SMART HOME PROJECT

By:

Alaa sherief Hamza

Nouran ayman mosbah

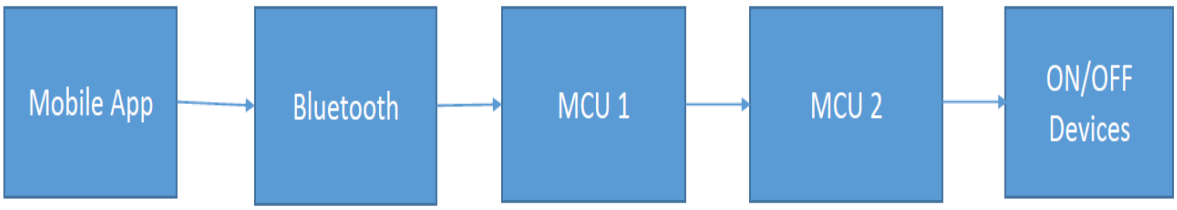
AMIT Instructor:
Eng.Karim El-nahal

Contents

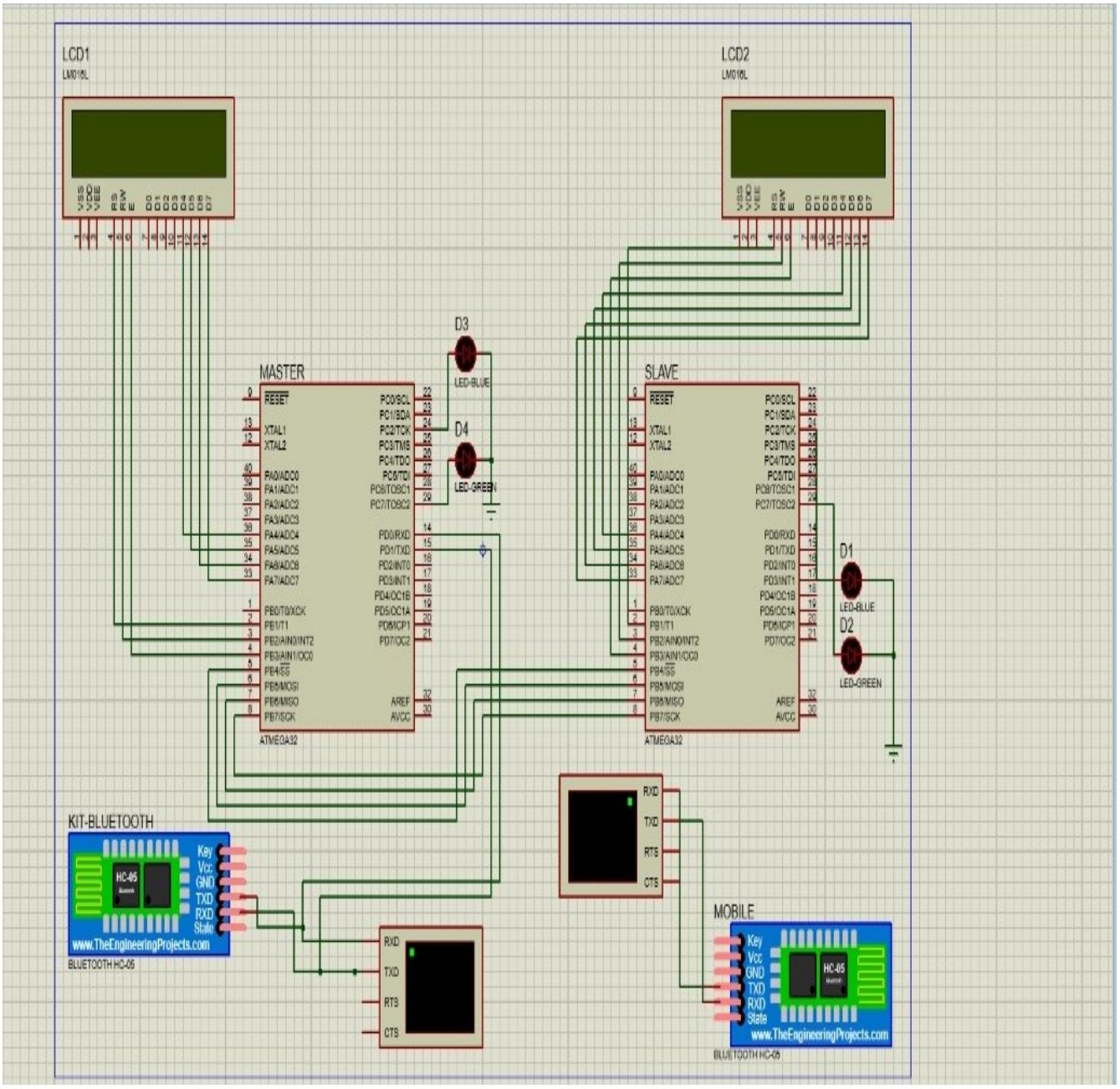
1 Abstract:

•This project is Smart Home based Bluetooth where we want to control home appliance wirelessly using Mobile App via Bluetooth.

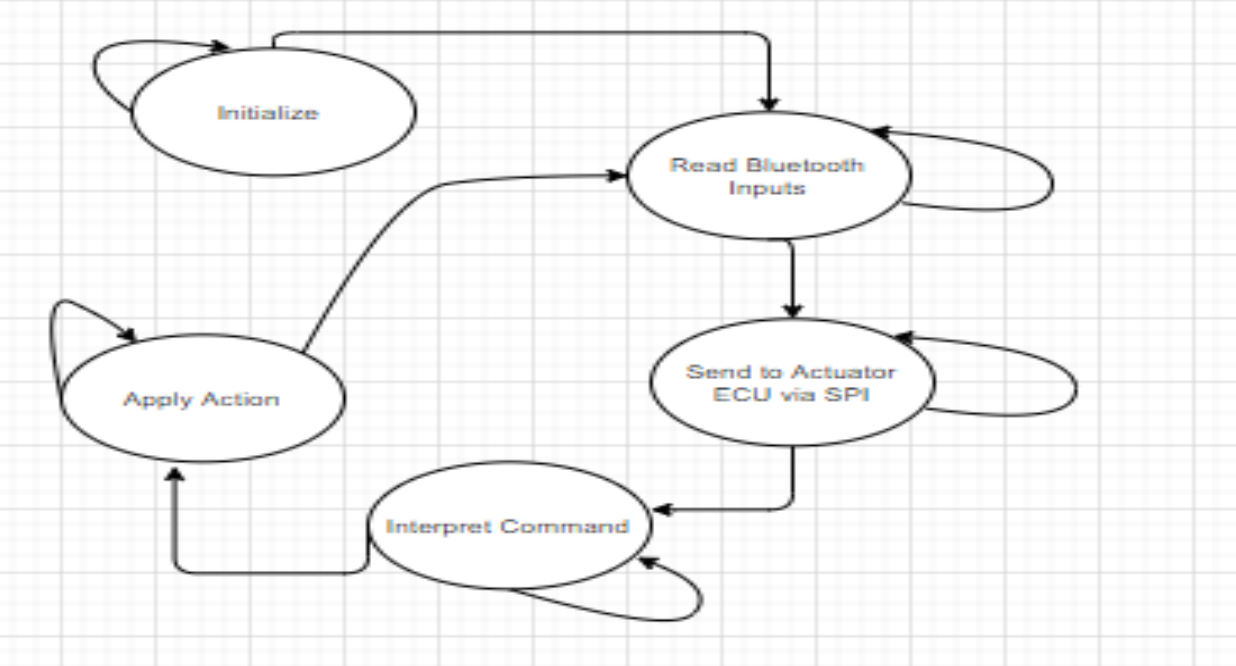
•Two ECU's Communicate with each other the first is a control ECU which takes the input from Bluetooth and send it to the Sink (Actuator) ECU via SPI to interpret which action should be taken



2 Schematic On Proteus:



3 Finite State Machine of the System:

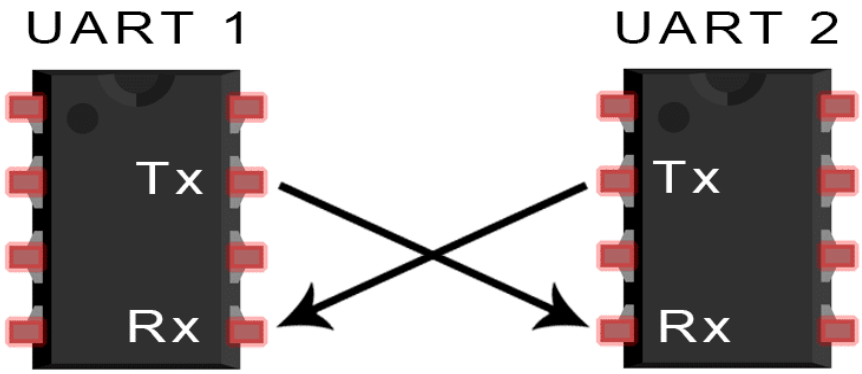


4 Describtion the Architecture and Design:

4.1 UART:

4.1.1 INTRODUCTION TO UART COMMUNICATION:

In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. Only two wires are needed to transmit data between two UARTs. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART:



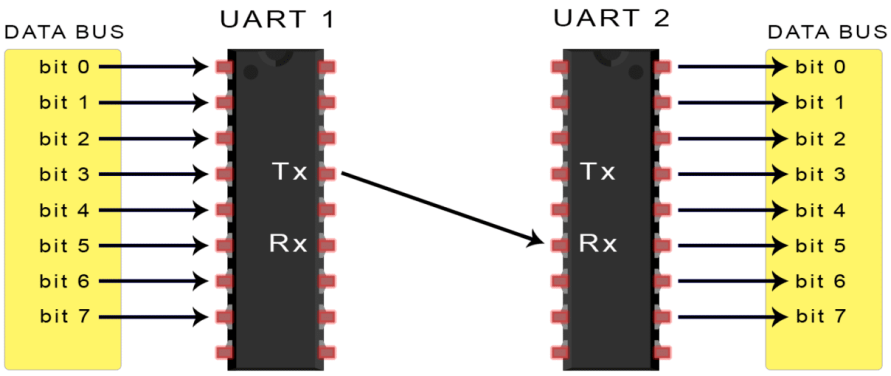
UARTs transmit data *asynchronously*, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits.

When the receiving UART detects a start bit, it starts to read the incoming bits at a specific frequency known as the *baud rate.* Baud rate is a measure of the speed of data transfer, expressed in bits per second (bps). Both UARTs must operate at about

the same baud rate. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off.

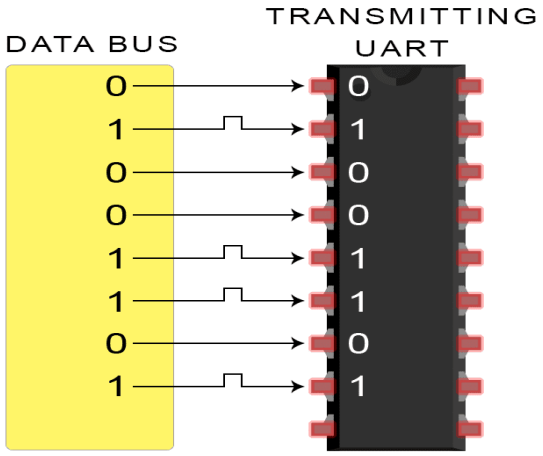
4.1.2 HOW UART WORKS:

The UART that is going to transmit data receives the data from a data bus. The data bus is used to send data to the UART by another device like a CPU, memory, or microcontroller. Data is transferred from the data bus to the transmitting UART in parallel form. After the transmitting UART gets the parallel data from the data bus, it adds a start bit, a parity bit, and a stop bit, creating the data packet. Next, the data packet is output serially, bit by bit at the Tx pin. The receiving UART reads the data packet bit by bit at its Rx pin. The receiving UART then converts the data back into parallel form and removes the start bit, parity bit, and stop bits. Finally, the receiving UART transfers the data packet in parallel to the data bus on the receiving end:

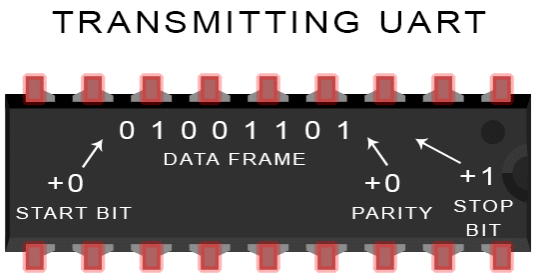


4.1.3 STEPS OF UART TRANSMISSION:

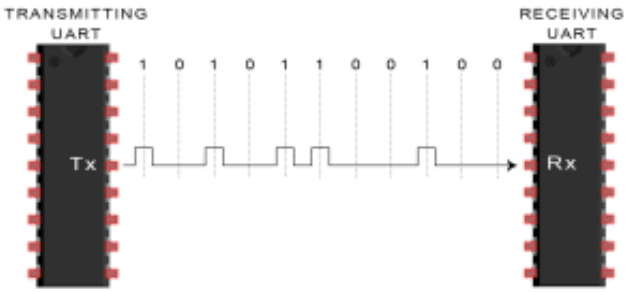
The transmitting UART receives data in parallel from the data bus:



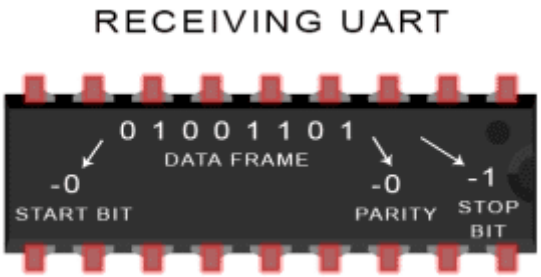
The transmitting UART adds the start bit, parity bit, and the stop bit(s) to the data frame:



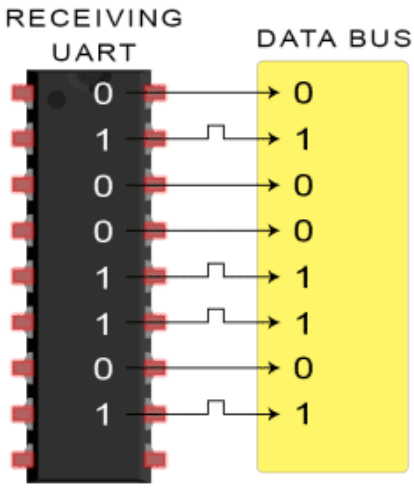
The entire packet is sent serially from the transmitting UART to the receiving UART. The receiving UART samples the data line at the pre-configured baud rate:



The receiving UART discards the start bit, parity bit, and stop bit from the data frame:



The receiving UART converts the serial data back into parallel and transfers it to the data bus on the receiving end:



4.1.4 ADVANTAGES:

Only uses two wires

No clock signal is necessary

Has a parity bit to allow for error checking

The structure of the data packet can be changed as long as both sides are set up for it

Well documented and widely used method

4.1.5 DISADVANTAGES:

The size of the data frame is limited to a maximum of 9 bits

Doesn't support multiple slave or multiple master systems

The baud rates of each UART must be within 10% of each other

4.2 SPI:

4.2.1 Introduction:

Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device you wish to talk to.

Programming for SPI:

Many microcontrollers have built-in SPI peripherals that handle all the details of sending and receiving data, and can do so at very high speeds. The SPI protocol is also simple enough that you (yes, you!) can write your own routines to manipulate the I/O lines in the proper sequence to transfer data. (A good example is on the Wikipedia SPI page.)

If you're using an Arduino, there are two ways you can communicate with SPI devices:

You can use the shift In () and shift Out () commands. These are software-based commands that will work on any group of pins, but will be somewhat slow.

Or you can use the SPI Library, which takes advantage of the SPI hardware built into the microcontroller. This is vastly faster than the above commands, but it will only work on certain pins.

You will need to select some options when setting up your interface. These options must match those of the device you're talking to; check the device's datasheet to see what it requires.

The interface can send data with the most-significant bit (MSB) first, or least-significant bit (LSB) first. In the Arduino SPI library, this is controlled by the set Bit Order () function.

The peripheral will read the data on either the rising edge or the falling edge of the clock pulse. Additionally, the clock can be considered "idle" when it is high or low. In the Arduino SPI library, both of these options are controlled by the set Data Mode () function.

SPI can operate at extremely high speeds (millions of bytes per second), which may be too fast for some devices. To accommodate such devices, you can adjust the data rate. In the Arduino SPI library, the speed is set by the set Clock Divider ( ) function, which divides the controller clock (16MHz on most Arduinos) down to a frequency between 8MHz (/2) and 125kHz (/128).

If you're using the SPI Library, you must use the provided SCK, COPI and CIPO pins, as the hardware is hardwired to those pins. There is also a dedicated CS pin that you can use (which must, at least, be set to an output in order for the SPI hardware to function), but note that you can use any other available output pin(s) for CS to your peripheral device(s) as well.

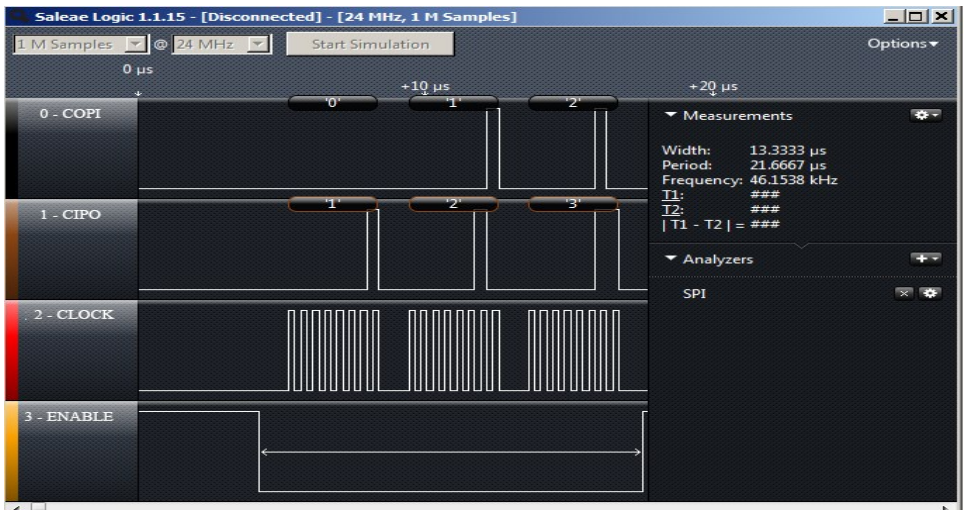
On older Arduinos, you'll need to control the CS pin(s) yourself, making one of them low before your data transfer and high afterward. Newer Arduinos such as the Due can control each CS pin automatically as part of the data transfer; see the Due SPI documentation page for more information.

4.2.2 Resources and Going Further:

Tips and Tricks:

Because of the high speed signals, SPI should only be used to send data over short distances (up to a few feet). If you need to send data further than that, lower the clock speed, and consider using specialized driver chips.

If things aren't working the way you think they should, a logic analyzer is a very helpful tool. Smart analyzers like the Saleae USB Logic Analyzer can even decode the data bytes for a display or logging.



4.2.3 Advantages of SPI:

It's faster than asynchronous serial

The receive hardware can be a simple shift register

It supports multiple peripherals

4.2.4 Disadvantages of SPI:

It requires more signal lines (wires) than other communications methods

The communications must be well-defined in advance (you can't send random amounts of data whenever you want)

The controller must control all communications (peripherals can't talk directly to each other)

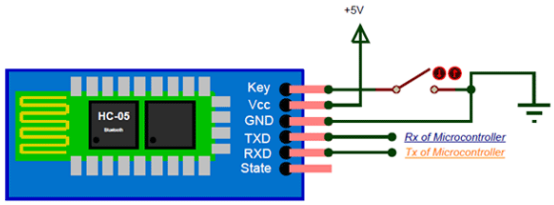
It usually requires separate CS lines to each peripheral, which can be problematic if numerous peripherals are needed.

5 HC-05 Bluetooth

5.1 How to Use the HC-05 Bluetooth module:

The HC-05 has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default device settings can be changed. We can operate the device in either of these two modes by using the key pin as explained in the pin description.

It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU as shown in the figure below.



During power up the key pin can be grounded to enter into Command mode, if left free it will by default enter into the data mode. As soon as the module is powered you should be able to discover the Bluetooth device as “HC-05” then connect with it using the default password 1234 and start communicating with it. The name password and other default parameters can be changed by entering into the

5.2 Applications:

1. Wireless communication between two microcontrollers

2. Communicate with Laptop, Desktops and mobile phones

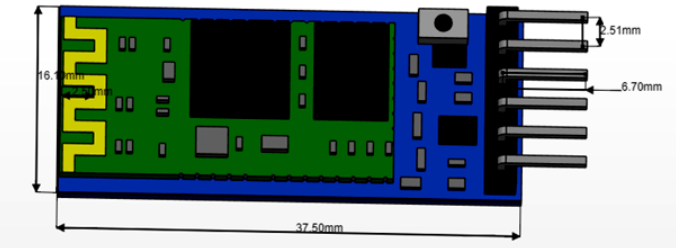
3. Data Logging application

4. Consumer applications

5. Wireless Robots

6. Home Automation

2D Model:



6 LCD:

6.1 Introduction:

LCD (Liquid Crystal Display) is a type of flat panel display which uses liquid crystals in its primary form of operation. LEDs have a large and varying set of use cases for consumers and businesses, as they can be commonly found in smartphones, televisions, computer monitors and instrument panels.

LCDs were a big leap in terms of the technology they replaced, which include light-emitting diode (LED) and gas-plasma displays. LCDs allowed displays to be much thinner than cathode ray tube (CRT) technology. LCDs consume much less power than LED and gas-display displays because they work on the principle of blocking

light rather than emitting it. Where an LED emits light, the liquid crystals in an LCD produces an image using a backlight.

As LCDs have replaced older display technologies, LCDs have begun being replaced by new display technologies such as OLEDs.

6.2 How LCDs work:

A display is made up of millions of pixels. The quality of a display commonly refers to the number of pixels; for example, a 4K display is made up of 3840 x2160 or 4096x2160 pixels. A pixel is made up of three subpixels; a red, blue and green—commonly called RGB. When the subpixels in a pixel change color combinations, a different color can be produced. With all the pixels on a display working together, the display can make millions of different colors. When the pixels are rapidly switched on and off, a picture is created.

The way a pixel is controlled is different in each type of display; CRT, LED, LCD and newer types of displays all control pixels differently. In short, LCDs are lit by a backlight, and pixels are switched on and off electronically while using liquid crystals to rotate polarized light. A polarizing glass filter is placed in front and behind all the pixels, the front filter is placed at 90 degrees. In between both filters are the liquid crystals, which can be electronically switched on and off.

LCDs are made with either a passive matrix or an active matrix display grid. The active matrix LCD is also known as a thin film transistor (TFT) display. The passive matrix LCD has a grid of conductors with pixels located at each intersection in the grid. A current is sent across two conductors on the grid to control the light for any pixel. An active matrix has a transistor located at each pixel intersection, requiring less current to control the luminance of a pixel. For this reason, the current

in an active matrix display can be switched on and off more frequently, improving the screen refresh time.

Some passive matrix LCD's have dual scanning, meaning that they scan the grid twice with current in the same time that it took for one scan in the original technology. However, active matrix is still a superior technology out of the two.

6.3 LEDs:

Introduction:

In the simplest terms, a light-emitting diode (LED) is a semiconductor device that emits light when an electric current is passed through it. Light is produced when the particles that carry the current (known as electrons and holes) combine together within the semiconductor material.

Since light is generated within the solid semiconductor material, LEDs are described as solid-state devices. The term solid-state lighting, which also encompasses organic LEDs (OLEDs), distinguishes this lighting technology from other sources that use heated filaments (incandescent and tungsten halogen lamps) or gas discharge (fluorescent lamps).

6.4 Different colors:

Inside the semiconductor material of the LED, the electrons and holes are contained within energy bands. The separation of the bands (i.e. the bandgap) determines the energy of the photons (light particles) that are emitted by the LED.

The photon energy determines the wavelength of the emitted light, and hence its color. Different semiconductor materials with different bandgaps produce different

colors of light. The precise wavelength (color) can be tuned by altering the composition of the light-emitting, or active, region.

LEDs are comprised of compound semiconductor materials, which are made up of elements from group III and group V of the periodic table (these are known as III-V materials). Examples of III-V materials commonly used to make LEDs are gallium arsenide (GaAs) and gallium phosphide (GaP).

Until the mid-90s LEDs had a limited range of colors, and in particular commercial blue and white LEDs did not exist. The development of LEDs based on the gallium nitride (GaN) material system completed the palette of colors and opened up many new applications.

6.5 Main LED materials:

The main semiconductor materials used to manufacture LEDs are:

Indium gallium nitride (InGaN): blue, green and ultraviolet high-brightness LEDs

Aluminum gallium indium phosphide (AlGaInP): yellow, orange and red high-brightness LEDs

Aluminum gallium arsenide (AlGaAs): red and infrared LEDs

Gallium phosphide (GaP): yellow and green LEDs

7 Atmega32

7.1 Features

- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution

  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash program memory
  - 1024Bytes EEPROM
  - 2Kbytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[(1)]
  - 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
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture
  - Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels

  - 8-channel, 10-bit ADC
  - 8 Single-ended Channels
  - 7 Differential Channels in TQFP Package Only
  - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - 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
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby
  - and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega32L
  - 4.5V - 5.5V for ATmega32
- Speed Grades
  - 0 - 8MHz for ATmega32L
  - 0 - 16MHz for ATmega32
- Power Consumption at 1MHz, 3V, 25°C
  - Active: 1.1mA

- Idle Mode: 0.35mA
- Power-down Mode: < 1μA

7.2 How to Use ATMEGA32

Using ATMEGA32 is similar to any other microcontroller. Similar to them it is not Plug and Play digital ICs. For working of ATMEGA32, first we need to save the appropriate program file in the ATMEGA32 FLASH memory. After dumping this program code, the controller executes this code to create the response.

7.3 Applications:

There are thousands of applications for ATMEGA32.

Temperature control systems

Analog signal measuring and manipulations.

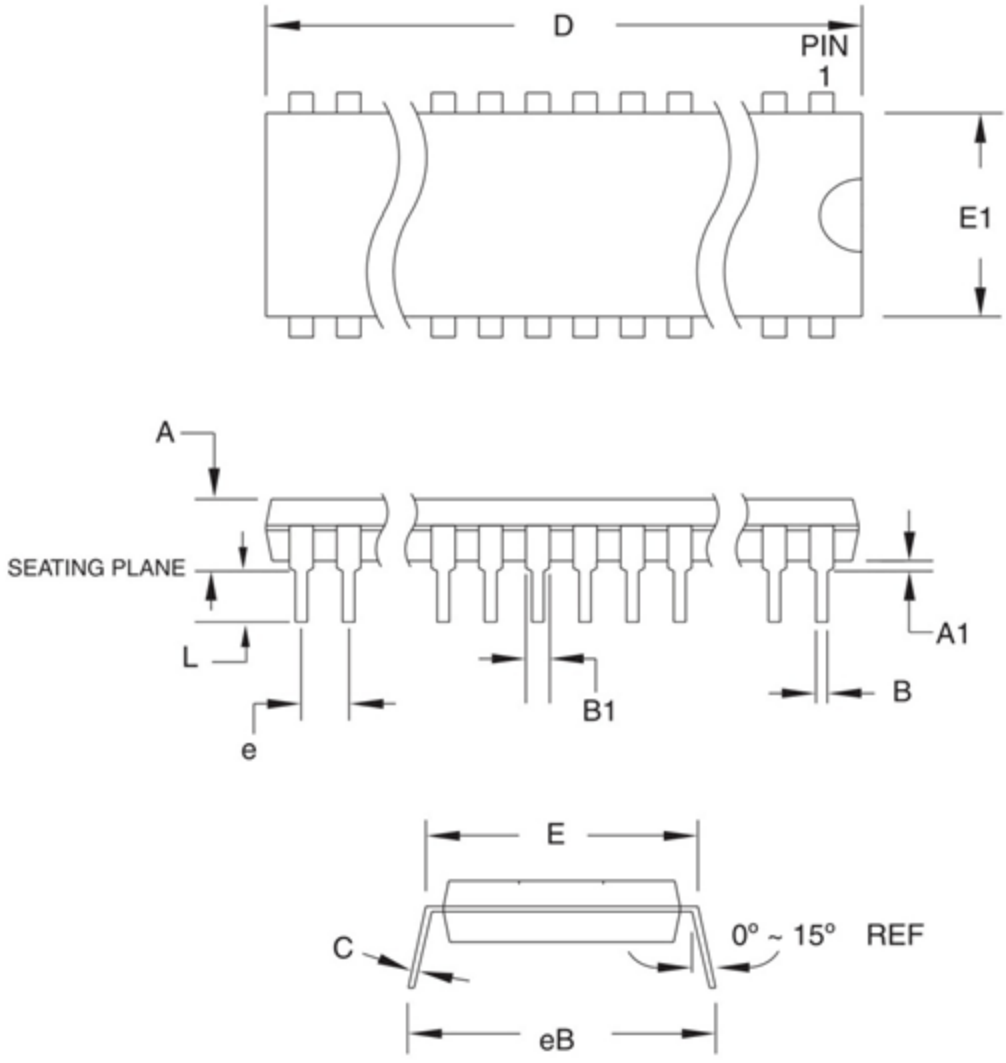
Embedded systems like coffee machine, vending machine.

Motor control systems.

Digital signal processing.

Peripheral Interface system.

2D Model


D
PIN
1
E1
A
SEATING PLANE
L
e
B1
B
A1
E
C
0° ~ 15° REF
eB