CHAPTER -1
INTRODUCTION

1.1 GENERAL INTRODUCTION

In today's scenario Railway safety becomes the most important aspect of railways all over the world. As we know the Railways is the cheapest mode of transportation, and due to manual operation, accidents are likely to happen. There are 30348 level crossings on Indian Railways across the country.18785 are manhandled and 11563 are non-man handled level crossings out of 303048 level crossings. To avoid accidents over the previous five years 4792 level crossings have been removed by the respective Zonal railways of Indian Railways. The Indian ministry of Railways made a decision focusing on eliminating all level crossings on availability of railway funds, which could be controlled automatically. The suggested system helps in achieving the safety and to prevent accidents at the level crossings that are non-man handled. The Automatic railway gate control system can be employed under non manhandled level crossing where the chances of accidents are higher and requirement of reliable operation are there. Since, the proposed model suggests an automatic system, it helps in reducing the error which is in manual operation and it will be used as a highly reliable source. The proposed model of automatic gate control at level crossings is highly economical based on the arrangement done by using Arduino and Servo motor which makes the design for use in almost every non man handled that is unmanned railway crossings. The proposed model suggests a design to control a railway level-crossing by servo motor using Arduino controller. The connection of the motor is done from Arduino with the help of a driver IC for controlling the railway gate. Two IR sensors are used to detect arrival of the train and two are used to detect departure of the train. IR sensors are checking the complete closing of the gate.

1.2 EXISTING SYSTEM

Existing system consists of only manual operations.

1.3 DISADVANTAGES OF EXISTING SYSTEM

There is no perfect control over the safety gates if there is no presence of a station master as it is with only manual operations.

1.4 PROPOSED SYSTEM

The proposed system consist of both IOT controls and manual operations

1.5 ADVANTAGES

As the existing system consists of both manual operations and IOT controls the station master can operate through mobile though he is not in the control room.

CHAPTER - 2

PROJECT DESCRIPTION

2.1 INTRODUCTIONS TO EMBEDDED SYSTEMS

Each day, our lives become more dependent on 'embedded systems', digital information technology that is embedded in our environment. More than 98% of processors applied today are in embedded systems, and are no longer visible to the customer as 'computers' in the ordinary sense. An Embedded System is a special-purpose system in which the computer is completely encapsulated by or dedicated to the device or system it controls. Unlike a general-purpose computer, such as a personal computer, an embedded system performs one or a few predefined tasks, usually with very specific requirements. Since the system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product. Embedded systems are often mass- produced, benefiting from economies of scale.

The increasing use of PC hardware is one of the most important developments in high-end embedded systems in recent years. Hardware costs of high-end systems have dropped dramatically as a result of this trend, making some projects feasible which previously would not have been done because of the high cost of non-PC-based embedded hardware. But software choices for the embedded PC platform are not nearly as attractive as the hardware.

Typically, an embedded system is housed on a single microprocessor board with the programs stored in ROM. Virtually all appliances that have a digital interface -- watches, microwaves, VCRs, cars -- utilize embedded systems. Some embedded systems include an operating system, but many are so specialized that the entire logic can be implemented as a single program. Physically, Embedded Systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights,

factory controllers, or the systems controlling nuclear power plants. The applications software on such processors is sometimes referred to as firmware. The simplest devices consist of a single microprocessor (often called a "chip"), which may itself be packaged with other chips in a hybrid system or Application Specific Integrated Circuit (ASIC). Its input comes from a detector or sensor and its output goes to a switch or activator which (for example) may start or stop the operation of a machine or, by operating a valve, may control the flow of fuel to an engine. As the embedded system is the combination of both software and hardware.

2.2 DEFINITION OF EMBEDDED SYSTEMS

Embedded system is defined as, for a particular/specific application implementing the software code to interact directly with that particular hardware that we built. Software is used for providing features and flexibility.

Hardware = {Processors, ASICs, Memory...} is used for Performance (& sometimes security).There are many definitions of embedded systems but all of these can be combined into a single concept. An embedded system is a special purpose computer system that is used for a particular task.

2.3 EXAMPLES OF EMBEDDED SYSTEMS

Embedded systems are found in a wide range of application areas. Originally they were used only for expensive industrial control applications, but as technology brought down the cost of dedicated processors, they began to appear in moderately expensive applications such as automobiles, communication and office equipment and television Today's embedded systems are so inexpensive that they are used in almost every electronic product in our life. Embedded systems are often designed for mass production.

1. Automatic Teller Machines
2. Cellular telephone and telephone switches
3. Computer network equipment
4. Computer printers

2.4 HISTORY OF EMBEDDED SYSTEMS

The first recognizably modern embedded systems was the Apollo Guidance Computer, developed by "Charles Stark Draper" at the MIT Instrumentation Laboratory. At the project inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass produced embedded system was the Autonetics D-17 guidance computer for the Minuteman missile, released in 1961.It was built from transistor logic and had a hard disk for main memory .When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high volume use of integrated circuits. This program alone reduced prices on quad and gate ICs from $1000/ each to $3/ each permitting their use in commercial products.

Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality. The first microprocessor for example, the Intel 4004, was designed for calculators and other small systems but still required many external memory and support chips. In 1978 National Engineering Manufacturers Association released a "standard" for programmable microcontrollers, including almost any computer based controllers, such as single board computers, numerical and event based controllers. Embedded Systems are designed to some specific task, rather than be a general-purpose computer for multitasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirement, allowing the system hardware to be simplified to reduce cost.

2.5 BLOCK DIAGRAM

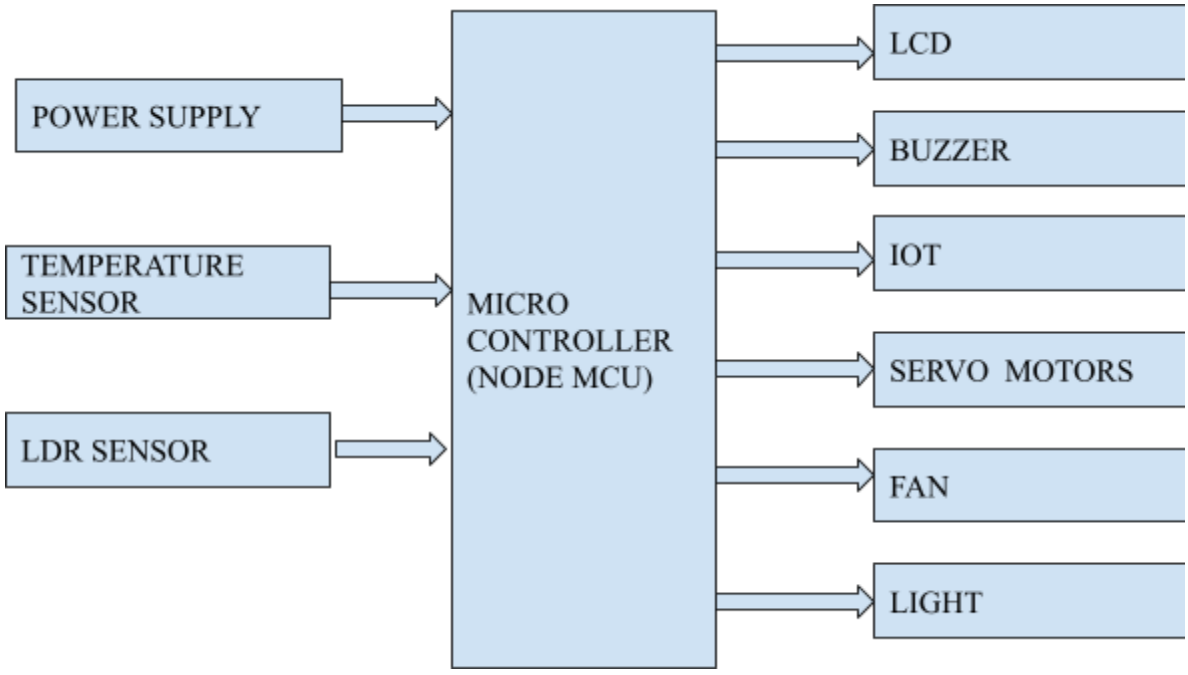

FIGURE 2.5 BLOCK DIAGRAM

In the proposed system, sensors are used to detect the train arrival and departure. The System uses three different sensors to control the rail arrival and departure. Arduino is used for program the sensors. As shown in functional diagram of our proposed model in Fig. 1. The following materials and components are used in this proposed system for automatic gateway control system. The block diagram as shown in Fig. 1, describes the working of our proposed system. Where an Arduino is used for connecting devices like IR sensor, Servo Motor. In the Proposed Model the servo motor is used for monitoring of cross Way and IR sensors is used to detect the motion of objects nearby the crossway, which when detects the train or motion of any other vehicle nearby the crossway it takes the corresponding action of whether opening or closing the gateway at the crossing.

2.6 HARDWARE COMPONENTS

1. Power supply
2. Micro controller
3. Temperature sensor
4. LDR sensor
5. Servo Motors
6. LCD Display
7. Buzzer

2.6.1 POWER SUPPLY

The DC power supply is practically converted to each and every stage in an electronic system. Thus a common requirement for all these phases will be the DC power supply. All low power system can be run with a battery. But, for a long time operating devices, batteries could prove to be costly and complicated. The best method used is in the form of an unregulated power supply –a combination of a transformer, rectifier and a filter. The diagram is shown below.

All devices will have a certain power supply limit and the electronic circuits inside these devices must be able to supply a constant DC voltage within this limit. This DC supply is regulated and limited in terms of voltage and current. But the supply provided from mains may be fluctuating and could easily break down the electronic equipment, if not properly limited. This work of converting an unregulated alternating

current (AC) or voltage to a limited Direct current (DC) or voltage to make the output constant regardless of the fluctuations in input, is done by a regulated power supply circuit.

All the active and passive electronic devices will have a certain DC operating point (Q-point or Quiescent point), and this point must be achieved by the source of DC power. A step down transformer is used to reduce the voltage level to the devices needs. In India, a 1 Ø supply is available at 230 volts. The output of the transformer is a pulsating sinusoidal AC voltage, which is converted to pulsating DC with the help of a rectifier.

This output is given to a filter circuit which reduces the AC ripples, and passes the DC components. But here are certain disadvantages in using an unregulated power supply.

Regulated power supply is an electronic circuit that is designed to provide a constant dc voltage of predetermined value across load terminals irrespective of ac mains fluctuations or load variations.

A regulated power supply essentially consists of an ordinary power supply and a voltage regulating device, as illustrated in the figure. The output from an ordinary power supply is fed to the voltage regulating device that provides the final output. The output voltage remains constant irrespective of variations in the ac input voltage or variations in output (or load) current.

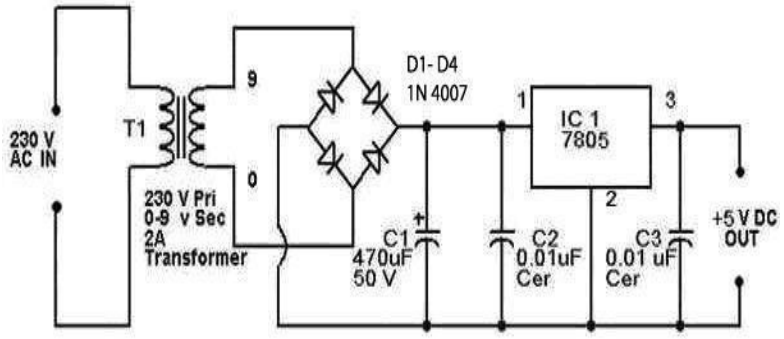

FIGURE 2.6.1A CIRCUIT DIAGRAM OF REGULATED POWER SUPPLY

2.6.2 TRANSFORMER

A transformer can be defined as a static device which helps in the transformation of electric power in one circuit to electric power of the same frequency in another circuit. The voltage can be raised or lowered in a circuit, but with a proportional increase or decrease in the current ratings.

2.6.2A WORKING PRINCIPLE OF TRANSFORMER

The main principle of operation of a transformer is mutual inductance between two circuits which is linked by a common magnetic flux. A basic transformer consists of two coils that are electrically separate and inductive, but are magnetically linked through a path of reluctance. The working principle of the transformer can be understood from the figure below.

As shown below the electrical transformer has primary and secondary windings. The core laminations are joined in the form of strips in between the strips you can see that there are some narrow gaps right through the cross-section of the core. These staggered joints are said to be 'imbricated'. Both the coils have high mutual inductance.A mutual electro-motive force is induced in the transformer from the alternating flux that is set up in the laminated core, due to the coil that is connected to a source of alternating voltage.

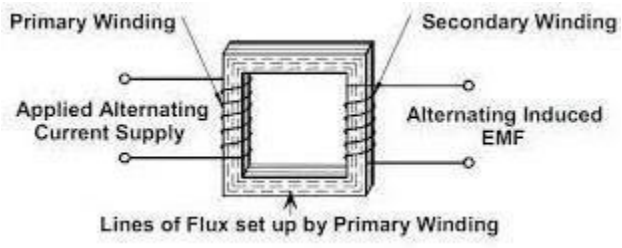

FIGURE 2.6.2A WORKING PRINCIPLE OF TRANSFORMER

2.6.3 RECTIFIER

A rectifier is an electrical device that converts alternating current (AC), which periodically reverses direction, to direct current (DC), which flows in only one direction. The process is known as rectification.

2.6.4 BRIDGE RECTIFIER

Bridge is a type of electrical circuit. Bridge rectifier is a type of rectifier in which diodes were arranged in the form of a bridge. This provides full wave rectification and is of low cost. So it is used in many applications.

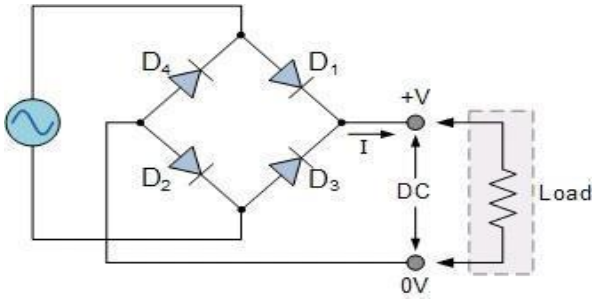

FIGURE 2.6.4 CIRCUIT DIAGRAM OF BRIDGE RECTIFIER

2.7 NODE MCU

The Internet of Things (IoT) has been a trending field in the world of technology. It has changed the way we work. Physical objects and the digital world are connected now more than ever. Keeping this in mind, Espressif Systems (A Shanghai-based Semiconductor Company) has released an adorable, bite-sized WiFi enabled microcontroller – **ESP8266**, at an unbelievable price! For less than $3, it can monitor and control things from anywhere in the world – **perfect for just about any IoT project**.

ESP-12E Module

The development board equips the ESP-12E module containing ESP8266 chip having **Tensilica Xtensa® 32-bit LX106 RISC microprocessor** which operates at **80 to 160 MHz** adjustable clock frequency and supports **RTOS**.

ESP-12E Chip

- Tensilica Xtensa® 32-bit LX106
- 80 to 160 MHz Clock Freq.
- 128kB internal RAM
- 4MB external flash

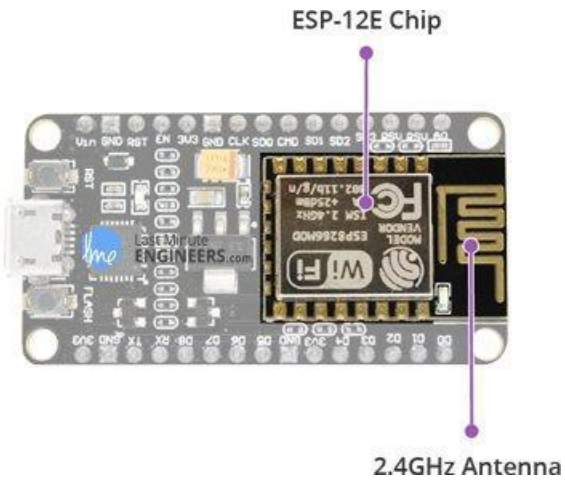

FIGURE 2.7.1 ESP CHIP

There's also **128 KB RAM and 4MB of Flash memory** (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IoT devices nowadays.

The ESP8266 Integrates **802.11b/g/n HT40 Wi-Fi transceiver**, so it can not only connect to a WiFi network and interact with the Internet, but it can also set up a network of its own, allowing other devices to connect directly to it. This makes the ESP8266 NodeMCU even more versatile.

Power Requirement

As the operating voltage range of ESP8266 is **3V to 3.6V**, the board comes with a LDO voltage regulator to keep the voltage steady at 3.3V. It can reliably supply up to 600mA, which should be more than enough when ESP8266 pulls as much as **80mA during RF**

transmissions. The output of the regulator is also broken out to one of the sides of the board and labeled as 3V3. This pin can be used to supply power to external components.

Power Requirement

- Operating Voltage: 2.5V to 3.6V
- On-board 3.3V 600mA regulator
- 80mA Operating Current
- 20 µA during Sleep Mode

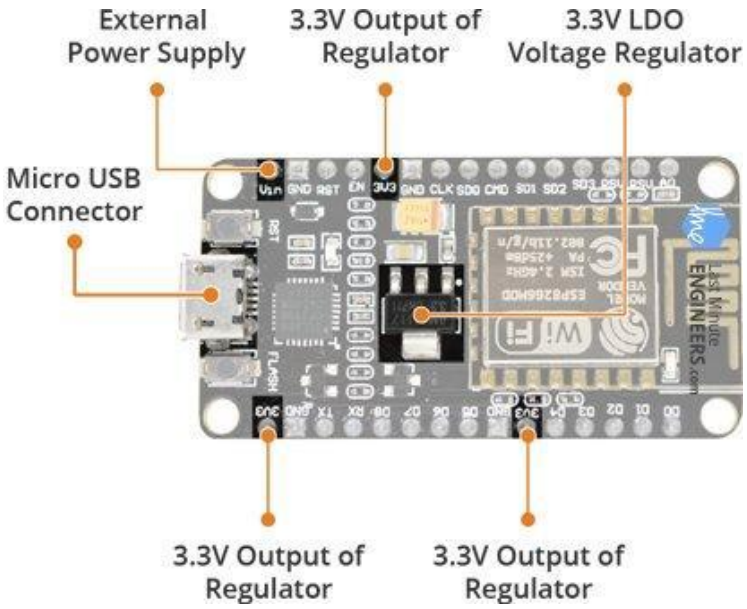

FIGURE 2.7.2 POWER REQUIREMENT OF NODE MCU

Power to the ESP8266 NodeMCU is supplied via the **on-board MicroB USB connector**. Alternatively, if you have a regulated 5V voltage source, the **VIN pin** can be used to directly supply the ESP8266 and its peripherals.

Warning:

The ESP8266 requires a 3.3V power supply and 3.3V logic levels for communication. The GPIO pins are not 5V-tolerant! If you want to interface the board with 5V (or higher) components, you'll need to do some level shifting.

Peripherals and I/O

The ESP8266 NodeMCU has total **17 GPIO pins** broken out to the pin headers on both sides of the development board. These pins can be assigned to all sorts of peripheral duties, including:

- **ADC channel** – A 10-bit ADC channel.
- **UART interface** – UART interface is used to load code serially.
- **PWM outputs** – PWM pins for dimming LEDs or controlling motors.
- **SPI, I2C & I2S interface** – SPI and I2C interface to hook up all sorts of sensors and peripherals.
- **I2S interface** – I2S interface if you want to add sound to your project.

Multiplexed I/Os

- 1 ADC channels
- 2 UART interfaces
- 4 PWM outputs
- SPI, I2C & I2S interface

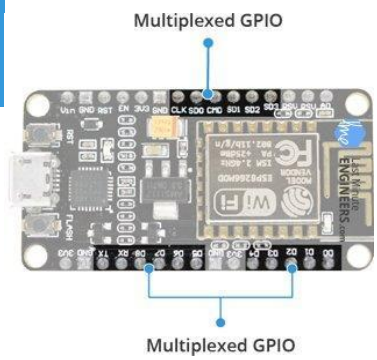

FIGURE 2.7.3 MULTIPLEXED I/O'S OF NODE MCU

Switches & Indicators

- RST – Reset the ESP8266 chip
- FLASH – Download new programs
- Blue LED – User Programmable

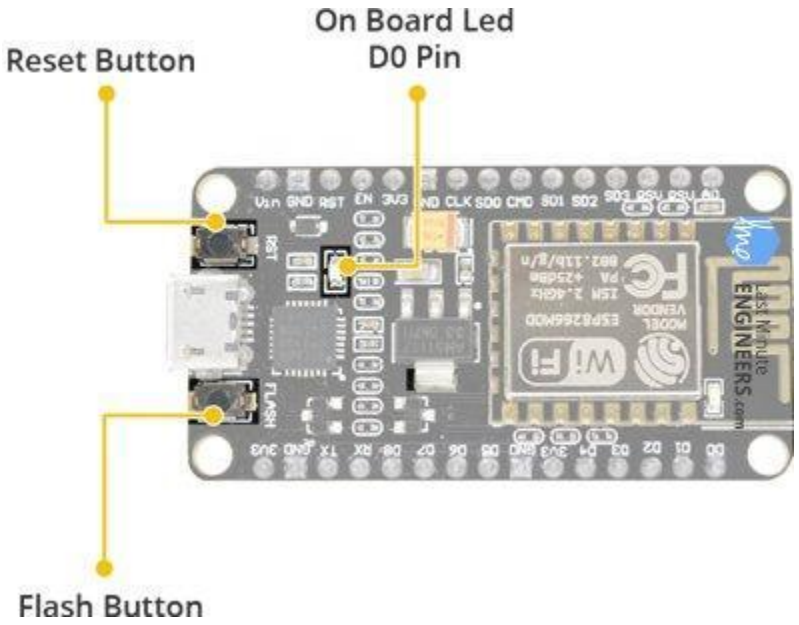

FIGURE 2.7.4 SWITCHES & INDICATORS OF NODE MCU

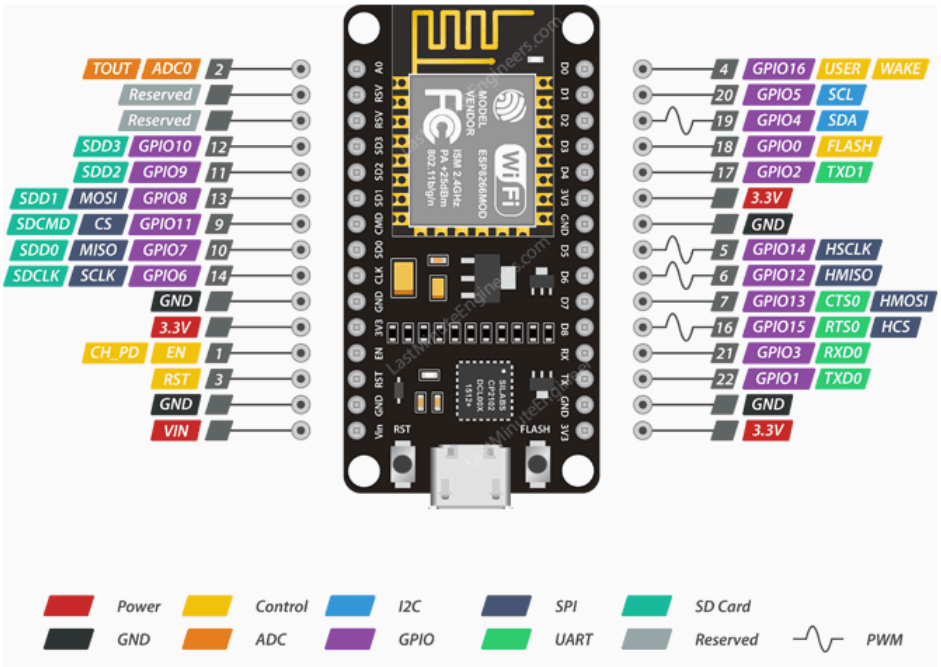

FIGURE 2.7.5 PIN CONFIGURATION OF NODEMCU

Power Pins There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.

GND is a ground pin of ESP8266 NodeMCU development board.

I2C Pins are used to hook up all sorts of I2C sensors and peripherals in your project. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins ESP8266 NodeMCU has 17 GPIO pins which can be assigned to various functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ADC Channel The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins ESP8266 NodeMCU has 2 UART interfaces, i.e. UART0 and UART1, which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. It supports fluid control. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO

SDIO Pins ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

PWM Pins The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 µs to 10000 µs, i.e., between 100 Hz and 1 kHz.

Control Pins are used to control ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

- EN pin – The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- RST pin – RST pin is used to reset the ESP8266 chip.

- WAKE pin – Wake pin is used to wake the chip from deep-sleep.

2.8 TEMPERATURE SENSOR

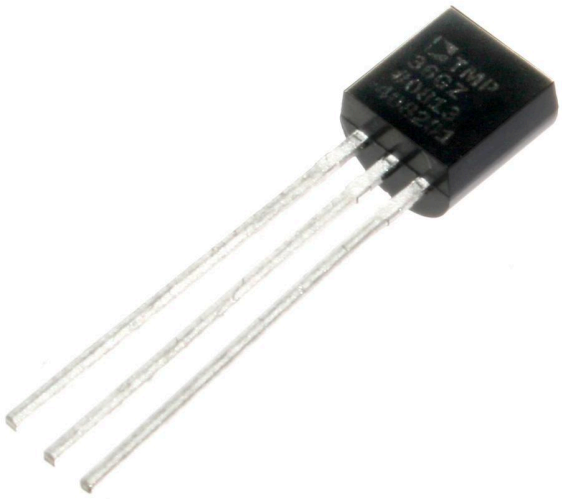

FIGURE 2.8 TEMPERATURE SENSOR

A temperature sensor is an electronic device that measures the temperature of its environment and converts the input data into electronic data to record, monitor, or signal temperature changes. There are many different types of temperature sensors.

2.9 LDR SENSOR

A light-dependent resistor or LDR is an example of an electrical component that responds to light. When light beams strike it, the resistance changes right away. An LDR's resistance levels can vary by several orders of magnitude. As the light level rises, the resistance value will decrease.

2.9 LDR SENSOR

2.10 SERVO MOTORS

There are lots of servo motors available in the market and each one has its own speciality and applications. The following two paragraphs will help you identify the right type of servo motor for your project/system.

Most of the hobby Servo motors operates from 4.8V to 6.5V, the higher the voltage higher the torque we can achieve, but most commonly they are operated at +5V. Almost all hobby servo motors can rotate only from 0° to 180° due to their gear arrangement so make sure you project can live with the half circle if no, you can prefer for a 0° to 360° motor or modify the motor to make a full circle. The gears in the motors are easily subjected to wear and tear, so if your application requires stronger and long running motors you can go with metal gears or just stick with normal plastic gear.

Next comes the most important parameter, which is the **torque** at which the motor operates. Again there are many choices here but the commonly available one is the 2.5kg/cm torque which comes with the Towerpro SG90 Motor. This 2.5kg/cm torque means that the motor can pull a weight of 2.5kg when it is suspended at a distance of 1cm. So if you suspend the load at 0.5cm then the motor can pull a load of 5kg similarly if you suspend the load at 2cm then can pull only 1.25. Based on the load which you use in the project you can select the motor with proper torque. The below picture will illustrate the same.

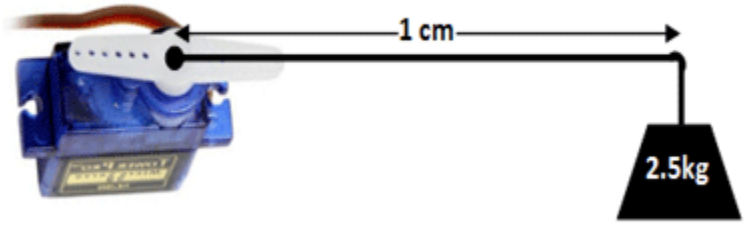

FIGURE 2.10 SERVO MOTOR

2.11 HOW TO USE SERVO MOTOR

After selecting the right Servo motor for the project, comes the question of how to use it. As we know there are three wires coming out of this motor. The description of the same is given on top of this page. To make this motor rotate, we have to power the motor with +5V using the Red and Brown wire and send PWM signals to the Orange color wire. Hence we need something that could generate PWM signals to make this motor work, this something could be anything like a 555 Timer or other Microcontroller platforms like Arduino, PIC, ARM or even a microprocessor like Raspberry Pie. Now, how to control the direction of the motor? To understand that let us look at the picture given in the datasheet.

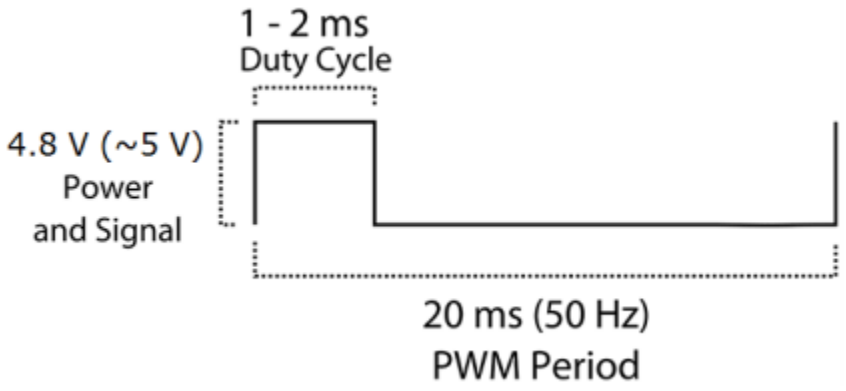

FIGURE 2.11 GENERATION OF PWM SIGNAL

From the picture we can understand that the PWM signal produced should have a frequency of 50Hz, that is the PWM period should be 20ms. Out of which the On-Time can vary from 1ms to 2ms. So when the on-time is 1ms the motor will be in 0° and when 1.5ms the motor will be 90°, similarly when it is 2ms it will

be 180°. So, by varying the on-time from 1ms to 2ms the motor can be controlled from 0° to 180°.

2.12 LCD DISPLAY

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

FIGURE 2.12 LCD DISPLAY

2.13 LCD PIN CONFIGURATION

- Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
- Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
- Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
- Pin4 (Register Select/Control Pin): This pin toggles among the command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
- Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
- Pin 6 (Enable/Control Pin): This pin should be held high to execute the Read/Write process, and it is connected to the microcontroller unit & constantly held high.
- Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller units like 0 to 7.
- Pin15 (+ve pin of the LED): This pin is connected to +5V
- Pin 16 (-ve pin of the LED): This pin is connected to GND.

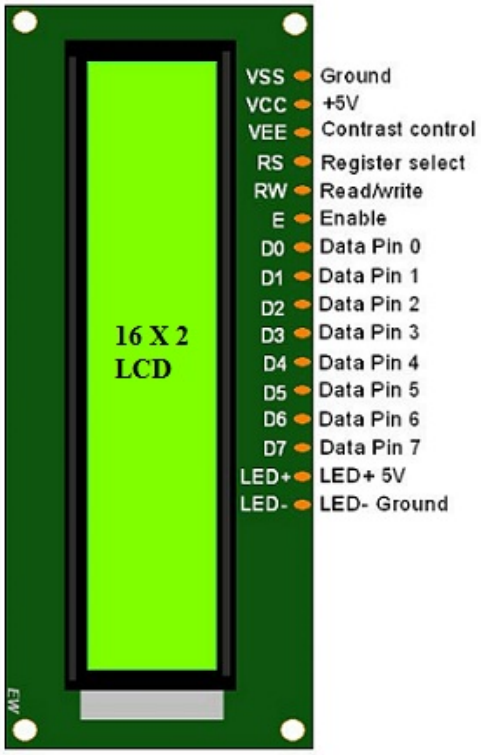

FIGURE 2.13 PIN CONFIGURATION OF LCD

2.14 FEATURES OF LCD

The features of this LCD mainly include the following.

- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit
- These are obtainable in Blue & Green Backlight
- It displays a few custom generated characters

2.15 LCD REGISTERS

A 16×2 LCD has two registers like data register and command register. The RS (register select) is mainly used to change from one register to another. When the register set is '0', then it is known as command register. Similarly, when the register set is '1', then it is known as data register.

Command Register

The main function of the command register is to store the instructions of command which are given to the display. So that predefined tasks can be performed such as clearing the display, initializing, set the cursor place, and display control. Here commands processing can occur within the register.

Data Register

The main function of the data register is to store the information which is to be exhibited on the LCD screen. Here, the ASCII value of the character is the information which is to be exhibited on the screen of LCD. Whenever we send the information to LCD, it transmits to the data register, and then the process will be starting there. When register set =1, then the data register will be selected.

2.16 16 X 2 LCD COMMANDS

The commands of LCD 16X2 include the following.

- For Hex Code-01, the LCD command will be the clear LCD screen
- For Hex Code-02, the LCD command will be returning home
- For Hex Code-04, the LCD command will be decrement cursor
- For Hex Code-06, the LCD command will be Increment cursor
- For Hex Code-05, the LCD command will be Shift display right
- For Hex Code-07, the LCD command will be Shift display left
- For Hex Code-08, the LCD command will be Display off, cursor off
- For Hex Code-0A, the LCD command will be cursor on and display off
- For Hex Code-0C, the LCD command will be cursor off, display on
- For Hex Code-0E, the LCD command will be cursor blinking, Display on
- For Hex Code-0F, the LCD command will be cursor blinking, Display on
- For Hex Code-10, the LCD command will be Shift cursor position to left
- For Hex Code-14, the LCD command will be Shift cursor position to the right
- For Hex Code-18, the LCD command will be Shift the entire display to the left
- For Hex Code-1C, the LCD command will be Shift the entire display to the right
- For Hex Code-80, the LCD command will be Force cursor to the beginning (1st line)
- For Hex Code-C0, the LCD command will be Force cursor to the beginning (2nd line)
- For Hex Code-38, the LCD command will be 2 lines and 5×7 matrix.

2.17 BUZZER

An audio signaling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

The pin configuration of the buzzer is shown below. It includes two pins namely positive and negative. The positive terminal of this is represented with the '+' symbol or a longer terminal. This terminal is powered through 6 Volts whereas the negative terminal is represented with the '-'symbol or short terminal and it is connected to the GND terminal.

FIGURE 2.17 BUZZER

CHAPTER 03

SOFTWARE SPECIFICATIONS

3.1 INTRODUCTION TO ARDUINO IDE

Arduino is a prototype platform (open-source) based on easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

3.2 KEY FEATURES OF ARDUINO IDE

Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.

You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).

Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.

Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.

Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload

our program on the Arduino board.

3.3 ARDUINO DATA TYPES

Data types in C refers to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in the storage and how the bit pattern stored is interpreted.

The following table provides all the data types that you will use During Arduino programming.

Void

The void keyword is used only in function declarations. It indicates that the function is expected to return no information to the function from which it was called.

Example:

```
Void Loop ( )
{
// rest of the code
}
```

Boolean

A Boolean holds one of two values, true or false. Each Boolean variable occupies one byte of memory.

Example:

```
Boolean state= false ; // declaration of variable with type boolean and
```

initialize it with false.

Boolean state = true ; // declaration of variable with type boolean and initialize it with false.

Char

A data type that takes up one byte of memory that stores a character value. Character literals are written in single quotes like this: 'A' and for multiple characters, strings use double quotes: "ABC".

However, characters are stored as numbers. You can see the specific encoding in the ASCII chart. This means that it is possible to do arithmetic operations on characters, in whichthe ASCII value of the character is used. For example, 'A' + 1 has the value 66, since theASCII value of the capital letter A is 65.

Example:

Char chr_a = ‘a’ ;//declaration of variable with type char and initialize it with character a.

Char chr_c = 97 ;//declaration of variable with type char and initialize it with character 97

Unsigned char

Unsigned char is an unsigned data type that occupies one byte of memory. The unsigned char data type encodes numbers from 0 to 255.

Example:

Unsigned Char chr_y = 121 ; // declaration of variable with type Unsigned char and initialize it with character y

Byte

A byte stores an 8-bit unsigned number, from 0 to 255.

Example:

byte m = 25 ;//declaration of variable with type byte and initialize it with 25

int

Integers are the primary data-type for number storage. int stores a 16-bit (2-byte) value. This yields a range of -32,768 to 32,767 (minimum value of -2^15 and a maximum value of (2^15) - 1).

The int size varies from board to board. On the Arduino Due, for example, an int stores a 32-bit (4-byte) value. This yields a range of -2,147,483,648 to 2,147,483,647 (minimum value of -2^31 and a maximum value of (2^31) - 1).

Example:

int counter = 32 ;// declaration of variable with type int and initialize it with 32.

Unsigned int

Unsigned int's (unsigned integers) are the same as int in the way that they store a 2 byte value. Instead of storing negative numbers, however, they only store positive values, yielding a useful range of 0 to 65,535 (2^16) - 1). The Due stores a 4 byte (32-bit) value, ranging from 0 to 4,294,967,295 (2^32 - 1).

Example:

Unsigned int counter= 60 ; // declaration of variable with type unsigned int and initialize it with 60.

Word

On the Uno and other ATMEGA based boards, a word stores a 16-bit unsigned number. On the Due and Zero, it stores a 32-bit unsigned number.

Example

word w = 1000 ;//declaration of variable with type word and initialize it with 1000.

Long Long variables are extended size variables for number storage, and store 32 bits (4 bytes), from 2,147,483,648 to 2,147,483,647.

Example:

Long velocity= 102346 ;//declaration of variable with type Long and initialize it with 102346

Unsigned long

Unsigned long variables are extended size variables for number storage and store 32 bits (4 bytes). Unlike standard longs, unsigned longs will not store negative numbers, making their range from 0 to 4,294,967,295 (2^32 - 1).

Example:

Unsigned Long velocity = 101006 ;// declaration of variable with type Unsigned Long and initialize it with 101006.

Short

A short is a 16-bit data-type. On all Arduinos (ATMega and ARM based), a short stores a 16-bit (2-byte) value. This yields a range of -32,768 to 32,767 (minimum value of -2^15 and a maximum value of (2^15) - 1).

Example:

short val= 13 ;//declaration of variable with type short and initialize it with 13

Float

Data type for floating-point number is a number that has a decimal point. Floating-point numbers are often used to approximate the analog and continuous values because they have greater resolution than integers.

Floating-point numbers can be as large as 3.4028235E+38 and as low as 3.4028235E+38. They are stored as 32 bits (4 bytes) of information.

Example:

float num = 1.352;//declaration of variable with type float and initialize it with 1.352.

Double

On the Uno and other ATMEGA based boards, Double precision floating-point number occupies four bytes. That is, the double implementation is exactly the same as the float, with no gain in precision. On the Arduino Due, doubles have 8-byte (64 bit) precisionExample:

double num = 45.352 ;// declaration of variable with type double and initialize it with 45.352.

3.4 STEPS TO UPLOAD THE PROGRAM IN ARDUINO BOARD

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1: First you must have your Arduino board (you can choose your favorite board) and a USB cable.

In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

FIGURE 3.4A USB CABLE

Step 2: Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.

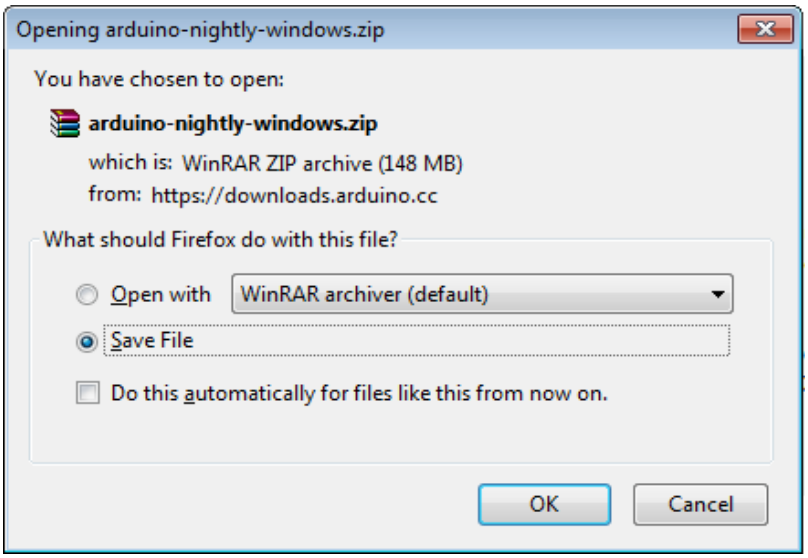

FIGURE 3.4B DOWNLOADING ARDUINO IDE

Step 3: Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a

small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4: Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Doubleclick the icon to start the IDE

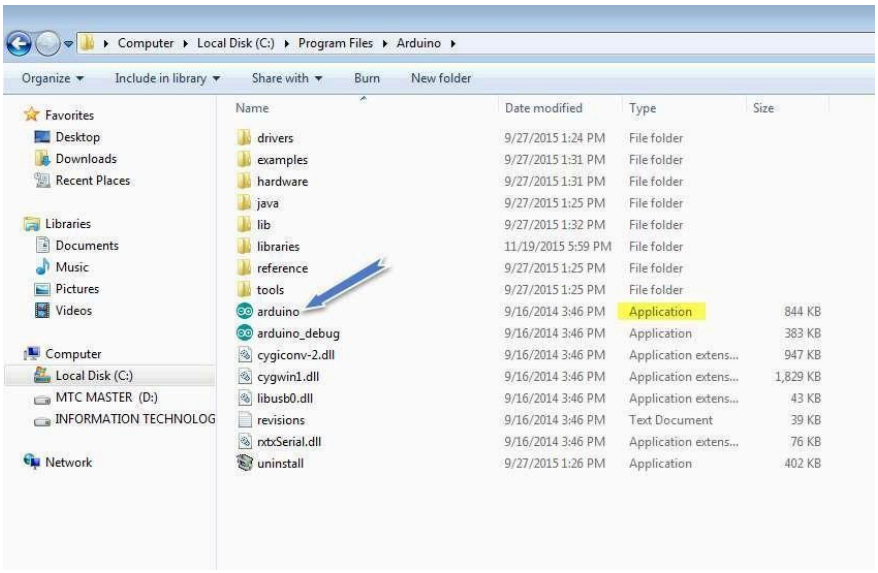

FIGURE 3.4C LAUNCHING OF ARDUINO IDE

Step 5: Open your first project.

Once the software starts, you have two options: Create a new project.

Open an existing project example.

Step 6: Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools -> Board and select your board

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using

Step 7: Select your serial port.

Select the serial device of the Arduino board. Go to Tools ->Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port

Step 8: Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Note: If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Note: If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

3.5 ARDUINO PROGRAMMING STRUCTURE

In this chapter, we will study in depth, the Arduino program structure and we will learn more new terminologies used in the Arduino world. The Arduino software is open- source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

Sketch: The first new terminology is the Arduino program called "sketch". Structure

Arduino programs can be divided in three main parts: Structure, Values (variables and constants), and Functions. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the Structure. Software structure consist of two main functions: Setup( ) function

Loop( ) function

```
Void setup ( )

{

}
```

PURPOSE:

The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

CHAPTER - 4

IMPLEMENTATION

4.1 SCHEMATIC DIAGRAM

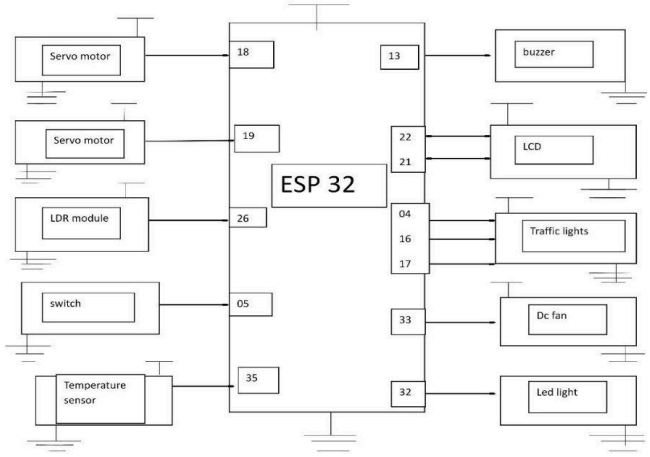

FIGURE 4.1 SCHEMATIC DIAGRAM OF PROPOSED SYSTEM

4.2 WORKING OF PROPOSED SYSTEM

The existing system is mainly working with the help of the code dumped in the arduino IDE software .When the train arrives to the station the station master knows that train is coming on to the platform so then he utilizes the IOT or he uses manual operation for opening the safety gates when the train arrives at the platform when the gates are opening it gives buzzer sound and shows the countdown for opening gates and when the train departures the station the station master will then closes the safety gates .The rotation or movement of safety gates will be done by servo motors.

4.3 FLOW CHART

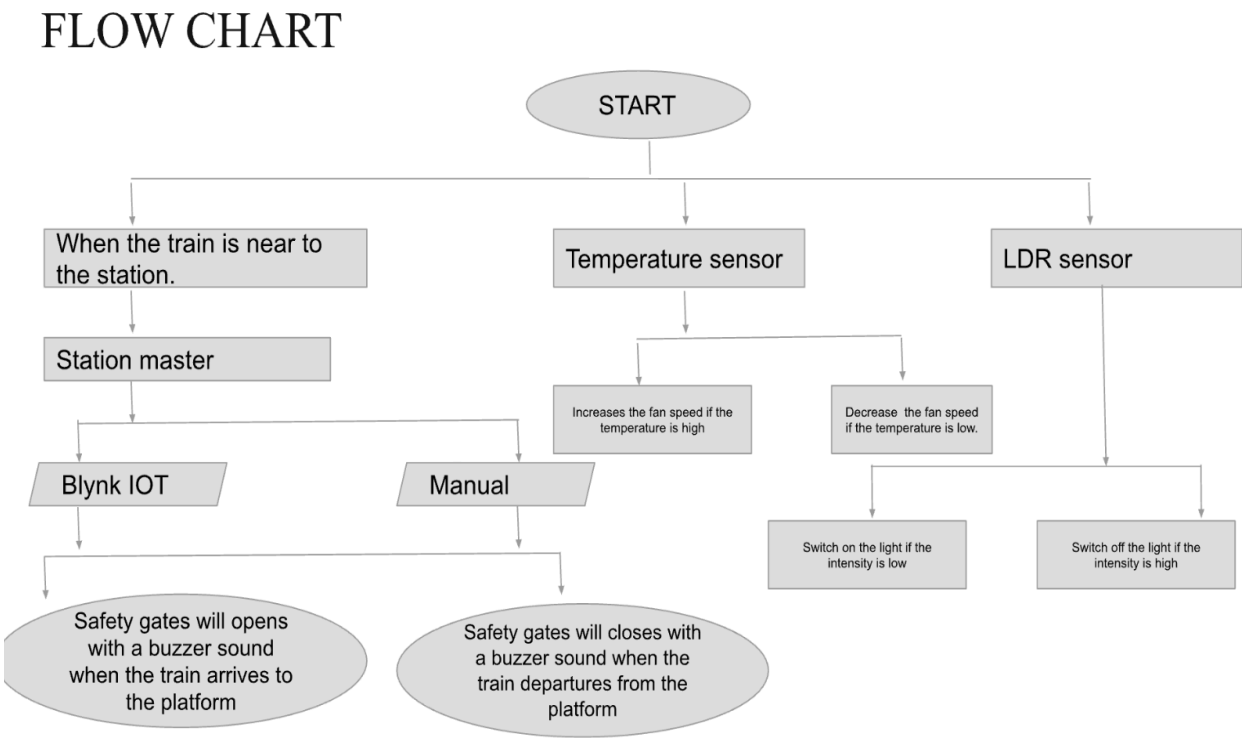

FIGURE 4.2 FLOWCHART OF PROPOSED SYSTEM

4.4 IMPLEMENTATION OF CODE

```
.#ifdef ESP32
  #include <Arduino.h>
  #include <WiFi.h>
  #include <AsyncTCP.h>
  #include <ESPAsyncWebServer.h>
  #include <AsyncElegantOTA.h>
#else
  #include <Arduino.h>
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
  #include <ESPAsyncWebServer.h>
  #include <AsyncElegantOTA.h>
#endif


AsyncWebServer server(80);


#define BLYNK_TEMPLATE_ID "TMPL3Kgt330bN"
#define BLYNK_TEMPLATE_NAME "RAILWAY PLATFORM SAFETY"
#define BLYNK_AUTH_TOKEN "8NMx9GZbYWrjR_6bnKXg33Es-TmCMN5n"
```

```
#define BLYNK_PRINT Serial

#ifdef ESP32

  #include <WiFi.h>

  #include <BlynkSimpleEsp32.h>

#else

  #include <ESP8266WiFi.h>

  #include <BlynkSimpleEsp8266.h>

#endif


char auth[] = "8NMx9GZbYWrjR_6bnKXg33Es-TmCMN5n";

char ssid[] = "projectb20";

char pass[] = "projectb20";


#include<LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);


#define sw 5

int sense;


int j=0;
```

```
#define buzzer 13


#define red_led 4

#define yellow_led 16

#define green_led 17


#define temp 35

int temp_sense;


#define ldr 26

int ldr_sense;


#define led 33


#define fan 32



#include<ESP32Servo.h>

Servo servo1;

Servo servo2;
```

```
int a=0;


int door=0;


WidgetLCD lcd1(V0);
WidgetLED indicator(V1);


void setup()
{
  Serial.begin(9600);


  pinMode(2,OUTPUT);digitalWrite(2,HIGH);


  pinMode(led,OUTPUT);digitalWrite(led,LOW);
  pinMode(fan,OUTPUT);digitalWrite(fan,LOW);


  pinMode(temp,INPUT);
  pinMode(ldr,INPUT);


  lcd.init();lcd.backlight();delay(50);
```

```
pinMode(red_led,OUTPUT);pinMode(yellow_led,OUTPUT);pinMode(green_led,OUTPUT);delay(50);

digitalWrite(red_led,HIGH);digitalWrite(yellow_led,LOW);digitalWrite(green_led,LOW);delay(50);

 pinMode(buzzer,OUTPUT);digitalWrite(buzzer,LOW);delay(50);

 pinMode(sw,INPUT_PULLUP);delay(50);

 lcd.clear();
 lcd.print("INITIATING SERVO");
 delay(100);

 servo1.attach(19);delay(50);
 servo.attach(18);delay(50);

 servo1.write(120);delay(100);
 servo2.write(120);delay(100);
 delay(1000);
```

```
lcd.clear();
lcd.print("CONNECTING TO...");
lcd.setCursor(0,1);
lcd.print(ssid);
delay(1000);


WiFi.begin(ssid, pass);
Serial.print("Connecting to WiFi ..");
while(WiFi.status() != WL_CONNECTED)
{
  Serial.print('.');
  delay(500);
}


Blynk.begin(auth,ssid,pass);
Serial.println("READY");


lcd.clear();lcd1.clear();
lcd.print(" WIFI CONNECTED ");lcd1.print(0,0," WIFI CONNECTED ");
lcd.setCursor(0,1);
lcd.print(WiFi.localIP());lcd1.print(0,1,WiFi.localIP());
```

```
  delay(5000);
  lcd.clear();lcd1.clear();


  lcd.clear();
  lcd.print(" RAIL  PLATFORM ");
  lcd.setCursor(0,1);
  lcd.print(" SAFETY  SYSTEM ");
  delay(2000);
  lcd.clear();
  lcd.print("  *  READY  *  ");
  delay(2000);


  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
    request->send(200, "text/plain", "Hi! I am ESP32.");
  });


  AsyncElegantOTA.begin(&server);    // Start ElegantOTA
  server.begin();
  Serial.println("HTTP server started");
}
BLYNK_CONNECTED()
```

```
{
 Blynk.syncVirtual(V2);
}
BLYNK_WRITE(V2)
{
 int door=param.asInt();
 if(door==1)
 {
  j=j+1;
  if(j>=4)
  {
   j=0;
  }
 }
}
void sensor_data()
{
 ldr_sense=digitalRead(ldr);
 Serial.print("LDR : ");Serial.println(ldr_sense);

 temp_sense=analogRead(temp);
```

```
temp_sense=temp_sense*0.080;
Serial.print("TEMP : ");Serial.println(temp_sense);

Blynk.virtualWrite(V3,temp_sense);

if(ldr_sense==1)
{
  digitalWrite(led,HIGH);
}
else
{
  digitalWrite(led,LOW);
}
if(temp_sense >= 30)
{
  digitalWrite(fan,HIGH);
}
else
{
  digitalWrite(fan,LOW);
}
```

```
}


void loop()
{
  sensor_data();


  sense=digitalRead(sw);
  sense=1-sense;
  Serial.print("SW : ");Serial.println(sense);


  lcd.clear();
  lcd.print(" * WELCOME TO * ");
  lcd.setCursor(0,1);
  lcd.print(" INDIAN RAILWAY ");

digitalWrite(2,HIGH);indicator.on();delay(300);digitalWrite(2,LOW);indicator.off();dela
y(300);


  if(sense==1)
  {
    j=j+1;
    if(j>=4)
```

```
  {
    j=0;
  }
}

if(j==1)
{
  j=2;
  digitalWrite(red_led,LOW);

    lcd.clear();
    for(int i=5; i>=0; i--)
    {
      digitalWrite(yellow_led,HIGH);
      digitalWrite(buzzer,HIGH);
      lcd.clear();
      lcd.print("OPEN IN : ");lcd.print(i);
      delay(500);
      digitalWrite(buzzer,LOW);
      digitalWrite(yellow_led,LOW);
      delay(500);
```

```
      lcd.clear();
      lcd.print(" HAPPY  JOURNEY ");
    }
    for(int s=120;s>=0;s=s-10)
    {
      Serial.println(s);
      servo1.write(s);delay(20);
      servo2.write(s);delay(20);
      delay(500);
    }

    digitalWrite(yellow_led,HIGH);
    digitalWrite(green_led,HIGH);
}
else if(j==3)
{
  j=0;
  digitalWrite(red_led,HIGH);

    lcd.clear();
    for(int i=5; i>0; i--)
```

```
    {
      digitalWrite(buzzer,HIGH);
      lcd.clear();
      lcd.print("CLOSE IN : ");lcd.print(i);
      delay(500);
      digitalWrite(buzzer,LOW);
      delay(500);
    }
    for(int s=0;s<=120;s=s+10)
    {
      Serial.println(s);
      servo1.write(s);delay(20);
      servo2.write(s);delay(20);
      delay(500);
    }
}
else
{
  pass;
}
```

```
digitalWrite(2,HIGH);indicator.on();delay(300);digitalWrite(2,LOW);indicator.off();dela
y(300);

  Blynk.run();

}
```

CHAPTER 5

OUTPUT SCREENSHOTS

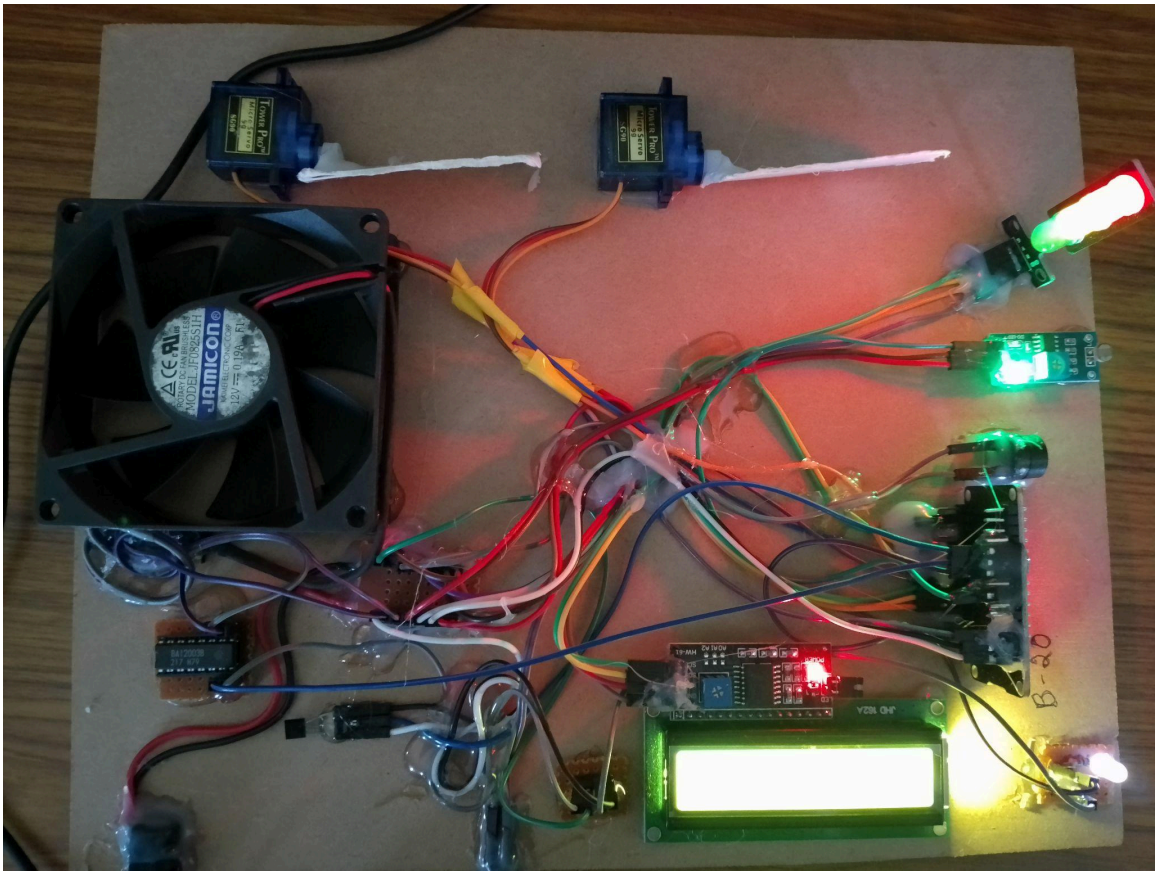

FIGURE 5.1 KIT OF PROPOSEDSYSTEM

FIGURE 5.2 LCD SHOWING COUNTDOWN OF CLOSING OF SAFETY GATES

FIGURE 5.3 LCD SHOWING HAPPY JOURNEY AFTER CLOSING OF GATES

CHAPTER 06

CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

The proposed model successfully made human safety possible in areas like railway crossing nearby rural as well as urban areas. The servo motor and senor like IR and Ultrasonic sensor processes combinable to make a system where an object/vehicle or train itself is sensed if it passes the gateway and corresponding actions are taken by motor to open or close the gateway, also the buzzer is used for warning the nearby area about the arrival of train which decreases the rate of accidents nearby the railway crossings. The sensors are installed before a genuine range from the gateway so that warning time for arrival of train or its departure is enough to take the corresponding decision and also in that much time the closing/opening of gateway through use of servo motor is done accordingly.

6.2 FUTURE SCOPE

In future the existing system can be implemented as an automated safety gates by using the Artificial Intelligence platform with IR sensors which detects the distance of the train from the platform.

REFERENCES

- http://www.indianrailways.gov.in/railwayboard/uploads/directorate/sign al/downloads/leve-crossing.pdf.
- http://www.indianrailways.gov.in/railwayboard/uploads/codesmanual/SEM.II/SignalEngineering%20ManualIICh14_data.htm .
- http://164.100.47.134/intranet/Indianrailway.pdf.
- http://www.mind.ilstu.edu/curriculum/medical_robotics/dcmotor.jpg.
- http://www.robosoftsystems.co.in/roboshop/media/catalog/product/pdf/I R_single.pdf.
- https://elecrom.wordpress.com/2008/02/19/how-to-make-simpleinfrared-sensor-modules/.
- http://en.wikipedia.org/wiki/Classification_of_railway_accidents.
- http://www.slideshare.net/AtchyuthSonti/automatic-railway-gatecontrol-12526197.