PROJECT REPORT
ON

MICROCONTROLLER BASED HOME AUTOMATION SYSTEM

A PROJECT BY

Submitted in partial fulfillment of the award of
The degree of
BACHELOR OF SCIENCE
IN
ELECTRICAL & ELECTRONIC ENGINEERING

SUBMITTED BY

Md. Shamsul Alam	**083800024**
Md. Mahafuzur Rahaman	**083800026**
Md. Raihan Zaman Khan	**083800044**
Md. Yiakub Hossain	**083800029**
Md. Shariful Islam	**083800018**

SUPERVISED BY

ABU SHAFIN MOHAMMAD MAHDEE JAMEEL
Lecturer, Department of Electrical and Electronic Engineering,
Faculty of Engineering and Technology,
Eastern University, Dhaka, Bangladesh

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EASTERN UNIVERSITY, DHAKA, BANGLADESH

JULY 2012

DECLARATION

We do here by solemnly declare that, the work presented in this project report has been carried by us and has not been previously submitted to any other university. This project is the outcomes of the investigation performed by us under the supervision of **Abu Shafin Mohammad Mahdee Jameel**, Lecturer, Department of Electrical and Electronic Engineering, Faculty of Engineering & Technology, Eastern University, Dhaka, Bangladesh.

External:

…………………………………

Md. Ashif Iqbal Sikder
Lecturer,
Department of Electrical and Electronic Engineering,
Eastern University, Dhaka, Bangladesh.

Supervisor:

- -

Abu Shafin Mohammad Mahdee Jameel
Lecturer,
Department of Electrical and Electronic Engineering,
Eastern University, Dhaka, Bangladesh.

Signatures of the Candidates:

…………………………………
Md. Shariful Islam
ID # 083800018

…………………………………
Md. Shamsul Alam
ID # 083800024

…………………………………
Md. Mahafuzur Rahaman
ID # 083800026

…………………………………
Md. Yiakub Hossain
ID # 083800029

…………………………………
Md. Raihan Zaman Khan
ID # 083800044

APPROVAL

The project report design and implementation of "Microcontroller based Home Automation System" has submitted to the following members of the board of examiners of the faculty of Engineering & Technology in partial fulfillment of the requirements for the degree of "Bachelor of Science in Electrical & Electronics Engineering" by the following students and has been accepted as satisfactory.

1. **MD. SHARIFUL ISLAM (ID # 083800018)**
2. **MD. SHAMSUL ALAM (ID # 083800024)**
3. **MD. MAHAFUZUR RAHAMAN (ID # 083800026)**
4. **MD. YIAKUB HOSSAIN (ID # 083800029)**
5. **MD.RAIHAN ZAMAN KHAN (ID # 083800044)**

………………………………………

Prof. Dr. Mirza Golam Rabbani,
Chairperson,
Department of EEE,
Eastern University, Dhaka, Bangladesh.

………………………………………

Prof. Dr. Mirza Golam Rabbani,
Acting Dean,
Faculty of Engineering & Technology,
Eastern University, Dhaka, Bangladesh.

DEDICATION

We, hereby, declare that the work presented in this project is the outcome of the investigation performed by us under the supervision of **Abu Shafin Mohammad Mahdee Jameel**, Lecturer, Department of Computer Science and Engineering, Eastern University. We also declare that no part of this project and thereof has been or is being submitted elsewhere for the award of any degree or diploma.

Also this project is dedicated to our beloved parents and our honorable teachers of Eastern University.

ACKNOWLEDGEMENT

During the one year project life in Eastern University, Dhaka we received a great deal of selflessly helps from a lot of people that we would like to say thanks to them.

First of all, we would like to pay our earnest gratitude to the almighty Allah for his endless blessing and mercy. The almighty gave us enough hardworking capability and persistence which allowed me to complete this project work.

It is a real pleasure to express our deepest appreciation, sincere gratitude and heartiest gratefulness to our project supervisor **ABU SHAFIN MOHAMMAD MAHDEE JAMEEL**, Lecturer, Department of EEE, Eastern University. For the endless hours of help, suggestion ideas and advice during the development of this project. This encouragement and motivation helped us to overcome all the difficulties.

We would like to express our heartiest gratefulness to our entire teacher here at the Department of Electrical & Electronic Engineering, Eastern University of Bangladesh. Discussion with many of them has enriched our conception and knowledge about this project work.

We also like to thank the many website owners who kept reading materials in the internet. This helped us to develop this project.

We would like to thank all Eastern University Faculties and staff for making friendly and fruitful co-operation time to time.

Finally, appreciations are placed before respectable parents, fellow classmates and friends for sharing their knowledge and ideas that contributed in accomplishing this project and to all who participates in many ways during this project. At last, we would like to remember our almighty Allah for whom we can get education.

ABSTRACT

In this project we made remote controlling home automation system. So there is no use to extra sensor. We used program based microcontroller technology in this project. So we used here programmable device ATmega8 which assist turn on or off the different load. Generally available load controller found in the market where this technology is not used usually.

It is essential to mention that; here we turn on or off different load as example Light, Fan, TV, etc. This mention work by the Microcontroller program and device used ATMega8.

The ultimate goal of this project was to create a functional microcontroller & remote based load controlling system. For this project, it was important that the Microcontroller scheme be able to save our time and easy to switching. With a more efficient Programmer and Microcontroller, this goal could be reached without consuming more load.

INDEX

TABLE OF CONTENTS: **PAGE NO.**

CONTENTS

LIST OF FIGURE

CHAPTER - I

INTRODUCTION OF MICROCONTROLLER SYSTEM

1.1 Project Objective

The objective of this thesis paper is to design programmable microcontroller base circuit which light turn on or off automatically by use remote control depend on RC-5 Protocol. A remote will use to giving command to control device on/off. And temperature sensor, water level sensor, day light sensor can be use for automated controlling home appliance or other device.

1.2 Overview of Document

The following reported documents, the procedures and methods used to design and implement a remote control home appliance, temperature control at room, day light sensing and water level alarm control etc chapter-by-chapter overview details below.

In the Chapter 02 of this document contains background information of the microcontroller, remote signal decoder (TSOP1738), temperature sensor DS18B20, light sensor (LDR), crystal , relay driver switching transistor, relay and other simple some active or passive device with detailing relevant theory which is crucial in further understanding of the design section. And over view of the An over view of currently available device is also contained in this section.

In the chapter 03 introduction about Smart home control system, the working procedure and relevant implementation of this system. It starts by describing the about overall work procedure, how we make work LDR as a sensor, about temperature and

In the chapter 04 contains our project circuit diagram and its operation. Furthermore, this chapter comments on the key operation implementation.

In the chapter 05 provides a brief overview of entire project, the software implementation of the driver, outlining the requirements of the software and how it is used to write code to control the hardware. How select chip. What was actually achieved and how performed it well. This chapter also summarizes which were made previously in this report.

In the chapter 06 provides a brief description about making PCB for this project and other project. The software uses to implementation the PCB design.

In the chapter 07 provides a brief description about hardware implementation & cost calculation

In the chapter 08 show our total completed hardware picture, 3D view create by software.

In the chapter 09 show Conclusion & Future Development

This report aims to provide a details and accurate analysis of the LED light scheme with the relevant theory associated with it. It is hoped that this report fulfills this aim and allows for a good understanding of the driver design and its capability.

1.3 MICROCONTROLLER:

Microcontroller can be termed as a single on chip computer which includes number of peripherals like RAM, EEPROM, Timers etc., required to perform some predefined task.

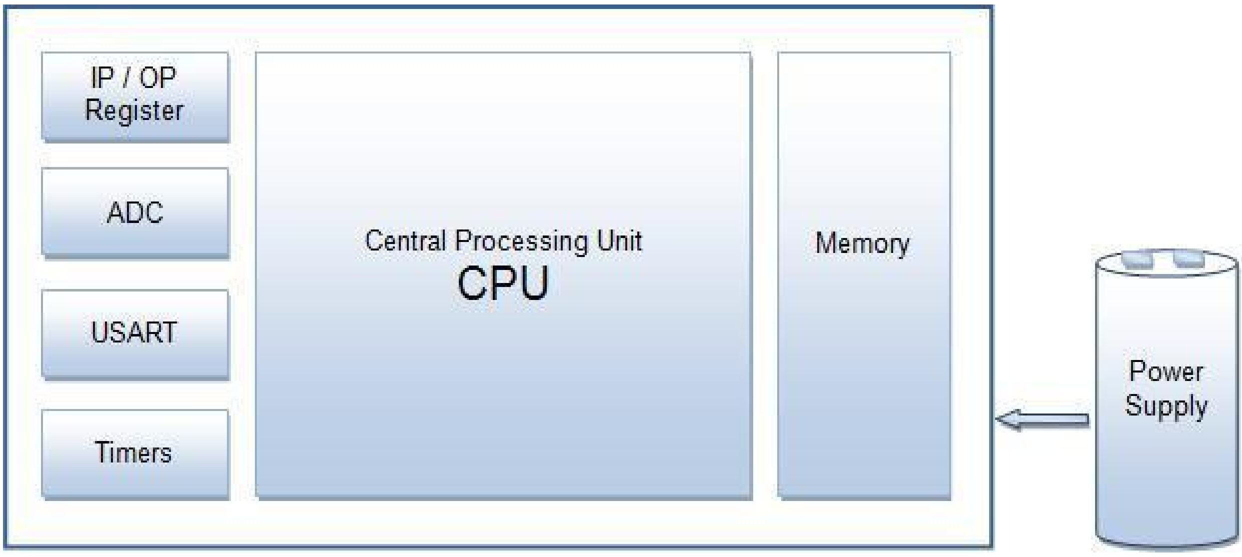


Figure-1.1: Architecture of AVR Microcontroller

Does this mean that the microcontroller is another name for a computer…?
The answer is NO!

The computer on one hand is designed to perform all the general purpose tasks on a single machine like you can use a computer to run a software to perform calculations or you can use a computer to store some multimedia file or to access internet through the browser, whereas the microcontrollers are meant to perform only the specific tasks, for e.g., switching the AC off automatically when room temperature drops to a certain defined limit and again turning it ON when temperature rises above the defined limit.

There are number of popular families of microcontrollers which are used in different applications as per their capability and feasibility to perform the desired task, most common of these are 8051, **AVR** and PIC microcontrollers.

1.4 DIFFERENCE BETWEEN MICROPROCESSOR AND MICROCONTROLLER:

A microcontroller differs from a microprocessor in many ways. The first and most important difference is its functionality. In order that the microprocessor may be used, other components such as memory must be added to it. Even though the microprocessors are considered to be powerful computing machines, their weak point is that they are not adjusted to communicating to peripheral equipment. Simply, In order to communicate with peripheral environment, the microprocessor must use specialized circuits added as External chips, in short microprocessors are the pure heart of the computers. This is how it was in the beginning and remains the same today.

On the other hand, the microcontroller is designed to be all of that in one. No other specialized external components are needed for its application because all necessary circuits which otherwise belong to peripherals are already built into it. It saves the time and space needed to design a device.

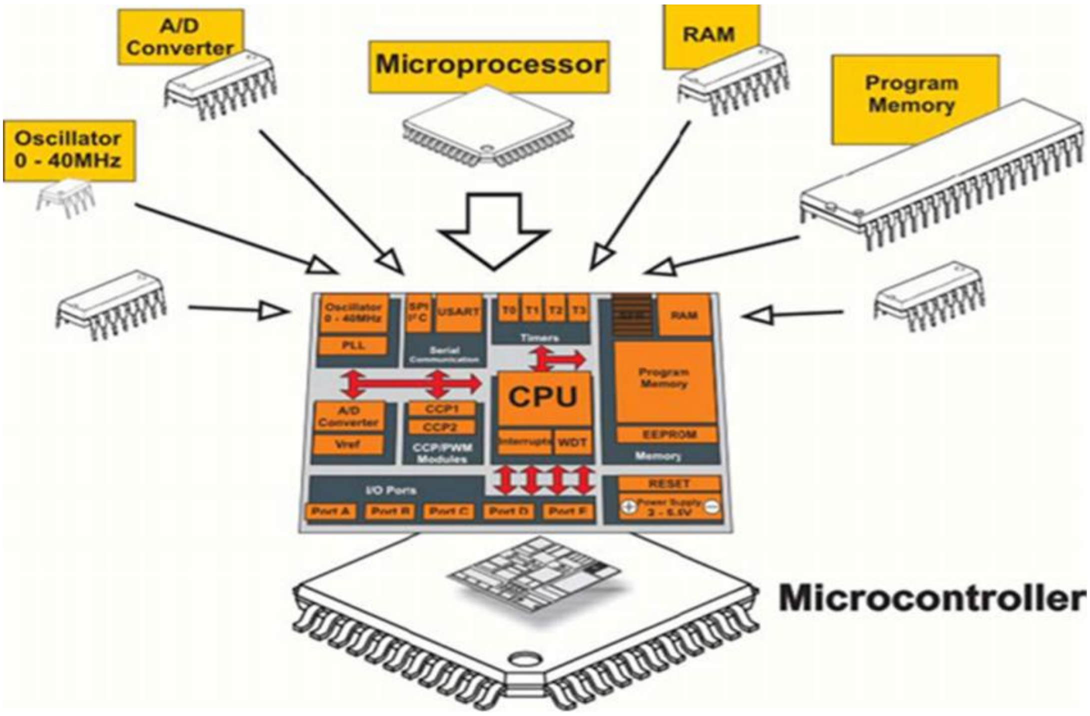


Figure-1.2: Different Between Microcontroller versus Microprocessor

Microprocessor = cpu
Microcontroller = cpu + peripherals + memory
Peripherals = ports + clock + timers + uarts + adc converters +lcd drivers + dac + other stuff.
Memory = eeprom + sram + eprom + flash
A microcontroller has a combination of all this stuff.
A microprocessor is just of cpu.

1.5 REGISTER:

A register or a memory cell is an electronic circuit which can memorize the state of one byte.

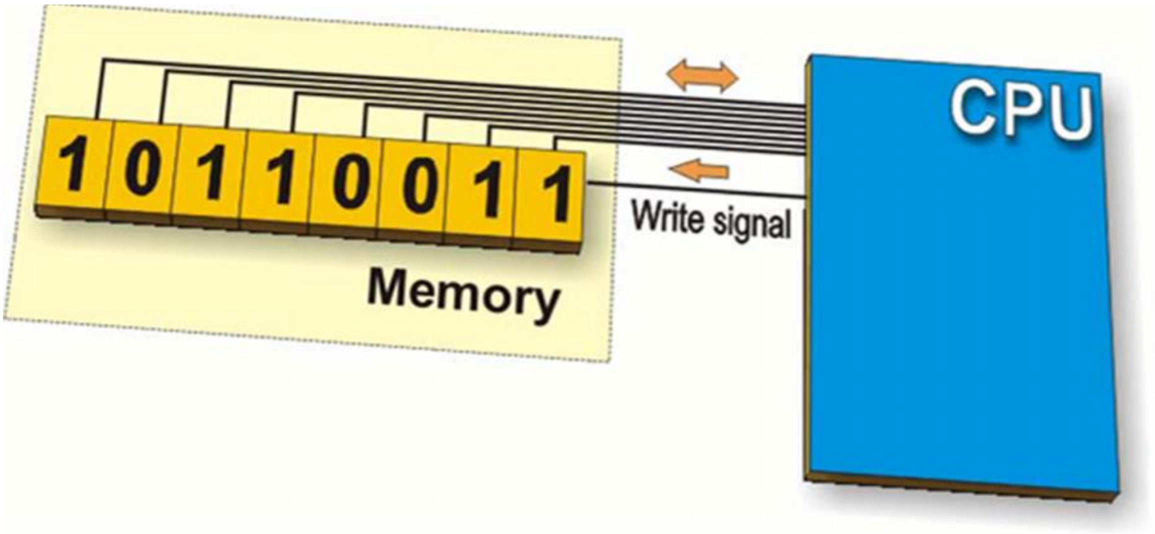


Figure -1.3: Connection Memory and CPU

1.6 SPECIAL FUNCTION REGISTER:

In addition to the registers which do not have any special and predetermined function, every microcontroller has a number of registers whose function is predetermined by the manufacturer. Their bits are connected (literally) to internal circuits such as timers, A/D converter, oscillators and others, which mean that they are directly in command of the operation of the microcontroller. Imagine eight switches which are in command of some smaller circuits within the microcontroller- you are right! Special Function Registers (SFRs) do exactly that.

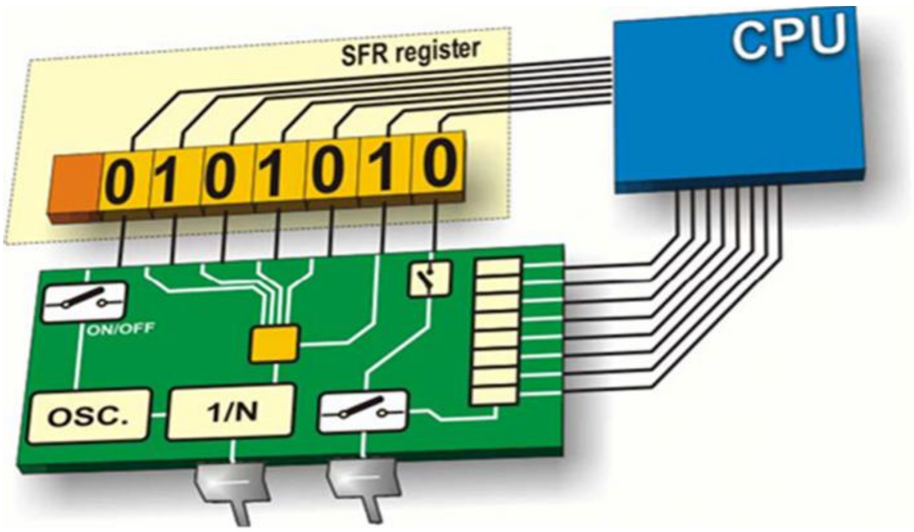


Figure -1.4: Connection SFR Register and CPU

1.7 INPUT/OUTPUT PORTS:

In order to make the microcontroller useful, it has to be connected to additional electronics, i.e. peripherals. Each microcontroller has one or more registers (called a "port") connected to the microcontroller pins. Why input/output? Because you can change the pin's function as you wish. For example, suppose you want your device to turn three signal LEDs and simultaneously monitor the logic state of five sensors or push buttons. Some of ports need to be configured so that there are three outputs (connected to the LEDs) and five inputs (connected to sensors). It is simply performed by software, which means that the pin's function can be changed during operation. One of the

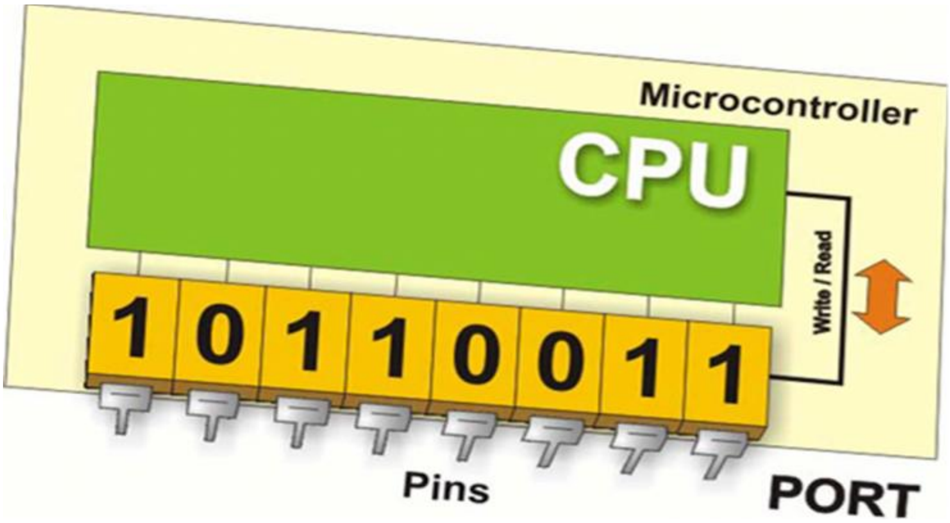


Figure -1.5: Connection PORT and CPU

More important specifications of input/output (I/O) pins is the maximum current they can handle. For most microcontrollers, current obtained from one pin is sufficient to activate an LED or other similar low-current device (10-20 mA). If the microcontroller has many I/O pins, then the maximum current of one pin is lower. Simply put, you cannot expect all pins to give maximum current if there are One of the more important specifications of input/output (I/O) pins is the maximum current they can handle. For most microcontrollers, current obtained from one pin is sufficient to activate an LED or other similar low-current device (10-20 mA). If the microcontroller has many I/O pins, then the maximum current of one pin is lower. Simply put, you cannot expect all pins to give maximum current if there are more than 80 of them on one microcontroller. Another way of putting it is that the maximum current stated in the data specifications sheet for the microprocessor is shared across all I/O ports.

Another important pin function is that it can have pull-up resistors. These resistors connect pins to the positive power supply voltage and their effect is visible when the pin is configured as an input connected to mechanical switch or push button. Newer versions of microcontrollers have pull-up resistors configurable by software. Usually, each I/O port is under control of another SFR, which means that each bit of that register determines the state of the corresponding microcontroller pin. For example, by writing logic one (1) tone bit of that control register SFR, the appropriate port pin is automatically configured as input.

It means that voltage brought to that pin can be read as logic 0 or 1. Otherwise, by writing zero to the SFR, the appropriate port pin is configured as an output. Its voltage (0V or 5V) corresponds to the state of the appropriate bit of the port register.

1.8 MEMORY UNIT:

Memory is part of the microcontroller used for data storage. The easiest way to explain it is to compare it with a filing cabinet with many drawers. Suppose, the drawers are clearly marked so that it is easy to access any of them. It is easy enough to find out the contents of the drawer by reading the label on the front of the drawer.

Each memory address corresponds to one memory location. The content of any location becomes known by its addressing. Memory can either be written to or read from. There are several types of memory within the microcontroller.

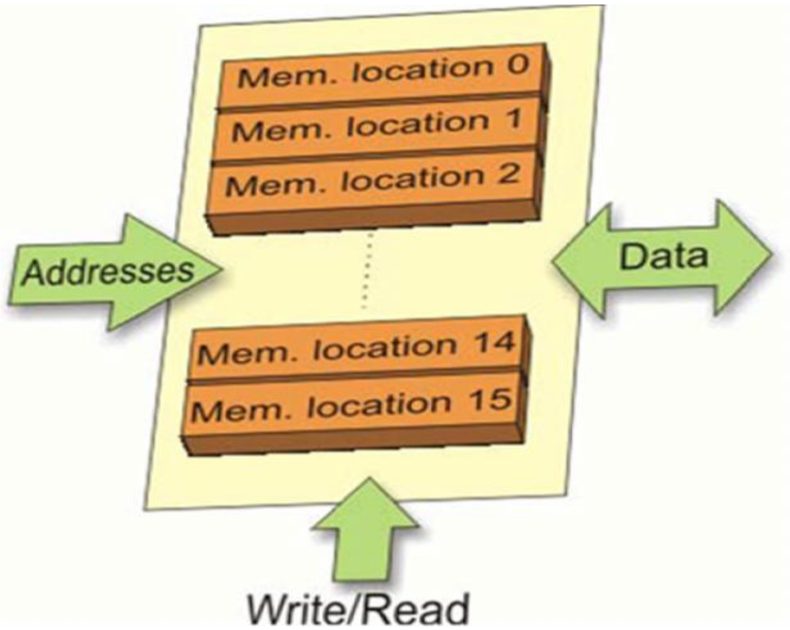


Figure -1.6: Memory Location

ROM (Read Only Memory) is used to permanently save the program being executed. The size of a program that can be written depends on the size of this memory. Today's microcontrollers commonly use 8-bit addressing, which means that they are able to address up to 64 Kb of memory, i.e. 65535 locations. As a novice, your program will rarely exceed the limit of several hundred instructions. There are several types of ROM. Exceed the limit of several hundred instructions. There are several types of ROM.

1.9 INTRRUPT:

The most programs use interrupts in regular program execution. The purpose of the microcontroller is mainly to react on changes in its surrounding. In other words, when some event takes place, the microcontroller does something... For example, when you push a button on a remote controller, the microcontroller will register it and respond to the order by changing a channel, turn the volume up or down etc. If the microcontroller spent most of its time endlessly a few buttons for hours or days... It would not be practical. The microcontroller has learnt during its evolution a trick. Instead of checking each pin or bit constantly, the microcontroller delegates the "wait issue" to the "specialist" which will react only when something attention worthy happens. The signal which informs the central processor about such an event is called an INTERRUPT.

1.10 CPU (Central Processor Unit)

As its name suggests, this is a unit which monitors and controls all processes inside the microcontroller. It consists of several smaller subunits, of which the most important are:

- Instruction Decoder is a part of the electronics which recognizes program instructions and runs other circuits on the basis of that. The “instruction set” which is different for each microcontroller family expresses the abilities of this circuit.

- Arithmetical Logical Unit (ALU) performs all mathematical and logical operations upon data.

- Accumulator is a SFR closely related to the operation of the ALU. It is a kind of working desk used for storing all data upon which some operation should be performed (addition, shift/move etc.). It also stores the results ready for use in further processing. One of the SFRs, called a Status Register (PSW), is closely related to the accumulator. It shows at any given moment the “status” of a number stored in the accumulator (number is greater or less than zero etc.).

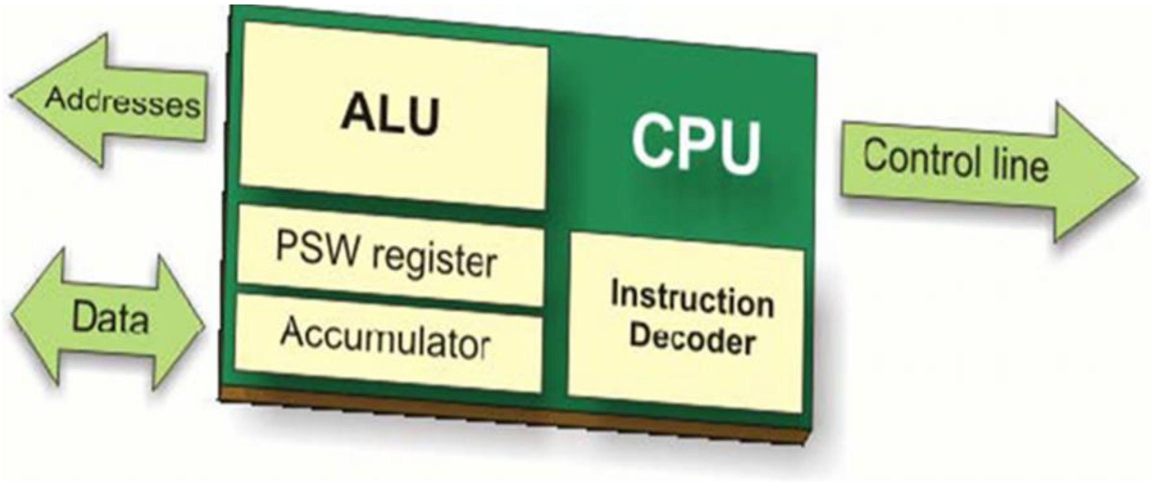


Figure -1.7: Central Processing Unit (CPU)

1.11 SPECIAL COMMUNICATION:

Parallel connections between the microcontroller and peripherals via input/output ports are the ideal solution for shorter distances- up to several meters. However, in other cases - when it is necessary to establish communication between two devices on longer distances it is not possible to use a parallel connection - such a simple solution is out of question. In these situations, serial communication is the best solution. Today, most microcontrollers have built in several different systems for serial communication as standard equipment. Which of these systems will be used depends on many factors of which the most important are:

- How many devices the microcontroller has to exchange data with?
- How fast the data exchange has to be?
- What is the distance between devices?
- Is it necessary to send and receive data simultaneously?

One of the most important things concerning serial communication is the Protocol which should be strictly observed. It is a set of rules which must be applied in order that the devices can correctly interpret data they mutually exchange. Fortunately, the microcontrollers automatically take care of this, so the work of the programmer/user is reduced to simple write (data to be sent) and read (received data).

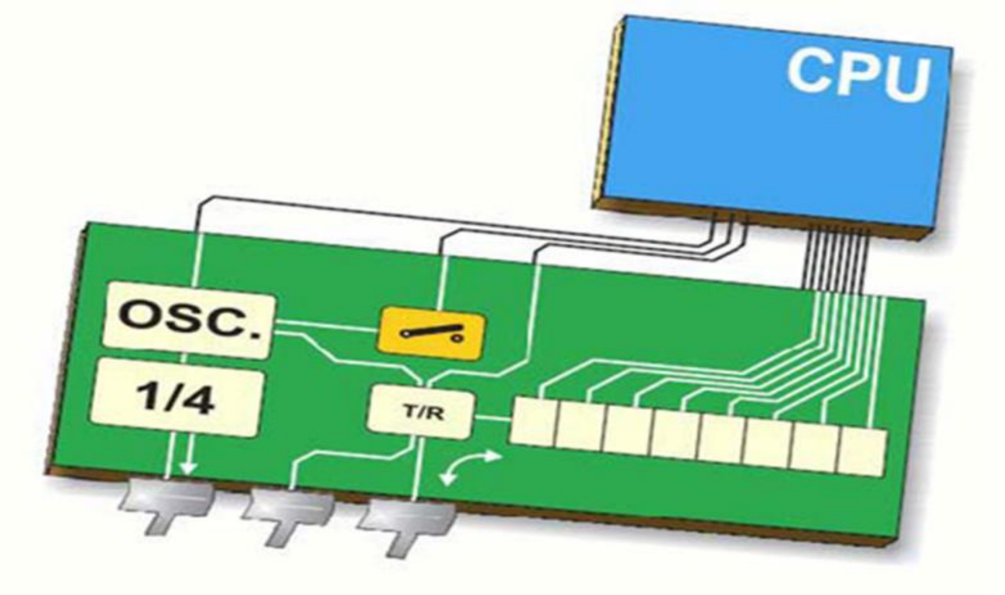


Figure -1.8: Connection Oscillator and CPU

1.12 OSCILLATOR:

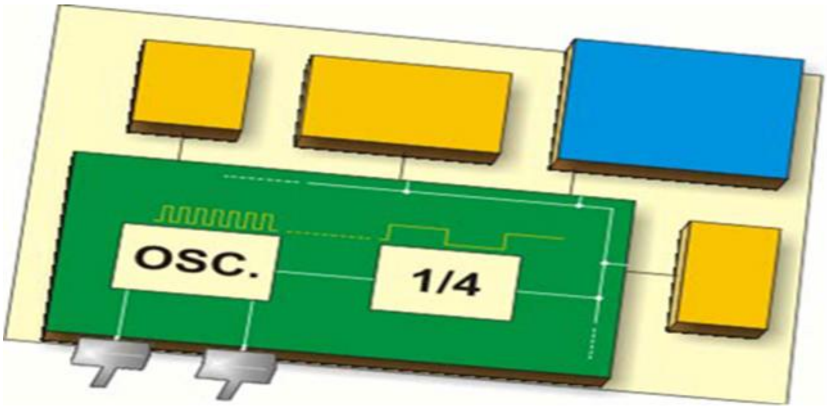


Figure -1.9: Oscillator

Even pulses coming from the oscillator enable harmonic and synchronous operation of all circuits of the microcontroller. The oscillator module is usually configured to use quartz crystal or ceramic resonator for frequency stabilization. Furthermore, it can also operate without elements for frequency stabilization (like RC oscillator). It is important to say that instructions are not executed at the rate imposed by the oscillator itself, but several times slower. It happens because each instruction is executed in several steps. In some microcontrollers, the same number of cycles is needed to execute any instruction, while in others; the execution time is not the same for all instructions. Accordingly, if the system uses quartz crystal with a frequency of 20 MHz, execution time of an instruction is not 50nS, but 200, 400 or 800nS, depending on the type of Microcontroller Unit (MCU)!

1.13 POWER SUPPLY UNIT:

There are two things worth attention concerning the microcontroller power supply circuit. Brown-out is a potentially dangerous state which occurs at the moment the microcontroller is being turned off or in situations when power supply voltage drops to the limit due to electric noise. As the microcontroller consists of several circuits which have different operating voltage levels, this state can cause its out-of-control performance. In order to prevent it, the microcontroller usually has built-in circuit for brown out reset. This circuit immediately resets the whole electronics when the voltage level drops below the limit.

Reset pin is usually marked as MCLR (Master Clear Reset) and serves for external reset of the microcontroller by applying logic zero (0) or one (1), depending on type of the microcontroller. In case the brown out circuit is not built in, a simple external circuit for brown out reset can be connected to this pin.

1.14 TIMERS/COUNTER:

The microcontroller oscillator uses quartz crystal for its operation. Even though it is not the simplest solution, there are many reasons to use it. Namely, the frequency of such oscillator is precisely defined and very stable; the pulses it generates are always of the same which makes them ideal for time measurement. Such oscillators are used in quartz watches. If it is necessary to measure time between two events, it is sufficient to count pulses coming from this oscillator. That is exactly what the timer does. Most programs use these miniature electronic “stopwatches”.

These are commonly 8- or 8-bit SFRs and their content is automatically incremented by each coming pulse. Once a register is completely loaded - an interrupt is generated! If the timer registers use an internal quartz oscillator for their operation then it is possible to measure time between two events (if the register value is T1 at the moment measurement has started, and T2 at the moment it has finished, then the elapsed time is equal to the result of subtraction T2-T1). If the registers use pulses coming from external source then such a timer is turned into a counter**.**

This is only a simple explanation of the operation itself.

1.13 COUNTER:

If a timer is supplying pulses into the microcontroller input pin then it turns into a counter. Clearly, it is the same electronic circuit. The only difference is that in this case pulses to be counted come through the ports and their duration (width) is mostly not defined. This is why they cannot be used for time measurement, but can be used to measure anything else: products on an assembly line, number of axis rotation, passengers etc. (depending on sensor in use).

1.14 A/D CONVERTER:

External signals are usually fundamentally different from those the microcontroller understands (Ones and Zeros), so that they have to be converted in order for the microcontroller to understand them. An analogue to digital converter is an electronic circuit which converts continuous signals to discrete digital numbers. This module is therefore used to convert some analogue value into binary number and forwards it to the CPU for further processing. In other words, this module is used for input pin voltage measurement (analogue value). The result of measurement is a number (digital value) used and processed later in the program.

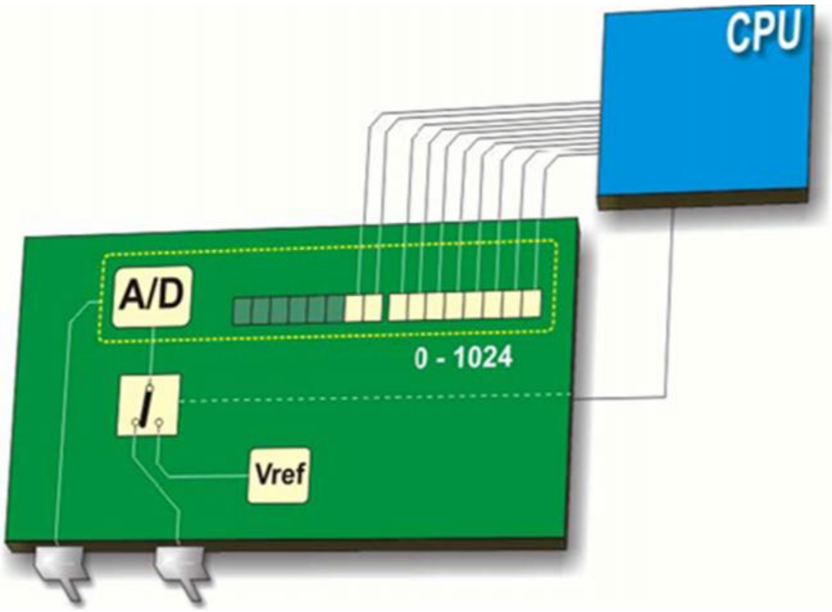


Figure-1.10: Connection Analogue to Digital Converter and CPU

1.15 RISC (Reduced Instruction Set Computer):

In this case, the microcontroller recognizes and executes only basic operations (addition, subtraction, copying etc.). All other more complicated operations are performed by combining these (for example, multiplication is performed by performing successive addition). The constrains are obvious (try by using only a few words, to explain to someone how to reach the airport in some other city). However, there are also some great advantages. First of all, this language is easy to learn.

Besides, the microcontroller is very fast so that it is not possible to see all the arithmetic "acrobatics" it performs. The user can only see the final result of all those operations. At last, it is not so difficult to explain where the airport is if you use the right words. For example: left, right, kilometers etc.

1.8 CISC (Complex Instruction Set Computer):

CISC is the opposite of RISC! Microcontrollers designed to recognize more than 200 different instructions can do much and are very fast. However, one needs to understand how to take all that such a rich language offers, which is not at all easy.

CHAPTER – II

AVR MICROCONTROLLER

2.1 AVR MICROCONTROLLER:

2.1.1 History of AVR

AVR was developed in the year 1996 by Atmel Corporation. The architecture of **AVR** was developed by Alf-Egil Bogen and Vegard Wollan. AVR derives its name from its developers and stands for **A**lf-Egil **B**ogen **V**egard **W**ollan **R**ISC **microcontroller**, also known as **Advanced Virtual RISC**. The AT90S8515 was the first microcontroller which was based on **AVR architecture** however the first microcontroller to hit the commercial market was AT90S1200 in the year 1997.

2.1.2 AVR microcontrollers Categories

AVR microcontrollers are available in three categories:

1. **TinyAVR** – Less memory, small size, suitable only for simpler applications
2. **MegaAVR** – These are the most popular ones having good amount of memory (upto 256 KB), higher number of inbuilt peripherals and suitable for moderate to complex applications.
3. **XmegaAVR** – Used commercially for complex applications, which require large program memory and high speed.

The following table compares the above mentioned AVR series of microcontrollers:

Series Name	Pins	Flash Memory	Special Feature
TinyAVR	6-32	0.5-8 KB	Small in size
MegaAVR	28-100	4-256KB	Extended peripherals
XmegaAVR	44-100	8-384KB	DMA , Event System included

2.1.3 What's special about AVR?

They are fast: AVR microcontroller executes most of the instructions in single execution cycle. AVRs are about 4 times faster than PICs, they consume less power and can be operated in different power saving modes. Let's do the comparison between the three most commonly used families of microcontrollers.

Descriptions	8051	PIC	AVR
SPEED	Slow	Moderate	Fast
MEMORY	Small	Large	Large
ARCHITECTURE	CISC	RISC	RISC
ADC	Not Present	Inbuilt	Inbuilt
Timers	Inbuilt	Inbuilt	Inbuilt

PWM Channels	Not Present	Inbuilt	Inbuilt

AVR is an 8-bit microcontroller belonging to the family of Reduced Instruction Set Computer (**RISC**). In RISC architecture the instruction set of the computer are not only fewer in number but also simpler and faster in operation. The other type of categorization is CISC (Complex Instruction Set Computers). We will explore more on this when we will learn about the architecture of AVR microcontrollers in following section.

Let's see what this entire means. What is 8-bit? This means that the microcontroller is capable of transmitting and receiving 8-bit data. The input/output registers available are of 8-bits. The AVR families controllers have register based architecture which means that both the operands for an operation are stored in a register and the result of the operation is also stored in a register. Following figure shows a simple example performing OR operation between two input registers and storing the value in Output Register.

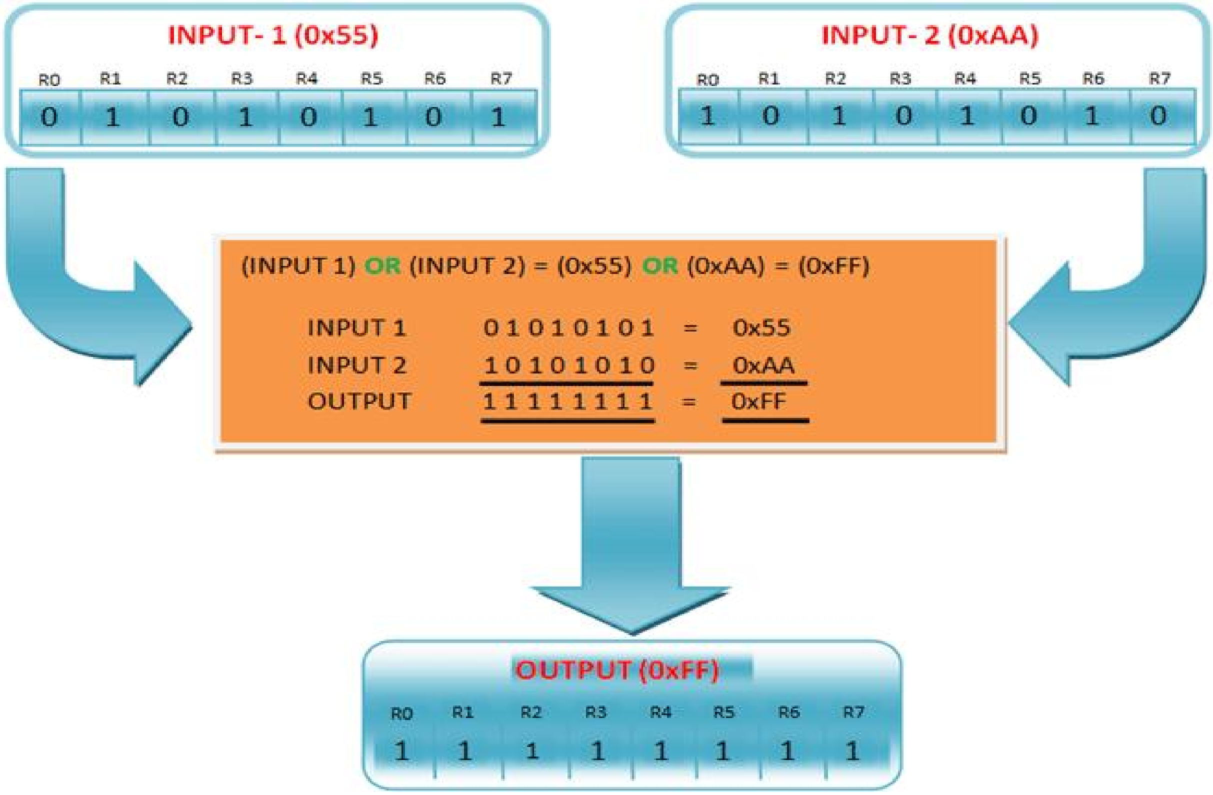


Figure-2.1: OR operation between two input registers and storing in Output Register

The CPU takes values from two input registers INPUT-1 and INPUT-2, performs the logical operation and stores the value into the OUTPUT register. All this happens in 1 execution cycle.

2.1.4 Features of Atmega8

In our journey with the AVR we were working on microcontroller, which is a 40-pin IC and belongs to the mega AVR category of AVR family. Some of the features of Atmega8 are:

- 8KB of Flash memory
- 1KB of SRAM
- 512 Bytes of EEPROM
- Available in 40-Pin DIP
- 8-Channel 10-bit ADC
- One 8-bit Timer/Counter
- 4 PWM Channels
- In System Programmer (ISP)
- Serial USART
- SPI Interface

- Two 8-bit Timers/Counters
- Digital to Analog Comparator.

2.2 Architecture of AVR

The AVR microcontrollers are based on the advanced RISC architecture and consist of 32 x 8-bit general purpose working registers. Within one single clock cycle, AVR can take inputs from two general purpose registers and put them to ALU for carrying out the requested operation, and transfer back the result to an arbitrary register.

The ALU can perform arithmetic as well as logical operations over the inputs from the register or between the register and a constant. Single register operations like taking a complement can also be executed in ALU. We can see that AVR does not have any register like accumulator as in 8051 family of microcontrollers; the operations can be performed between any of the registers and can be stored in either of them.

AVR follows Harvard Architecture format in which the processor is equipped with separate memories and buses for Program and the Data information. Here while an instruction is being executed, the next instruction is pre-fetched from the program memory.

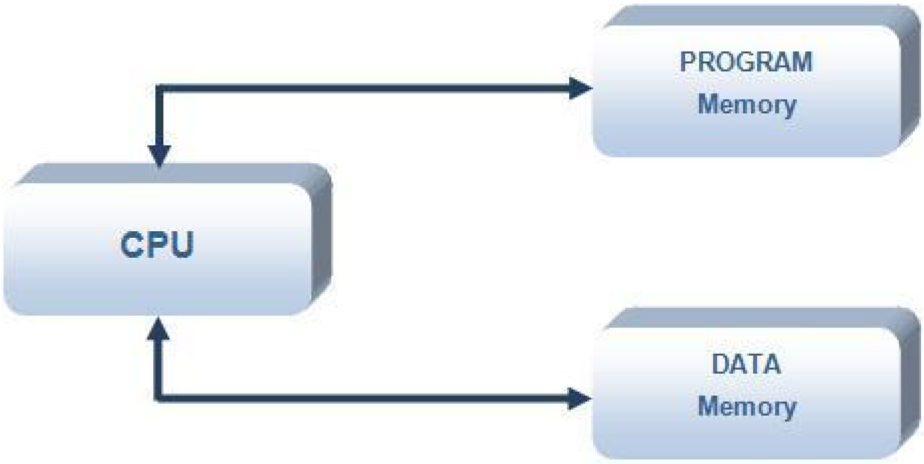


Figure-2.2: AVR Architecture Format

Since AVR can perform single cycle execution, it means that AVR can execute 1 million instructions per second if cycle frequency is 1MHz. The higher is the operating frequency of the controller, the higher will be its processing speed. We need to optimize the power consumption with processing speed and hence need to select the operating frequency accordingly.

There are two flavors for Atmega8 microcontroller:

1. **Atmega8**:- Operating frequency range is 0 – 8 MHz.
2. **Atmega8L**:- Operating frequency range is 0 – 8 MHz.

If we are using a crystal of 8 MHz = 8 x 10^6 Hertz = 8 Million cycles, then AVR can execute 8 million instructions.

Figure-2.3: Pin Diagram of **ATMEGA8**

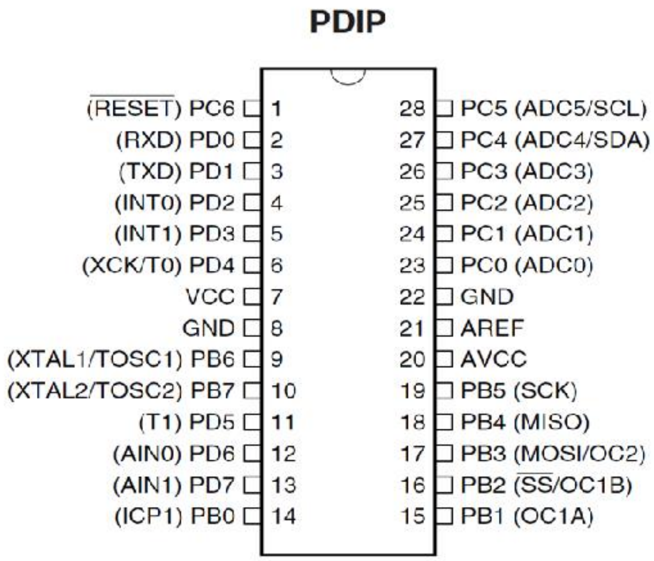
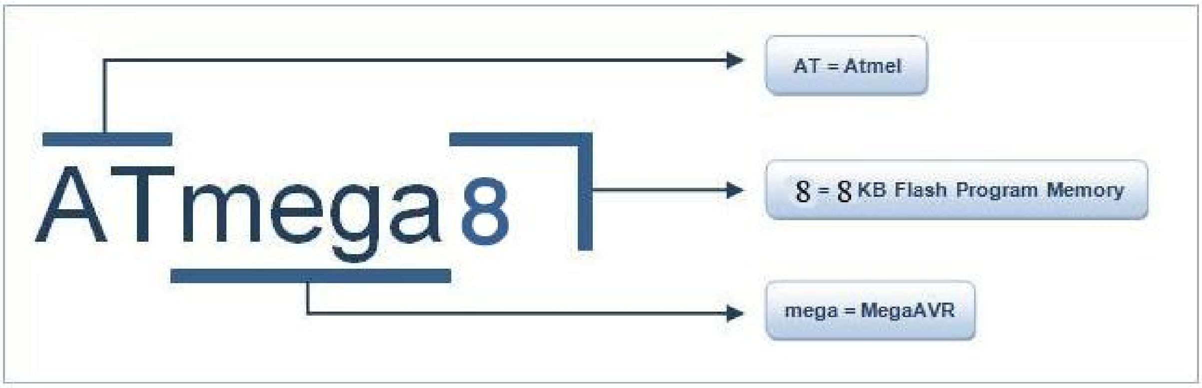


Figure-2.4: Pin Description of **ATMEGA8**

2.3 Naming Convention.

The AT refers to Atmel the manufacturer, Mega means that the microcontroller belong to MegaAVR category, 8 signifies the memory of the controller, which is 8KB.



F

igure-2.5: Naming Convention of Atmega8 - AVR Family

2.4 Architecture Diagram:

<u>**Following points explain the building blocks of architecture**</u>**:**

- <u>I/O Ports</u>: Atmega8 has four (PORTA, PORTB, PORTC and PORTD) 8-bit input-output ports.
- <u>Internal Calibrated Oscillator</u>: Atmega8 is equipped with an internal oscillator for driving its clock. By default Atmega8 is set to operate at internal calibrated oscillator of 1 MHz. The maximum frequency of internal oscillator is 8Mhz. Alternatively, Atmega8 can be operated using an external crystal oscillator with a maximum frequency of 8MHz. In this case you need to modify the fuse bits. (Fuse Bits will be explained in a separate tutorial).

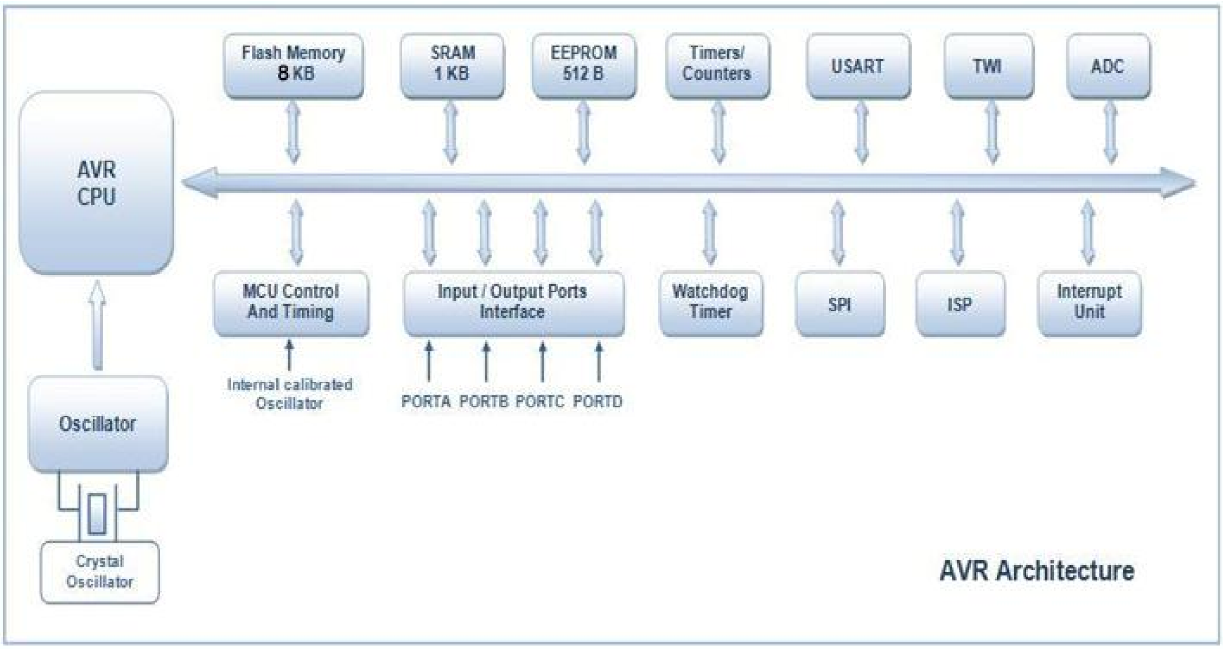


Figure-2.6 Architecture of Atmega8 - AVR Microcontrollers

- **ADC Interface:** Atmega8 is equipped with an 8 channel ADC (**Analog to Digital Converter**) with a resolution of 10-bits. ADC reads the analog input for e.g., a sensor input and converts it into digital information which is understandable by the microcontroller.
- **Timers/Counters:** Atmega8 consists of two 8-bit and one 8-bit timer/counter. Timers are useful for generating precision actions for e.g., creating time delays between two operations.
- **Watchdog Timer:** Watchdog timer is present with internal oscillator. Watchdog timer continuously monitors and resets the controller if the code gets stuck at any execution action for more than a defined time interval.
- **Interrupts:** Atmega8 consists of 21 interrupt sources out of which four are external. The remaining are internal interrupts which support the peripherals like USART, ADC, Timers etc.
- **USART: Universal Synchronous and Asynchronous Receiver and Transmitter** interface is available for interfacing with external device capable of communicating serially (data transmission bit by bit).

2.5 Memory of ATMEGA8

Atmega8 consist of three different memory sections:

1. **Flash EEPROM**: Flash EEPROM or simple flash memory is used to store the program dumped or burnt by the user on to the microcontroller. It can be easily erased electrically as a single unit. Flash memory is non-volatile i.e., it retains the program even if the power is cut-off. Atmega8 is available with 8KB of in system programmable Flash EEPROM.

2. **Byte Addressable EEPROM**: This is also a nonvolatile memory used to store data like values of certain variables. Atmega8 has 512 bytes of EEPROM; this memory can be useful for storing the lock code if we are designing an application like electronic door lock.

3. **SRAM**: Static Random Access Memory, this is the volatile memory of microcontroller i.e., data is lost as soon as power is turned off. Atmega8 is equipped with 1KB of internal SRAM. A small portion of SRAM is set aside for general purpose registers used by CPU and some for the peripheral subsystems of the microcontroller.

- **ISP**: AVR family of controllers have **In System Programmable** Flash Memory which can be programmed without removing the IC from the circuit, ISP allows to reprogram the controller while it is in the application circuit.

- **SPI**: **Serial Peripheral Interface**, SPI port is used for serial communication between two devices on a common clock source. The data transmission rate of SPI is more than that of USART.

- **TWI**: **Two Wire Interface** (TWI) can be used to set up a network of devices, many devices can be connected over TWI interface forming a network, the devices can simultaneously transmit and receive and have their own unique address.

- **DAC**: Atmega8 is also equipped with a **Digital to Analog Converter** (DAC) interface which can be used for reverse action performed by ADC. DAC can be used when there is a need of converting a digital signal to analog signal.

2.6 Various microcontrollers of MegaAVR series:

Atmega32 are other members of MegaAVR series controllers. They are quite similar to Atmega8 in architecture. Low power version MegaAVR controllers are also available in markets. The following table shows the comparison between different members of MegaAVR family:

Part Name	ROM	RAM	EEPROM	I/0 Pins	Timer	Interrupts	Operation Voltage	Operating frequency	Packaging
	8KB	1KB	512B	23	3	19	4.5-5.5 V	0-8 MHz	28
L	8KB	1KB	512B	23	3	19	2.7-5.5 V	0-8 MHz	28
Atmega8	8KB	1KB	512B	32	3	21	4.5-5.5 V	0-8 MHz	40
Atmega8L	8KB	1KB	512B	32	3	21	2.7-5.5 V	0-8 MHz	40
ATmega32	32KB	2KB	1KB	32	3	21	4.5-5.5 V	0-8 MHz	40
ATmega32 L	32KB	2KB	1KB	32	3	21	2.7-5.5 V	0-8 MHz	40

In order to avoid tedious explanations and endless story about the useful features of different microcontrollers, this book describes the operation of one particular model belonging to "high middle class". It is about AVR- powerful enough to be worth attention and simple enough to be easily presented to everybody.

2.7 MICROCONTROLLER FEATURES':

2.7.1 ADVANCED RISC ARCHITECTURE.

- 130 Powerful Instructions – Most Single-clock Cycle Execution.
- 32 x 8 General Purpose Working Registers.
- Fully Static Operation.
- Up to 8 MIPS Throughput at 8 MHz.
- On-chip 2-cycle Multiplier.

2.7.2 HIGH ENDURANCE NON-VOLATILE MEMORY SEGMENTS.

- 8K Bytes of In-System Self-programmable Flash program memory.
- 256 Bytes EEPROM.
- 1K Byte Internal SRAM.
- Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
- 1K Byte Internal SRAM.
- Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
- Data retention: 20 years at 85°C/100 years at 25°C.
- Optional Boot Code Section with Independent Lock Bits.
- In-System Programming by On-chip Boot Program. True Read-While-Write Operation
- Programming Lock for Software Security

2.7.3 PERIPHERAL FEATURES.

- Timer0: 8 – bit Timer / Counter with 8- bit prescaler
- Timer1: 8- bit Timer / Counter with prescaler can be incremented during Sleep via External Crystal/clock
- Timer2: 8 – bit Timer / Counter with 8- bit period Register, rescale and prescaler
- One 8-bit Timer/Counter with Separate prescaler, Compare Mode, and Capture
- 10 bit multi-channel Analog to digital converter
- Synchronous Serial port (SSP) with SPITM (Master mode) and I2CTM (Master/ Slave)
- Real Time Counter with Separate Oscillator.
- Three PWM Channels.
- 8-channel ADC in TQFP and QFN/MLF package Eight Channels 10-bit Accuracy.
- 6-channel ADC in PDIP package Six Channels 10-bit Accuracy
- Byte-oriented Two-wire Serial Interface.
- Programmable Serial USART.
- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator.
- On-chip Analog Comparator.

2.7.4 SPECIAL MICROCONTROLLER FEATURES.

- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated RC Oscillator.
- External and Internal Interrupt Sources.
- Eight Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby.

2.7.5 I/O AND PACKAGES.

- 32 Programmable I/O Lines.
- 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF.

2.7.6 OPERATING VOLTAGES AND CURRENT

- 2.7V - 5.5V (L)
- 4.5V - 5.5V (H)

2.7.7 OPERATING SPEED GRADES:

- Clock input ()
- 0 - 8MHz (L)
- 0 - 8MHz ()

2.7.8 POWER CONSUMPTION () AT 4MHz 3V, 25°C.

- Active: 3.6 mA
- Idle Mode: 1.0 mA
- Power-down Mode: 0.5 µA.

2.8 Microcontroller clock

In order to step through the instructions the microcontroller needs a clock frequency to orchestrate the movement of the data around its ele4ctrinics results. This can be provided by 2 capacitors and a crystal or by an internal oscillator circuit.
In the 8F84 microcontroller there are four oscillator options.

- An RC (Resister/Capacitor) oscillator which provides a low cost solution.
- An LP oscillator, i.e. 32 KHZ crystal, which minimizes power consumption.
- XT which uses a standard crystal configuration.
- HS Is the high speed oscillator option.

Common crystal frequencies would be 32 KHZ, 1MHZ, 4MHZ, 10MHZ and 20MHZ.
Inside the microcontroller there is an area where the processing (the clever work), such as mathematical and logical operations are performed, this is known as the central processing unit or CPU. There is also a region where event timing is performed and another for interfacing to the outside world through parts.

2.9 THE MICROCONTROLLER SYSTEM:

The block diagram of the microcontroller is shown in figure.

Figure-2.7: The basic microcontroller system

- The input components would consist of digital devices such as switches, push buttons, pressure mats, flat switches, keypads, radio receivers etc. and analogue sensors such as light dependent resistors, thermostats, gas sensors etc.
- The control unit is of course the microcontroller. The microcontroller will monitor the inputs and as a result the program would turn outputs on and off. The microcontroller stores the program in its memory, and executes the instructions under the control of the clock circuit.
- The output devices would be made up from LEDS, buzzers, motors, radio transmitters, 7 segment displays, heaters, fans etc.

The most obvious choice then for the microcontroller is how many digital inputs, analogue inputs and outputs does the system require. This would then specify the minimum number of inputs and outputs that the microcontroller must have. If the analogue inputs are used then the microcontroller must have an Analogue to Digital module inside.

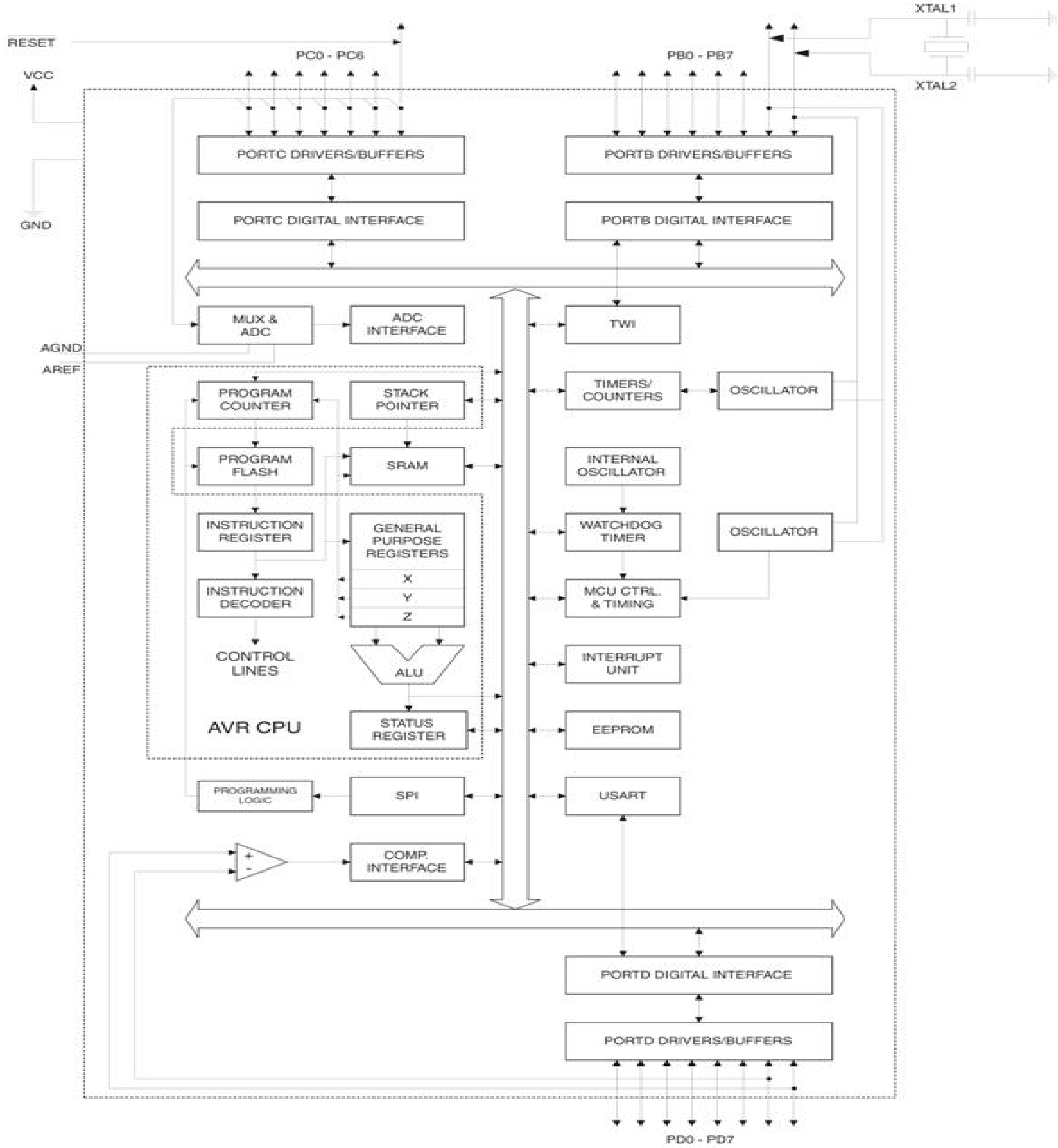


Figure-2.8: Block Diagram of AVR Microcontroller

CHAPTER – III

INTRODUCTION OF SMART ROOM
or
HOME AUTOMATION SYSTEM

3.1 SMART HOME CONTROL SYSTEM:

Figure-3.1: Smart Room System

Smart home provides a single system approach to the management of an extensive array of prestige electronics and controls. Like a talented conductor guiding and overseeing the virtuosos in an orchestra, Smart room Smart Homes Control Systems are designed to direct the various individual smart home technologies and reduce their management to a single, user-friendly system that can incorporate:

- Multi Room Audio and Video Distribution
- Automated Gates and Locks
- Intercom, CCTV and Security Systems
- Multiple Scene Lighting Design
- Motorized Blinds and Curtains
- Environmental Controls & HVAC

Smart room Touch-Screens and Touch-Panels can be used to control multiple zones of distributed audio and video, as well as adaptable scene lighting design, thereby providing the potential for infinite creativity and variety in different areas of the home. It can also maintain home security by automatically locking and unlocking windows, doors and gates, and by managing CCTV and intercom hardware and operating systems.

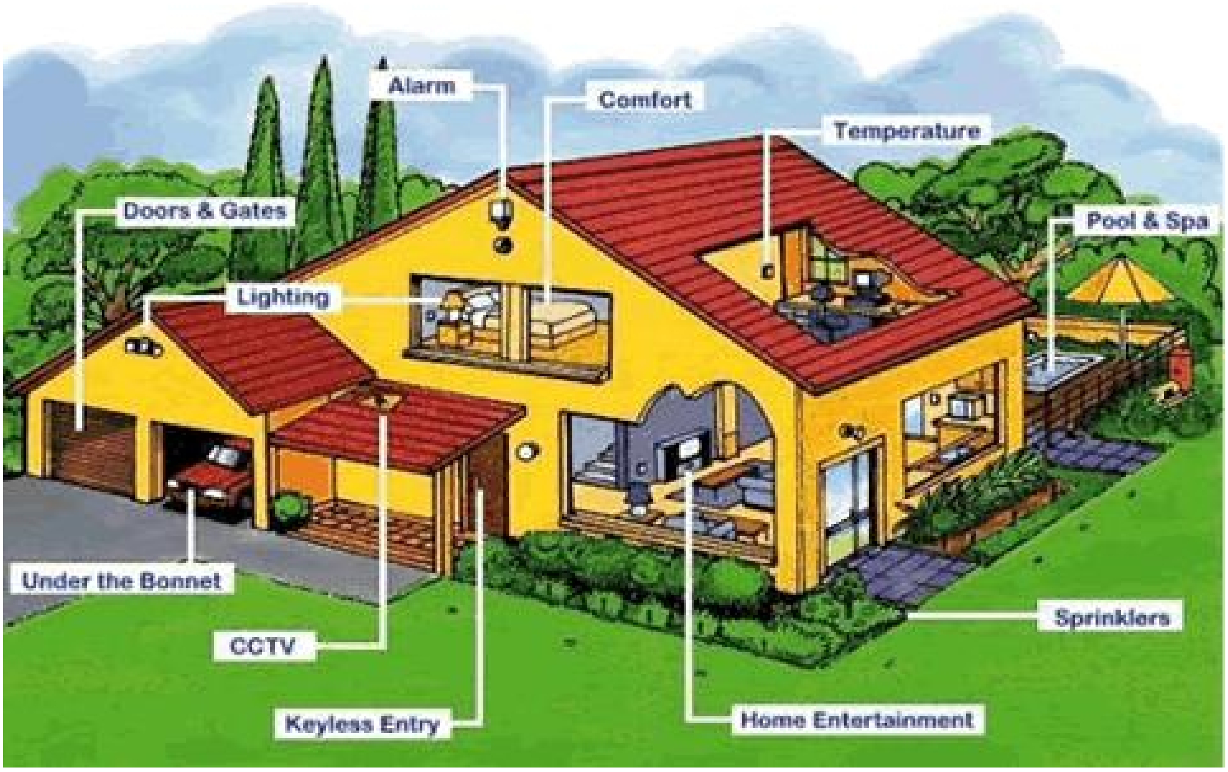


Figure-3.2: Smart Home System

One hugely important feature of the system is the ability to access, survey and control elements of your home from anywhere in the world via the internet, which has incredibly important implications for security, home maintenance and peace of mind.

Smart Home Control brings an Extravagant Simplicity to the combined power of your 21st Century technologies, with a beautifully simple interface and a reduced number of menu options for what would otherwise be a long and complicated checklist of processes. With Smart room Smart Home taking care of your custom installation, you can be sure of a bespoke and immaculate system to complement the personal design and feel that you have chosen to create in your home.

Security is a prime concern in our day-today life. Everyone wants to be as much secure as possible. An access control for doors forms a vital link in a security chain. The microcontroller based Door locker is an access control system that allows only authorized persons to access a restricted area. This system is fully controlled by the 8 bit microcontroller ATmega8. The password is stored in the EPROM so that we can change it at any time.

The system has a Keypad by which the password can be entered through it. When the entered password equals with the password stored in the memory then the relay gets on and so that the door is opened. If we entered a wrong password for more than three times then the Alarm is switched on.

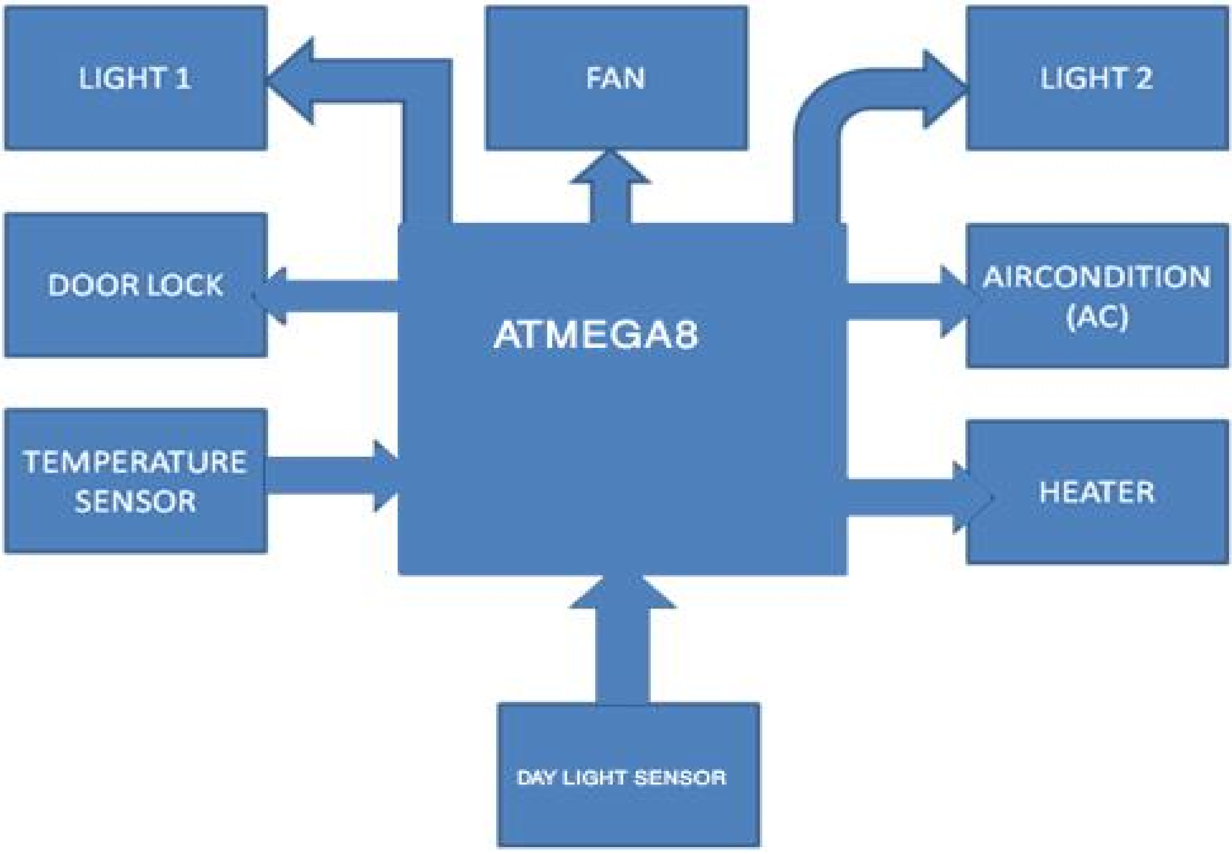


Figure-3.3: BLOCK DIAGRAM OF SMART ROOM

CHAPTER – IV

CIRCUIT DIAGRAM, DESCRIPTION & OPERATION

4.1 AC AND HEATER CONTROL:

4.1.2 CIRCUIT DIAGRAM

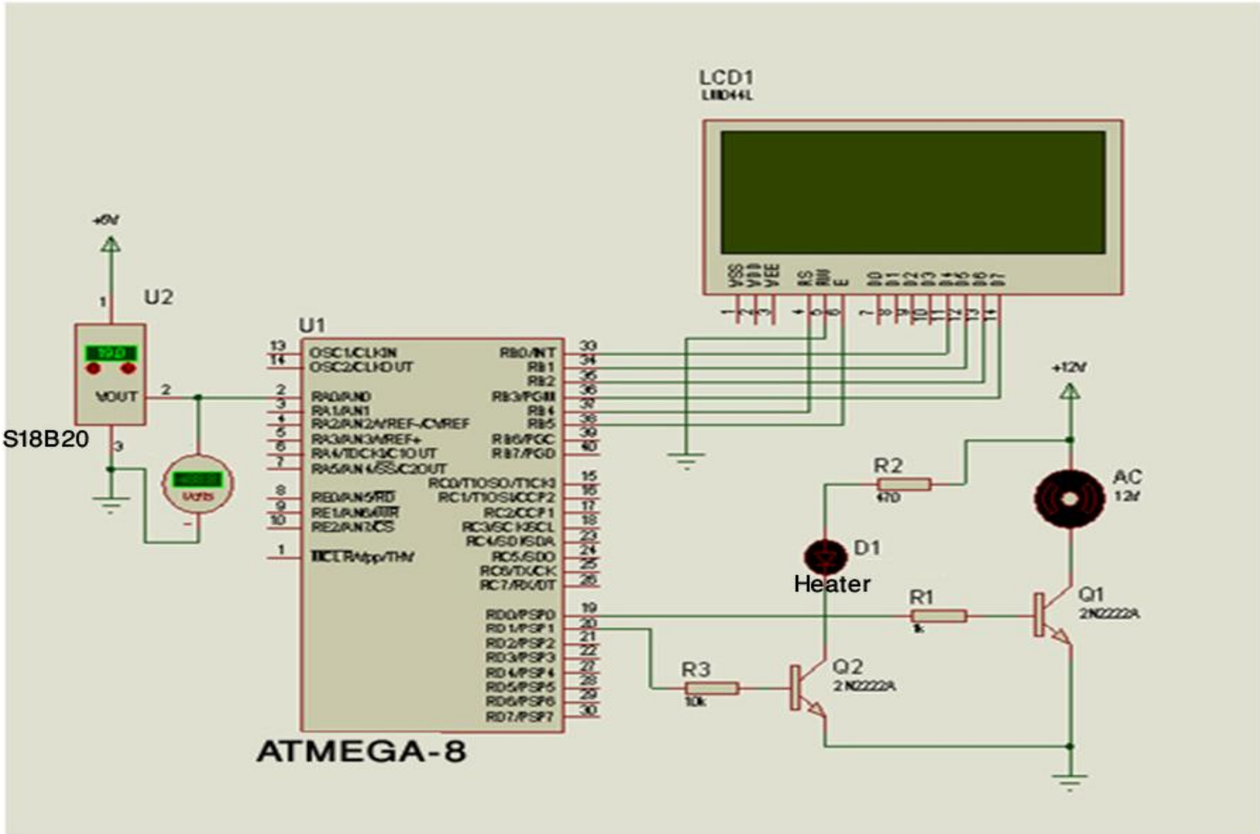


Figure-4.1: Circuit diagram of AVR microcontroller based AC and Heater Control system.

4.1.2 CIRCUIT DESCRIPTION:

The Port of the microcontroller RB2 is used as input and Port RD (11-12) used as output. Input pin is connected with a Temperature Sensor Which sense the temperature. The two output terminals are connected with the base terminal of two transistors which are help to ON the transistors by biasing the gate terminal. Two Relay connected with the transistor collector terminal which are helps to connect or disconnect the loads or parts of the circuit.

4.1.3 CIRCUIT OPERATION:

Temperature sensor is sense temperature and convert electrical signal. This signals one wire serial digital signal. This sensor can measure temperature & show on LCD. If temp>25 Then PORTD5=1 that is AC is on and If temp >=24.5 then PORD5=0 that is AC is off. If temp<18 Then PORTD6=1 that is Heater is on and If temp>= 24 Then PORTD6=0 that is Heater is off.

4.2 Temperature meter, day light sensor & water level sensor

4.2.1 CIRCUIT DIAGRAM

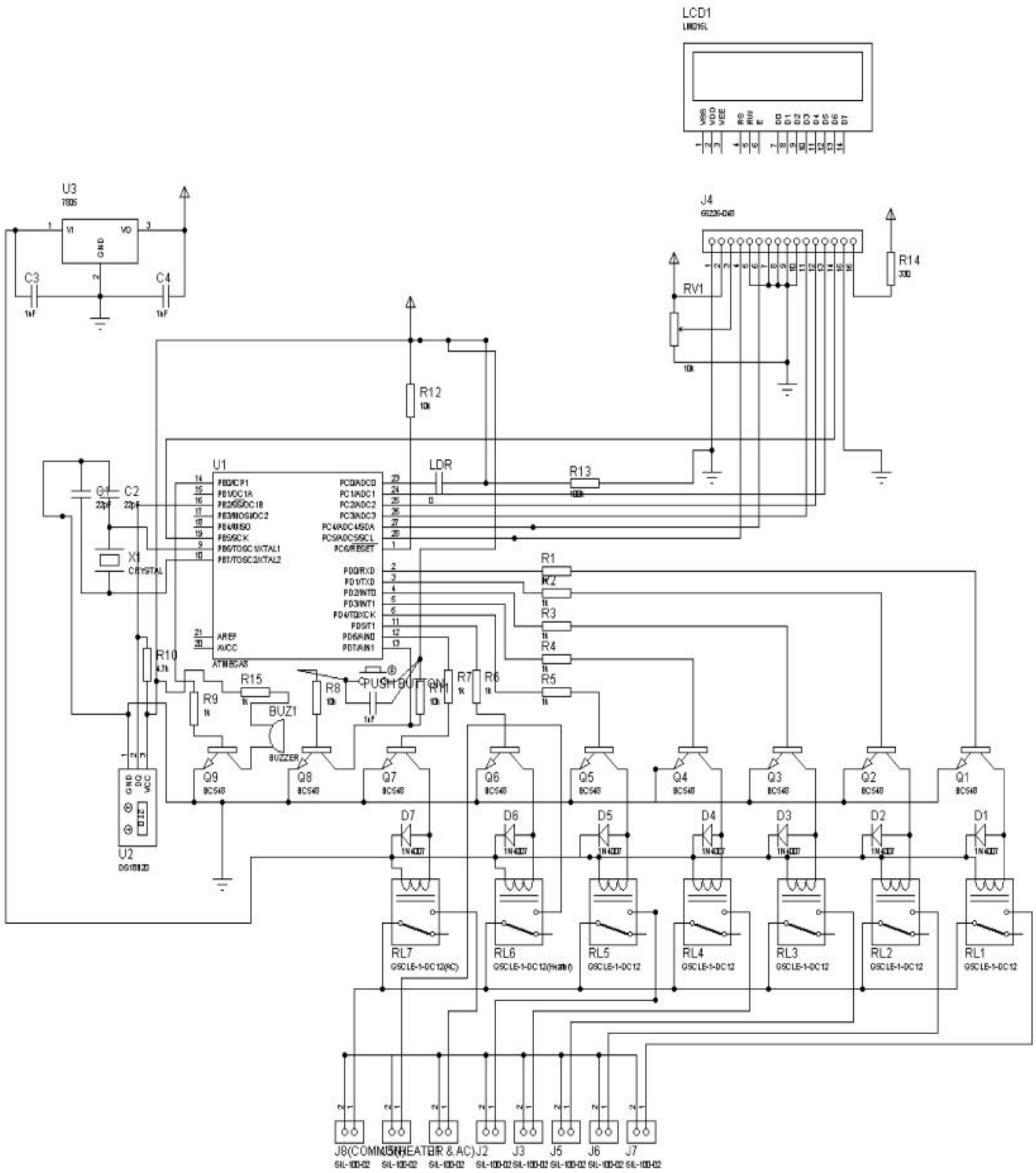


Figure-4.2: Integration of Temperature meter, day light sensor & water level sensor

4.2.2 CIRCUIT DESCRIPTION:

The Port of the microcontroller RD7 is used as input and Port RB 0 used as output. Input pin is connected with a water level sensor. When the input pin RD7 will low then micro controller sense water on floor and generate a signal and high the port RB 0 then a buzzer will generate an alarm signal for us.

4.2.3 CIRCUIT OPERATION:

The buzzer will be on when corresponding pin low. For alarming us that water on floor at room.

4.3 REMOTE CONTROL CIRCUIT

4.3.1 CIRCUIT DIAGRAM

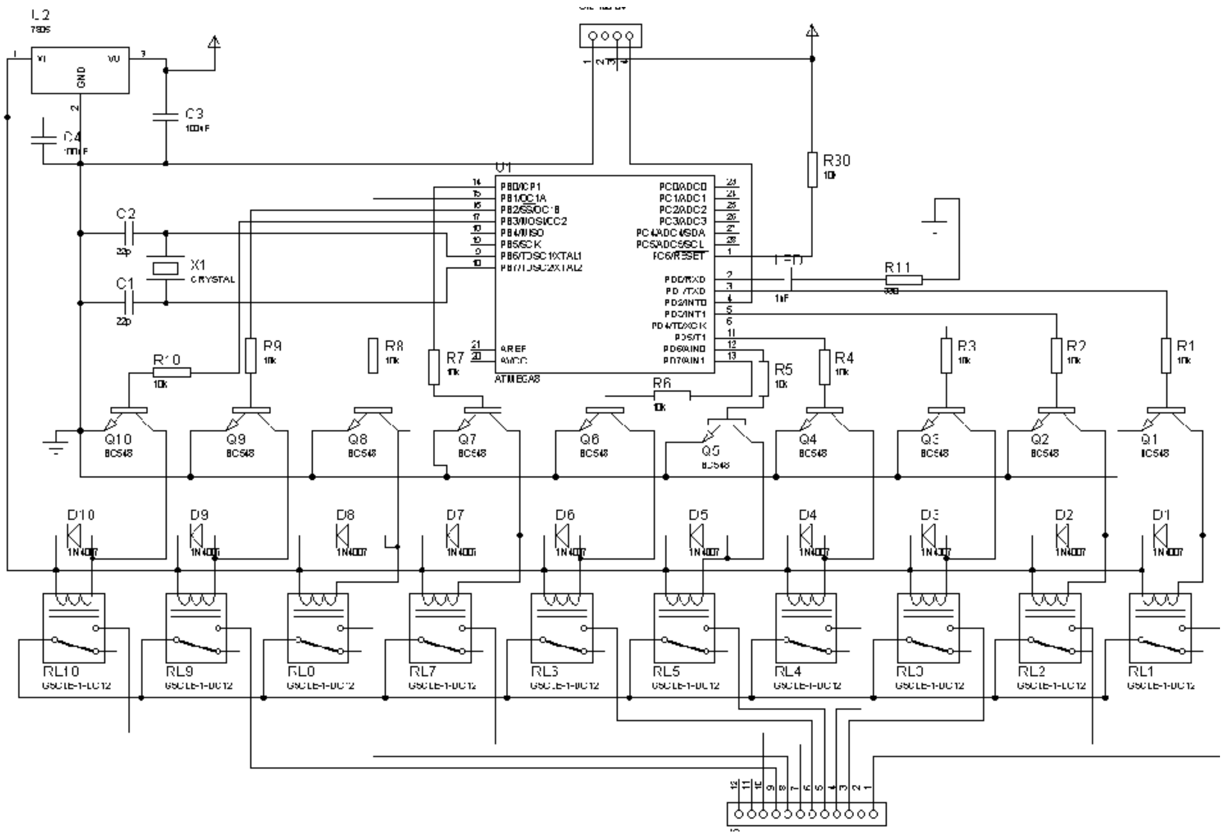


Figure-4.4: remote control circuit.

4.3.2 CIRCUIT DESCRIPTION:

The Port of the microcontroller RD (2) is used as input and Port RD (1-7) and Port RB (0-3) used as output. Input pin is connected with a decoder TSOP 1738. The Three output terminals are connected with the base terminal of three transistors which are help to ON the transistors by biasing the gate terminal. Three Relay connected with the transistor collector terminal which are helps to connect or disconnect the loads or parts of the circuit.

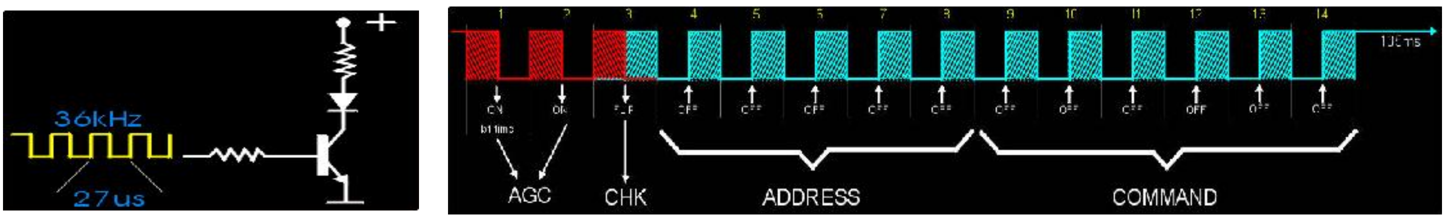


Figure-4.3: remote signal generator and generated signal.

4.3.2 CIRCUIT OPERATION:

The loads (fan and Light) will be on when corresponding remote switch is press and will be on/ off when corresponding switch signal press.

CHAPTER – V

SOFTWARE, SIMULATION & IMPLEMENTATION

This chapter discusses how the software for this thesis was implemented into the device. The process undertaken to reach a software solution which meets the specifications.

5.1 Software Development

The Firmware for the controller was developed with the aid of BASCOM AVR, which is a program that allows the user to write and compile BASIC programs for the Atmel AVR range of micro-controllers. It also allows for in-system programming thus making an ideal software development tool.

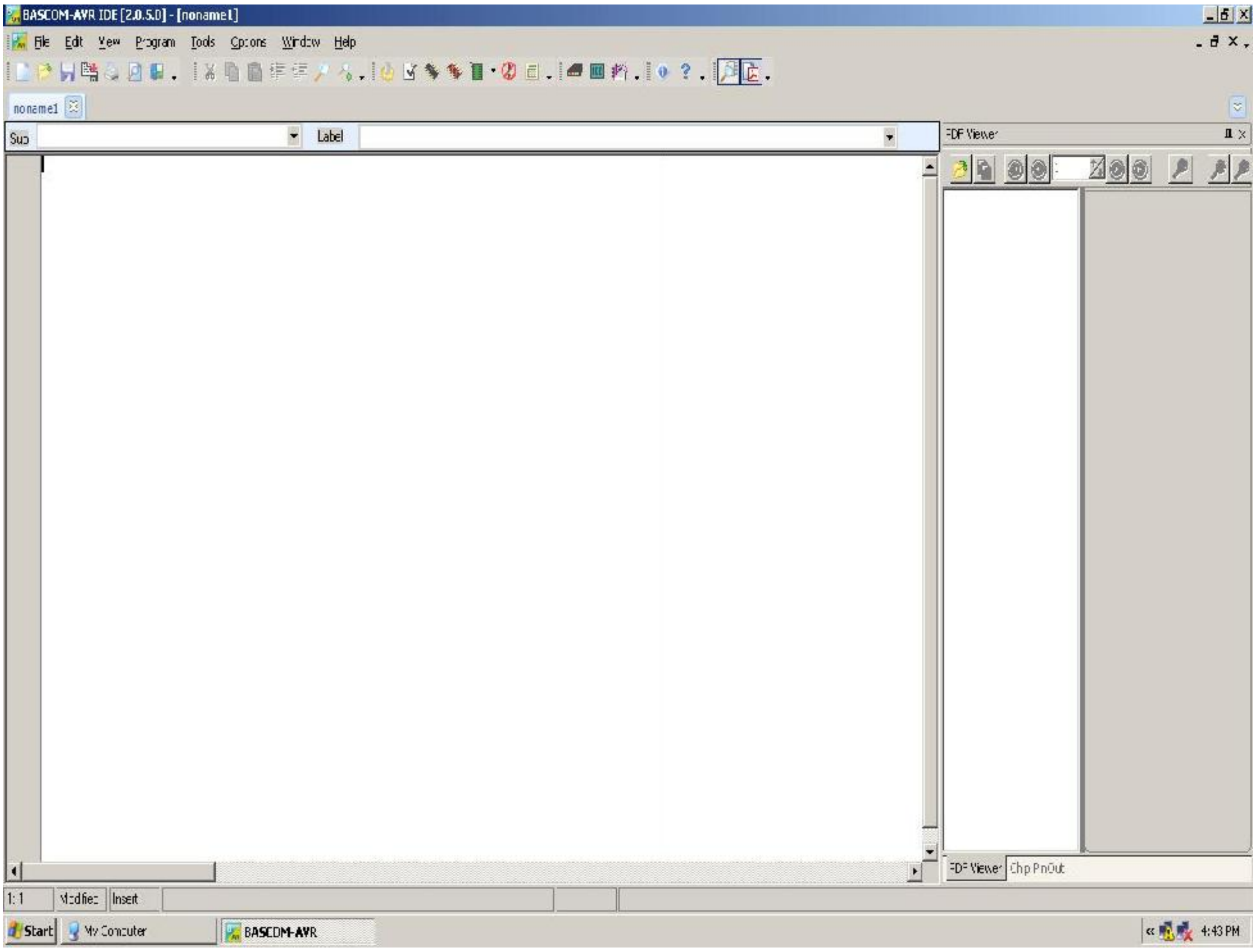


Figure-5.1: BASCOM AVR microcontroller development system.

5.1.1 Chief selection

The fast task is to select the chip. Here for our work we select the chip is ATmega8 and we also determined the clock 16.000000 MHz Figure 01, below, shows the new AVR project window within BASCOM AVR

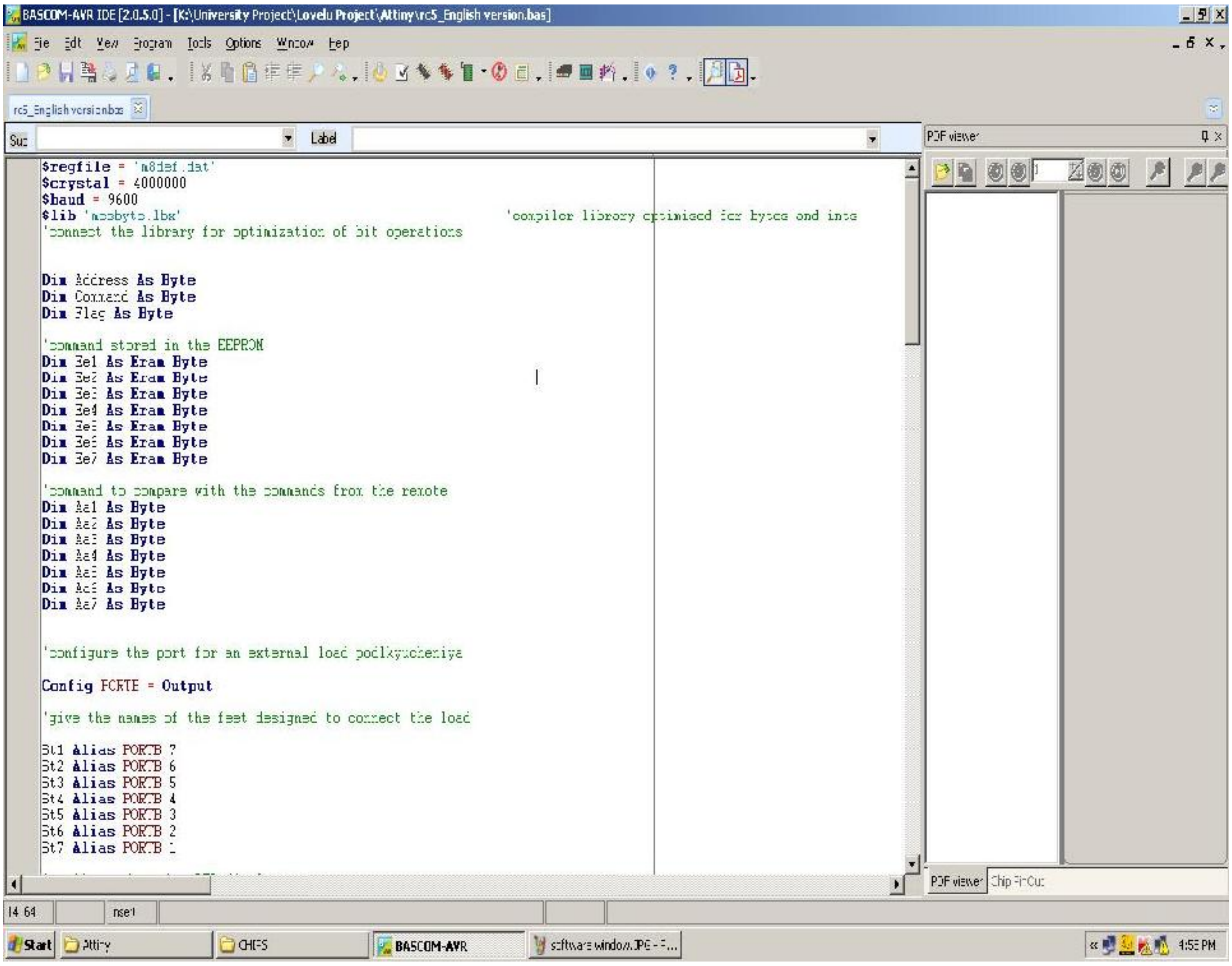


Figure-5.2: BASCOM AVR Window Showing Chip selection Options

5.1.2 Enabling Analogue to Digital Conversion

Enabling the ADC is needed for use in the control functions of the device. Using the ADC is also made simpler by using BASCOMAVR. Enabling ADC on the MCU is achieved by ticking the ADC Enabled selection in BASCOMAVR, and then selects the use 8 bit option and then selects the voltage reference as the voltage across the AREF pin. BASCOMAVR also gives an option for the ADC speed, and in this case it was chosen to be the fastest available, that being 1000.000 kHz.

5.1.3 Enabling Port D as an output

Enabling the Port C & D as output is needed for use in the control functions of the device. Using the port D (0~5). Port C use for LCD display.

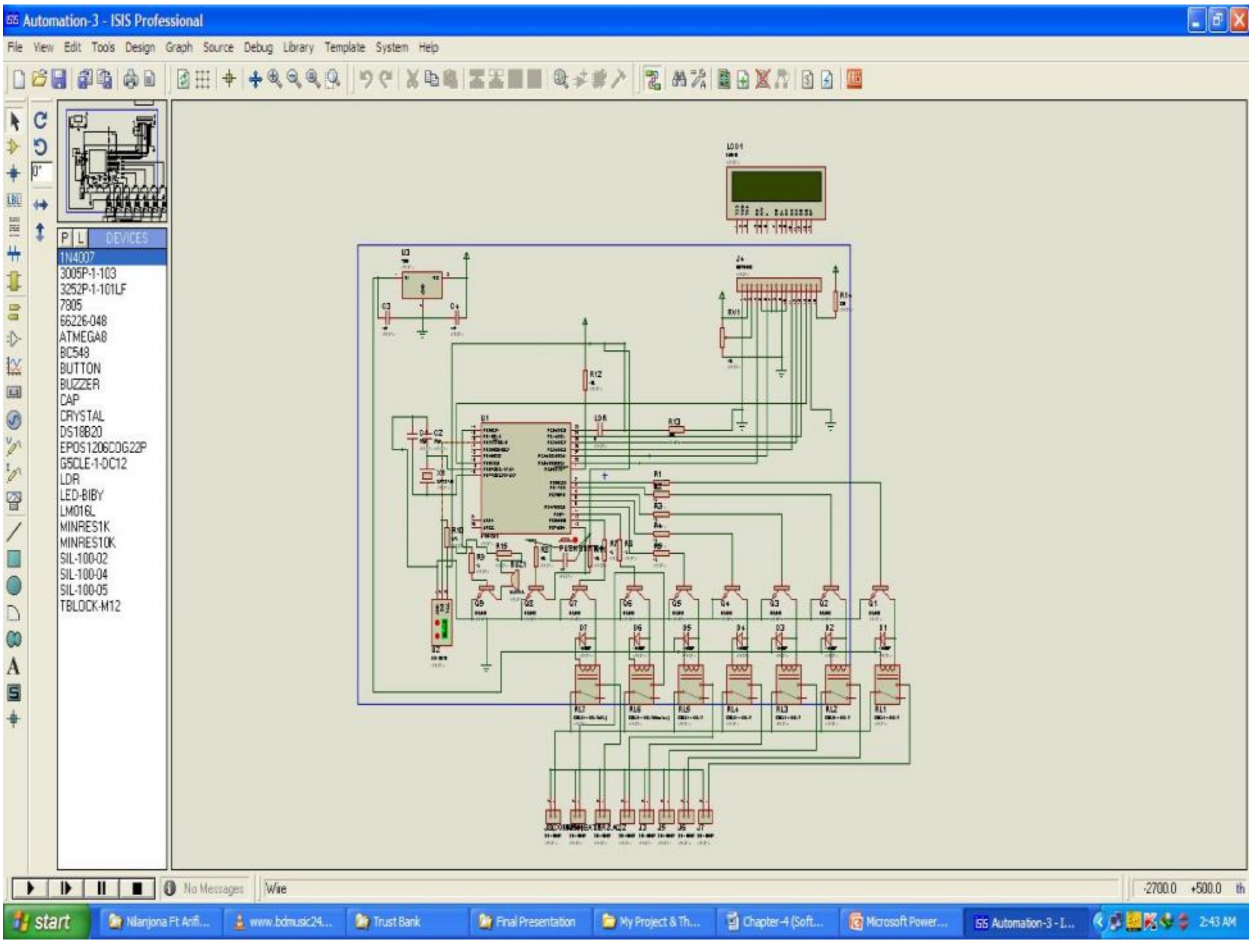


Figure-5.3: BASCOM AVR Window Showing Circuit Diagram Options

5.2 FINAL CODE

Chip type: ATmega8
Program type: Application
AYR Core Clock frequency: 16.000000 MHz
Memory model: Small 8KB
External RAM size : 0
Data Stack size : 256

```
$regfile = "m8def.dat"
$crystal = 16000000
$baud = 9600
$lib "mcsbyte.lbx”                                  'compiler library
optimized for bytes and ints
                                                    'Connect the library
for optimization of bit operations


Dim Address As Byte
Dim Command As Byte
```

```
Dim Flag As Byte
```

```
'command stored in the EEPROM
Dim Ee1 As Eram Byte
Dim Ee2 As Eram Byte
Dim Ee3 As Eram Byte
Dim Ee6 As Eram Byte
Dim Ee5 As Eram Byte
Dim Ee6 As Eram Byte
Dim Ee7 As Eram Byte

'command to compare with the commands from the remote
Dim Aa1 As Byte
Dim Aa2 As Byte
Dim Aa3 As Byte
Dim Aa6 As Byte
Dim Aa5 As Byte
Dim Aa6 As Byte
Dim Aa7 As Byte

'configure the port for an external load podlkyucheniya

Config Portb = Output

'give the names of the feet designed to connect the load

St1 Alias Portb.7
St2 Alias Portb.6
St3 Alias Portb.5
St6 Alias Portb.6
St5 Alias Portb.3
St6 Alias Portb.2
St7 Alias Portb.1                      'configure feet for LED display

Config Portd.6 = Output
Config Portd.5 = Output

Red Alias Portd.6                                            'red LED
Green Alias Portd.5                                          'green LED

Config Rc5 = Pind.2                                          'configure step
to connect the sensor

On Int1 Button                                               'connect button
Config Int1 = Falling                                        'termination of
the button is triggered by falling edge

Enable Interrupts                                            'enables the
interrupt
Enable Int1                                                  'enable external
interrupt

'take out the value of teams from non-volatile memory and assign a variable
working
Aa1 = Ee1
Aa2 = Ee2
Aa3 = Ee3
Aa6 = Ee6
Aa5 = Ee5
Aa6 = Ee6
Aa7 = Ee7

Do
```

```
Getrc5(address , Command)                                    'address and
take command from the remote
```

```
  If Address >= 0 And Address < 32 Then                    'work with any
remote control

      Green = 1                                            'lit green LED

      Command = Command And &B01111111

      If Flag = 0 Then

           Select Case Command                             'If the flag is
not worth recording, select which port switch

            Case Aa1 : Toggle St1
            Case Aa2 : Toggle St2
            Case Aa3 : Toggle St3
            Case Aa6 : Toggle St6
            Case Aa5 : Toggle St5
            Case Aa6 : Toggle St6
            Case Aa7 : Toggle St7

          End Select

       Else                                                'otherwise, if
the flag is worth writing, write the command in the EEPROM

        Select Case Flag                                   'depending on
the number of write command flag in its memory cell

           Case 1 : Ee1 = Command                          'write the team

number in the EEPROM

           Case 2 : Ee2 = Command
           Case 3 : Ee3 = Command
           Case 6 : Ee6 = Command
           Case 5 : Ee5 = Command
           Case 6 : Ee6 = Command
           Case 7 : Ee7 = Command

        End Select

         Flag = 0                                          'reset the flag
entries
         Red = 0                                           'extinguish the
red LED

        Aa1 = Ee1                                          'take out all of
the non-volatile memory

        Aa2 = Ee2
        Aa3 = Ee3
        Aa6 = Ee6
        Aa5 = Ee5
        Aa6 = Ee6
        Aa7 = Ee7

      End If

    Waitms 200
    Green = 0                                    'extinguish the green LED

    Wait 1
```

```
End If
```

```
Loop

'interrupted by pressing

  Button:

Red = 1                                                      'Red LED is lit

  Incr Flag

    If Flag = 8 Then                                         'If you press
the button 8 times
     Flag = 0                                                'Team record
will not
     Red = 0                                                 'LED will
extinguish
    End If

 Select Case Flag                                            'Depends on the
value at which the leg is a

  Case 1 : Portb = &B10000000
  Case 2 : Portb = &B01000000
  Case 3 : Portb = &B00100000
  Case 6 : Portb = &B00010000
  Case 5 : Portb = &B00001000
  Case 6 : Portb = &B00000100
  Case 7 : Portb = &B00000010
  Case 0 : Portb = &B00000000

 End Select

  Waitms 600                                                 'Get rid of the
bounce button
  Gifr = 128                                                 'Reset bit of
the interrupt INT1

  Return

End
```

5.3 Designs on Proteus:

The **Designs** for the controller was developed with the aid of **Proteus**, which is a program that allows the user to design schematic, simulate and design PCB for the Atmel AVR range of micro-controllers. It also allows for in-system simulation thus making an ideal computer based tool.

5.3.1 Here is given below Schematic design for remote control

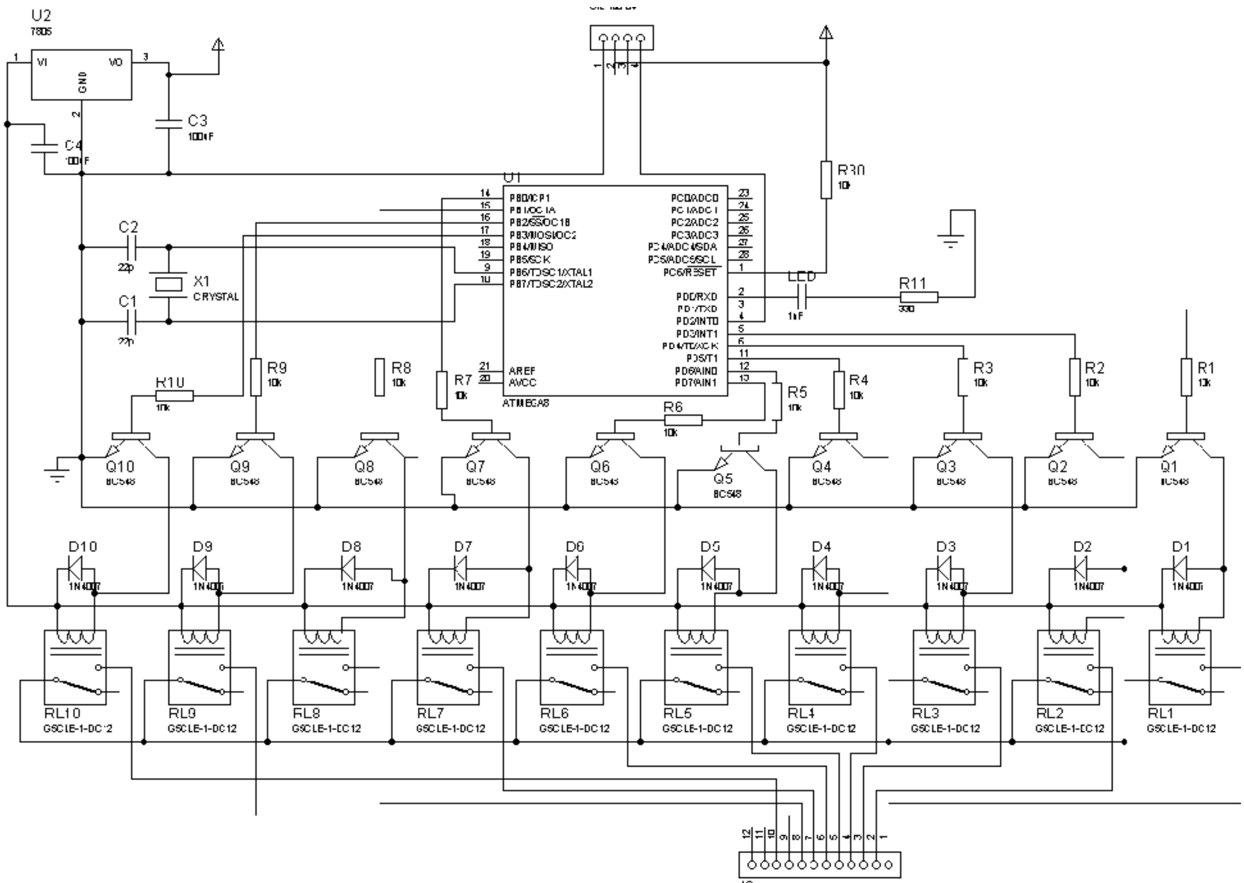


Figure-5.4: Design on Proteus for remote control

5.3.2 Here is given below PCB design for remote control

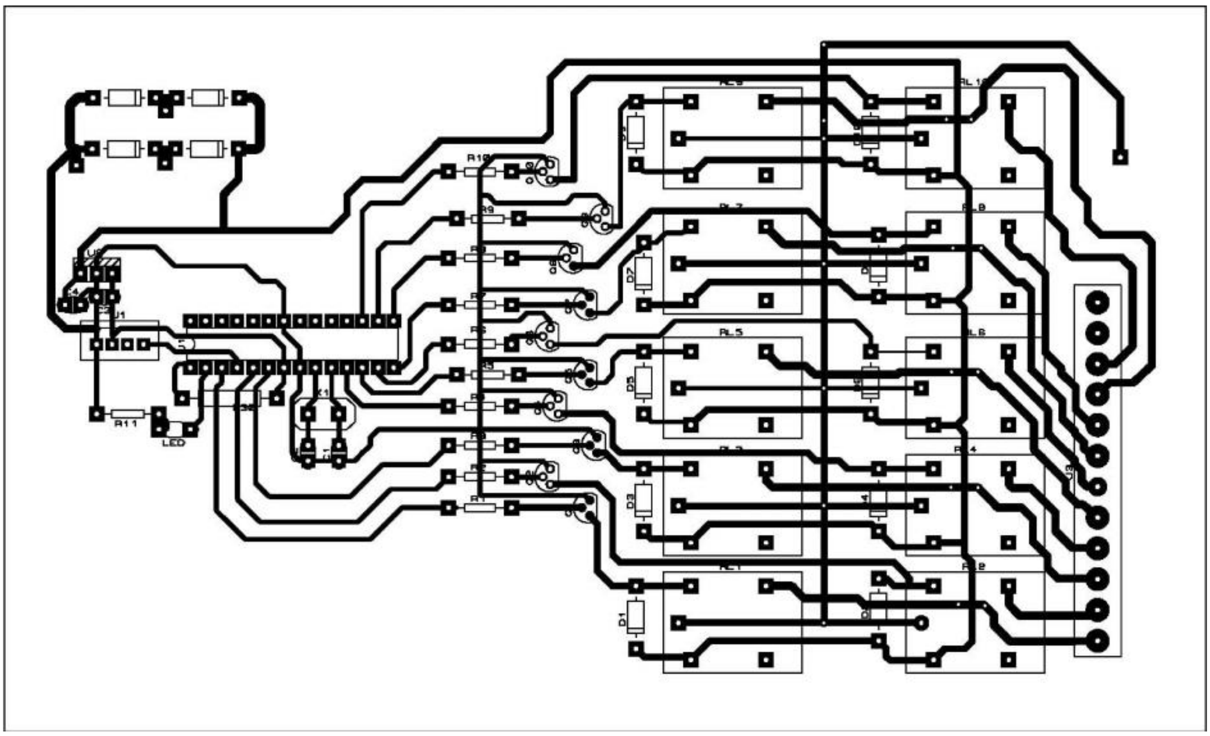


Figure-5.5: PCB design for remote control

5.3.3 Here is given below 3D design for remote control

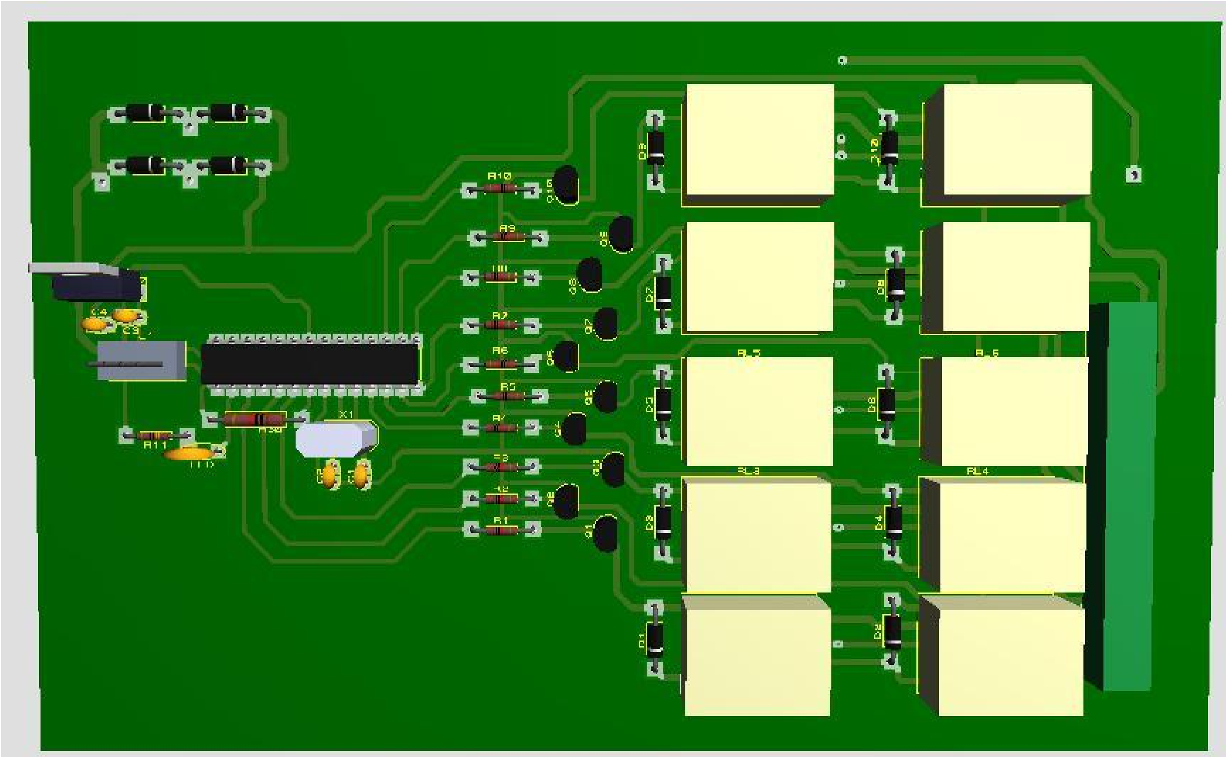


Figure-5.6: 3D design for remote control

5.3.4 Here is given below Schematic design for temperature meter, controller, day light sensor, & water level sensor

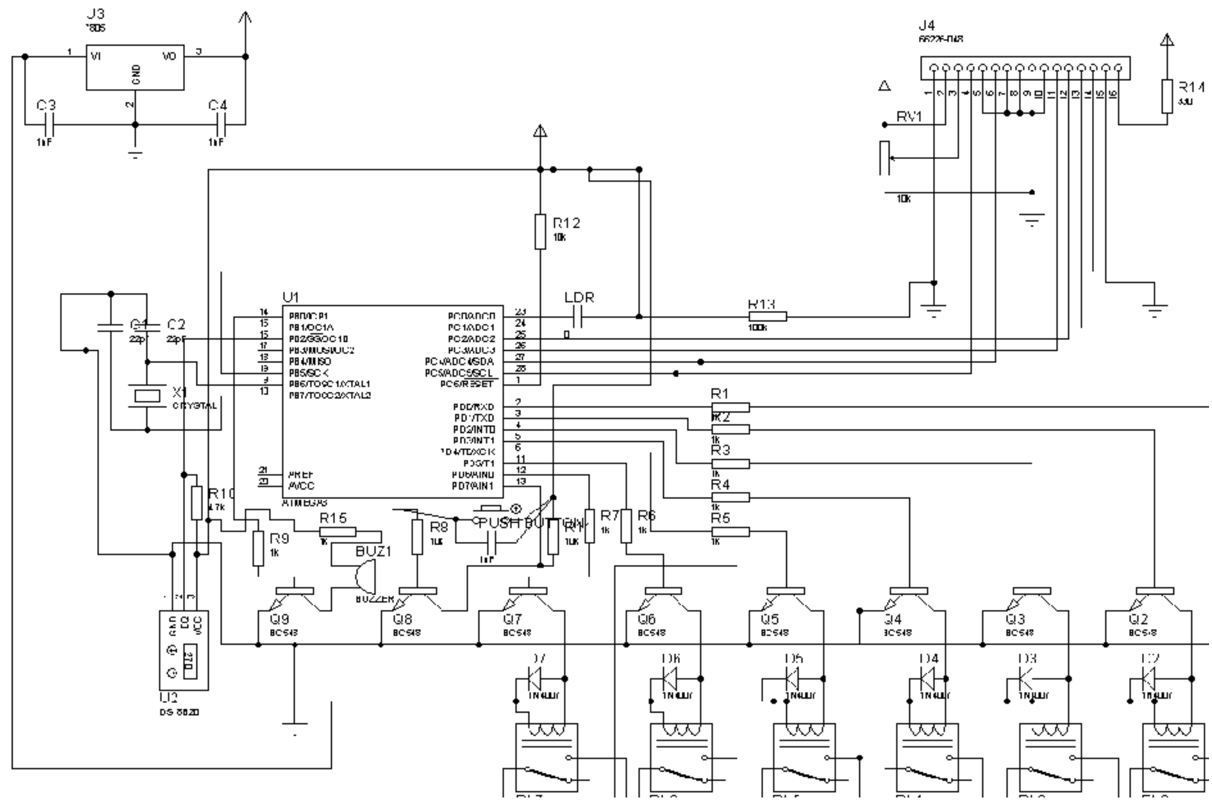


Figure-5.7: temperature meter, controller, day light sensor, & water level sensor

5.3.5 Here is given below PCB design for temperature meter, controller, day light sensor, & water level sensor

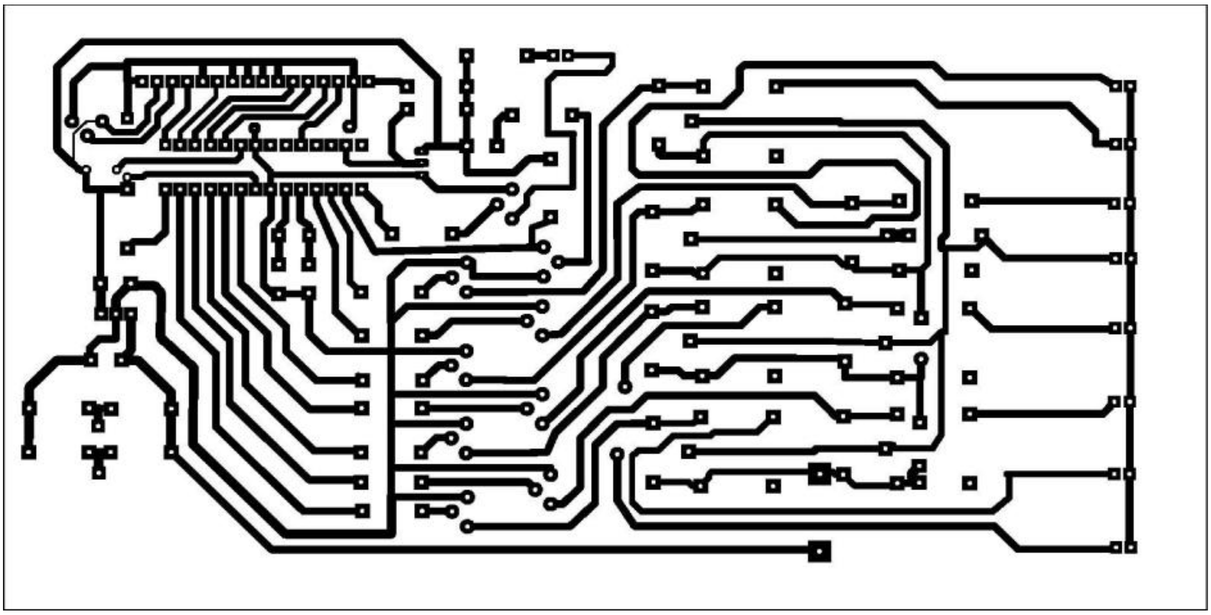


Figure-5.8: PCB Design temperature meter, controller, day light sensor, & water level sensor

5.3.6 Here is given below 3D Design temperature meter, controller, day light sensor, & water level sensor

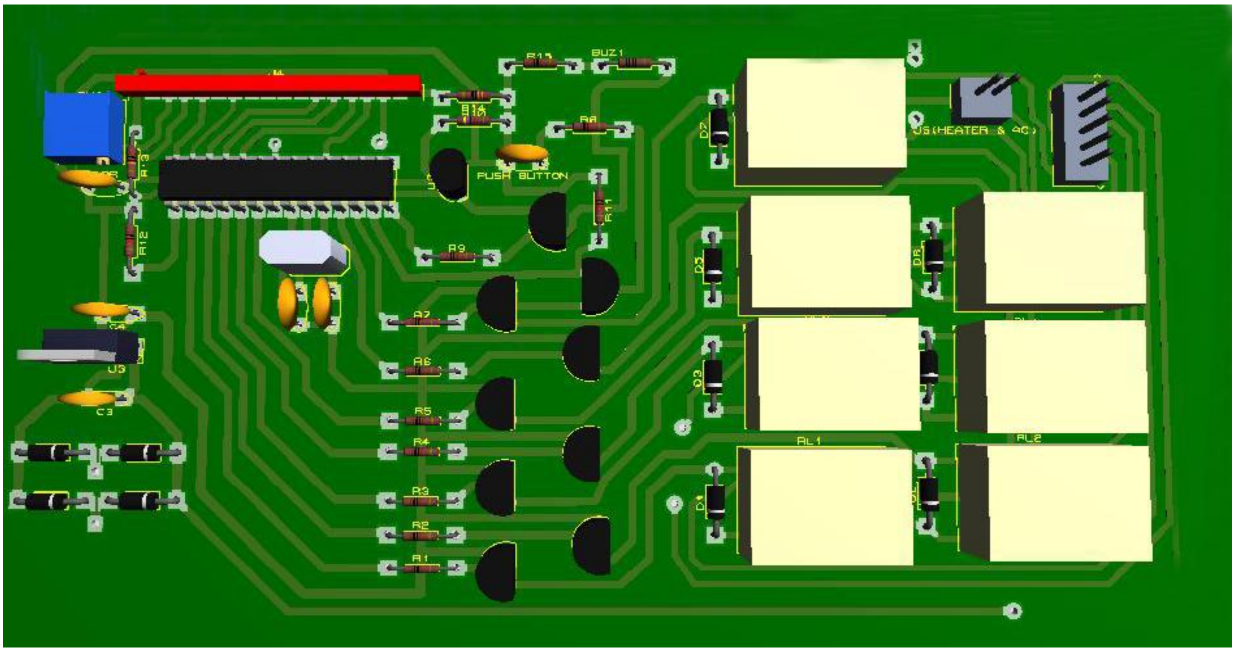


Figure-5.9: 3D Design temperature meter, controller, day light sensor, & water level sensor

5.4 SUMMERY

This chapter has detailed the software implementation into the ATMega8 programmable device.

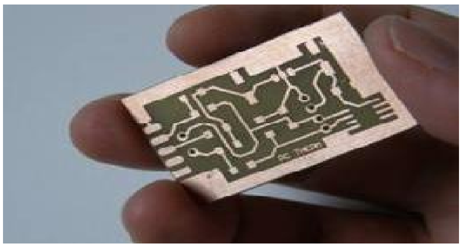
CHAPTER – VI

PCB DESIGNING PROCEDURE

In this chapter we are discussed about Printed Circuit Board Design Procedure at home.

6.1 PCBs at home

Here we're going to show how to make simple single-sided PCBs in a snap, using widely available materials. This technique works reliably for thin tracks down to 10 mils, and is suitable for most surface-mount parts.



6.2 Necessary Materials & Uses

Required materials		We used...	Where to find
	**Magazines Paper**	*magazine paper*	*We bought it from paper shop*
	**Laser printer**	*Samsung ML1610 with original cartridge. Inkjet printers don't work.*	*We have own printer*
	**Household clothes iron**	*Philips clothes iron*	*At my home*
	**Copper clad laminate**	*laminate 1.6 mm thick (35um copper)*	*At electronics shop*
	**Etching solution**	*Ferric chloride solution, 1 litter.*	*At chemical shop*
	**Kitchen scrubs**	Kitchen scrubs	*At kitchen*
	**Thinner (e.g. acetone)**	Thinner	*From hardware shop*

We also used:

- Cutter/Blade
- Insulation tape
- Sand & kitchen paper
- Cotton wool & vice
- Hacksaw.

6.3 Making Procedure:

6.3.1 Theory:

Laser printers use plastic toner, not ink, to draw images. Toner is the black powder. Toner is plastics, so toner is resistant to etching solutions which is used for making PCBs - if only it marked on copper!

Here we use the toner-transfer principle. Like most plastics, toner melts with heat, turning in a sticky, glue-like paste. So it can print on paper as usual, place the sheet face-down on PCB copper, and melt toner on copper applying heat and pressure. Then we got paper toner-glued to PCB copper. Last step is to find a way to remove paper leaving toner on the copper.

6.3.2 The paper for print:

The perfect paper should be: glossy, thin, and cheap. This kind of stuff that looks lustrous and shiny when new, but so cheap it quickly turns into pulp when wet. Almost any glossy magazine paper will work. I like thin paper over thick one, and prefer recycled paper over new paper.

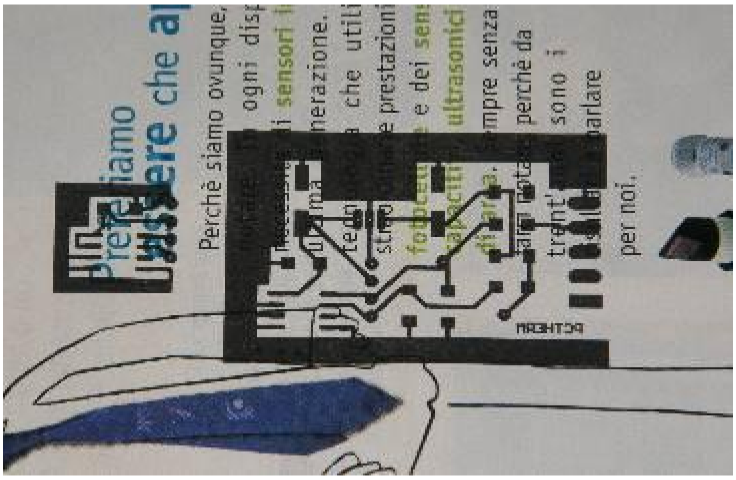
6.3.3 Printer setup

Laser printers are not designed for handling thin, cheap paper, so we help them feeding the sheets manually instead of using the paper tray. Selecting a straight paper path minimizes the chances of clogging. This is usually achieved setting the printer as if it we're printing on envelopes.

We need to put as much toner on paper as possible, so we disable "toner economy modes" and set printer properties to the maximum contrast and blackness possible. We need to print our PCB to exact size, so we were disable any form of scaling/resizing (e.g. "fit to page").

6.3.4 Printing

We Print our PCB layout as usual, except we must setup the printer as described above and we must need to print a mirrored layout.



This is our remote controller & temperature controller circuit printed on magazine paper. It was a mirror image of the circuit. Placing some text helps recognizing when the layout was mirrored. Text will read straight again once the image is transferred on copper.

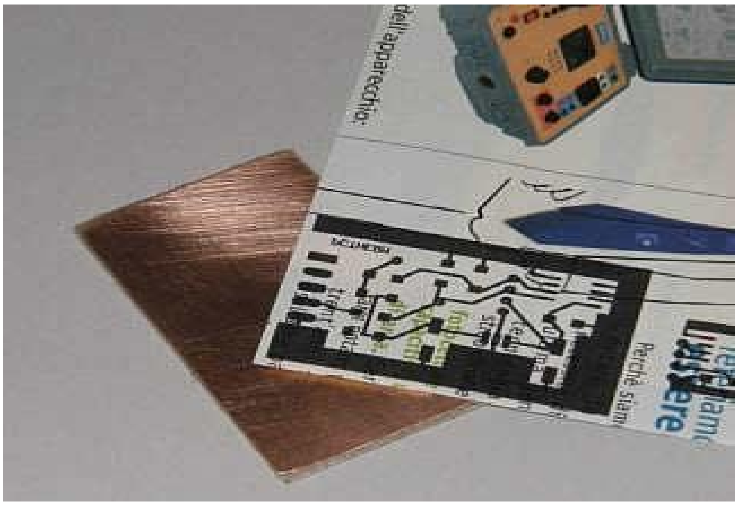
6.3.5 Resizing the raw CCB board

PCB material was copper laminated board. So we mark first board size. We score BOTH board sides with a blade cutter (be careful) or another sharp, hardened tool (e.g. a small screwdriver tip). Ensure to scratch edge-to-edge. Repeat this step 5-6 times on each side.

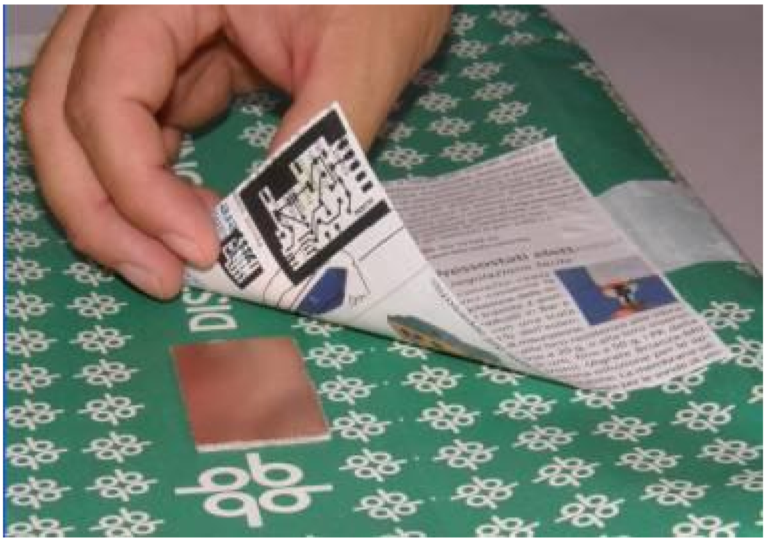
Then we bend the board. The board was beaked before reaching a 30 degrees bend. It will break quite abruptly so be prepared and protect wer hands with gloves.

To make paper alignment easier, cut a piece of PCB material that is larger (at least 10mm/0, 39 inch for each side) than the final PCB.

6.3.6 Preparing for transfer


dell'apparecchio:

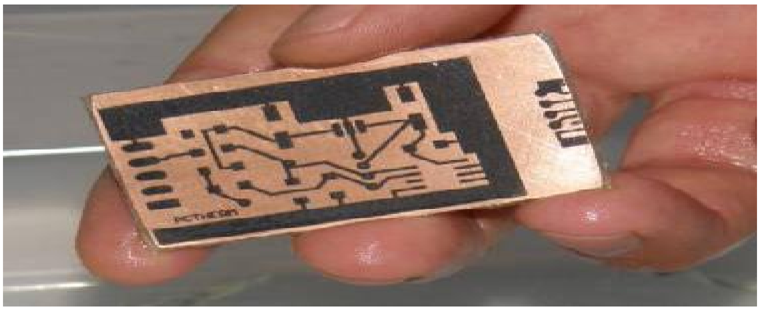
To make paper alignment easy, we cut excess paper around one corner (leave a small margin though). We leave plenty of paper on the other sides to fix the paper to the desk. As the board was larger than the final PCB, there is large margin for easy placement of paper on copper.



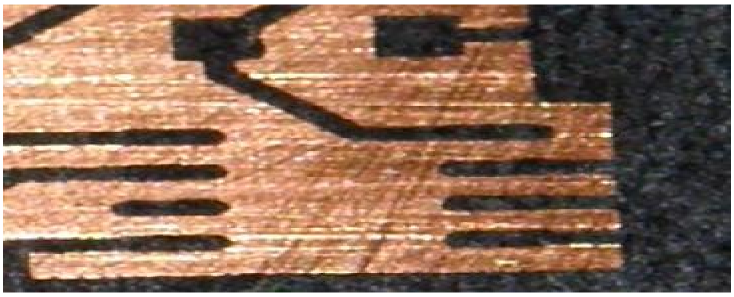
We gave the iron to its maximum heat (COTTON position). While the iron warms up, we position the materials on the table. We work on a table to ironing board as its soft surface makes it difficult to apply pressure and keep the PCB in place. Protect table surface with flat, heat-resistant material (e.g. old magazines) and place the board on top, copper face up. Lock the board in place with double-adhesive tape. Position the PCB printout over the copper surface, toner down, and align paper and board corners. Lock the paper with scotch tape along one side only. This way, we can flip the paper in and out instantly.

6.3.7 We Iron it!

Flip out the paper, and preheat copper surface placing the iron on top of it for 30 seconds. Remove the iron; flip back paper into its previous position over the copper. It is essential that paper does not slip from its position. We can also cover with a **second sheet** of blank paper to distribute pressure more evenly. Keep moving the iron, while pressing down as evenly as we can, for about one minute. Remove the iron and let the board to cool down.



The board with all paper removed. It is OK if some microscopic paper fibers remain on the toner (but remove any fiber from copper), giving it a silky feeling. It is normal that these fibers turn a little white when dry.

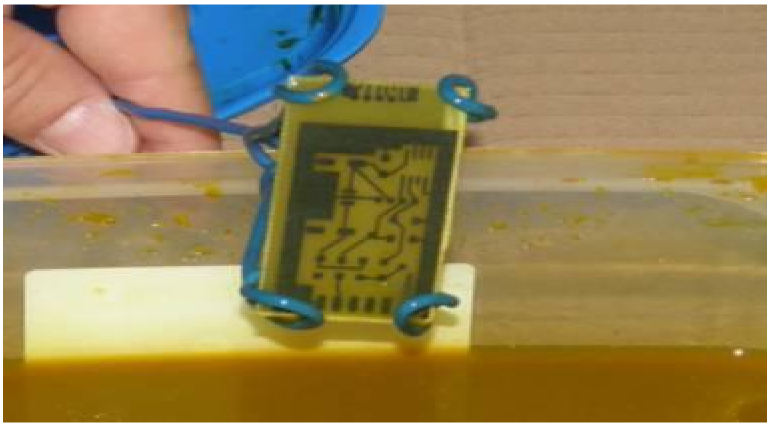


Magnified view of the tracks, these are 1206 pads and SO8 SMT pads, connected by 20 mils tracks. Some white fibers show up on the black toner surface.

6.3.8 Etching:

There are many alternatives for etching liquids, and we can use the one that. We use ferric chloride (the brown stuff): it's cheap, can be reused many times, and doesn't require heating. Actually, moderate heating can speed up etching, but I find it reasonably fast also at room temperature (10…15 minutes).

First impression may be that nothing happens, but in less than 10 minutes some copper is removed, making first tracks to appear. From now on, stir continuously and check often, as the process completes rather quickly. We don't want to overdo it, otherwise thinner tracks start being eroded sideways. As a rule of thumb, stop 30 seconds after we don't see any copper leftovers over large areas.



Rinse the board with plenty, plenty, plenty of water I store the etching solution in the same plastic box used for etching. When the job is done I just put the hermetic lid on. To further minimize risks of leakage, I put the container inside the bigger one I use for rinsing, put the second lid, and store it in a safe place.

6.3.9 Finishing touches

A few drops of thinner (nail polish remover works well) on a pinch of cotton wool will remove completely the toner, bringing back the copper surface. Rinse carefully and dry with a clean cloth or kitchen paper.
Trim to final size and refine edges with sandpaper.

3.4 Printed Circuit Board Design

The printed circuit board (PCB) were developed using the specialized software, proteus. The PCB diagrams are shown in following figure.

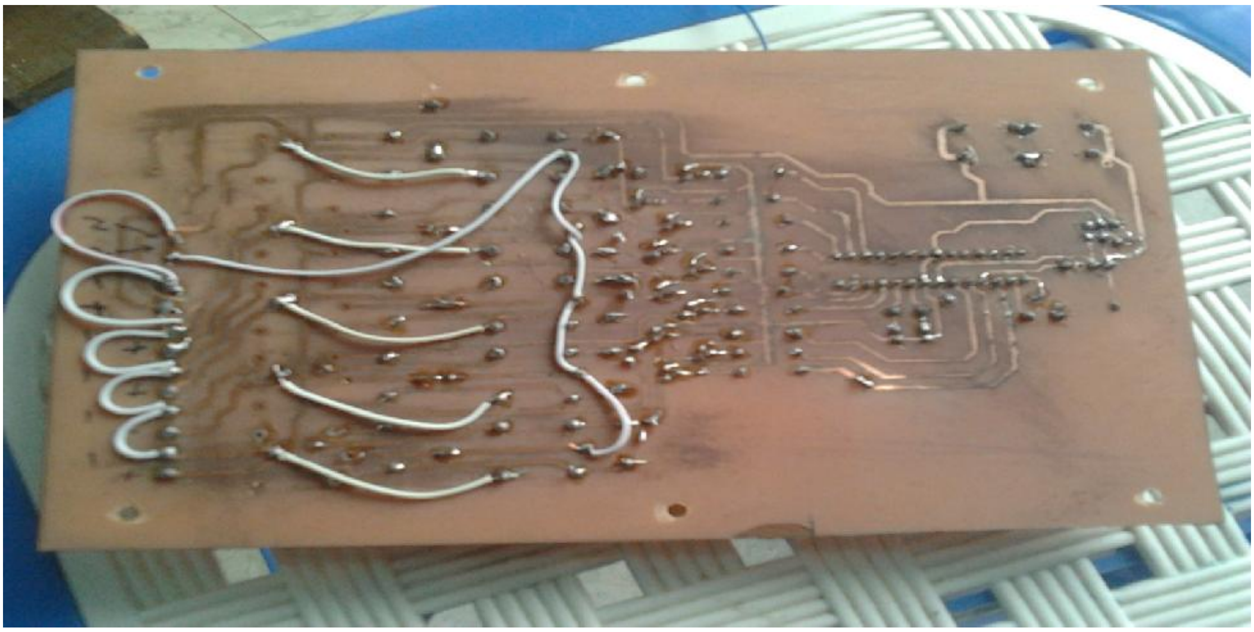


Figure: PCB design of full circuit.

CHAPTER – VII

PARTS LIST & COST CALCULATION

7.1 Parts list

Given here below detail uses parts for this project.

S/L No.	Parts List	Part No.	Quantity	Remark
1.	Microcontroller	Mega8	02	AVR µcontroller
2.	LCD Display	ML1640	01	16*2 line display
3.	Temperature Sensor	DS18b20	01	Temperature Sensor
4.	Transistor	BC548B	15	Bipolar transistor for relay driver
5.	Voltage Regulator	7805CV	02	DC 5V regulator IC
6.	Crystal	16Mhz	02	16Mhz clock pulse generator
7.	IR Sensor	TSOP1738	01	38 KHz remote signal decoder
8.	Diode	1N4007	20	Ordinary general purpose diode
9.	Relay	12vdc	10	12V Dc ordinary relay
10.	Resistors		30	Ordinary general purpose resistor
11.	Capacitor	1000µF	02	Filter Capacitor
12.	CCB Board	12 sq inch	01	Laminated Copper board
13.	Connectors		5	
14.	Wire		10feet	
15.	Transformer		01	220V/12V step down transformer

Table-7.1: Part list of our product

7.2 Cost of our Product

S/L No.	Parts List	Part No.	Quantity	Price (BDT)	Total Amount (BDT)
01	Microcontroller	Mega8	02	90*2	180
02	LCD Display	ML1640	01	250	250
03	Temperature Sensor	DS18b20	01	180	180
04	Transistor	BC548B	15	15*2=30	30
05	Voltage Regulator	7805CV	02	10*2=20	20
06	Crystal	16Mhz	02	10*2=20	20
07	IR Sensor	TSOP1738	01	15	15
08	Diode	1N4007	20	20*2=40	40
09	Relay	12vdc	10	20*10=200	200
10	Resistors		30	30*1=30	30
11	Capacitor	1000µF	02	15*2=30	30
12	CCB Board	12 sq inch	01	1*200=200	200
13	Connectors		5	15	75
14	Wire		10feet	10	100
15	Transformer		01	75	75
				Others:	500
				Ground Total:	1945

Table-7.2: Total Cost Calculation

CHAPTER – VIII

COMPLETE IMPLEMENTED CIRCUIT

8.1 Circuit on PCB

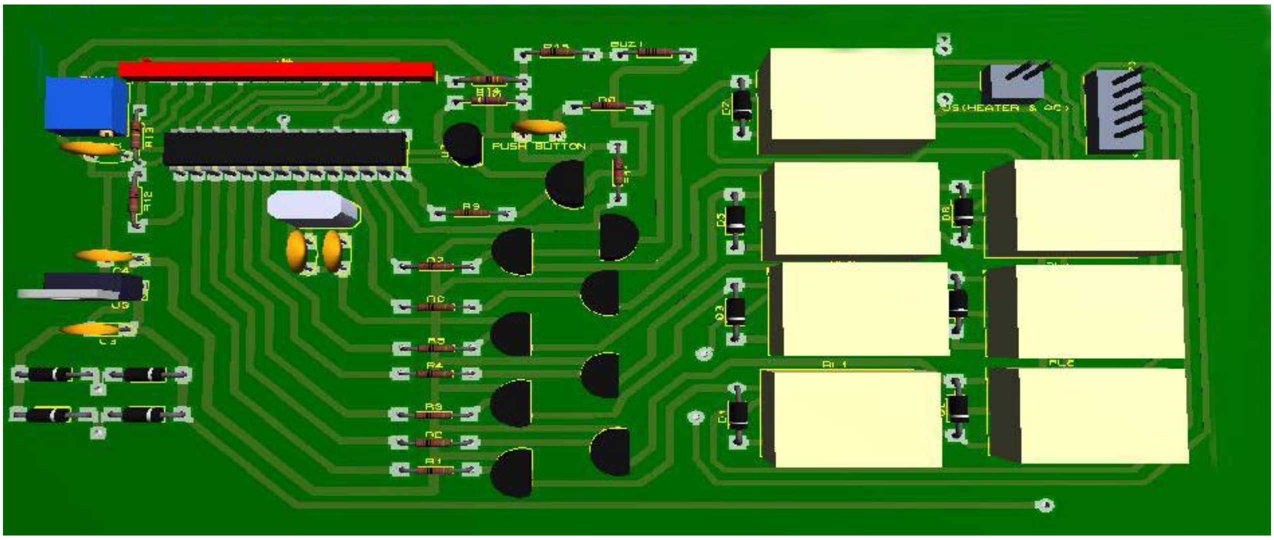


Figure-8.1: Final product of our project-1

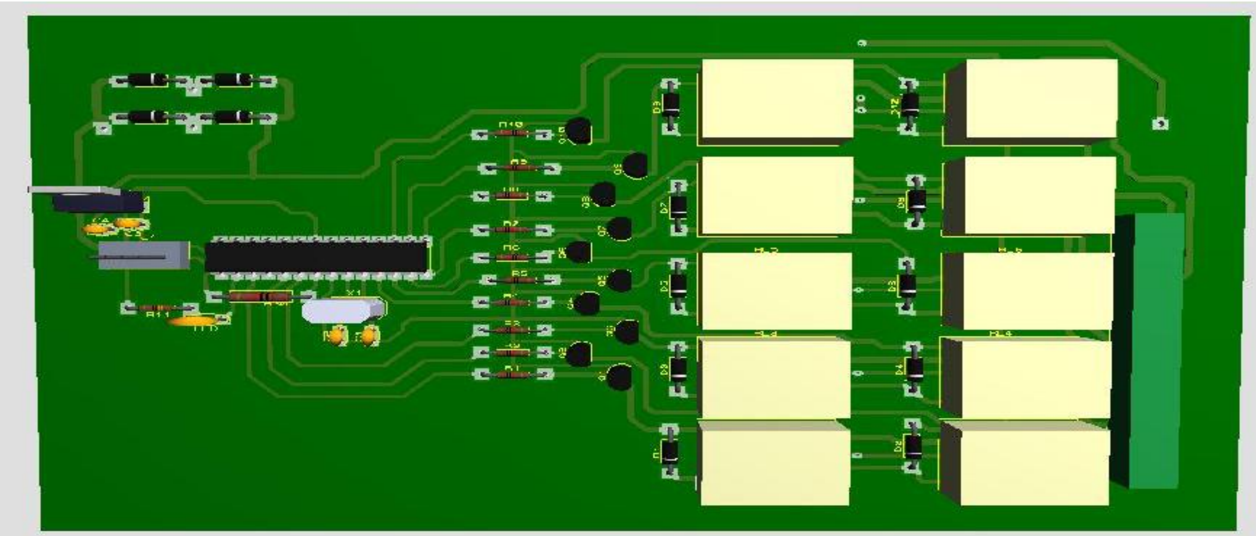


Figure-8.2: Final product of our project-2

8.2 PRACTICAL CIRCUIT OF THE THESIS:

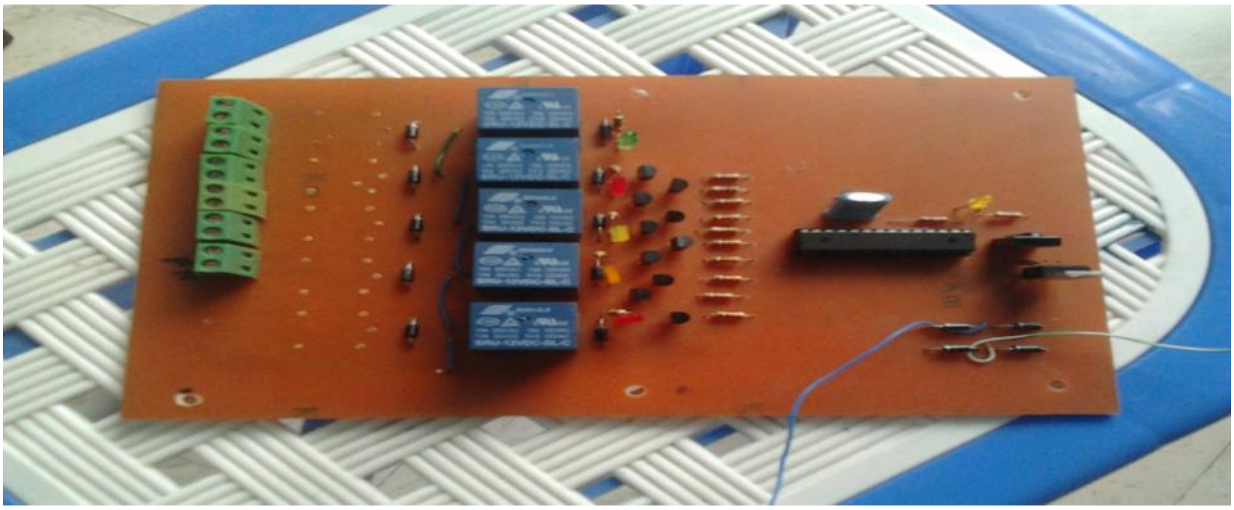


Figure-8.3: Remote control Circuit

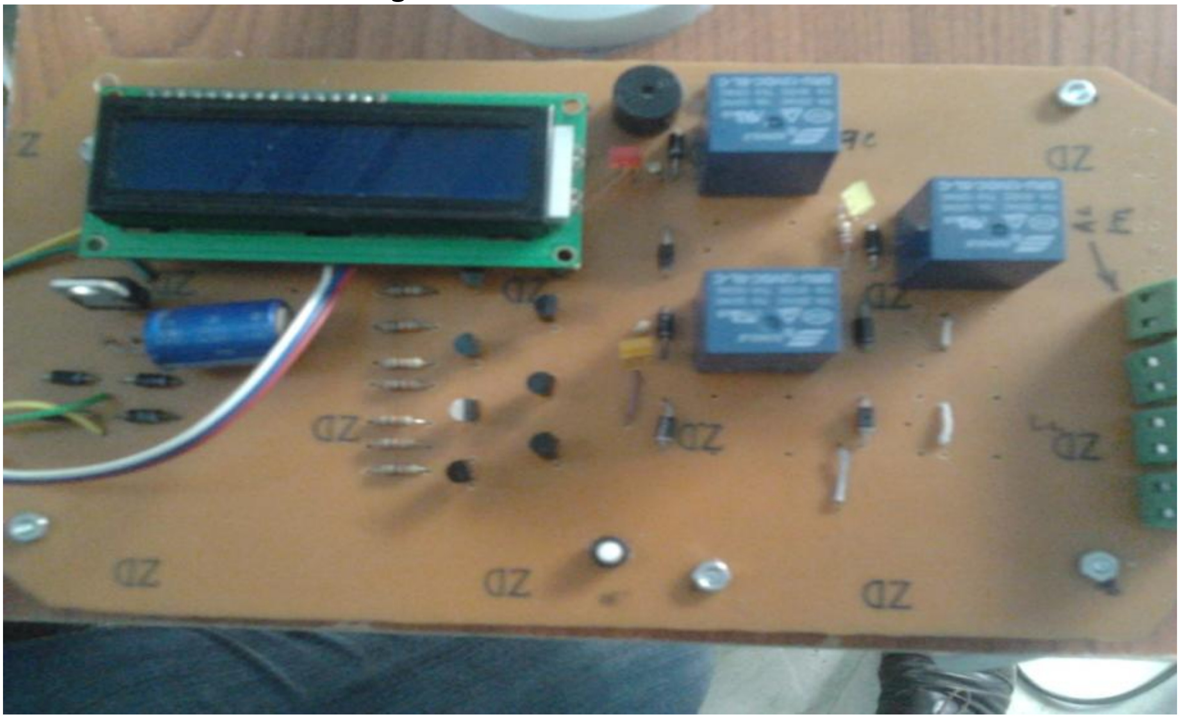


Figure-8.4: Temperature controller, water level alarm & Day light sensor Circuit

Figure-8.5: Overall Circuit Of This Thesis

CHAPTER – IX

CONCLUSION & FUTURE DEVELOPMENT

9.1 Conclusion

The final device managed to achieve the following results:

How we can load on off automatically depends on remote control we learnt through this project. We also learnt about microcontroller through this project. We learnt how program write and program load in the programmable device. Beside this we also gain concept of microcontroller related software & device. In this project by applying sensor and transistor we achieved practical knowledge about that.

The device is controlled by an Atmel Atmega8L which is fast enough and has enough memory to carry out all required tasks.

The entire design has a very efficiency for sensing temperature, light & water.

It can be seen from the above points, that the device has fulfilled the requirements and specifications that was placed upon it and has also gone some way in having additional features that would serve it as an excellent device.

This device demonstrates that engineering is an interesting and exciting field which can help to benefit countless people in other parts of the world.

9.2 Future Developments and Improvements

There are a number of ways in which this device could be improved in any future development.

In future we want to add humidity sensor to measure the humidity at room.

Humidity sensor needed to control the humidity at room.

Temperature sensor needed to control the temperature.

More efficient software would also benefit the device and the means of upgrading the software on the MCU has been catered for our project.

Using Triac instead relay that can be provided low power consume and given the more efficiency. Reducing the power losses in the system would increase the efficiency of the device .In future we use 4*3 keypad for password based door lock protection.

One last major improvement would be to use as many surface mounted components as possible, particularly using a surface mounted MCU and resistors and as many other components that could be replaced with surface mounted equivalents. Future we will design a better & compact PCB and good quality circuit by using surface mounted component.