CHAPTER-5

FIRMWARE DEVELOPMENT

This Chapter explains the stepwise process of firmware development, which includes the transfer functions for home appliances (heater, bulb, exhaust fan) and sensor used, for proper modelling of the system with PID controller. Implementation of PID with GA and PSO and its programming Flow charts for appliances are also explained.

5.1 Mathematical Model

To find the tuning parameters of PID controller a number of tuning methods are available. The choice of method to be used depends on the type of the system, which is to be controlled.

The first transfer function of the system is obtained by mathematical modeling of the controller part of the system.

Through mathematical modelling transfer function for each appliance has been identified and an optimum value of the firing angle is chosen to trigger the dimming circuit of the system to control the appliance at optimum level.

5.1.1 Transfer Function for Heater

The heater element shows non-linear relationship between the applied voltage and the energy generated. The heating levels are controlled by an error signal generated by the difference of input value and sensory data.

The transfer function for heater[http://web.itu.edu.tr/]is as given by equation-5.1.

$$T(s) = (e^{-Ls})/(1 + sT) \quad \text{.....(5.1)}$$

Where

L – Delay time

T – Rise time

5.1.2 Transfer Function for Bulb

Bulb is having a filament element and has transfer function [https://intranet.ee.ic.ac.uk/t.clarke/ee2lab/handouts/D/Instruction_exptD.pdf] given in equation-5.2.

$$T(s) = 1/(sL + R) \quad \text{.....(5.2)}$$

Where

L – Inductance of filament

R – Resistance of filament

In case of 100W bulb R=52Ω, L=10mH

Then transfer function of bulb comes out -

$$T(s) = 1/(.01s + 52) \quad \text{.....(5.3)}$$

5.1.3 Transfer Function for Exhaust Fan

Transfer function of exhaust [53] is given in equation-5.4.

$$\frac{dTe}{dVs} = T(s) = \frac{3.89\left(1 + \frac{s}{123}\right)}{1 + \frac{0.18s}{314} + \frac{s^2}{314^2}} \quad \text{.....(5.4)}$$

Where

Te= Electrical torque

Vs= Supply voltage

5.1.4 Transfer Function for Temperature Sensor

The temperature sensor is a semiconductor device and specified as 10mV/°C [sunrom.com], i.e.

$$T(s) = \frac{V_0}{T} = 0.01 \quad \text{.....(5.5)}$$

Where Vo is the sensor output voltage in volts, and T is the temperature in °C.

5.2 System Modeling

System modeling is done through simulation on MATLAB tool with the help of PID controller and optimizing algorithm.

5.2.1 PID Controller

The system model is designed with MATLAB Simulink and the optimized values of tuning parameters are calculated. Then program code for PID is written in Embedded 'C' and employed on Atmega16 microcontroller.Atmega16 is RISC architecture based 8 bit microcontroller with 131 instruction set. It can be operated on a maximum frequency of 16MHz. It has flash memory of 16 KB, static RAM of 1 KB and EEPROM of 512 Bytes. It has 32 I/O (input/output) lines. It is supported by ADC,USART, SPI, Analog Comparator, JTAG etc.

PID controller is used to find the tuning parameters. The parameters are calculated on the basis of error signal generated from the difference of sensor value and user's defined value. As shown in Fig.5.1, the generated tuning parameters are then used to control the appliances at desired level.

The following command set is used for implementation of the PID-

//all GAIN values have been computed by PID tuner

-#define Prop_GAIN…

#define Int_GAIN…

#define Der_GAIN …

#define dt_ 0.5

int set_**X** = 0 , previous error = 0; // **X** may be temperature/light intensity/humidity

int error, feedback_**X**, output;

int D_error = 0, I_error = 0;

previous error = set_**X** - feedback_**X**;

error = set_**X** - feedback_**X**;

Integral error += (error)*dt;

Derivative error = (error – previous error)/dt;

output = (Prop_GAIN * error) + (Int_GAIN * Integral error) + (Derivative GAIN * Derivative error);

previous error = error;

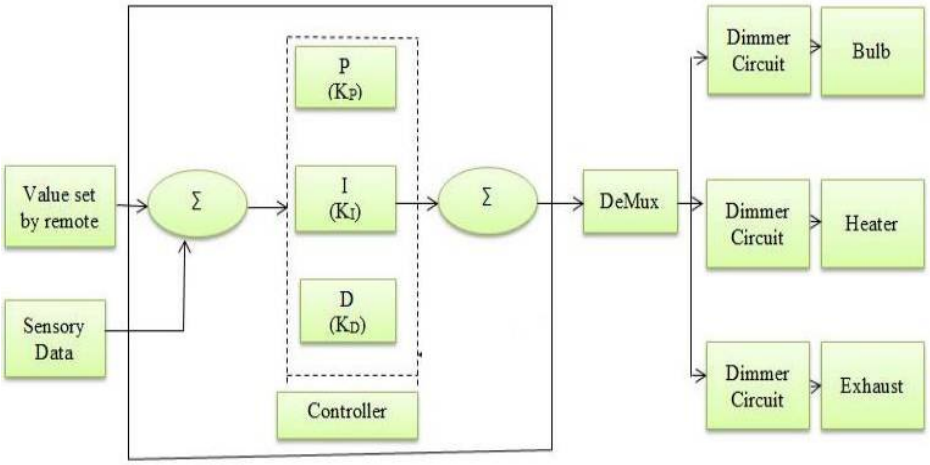

Fig.5.1 Block Diagram for PID controller

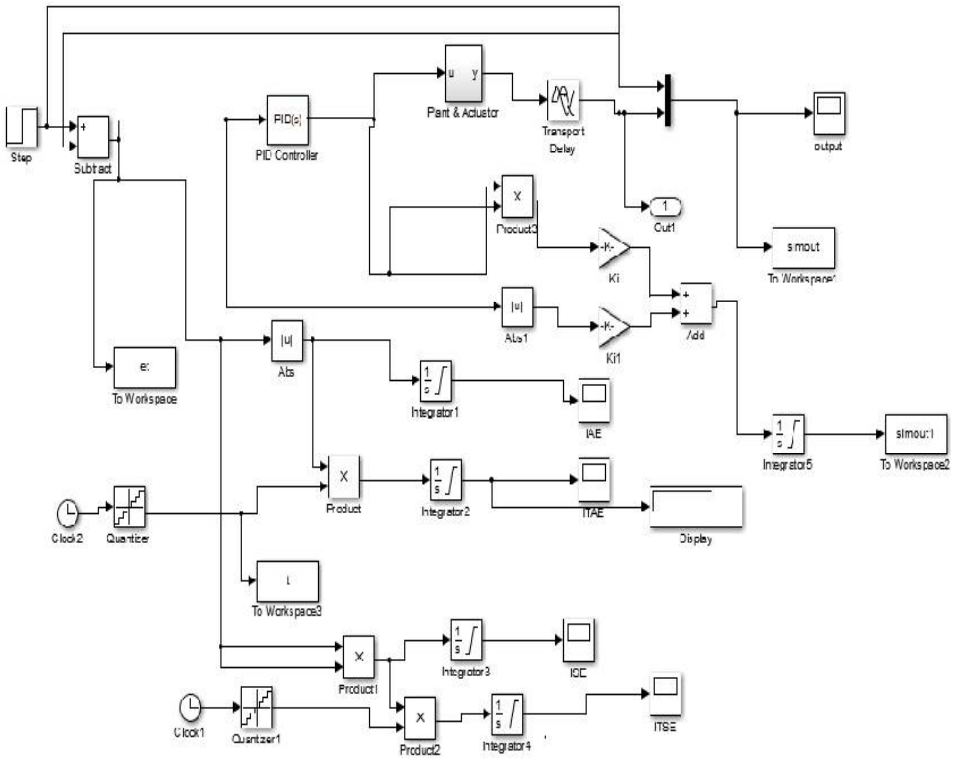

Fig.5.2 MATLAB Simulink model for PID controller

5.3 Optimization Algorithms with PID Controller

Two optimization algorithms- genetic algorithm and particle swarm optimization are analyzed for the designed system. These two algorithms are selected on the basis of literature review. Literature review suggests that GA and PSO algorithms are suitable for optimizing the physical parameters. Though no actual hardware implementation is found in literature, but a simulation approach is discussed. On the basis of study of these two methods, these are analyzed first by simulation and then are implemented on actual developed hardware for power consumption analysis.

5.3.1 PID Controller with GA

Digital Processor for Implementation of GA-PID

The following objective function is used to implement GA.

$$IAE=\int_0^T |e(t)|\, dt \qquad (5.6)$$

Fig.5.3 shows the genetic algorithm to find out the values for tuning parameters of PID controller and is denoted as GA-PID controller. The error signal is generated with the difference of input value from remote control and feedback signal from sensor.

Appliances are connected to controller through dimmer circuits. The level at which is to trigger is decided by the controller.

GA helps to calculate the optimized tuning parameters to control the PID controller and on the basis of error signal and tuning parameters, the triggering level is decided.

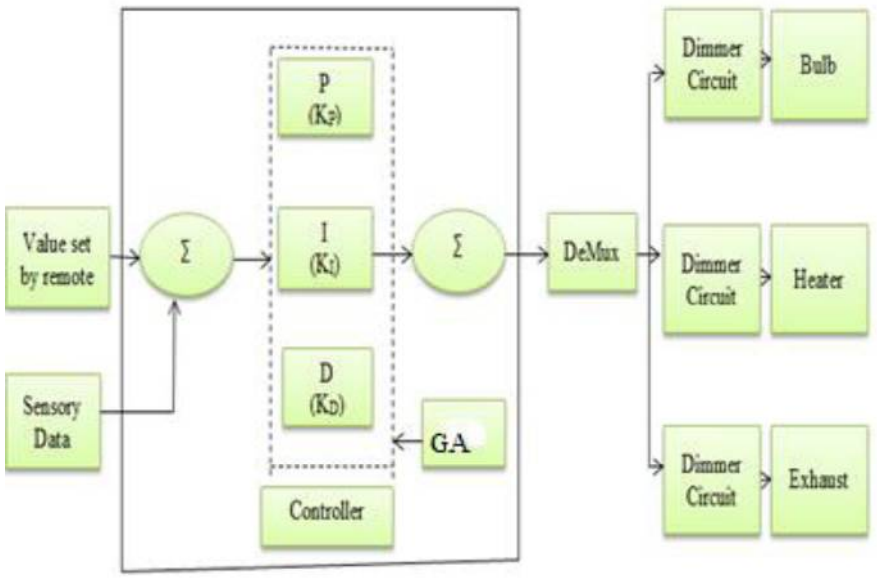

Fig.5.3 Block diagram for GA-PID controller

5.3.2 PID controller with PSO

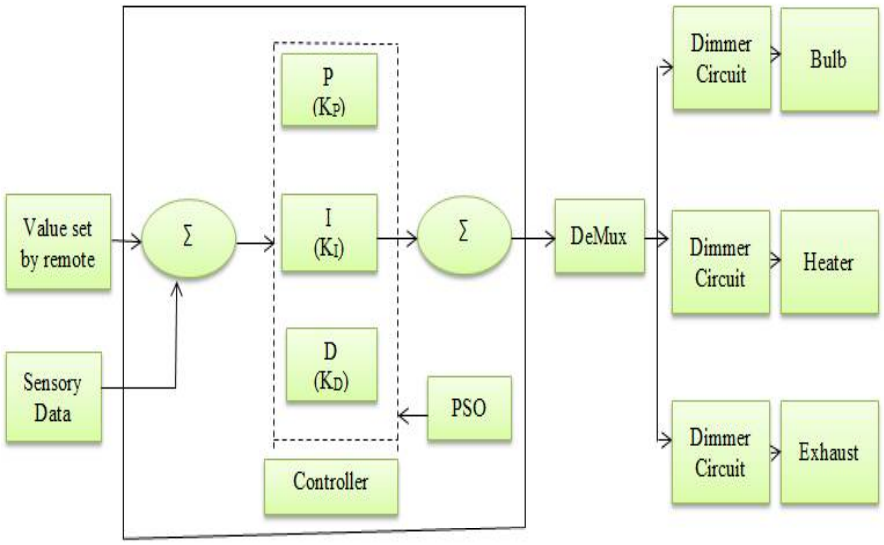

Fig.5.4 Block diagram for PSO-PID controller

Fig.5.4 shows particle swarm optimization algorithm to find out the values for tuning parameters of PID controller and is denoted as PSO-PID controller. The error signal is generated with the difference of input value from remote control and feedback signal from sensor.

5.4 AVR Studio-4

To program the microcontrollers the WinAVR is used. The source code is generated in Embedded 'C' language and converted into .Hex file to burn in the microcontroller.

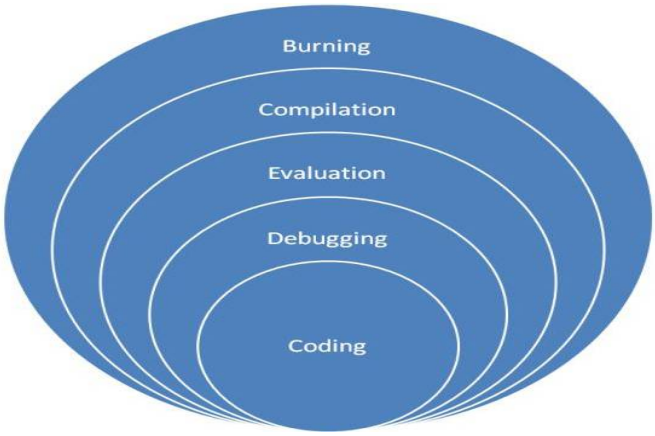

Fig.5.5 Process for Software Developments

WinAVR is an open source software development tool for the Atmel AVR series of RISC based microcontroller. It includes the GNU GCC compiler for C and C++. Fig.5.6 shows AVR studio programming window. Robokits AVR USB Programmer is used to burn the developed hex file in controller.

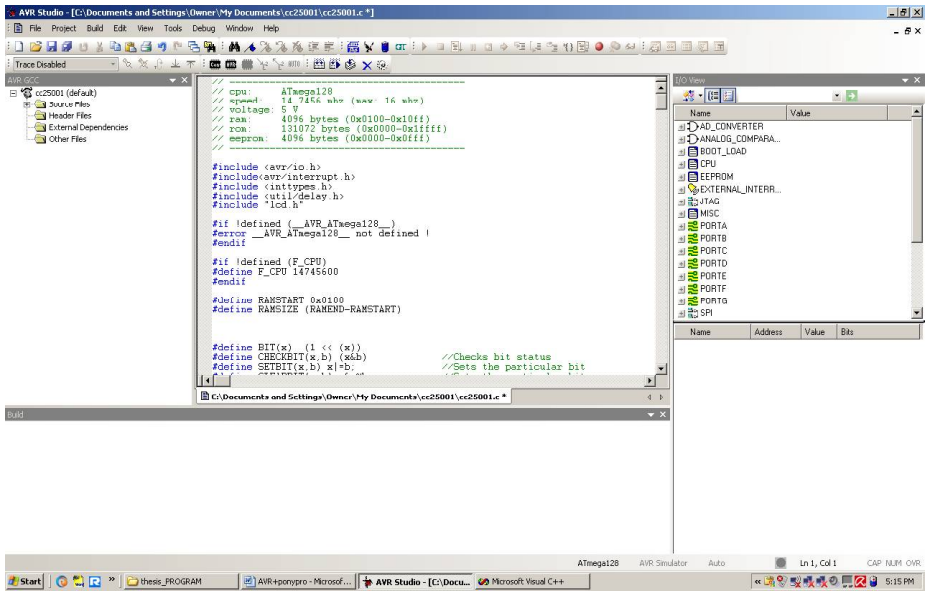

Fig.5.6 AVR Studio programming window

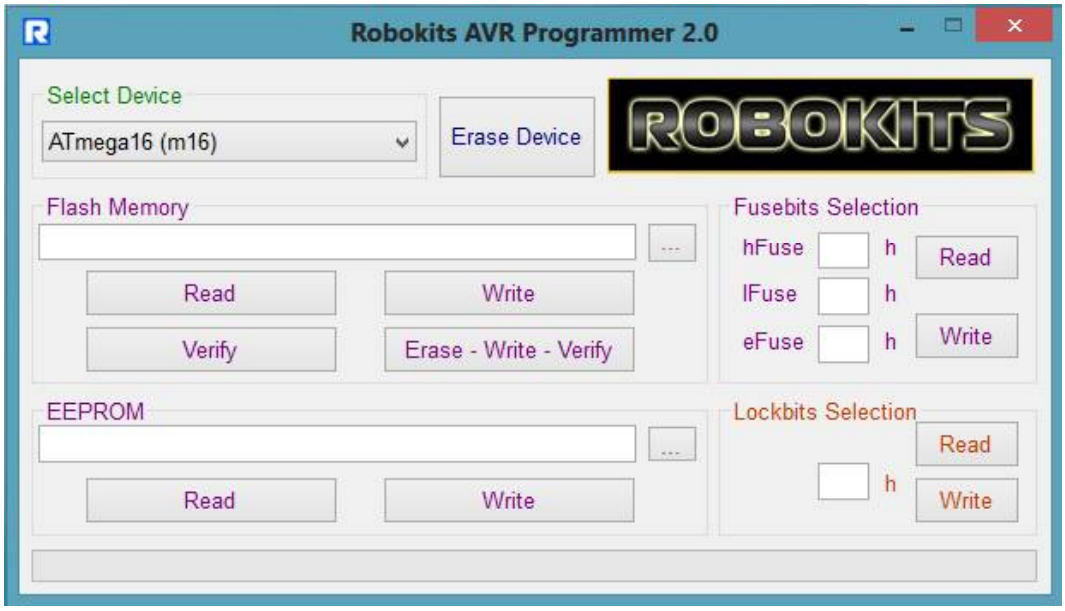

Fig.5.7 Robokits AVR USB Programmer window

5.5 Programming Flow Chart for Remote Control in Semi-autonomous Mode

Fig.5.8 shows the Flow chart, which discusses all the programming steps to develop firmware for remote control in semi-autonomous mode, with Embedded 'C'.

The functions for LCD, USART and ADC are initialized. After the initialization of remote control, the controller waits for an interrupt signal. When the interrupt signal is received user will have the option of selecting the mode of operation.

If semiautonomous mode is selected then values are to be set by the user by using switches available on remote control. After selecting the mode of operation controller will wait for another signal. For every appliance, combinations of switches are available on remote control for the selection of dimming level of appliance. Signal is generated by pressing the proper combination of switches which will be processed by the controller. There are nine switches for different functions of the controller. The 'OK' switch is pressed to send the data in the form of data packet with unique ID to the receiver section. The same data is displayed on LCD placed on remote control.

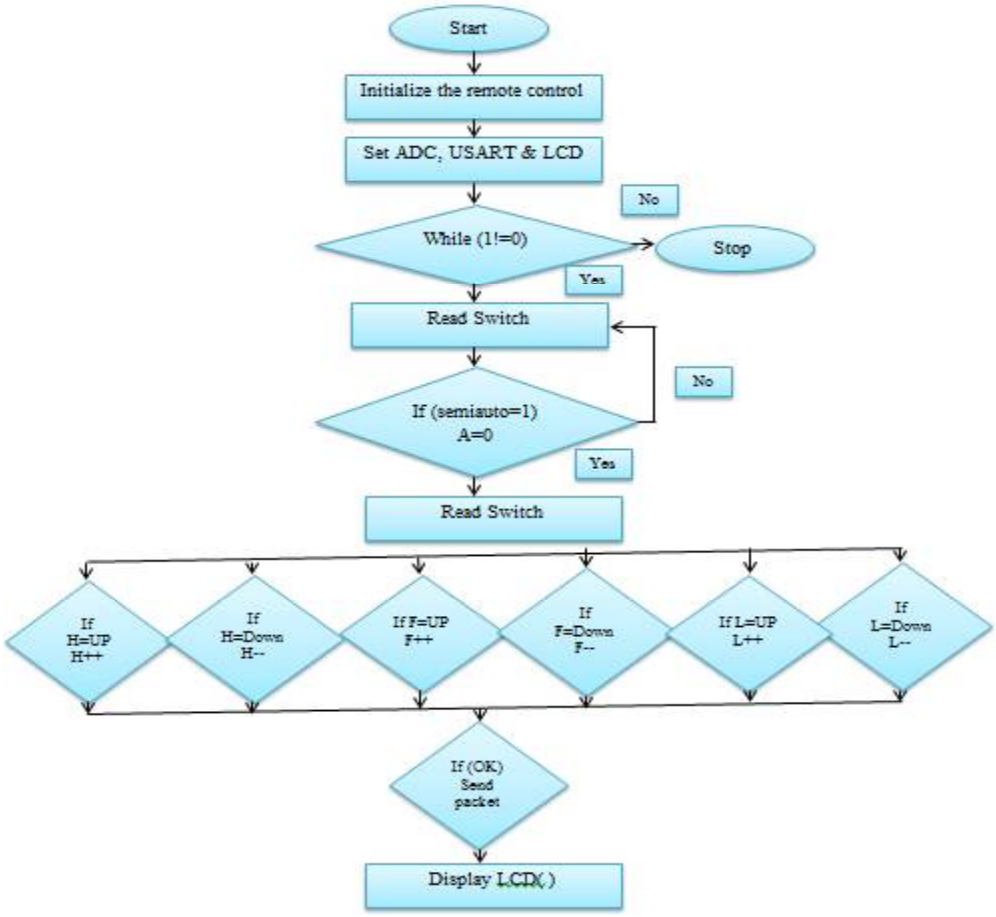

Fig.5.8 Flow chart for remote control in semi-autonomous mode

Zigbee is used as wireless communication module between remote control and receiver. The data is transmitted in the form of packet having length of five byte with unique ID.

Data Packet Format:

0xEA	Autonomous / Semiautonomous	Node ID	Data	0x0D

5.6 Programming Flow Chart for Remote Control in Autonomous Mode

Fig.5.9 shows the Flow chart to discuss the steps involved to develop firmware for remote control in autonomous mode, with Embedded 'C' language.

The functions for LCD, USART and ADC are initialized. When the interrupt signal is received user will have the option of selecting the mode of operation. After the initialization of receiver section, the controller waits for an interrupt signal. In Autonomous mode sensors (temperature/humidity and LDR), remote control plays important role to maintain the ambient conditions. The user is required to just feed the required values of physical parameters i.e temperature, humidity and light intensity through switch array. Sensor is used to sense the present value of parameter in the surrounding. The so measured sensor value (act as feedback signal to receiver section) and input value by user (act as reference signal to receiver section) are transmitted to the receiver section trough Zigbee in form of an unique packet.

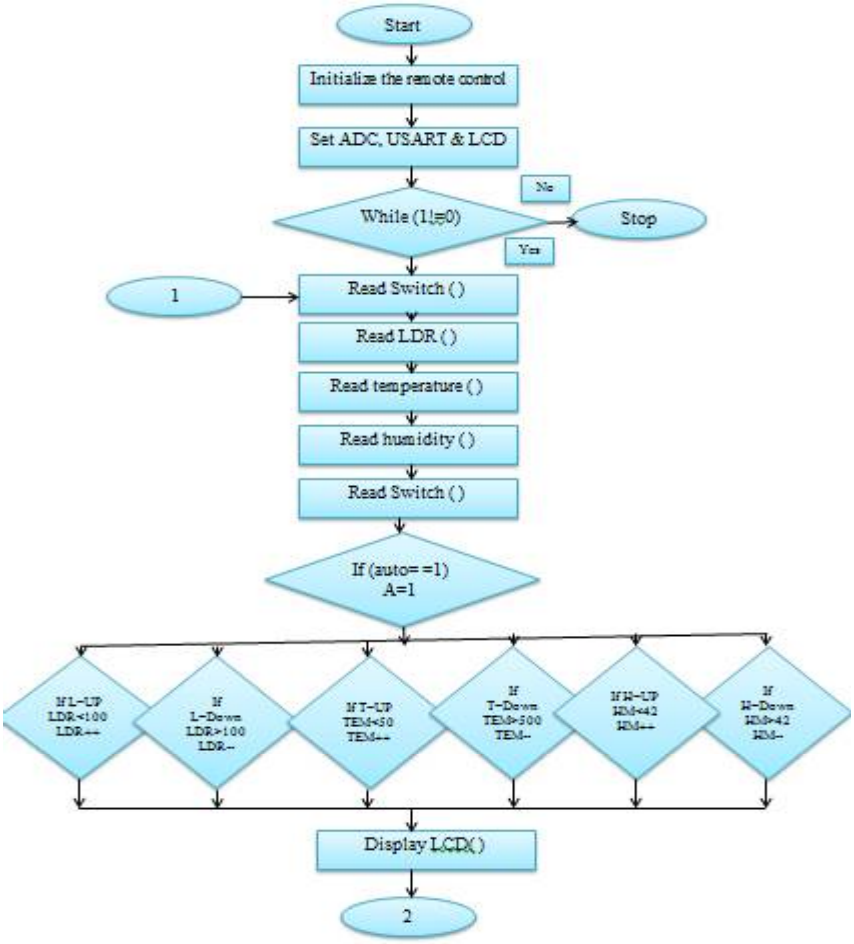

Fig.5.9 Flow chart for remote section in autonomous mode

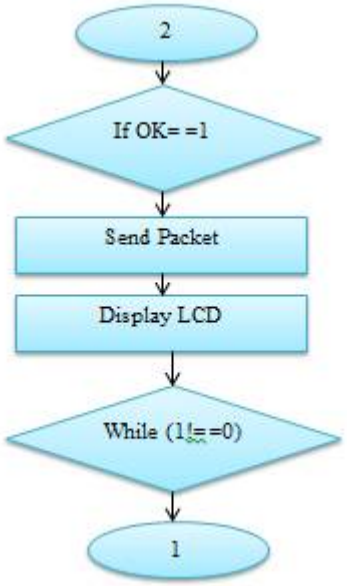

Fig.5.10 Flow chart for remote section in autonomous mode (contd..)

5.7 Programming Flow Chart for Receiver Section

Fig.11 shows the interrupt subroutine and Fig.5.12 shows the Flow chart to discuss the steps involved to develop firmware for receiver section, in Embedded 'C' language.

Receiver section plays different roles in two different modes (Semi-autonomous and autonomous). In semi-autonomous mode, it is only to follow the values defined by the user at remote control to set the dimming level of appliance. But in autonomous mode, receiver is to generate an error signal by calculating the difference between sensor value and input value from remote control. This error signal is used as objective function for the algorithm used with the PID controller. Kp, Ki and Kd values are being calculated by PID, GA-PID and PSO-PID then appliance is triggered at the required dimming level itself. It is done by judging the requirement of the system to shift the level at upper side or lower side. After selecting appropriate dimming level the appliance is to work on the directed voltage level, as discussed in chapter-4. In the developed system three appliances (Heater, exhaust fan and bulb) has been controlled.

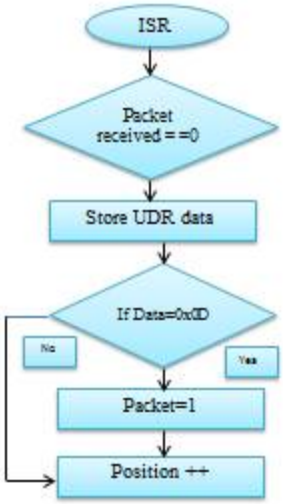

Fig.5.11 Flow chart for interrupt for receiver section

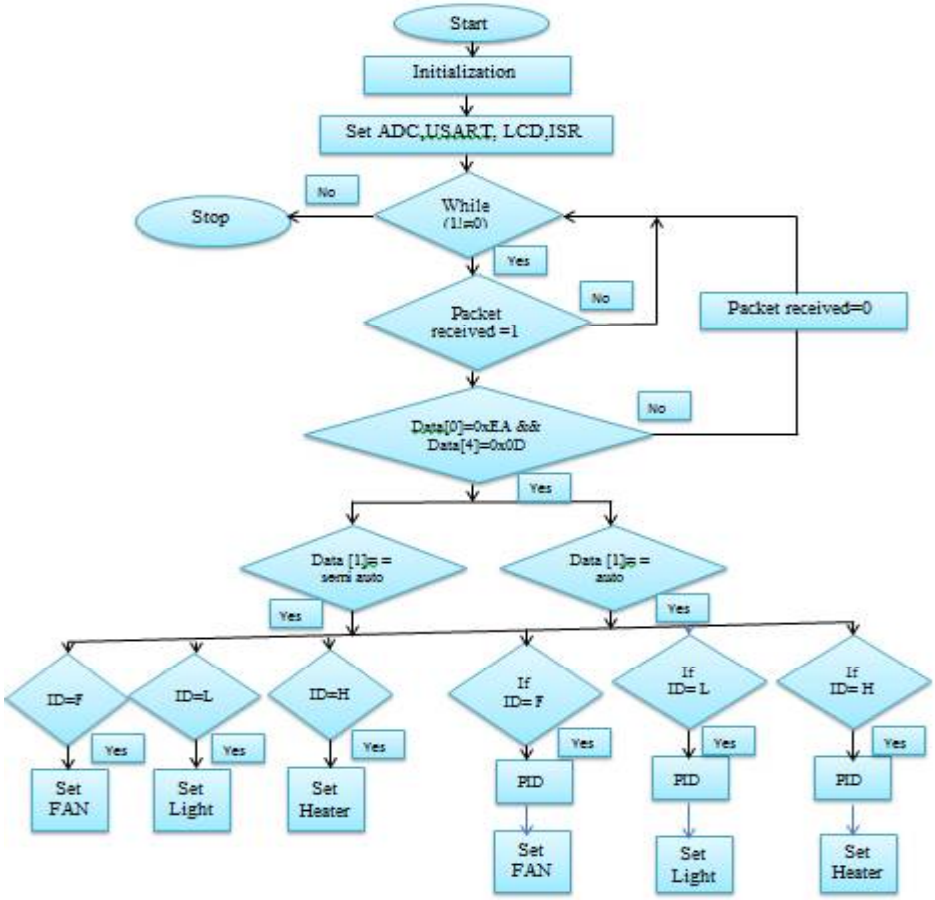

Fig.5.12 Flow chart for receiver section

5.8 Chapter Summary

The chapter describes the steps followed to develop complete firmware for the designed system. It includes programming Flow chart of microcontroller both in autonomous and semi-autonomous mode for remote control and receiver section. The operation of PID controller is discussed with optimizing algorithms. The transfer function for heater, bulb and exhaust fan is described which are required to develop mathematical model of the system.