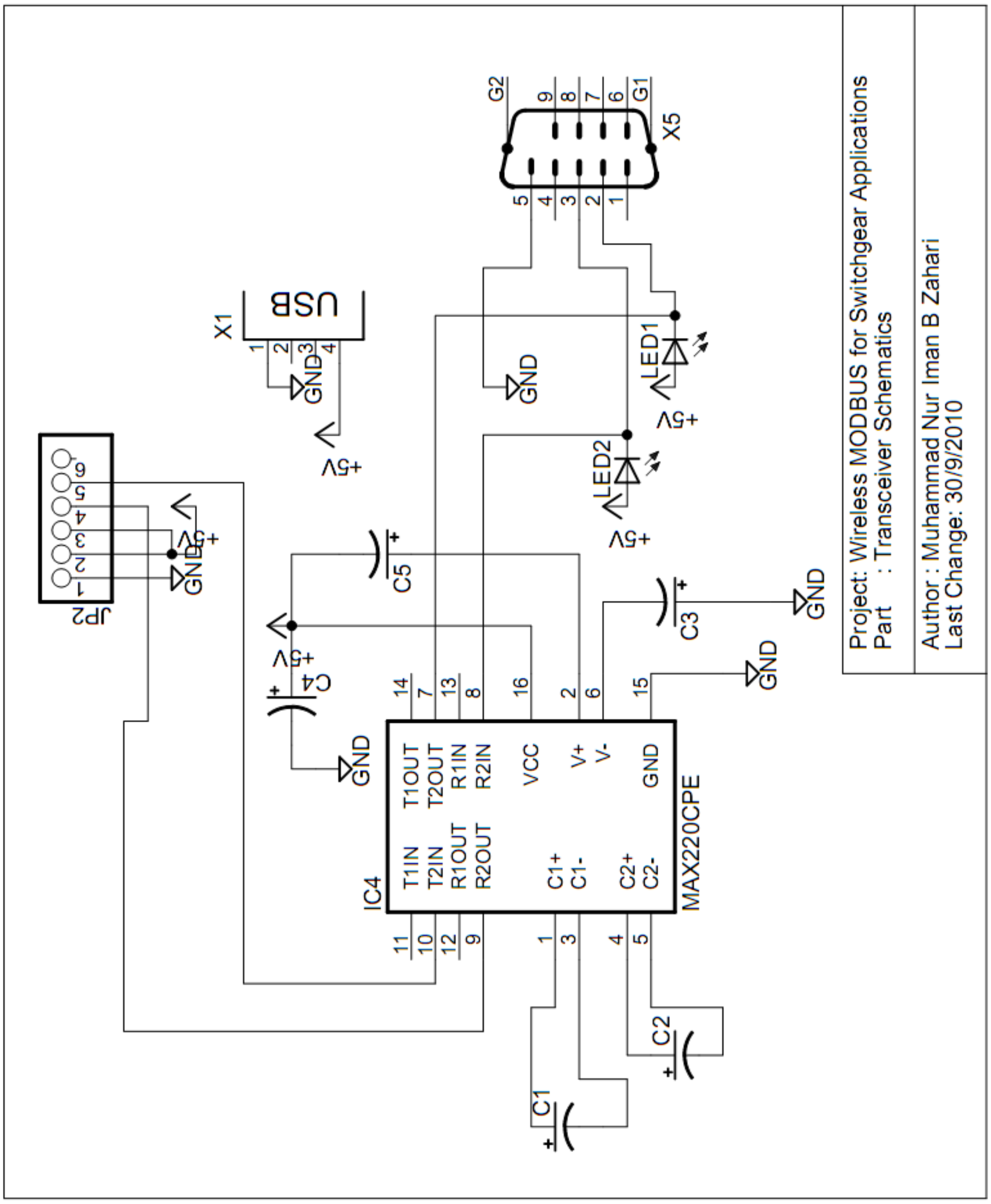
Appendix IX

Main Board PCB



Appendix X

Transceiver Schematics



Appendix XI

Transceiver PCB