A Project Report

on

IOT ESTABLISHED AGILE PARKING SYSTEM

Submitted to

Jawaharlal Nehru Technological University, Kakinada

In the Partial fulfillment of the requirement for the Award of the degree of

BACHELOR OF TECHNOLOGY

In

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By

J.MAHINDRA	**(198H1A0421)**
A.AKANKSHA	**(198H1A0401)**
M.VENKATA SAI KRISHNA	**(198H1A0441)**
K. SUDHEER	**(198H1A0427)**
M.PREETHI	**(198H1A0442)**

Under Esteemed Guidance Of

Mrs. O. PRIYANKA

Assistant Professor

COLLEGE OF ENGINEERING & TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University, Kakinada

Approved by A.I.C.T.E, Recognized by Govt. of AP)

Paritala, Kanchikacherla Mandal, Vijayawada -521180

2022-2023

MVR
COLLEGE OF ENGINEERING & TECHNOLOGY

Approved by A.I.C.T.E, New Delhi, Affiliated to JNTUK, Kakinada

Accredited By NBA(CSE), Accredited by NAAC with 'A' Grade, ISO:9001-2008

Paritala, Kanchikacherla Mandal, Vijayawada -521180

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"IOT ESTABLISHED AGILE PARKING SYSTEM"** *is a bonafide record of the major project done by* **J. MAHINDRA (198H1A0421), A. AKANKSHA (198H1A0401),M.VENKATA SAI KRISHNA (198H1A0441),K. SUDHEER (198H1A0427),M PREETHI(198H1A0442)** *Under my supervision and guidance, in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Electronics & Communication Engineering from Jawaharlal Nehru Technological University,Kakinada for the year of 2022*

Mrs. O. PRIYANKA

(Project Guide) **(Head of the Department)**

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

The present project work is the several days study of the various aspects of the project development. During this effort in the present study, we have received a great amount of help from our secretary and correspondent **Mr. M. SRINIVASA BABU** which we wish toacknowledge and thank from depth of our hearts

We are thankful to our principal **Dr.U YEDUKONDALU** for permitting and encouraging us in doing this project.

We are deeply intended to **Mr. K. SATHYANARAYANA, M. Tech, Head of the department, E.C.E**, whose motivation and contrast encouragement has led to pursue a project in the field of embedded electronics.

We are very much obliged and thankful to our internal guide **Mrs.O. PRIYANKA,** assistant professor for providing this opportunity and contrast encouragement given by them during the course. We are grateful to them valuable guidance and suggestions during my project work.

Our parents have put ourselves ahead of themselves, Because of their hard work and dedication; we had opportunities beyond our wildest dreams. Our heartfelt thanks to them for giving us all we ever needed to be successful student and individual.

Finally, we express our thanks to all our other professors, Classmates, friends, neighbours, and our family members who helped us for the completion of our project and without infinite love and patience this would never have been possible.

J.MAHINDRA	**(198H1A0401)**
A. AKANKSHA	**(198H1A0421)**
M. VENKATA SAIKRISHNA	**(198H1A0441)**
K. SUDHEER	**(198H1A0427)**
M. PREETHI	**(198H1A0442)**

DECLARATION

We are the members of the project **"IOT ESTABLISHED AGILE PARKING SYSTEM"** hereby declare that the matter embodied and this project is the genuine work done by us and has not been submitted either to this university or to any to any other university/institute for the fulfilment of the requirement of any other course of study.

J.MAHINDRA	**(198H1A0421)**
A. AKANKSHA	**(198H1A0401)**
M. VENKATA SAIKRISHNA	**(198H1A0441)**
K.SUDHEER	**(198H1A0427)**
M. PREETHI	**(198H1A0442)**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The proposed project is a smart car parking system that utilizes IR sensors, RFID, servo motor, LCD, WiFi, Firebase, and IoT cloud technology to automate the process of parking monitoring and billing. The system aims to eliminate the need for manual ticketing and parking management, making the process more efficient, convenient, and secure.

The system uses IR sensors to detect the presence of vehicles in the parking lot, and RFID technology to identify the vehicle and the owner. The system then assigns a parking spot to the vehicle and displays the information on an LCD screen. A servo motor is used to open and close the parking gate to allow the vehicle to enter or exit the parking lot.

The system is connected to a WiFi network and communicates with a Firebase server through an IoT cloud platform. The server stores the parking information and calculates the parking time and fee based on the parking duration and the vehicle type. The system then displays the parking fee on the LCD screen and accepts payment through a payment gateway.

The proposed system offers several advantages, including the elimination of manual ticketing, reduced waiting time for parking, automated billing and payment, improved security and convenience, and real-time parking monitoring. The system can be implemented in various settings, such as shopping malls, airports, hospitals, and office buildings, to provide efficient and reliable parking services

CHAPTER-1

1.1 INTRODUCTION

With the rapid growth of urbanization and increasing number of vehicles, parking has become a significant issue in urban areas. The conventional parking systems are time-consuming, inefficient, and prone to errors, leading to congestion, frustration, and loss of revenue. To overcome these challenges, smart parking systems have emerged as a promising solution, leveraging advanced technologies to automate the parking process, monitor the parking lot, and optimize the parking space utilization. Among various types of smart parking systems, the IoT-based car parking monitoring and billing system has gained significant attention due to its real-time monitoring and control capabilities, remote access, and seamless integration with payment systems.

The proposed system utilizes a combination of IR sensors, RFID, servo motor, LCD, WiFi, Firebase, and IoT cloud technology to provide a comprehensive solution for parking monitoring and billing. The system aims to provide a seamless parking experience for the users by automating the parking process, eliminating manual ticketing, and providing real-time parking information and billing.

1.2 PROBLEM STATEMENT:

The conventional parking systems are time-consuming, inefficient, and prone to errors, leading to congestion, frustration, and loss of revenue. The manual ticketing and billing process is often inaccurate and unreliable, leading to disputes and discrepancies. Moreover, the lack of real-time monitoring and control makes it difficult to manage the parking lot efficiently and optimize the parking space utilization. The existing automated parking systems are often expensive, complex, and require specialized infrastructure, making them difficult to implement and maintain. Therefore, there is a need for a cost-effective, efficient, and reliable parking system that can automate the parking process, monitor the parking lot, and optimize the parking space utilization. The proposed IoT-based car parking monitoring and billing system aims to address these challenges and provide a seamless parking experience for the users.

1.3 LITERATURE SURVEY:

The existing literature on automated car parking systems reveals several approaches and techniques used to design and implement efficient and reliable parking systems. The use of IoT, RFID, and other wireless technologies has enabled the development of intelligent parking systems that can automate the parking process, monitor the parking lot, and optimize the parking space utilization.

One of the studies by Huang et al. (2018) proposed an IoT-based smart parking system that uses RFID and wireless sensor networks to manage the parking lot efficiently. The system comprises three main components: a parking space detection module, a billing module, and a user interface module. The parking space detection module uses RFID technology to detect the presence of vehicles in the parking lot and sends the data to the billing module, which calculates the parking fee based on the parking duration. The user interface module allows the users to check the parking availability, reserve the parking space, and pay the parking fee through a mobile app or a website. The results showed that the proposed system can reduce the parking time, enhance the user experience, and increase the parking revenue.

Another study by Alam et al. (2019) proposed an IoT-based smart parking system that uses infrared sensors and cloud computing to manage the parking lot. The system comprises three main components: a sensor module, a cloud server, and a mobile app. The sensor module uses infrared sensors to detect the presence of vehicles in the parking lot and sends the data to the cloud server, which processes the data and generates the parking fee. The mobile app allows the users to check the parking availability, reserve the parking space, and pay the parking fee. The results showed that the proposed system can reduce the parking time, improve the parking space utilization, and enhance the user experience.

Similarly, a study by Wang et al. (2017) proposed an IoT-based parking system that uses image processing and cloud computing to manage the parking lot. The system comprises three main components: a camera module, a cloud server, and a mobile app. The camera module captures the image of the vehicle and sends it to the cloud server, which processes the image and generates the parking fee. The mobile app allows the users to check the parking availability, reserve the parking space, and pay the parking fee. The results showed that the proposed system can reduce the parking time, increase the parking revenue, and improve the parking space utilization.

Overall, the existing literature suggests that the use of IoT, RFID, and other wireless technologies can enable the development of efficient and reliable parking systems that can automate the parking process, monitor the parking lot, and optimize the parking space utilization. However, the implementation of such systems requires careful consideration of the infrastructure, cost, security, and user experience.

Another study by Mohanty et al. (2019) proposed an IOT based smart parking system that uses ultrasonic sensors and machine learning algorithms to manage the parking lot. The system comprises three main components: a sensor module, a cloud server, and a mobile app. The sensor module uses ultrasonic sensors to detect the presence of vehicles in the parking lot and sends the data to the cloud server, which processes the data and generates the parking fee. The machine learning algorithms are used to predict the parking availability and optimize the parking space utilization. The mobile app allows the users to check the parking availability, reserve the parking space, and pay the parking fee. The results showed that the proposed system can reduce the parking time, increase the parking revenue, and improve the parking space utilization.

In another study, Singh et al. (2020) proposed an IoT-based smart parking system that uses computer vision and deep learning algorithms to manage the parking lot. The system comprises three main components: a camera module, a cloud server, and a mobile app. The camera module captures the image of the vehicle and sends it to the cloud server, which processes the image and generates the parking fee. The deep learning algorithms are used to detect the license plate and recognize the vehicle type. The mobile app allows the users to check the parking availability, reserve the parking space, and pay the parking fee. The results showed that the proposed system can reduce the parking time, enhance the security, and improve the user experience.

Finally, a study by Han et al. (2018) proposed an IoT-based smart parking system that uses LoRaWAN and cloud computing to manage the parking lot. The system comprises three main components: a sensor module, a gateway module, and a cloud server. The sensor module uses LoRaWAN technology to detect the presence of vehicles in the parking lot and sends the data to the gateway module, which forwards the data to the cloud server. The cloud server processes the data and generates the parking fee. The results showed that the proposed system can reduce the communication cost, enhance the scalability, and improve the reliability.

In conclusion, the literature survey suggests that IoT-based smart parking systems have the potential to revolutionize the traditional parking systems and offer several advantages such as efficient parking management, real-time monitoring, and optimization of parking space utilization. The integration of various wireless technologies such as RFID, infrared sensors, ultrasonic sensors, computer vision, and LoRaWAN can enable the development of customized and cost-effective parking solutions that cater to the diverse needs of the users. However, the implementation of such systems requires careful consideration of the infrastructure, cost, security, and user experience.

1.4 Existing system:

The existing parking systems usually rely on manual methods, such as parking attendants, tickets, and barriers, which are often inefficient, time-consuming, and prone to errors. In addition, the traditional parking systems cannot provide real-time information about the parking availability and occupancy, which often leads to frustration and wasted time for the drivers. Moreover, the traditional parking systems do not have the capacity to optimize the parking space utilization and revenue generation.

Some parking systems have attempted to incorporate technology to address some of these issues, but they still fall short of the capabilities of IoT-based smart parking systems. For instance, some parking systems use sensors to detect the presence of vehicles and display the parking availability on LED screens. However, these systems are limited in their scope and cannot offer real-time monitoring, optimization, or personalized services.

Furthermore, some parking systems use mobile apps or web interfaces to allow the users to reserve and pay for parking spots. While these systems offer some convenience to the users, they still rely on manual methods to detect the parking occupancy and cannot offer optimization or automation.

Therefore, there is a need for a more advanced and comprehensive parking system that can leverage the potential of IoT technologies to offer real-time monitoring, optimization, and automation of parking management.

CHAPTER-2

2.1 PROPOSED SYSTEM

The proposed IoT-based smart parking system consists of several components that work together to provide real-time monitoring, optimization, and automation of parking management. The functionality of each component is described below:

IR Sensors: The IR sensors are used to detect the presence and absence of vehicles in the parking space. These sensors are installed at each parking spot and connected to the microcontroller.

RFID: RFID tags are used to uniquely identify each vehicle entering the parking space. These tags are attached to the windshield of the vehicle and are read by the RFID reader located at the entrance.

Servo Motor: The servo motor is used to control the barrier at the entrance and exit of the parking lot. It is connected to the microcontroller and is programmed to open and close the barrier when the RFID tag is read.

LCD: The LCD display is used to display real-time information about the parking availability and occupancy. It is connected to the microcontroller and updates in real-time based on the sensor data.

WiFi: WiFi module is used to connect the microcontroller to the internet and transmit the data to the cloud server.

Firebase: Firebase is a cloud database and authentication service that is used to store the data collected from the sensors and display it in a user-friendly format on the mobile app.

IoT Cloud: The IoT cloud platform is used to collect, store, and analyze the data collected from the sensors. It is used to monitor the parking occupancy, optimize the parking space utilization, and generate reports and alerts.

Mobile App: The mobile app is used to provide a user-friendly interface for the drivers to find and reserve parking spots. It shows real-time information about the parking availability and occupancy, and allows the users to reserve and pay for parking spots.

Overall, these components work together to provide a seamless and efficient parking management system that leverages the potential of IoT technologies to optimize the parking space utilization and provide personalized services to the users.

2.2 BLOCK DIAGRAM:

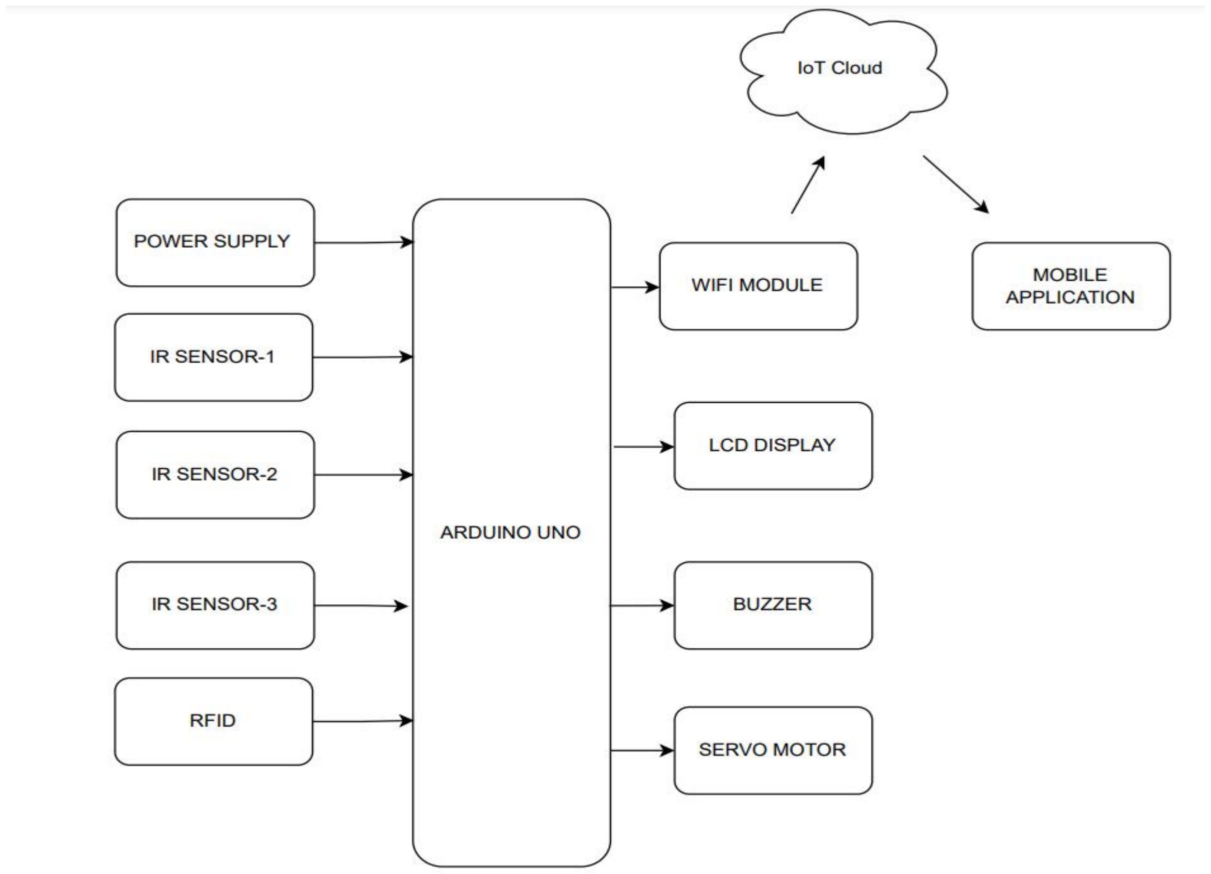


Fig:2.1 Block diagram of Agile parking system

2.2 .1 Working Flow:

The proposed IoT-based smart parking system consists of several components that work together to provide real-time monitoring, optimization, and automation of parking management. The functionality of each component is described below:

IR Sensors: The IR sensors are used to detect the presence and absence of vehicles in the parking space. These sensors are installed at each parking spot and connected to the microcontroller.

RFID: RFID tags are used to uniquely identify each vehicle entering the parking space. These tags are attached to the windshield of the vehicle and are read by the RFID reader located at the entrance.

Servo Motor: The servo motor is used to control the barrier at the entrance and exit of the parking lot. It is connected to the microcontroller and is programmed to open and close the barrier when the RFID tag is read.

LCD: The LCD display is used to display real-time information about the parking availability and occupancy. It is connected to the microcontroller and updates in real-time based on the sensor data.

WiFi: WiFi module is used to connect the microcontroller to the internet and transmit the data to the cloud server.

Firebase: Firebase is a cloud database and authentication service that is used to store the data collected from the sensors and display it in a user-friendly format on the mobile app.

IoT Cloud: The IoT cloud platform is used to collect, store, and analyze the data collected from the sensors. It is used to monitor the parking occupancy, optimize the parking space utilization, and generate reports and alerts.

Mobile App: The mobile app is used to provide a user-friendly interface for the drivers to find and reserve parking spots. It shows real-time information about the parking availability and occupancy, and allows the users to reserve and pay for parking spots.

Overall, these components work together to provide a seamless and efficient parking management system that leverages the potential of IoT technologies to optimize the parking space utilization and provide personalized services to the users.

CHAPTER-3

3.1 HARDWARE DESCRIPTION

3.1.1 Arduino

Arduino is open source physical processing which is base on a microcontroller board and an incorporated development environment for the board to be programmed. Arduino gains a few inputs, for example, switches or sensors and control a few multiple outputs, for example, lights, engine and others. Arduino program can run on Windows, Macintosh and Linux operating systems (OS) opposite to most microcontrollers' frameworks which run only on Windows. Arduino programming is easy to learn and apply to beginners and amateurs. Arduino is an instrument used to build a better version of a computer which can control, interact and sense more than a normal desktop computer. It's an open-source physical processing stage focused around a straightforward microcontroller board, and an environment for composing programs for the board. Arduino can be utilized to create interactive items, taking inputs from a diverse collection of switches or sensors, and controlling an assortment of lights, engines, and other physical outputs. Arduino activities can be remaining solitary, or they can be associated with programs running on your machine (e.g. Flash, Processing and Maxmsp.) The board can be amassed by hand or bought pre assembled; the open-source IDE can be downloaded free of charge. Focused around the Processing media programming environment, the Arduino programming language is an execution of Wiring, a comparative physical computing platform.

Fig 3.1- Arduino symbol

3.2 Choosing Arduino

There are numerous different microcontrollers and microcontroller platforms accessible for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and numerous others offer comparative usefulness. These apparatuses take the chaotic subtle elements of microcontroller programming and wrap it up in a simple to-utilize bundle. Arduino additionally rearranges the methodology of working with microcontrollers; moreover it offers some advantages for instructors, students, and intrigued individuals:

• Inexpensive - Arduino boards are moderately cheap compared with other microcontroller boards. The cheapest version of the Arduino module can be amassed by hand, and even the preassembled Arduino modules cost short of what $50.

• Cross-platform - The Arduino programming runs multiple operating systems Windows, Macintosh OSX, and Linux working frameworks. So we conclude that Arduino has an advantage as most microcontroller frameworks are constrained to Windows.

• Straightforward, clear programming method - The Arduino programming environment is easy to use for novices, yet sufficiently versatile for cutting edge customers to adventure as well. For educators, its favorably engaged around the Processing programming environment, so
understudies finding ways to understand how to program in that environment will be familiar
with the nature of arduino.

• Open source and extensible programming. The Arduino program language is available as open source, available for development by experienced engineers. The lingo can be reached out through C++ libraries, and people expecting to understand the specific purposes of different interests can make the leap from Arduino to the AVR C programming language on which it is based. Basically, you can incorporate AVR-C code clearly into your Arduino programs if you have to.

• Open source and extensible hardware - The Arduino is concentrated around Atmel's Atmega8 and Atmega168 microcontrollers. The plans for the modules are circulated under a Creative Commons license, so experienced circuit designers can make their own particular interpretation of the module, extending it and improving it. slightly inexperienced customers can build the breadboard variation of the module remembering the finished objective to perceive how it capacities and save money.

3.3 ARDUINO UNO:

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform;

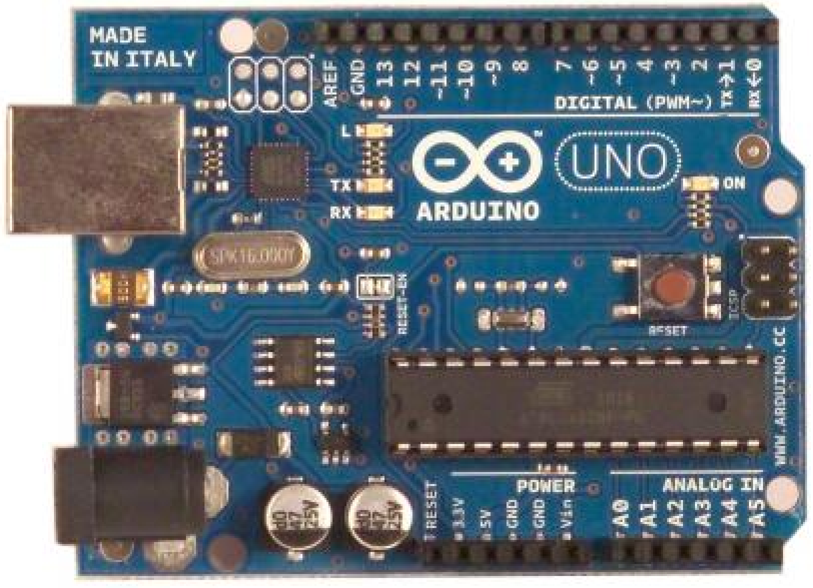


Fig 3.2- Arduino board

3.3.1 Technical specifications of arduino:

Microcontroller: ATmega328

Operating Voltage: 5V

Input Voltage (recommended): 7-12V

Input Voltage (limits): 6-20V

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

Analog Input Pins 6

DC Current per I/O Pin 40 mA

DC Current for 3.3V Pin 50 mA

Flash Memory

32 KB of which 0.5 KB used by

bootloader

SRAM 2 KB

EEPROM 1 KB

Clock Speed 16 MHz

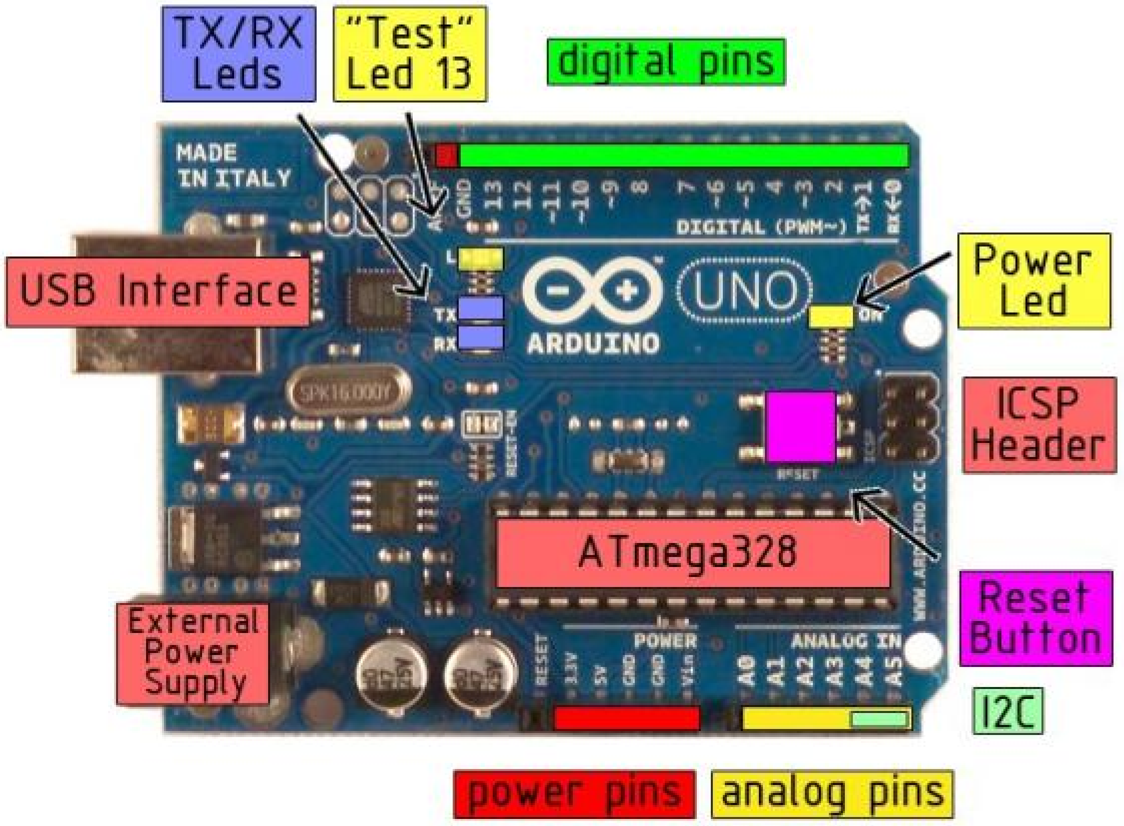


Fig .3.3-Arduino UNO pin description

3.3.2 POWER

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

• **VIN**. 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.

• **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**. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

• **GND**. Ground pins.

3.3.3 MEMORY:

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

3.3.4 INPUT/OUTPUT

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

• **Serial:** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .

• **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 attachInterrupt() function for details.

• **PWM:** 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

• **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. 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. The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference() function. Additionally, some pins have specialized functionality:

• **I2C:** 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library. There are a couple of other pins on the board:

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

• **Reset:** Bring this line LOW to reset the micro controller. Typically used to add a reset button to shields which block the one on the board.

3.3.5 COMMUNICATION:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-toserial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

3.3.6 Programming

The Arduino Uno can be programmed with the Arduino software. The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C headerfiles). You can also bypass the bootloader and program the microcontroller through the ICSP (InCircuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by: On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. 10 On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

3.3.7 Automatic(software) Reset:

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of theATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line. 11

3.3.8 USB Overcurrent Protection:

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

3.4 Physical Characteristics:

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

3.4.1 IR(Infrared) Proximity Sensor:

An **IR Sensor** is an electronic device that performs **IR signal** scan in the specific frequency range defined by the standards and converts it to electric signals on its **digital output** pin (usually as a signal PIN as OUT PIN). Proximity sensors are used in touchscreen phones, in other devices. Display is disabled during the call so that even if the cheek is in touch with the touchscreen, it will not affect it.

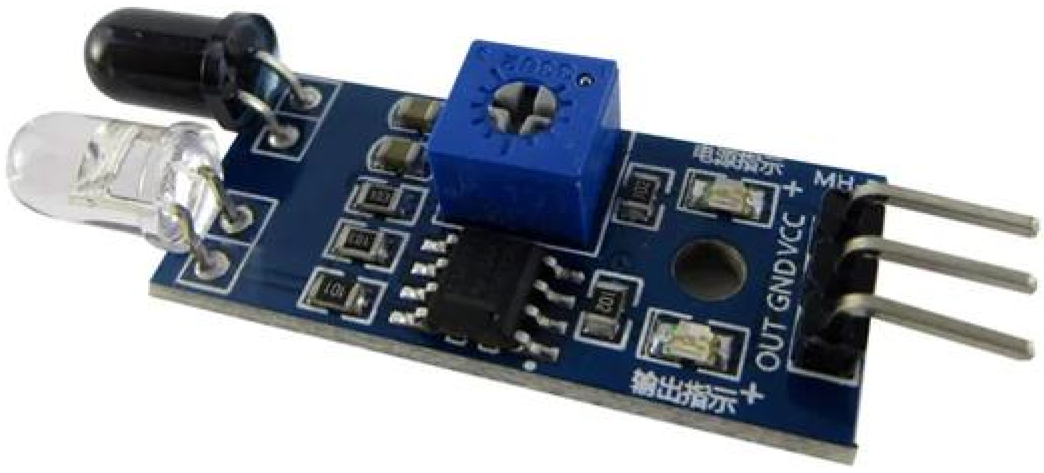


Fig 3.4-IR sensor

This Easy Electronics' multipurpose infrared sensor(IR Proximity Sensor) can also be used as **Barrier sensor, Line Sensing Robotics and Encoding sensor**. When an object is placed in front of the sensor with a logical zero (0V) output, it provides a digital output of **1 or 0**. Whether the device is getting enough power with inbuilt LED indicators and start experimenting with your logic.

3.5 Other important references:

Introduction to the arduino.

Arduino software installation.

Temperature sensor and types.

How to interface LM35 sensor with Arduino Uno

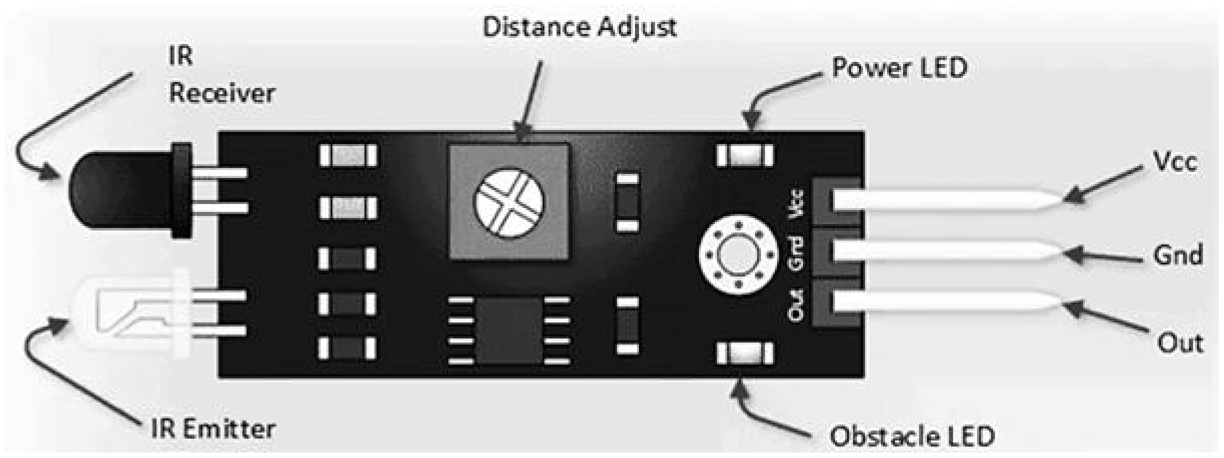


FIg 3.5- IR sensor description

3.5.1 Types of proximity sensor

Inductive Proximity Sensors: Inactive proximity sensors are contactless sensors used only to detect metal objects. This is based on the entry rule, once a metal object comes to him, the coil with the oscillator runs.

Capacitive Proximity Sensors: Capacitive proximity sensors are contactless sensors that detect both metallic and non-metallic objects, including liquid, powders, and granular. It operates by detecting a change in capacitance.

Ultrasonic Proximity Sensors: The third ultrasonic sensor in this list is the ultrasonic sensors that emit the high frequency ultrasonic range to detect the existence of objects. It works with power. Like capacitive sensors.

IR Proximity Sensor: IR, in short for infrared, detects the presence of an object by emitting a beam of infrared light. It works similarly to ultrasonic sensors, though instead of using sonic waves, IR is transmitted.

Photoelectric Proximity Sensor: Photoelectric proximity sensors are ones that use high-end photoelectric technology, it emits a light beam that's capable of detecting all sorts of objects! It has the following 3 different models; Reflective, Through-beam, and Retro-reflective. Each model offers varying light emission methods, though they are all highly efficient when it comes to distance detection.
.

LiDAR Proximity Sensor: LiDAR, in short for Light Detection and Ranging, is a higher-end sensing technology that provides excellent max detection range with fast update rates

3.6 Working Principle of IR Proximity Sensor :

The emitter is an **IR LED** and the detector is an IR photodiode. The **IR photodiode** is sensitive to the IR light emitted by an IR LED. The photo-diode's resistance and output voltage change in proportion to the IR light. .

Proximity sensors employ reflective indirect incidence principle. The photodiode receives the radiation emitted by the IR LED once reflected back by the object. Closer the object, higher will be the intensity of the incident radiation on the photodiode. This intensity is converted to voltage to determine the distance.

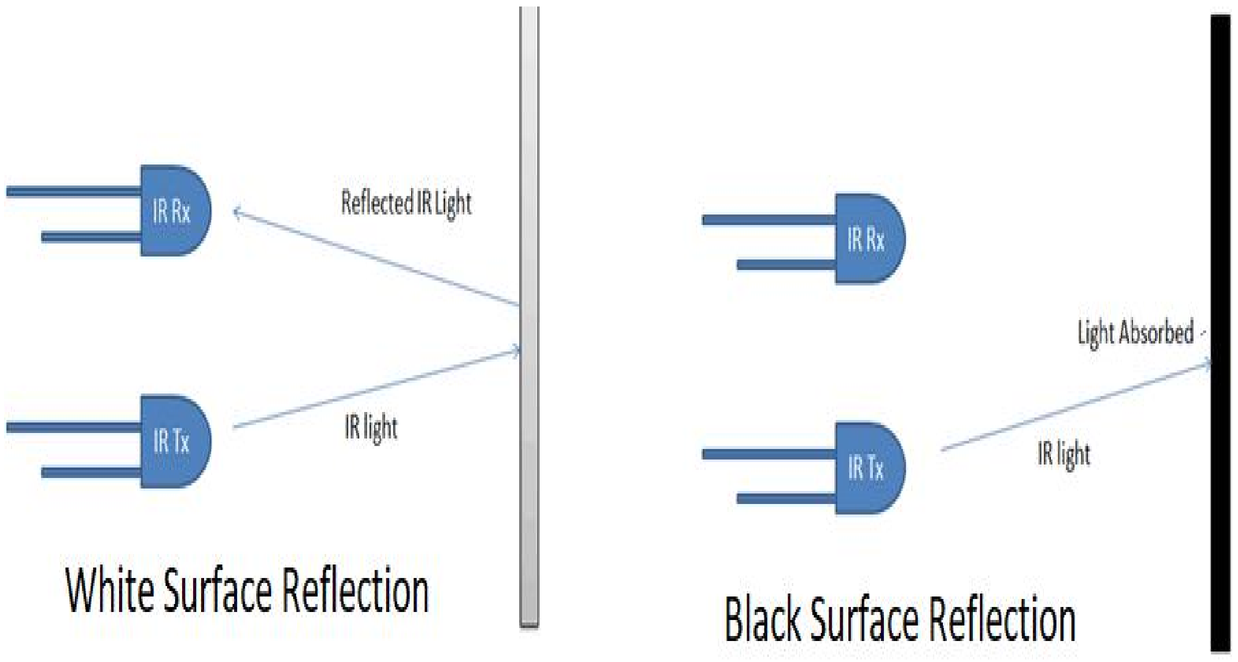
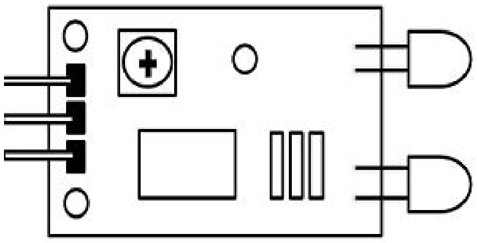


Fig.3.5.1-IR Sensor

This proximity sensor is used to detect objects and obstacles in front of the sensor. The sensor keeps transmitting infrared light and is known by sensors monitoring the reflective light of the object when an object approach. To avoid obstacles, RPM can be contacted with less tachometer for rotation objects such as automatic doors, parking aid devices or security alarm systems or fan blades.

3.6.1

PinConfiguration:.



Pin Name	Description
VCC	Power Supply Input
GND	Power Supply Ground
OUT	Active High Output

3.6.2 RC522 RFID Module

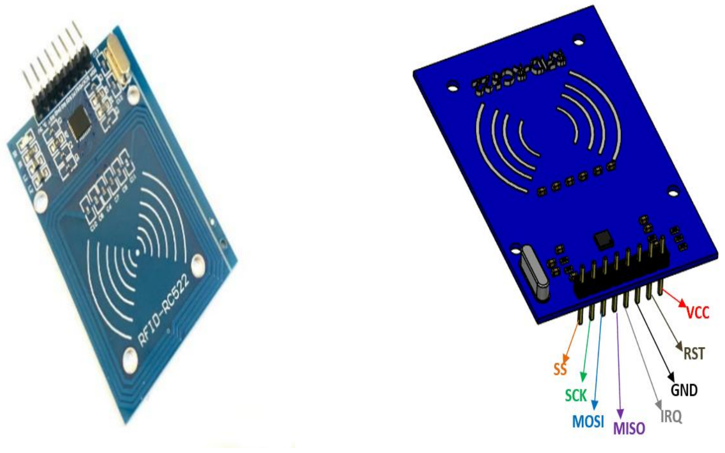


Fig 3.6- RFID reader

The **RC522** is a **13.56MHz RFID module** that is based on the **MFRC522 controller from NXP semiconductors**. The module can supports I2C, SPI and UART and normally is shipped with a RFID card and key fob. It is commonly used in attendance systems and other person/object identification applications.

3.6.3 RC522 Pin Configuration

Pin Number	Pin Name	Description
1	Vcc	Used to Power the module, typically 3.3V is used
2	RST	Reset pin – used to reset or power down the module
3	Ground	Connected to Ground of system
4	IRQ	Interrupt pin – used to wake up the module when a device comes into range
5	MISO/SCL/Tx	MISO pin when used for SPI communication, acts as SCL for I2c and Tx for UART.
6	MOSI	Master out slave in pin for SPI communication
7	SCK	Serial Clock pin – used to provide clock source
8	SS/SDA/Rx	Acts as Serial input (SS) for SPI communication, SDA for IIC and Rx during UART

RC522 Features

- 13.56MHz RFID module
- Operating voltage: 2.5V to 3.3V
- Communication : SPI, I2C protocol, UART

- Maximum Data Rate: 10Mbps
- Read Range: 5cm
- Current Consumption: 13-26mA
- Power down mode consumption: 10uA (min)

3.6.4Using RC522 RFID Module:

The RC522 is a RF Module that consists of a RFID reader, RFID card and a key chain. The module operates 13.56MHz which is industrial (ISM) band and hence can be used without any license problem. The module operates at 3.3V typically and hence commonly used in 3.3V designs. It is normally used in application where certain person/object has to be identified with a unique ID.

The keychain has 1kB memory in it which can be used to stored unique data. The RC522 reader module can both read and write data into these memory elements. The reader can read data only form passive tags that operate on 13.56MHz.

3.6.5 Operating RC522 RFID Module:

The RC522 has an operating voltage between 2.5V to 3.3V and hence is normally powered by 3.3V and should be used with 3.3V communication lines. But, the communication pins of this module are 5V tolerant and hence it can be used with 5V microcontrollers also like Arduino without any additional hardware. The module supports SPI, IIC and UART communication but out of these SPI is often used since it is the fasted with a maximum data rate of 10Mbps.

Since in application, most of the time reader module will be waiting for the tag to come into proximity. The Reader can be put into power down mode to save power in battery operated applications. This can be achieved by using the IRQ pin on the module. The minimum current consumed by the module during power down mode will be 10uA only. The module can be easily used with Arduino because of its readily available RC522 RFID Arduino library from Miguel Balboa. You can visit his GitHub page for more details on how to use it with Arduino.

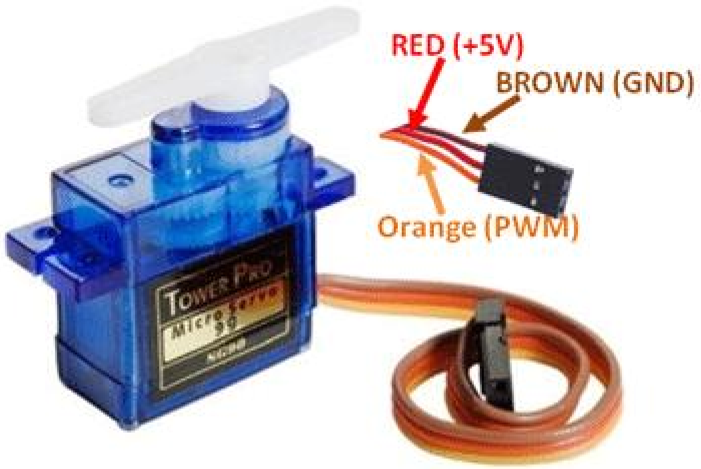
3.6.6 Applications:

- Automatic billing systems

- Attendance systems
- Verification/Identification system
- Access control systems

3.7 Servo Motor SG-90:

(A)



(B)

Fig 3.7-Servo Motor SG90

3.7.1 Wire Configuration:

Wire Number	Wire Colour	Description
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange	PWM signal is given in through this wire to drive the motor

3.7.2 TowerPro SG-90 Features

- Operating Voltage is +5V typically
- Torque: 2.5kg/cm
- Operaring speed is 0.1s/60°
- Gear type: Plastic
- Rotation: 0° -180°
- Weight of motor:9gm
- Packages include gear horns and screws

3.7.3 Selecting your Servo Motor

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.

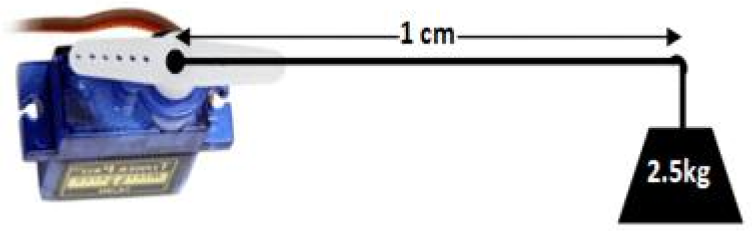


Fig 3.9 Servo motor weight range

3.7.4 Using Servo Motor:

After selecting the right Servo motor for the project, comes the question 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 colour 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 a look at the picture given in the datasheet.

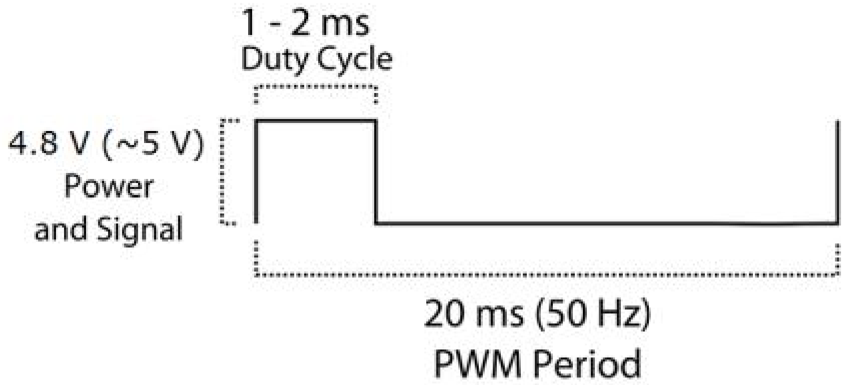


Fig.3.9.1-Servo motor graph

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°

3.7.5 Applications:

- Used as actuators in many robots like Biped Robot, Hexapod, robotic arm etc..
- Commonly used for steering system in RC toys
- Robots where position control is required without feedback
- Less weight hence used in multi DOF robots like humanoid robots

3.8 LCD:

3.8.1 LCD 16×2 Pin Configuration and Its Working

Nowadays, we always use the devices which are made up of LCDs such as CD players, DVD players, digital watches, computers, etc. These are commonly used in the screen industries to replace the utilization of CRTs. Cathode Ray Tubes use huge power when compared with LCDs, and CRTs heavier as well as bigger. These devices are thinner as well power consumption is extremely less. The LCD 16×2 working principle is, it blocks the light rather than dissipate. This article discusses an overview of LCD 16X2, pin configuration and its working.

3.8.2 LCD 16×2?

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.

Fig 3.10-LCD display

3.8.3 LCD 16×2 Pin Diagram

The 16×2 LCD pinout is shown below.

- 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 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 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 unit 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.

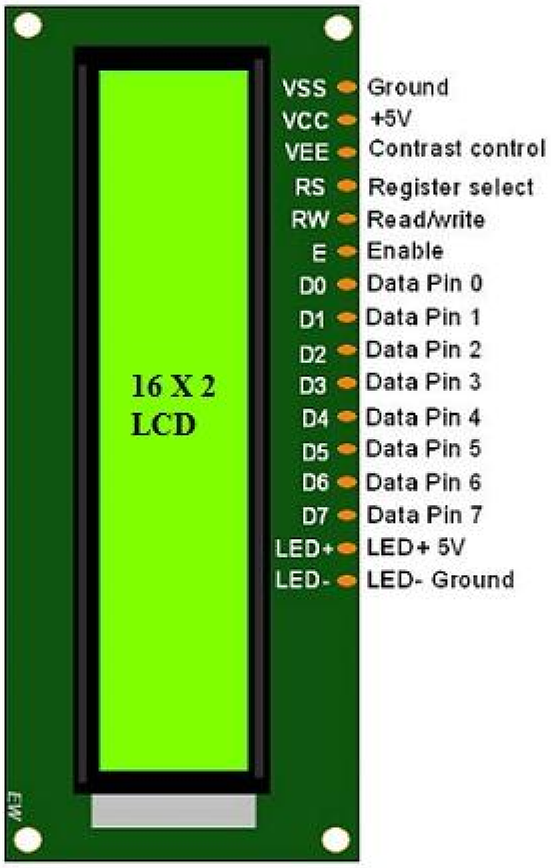


Fig 3.11- LCD pin names

3.8.4 Features of LCD16x2

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

3.8.5 Registers of LCD:

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.

3.8.6 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.

3.8.7 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.

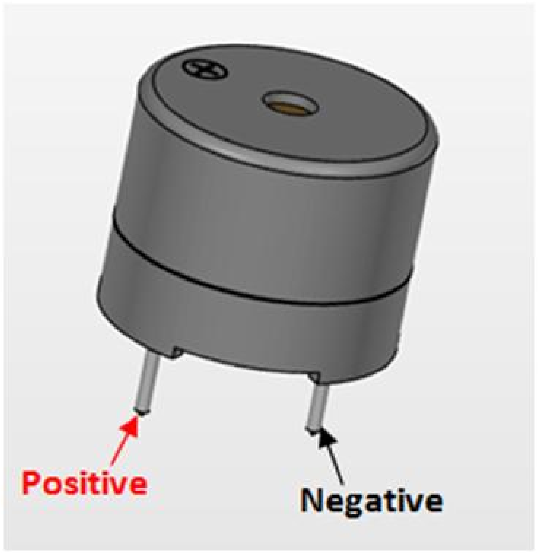
3.8.8 16×2 LCD Commands:

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 leftss
- 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

Fig 3.12-Active Passive Buzzer



3.9 Buzzer Pin Configuration:

Pin Number	**Pin Name**	**Description**
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
2	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

3.9.1 Buzzer Features and Specifications:

- Rated voltage:6VDC
- Opearting Voltage:4-8V DC
- Rated current:<30mA
- Sound Type: Continuous Beep
- Resonant Frequency:~2300Hz
- Small and neat sealed package
- Breadboard and perfboard friendly

3.9.2 Using Buzzer:

A buzzer is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard, Perf Board and even on PCBs which makes this a widely used component in most electronic applications.

There are two types are buzzers that are commonly available. The one shown here is a simple buzzer which when powered will make a Continuous Beeeeeeppp.... sound, the other type is called a readymade buzzer which will look bulkier than this and will produce a Beep. Beep. Beep. Sound due to the internal oscillating circuit present inside it. But, the one shown here is most widely used because it can be customised with help of other circuits to fit easily in our application.

This buzzer can be used by simply powering it using a DC power supply ranging from 4V to 9V. A simple 9V battery can also be used, but it is recommended to use a regulated +5V or +6V DC supply. The buzzer is normally associated with a switching circuit to turn ON or turn OFF the buzzer at required time and require interval.

3.9.3 Applications of Buzzer:

- Alarming Circuits, where the user has to be alarmed about something
- Communication equipments
- Automobile electronics
- Portable equipments, due to its compact size

3.10 WIFI MODULE (ESP8266):

ESP8266 was designed by the Chinese company Espressif Systems for uses in Internet of Things (IoT) systems. ESP8266 is a complete WiFi system on chip that incorporates a 32-bit processor, some RAM and depending on the vendor between 512KB and 4MB of flash memory. This allows the chip to either function as a wireless adapter that can extend other systems with WiFi functionality, or as a standalone unit that can by itself execute simple applications. Depending on the specific module variant (ESP-1 to ESP-12 at the time of this thesis) between 0 and 7 General Purpose Input/Output (GPIO) pins are available, in addition to Rx and Tx pins of the UART, making the module very suitable for IoT applications. The Software Development Kit (SDK) provided by Espressif contains a lightweight implementation of a TCP/IP control stack (lwIP) for WiFi communication. The modules houses libraries for optional services such as Dynamic Host Configuration Protocol (DHCP), Domain Name System (DNS), JavaScript Object Notation (JSON) and Secure Socket Layer (SSL) libraries for Application Level programming. It incorporates 802.11 MAC extensions such as 802.11b/g/n/d/e/h/i/k/r that manage signal transmission, encapsulation, encryption, collision management and roaming functionality. The chip generally comes as part of a module, soldered to a Printed Circuit Board (PCB), however it is possible to purchase only the chip itself in order to create a truly custom module. The module variants currently available on the market may include an antenna (PCB or ceramic) or a U-FL connector, a hardware component for serial communication and a myriad of other auxiliary components such as resistors, capacitors and LEDs.

3.11 Overview and Specification:

The module comes in many different variations (ESP-01 to ESP-12), along with non-Espressif vendors such as Olimex and NodeMCU. The main differences between these modules are size and additional components of the PCB, with some having an inbuilt PCB antenna and up to 7 GPIO pins, while others provide no easy access to GPIO and no antenna, but at a lower cost and module dimensions. The processor inside the module is a low power, 80MHz, 32bit Tensilica Xtensa LX. It is classified as a DPU, which is Tensilicas own type of CPU combining the strengths of a traditional CPU and a DSP to achieve better performance for data-intensive tasks. Multiple compilation tools exist for this processor with the ESP community even attempting to design their own version of gcc compiler to achieve more efficient code density and better performance. The amount of programmable memory varies depending on the module manufacturer, but generally ESPs come with either 512KB, 1MB, 2MB or 4MB of flash memory. ESP8266 is interrupt driven, with a relatively simple OS and three levels of task priority, meaning that only three user tasks that can respond to interrupts can be defined. A function user_init() configures the module once its provided with power, and can be used to schedule the next task, or define a fully event-driven configuration.

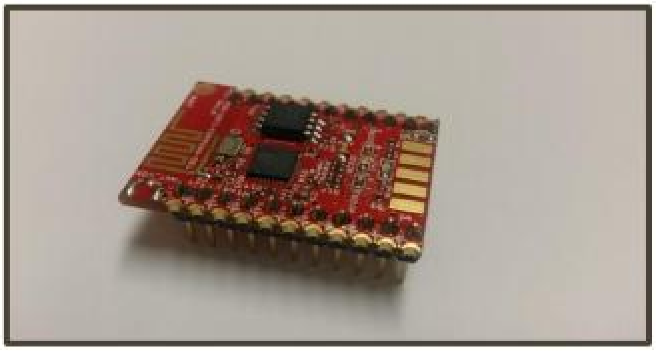
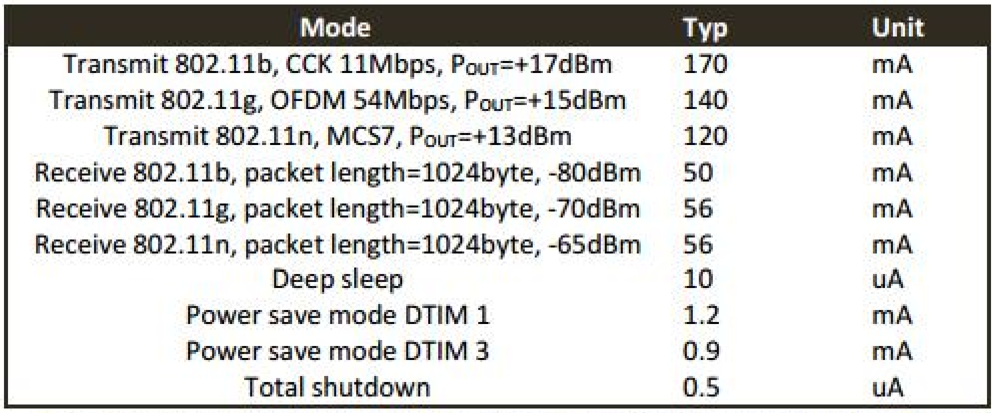


Fig.3.14-Node MCU

The module used in this thesis is an Olimex MOD-WIFI-ESP8266-DEV with all of the basic components of ESP8266, a PCB antenna, crystal and an easily accessed UART with support for SPI and I2C, 2Mbytes of flash, but more importantly for this thesis it has all the available chip pins mapped out for easier access.

3.12 Power Supply and Consumption:

Being a WiFi SoC, this chip requires a fair amount of power to operate its transceiver. It has incorporated some impressive power management features, including highly integrated components that allow for greater optimization and increased efficiency. All this makes ESP8266 one of the least power-hungry chips in the WiFi IC industry! Unfortunately its levels of demand are still higher than of those based on wireless technologies such as Bluetooth, or ZigBee. The official ESP8266 datasheet states this regarding current draw:



Mode	Typ	Unit
Transmit 802.11b, CCK 11Mbps, P_{OUT}=+17dBm	170	mA
Transmit 802.11g, OFDM 54Mbps, P_{OUT}=+15dBm	140	mA
Transmit 802.11n, MCS7, P_{OUT}=+13dBm	120	mA
Receive 802.11b, packet length=1024byte, -80dBm	50	mA
Receive 802.11g, packet length=1024byte, -70dBm	56	mA
Receive 802.11n, packet length=1024byte, -65dBm	56	mA
Deep sleep	10	uA
Power save mode DTIM 1	1.2	mA
Power save mode DTIM 3	0.9	mA
Total shutdown	0.5	uA

Table 3.2 ESP8266EX current draw at 3.3V as listed in the official documentation from Espressif

However this is just the power consumption of the ESP8266EX chip, the entire module, featuring additional hardware such as LEDs, crystals, capacitors and registers revealed that actual consumption of the MOD-WiFi-ESP8266-DEV varied greatly from this table. The approximate idle (while ready to receive packets) current of the module was measured to be 70mA, with somewhat higher when receiving packets in 802.11n mode. Transmission drew 80mA current.

The module was also prone to high current spikes in the range of 300mA at unpredictable points in time, often causing a full module restart. This problem does seem to have been dealt with in most recent SDK, however most manufacturers still choose to add additional capacitances parallel to power supply in order to prevent such instances from occurring. The ESP8266 can operate in a total of 3 power saving modes all of which sacrifice a portion of functionality to achieve lower power consumption. These modes are: Light Sleep• Modem Sleep• Deep Sleep• The Light and Modem modes are so called WiFi sleep modes. They are designed to be used when the module is in STA mode, meaning it does not have to actively send beacons to announce its presence and verify clients' statuses. The Light Sleep WiFi mode is to be used when the module needs to maintain WLAN connection without actively transmitting or receiving data, allowing the CPU to operate at a lower voltage (or be suspended altogether) and turning off the WiFi modem between AP status beacons. This would allow the module to save power while still answer to beacons, effectively maintaining the wireless connection. In this mode a DTIM3 setup at the AP, with 300ms sleep and 3ms wake cycle can, according to the datasheet, lower power consumption to 0.9mA [15]. Modem Sleep is used when the CPU needs to be active. In this mode the WiFi modem is turned off between the AP status beacons, maintaining the connection at minimal cost while allowing the CPU to perform without interference. A similar DTIM3 setup as in previous example leaves the current consumption at 15mA [15]. The Deep Sleep turns of all functionality (CPU and WiFi modem), while maintaining only the RTC clock, allowing the module to be woken up by a timed interrupt. When waking up, the module performs a complete reset, meaning all RAM data is erased (although there is limited space in RTC memory block that does not get erased) and the WiFi connection needs to be re-established. Espressif claims a 300s sleep and 1s wake cycle (claimed as enough to connect to AP) results in average consumption of less than 1mA [15]. Important to note is that WiFi sleep-modes described above may not always be true. In test conducted in this thesis they only seemed to lower the power consumption to 20-50mA in Light Sleep and 40mA in Modem Sleep. Although conventional Deep Sleep resulted in an average of 10uA, there were instances when the deep-sleep sequence seemed to execute in an incorrect manner, leaving the power consumption at standby levels (~80mA). This issue has been known to Espressif that has promised to address it in future releases of the SDK. It should also be noted that the module is specified for voltages between 1.7V and 3.6V, meaning it can be powered by two AA alkaline batteries placed in series (achieving 3V). However the ESP will not work with a Lithium Ion (or LiPo) based batteries without additional power regulating circuits.

3.13 Serial Communication:

ESP8266 has multiple peripherals through which it can interface with other modules in a classic embedded fashion. In this section only the setup of the communication link will be presented, since the exact flow of bits to achieve such communication was handled automatically by the module and is therefore deemed of no immediate interest for this thesis. In this case classical UART was used to decode output and encoding data to be sent to the sensor. Similarly EM50 data logger has an UART of its own and can do the same thing on its end. Serial asynchronous communication does not require a common clock, however in order for the data to be processed correctly and at right intervals a common baud rate (can be viewed as symbols per second) needs to be set for both devices. The baud-rates supported by ESPs UART component range from 9600 to 921600bps, while the EM50 is configured for 9600bps as default.

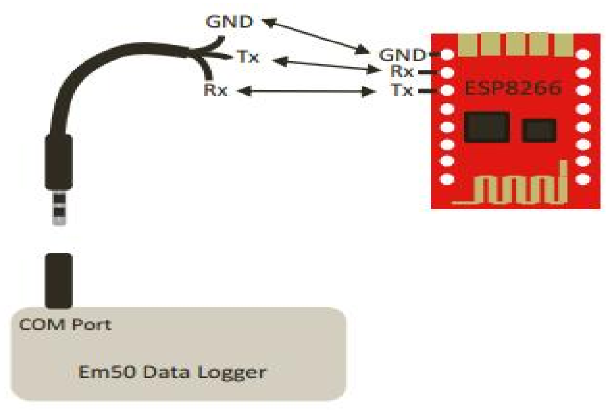


Fig.3.16-Serial communication

CHAPTER-4

SOFTWARE DESCRIPTION

4.1 ARDUINO IDE:

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

4.1.1Writing Sketches:

Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

✓

Verify

Checks your code for errors compiling it.

→

Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the **File | Sketchbook**menu instead.

Save

Saves your sketch.

Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus: **File**, **Edit**, **Sketch**, **Tools**, **Help**. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

File

- *New*

  Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- *Open*

  Allows to load a sketch file browsing through the computer drives and folders.
- *Open Recent*

  Provides a short list of the most recent sketches, ready to be opened.
- *Sketchbook*

  Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

- *Examples*

  Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- *Close*

  Closes the instance of the Arduino Software from which it is clicked.
- *Save*

  Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.
- *Save as...*

  Allows to save the current sketch with a different name.
- *Page Setup*

  It shows the Page Setup window for printing.
- *Print*

  Sends the current sketch to the printer according to the settings defined in Page Setup.
- *Preferences*

  Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- *Quit*

  Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

Edit

- *Undo/Redo*

  Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- *Cut*

  Removes the selected text from the editor and places it into the clipboard.
- *Copy*

  Duplicates the selected text in the editor and places it into the clipboard.
- *Copy for Forum*

  Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.

- *Copy as HTML*
  Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- *Paste*
  Puts the contents of the clipboard at the cursor position, in the editor.
- *Select All*
  Selects and highlights the whole content of the editor.
- *Comment/Uncomment*
  Puts or removes the // comment marker at the beginning of each selected line.
- *Increase/Decrease Indent*
  Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- *Find*
  Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- *Find Next*
  Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- *Find Previous*
  Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Sketch

- *Verify/Compile*
  Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- *Upload*
  Compiles and loads the binary file onto the configured board through the configured Port.
- *Upload Using Programmer*
  This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash

memory for your sketch. Please note that this command will NOT burn the fuses. To do so a *Tools -> Burn Bootloader* command must be executed.

- *Export* compiled binary

  Saves a .hex file that may be kept as archive or sent to the board using other tools.

- *Show Sketch Folder*
  Opens the current sketch folder.
- *Include Library*
  Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- *Add File...*
  Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

Tools

- *Auto Format*
  This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- *Archive Sketch*
  Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.
- *Fix Encoding & Reload*
  Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- *Serial Monitor*
  Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.
- *Board*
  Select the board that you're using. See below for descriptions of the various boards.

- *Port*
  This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.
- *Programmer*
  For selecting a harware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.
- *Burn Bootloader*
  The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the **Boards** menu before burning the bootloader on the target board. This command also set the right fuses.

Help

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

- *Find in Reference*
  This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

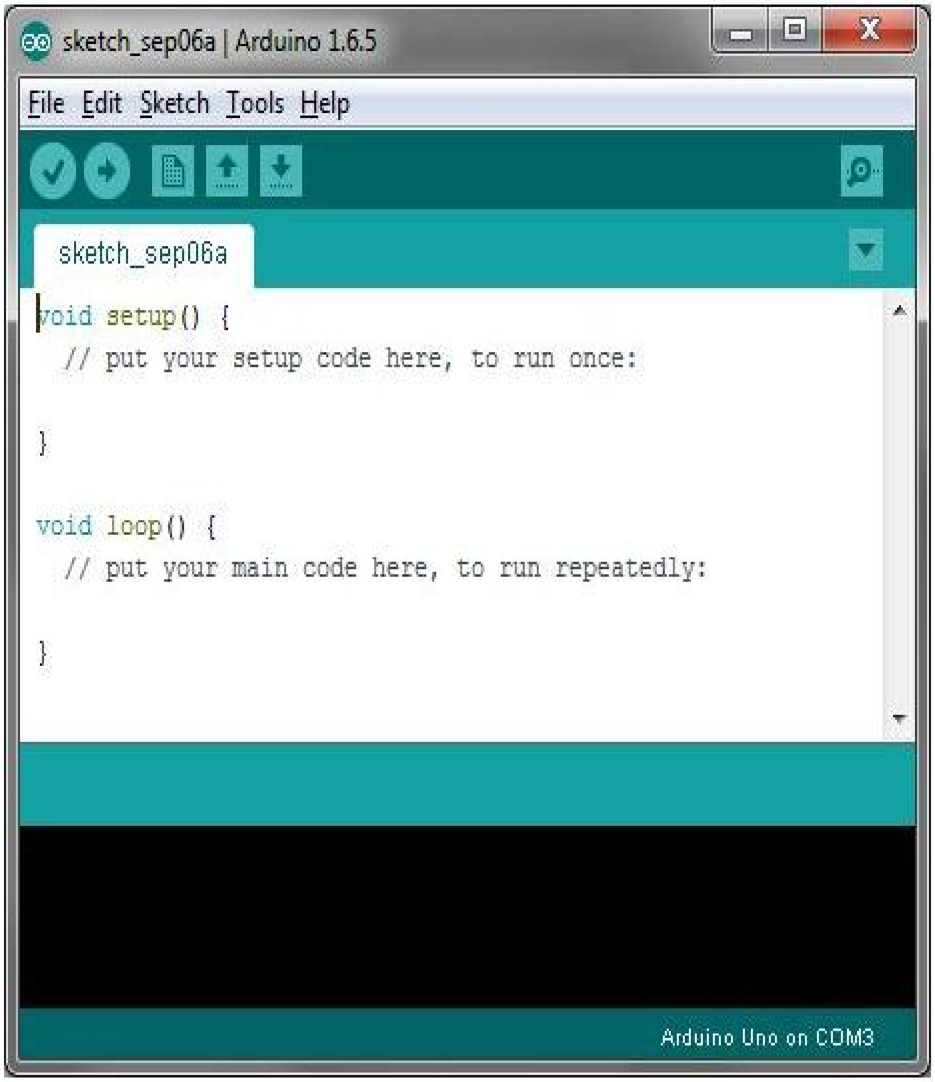
4.2 Arduino IDE: Initial Setup:

This is the Arduino IDE once it's been opened. It opens into a blank sketch where you can start programming immediately. First, we should configure the board and port settings to allow us to upload code. Connect your Arduino board to the PC via the USB cable.

To program the Arduino Uno we will use the Arduino IDE. This tutorial will guide you step-by-step through the installation process.

As the Arduino is programmed via serial port and the on-board USB-to-Serial converter, you need to install a driver for this converter too. What might surprise you is, that there are different versions of the Arduino

Uno. First there are SMD versions and versions using the ATmega328P microcontroller as a DIP package. More important for connecting it to your computer is the fact, that the original Arduino Uno uses an ATmega16U2 microcontroller as USB-to-Serial converter, while other boards use a cheaper CH340 converter chip.

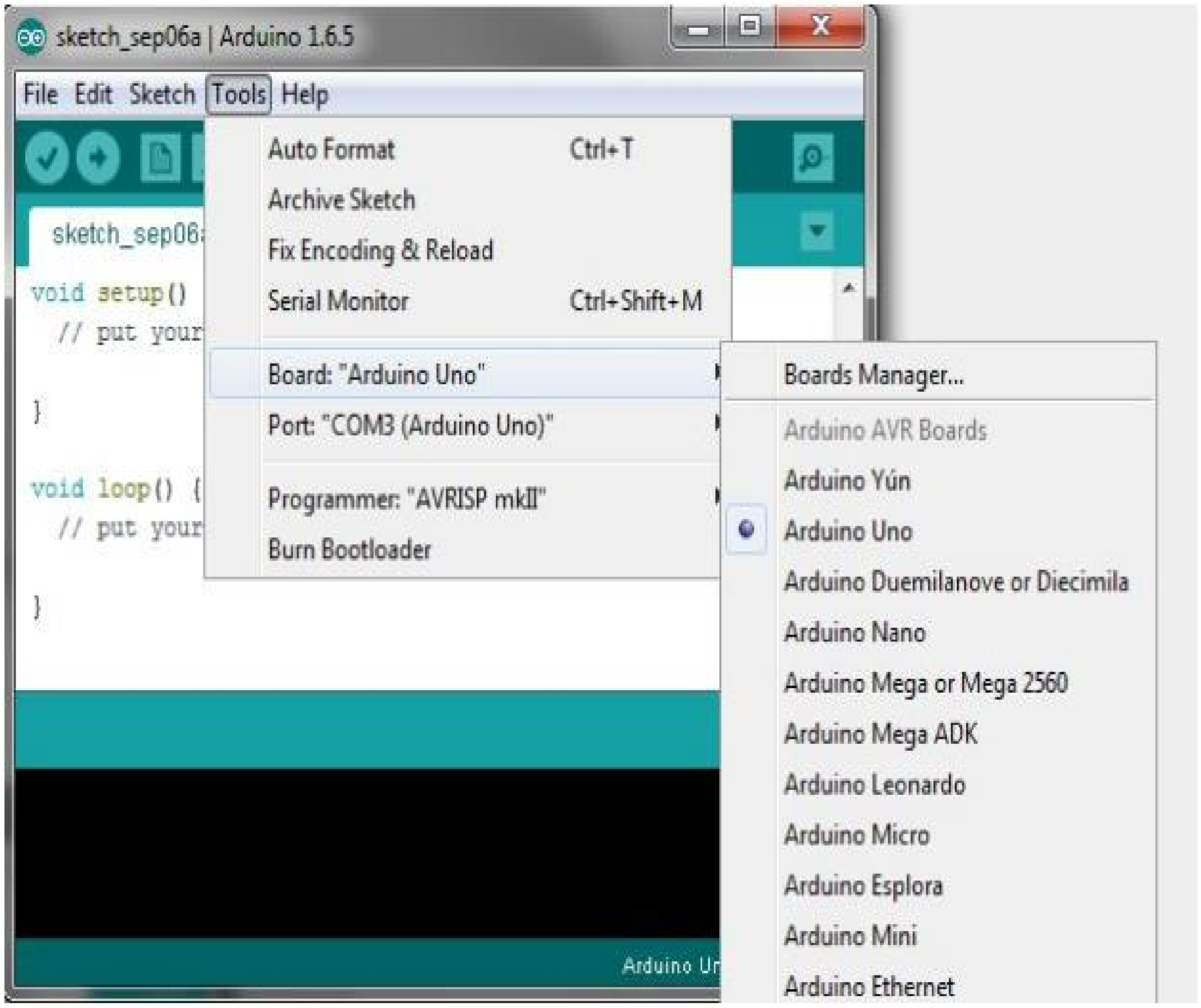


```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig.4.1-Deafault Arduino window

4.2.1 IDE: Board Setup

You have to tell the Arduino IDE what board you are uploading to. Select the Tools pulldown menu and go to Board. This list is populated by default with the currently available Arduino Boards that are developed by Arduino. If you are using an Uno or an Uno-Compatible Clone (ex. Funduino, SainSmart, IEIK, etc.), select Arduino Uno. If you are using another board/clone, select that board.

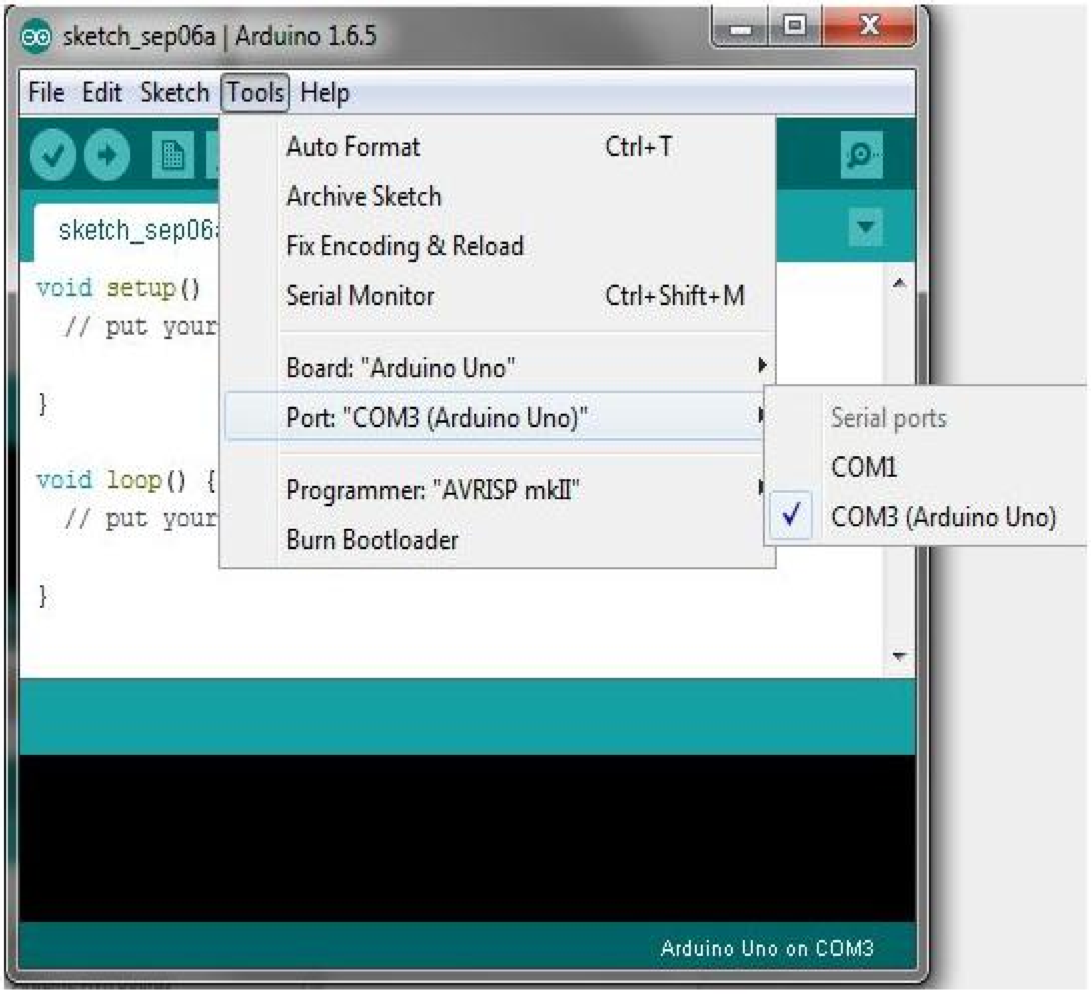


```
void setup()
  // put your

}

void loop() {
  // put your

}
```

Fig.4.2-Arduino IDE: Board Setup Procedure

4.2.2 IDE: COM Port Setup

If you downloaded the Arduino IDE before plugging in your Arduino board, when you plugged in the board, the USB drivers should have installed automatically. The most recent Arduino IDE should recognize connected boards and label them with which COM port they are using. Select the Tools pulldown menu and then Port. Here it should list all open COM ports, and if there is a recognized Arduino Board, it will also give it's name. Select the Arduino board that you have connected to the PC. If the setup was successful, in the bottom right of the Arduino IDE, you should see the board type and COM number of the board you plan to program. Note: the Arduino Uno occupies the next available COM port; it will not always be COM3.



```
void setup()
  // put your

}

void loop() {
  // put your

}
```

Fig.4.3-Arduino IDE:COM Port Setup

4.2.3 Testing Your Settings:

Uploading Blink One common procedure to test whether the board you are using is properly set up is to upload the “Blink” sketch. This sketch is included with all Arduino IDE releases and can be accessed by the File pull-down menu and going to Examples, 01.Basics, and then select Blink . Standard Arduino Boards include a surface-mounted LED labeled “L” or “LED” next to the “RX” and “TX” LEDs, that is connected to digital pin 13. This sketch will blink the LED at a regular interval, and is an easy way to confirm if your board is set up properly and you were successful in uploading code. Open the “Blink” sketch and press the “Upload” button in the upper-left corner to upload “Blink” to the board.

Upload Button:

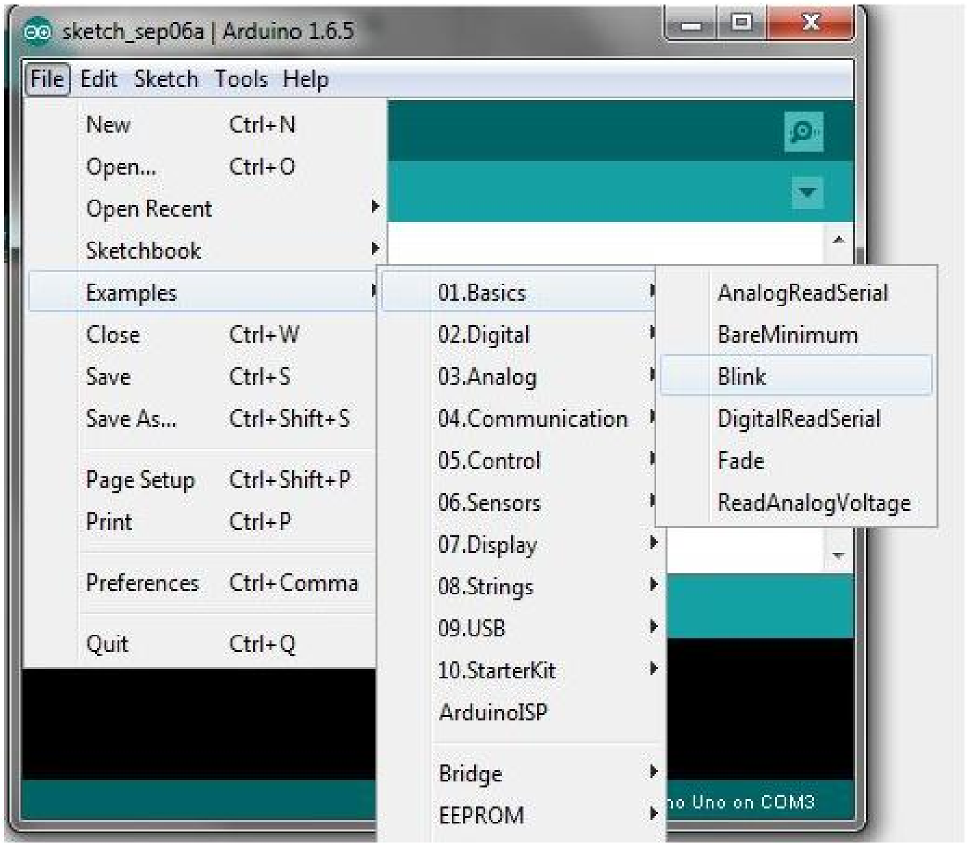


Fig.4.4-Arduino IDE:Loading Blink Sketch

4.2.4 PROTEUS:

Generally we are listening the words **PCB’s**, **PCB layout**, **PCB designing**, ect. But what is PCB? Why we are using this PCB? We want to know about all these things as a electronic engineer. PCB means Printed Circuit Board. This is a circuit board with printed copper layout connections. These PCB’s are two types. One is dotted PCB and another one is layout PCB. The two examples are shown in below.

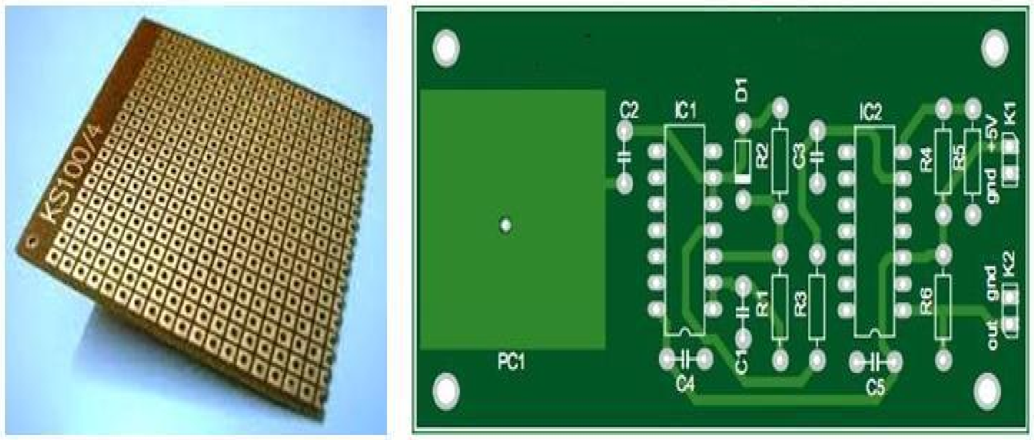


Fig.4.5-dotted PCB & Layout PCB

4.2.5 Difference between the dotted PCB and layout PCB?

In dotted PCB board only dots are available. According to our requirement we can place or insert the components in those holes and attach the components with wires and soldering lid. In this dotted PCB we can make the circuit as out wish but it is very hard to design. There are so many difficulties are there. Those are connecting the proper pins, avoiding shot connections and etc. Coming to the layout PCB this is simple to design. First we select the our circuit and by using different PCB designing software's, design the layout of the circuit and by itching process preparing the copper layout of our circuit and solder the components in the correct places. It is simple to design, take less time to design, no shortages, looking nice and perfect.

Up to now we have discussed about types of PCB's and difference between the types. Now we can discuss about **PCB designing software**. There are so many PCB designing softwares available. Some are **Express PCB**, **eagle PCB**, **PCB Elegance**, **free PCB**, o**pen circuit design**, **zenith PCB** and**Proteus** etc. Apart from remaining Proteus is different. Proteus is design suit and PCB layout designing software. In Proteus we can design any circuit and simulate the circuit and make PCB layout for that circuit.

4.3 Introduction to Proteus:

Proteus professional is a software combination of ISIS schematic capture program and ARES PCB layout program. This is a powerful and integrated development environment. Tools in this suit are very easy to use and these tools are very useful in education and professional PCB designing. As professional PCB designing software with integrated space based auto router, it provides features such as fully featured schematic capture, highly configurable design rules, interactive SPICE circuit simulator, extensive support for power planes, industry standard CADCAM & ODB++ output and integrated 3D viewer.

Up to know we have discussed about the basics and software description. Now we are entering into the designing section. Run the ISIS professional program by clicking the icon on the desktop, then this splash screen will appear.

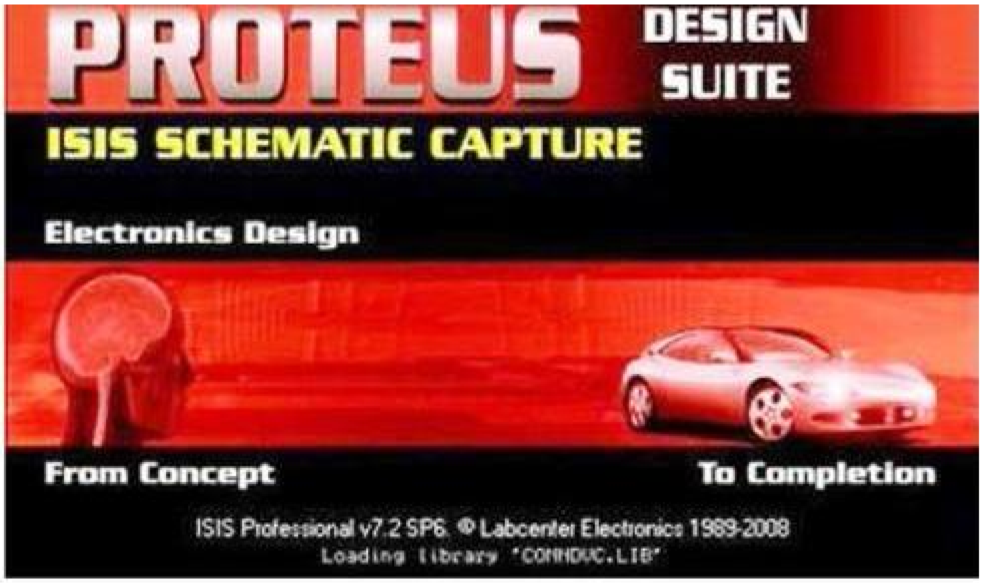

PROTEUS
DESIGN SUITE
ISIS SCHEMATIC CAPTURE
Electronics Design
From Concept
To Completion
ISIS Professional v7.2 SP6. © Labcenter Electronics 1989-2008
Loading library 'CONNDUC.LIB'

Fig.4.6-Design Suite

Next, a work space with interface buttons for designing circuit will appear as shown in figure below. Note that there is a blue rectangular line in the workspace; make sure that whole circuit is designed inside the rectangular space.

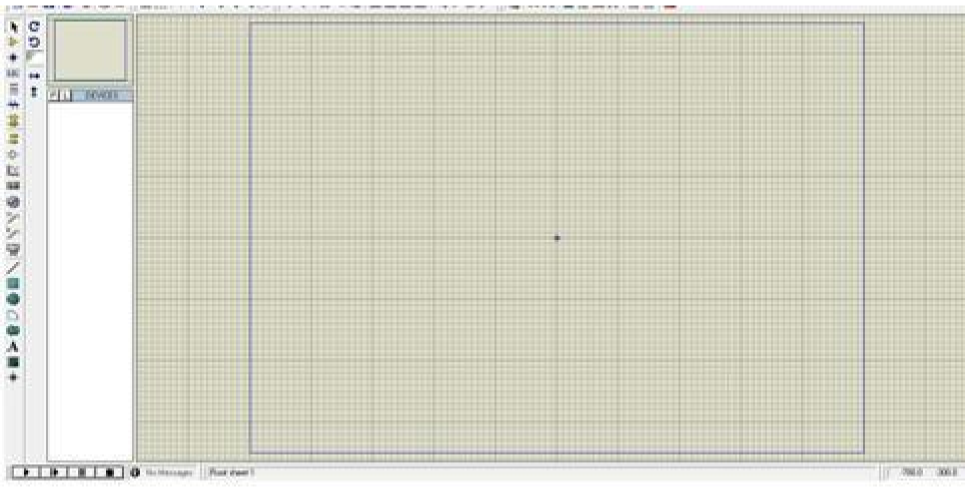


Fig.4.6-Graph

Next step is selecting the components to our required circuit. Let us take one example is designing of 38 kHz frequency generator by using 555 timer IC. The circuit diagram is shown in below image.

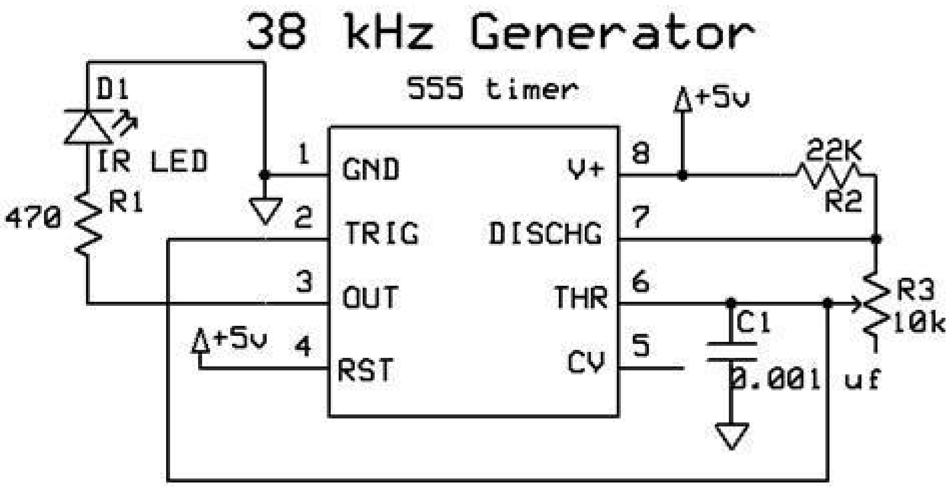
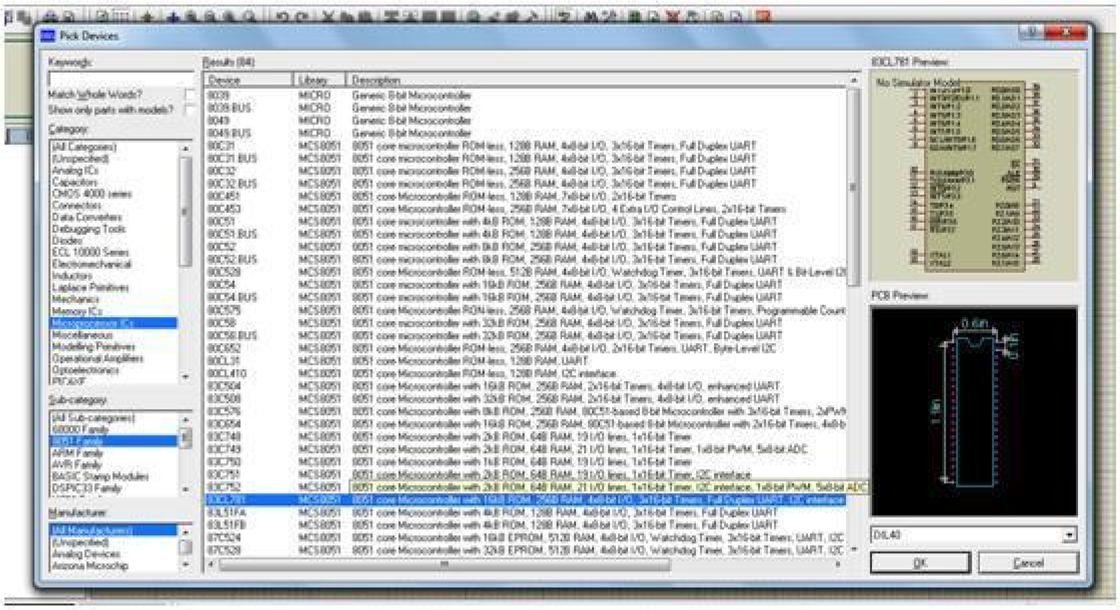


Fig.4.7-Circuit diagram

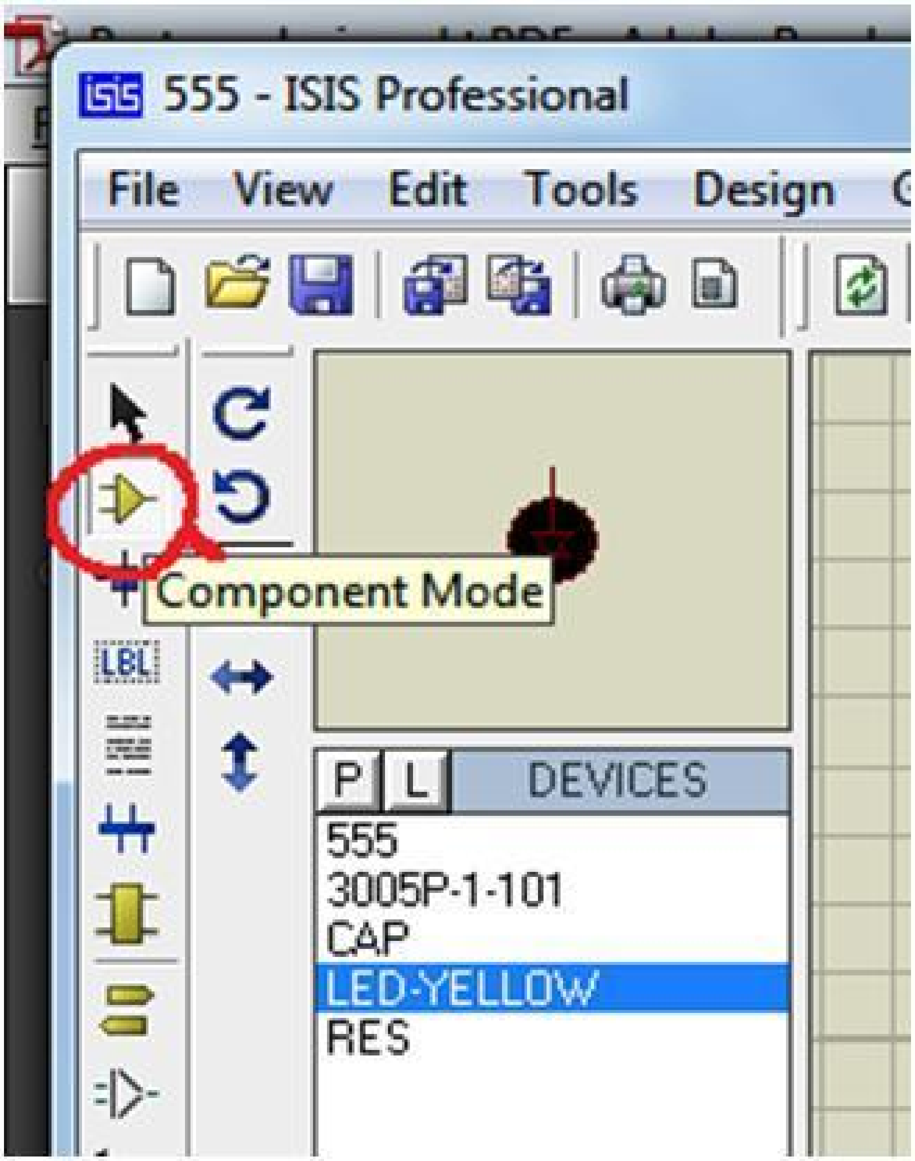
In the above circuit the required components are 555 timer IC, 470? and 22K? resisters, 10K? variable resister, 0.001µf capacitor and one IR LED. So select the components from library. In menu bar library > pick device/ symbol. Then one window will open that shown in below.



Device	Library	Description
8039	MICRO	Generic 8-bit Microcontroller
8039 BUS	MICRO	Generic 8-bit Microcontroller
8049	MICRO	Generic 8-bit Microcontroller
8049 BUS	MICRO	Generic 8-bit Microcontroller
80C31	MCS8051	8051 core microcontroller ROM-less, 128B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C31 BUS	MCS8051	8051 core microcontroller ROM-less, 128B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C32	MCS8051	8051 core microcontroller ROM-less, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C32 BUS	MCS8051	8051 core microcontroller ROM-less, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C451	MCS8051	8051 core Microcontroller ROM-less, 128B RAM, 7x8-bit I/O, 2x16-bit Timers
80C453	MCS8051	8051 core Microcontroller ROM-less, 256B RAM, 7x8-bit I/O, 4 Extra I/O Control Lines, 2x16-bit Timers
80C51	MCS8051	8051 core microcontroller with 4kB ROM, 128B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C51 BUS	MCS8051	8051 core microcontroller with 4kB ROM, 128B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C52	MCS8051	8051 core microcontroller with 8kB ROM, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C52 BUS	MCS8051	8051 core microcontroller with 8kB ROM, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C528	MCS8051	8051 core Microcontroller ROM-less, 512B RAM, 4x8-bit I/O, Watchdog Timer, 3x16-bit Timers, UART & Bit-Level I2
80C54	MCS8051	8051 core microcontroller with 16kB ROM, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C54 BUS	MCS8051	8051 core microcontroller with 16kB ROM, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C575	MCS8051	8051 core Microcontroller ROM-less, 256B RAM, 4x8-bit I/O, Watchdog Timer, 3x16-bit Timers, Programmable Count
80C58	MCS8051	8051 core microcontroller with 32kB ROM, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C58 BUS	MCS8051	8051 core microcontroller with 32kB ROM, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
80C652	MCS8051	8051 core Microcontroller ROM-less, 256B RAM, 4x8-bit I/O, 2x16-bit Timers, UART, Byte-Level I2C
80CL31	MCS8051	8051 core Microcontroller ROM-less, 128B RAM, UART
80CL410	MCS8051	8051 core Microcontroller ROM-less, 128B RAM, I2C interface
83C504	MCS8051	8051 core Microcontroller with 16kB ROM, 256B RAM, 2x16-bit Timers, 4x8-bit I/O, enhanced UART
83C508	MCS8051	8051 core Microcontroller with 32kB ROM, 256B RAM, 2x16-bit Timers, 4x8-bit I/O, enhanced UART
83C576	MCS8051	8051 core Microcontroller with 8kB ROM, 256B RAM, 80C51-based 8-bit Microcontroller with 3x16-bit Timers, 2xPW
83C654	MCS8051	8051 core Microcontroller with 16kB ROM, 256B RAM, 80C51-based 8-bit Microcontroller with 2x16-bit Timers, 4x8-b
83C748	MCS8051	8051 core Microcontroller with 2kB ROM, 64B RAM, 19 I/O lines, 1x16-bit Timer
83C749	MCS8051	8051 core Microcontroller with 2kB ROM, 64B RAM, 21 I/O lines, 1x16-bit Timer, 1x8-bit PWM, 5x8-bit ADC
83C750	MCS8051	8051 core Microcontroller with 1kB ROM, 64B RAM, 19 I/O lines, 1x16-bit Timer
83C751	MCS8051	8051 core Microcontroller with 2kB ROM, 64B RAM, 19 I/O lines, 1x16-bit Timer, I2C interface
83C752	MCS8051	8051 core Microcontroller with 2kB ROM, 64B RAM, 21 I/O lines, 1x16-bit Timer, I2C interface, 1x8-bit PWM, 5x8-bit ADC
83CL781	MCS8051	8051 core Microcontroller with 16kB ROM, 256B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART, I2C interface
83L51FA	MCS8051	8051 core Microcontroller with 4kB ROM, 128B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
83L51FB	MCS8051	8051 core Microcontroller with 4kB ROM, 128B RAM, 4x8-bit I/O, 3x16-bit Timers, Full Duplex UART
87C524	MCS8051	8051 core Microcontroller with 16kB EPROM, 512B RAM, 4x8-bit I/O, Watchdog Timer, 3x16-bit Timers, UART, I2C
87C528	MCS8051	8051 core Microcontroller with 32kB EPROM, 512B RAM, 4x8-bit I/O, Watchdog Timer, 3x16-bit Timers, UART, I2C

Fig.4.8-Component Selection

There is another way to select the components. In work space left side there is a tool bar. In that tool bar click the component mode button or pick from library.



Select the all components from library, that components are added to devices list. Click on the device and change the angle of the device by using rotate buttons. Then click in the work space then the selected component is placed in work space. Place all the devices in work space and put the curser at the component pin end then draw the connections with that pen symbol. Connect all the components according to circuit then that designed circuit is show in below image.

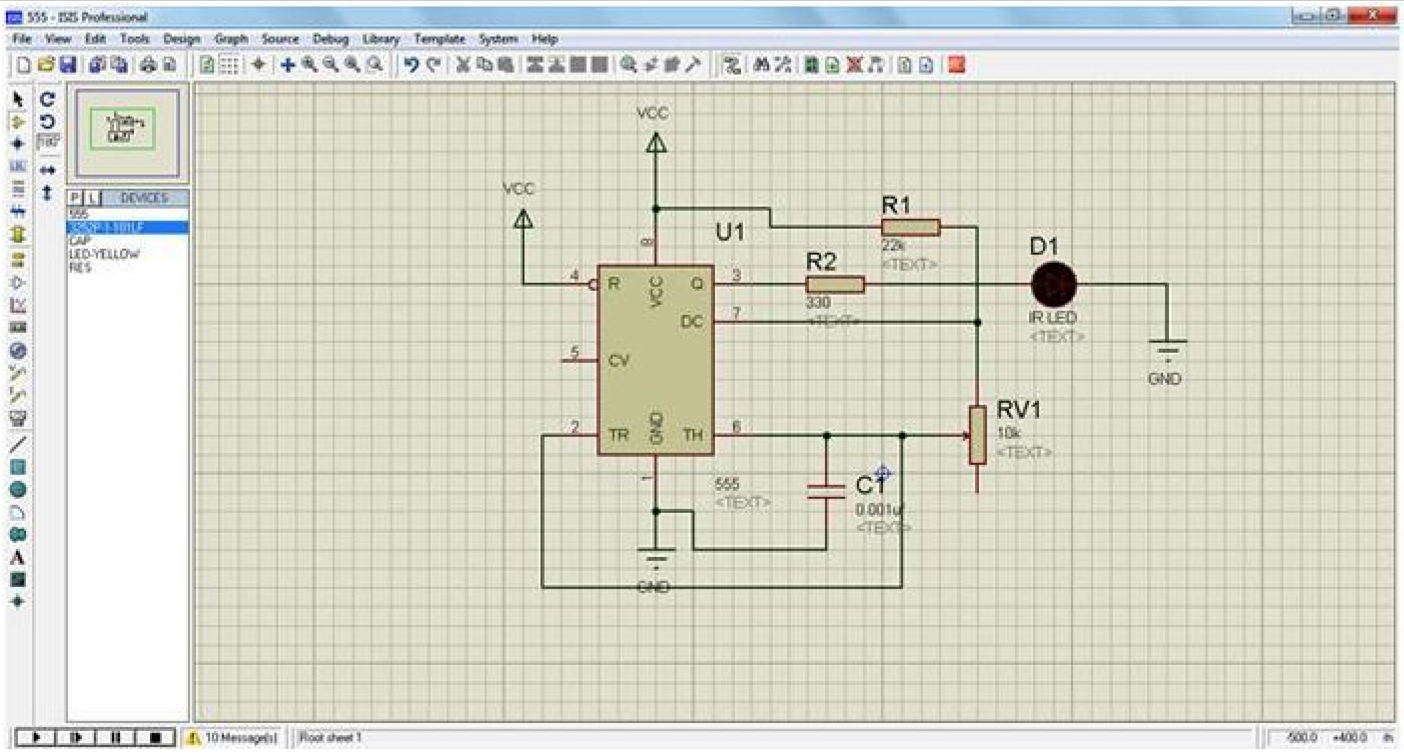
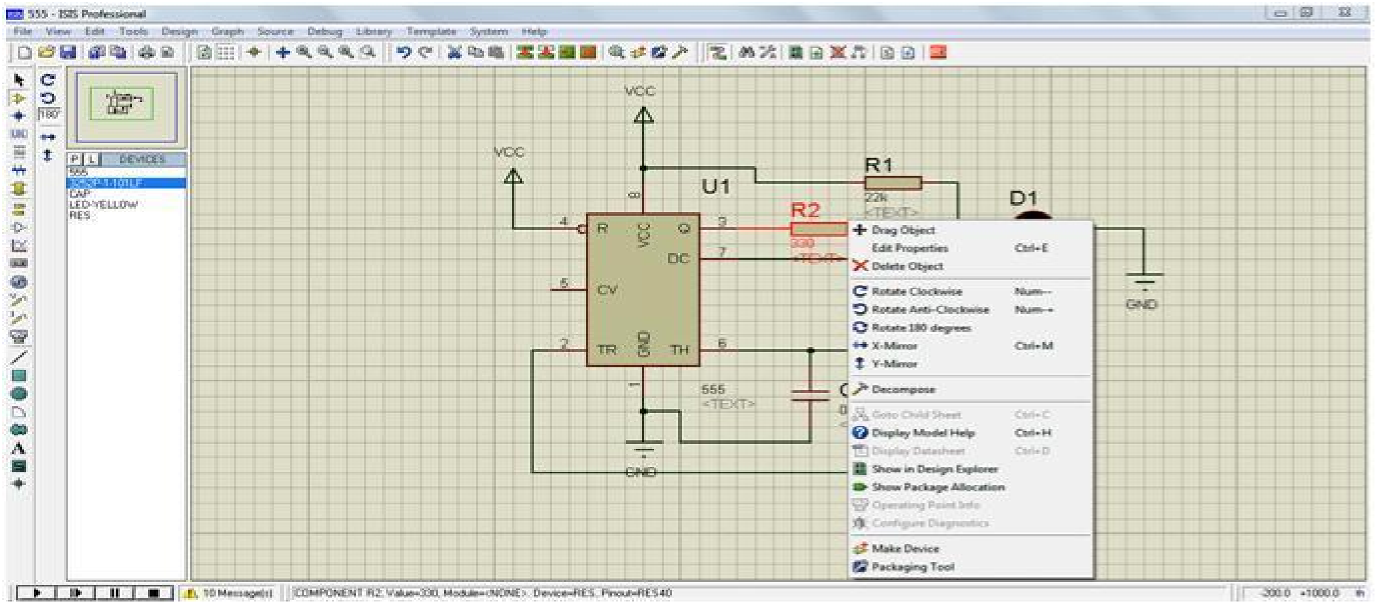


Fig.4.9-Circit Connections

If any modifications want to do to the component place the mouse point and click on right button then option window will open. That is shown in below figure.



COMPONENT R2, Value=330, Module=<NONE>, Device=RES, Pinout=RES40

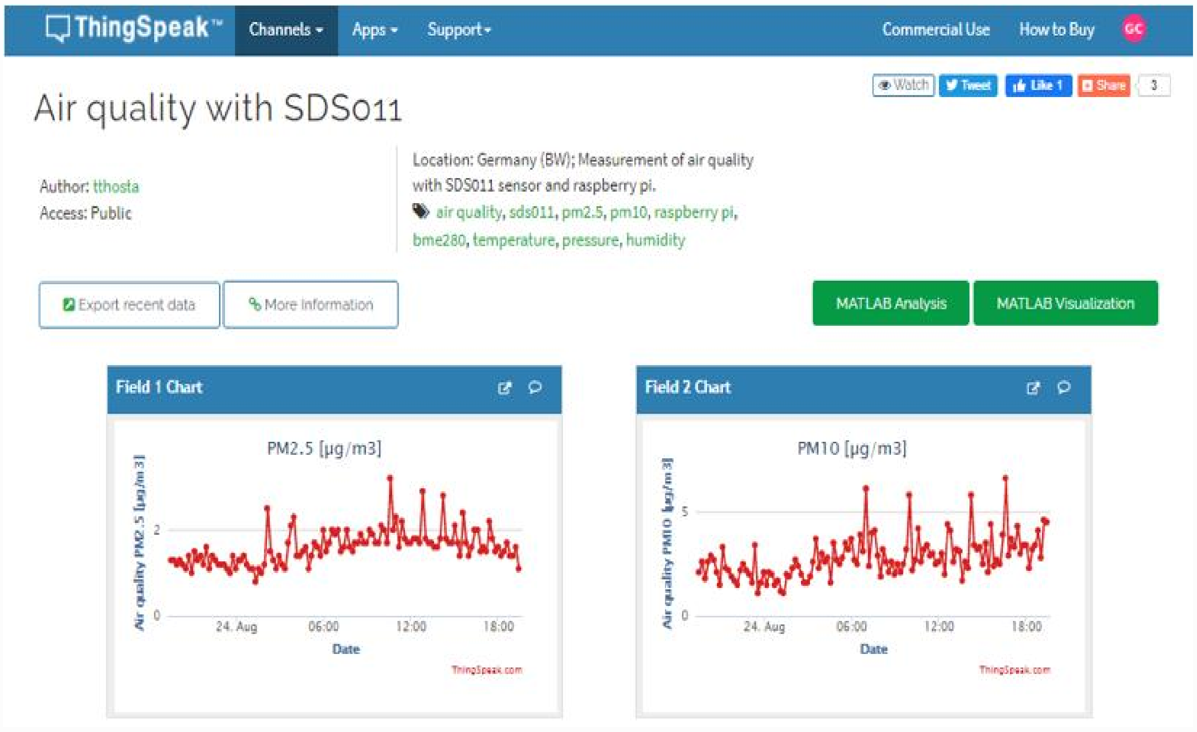
Fig.4.10-Circuit Designing

After completion of designing save with some mane and debug it. This is virtual simulation means without making circuit we can see the result in virtually through this software and we can **design the PCB layout** to our required circuit with this software.

4.4 ThingSpeak IoT Platform:

ThingSpeak is IoT platform for user to gather real-time data; for instance, climate information, location data and other device data. In different channels in ThingSpeak, you can summarize information and visualize data online in charts and analyze useful information.

ThingSpeak can integrate IoT:bit (micro:bit) and other software/ hardware platforms. Through IoT:bit, you can upload sensors data to ThingSpeak (e.g. temperature, humidity, light intensity, noise, motion,raindrop, distance and other device information).



Air quality with SDS011

Author: tthosta
Access: Public

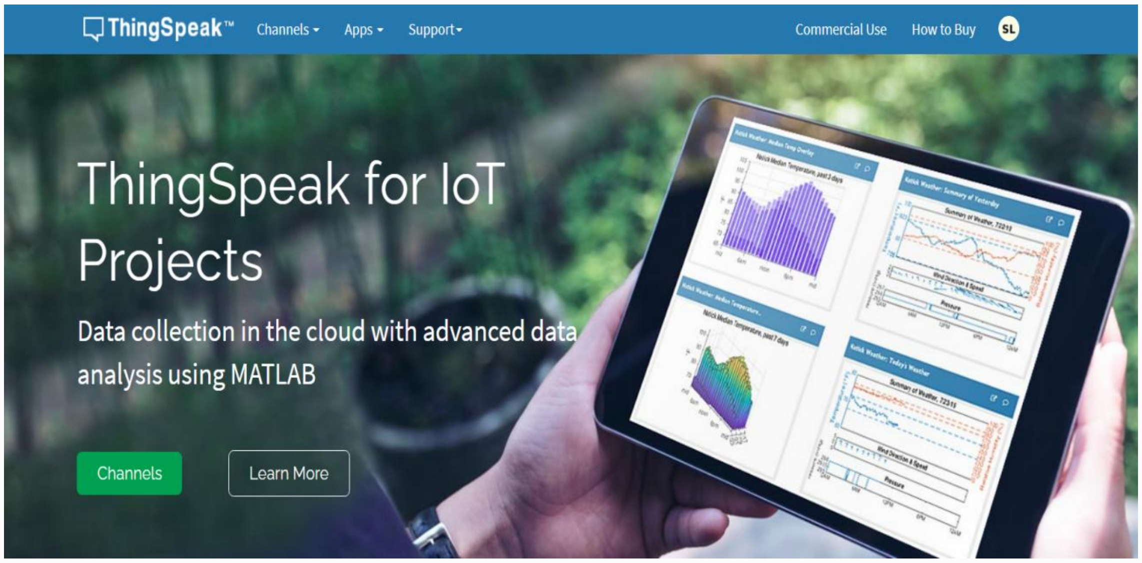
Location: Germany (BW); Measurement of air quality with SDS011 sensor and raspberry pi.
air quality, sds011, pm2.5, pm10, raspberry pi, bme280, temperature, pressure, humidity

4.11. Thingspeak Configuration

Goal:
we need to create the thingspeak channel and get the key

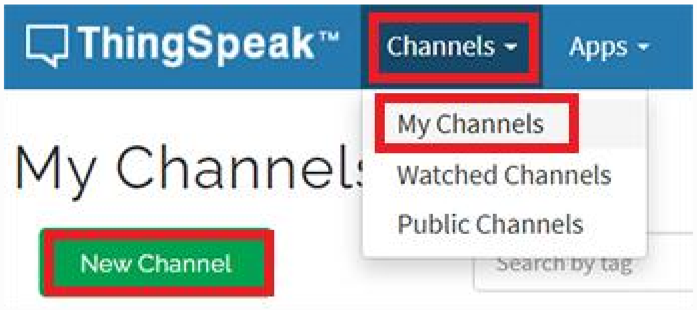
Step 1

Go to https://thingspeak.com/, register an account and login to the platform


ThingSpeak™
Channels
Apps
Support
Commercial Use
How to Buy
ThingSpeak for IoT Projects
Data collection in the cloud with advanced data analysis using MATLAB
Channels
Learn More

Step 2

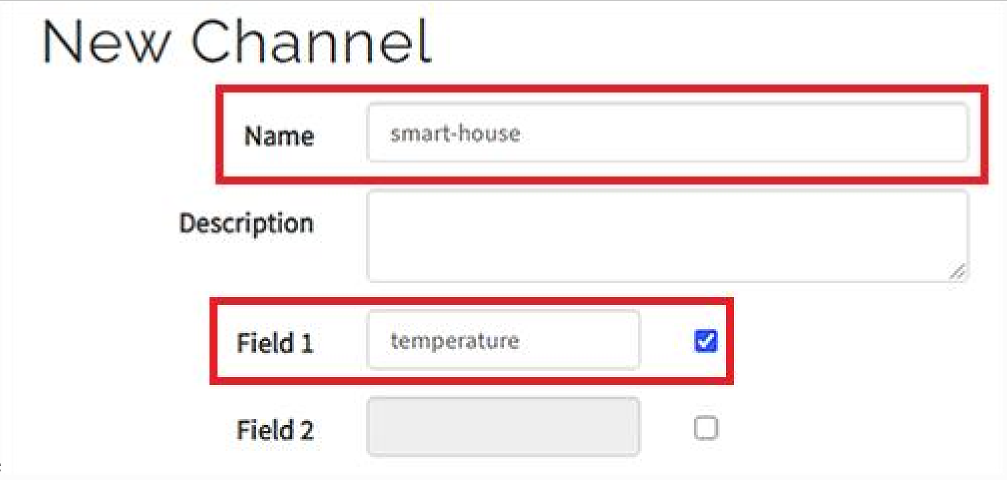
Choose Channels -> My Channels -> New Channel



Step 3

Input Channel name, Field1 , then click "Save Channel"

- Channel name: smart-house



New Channel

Name: smart-house

Description:

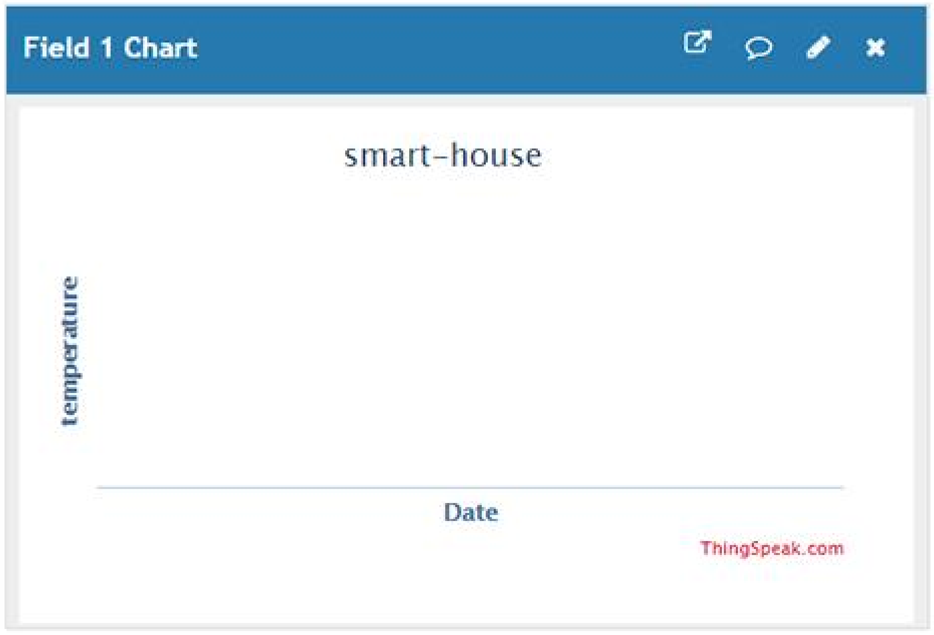
Field 1: temperature ✓

Field 2: ☐

- Field 1: Temperature

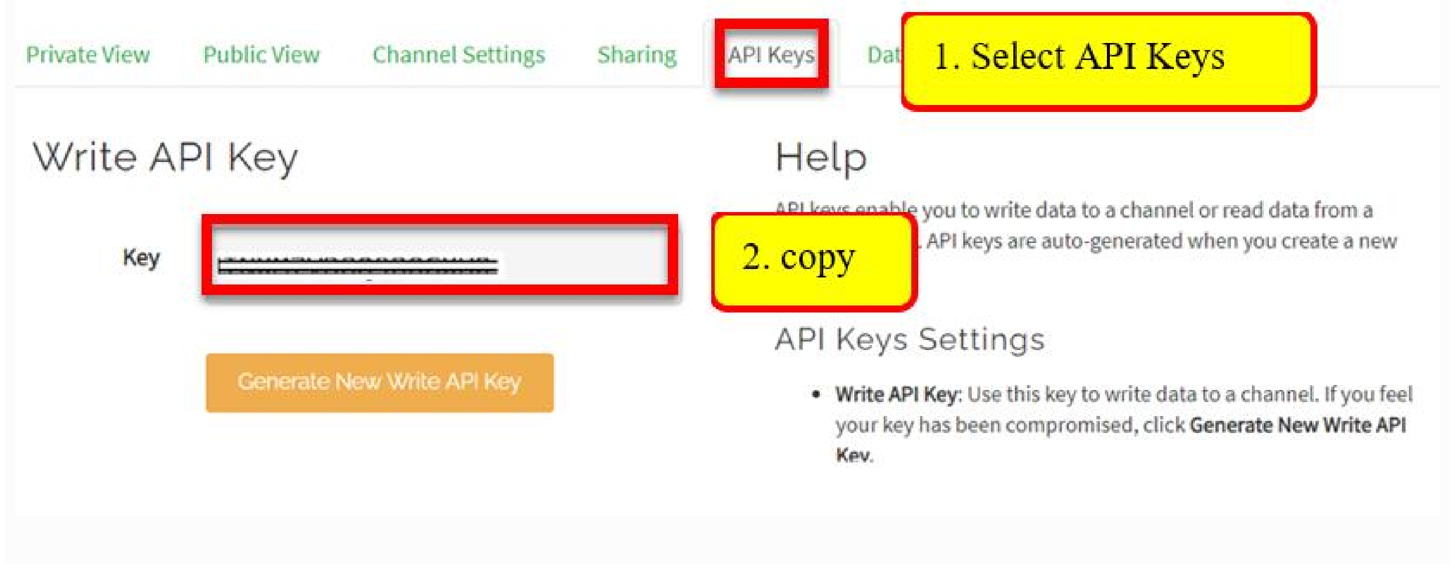
Step 4

You will see a chat for data field1



Step 5

Open your web browser, go to https://thingspeak.com , select your channel > “API Keys” , copy the API key as follows:



Write API Key

Key [redacted]

Help

API keys enable you to write data to a channel or read data from a . API keys are auto-generated when you create a new

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**

4.5 MIT APPINVENTOR:

The smartphone is an information nexus in today’s digital age, with access to a nearly infinite supply of content on the web, coupled with rich sensors and personal data. However, people have difficulty harnessing the full power of these ubiquitous devices for themselves and their communities. Most smartphone users consume technology without being able to produce it, even though local problems can often be solved with mobile devices. How then might they learn to leverage smartphone capabilities to solve real-world, everyday problems? MIT App Inventor is designed to democratize this technology and is used as a tool for learning computational thinking in a variety of educational contexts, teaching people to build apps to solve problems in their communities. MIT App Inventor is an online development platform that anyone can leverage to solve real-world problems. It provides a web-based “What you see is what you get” (WYSIWYG) editor for building mobile phone applications targeting the Android and iOS operating systems. It uses a block-based programming language built on Google Blockly (Fraser, 2013) and inspired by languages such as StarLogo TNG (Begel & Klopfer, 2007) and Scratch (Resnick et al., 2009; Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010), empowering anyone to build a mobile phone app to meet a need. To date, 6.8 million people in over 190 countries have used App Inventor to build over 24 million apps. We offer the interface in more than a dozen languages. People around the world use App Inventor to provide mobile solutions to real problems in their families, communities, and the world. The platform has also been adapted to serve requirements of more specific populations, such as building apps for emergency/first responders (Jain et al., 2015) and robotics (Papadakis & Orfanakis, 2016). In this chapter, we describe the goals of MIT App Inventor and how they have influenced our design and development—from the program's inception at Google in 2008, through the migration to MIT, to the present day. We discuss the pedagogical value of MIT App Inventor and its use as a tool to teach and encourage people of all ages to think and act computationally. We also describe three applications developed by students in different parts of the world to solve real issues in their communities. We conclude by discussing the limitations and benefits of tools such as App Inventor and proposing new directions for research.

4.5.1 MIT App Inventor Overview:

The MIT App Inventor user interface includes two main editors: the design editor and the blocks editor. The design editor, or designer (see Fig. 3.1), is a drag and drop interface to lay out the elements of the application's user interface (UI). The blocks editor (see Fig. 3.2) is an environment in which app inventors can visually lay out the logic of their apps using color-coded blocks that snap together like puzzle pieces to describe the program. To aid in development and testing, App Inventor provides a mobile app called the App Inventor Companion (or just "the Companion") that developers can use to test and adjust the behavior of their apps in real time. In this way, anyone can quickly build a mobile app and immediately begin to iterate and test MIT App Inventor:

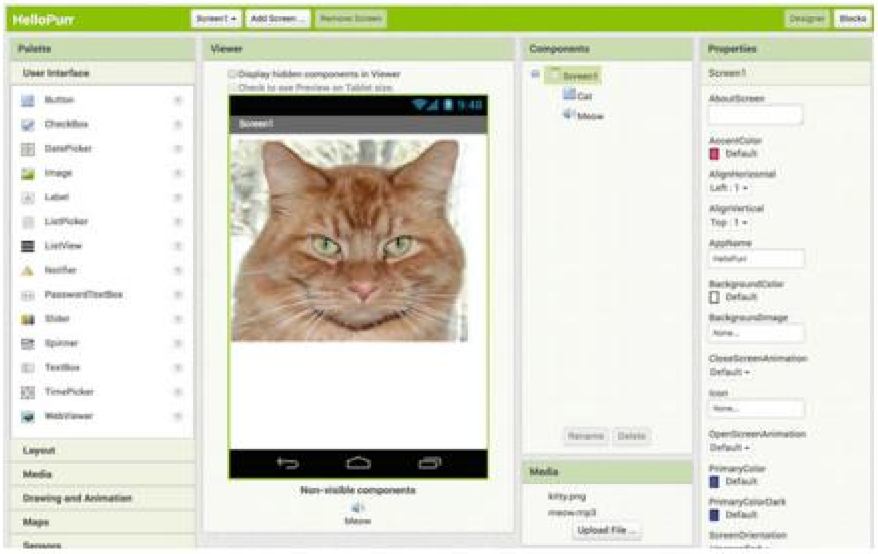


Fig. 4.12-App Inventor's design editor.

In the design of MIT App Inventor, introducing mobile app development in educational contexts was a central goal. Prior to its release, most development environments for mobile applications were clunky, only accessible with expertise in systems level or embedded programming, or both. Even with Google's Android operating system and the Java programming language, designing the user interface was a complex task. Further, use of the platform required familiarity with Java syntax and semantics, and the ability to debug Java compilation errors (e.g., misspelled variables or misplaced semicolons) for success. These challenges presented barriers to entry for individuals not versed in computer science, App Inventor's target demographic. We briefly highlight and discuss design goals for the App Inventor project, specifically, the use of components to abstract some of the complexity of platform behavior, and the use of blocks to eliminate complexity of the underlying programming language. These goals can be further explained as aligning the visual language to the mental models of young developers and enabling exploration through fast, iterative design.

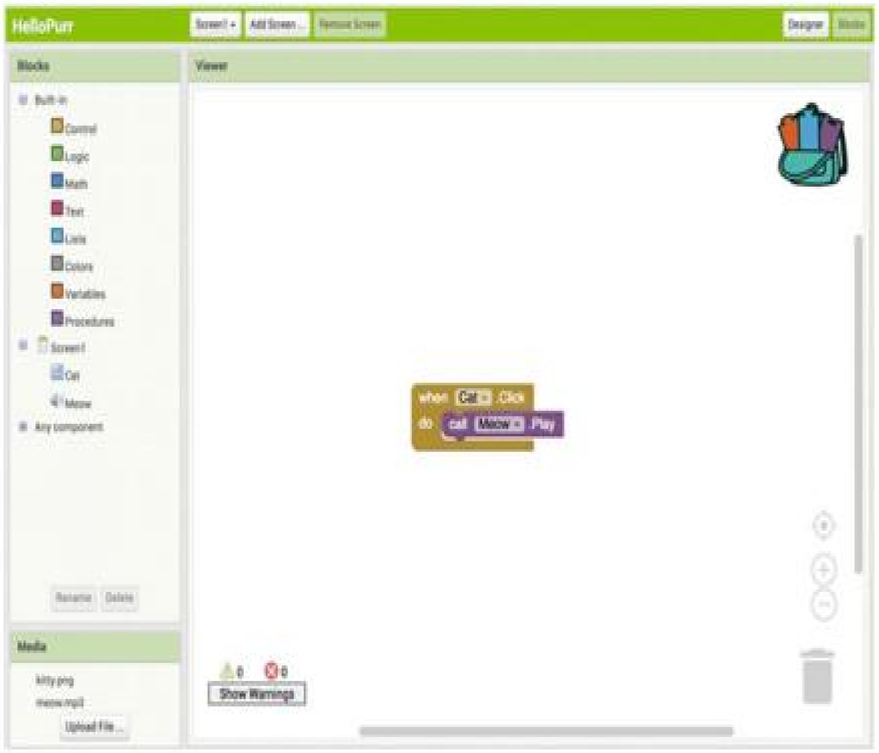


Fig.4.13- App Inventor's blocks editor

CHAPTER-5

5.1 APPLICATIONS

- The smart parking system can be implemented in
- Shopping Malls
- Restaurants
- Theaters

5.2 ADVANTAGES

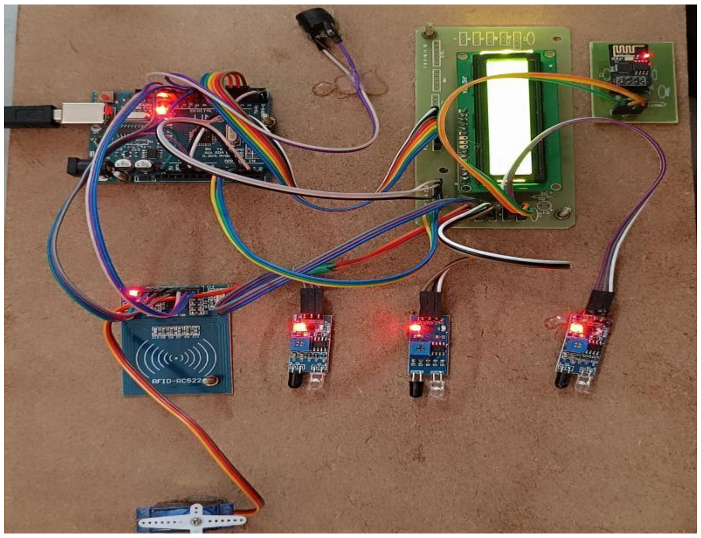
- Shorter waiting time at parking place..
- It saves Fuel,Money,Space and Time.
- Reduces Pollution and Traffic.
- Carbon emission is reduced.
- Efficiency

5.3 DISADVANTAGES

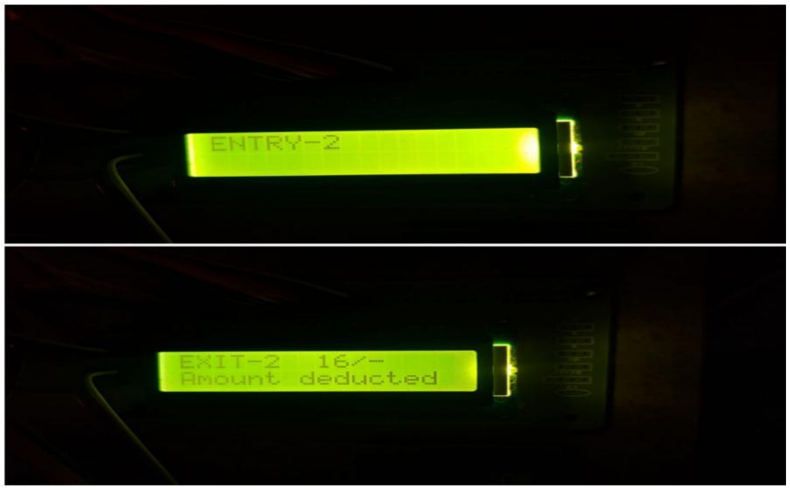
- Requires regular maintanance

CHAPTER-6

RESULT



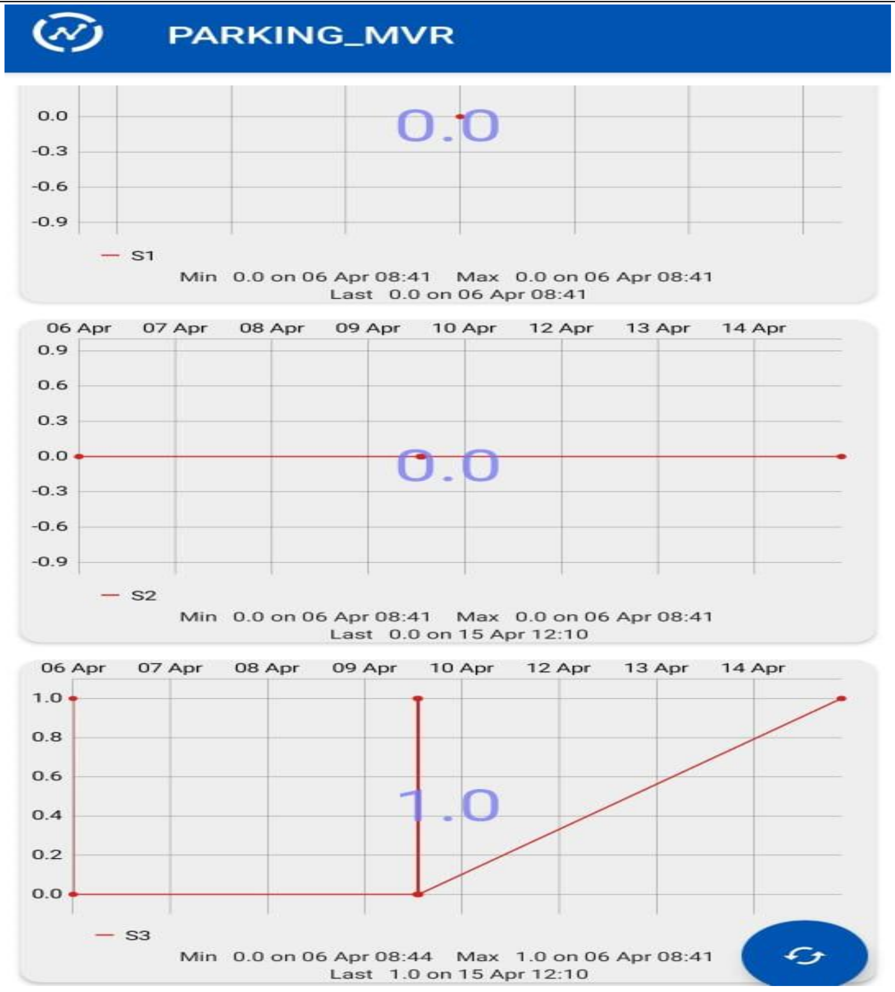
This is the prototype of our project(Iot established agile parking system)



```
ENTRY-2
```

```
EXIT-2  16/-
Amount deducted
```

The above pictures depicts that entry and exits of car into the slots,whenever car enters,it will displays in LCD as above based on the RFID tag.


PARKING_MVR

S1
Min 0.0 on 06 Apr 08:41 Max 0.0 on 06 Apr 08:41
Last 0.0 on 06 Apr 08:41

S2
Min 0.0 on 06 Apr 08:41 Max 0.0 on 06 Apr 08:41
Last 0.0 on 15 Apr 12:10

S3
Min 0.0 on 06 Apr 08:44 Max 1.0 on 06 Apr 08:41
Last 1.0 on 15 Apr 12:10

In the above picture it shows graph strike in the slot S3, It indicates that , the slot 3 has been occupied. So, whenever the slot has booked it will show as above picture

CHAPTER-7

CONCLUSION

In conclusion, the proposed IoT-based smart parking system offers several benefits over the traditional parking systems, such as improved user experience, real-time monitoring and optimization, automated payment, and data analytics. The use of IoT technologies, such as RFID, IR sensors, servo motors, LCD display, WiFi, and cloud computing, makes the system smart, efficient, and reliable. The system is user-friendly, and the mobile app provides an easy and convenient way to reserve and pay for the parking spot. The system is scalable and can be implemented in parking lots of various sizes and types, such as commercial parking lots, shopping malls, airports, and hospitals, among others.

CHAPTER-8

FUTURE SCOPE

The proposed system can be further enhanced in various ways to improve its functionalities and usability. Some potential future scope areas are as follows:

- Integration with AI and Machine Learning: The system can be integrated with AI and machine learning algorithms to predict parking spot availability, dynamically allocate parking spots, and optimize the parking space utilization.
- Integration with Autonomous Vehicles: With the emergence of autonomous vehicles, the parking system can be integrated with the autonomous vehicle's navigation system to enable autonomous parking, which will further improve the parking experience.
- Integration with Electric Vehicle Charging Stations: The system can be integrated with electric vehicle charging stations to enable users to reserve and pay for the parking spot and charging simultaneously.
- Integration with Smart City Infrastructure: The system can be integrated with the smart city infrastructure to provide real-time traffic information and suggest the optimal route to the parking lot.
- Integration with Blockchain Technology: The system can be integrated with blockchain technology to provide secure and transparent payment transactions and ensure data privacy.

Overall, the proposed system has immense potential for further development and can be integrated with various emerging technologies to enhance its functionalities and make it more smart, efficient, and sustainable.

REFERENCES:

1. P. Kumar, P. Kumar, R. Jha, and R. Jha, *"IoT based smart parking system: A review,"* in 2020 4th International Conference on Inventive Computation Technologies (ICICT), 2020, pp. 712-715.
2. H. M. Yousuf, M. R. Hossain, and M. I. Hossain, "An intelligent IoT-based parking management system using raspberry pi," in 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pp. 1-6.
3. M. M. Rashid, M. N. Uddin, M. R. Islam, and A. I. A. Rahman, "Development of IoT-based smart parking system using Raspberry Pi," in 2019 22nd International Conference on Computer and Information Technology (ICCIT), 2019, pp. 1-6.
4. K. M. Kafi, S. U. Ahmed, and T. Mahmud, "IoT based smart parking system using RFID technology," in 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2017, pp. 1-5.
5. A. Ahmed, A. M. A. Haidar, and M. Al-Rakhami, "An IoT-based smart parking system using IR sensors," in 2017 International Conference on Computational Science and Computational Intelligence (CSCI), 2017, pp. 1389-1392.
6. A. Roy, A. Majumdar, and D. K. Bhattacharyya, "RFID and IoT based smart parking system using Arduino," in 2018 International Conference on Computing, Power and Communication Technologies (GUCON), 2018, pp. 20-24.
7. M. A. Sayeed, S. S. Islam, S. S. Jahan, and M. R. Haque, "IoT-based smart parking management system using ultrasonic sensor," in 2020 23rd International Conference on Computer and Information Technology (ICCIT), 2020, pp. 1-6.
8. N. N. Hoque, N. Z. Khan, and M. T. Rahman, "IoT based smart parking management system using ultrasonic sensor," in 2020 International Conference on Innovation in Engineering and Technology (ICIET), 2020, pp. 1-6.
9. N. Chowdhury, N. B. M. Kamrul Hasan, and M. S. Rahman, "IoT-based smart parking management system using Android application," in 2018 IEEE International Conference on Innovations in Science, Engineering and Technology (ICISET), 2018, pp. 1-5.

ARDUINO CODE

```
#include <LiquidCrystal.h>
#include<Servo.h>
#include <SoftwareSerial.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
Servo myservo;
Servo myservo1;
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.
int cnt=0;
int buz = 8;
int s1 = A1;
int s2 = A2;
int s3 = A3;


 int pos;
int s1s=0;
int s2s=0;
int s3s=0;


int am1=100;
int am2=100;
int am3=100;
int vhno,ts1,ts2,ts3,tm,vhn=0;
int x;
 int bk1=0;
    int bk2=0;
    int bk3=0;
    int bk4=0;
 int xx=48;
void setup() {

Serial.begin(9600);
lcd.begin(16,2);
lcd.print("WELCOME");
  SPI.begin();      // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522
pinMode(s1,INPUT);
pinMode(s2,INPUT);
pinMode(s3,INPUT);

pinMode(buz,OUTPUT);
myservo.attach(A0);

delay(1000);

for (pos = 90; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'

    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

```
void gate()
{

    for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 180 degrees
  // in steps of 1 degree
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                       // waits 15ms for the servo to reach the position
 }
 delay(5000);
 for (pos = 90; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                       // waits 15ms for the servo to reach the position
 }
}



void loop()
{
  lcd.clear();
  lcd.print("WELCOME");

  int sl1=1-digitalRead(s1);
  int sl2=1-digitalRead(s2);
  int sl3=1-digitalRead(s3);

  if(Serial.available())
  {
   xx=Serial.read();
```

```
     delay(1);
  }

  if(xx=='1')
  {
    bk1=1;
    bk2=0;
    bk3=0;
    bk4=0;
  }

   if(xx=='2')
  {
    bk1=0;
    bk2=1;
    bk3=0;
    bk4=0;
  }

      if(xx=='3')
  {
    bk1=1;
    bk2=1;
    bk3=0;
    bk4=0;
  }

      if(xx=='4')
  {
   bk1=0;
   bk2=0;
   bk3=1;
   bk4=0;
  }

    if(xx=='5')
  {
   bk1=1;
   bk2=0;
   bk3=1;
   bk4=0;
  }

    if(xx=='6')
  {
   bk1=0;
   bk2=1;
   bk3=1;
   bk4=0;
  }

    if(xx=='7')
  {
   bk1=1;
   bk2=1;
   bk3=1;
   bk4=0;
  }
```

```
    if(xx=='0')
  {
   bk1=0;
   bk2=0;
   bk3=0;
   bk4=0;
  }

  if(bk1==1)
    sl1=2;
  if(bk2==1)
    sl2=2;
  if(bk3==1)
    sl3=2;



  lcd.setCursor(0,1);
  lcd.print("A:"+String(sl1) + " B:"+String(sl2) + " C:"+String(sl3) );
  if(cnt>5)
  {
   cnt=0;

  Serial.println(",0,"+String(sl1) + ","+String(sl2) + ","+String(sl3) + ","+"0");
  }
  scan_rfid();
  if(vhn>0)
  {
```

```
  if(sl1==0 || sl2==0 || sl3==0 )
  {
           if(vhn==1)
           {
              vhno=1;
              s1s=s1s+1;

              if(s1s==1)
              {
                lcd.clear();
                lcd.print("ENTRY-1");
                  ts1=millis();
                  gate();
              }
              if(s1s==2)
              {
                  s1s=0;

                  tm=(millis()-ts1)/1000;
                 lcd.clear();
                 lcd.print("EXIT-1  "+ String(tm)+"/-");
                  delay(1000);
                  if(am1>=tm)
                    {
                      am1=am1-tm;
                      lcd.setCursor(0,1);
                      lcd.print("Amount deducted");
                    }
                  else
```

```
        {

            lcd.clear();
            lcd.print("No Balance");
            digitalWrite(buz,1);
             delay(1000);
            digitalWrite(buz,0);
        }

        gate();
    }
}



if(vhn==2)
{
    vhno=2;
    s2s=s2s+1;

    if(s2s==1)
    {
      lcd.clear();
      lcd.print("ENTRY-2");
        ts2=millis();
        gate();
    }
    if(s2s==2)
    {
        s2s=0;
```

```
         tm=(millis()-ts2)/1000;
        lcd.clear();
        lcd.print("EXIT-2  "+ String(tm)+"/-");
         delay(1000);
         if(am2>=tm)
           {
              am2=am2-tm;
              lcd.setCursor(0,1);
              lcd.print("Amount deducted");
           }
         else
         {

             lcd.clear();
             lcd.print("No Balance");
             digitalWrite(buz,1);
              delay(1000);
             digitalWrite(buz,0);
         }

         gate();
     }
 }



 if(vhn==3)
 {
     vhno=3;
```

```
s3s=s3s+1;


if(s3s==1)
{
  lcd.clear();
  lcd.print("ENTRY-3");
    ts3=millis();
    gate();
}
if(s3s==2)
{
    s3s=0;


    tm=(millis()-ts3)/1000;
   lcd.clear();
   lcd.print("EXIT-3  "+ String(tm)+"/-");
    delay(1000);
    if(am3>=tm)
      {
         am3=am3-tm;
         lcd.setCursor(0,1);
         lcd.print("Amount deducted");
      }
    else
     {


        lcd.clear();
        lcd.print("No Balance");
        digitalWrite(buz,1);
```

```
                    delay(1000);
                   digitalWrite(buz,0);
                 }

                 gate();
               }
             }



   }

          else
          {

             lcd.clear();
             lcd.print(" Parking Full");
             digitalWrite(buz,1);
             delay(2000);
              digitalWrite(buz,0);
          }
vhn=0;
   }
   delay(500);
   cnt=cnt+1;
}

void scan_rfid()
 {
  if ( ! mfrc522.PICC_IsNewCardPresent())
 {
  return;
 }

 if ( ! mfrc522.PICC_ReadCardSerial())
 {
  return;
 }

 //Serial.print("UID tag :");
 String content= "";
 byte letter;
 for (byte i = 0; i < mfrc522.uid.size; i++)
 {
  // Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
   //Serial.print(mfrc522.uid.uidByte[i], HEX);
   content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
   content.concat(String(mfrc522.uid.uidByte[i], HEX));
 }
 //Serial.println();
 //Serial.print("Message : ");
 content.toUpperCase();
// Serial.println(content.substring(1));
 if(content.substring(1)=="EB 33 89 2B")
 {
  vhn=1;
 }
```

```
  if(content.substring(1)=="7B 73 1D 85")
 {
 vhn=2;
 }

  if(content.substring(1)=="CC 59 1D 85")
 {
  vhn=3;
 }

  if(content.substring(1)=="81 A6 57 40")
 {
  vhn=4;
 }
 delay(500);

}
```