A Real-Time (or) Field-based Research Project Report

On

**AUTONOMOUS ROBOT FOR REFINERY INSPECTION**

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

C.B. DIVYA (227R1A05D6)

E.VIVEK (227R1A05E0)

MOHAMMAD SUBHAN (227R1A05G3)

Under the Guidance of

**Ms.Bhagyasree**

(Assistant Professor)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### CMR TECHNICAL CAMPUS

#### UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V),

Medchal Road, Hyderabad-501401.

##### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



## CERTIFICATE

This is to certify that the project entitled **“AUTONOMOUS ROBOT FOR REFINERY INSPECTION”** being submitted by C.**B.DIVYA(227R1A05D6),E.VIVEK (227R1A05E0), MOHAMMAD SUBHAN(227R1A05G3)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the CMR Technical Campus, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-2024.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

**Dr. K. SRUJAN RAJU**

|  |  |
| --- | --- |
| **(Professor)** | **Dr. A. Raji Reddy** |
| INTERNAL GUIDE | DIRECTOR |
| **Dr. K. Srujan Raju**  HOD |  |
| **Submitted for viva voice Examination held on\_\_\_\_\_\_\_** |  |

**ABSTRACT**

Industrial safety is one of the main aspects of industry specially refining industry. To avoid any

types of unwanted phenomena all refining industry follows some basic precaution and phenomena.

Communication is the main key factor for any industry today to monitor different parameters and take necessary actions accordingly to avoid any types of hazards.To implement a robotic system to

autonomously navigate in an oil and gas refinery and it must be able to communicate with the control room and also localize it and alert workers in hazardous leakages and other accidents. Oil and gas refineries can be a dangerous environment for numerous reasons, including heat, gasses and humidity at the refinary. In order to augment how human operators interact with this environment, a mobile robotic platform is developed. This paper focuses on the use of WiFi for communicating with and localizing the robot. All the algorithms implemented are tested in real world scenarios with the robot developed and results are promising.

**TABLE OF CONTENTS**

**CHAPTER 1 : INTRODUCTION 1-7**

1. 1 introduction

1.2 classification

1.3 other common parts found on many embedded systems

1.3.1 specification

1.3.2 system-synthesis

1.3.3 implementation-synthesis

1.3.4 prototyping

1.3.5 applications

**CHAPTER 2 : LITERATURE SURVEY 8-9**

**CHAPTER 3 : THEORY 10-16**

3.1 Embedded systems

3.1.1 Introduction to Embedded systems

3.1.2 Classification

3.1.3 Other common parts found on many Embedded systems

3.1.4 Design process

3.2 Arduino

**CHAPTER 4: ANALYSIS AND DESIGN 17-27**

4.1 Hardware Components

4.2 Block Diagram

4.3 Software Description

**CHAPTER 5 : IMPLEMENTATION 28-34**

**CHAPTER 6 : TESTING AND DEBUGGING/ RESULT 35**

**CHAPTER 7 : CONCLUSION 36**

**CHAPTER 8 : REFERENCE 37**

**CHAPTER 9 : APPENDICES 38**

**CHAPTER-1**

**INTRODUCTION TO EMBEDDED SYSTEMS**

Many embedded systems have substantially different design constraints than desktop computing applications. No single characterization applies to the diverse spectrum of embedded systems. However, some combination of cost pressure, long life-cycle, real¬-time requirements, reliability requirements, and design culture dysfunction can make it difficult to be successful applying traditional computer design methodologies and tools to embedded applications. Embedded systems in many cases must be optimized for life-cycle and business-driven factors rather than for maximum computing throughput. There is currently little tool support for expanding embedded computer design to the scope of holistic embedded system design. However, knowing the strengths and weaknesses of current approaches can set expectations appropriately, identify risk areas to tool adopters, and suggest ways in which tool builders can meet industrial needs. If we look around us, today we see numerous appliances which we use daily, be it our refrigerator, the microwave oven, cars, PDAs etc. Most appliances today are powered by something beneath the sheath that makes them do what they do. These are tiny microprocessors, which respond to various keystrokes or inputs. These tiny microprocessors, working on basic assembly languages, are the heart of the appliances. We call them embedded systems. Of all the semiconductor industries, the embedded systems market place is the most conservative, and engineering decisions here usually lean towards established, low risk solutions. Welcome to the world of embedded systems, of computers that will not look like computers and won’t function like anything we are familiar with.

* 1. **CLASSIFICATION**

Embedded systems are divided into autonomous, realtime, networked & mobile categoriesAutonomous systems

They function in standalone mode. Many embedded systems used for process control in manufacturing units& automobiles fall under this category.

**Real-time embedded systems**

These are required to carry out specific tasks in a specified amount of time. These systems are extensively used to carry out time critical tasks in process control.

**Networked embedded systems**

They monitor plant parameters such as temperature, pressure and humidity and send the data over the network to a centralized system for on line monitoring.

**Mobile gadgets**

Mobile gadgets need to store databases locally in their memory. These gadgets imbibe powerful computing & communication capabilities to perform realtime as well as nonrealtime tasks and handle multimedia applications. The embedded system is a combination of computer hardware, software, firmware and perhaps additional mechanical parts, designed to perform a specific function. A good example is an automatic washing machine or a microwave oven. Such a system is in direct contrast to a personal computer, which is not designed to do only a specific task. But an embedded system is designed to do a specific task with in a given timeframe, repeatedly, endlessly, with or without human interaction.

**Hardware**

Good software design in embedded systems stems from a good understanding of the hardware behind it. All embedded systems need a microprocessor, and the kinds of microprocessors used in them are quite varied. A list of some of the common microprocessors families are: ARM family, The Zilog Z8 family, Intel 8051/X86 family, Motorola 68K family and the power PC family. For processing of information and execution of programs, embedded system incorporates microprocessor or micro- controller. In an embedded system the microprocessor is a part of final product and is not available for reprogramming to the end user. An embedded system also needs memory for two purposes, to store its program and to store its data. Unlike normal desktops in which data and programs are stored at the same place, embedded systems store data and programs in different memories. This is simply because the embedded system does not have a hard drive and the program must be stored in memory even when the power is turned off. This type of memory is called ROM. Embedded applications commonly employ a special type of ROM that can be programmed or reprogrammed with the help of special devices.

**1.2 OTHER COMMON PARTS FOUND ON MANY EMBEDDED SYSTEMS**

• UART& RS232

• PLD

• ASIC’s& FPGA’s

• Watch dog timer etc.

**1.3 DESIGN PROCESS**

Embedded system design is a quantitative job. The pillars of the system design methodology are the separation between function and architecture, is an essential step from conception to implementation. In recent past, the search and industrial community has paid significant attention to the topic of hardware-software (HW/SW) codesign and has tackled the problem of coordinating the design of the parts to be implemented as software and the parts to be implemented as hardware avoiding the HW/SW integration problem marred the electronics system industry so long. In any large scale embedded systems design methodology, concurrency must be considered as a first class citizen at all levels of abstraction and in both hardware and software. Formal models & transformations in system design are used so that verification and synthesis can be applied to advantage in the design methodology. Simulation tools are used for exploring the design space for validating the functional and timing behaviors of embedded systems. Hardware can be simulated at different levels such as electrical circuits, logic gates, RTL e.t.c. using VHDL description. In some environments software development tools can be coupled with hardware simulators, while in others the software is executed on the simulated hardware. The later approach is feasible only for small parts of embedded systems. Design of an embedded system using Intel’s 80C188EB chip is shown in the figure. Inorder to reduce complexity, the design process is divided in four major steps: specification, system synthesis, implementation synthesis and performance evaluation of the prototype.

**1.3.1 SPECIFICATION**

During this part of the design process, the informal requirements of the analysis are transformed to formal specification using SDL.

**1.3.2 SYSTEM-SYNTHESIS**

For performing an automatic HW/SW partitioning, the system synthesis step translates the SDL specification to an internal system model switch contains problem graph& architecture graph. After system synthesis, the resulting system model is translated back to SDL.

**1.3.3 IMPLEMENTATION-SYNTHESIS**

SDL specification is then translated into conventional implementation languages such as VHDL for hardware modules and C for software parts of the system.

**1.3.4 PROTOTYPING**

On a prototyping platform, the implementation of the system under development is executed with the software parts running on multiprocessor unit and the hardware part running on a FPGA board known as phoenix, prototype hardware for Embedded Network Interconnect Accelerators.

**1.3.5 APPLICATIONS**

Embedded systems are finding their way into robotic toys and electronic pets, intelligent cars and remote controllable home appliances. All the major toy makers across the world have been coming out with advanced interactive toys that can become our friends for life. ‘Furby’ and ‘AIBO’ are good examples at this kind. Furbies have a distinct life cycle just like human beings, starting from being a baby and growing to an adult one. In AIBO first two letters stands for Artificial Intelligence. Next two letters represents robot. The AIBO is robotic dog. Embedded systems in cars also known as Telematic Systems are used to provide navigational security communication & entertinment services using GPS, satellite. Home appliances are going the embedded way. LG electronics digital DIOS refrigerator can be used for surfing the net, checking e-mail, making video phone calls and watching TV.IBM is developing an air conditioner that we can control over the net. Embedded systems cover such a broad range of products that generalization is difficult. Here are some broad categories.

Requirement analysis

Specification

System

architecture

H/w design

S/w design

S/w  
implementation

H/w testing

S/w testing

System integration

H/w implementation

System validation

Operation

Maintenance Evolution

Fig 1.1: Embedded Development Life Cycle

• Aerospace and defence electronics: Fire control, radar, robotics/sensors, sonar.

• Automotive: Autobody electronics, auto power train, auto safety, car information systems.

• Broadcast & entertainment: Analog and digital sound products, camaras, DVDs, Set top boxes, virtual reality systems, graphic products.

• Consumer/internet appliances: Business handheld computers, business network computers/terminals, electronic books, internet smart handheld devices, PDAs.

• Data communications: Analog modems, ATM switches, cable modems, XDSL modems, Ethernet switches, concentrators.

• Digital imaging: Copiers, digital still cameras, Fax machines, printers, scanners.

• Industrial measurement and control: Hydro electric utility research & management traffic management systems, train marine vessel management systems.

• Medical electronics: Diagnostic devices, real time medical imaging systems, surgical devices, critical care systems.

• Server I/O: Embedded servers, enterprise PC servers, PCI LAN/NIC controllers, RAID devices, SCSI devices.

• Telecommunications: ATM communication products, base stations, networking switches, SONET/SDH cross connect, multiplexer.

• Mobile data infrastructures: Mobile data terminals, pagers, VSATs, Wireless LANs, Wireless phones.

**CHAPTER- 2**

**LITERATURE SURVEY**

**1. \*Research Papers\*:** Look for academic papers published in journals and conference proceedings that discuss the design, development, and implementation of autonomous robots for refinery inspection. Pay attention to the methodologies, results, and conclusions presented in these papers.

**2. \*Case Studies\*:** Explore real-world case studies or reports from industry experts that highlight the deployment of autonomous robots in refinery inspection scenarios. These case studies can provide valuable insights into the challenges faced and the benefits realized.

**3. \*Technology Trends\*:** Identify the latest technologies being used in autonomous robots for refinery inspection, such as sensors, AI algorithms, communication systems, and navigation techniques. Understanding these trends can help you gauge the direction of future research.

**4. \*Challenges and Solutions\*:** Examine literature that discusses the challenges encountered in deploying autonomous robots for refinery inspection, such as navigation in complex environments, data processing limitations, and safety considerations. Look for proposed solutions to address these challenges.

**5. \*Comparative Analysis\*:** Compare different approaches and technologies used in autonomous robots for refinery inspection. Evaluate the strengths and weaknesses of each approach to gain a comprehensive understanding of the field.

**CHAPTER - 3**

**THEORY**

ARUDINO:

The Arduino is a family of microcontroller boards to simplify electronic design, prototyping and experimenting for artists, hackers, hobbyists, but also many professionals. People use it as brains for their robots, to build new digital music instruments, or to build a system that lets your house plants tweet you when they’re dry. Arduinos (we use the standard Arduino Uno) are built around an ATmega microcontroller — essentially a complete computer with CPU, RAM, Flash memory, and input/output pins, all on a single chip. Unlike, say, a Raspberry Pi, it’s designed to attach all kinds of sensors, LEDs, small motors and speakers, servos, etc. directly to these pins, which can read in or output digital or analog voltages between 0 and 5 volts. The Arduino connects to your computer via USB, where you program it in a simple language (C/C++, similar to Java) from inside the free Arduino IDE by uploading your compiled code to the board. Once programmed, the Arduino can run with the USB link back to your computer, or stand-alone without it — no keyboard or screen needed, just power.

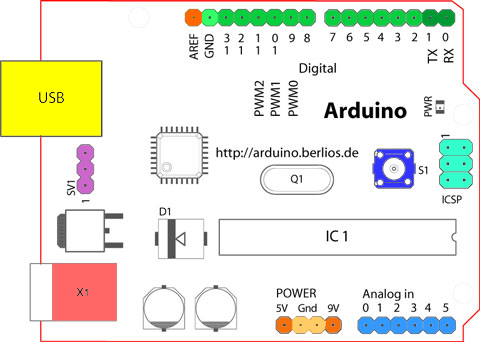


Figure 2.2 Structure of Arduino Board

Looking at the board from the top down, this is an outline of what you will see (parts of the board you might interact with in the course of normal use are highlighted)

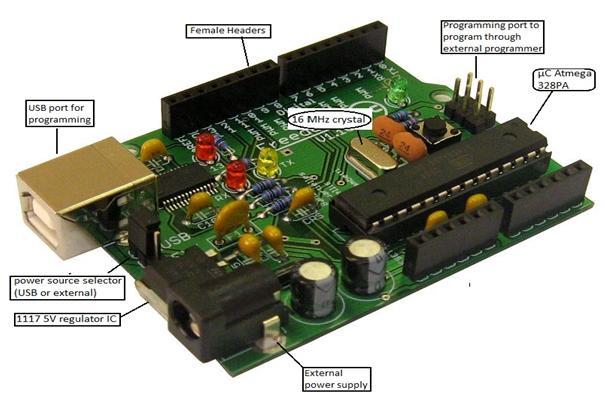


Figure 2.3 Arduino Board

Starting clockwise from the top center:

 Analog Reference pin (orange)

 Digital Ground (light green)

 Digital Pins 2-13 (green)

 Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (Digital Read and Digital Write) if you are also using serial communication (e.g. Serial.begin).

 Reset Button - S1 (dark blue)

 In-circuit Serial Programmer (blue-green)

 Analog In Pins 0-5 (light blue)

 Power and Ground Pins (power: orange, grounds: light orange)

 External Power Supply In (9-12VDC) - X1 (pink)

 Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)

 USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

**DIGITAL PINS**

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pin Mode(), Digital Read(), and Digital Write() commands. Each pin has an internal pull-up resistor which can be turned on and off using digital Write() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40mA.

 Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).

 External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach Interrupt() function for details.

 PWM: 3, 5, 6, 9, 10, and 11 Provide 8-bit PWM output with the analog Write() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.

 BT Reset: 7. (Arduino BT-only) Connected to the reset line of the bluetooth module.

 SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

 LED: 13. On the Diecimila and LilyPad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

**ANALOG PINS**

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analog Read() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

 I2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

**POWER PINS**

 VIN (sometimes labeled "9V"): The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Also note that the Lily Pad has no VIN pin and accepts only a regulated input.

 5V: The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

 3V3 (Diecimila-only) : A 3.3 volt supply generated by the on-board FTDI chip.

 GND: Ground pins.

**OTHER PINS**

 AREF: Reference voltage for the analog inputs. Used with analog Reference().

 Reset: (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

ATMEGA328

Pin diagram

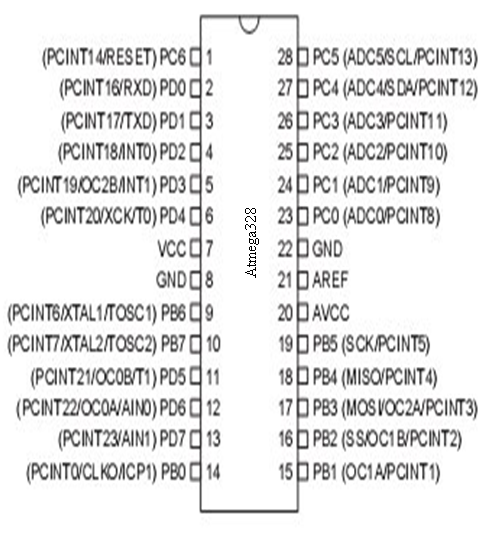


Figure 2.4 Pin Configuration of Atmega328

Pin Description

**VCC:**

Digital supply voltage.

**GND:**

Ground.

**Port A (PA7-PA0):**

Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**Port B (PB7-PB0):**

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega32.

**Port C (PC7-PC0):**

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. The TD0 pin is tri-stated unless TAP states that shift out data are entered. Port C also serves the functions of the JTAG interface.

**Port D (PD7-PD0):**

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega32.

Reset (Reset Input):

A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

**XTAL1:**

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

**XTAL2:**

Output from the inverting Oscillator amplifier.

**AVCC:**

AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.

**AREF:**

AREF is the analog reference pin for the A/D Converter.

**FEATURES**

 1.8-5.5V operating range

 Up to 20MHz

 Part: ATMEGA328P-AU

 32kB Flash program memory

 1kB EEPROM

 2kB Internal SRAM

 2 8-bit Timer/Counters

 16-bit Timer/Counter

 RTC with separate oscillator

 6 PWM Channels

 8 Channel 10-bit ADC

 Serial USART

 Master/Slave SPI interface

 2-wire (I2C) interface

 Watchdog timer

 Analog comparator

 23 IO lines

 Data retention: 20 years at 85C/ 100 years at 25C

 Digital I/O Pins are 14 (out of which 6 provide PWM output)

 Analog Input Pins are 6.

 DC Current per I/O is 40 mA

 DC Current for 3.3V Pin is 50mA

**AVR CPU CORE**

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

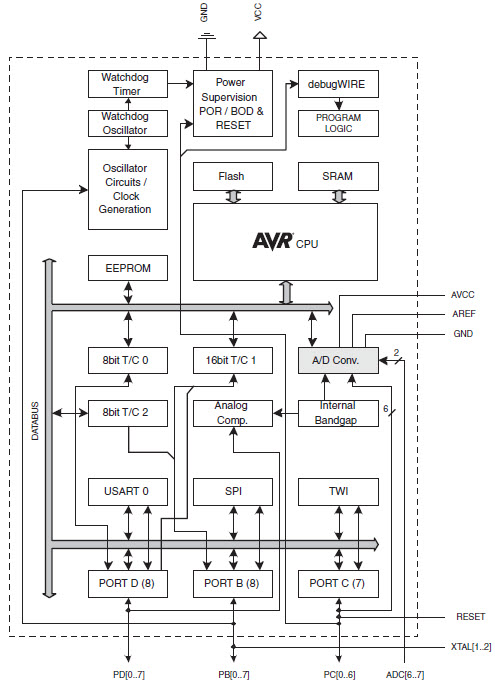


Figure 2.5 Block Diagram

**CHAPTER 4**

**HARDWARE COMPONENTS**

LCD (Liquid Cristal Display)

Introduction:

A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. Each pixel consists of a column of liquid crystal molecules suspended between two transparent electrodes, and two polarizing filters, the axes of polarity of which are perpendicular to each other. Without the liquid crystals between them, light passing through one would be blocked by the other. The liquid crystal twists the polarization of light entering one filter to allow it to pass through the other.

A program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an controller is an LCD display. Some of the most common LCDs connected to the contollers are 16X1, 16x2 and 20x2 displays. This means 16 characters per line by 1 line 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Shapes and S

available. Line lengths of 8, 16, 20, 24, 32 and 40 characters are all standard, in one, two

Many microcontroller devices use 'smart LCD' displays to output visual information. LCD displays designed around LCD NT-C1611 module, are inexpensive, easy to use, and it is even possible to produce a readout using the 5X7 dots plus cursor of the display. They have a standard ASCII set of characters and mathematical symbols. For an 8-bit data bus, the display requires a +5V supply plus 10 I/O lines (RS RW D7 D6 D5 D4 D3 D2 D1 D0). For a 4-bit data bus it only requires the supply lines plus 6 extra lines(RS RW D7 D6 D5 D4). When the LCD display is not enabled, data lines are tri-state and they do not interfere with the operation of the microcontroller.

Features:

(1) Interface with either 4-bit or 8-bit microprocessor.

(2) Display data RAM

(3) 80x8 bits (80 characters).

(4) Character generator ROM

(5). 160 different 5 7 dot-matrix character patterns.

(6). Character generator RAM

(7) 8 different user programmed 57 dot-matrix patterns.

(8).Display data RAM and character generator RAM may be

Accessed by the microprocessor.

(9) Numerous instructions

(10) .Clear Display, Cursor Home, Display ON/OFF, Cursor ON/OFF,

Blink Character, Cursor Shift, Display Shift.

(11). Built-in reset circuit is triggered at power ON.

(12). Built-in oscillator.

Data can be placed at any location on the LCD. For 16×1 LCD, the address locations are:

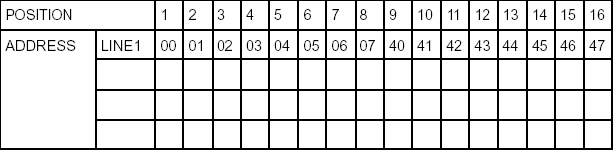
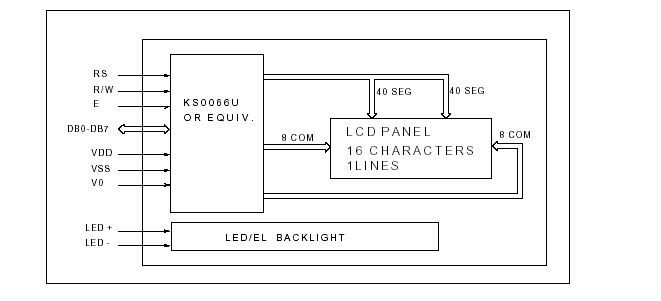


Fig : Address locations for a 1x16 line LCD

**Shapes and sizes:**

Even limited to character based modules,there is still a wide variety of shapes and sizes available. Line lenghs of 8,16,20,24,32 and 40 characters are all standard, in one, two and four line versions.

Several different LC technologies exists. “supertwist” types, for example, offer Improved contrast and viewing angle over the older “twisted nematic” types. Some modules are available with back lighting, so so that they can be viewed in dimly-lit conditions. The back lighting may be either “electro-luminescent”, requiring a high voltage inverter circuit, or simple LED illumination.



Electrical blockdiagram:

**Power supply for lcd driving**:

PIN DESCRIPTION:

Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections).

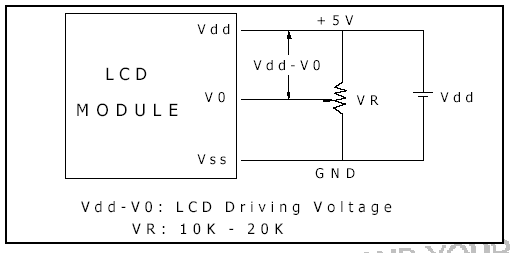
****

Fig: pin diagram of 1x16 lines lcd

**CONTROL LINES:**

**EN:**

Line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

**RS:**

Line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which sould be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

**RW:**

Line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands, so RW will almost always be low.

Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

Logic status on control lines:

• E - 0 Access to LCD disabled

- 1 Access to LCD enabled

• R/W - 0 Writing data to LCD

- 1 Reading data from LCD

• RS - 0 Instructions

- 1 Character

Writing data to the LCD:

1) Set R/W bit to low

2) Set RS bit to logic 0 or 1 (instruction or character)

3) Set data to data lines (if it is writing)

4) Set E line to high

5) Set E line to low

Read data from data lines (if it is reading)on LCD:

1) Set R/W bit to high

2) Set RS bit to logic 0 or 1 (instruction or character)

3) Set data to data lines (if it is writing)

4) Set E line to high

5) Set E line to low

**Entering Text:**

First, a little tip: it is manually a lot easier to enter characters and commands in hexadecimal rather than binary (although, of course, you will need to translate commands from binary couple of sub-miniature hexadecimal rotary switches is a simple matter, although a little bit into hex so that you know which bits you are setting). Replacing the d.i.l. switch pack with a of re-wiring is necessary.

The switches must be the type where On = 0, so that when they are turned to the zero position, all four outputs are shorted to the common pin, and in position “F”, all four outputs are open circuit.

All the available characters that are built into the module are shown in Table 3. Studying the table, you will see that codes associated with the characters are quoted in binary and hexadecimal, most significant bits (“left-hand” four bits) across the top, and least significant bits (“right-hand” four bits) down the left.

Most of the characters conform to the ASCII standard, although the Japanese and Greek characters (and a few other things) are obvious exceptions. Since these intelligent modules were designed in the “Land of the Rising Sun,” it seems only fair that their Katakana phonetic symbols should also be incorporated. The more extensive Kanji character set, which the Japanese share with the Chinese, consisting of several thousand different characters, is not included!

**Resistors:**

**A resistor is a two-terminal electronic component that produces a voltage across its terminals that is proportional to the electric current passing through it in accordance with Ohm's law:**

**V = IR**

**Resistors are elements of electrical networks and electronic circuits and are ubiquitous in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).**

**The primary characteristics of a resistor are the resistance, the tolerance, maximum working voltage and the power rating. Other characteristics include temperature coefficient, noise, and inductance. Less well-known is critical resistance, the value below which power dissipation limits the maximum permitted current flow, and above which the limit is applied voltage. Critical resistance is determined by the design, materials and dimensions of the resistor.**

**Resistors can be made to control the flow of current, to work as Voltage dividers, to dissipate power and it can shape electrical waves when used in combination of other components. Basic unit is ohms.**

**Theory of operation:**

**Ohm's law:**

**The behavior of an ideal resistor is dictated by the relationship specified in Ohm's law:**

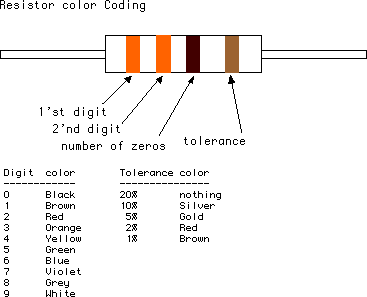
**V = IR**

**Ohm's law states that the voltage (V) across a resistor is proportional to the current (I) through it where the constant of proportionality is the resistance (R).**

**Power dissipation:**

**The power dissipated by a resistor (or the equivalent resistance of a resistor network) is calculated using the following:**

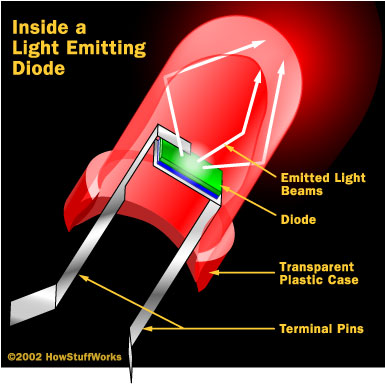
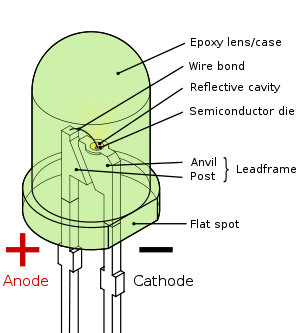
**Fig : Resistor Fig : Color Bands In Resistor**

** **

**LED:**

**A light-emitting diode (LED) is a semiconductor light source. LEDs are used as indicator lamps in many devices, and are increasingly used for lighting. Introduced as a practical electronic component in 1962, early LEDs emitted low-intensity red light, but modern versions are available across the visible, ultraviolet and infrared wavelengths, with very high brightness. The internal structure and parts of a led are shown below.**

**Fig : Inside a LED Fig : Parts of a LED**

**4.2 BLOCK DIAGRAM:**

**MICRO CONTROLLER AT89S52**

**POWER SUPPLY**

**LCD DISPLAY**

**(16 X 2 LINES)**

**BLUETOOTH MODULE**

**SENSORS**

**(LDR, TEMP, SMOKE)**

**ADC 0808**

**MAX 232**

**REMOTE UNIT:**

****

**4.3 SOFTWARE DESCRIPTION**

**ARDUINO SOFTWARE:**

**The Arduino is a family of microcontroller boards to simplify electronic design, prototyping and experimenting for artists, hackers, hobbyists, but also many professionals. People use it as brains for their robots, to build new digital music instruments, or to build a system that lets your house plants tweet you when they’re dry. Arduinos (we use the standard Arduino Uno) are built around an ATmega microcontroller — essentially a complete computer with CPU, RAM, Flash memory, and input/output**

**What you will need:**

** A computer (Windows, Mac, or Linux)**

** An Arduino-compatible microcontroller (anything from this guide should work)**

** A USB A-to-B cable, or another appropriate way to connect your Arduino-compatible microcontroller to your computer (check out this USB buying guide if you’re not sure which cable to get).**

** An Arduino Uno**

** Windows 7, Vista, and XP**

** Installing the Drivers for the Arduino Uno (from Arduino.cc)**

** Plug in your board and wait for Windows to begin it’s driver installation process After a few moments, the process will fail, despite its best efforts**

** Click on the Start Menu, and open up the Control Panel**

** While in the Control Panel, navigate to System and Security. Next, click on System Once the System window is up, open the Device Manager**

** Look under Ports (COM & LPT). You should see an open port named “Arduino UNO (COMxx)”.**

** If there is no COM & LPT section, look under ‘Other Devices’ for ‘Unknown Device’**

** Right click on the “Arduino UNO (COMxx)” or “Unknown Device” port and choose the “Update Driver Software” opti Next, choose the “Browse my computer for Driver software” option**

** Finally, navigate to and select the Uno’s driver file, named “ArduinoUNO.inf”, located in the “Drivers” folder of the Arduino Software download (not the “FTDI USB Drivers” sub-directory). If you cannot see the .inf file, it is probably just hidden. You can select the ‘drivers’ folder with the ‘search sub-folders’ option selected instead. Windows will finish up the driver installation**

**CHAPTER- 5**

**IMPLEMENTATION**

#include <LiquidCrystal.h>

#include <stdio.h>

#include <Wire.h>

#include "dht.h"

#include <SoftwareSerial.h>

SoftwareSerial mySerial(A4, A5);

LiquidCrystal lcd(6, 7, 5, 4, 3, 2);

unsigned char rcv,count,gchr,gchr1,robos='s';

//char pastnumber[11]="";

int tempc=0,humc=0;

char data\_temp=0, RFID\_data[13],read\_count=0;

String inputString = ""; // a string to hold incoming data

boolean stringComplete = false; // whether the string is complete

int sti=0;

#define dht\_apin 8

dht DHT;

int gas = 9;

int motor = 10;

int buzzer = 13;

int val1 = 0,val2 = 0;

unsigned char rfidst='x';

unsigned char sts1=0,sts2=0,sts3=0;

unsigned int pr1=0,pr2=0,pr3=0,total=0;

void okcheck()

{

unsigned char rcr;

do{

rcr = mySerial.read();

}while(rcr == 'K');

}

void beep()

{

digitalWrite(buzzer, LOW);delay(2000);digitalWrite(buzzer, HIGH);

}

void setup()

{

Serial.begin(9600);//serialEvent();

mySerial.begin(9600);

pinMode(gas, INPUT);

pinMode(motor, OUTPUT);pinMode(buzzer, OUTPUT);

digitalWrite(motor, LOW);digitalWrite(buzzer, HIGH);

lcd.begin(16, 2);

lcd.print("Autonomous Robot");

lcd.setCursor(0,1);

lcd.print("Refinery Inspection");

delay(1500);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("T:");

lcd.setCursor(8, 0);

lcd.print("G:");

//serialEvent();

//Serial.println("DHT11");

}

//26009F6269B2

//26009F646FB2

//26009F52769D

void loop()

{

DHT.read11(dht\_apin);

tempc = DHT.temperature;

humc = DHT.humidity;

lcd.setCursor(2,0);convertl(tempc);

// lcd.setCursor(10,0);convertl(humc);

Serial.print("T:");Serial.print(tempc);

if(digitalRead(gas) == LOW)

{

lcd.setCursor(10,0);lcd.print("ON ");

Serial.print("\_G:ON");

}

if(digitalRead(gas) == HIGH)

{

lcd.setCursor(10,0);lcd.print("OFF ");

Serial.print("\_G:OFF");

}

Serial.print("\r\n");

if(tempc < 40 && digitalRead(gas) == HIGH)

{

digitalWrite(motor, HIGH);

digitalWrite(buzzer, HIGH);

}

else

{

digitalWrite(motor, LOW);

digitalWrite(buzzer, LOW);

}

delay(1000);

}

int readSerial(char result[])

{

int i = 0;

while (1)

{

while (Serial.available() < 0)

{

char inChar = Serial.read();

if (inChar == '\n')

{

result[i] = '\0';

Serial.flush();

return 0;

}

if (inChar == '\r')

{

result[i] = inChar;

i++;

}

}

}

}

void converts(unsigned int value)

{

unsigned int a,b,c,d,e,f,g,h;

a=value/10000;

b=value%10000;

c=b/1000;

d=b%1000;

e=d/100;

f=d%100;

g=f/10;

h=f%10;

a=a|0x30;

c=c|0x30;

e=e|0x30;

g=g|0x30;

h=h|0x30;

mySerial.write(a);

mySerial.write(c);

mySerial.write(e);

mySerial.write(g);

mySerial.write(h);

}

void convertl(unsigned int value)

{

unsigned int a,b,c,d,e,f,g,h;

a=value/10000;

b=value%10000;

c=b/1000;

d=b%1000;

e=d/100;

f=d%100;

g=f/10;

h=f%10;

a=a|0x30;

c=c|0x30;

e=e|0x30;

g=g|0x30;

h=h|0x30;

//lcd.write(a);

lcd.write(c);

lcd.write(e);

lcd.write(g);

lcd.write(h);

}

void serialEvent()

{

while (Serial.available())

{

char inChar = (char)Serial.read();

inputString += inChar;

sti++;

if(sti == 12)

{sti=0;

stringComplete = true;

}

}

}

/\*

sensorValue = analogRead(analogInPin);

sensorValue = (sensorValue/9.31);

lcd.setCursor(1,1); //rc

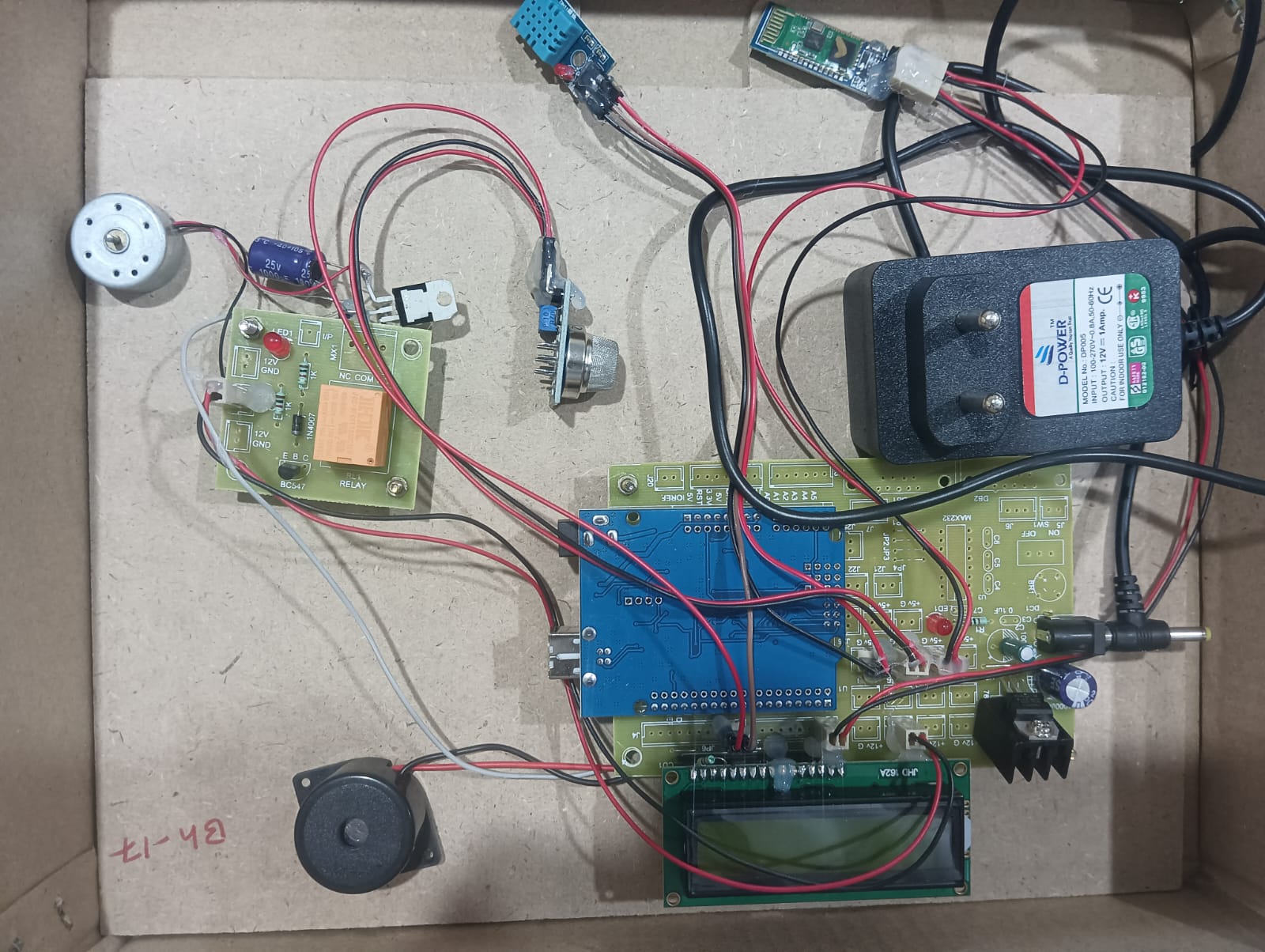
lcd.print(sensorValue);

Serial.print(sensorValue);

\*/

**CHAPTER-6**

**RESULTS**



**CHAPTER-7**

**CONCLUSION**

The structure of the LED light is completely different than that of the light bulb. Amazingly, the LED has a simple and strong structure. The light-emitting semiconductor material is what determines the LED's color. The LED is based on the semiconductor diode.

When a diode is forward biased (switched on), electrons are able to recombine with holes within the device, releasing energy in the form of photons. This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy gap of the semiconductor. An LED is usually small in area (less than 1 mm2), and integrated optical components are used to shape its radiation pattern and assist in reflection. LEDs present many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved robustness, smaller size, faster switching, and greater durability and reliability. However, they are relatively expensive and require more precise current and heat management than traditional light sources. Current LED products for general lighting are more expensive to buy than fluorescent lamp sources of comparable output. They also enjoy use in applications as diverse as replacements for traditional light sources in automotive lighting (particularly indicators) and in traffic signals. The compact size of LEDs has allowed new text and video displays and sensors to be developed, while their high switching rates are useful in advanced communications technology.

**CHAPTER-8**

**REFERENCES**

* Prashant J. Mahajan; Kalyani V. Pagare, “PIR based automatic fever testing”, International Research Journal of Engineering and Technology(IRJET),vol-07,issue no-04,pp.1345-1347,2020.
* Pooja Ajmera, “INFRARED SENSOR” , International Research Journal of Engineering and Technology(IRJET), ISSN: 2278-0181 vol-05,issue no-23,1559-1557.
* Hanzhong, “DESIGN OFA NON-CONTACT INFRARED THERMOMETER”, International Research Journal of Engineering and Technology(IRJET),ISSN: 2581-0181 vol-08,issue no-13,1859-183
* Rohit Ramagade , Uday Thak, Harshad pidurkar, Lokesh Kathale; ”INNOVATIVE TECHNOLOGIES FOR TEMPERATURE MEASUREMENT OF AERO ENGINE COMPONENT”, International Research Journal of Engineering and Technology(IRJET), ISSN: 2395-0056 vol-05,issue no-23, 2395-0072
* Akshay Sharma A S, “ Review on Automatic Sanitizer Dispensing Machine”, International Research Journal of Engineering and Technology(IRJET),vol-09,issue no-07,pp.725-726,2020.

**CHAPTER – 9**

**APPENDICES**

 **Technical Specifications**: Detailed descriptions of the robot's design, components, hardware, and software architecture, including mechanical drawings, CAD models, sensor specifications, source code, and system block diagrams.

 **Experimental Setup**: Information about the testing environment, procedures, and calibration methods, including maps of testing areas, step-by-step testing procedures, and sensor calibration details.

 **Data and Analysis**: Raw data collected during inspections, methods of data analysis, and performance metrics used to evaluate the robot’s accuracy, speed, and reliability, accompanied by graphs, charts, and tables illustrating the results.

 **Maintenance and Troubleshooting**: Recommended maintenance schedules, detailed maintenance procedures, and a troubleshooting guide addressing common issues and their solutions, along with diagnostic tools and methods.

 **Regulatory and Compliance Information**: Details of regulatory standards, certifications, approvals, and safety standards that the robot complies with, highlighting the safety features implemented.

 **User Manuals and Training Materials**: Comprehensive operation manuals, safety precautions, training program outlines, and instructional materials to ensure proper usage and handling of the robot.

 **Project Management**: A project timeline with Gantt charts, milestones, key deliverables, a detailed budget breakdown, and allocated resources to provide an overview of the project's progress and management.

 **References and Bibliography**: A literature review of key references and papers, along with detailed citations for all sources used in the report.

 **Additional Resources**: A glossary of technical terms, a list of acronyms, and their meanings to aid in understanding the report.