CHAPTER 3

METHODOLOGY

3.1 Introduction

A success project is developed based on a suitable and reliable project design cycle. As for this project, the project design is consist of five structured phases which are:

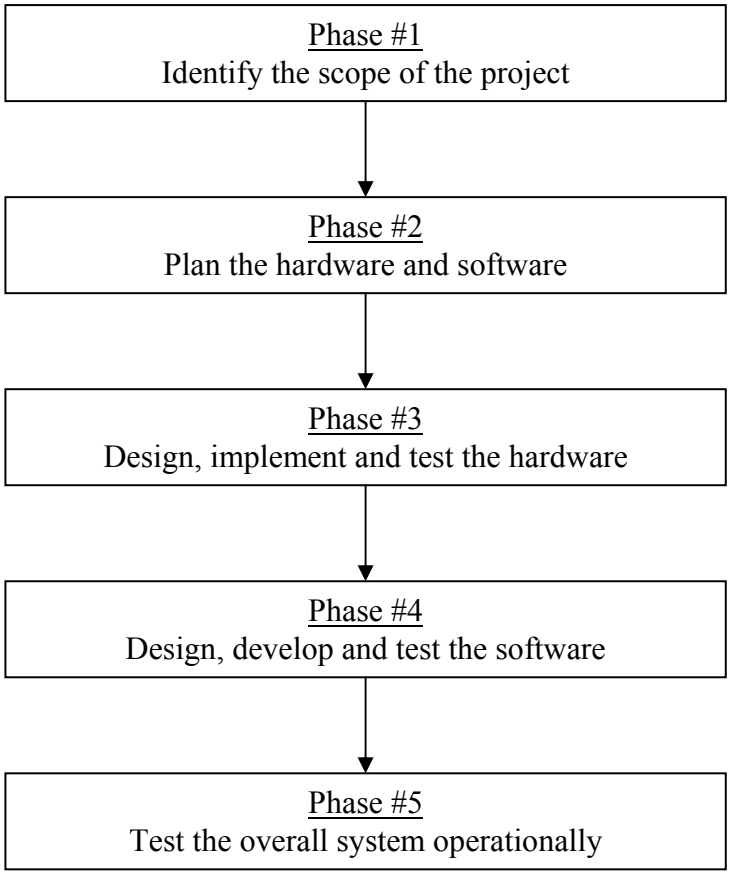


Figure 3.1: Structured Project Design Phase

3.2 Phase #1 - Identify The Scope Of The Project

A scope of a project covers the overall system requirement which needs to be accomplished. In this project, it is aim to achieve a wireless communication for data transmission from a transmitter side to a receiver side by using embedded controller. The data transmitted is the digital data which has been converted from an analog signal by the ADC. To implement the application of this project, a Graphical User Interface (GUI) software needs to be develop for monitoring purpose. A basic block diagram is illustrated below to have a better understanding on steps taken to obtain the desired outcome of the project.

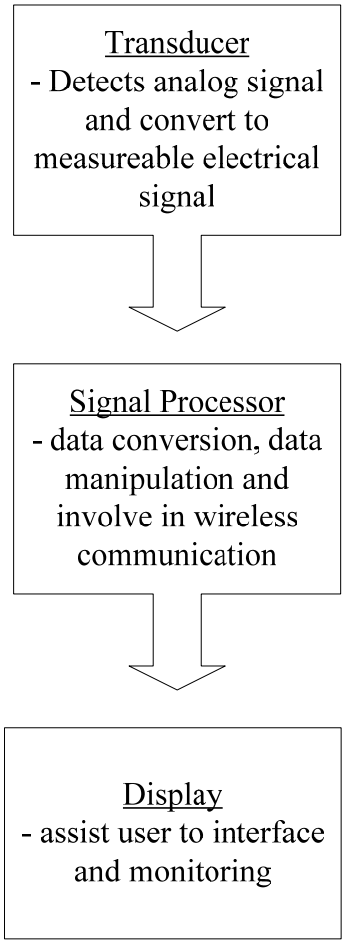


Figure 3.2: The Basic Block Diagram

The main feature of this project would be the wireless communication where it replaces the cable as the conventional data communication type. Nevertheless, the procedures to implement the device are not easy where it needs sufficient understanding on how it operates.

3.3 Phase #2 - Plan The Hardware And Software

With the Phase 1 has been fully accomplished, the next task is performing the Phase 2. It can be started by extracting the basic idea of the project based on the Figure 3.2. The first box of the block diagram is the Transducer box which defines a device that is capable to convert a specific analog signal into a measureable electrical signal. This project has two types of transducer which are the temperature sensor and the potentiometer.

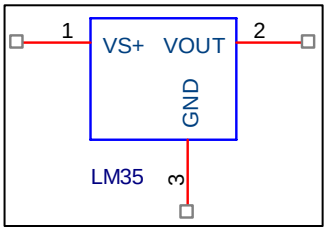
3.3.1 Temperature Sensor

A temperature sensor works by detecting the temperature surrounding and convert it into an amount of voltage. For this project, the temperature sensor used is the Precision Centigrade Temperature Sensor, LM35DZ from National Semiconductor. It generates output in voltage linearly proportional to the Celsius (Centrigrade) temperature. The main features of LM35DZ are :

- the scale factor is linear + 10.0 mV/ °C
- it does not require any external calibration or trimming and maintains an accuracy of +/- 0.4 °C at room temperature and +/- 0.8 °C over a range of 0°C to +1000°C
- it draws only 60 μA from its supply and possesses a low self-heating capability.

This equation is used to convert the output voltage into temperature detected:

$$Temperature, °C = V_{out} * 100°C / V \qquad (3.1)$$

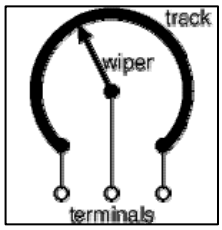


(a)

Figure 3.3: LM35DZ Schematic Diagram

3.3.2 Potentiometer

Potentiometer also known as a variable resistor typically it's consist of three terminals which the track is connected at both ends and a wiper which is the moving terminal. All the terminals are used and it works likes a voltage divider where it capable of providing a voltage range from zero up to the maximum of the power supply. As the wiper is being tuned, what actually happens is that the resistance values (R1 and R2 on Figure 3.5) are adjusted and produce a new output voltage.



(a)

Figure 3.4: Potentiometer Schematic Diagram

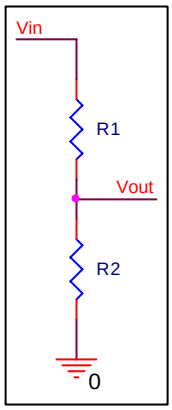


Figure 3.5: Voltage Divider Circuit

Moving on the second box of the block diagram is the Signal Processor box which involve the Analog to Digital Converter 0816 (ADC 0816) Integrated Circuit (IC), Analog Multiplexer DG406 chip, Hex Inverter (IC 74HC04), Line Driver (IC 74HCT541), D Flip-Flop (IC 74LS74), EIA-232 Driver/Receiver (IC MAX 232), Microcontroller (IC AT89S52) and Radio Frequency (RF) Module.

3.3.3 IC ADC 0816

This device which manufactured from National Semiconductor is an 8 bit ADC and has sixteen channels multiplexer to detect 16 different analog signals. Based on Figure 3.6**,** the address channel is selected via four address pins, A, B, C and D. The channel will only be activated if only the Expansion Control is set to high (1).

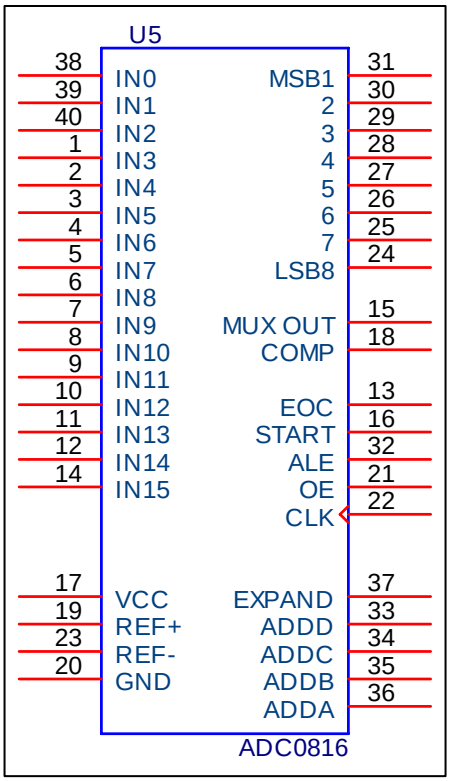


Figure 3.6: IC ADC 0816 Pin Configuration

D	C	B	A	EC	CHANNEL
0	0	0	0	1	IN0
0	0	0	1	1	IN1
0	0	1	0	1	IN2
0	0	1	1	1	IN3
0	1	0	0	1	IN4
0	1	0	1	1	IN5
0	1	1	0	1	IN6
0	1	1	1	1	IN7
1	0	0	0	1	IN8
1	0	0	1	1	IN9
1	0	1	0	1	IN10
1	0	1	1	1	IN11
1	1	0	0	1	IN12
1	1	0	1	1	IN13
1	1	1	0	1	IN14
1	1	1	1	1	IN15

Table 3.1: ADC 0816 Address Selection

To obtain data from ADC 0816, the Address Latch Enable (ALE), Output Enable (OE) and Start Conversion (SC) need to be initially set to low (0) and End Of Conversion (EOC) is also needs to be initially set to high (1). Next, an analog channel is selected by activate the address pins A, B, C and D according Table 3.1. After that, ALE pin is activated in order to latch in the address. Then, the SC pin is activated to initiate conversion. After that, EOC is monitor to see whether conversion is done. If it is low (0), the conversion is done. Finally, EOC and OE are set to high (1) to read the digital data.

3.3.4 16 Channel Analog Multiplexer (IC DG406)

This chip from Maxim is function as a 16 analog multiplexer where it has been redesigned with new features; guaranteed matching between channel and flatness over the specified signal range. It is also a low on-resistance muxes where it can perform well in either direction and guaranteed low charge injection. Moreover, it produce low input off-leakage current over temperature around below than 5nA at +85°C.

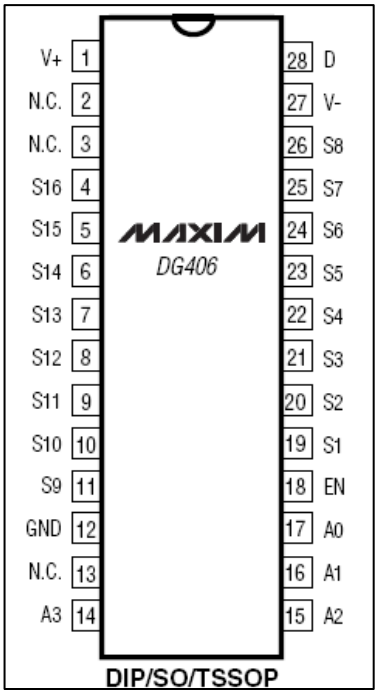


Figure 3.7: IC DG406 Pin Configuration

D	C	B	A	EC	CHANNEL
0	0	0	0	1	IN0
0	0	0	1	1	IN1
0	0	1	0	1	IN2
0	0	1	1	1	IN3
0	1	0	0	1	IN4
0	1	0	1	1	IN5
0	1	1	0	1	IN6
0	1	1	1	1	IN7
1	0	0	0	1	IN8
1	0	0	1	1	IN9
1	0	1	0	1	IN10
1	0	1	1	1	IN11
1	1	0	0	1	IN12
1	1	0	1	1	IN13
1	1	1	0	1	IN14
1	1	1	1	1	IN15

Table 3.2: DG406 Multiplexer Address Selection

3.3.5 EIA-232 Drivers/Receivers (IC MAX232)

This device operates as a line driver (voltage converter) to supply Electronics Industry Association 232 (EIA-232) voltage level. It converts Recommended Standard 232 (RS232) signal to Transistor-Transistor-Logic (TTL) voltage levels and vice versa with the power source of +5V. For this chip, it has two sets of line drivers for transferring and receiving data and it requires four capacitors ranging from 1 to 22 μF .

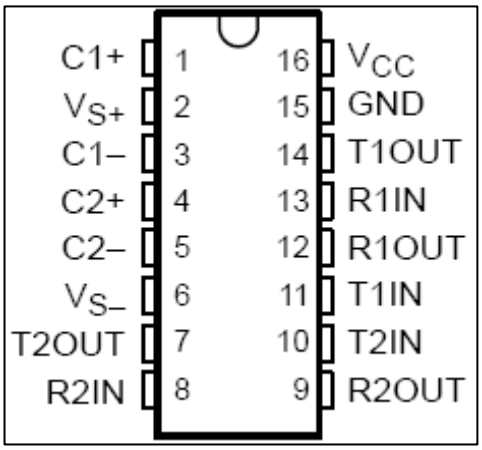


Figure 3.8: IC MAX232 Pin Configuration

3.3.6 Octal Buffer/Line Driver (IC 74HCT541)

This chip is designed to work as a buffer or line driver where it is crucial for data transmission stability. In electronic field, this chip is used to prevent miss identical data during data transmission due to the instability device. Thus, it helps to reduce error in the overall system. Apart from that, this chip holds the responsibility to establish the programming of the Flash – Serial Mode with the conncetion of a microcontroller from Atmel. Typically the pin involved in this programming is the pin 1.5/MOSI, pin 1.6/MISO and pin 1.7/SCK from the microcontroller part and connected to the bus outputs of IC 74HCT541. A detail description on this process is explained later in the circuit elaboration below.

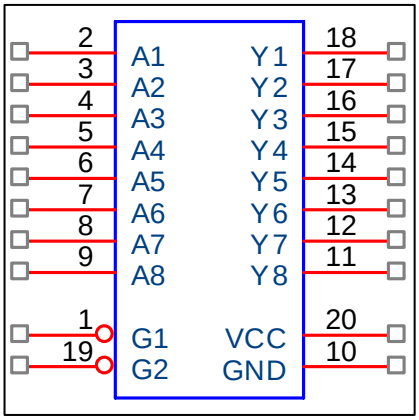


Figure 3.9: IC 74HCT541 Pin Configuration

3.3.7 D Type Positive Edge Flip-Flops (IC 74LS74)

IC 74LS74 is a logic device which has two sets of D flip-flops. In most application, this chip is commonly used to create latency in a system. Since it has two sets of D flip-flop, it can be used with a single chip or by combining a number of this chip for longer latency.

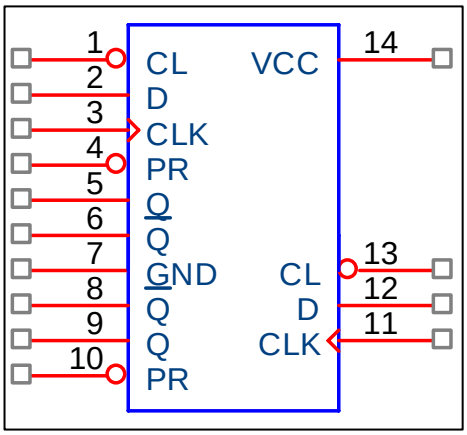


Figure 3.10: D Flip – Flop Chip Pin Configuration (IC74LS74)

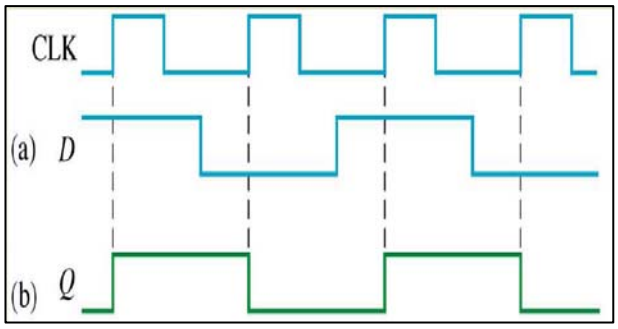


Figure 3.11: D Flip - Flop Timing Diagram

3.3.8 Inverter Gate (IC 74HC04)

This chip performs among the basic logic gates in digital electronics field which is an inverter. When the input is low (0), the output is high (1); when the input is high (1), the output is low (0), thereby producing an inverted output pulse.

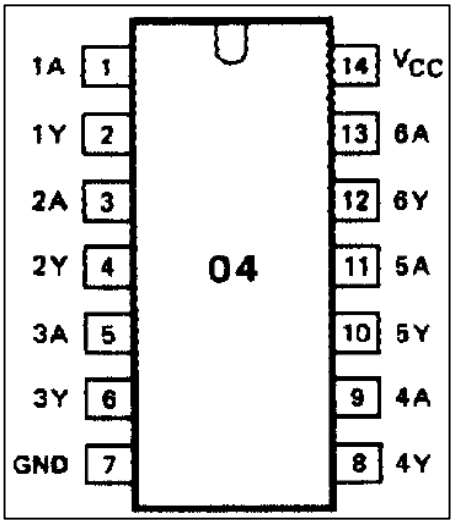


Figure 3.12: IC 74LS04 Pin Configuration

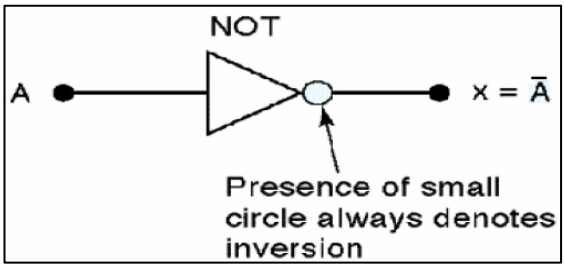


Figure 3.13: Inverter Gate Symbol

3.3.9 Microcontroller Chip (IC AT89S52)

This microcontroller is designed as an 8-bit microcontroller with 8K bytes of in-system programmable Flash memory. The manufacturer, Atmel has fully consider its architecture whereby it is compatible with all MCS®-51 products. Thus, all the instruction set is similar with the Intel 8051 Microcontroller. Apart from that, the user will find this chip helpful with its feature, in-system programmable where the chip is easily programmed by attach with the circuit board. With its 256 x 8-bit internal memory makes the programming more powerful.

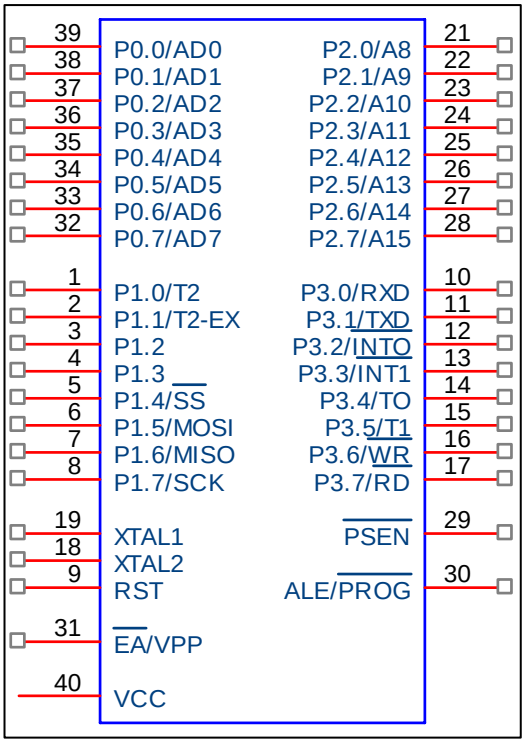


Figure 3.14: IC AT89S52 Pin Configuration

3.3.10 Radio Frequency (RF) Module TLP/RLP 434A

RF transmitter module, TLP 434A and RF receiver module, RLP 434A from Laipac Technology is designed to suit for any wireless applications. It is capable to reach 500ft of range in open area. The typical use frequency is 433.9MHz while 418Mhz and 315Mhz are alternative frequencies. This module used Amplitude Shift Keying (ASK) as the data communication method.

The main feature for the transmitter is that it is based on SAW resonator and capable to accept both linear and digital inputs making the RF development become easier. For the pin description, pin 1 is assigned to Ground (GND), pin 2 is the Data In pin, pin 3 is the VCC pin and the pin 4 is the Antenna (RF Output) pin. As for the receiver, it has 3μV, -103dbm and the typical current is only 3.5mA for 5V operation. As for the pin configuration, pin 1, pin 6 and pin 7 is assigned to GND, pin 2 is the Digital Data Output pin, pin 3 is the Linear Output pin, pin 4 and pin 5 is the VCC pin and pin 8 is the Antenna pin.

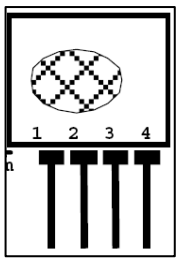


Figure 3.15: RLP 434 SAW Based Receiver Part

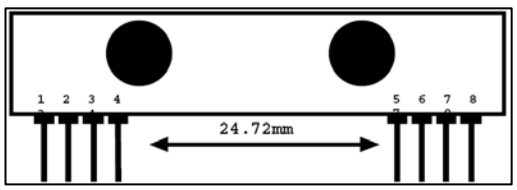


Figure 3.16: TLP 434A Ultra Small Transmitter Part

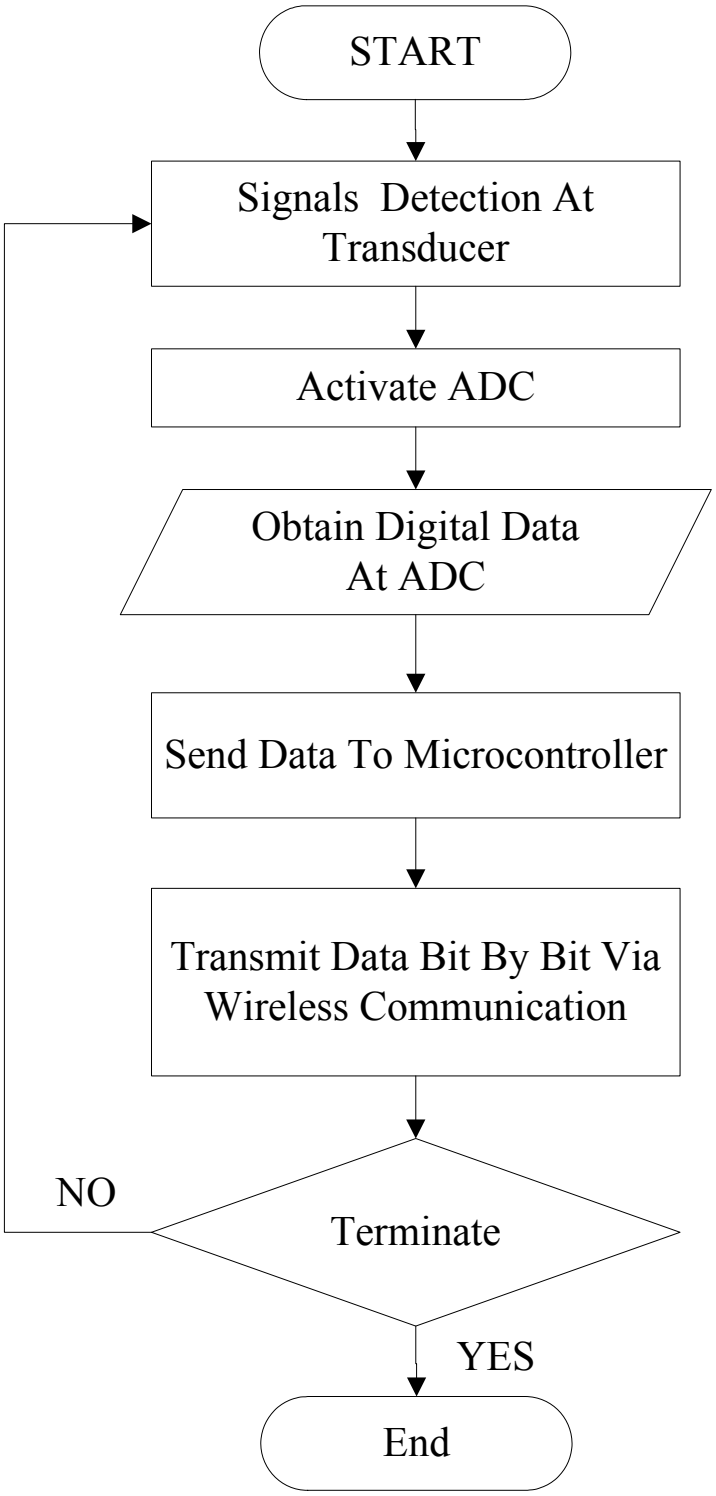


Figure 3.17: Transmitter Part Flow Chart

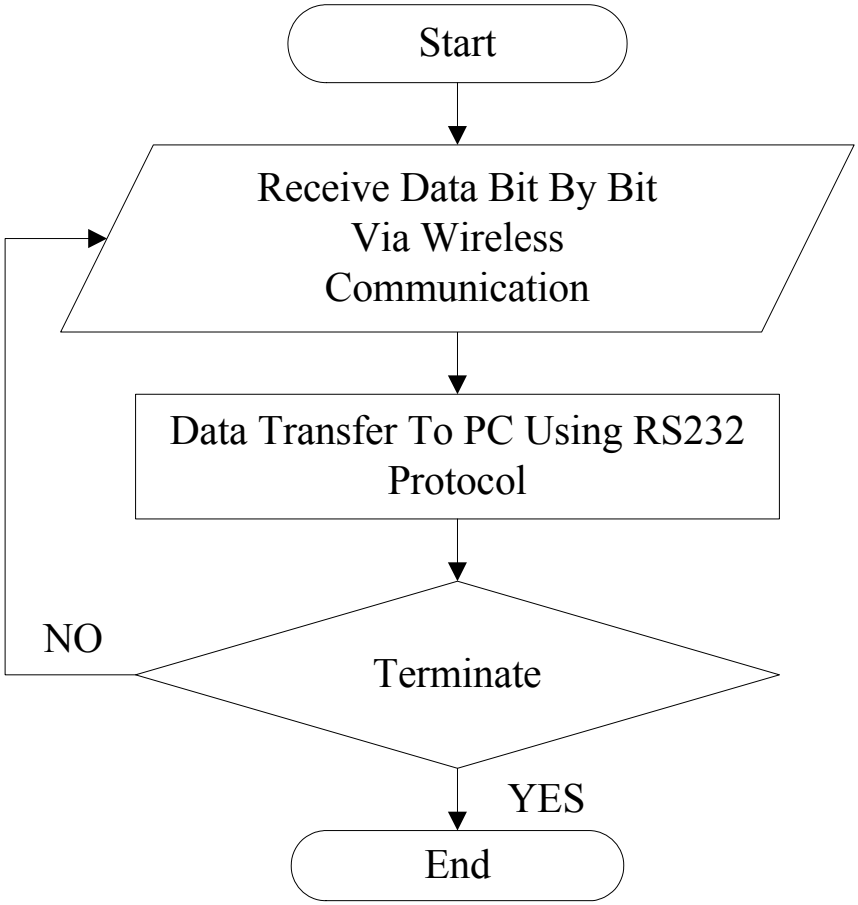


FIGURE 3.18: Receiver Part Flow Chart

As the specifications and features of the major components above are being studied, the next step is to design the system. For a wireless communication to be establish, a transmitter and a receiver needs to be develop. Thus, Figure 3.17 and Figure 3.18 are the flow chart for the transmitter and the receiver part of this project respectively.

The last box of the block diagram is the Display box where it focus on the interface between user and the project. As the project title implies, the data obtained based on the analog signals needs to be monitor frequently. The most basic display panel is the PC where the data is being display by using a software as the GUI. Therefore, this project used the Visual Basic® 6.0 software from Microsoft® Corporation as the GUI. The flow chart of this task is shown at Figure 3.19. The data is display and monitor in the form of graph.

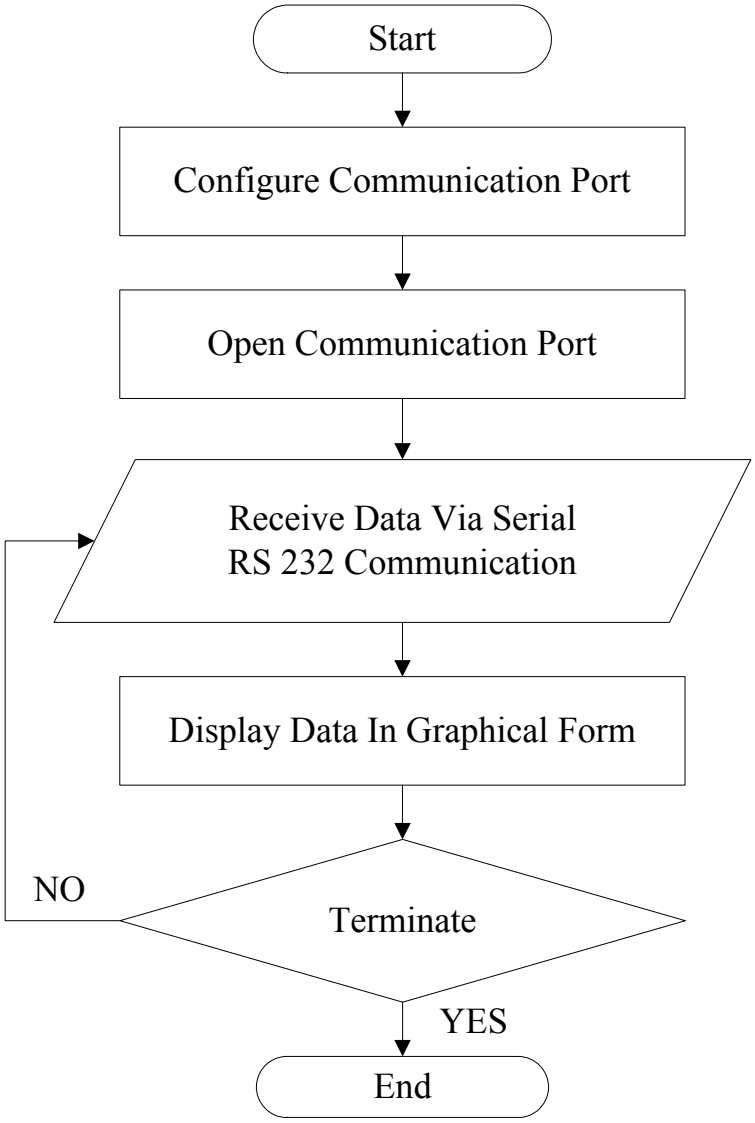


Figure 3.19: GUI Flow Chart

Based on the Figure 3.19, the Serial RS 232 Communication is used to interface the microcontroller board with the PC. In this project, it is the simplex communication where by the data is transmitted in one way from the microcontroller board to the PC. For this purpose, a DB9 connector with cable is required which compatible with the PC communication port. It involved pin 2 for receive data, pin 3 for transmitted data and pin 5 for ground. The connection of pin 2 and pin 3 are switched between the PC communication port and the microcontroller port in order to establish a communication, one side will transmit and another side will receive the data. Figure 3.20 illustrated this statement.

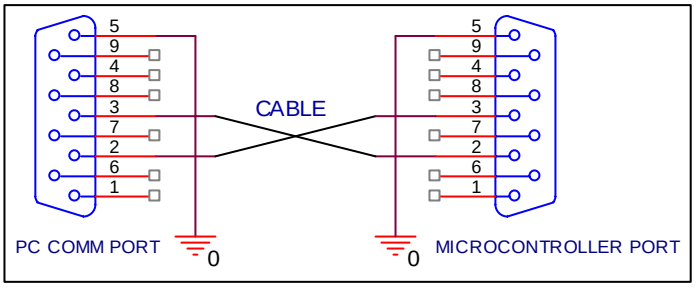


Figure 3.20: Serial RS232 Configuration

3.4 Phase #3 - Design, Implement And Test The Hardware

Referring the flow charts from Phase 2, the hardware development for this project can be done. To operate the AT89S52 microcontroller, the 74HCT541 needs to be designed as an In-System Flash microcontroller programmer. The design is based on the circuit of the In System Programmable (ISP) Flash Microcontroller Programmer from Version 3.0 by Mohammad Asim Khan which is shown at Figure 3.22. Based on that figure, 74HCT541 chip works as an isolator and buffer the parallel port signal. Apart from the circuit diagram, the software for read and write the desired program into the microcontroller via Serial Programming Interface (SPI) is also available for this version. This can be seen at Figure 3.21.

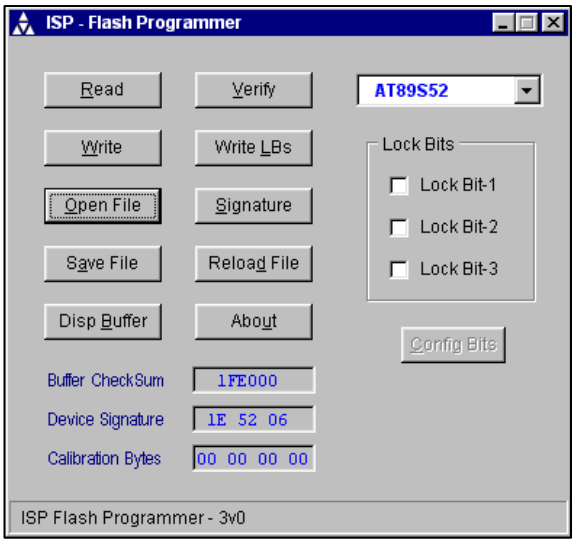


Figure 3.21: ISP Flash Microcontroller Programmer Version 3.0a Software

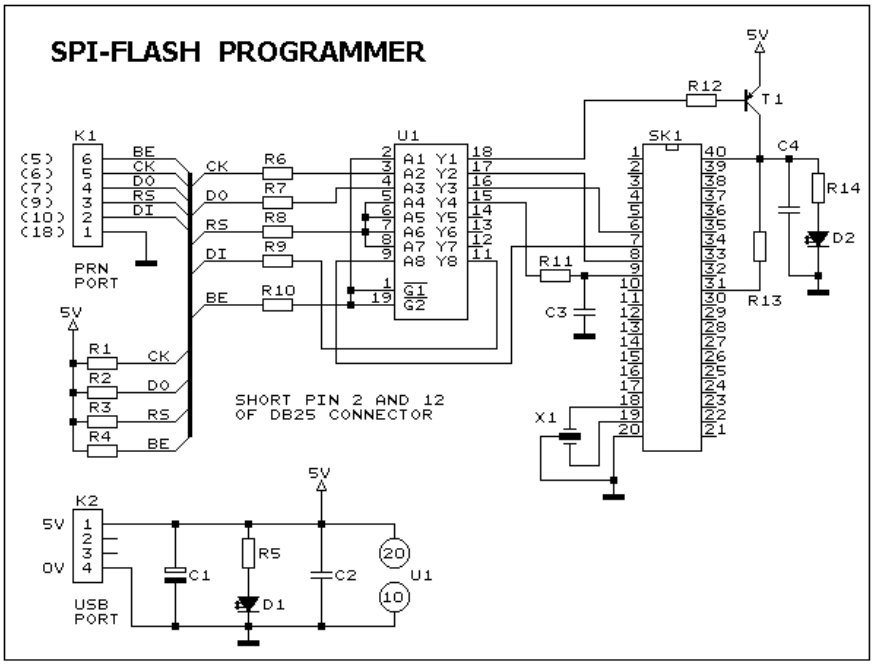


Figure 3.22: ISP Flash Microcontroller Programmer Version 3.0 Circuit Diagram

To download the program from PC to microcontroller, the file must be in HEX file and it involved a cable that has a DB25 connector port (parallel port) to plug into the PC parallel port. Only 8 pins are used which is pin 5, 6, 7, 9, 10, 18 and pin 2 and 25 are short for auto hardware detection. As for the other side, DB9 connector port is required to plug in with the microcontroller board which used the pin 4, 5, 6, 7,8 and 9. Figure 3.23 illustrate the loader cable pin configuration.

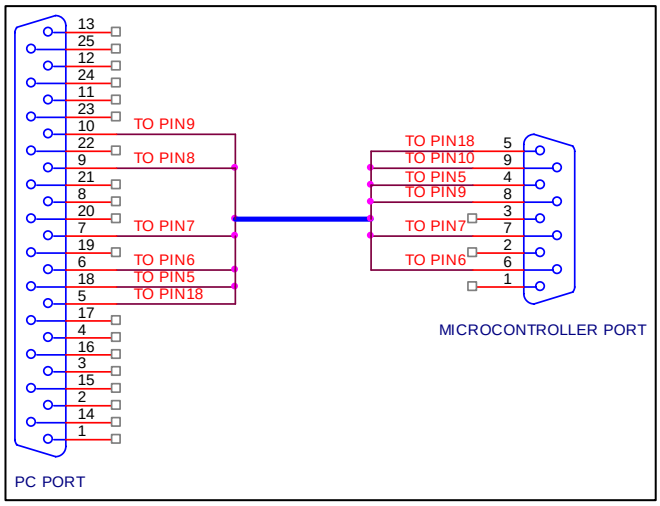


Figure 3.23: Loader Cable Pin Configuration

In order to enable ADC0816 and function in synchronize with AT89S52, 74LS74 chip is needed to supply clock pulse for ADC0816. This is done by tapping from the crystal of the microcontroller. A typical D Flip-Flop 74LS74 chip will divide the frequency by 2 if its $\overline{Q}$ connected with the D input. Based on the ADC0816 datasheet, the typical operating frequency is approximately 640kHz at 5 Volts. For the crystal frequency is 11.0592Mhz. Thus,

$$\frac{11.0592MHz}{2} = 5.5296MHz$$

$$\frac{5.5296MHz}{2} = 2.7648MHz$$

$$\frac{2.7648MHz}{2} = 1.3824MHz$$

$$\frac{1.3824MHz}{2} = 691.2kHz$$

As a result, 4 D flip – flops are required to obtain the operating frequency of ADC0816 as shown at Figure 3.24.

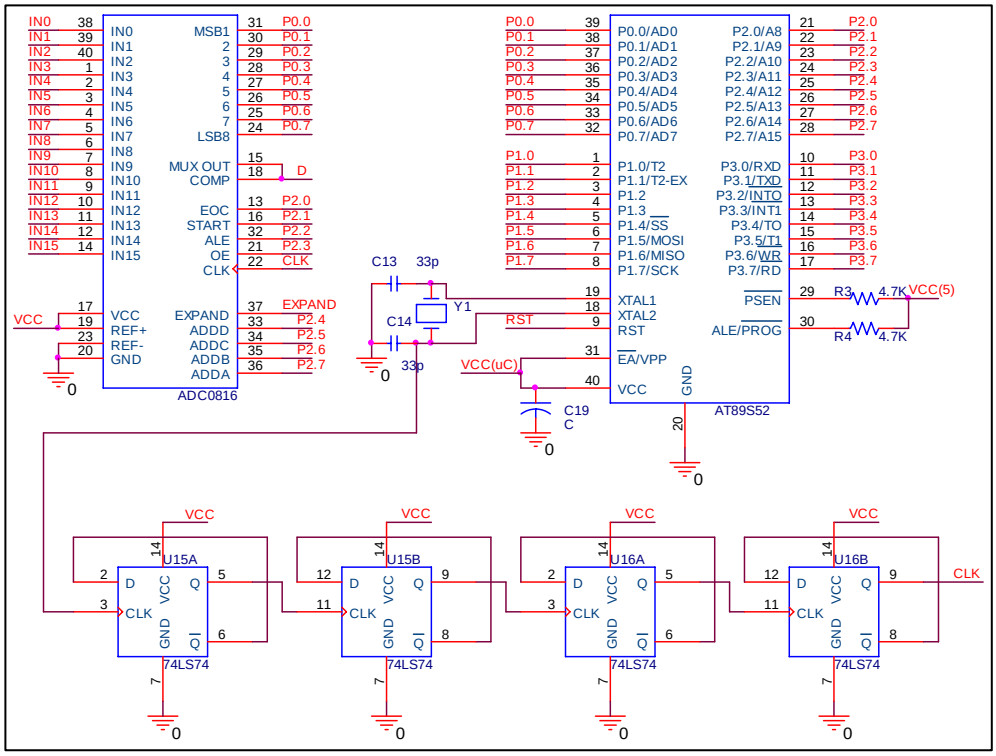


Figure 3.24: ADC0816 With External Clock

To enhance this project, addition analog channels are designed to supply more data for monitoring purpose. As the ADC0816 has 16 analog channels, this project is added with a DG406 analog multiplexer that has 16 analog channels. Thus, a total of 32 analog channels are available for this project and any channel can be selected at one time. To implement these 32 analog channels, a 74HC04 chip is needed as a switch to enable DG406 and disable ADC0816 or vice versa at one time. This chip is control by one of any port signal from the microcontroller where the signal for EC at ADC0816 has to be low (0) to enable DG406. This is done by supplying signal to the Enable (EN) pin at DG406. Appendix list all the address accessible for this project.

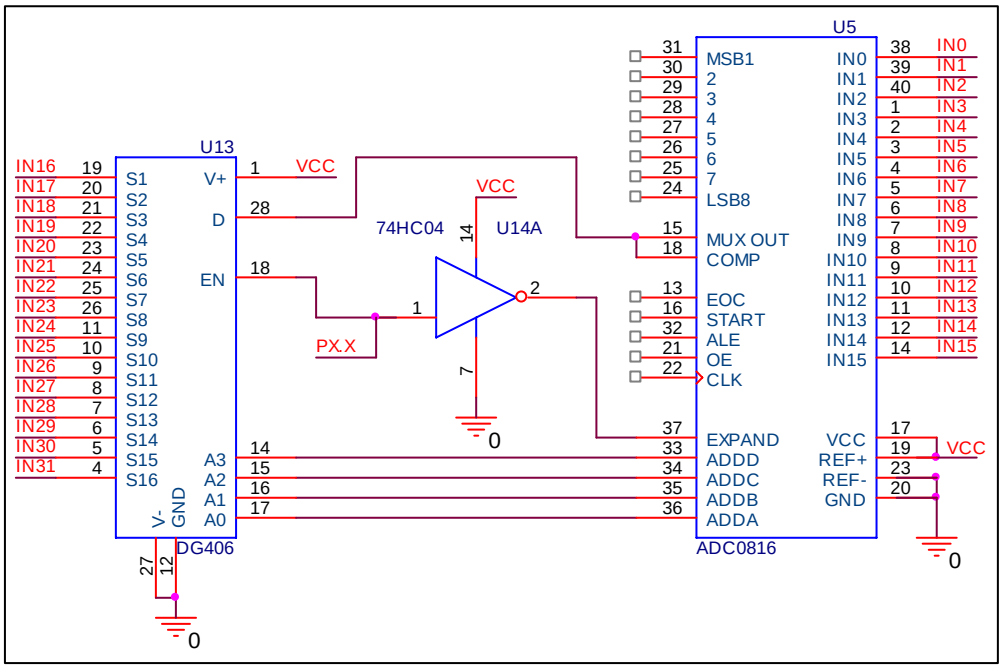


Figure 3.25: 32 Analog Channels Circuit Diagram

This project will not be function without the existence of the AT89S52 Microcontroller. It works as the brain for the entire project where it generates signals to activate and deactivate which device to be enable. Moreover, it is also the medium of temporary data storage for the wireless communication. As the digital data is obtained, it will then be send to the transmitter RF module for wireless data transmission.

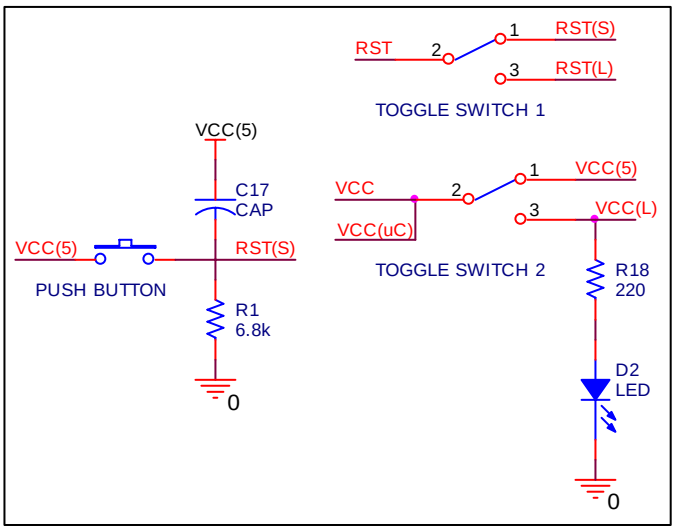


Figure 3.26: The Circuit Switches

At the receiver board, it can seen it has two toggle switches and one push button which the transmitter board is also has one. These parts are important as it acts as the initial procedure for the board's loading and operation. For the board to perform as a loader, the toggle switch 1 must be toggle to RST(L) and toggle switch 2 must be to toggle to VCC(L). As a result; the pin $\overline{EA}/VPP$ at pin 31 of the IC AT89S52 is being attached with GND whereby it enables the microcontroller to fetch code from external program memory (Asim Khan's Software) that starts at 0000H up to FFFFH. Meanwhile, this will also create a connection between pin 1.5(MOSI), pin 1.6(MISO) and pin1.7(SCK) via IC 74HCT541 which enable the serial programming. Moreover, $\overline{PSEN}$ is supplied with low (0) signal where it read strobe to external program memory and the $\overline{PROG}$ is also supplied with low (0) signal which produce the program pulse input for the microcontroller. After done the programming, the toggle switch 1 is toggle to RST(S) and toggle switch 2 is toggle to VCC(5) where it results the programming session is over and $\overline{PROG}$ and $\overline{PSEN}$ is supplied with high (1) signal and the push button can be used to reset the microcontroller during any program is running.

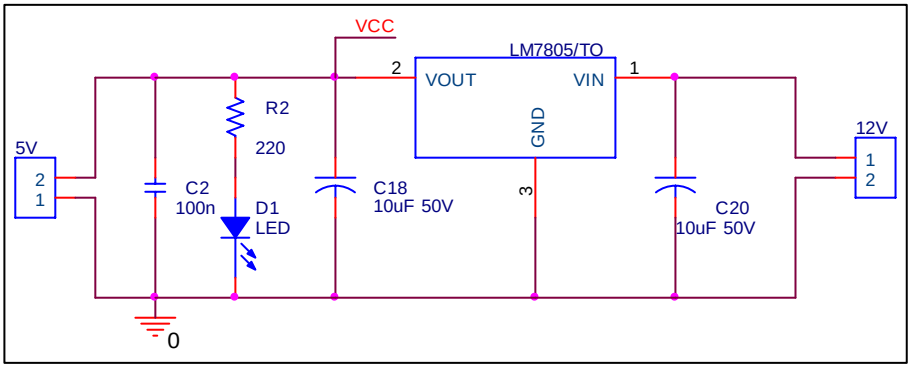


Figure 3.27: Power Supply Circuit

In order any electronic circuit to be operate, a power source must be supply. So, this project utilize two different power supplies, a fix 5V or a 12V. If the board is supplied with a 12V power, then LM7805 will be function as a voltage regulator where it convert the incoming power into a fix 5V power. As a result, the board is capable to be activate with any source power. For safer conversion, the LM7805 needs to be attached with a heatsink in order to absorb and dissipate the heat produce by the power conversion.

The complete schematic diagram for both transmitter and receiver part are shown at Figure 3.28 and Figure 3.29 respectively.

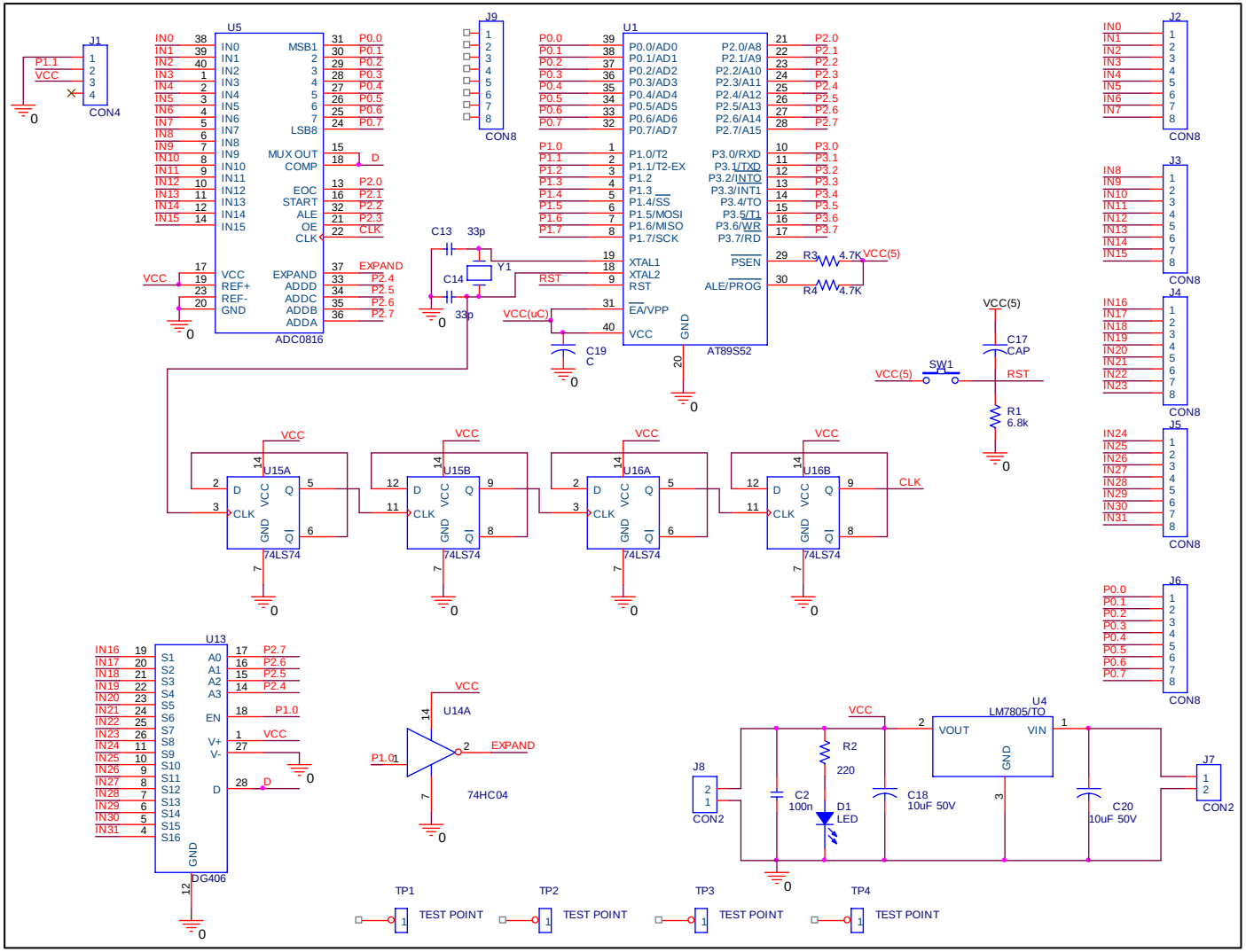


Figure 3.28: The Overall Transmitter Part Circuit

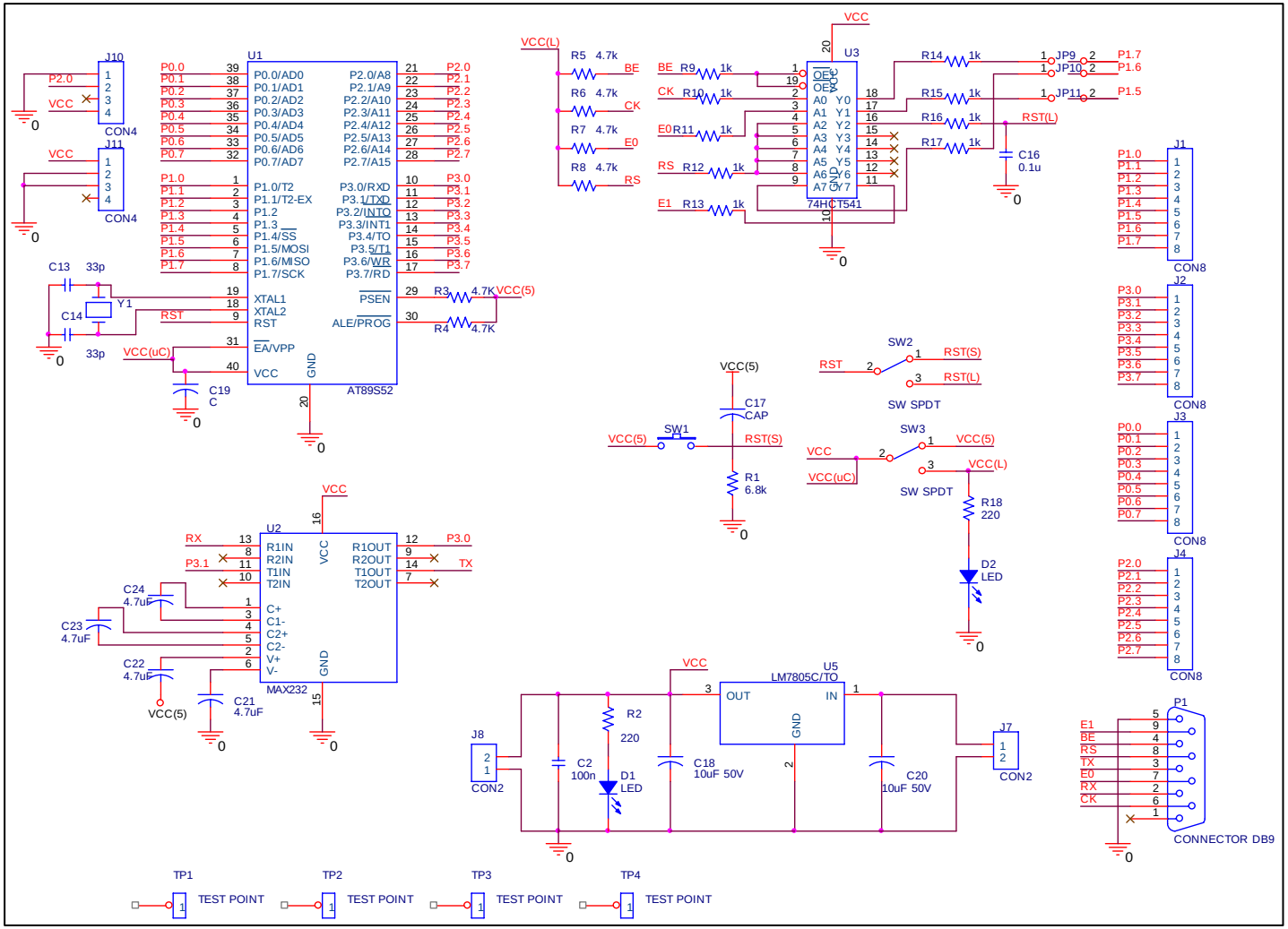


Figure 3.29: The Overall Receiver Part Circuit

3.4.1 Schematic Circuit Design

The transmitter and receiver circuit design is very crucial in this project to increase reliability and simplicity. Initially, a study on the existing ISP board is done in order to get a better view of the circuit's function. With the information gained, the circuit design is started by drawing the schematic on the OrCAD® Capture CIS software. As the drawings are done, the next step is to identify the Printed Circuit Board (PCB) footprint that is best fit with all the components used. It is very important to have the exact and correct PCB footprint in order to prevent error during the circuit construction. To identify the desired PCB footprint, choose OrCAD® Layout and click Tools>Library Manager. It is much easier if the components are already in hand so that it can be used to select its PCB footprint exactly. The list of PCB footprints used in this project can be refer on Appendix A. As the final touch for the schematic drawing, the procedures on Figure 3.30 have to be done. This is for the initial step for PCB layout.

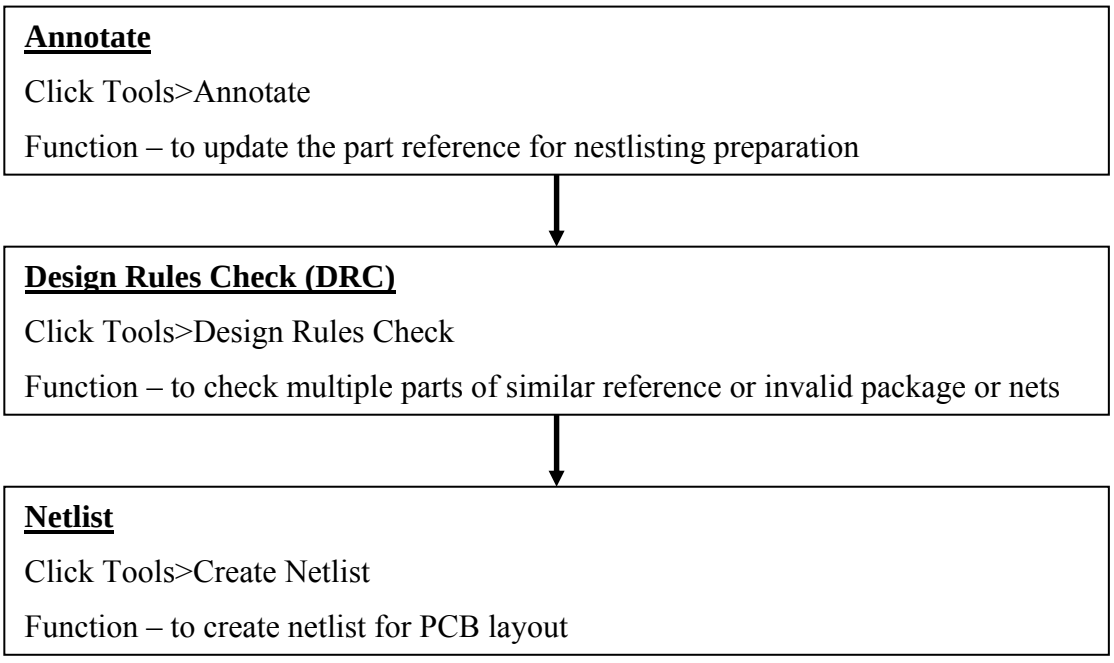
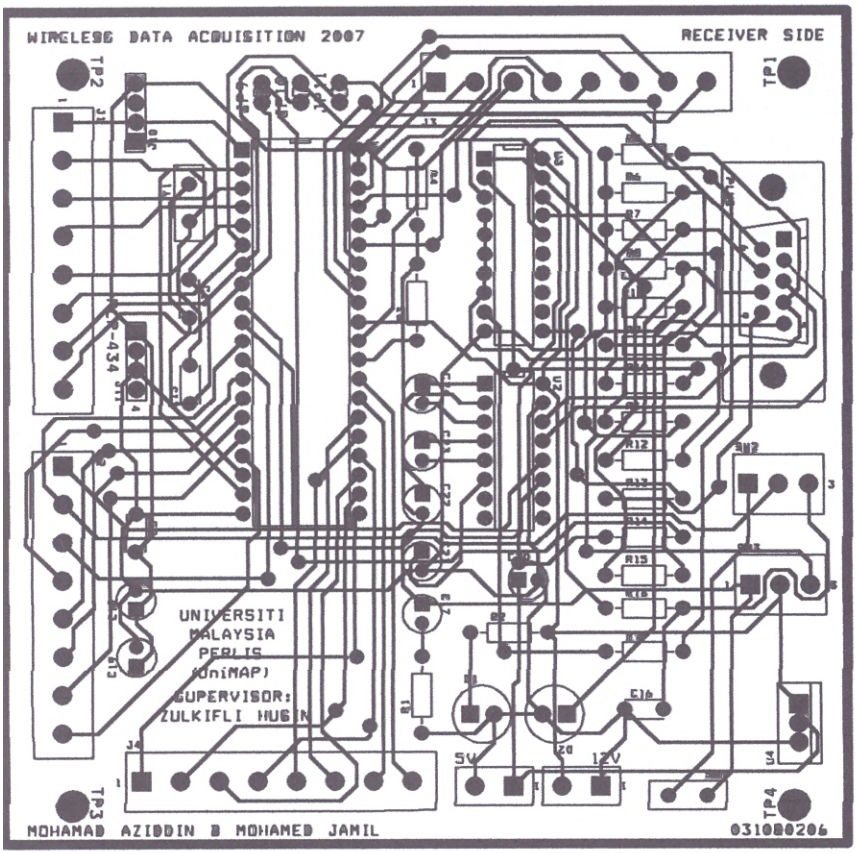


Figure 3.30: Final procedures for schematic drawing

3.4.2 PCB Fabrication

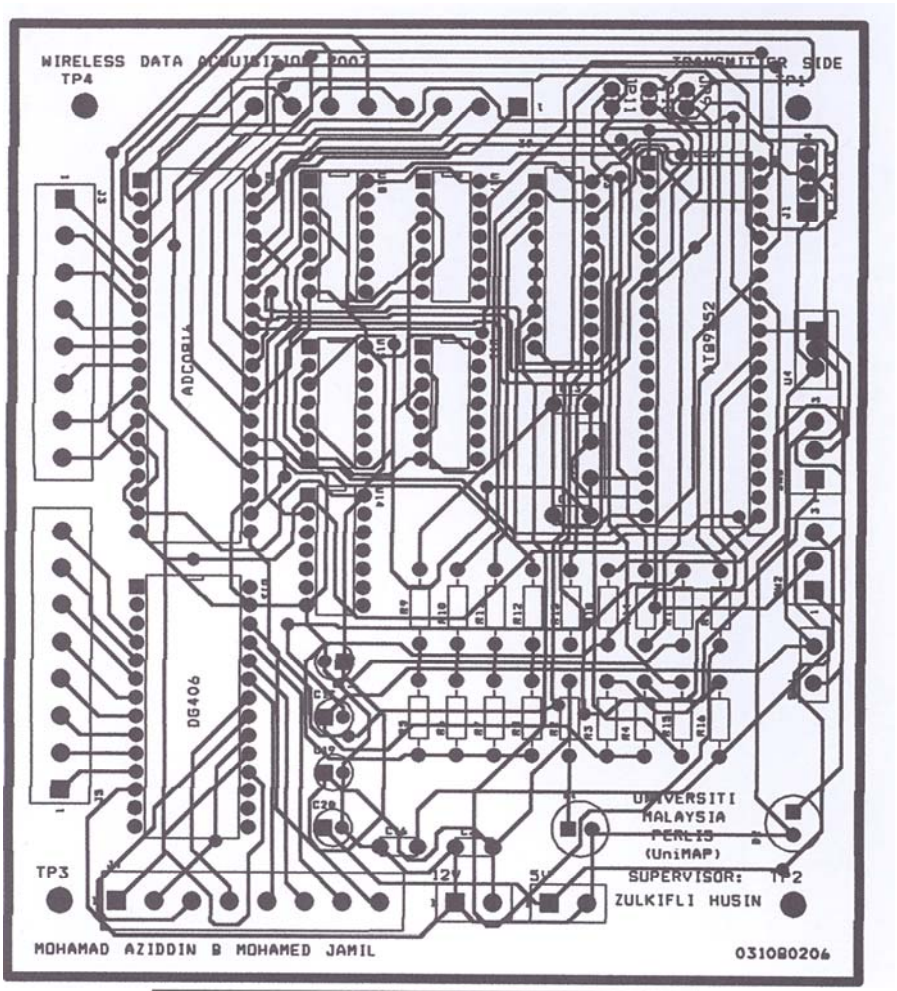
This project used OrCAD Layout software to make the PCB which is a double layer board. Since both layer are used, thus much consideration must be taken on its size of trace, space width, via (connection between top layer and bottom layer) and size of hole which will be drill. The procedure starts by choosing OrCAD Layout program and load the netlist created from OrCAD Capture CIS. As no error occurred during AutoECO process, the layout design can be started by clicking the Component Tool at the control panel. This will enable the components to be arranged as desired design.



DRILL CHART				
SYM	DIAM	TOL	QTY	NOTE
×	0.030		31	
+	0.034		140	
◇	0.040		11	
■	0.048		18	
⊡	0.055		45	
⁂	0.120		4	
TOTAL			249	

Figure 3.31: The Receiver Layout

The arrangement is crucial where the components need to be in the right place to reduce the wire path in order to avoid stressful fabrication. After the arrangement had done, the next step is to route the board. It is best done by using SmartRoute Tool at the Layout tool. The advantage of this tool is that it reduces the number of via and create better route compared to Autoroute Board at the Auto command of Layout. The transmitter layout and the receiver layout are displayed on Figure 3.31 and Figure 3.32 respectively.



DRILL CHART				
SYM	DIAM	TOL	QTY	NOTE
◇	0.030		27	
+	0.034		246	
■	0.048		22	
⊞	0.055		45	
⁂	0.120		4	
TOTAL			344	

Figure 3.32: The Transmitter Layout

As for the Liquid Crystal Display (LCD) board, the fabrication method is differs compared to the transceiver board. The method used is self-fabrication where it can be done in our own house. The first step is to have the layout being go through the Run Post Processor process and then duplicate it by photocopy on an Overhead Projector (OHP) paper. Next, the layout is being ironed on an empty PCB board until all the layout is approximately transferred to the PCB. Next, the PCB will then go through the etching process where acid is used to discard the unwanted copper. The result is the desired layout and it is proceed by the drilling holes and soldering the components. Finally, the PCB is being coated with a clear layer to avoid rust on the copper.

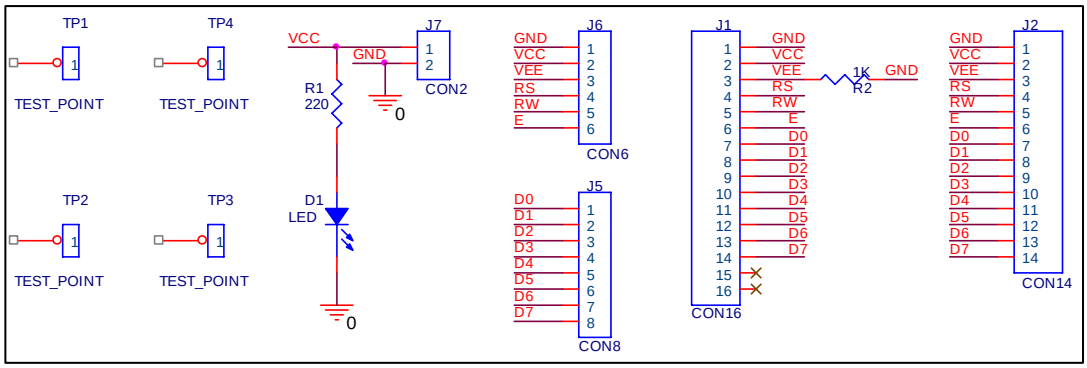


Figure 3.33: LCD Schematic Diagram

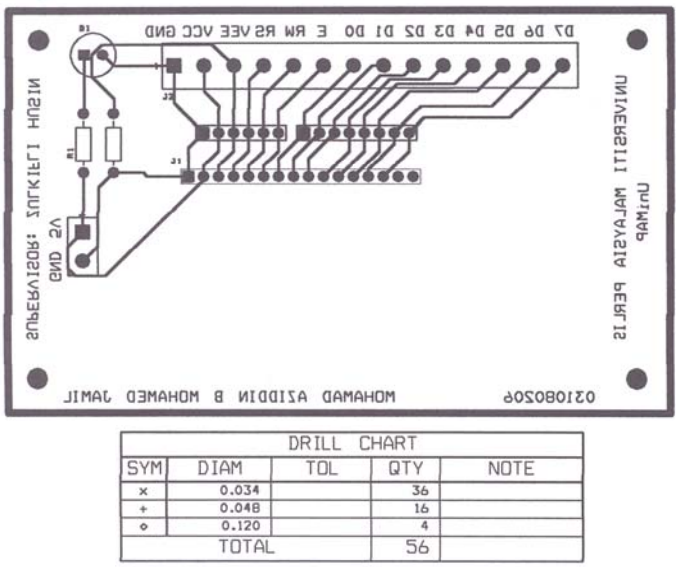


Figure 3.34: LCD PCB Layout

3.5 Phase #4- Design, Develop And Test The Software

In this phase, the main task is to develop the software part that will run the project entirely. It is done part by part in order to detect any error or malfunction along with the hardware part which has completed. The GUI development used the Visual Basic® 6.0 software while the programming code is done in C language via Keil MicroVision®3 software trial version.

3.5.1 The GUI Development

The GUI is one of the important part for this project as it display the data from microcontroller for data monitoring and analysis. The design template has to be user friendly for best usage. For this project, the main objective is to display data received in graphical form. As a transducer detects and translate an analog signal, the data will go through a conversion at the ADC and become a digital data format. This digital data is converted in an eight bit data. This data will be send using the wireless data transmission between the transceiver board. Next, it will reach a PC and being manipulated to be display in graphical form with the help of Visual Basic® 6.0 software.

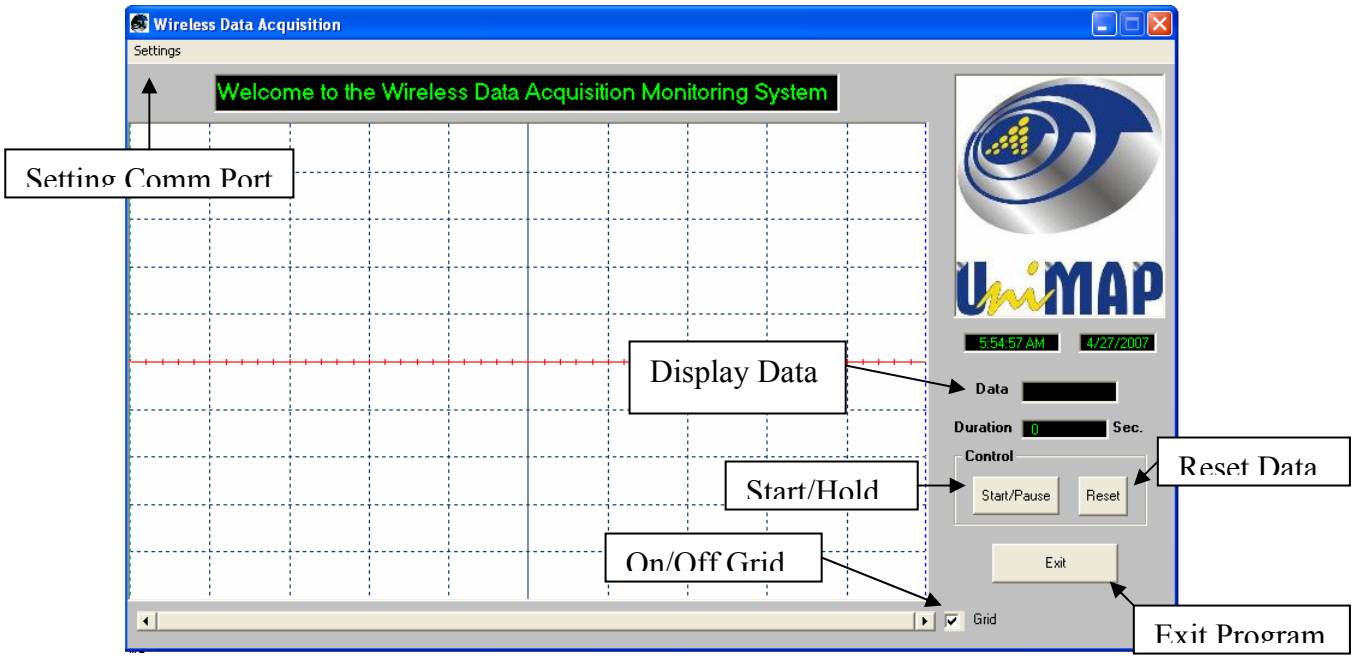


Figure 3.35: GUI Monitoring System Panel

Since the data used the serial RS232 communication, therefore an initialization needs to be done which are the baud rate,data bits, parity, stop bit and the COM port at the PC. The baud rate is the number of signal changes per second or transition speed between Mark (negative) and Space (positive) which range from 110 to 19200, data bits is the length of data in bit which has one Least Significant Bit (LSB) and one Most Significant Bit (MSB), the parity bit is an optional bit mainly for bit error checking. It can be odd, even, none, Mark and Space. Stop bit is used to frame up the data bits and usually combined with the start bit. These bits are always represented by a negative voltage and can be 1, 1.5 and 2 stop bits. And for the COM port is the selection of the available COM port at the PC. Most PC has 2 COM port which serial RS232 communication compatible. The commonly used setting to establish a serial RS232 communication is 9600 baud rate, none parity, 8 data bits, 1 stop bit and COM port 1. This can be done by clicking 'settings' at the GUI panel.

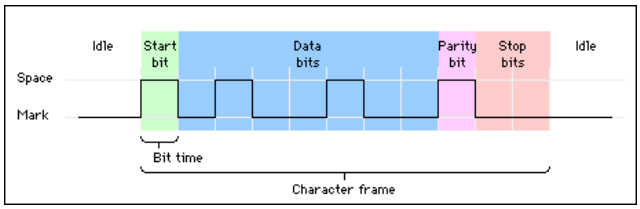


Figure 3.36: Bit Format

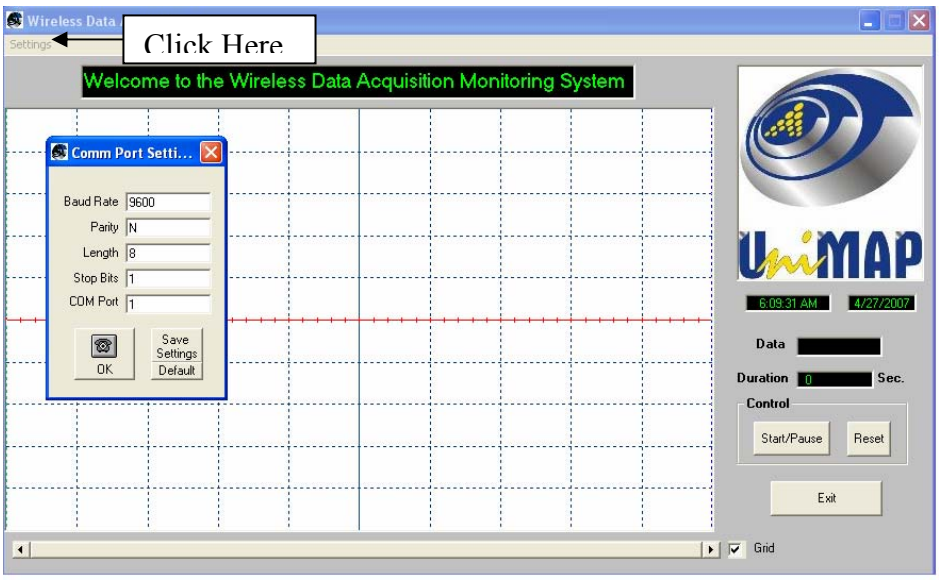


Figure 3.37: Serial RS232 Initialization at the GUI

As this project relates with data collection, thus the data obtained from the microcontroller needs to be collected and saved. This can be done by using the GUI monitoring system where it automatically saves the data received in a notepad. The data being saved is the date and time during the data being collected and data value itself. Figure 3.38 represents this statement. The overall source code for GUI monitoring system can be referred to Appendix A

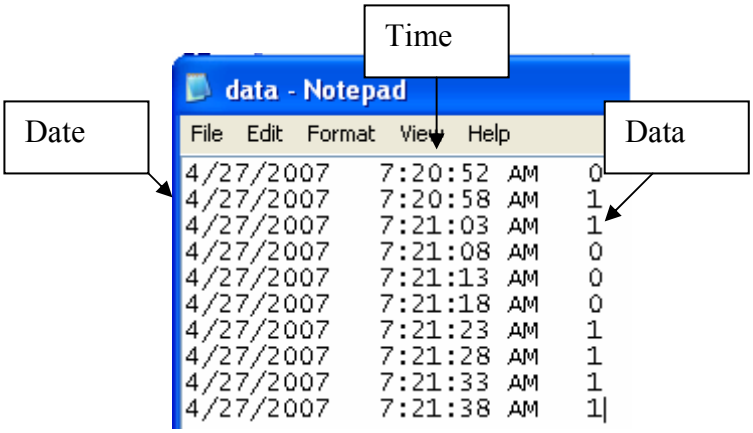


Figure 3.38: Data Obtained Being Saved In A Notepad

3.5.2 Microcontroller Testing

As the boards are being fabricated, a test procedure on the board should be made. This is to identify if any error or malfunction is detected during its operation. The test is by checking all ports of the microcontroller whether it can perform as an input and output (IO) port. The program that will executed is to toggle the test port with high(1) and low(0).

```
#include <reg51.h>
sbit mybit = P2^4;                  //set P2.4 as mybit

void time(unsigned char delay)      /****** DELAY ******/
{
        unsigned char x;            //set x as unsigned character
        for (x=0;x<delay;x++)       //make loop as x value
```

```
        {
                TMOD = 0x20;        //use Timer 1, 8 - bit auto reload
                TH1=0xFD;           //Set baud rate to 9600
                SCON=0x50;          //use serial mode 1,8-bit data,1 stop bit,1 start bit
                TR1=1;              //start timer
                TH0 = 0x4c;         //50ms delay
                TL0 = 0x00;         //set TL0 as 0
                TF0 = 0;            //Clear rollover flag
                TR0 = 1;            //on timer
                while (TF0 == 0);   //monitor TF0
                TR0 = 0;            //off timer
        }
}

void main ()                        /*****main****/
{
        while(1)                    //repeat forever
        {
                        mybit =0;   //toggle mybit to 0
                        time(100);  // call delay 100 times
                        mybit =1;   //toggle my bit to 1
                        time(100);  // call delay 100 times
        }
}
```

Referring to the source code above,a timer is configured to provide a specific delay time by simply setting the starting count and then waiting for the TFx to indicate a rollover-the end of the time delay. The numbers loaded into the counters determine the time delay for one time through the loop. The frequency driving the counters is the system clock divided, so the time per count is obtained. The counters count up from the preset count to 0xFFFF before rolling over and setting the timer flag. Therefore, the counter preload numbers must be subtracted from 0x10000.

To generate a 50ms delay via timer, the calculation below is done.

$$\text{Counter clock frequency} = \frac{systemclock}{12} = \frac{11.0592MHz}{12} = 921.6kHz$$

$$\text{Time per clock count} = \frac{1}{counterclockfrequency} = \frac{1}{921.6kHz} = 1.085\mu s$$

$$\text{Counts for the delay} = \frac{delaytime}{timeperclockcount} = \frac{50ms}{1.085\mu s} = 46{,}080$$

Preload count = 0x10000 – counts for the delay

= 65,536 – 46,080 = 19,456 = 0x4C00

3.5.3 Analog to Digital Converter Testing

This testing is another important part for this project as it involved the brain of the project, the AT89S52 chip. The task is to generate signals from AT89S52 chip to activate the ADC and it will convert the data from the analog channel with address 0000.

```
#include <reg51.h>
sbit ADDR_A= P0^0;                    //set Port0.0 as Address A
sbit ADDR_B= P0^1;                    //set Port0.1 as Address B
sbit ADDR_C= P0^2;                    //set Port0.2 as Address C
sbit ADDR_D= P0^3;                    //set Port0.3 as Address D
sbit OE = P0^4;                       //set Port0.4 as OE
sbit EOC = P0^5;                      //set Port0.5 as EOC
sbit SC = P0^6;                       //set Port0.6 as SC
sbit ALE = P0^7;                      //set Port0.7 as ALE
sfr MYDATA = 0xA0;                    //declare SFR at Port2 = 0xA0
void time(unsigned char delay);

void main ()                          /****** Main ******/
{
```

```
        unsigned char value;                //define value as unsigned character
        P2 = 0xFF;                          // set as input
        SC = 0;                             //disable SC
        ALE = 0;                            //enable ALE
        OE = 0;                             //disable OE
        EOC=1;                              //activate EOC
        while(1)                            //repeat forever
                {
                        ADDR_D = 0;         //set Address D = 0
                        ADDR_C = 0;         //set Address C = 0
                        ADDR_B = 0;         //set Address B = 0
                        ADDR_A = 0;         //set Address A = 0
                        time(1);            //set delay once
                        ALE = 1;            //enable latch address
                        time(1);            //set delay once
                        SC = 1;             // ready to convert
                        time(10);           //set delay 10 times
                        ALE = 0;            //disable ALE
                        SC = 0;             //end of intial signal
                        while(EOC==0);      //monitor conversion period
                        while(EOC==1);      //monitor converted data done
                        OE = 1;             //ready to read data
                        time(10);           //set delay 10 times
                        value = MYDATA ;    //data transfer to P2
                        OE = 0;             //disable OE
                }
}

void time(unsigned char delay)          /****** DELAY ******/
{
        unsigned char x;                //set x as unsigned character
        for (x=0;x<delay;x++)           //make loop as x value
```

```
    {
        TMOD = 0x20;        //use Timer 1, 8 - bit auto reload
        TH1=0xFD;           //Set baud rate to 9600
        SCON=0x50;          //use serial mode 1,8-bit data,1 stop bit,1 start bit
        TR1=1;              //start timer
        TH0 = 0x4c;         //50ms delay
        TL0 = 0x00;         //set TL0 as 0
        TF0 = 0;            //Clear rollover flag
        TR0 = 1;            //on timer
        while (TF0 == 0);   //monitor TF0
        TR0 = 0;            //off timer
    }
}
```

To verify the digital output from ADC, some calculation needs to be done. Since ADC 0816 is an 8 bit resolution, so it has 256 step resolutions where each step size is the smallest change that can be discerned by the ADC. Therefore, the equation below must be used:

$$Dout = \frac{Vin}{Stepsize} \qquad (3.2)$$

where Dout = digital data output (in decimal), Vin = analog input voltage and step size (resolution) = Vref/256

Let say the Vref is 2.56V, thus the step size is 2.56V/256 = 10mV. Next, we used the 10kΩ potentiometer as the transducer. If it is tuned half of the value, it produce the voltage; $Vin = \frac{2.5k\Omega}{10k\Omega} x5V = 1.25V$. Referring the equation 3.2, the estimated Dout is

$$Dout = \frac{1.25V}{10mV} = 125 = 01111101_b$$

As a result, the value of 125 in decimal or 01111101_b should be display at the digital output. To make a clear view, a LED is attached on each the digital output pin of ADC0816 as in Figure X. Therefore, the LED will on and off based on that Figure 3.39.

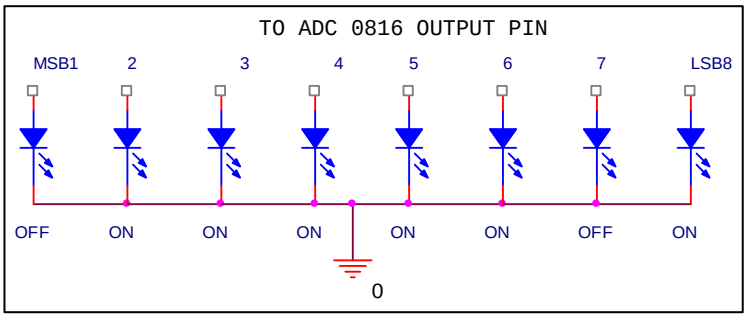


Figure 3.39: Output Verification Using LED

3.5.4 RF Testing Module (RLP434A & TLP434A) And Serial Communication

RF module acts an important aspect in this project. The test includes both hardware and software part. The test is to send low (0) and high (1) signal from port 2.0 of AT89S52 chip to the RF transmitter part continuously. After that, the signal will be transmitted wirelessly to the RF receiver part. To verify the signal transmission, a Light Emitting Diode (LED) is located at the Data In pin at the RF transmitter part and at the Data Out pin at the receiver part. As for the serial communication testing, MAX232 chip is used as a line driver which send the data received to the PC and display on the HyperTerminal© or by using the GUI.

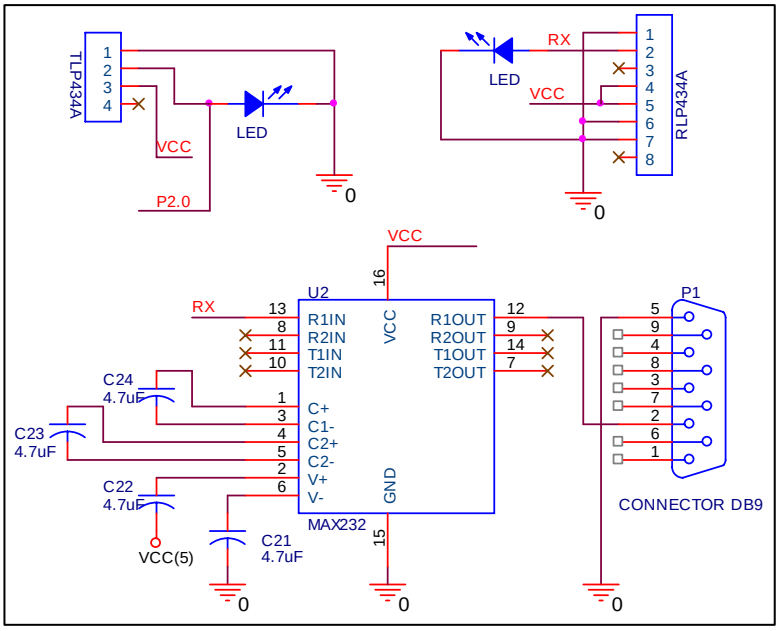


Figure 3.40: RF Module And Serial Communication Test Circuit

```
#include <reg51.h>
sbit mybit = P2^0;                  //Declare P2.0
void main ()                        /****** MAIN ******/
{
        while(1) {                  //Repeat forever
                mybit =0;           //Toggle P2.0 to low
                time(100);
                SBUF=mybit;         //Send data to SBUF register
                while(TI==0);       //wait until transmitted
                TI=0;
                mybit =1;           //Toggle P.0 to high
                time(100);
                SBUF=mybit;
                while(TI==0);
                TI=0;}
}
void time(unsigned char delay)      /****** DELAY ******/
```

```
{ unsigned char x;
 for (x=0;x<delay;x++){
                TMOD = 0x20;        //use Timer 1, 8 - bit auto reload
                TH1=0xFD;           //Set baud rate to 9600
                SCON=0x50;          //use serial mode 1,8-bit data,1 stop bit,1 start bit
                TR1=1;              //start timer
                TH0 = 0x4c;         //50ms delay
                TL0 = 0x00;         //set TL0 to 0
                TF0 = 0;            //Clear rollover flag
                TR0 = 1;            //on timer
                while (TF0 == 0);   //monitor TF0
                TR0 = 0;}           //off timer
}
```

3.5.5 LCD Display Testing

In this project, the LCD is used to display the readings from ADC. To activate the LCD, it must be initialize to set the display for an operating mode to fit the project. For testing purpose, the LCD is initialized to display the word 'TEST' at the first line and increment to the right.

```
#include <reg51.h>
sfr ldata = 0xA0;                   //set data from P2
sbit rs = P3^5;                     //set Port3.5 as RS
sbit rw = P3^6;                     //set Port3.6 as RW
sbit en = P3^7;                     //set Port3.7 as EN
void lcdcmd(unsigned char value);
void lcddata(unsigned char value);
void time(unsigned char delay);
```

```
void main()                                  /****main****/
{
        lcdcmd(0x38);                        // 2lines and 5x7 matrix
        time(5);                             //set delay 5 times
        lcdcmd(0x0E);                        //display on,cursor blinking
        time(5);                             //set delay 5 times
        lcdcmd(0x01);                        //clear display screen
        time(5);                             //set delay 5 times
        lcdcmd(0x06);                        //increment cursor to the right
        time(5);                             //set delay 5 times
        lcdcmd(0x80);                        //force cursor to blink at 1st line
        time(5);                             //set delay 5 times
        lcddata('T');                        //send ASCII T
        time(100);                           //set delay 100 times
        lcddata('E');                        //send ASCII E
        time(100);                           //set delay 100 times
        lcddata('S');                        //send ASCII S
        time(100);                           //set delay 100 times
        lcddata('T');                        //send ASCII T
        time(100);                           //set delay 100 times
}

void lcdcmd(unsigned char value)             /******Command Subroutine******/
{
        ldata = value;                       //send ASCII data
        rs = 0;                              //set RS as 0
        rw = 0;                              //set RW as 0
        en = 1;                              //set EN as 1
        time(5);                             //set delay 5 times
        en = 0;                              //set EN  as 0
        return;
}
```

```
void lcddata(unsigned char value)                /******Data Subroutine******/
{
        ldata = value;                          //send ASCII data
        rs = 1;                                 //set RS as 1
        rw = 0;                                 //set RW as 0
        en = 1;                                 //set EN as 1
        time(10);                               //set delay 10 times
        en = 0;                                 //set EN as 0
        return;
}

void time(unsigned char delay)          /****** DELAY ******/
{
        unsigned char x;                //set x as unsigned character
        for (x=0;x<delay;x++)           //make loop as x value
        {
                TMOD = 0x20;            //use Timer 1, 8 - bit auto reload
                TH1=0xFD;               //Set baud rate to 9600
                SCON=0x50;              //use serial mode 1,8-bit data,1 stop bit,1 start bit
                TR1=1;                  //start timer
                TH0 = 0x4c;             //50ms delay
                TL0 = 0x00;             //set TL0 as 0
                TF0 = 0;                //Clear rollover flag
                TR0 = 1;                //on timer
                while (TF0 == 0);       //monitor TF0
                TR0 = 0;}               //off timer
        }
}
```

3.6 Phase #5 - Test the overall system operationally

As the entire test on hardware and software has been completed, the final phase is to operate the overall project. This phase is crucial to assure that every aspects of the project work as intended.

```
#include <reg51.h>
sbit ADDR_A= P0^0;                  //set Port0.0 as Address A
sbit ADDR_B= P0^1;                  //set Port0.1 as Address B
sbit ADDR_C= P0^2;                  //set Port0.2 as Address C
sbit ADDR_D= P0^3;                  //set Port0.3 as Address D
sbit OE = P0^4;                     //set Port0.4 as OE
sbit EOC = P0^5;                    //set Port0.5 as EOC
sbit SC = P0^6;                     //set Port0.6 as SC
sbit ALE = P0^7;                    //set Port0.7 as ALE
sfr MYDATA = 0xA0;                  //declare SFR at Port2 = 0xA0
sbit rs = P1^7;                     //set Port1.7 as RS
sbit rw = P1^6;                     //set Port1.6 as RW
sbit en = P1^5;                     //set Port1.5 as EN
sbit rf = P3^0;                     //set Port3.0 as RF
sbit regALSB = ACC^0;               //least significant bit 1st

void lcdcmd(unsigned char value);
void lcddata(unsigned char value);
void time(unsigned char delay);
void vb(unsigned char number);
void display(unsigned char digital);

void main ()                        /****main****/
{
        unsigned char store,r,s,t,u;
        MYDATA = 0xFF;              // set as input
        SC = 0;
```

```
        ALE = 0;
        OE = 0;
        EOC=1;
        while(1)
                {
                        ADDR_D = 0;             //set Address D = 0
                        ADDR_C = 0;             //set Address C = 0
                        ADDR_B = 0;             //set Address B = 0
                        ADDR_A = 0;             //set Address A = 0
                        ALE = 1;                //enable latch address
                        time(2);                //set delay twice
                        SC = 1;                 // ready to convert
                        time(2);                //set delay twice
                        ALE = 0;                //disable ALE
                        SC = 0;                 //end of intial signal
                        time(2);                //set delay twice
                        while(EOC==0);          //monitor converted period
                        while(EOC==1);          //monitor conversion data done
                        OE = 1;                 //ready to read data
                        time(100);              //set delay 100 times
                        store = MYDATA ;        //digital data transfer to P3
                        time(100);              //set delay 100 times
                        OE = 0;                 //disable RD for next round
                        vb(1);                  //display pc
                        r = (MYDATA / 10);      //convert HEX to ASCII
                        s = (MYDATA % 10) + 0x30;
                        t = (r % 10) + 0x30;
                        display(s);             //display at LCD
                        display(t);             //display at LCD
                }
}
```

```
void transmit(unsigned char MYDATA)              /***Transmit subroutine***/
{
        unsigned char x;
        ACC = MYDATA;                            //send data to accumulator
        for(x=0;x<8;x++)                         //set 8 loops
        {
                TMOD = 0x20;                     //use Timer 1, 8 - bit auto reload
                TH1=0xFD;                        //Set baud rate to 9600
                SCON=0x50;          //use serial mode 1,8-bit data,1 stop bit,1 start bit
                TR1=1;                           //start timer
                rf = regALSB;                    //send data bit by bit
                ACC = ACC >> 1;                  //shift right
                SBUF= ACC;                       //insert signal in buffer
                while(TI==0);                    //wait till transmitted
                TI=0;                            //disable TI
                time(10);                        //set delay 10 times
                time(10);                        //set delay 10 times
        }
}
void vb(unsigned char number)                    /****VB subroutine****/
{
        unsigned char y;                         //set y as unsigned char
        for (y=0;y<number;y++)                   //set loop
        {
                TMOD = 0x20;//initialize data transfer to serial port T1,8bit auto reload
                TH1=0xFD;                        //9600 baud rate
                SCON=0x50;                       //initialize Serial CONtrol
                TR1=1;                           //start timer
                SBUF=MYDATA;                     //digital data into sbuf
                while(T1==0);                    //wait till tx
                T1=0;                            //disable timer
                time(100);                       //set delay 100 times
```

```
        }
}
void time(unsigned char delay)                  /****delay subroutine****/
{
        unsigned char x;                        //define x as unsigned char
        for (x=0;x<delay;x++)                   //set loop
        {
                TH0 = 0x4c;                     //4C=50ms
                TL0 = 0x00;                     //Clear rollover flag
                TF0 = 0;                        //Set TF0 to 0
                TR0 = 1;                        //on timer
                while (TF0 == 0);               //monitor TF0
                TR0 = 0;}                       //off timer
                }
void display(unsigned char digital)             /****LCD subroutine****/
{
        unsigned char s,t;
        sfr ldata = MYDATA;
        lcdcmd(0x38);                           //2 lines,5x7 matrix
        time(5);                                //set delay 5 times
        lcdcmd(0x0E);                           //display on,cursor blinking
        time(5);                                //set delay 5 times
        lcdcmd(0x01);                           //clear screen display
        time(5);                                //set delay 5 times
        lcdcmd(0x06);                           //increment cursor,to right
        time(5);                                //set delay 5 times
        lcdcmd(0x80);                           //force cursor to begining of 1st line
        time(5);                                //set delay 5 times
        lcddata('D');                           //send ASCII character D
        time(100);                              //set delay 100 times
        lcddata('A');                           //send ASCII character A
```

```
        time(100);                          //set delay 100 times
        lcddata('T');                       //send ASCII character T
        time(100);                          //set delay 100 times
        lcddata('A');                       //send ASCII character A
        time(100);                          //set delay 100 times
        lcddata(':');                       //send ASCII character :
        time(100);                          //set delay 100 times
        lcddata(s);                         //send ASCII character from s
        time(100);                          //set delay 100 times
        lcddata(t);                         //send ASCII character from t
        time(100);                          //set delay 100 times
}
void lcdcmd(unsigned char value)            /******Command Subroutine******/
{
        ldata = value;                  //send ASCII data
        rs = 0;                         //set RS as 0
        rw = 0;                         //set RW as 0
        en = 1;                         //set EN as 1
        time(5);                        //set delay 5 times
        en = 0;                         //set EN  as 0
        return;
}
void lcddata(unsigned char value)           /******Data Subroutine******/
{
        ldata = value;                  //send ASCII data
        rs = 1;                         //set RS as 1
        rw = 0;                         //set RW as 0
        en = 1;                         //set EN as 1
        time(10);                       //set delay 10 times
        en = 0;                         //set EN as 0
        return;
}
```