ABSTRACT

In agricultural regions, animal intrusion into crop fields poses a significant threat to food security and farmers' livelihoods. Traditional methods of deterring animals are often labor-intensive, ineffective over long periods, or harmful to the animals and environment. This project proposes a smart, humane, and efficient solution to mitigate the problem using cutting-edge technology. We aim to develop an automated system that utilizes the YOLOv5 algorithm for real-time animal detection, enabling immediate action through non-lethal repellent measures. Upon detection, the system will notify farmers via mobile notifications and activate a series of deterrents: LED lights, auditory warnings, and physical movements, all designed to safely scare away the intruders without causing them harm.

CHAPTER-1

1.1 INTRODUCTION

In agricultural regions, the intrusion of animals into crop fields poses a significant threat to food security and farmers' livelihoods. Traditional methods of deterring animals are often labor-intensive, ineffective over long periods, or harmful to the animals and environment. To address this challenge, this project proposes a smart, humane, and efficient solution leveraging cutting-edge technology.

The project aims to develop an automated system that utilizes the YOLOv5 algorithm for real-time animal detection, enabling immediate action through non-lethal repellent measures. Upon detecting intruding animals, the system will notify farmers via mobile notifications and activate a series of deterrents, including LED lights, auditory warnings, and physical movements. These deterrents are designed to safely scare away intruders without causing harm to them.By combining advanced computer vision with non-lethal deterrent measures, the proposed system offers a humane and effective solution to mitigate animal intrusion in crop fields. It not only enhances food security and protects farmers' livelihoods but also promotes coexistence between humans and wildlife while minimizing environmental impact.

PROBLEM STATEMENT:

In agricultural regions, the intrusion of animals into crop fields poses a significant threat to food security and farmers' livelihoods. Traditional methods of deterring animals, such as fences and scarecrows, are often labor-intensive, ineffective over long periods, or harmful to the animals and environment. Additionally, manual monitoring for animal intrusion is time-consuming and may not provide timely intervention, leading to crop damage and financial losses for farmers.

Furthermore, the indiscriminate use of lethal measures to deter animals can have detrimental effects on wildlife populations and ecosystems, leading to imbalances in the natural environment. There is a pressing need for a smarter, more humane, and efficient solution to mitigate the problem of animal intrusion in agricultural areas.

Therefore, the problem statement revolves around the following key challenges:

Ineffective Traditional Methods: Existing methods of deterring animals, such as fences and scarecrows, are often inadequate in providing long-term protection and may not effectively deter all types of animals.Labor-Intensive Monitoring: Manual monitoring for animal intrusion is labor-intensive and time-consuming, leading to delays in detecting and addressing the problem.

Harmful Deterrent Measures: The use of lethal or harmful deterrent measures can have adverse effects on wildlife and the environment, leading to imbalances in ecosystems.

Crop Damage and Financial Losses: Animal intrusion can result in significant crop damage and financial losses for farmers, impacting food security and livelihoods in agricultural communities.

Addressing these challenges requires the development of a smart, humane, and efficient solution that leverages cutting-edge technology to detect animal intrusion in real-time and implement non-lethal deterrent measures effectively. By mitigating the problem of animal intrusion, such a solution can enhance food security, protect farmers' livelihoods, promote coexistence between humans and wildlife, and minimize environmental impact in agricultural regions.

OBJECTIVES:

Develop Automated Animal Detection System: Design and develop a system capable of automated animal detection using the YOLOv5 algorithm or similar computer vision techniques. This system should accurately identify intruding animals in real-time.

Implement Non-Lethal Deterrent Measures: Integrate non-lethal deterrent measures into the system, including LED lights, auditory warnings, and physical movements. Ensure these measures are effective in safely scaring away intruding animals without causing harm. Enable Immediate Action: Enable the system to trigger deterrent measures immediately upon detecting intruding animals, minimizing crop damage and financial losses for farmers. Mobile Notification Integration: Implement a mobile notification system to alert farmers in real-time when animal intrusion is detected. Ensure that farmers receive timely notifications to take appropriate action.

EXISTING METHODS AND DRAWBACKS:

Physical Barriers:

Method: Fences, barriers, and scarecrows are commonly used to deter animals from entering crop fields.

Drawbacks: While effective to some extent, physical barriers require significant maintenance, may be costly to install and maintain, and can be breached by determined or large animals. Additionally, they can disrupt wildlife movement and contribute to habitat fragmentation.

Chemical Deterrents:

Method: Chemical deterrents such as pesticides or repellents are sometimes used to deter animals from crop fields.

Drawbacks: Chemical deterrents can have harmful effects on both wildlife and the environment. They may also require frequent reapplication and can be expensive.

Scare Tactics:

Method: Loud noises, flashing lights, and scare devices are used to frighten animals away from crop fields.

Drawbacks: While scare tactics can be effective initially, animals may become habituated to them over time, rendering them less effective. Additionally, loud noises can disturb nearby communities and may not be suitable for all environments.

Trapping and Removal:

Method: Traps or snares are used to capture and remove animals from crop fields.

Drawbacks: Trapping and removal methods can be labor-intensive, time-consuming, and may not be practical for large or elusive animals. They can also be stressful or harmful to the captured animals if not handled properly.

Predator Introduction:

Method: Introducing natural predators of crop-damaging animals into the environment to control their populations.

Drawbacks: Introducing predators can disrupt ecosystems and may not be feasible in all environments. It can also be difficult to control the behavior and impact of introduced predators.

Guard Animals:

Method: Using guard animals such as dogs or llamas to patrol crop fields and deter intruding animals.

Drawbacks: While effective in some cases, guard animals require training, care, and may not be suitable for all types of crops or environments. They can also be susceptible to predation themselves.

CHAPTER-2

2.1 PROPOSED SYSTEM

Proposed System Description:

The proposed system is an innovative solution designed to mitigate the problem of animal intrusion in crop fields using advanced technology. It integrates deep learning with hardware components to create a smart and effective deterrent system. The system consists of a laptop running the YOLOv5 algorithm for real-time animal detection, which communicates with an Arduino Uno microcontroller. Upon detecting wild animals, the system sends notifications to farmers via Telegram and triggers a series of deterrents, including Neopixel lights, a DF Mini MP3 player for playing scaring sounds, and a servo motor for physical movement to scare animals away.

Animal Detection using YOLOv5 Algorithm:

The system utilizes the YOLOv5 algorithm, a state-of-the-art deep learning model, to detect wild animals in real-time. The algorithm runs on a laptop equipped with a webcam or external camera for video input.

Communication with Farmers via Telegram:

Upon detecting animals, the system sends immediate notifications to farmers via Telegram, a messaging platform. This allows farmers to receive real-time alerts about animal intrusion and take necessary action.

Activation of Deterrents using Arduino Uno:

The laptop communicates with an Arduino Uno microcontroller, which controls the activation of deterrent devices. The Arduino Uno is connected to various hardware components, including Neopixel lights, a DF Mini MP3 player, and a servo motor.

Neopixel Lights:

Neopixel lights are RGB LED lights capable of emitting a wide range of colors and patterns. Upon receiving signals from the Arduino Uno, the Neopixel lights illuminate in a bright and attention-grabbing manner, intended to scare away intruding animals.

DF Mini MP3 Player:

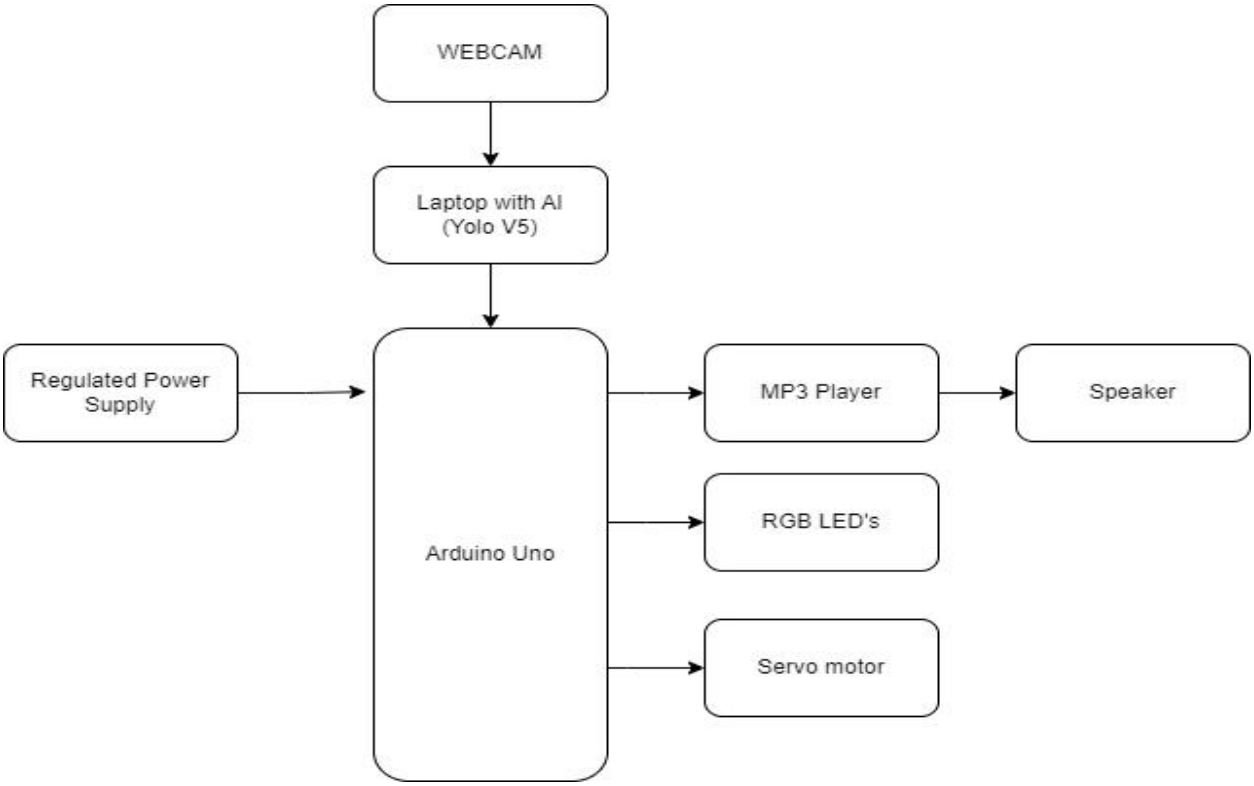
The DF Mini MP3 player is programmed to play scaring sounds or loud noises upon receiving signals from the Arduino Uno. These sounds, such as predator calls or distress calls, are designed to deter animals from approaching the crop fields.

Servo Motor:

A servo motor is used to create physical movement or motion to further scare away animals. For example, the servo motor can be attached to a scarecrow or other scare device, causing it to move unpredictably and intimidate intruding animals.

By combining advanced deep learning techniques with hardware components, the proposed system offers a comprehensive and effective solution to mitigate animal intrusion in crop fields. It provides farmers with real-time alerts and activates deterrents to scare away animals without causing harm, ultimately protecting crops and promoting coexistence between humans and wildlife.

2.2 BLOCK DIAGRAM:



IMPLEMENTATION

Setting Up YOLOv5 on Laptop:

Install the necessary software and dependencies to run YOLOv5 on the laptop.

Download and configure the YOLOv5 model for real-time animal detection.

Test the YOLOv5 algorithm with sample video feeds to ensure proper functioning.

Integrating Telegram Notification:

Set up a Telegram bot and obtain the API token.

Implement code on the laptop to send notifications to farmers via the Telegram bot when animals are detected.

Test the notification system to ensure timely and accurate alerts.

Configuring Arduino Uno:

Set up the Arduino Uno microcontroller and install the necessary development environment (e.g., Arduino IDE).

Write Arduino code to receive signals from the laptop and control the various deterrent devices.

Test the communication between the laptop and Arduino Uno to ensure proper data transmission.

Connecting and Programming Neopixel Lights:

Connect Neopixel lights to the Arduino Uno according to the hardware specifications.

Write Arduino code to control the Neopixel lights and trigger them upon receiving signals from the laptop.

Test the Neopixel lights to ensure they illuminate correctly and produce the desired effects.

Setting Up DF Mini MP3 Player:

Connect the DF Mini MP3 player to the Arduino Uno and configure it to play scaring sounds.

Write Arduino code to control the DF Mini MP3 player and play sounds upon receiving signals from the laptop.

Test the DF Mini MP3 player to ensure it plays the desired sounds effectively.

Installing and Configuring Servo Motor:

Mount the servo motor in a suitable location near the crop fields.

Connect the servo motor to the Arduino Uno and program it to move in response to signals from the laptop.Test the servo motor to ensure it moves smoothly and effectively scares away animals.

ADVANTAGES AND APPLICATIONS

Efficient Animal Detection: The use of the YOLOv5 algorithm enables efficient and accurate detection of intruding animals in real-time, allowing for immediate action to be taken.

Timely Alerts: The integration of Telegram notifications ensures that farmers receive timely alerts on their mobile devices when animal intrusion is detected, enabling them to respond promptly.

\Non-Lethal Deterrents: The system utilizes non-lethal deterrent measures such as Neopixel lights, scaring sounds, and servo motor movements, minimizing harm to animals while effectively deterring them from crop fields.Humane Solution: By employing non-lethal deterrents and prioritizing the safety and well-being of both farmers and wildlife, the system offers a humane solution to mitigate animal intrusion.

Cost-Effective: Compared to traditional methods such as physical barriers or chemical deterrents, the proposed system can be more cost-effective in the long term, requiring less maintenance and reducing the need for expensive deterrent measures.

Environmental Sustainability: The use of non-lethal deterrents and the reduction of chemical use contribute to environmental sustainability and minimize the impact on wildlife and ecosystems.

Applications:

Agricultural Protection: The system is primarily designed for use in agricultural regions to protect crops from animal intrusion. It can be applied to various types of crops, including fruits, vegetables, grains, and cash crops.

Wildlife Conservation: In addition to protecting crops, the system can also be used in wildlife conservation efforts to deter animals from entering protected areas or sensitive habitats.

Urban Gardens and Orchards: The system can be implemented in urban gardens, orchards, and community gardens to prevent damage from urban wildlife such as deer, rabbits, and birds.

Forestry and Plantations: The system can be adapted for use in forestry and plantations to protect tree seedlings and saplings from browsing animals, helping to ensure successful reforestation efforts.

Vineyards and Wineries: In vineyards and wineries, the system can be deployed to deter animals such as deer and birds, which can cause significant damage to grapevines and impact wine production.

Research and Monitoring: The system can also have applications in wildlife research and monitoring, providing researchers with valuable data on animal behavior and movement patterns in agricultural landscapes.

Overall, the proposed system offers a versatile and effective solution for mitigating animal intrusion in various agricultural and environmental settings, promoting coexistence between humans and wildlife while safeguarding crops and natural resources.

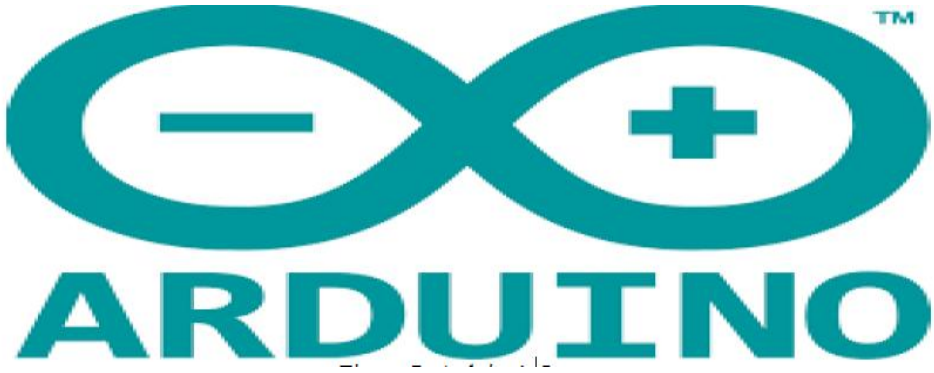
CHAPTER-3

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 preassembled; 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.

Figure 7- Arduino's



Why 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.

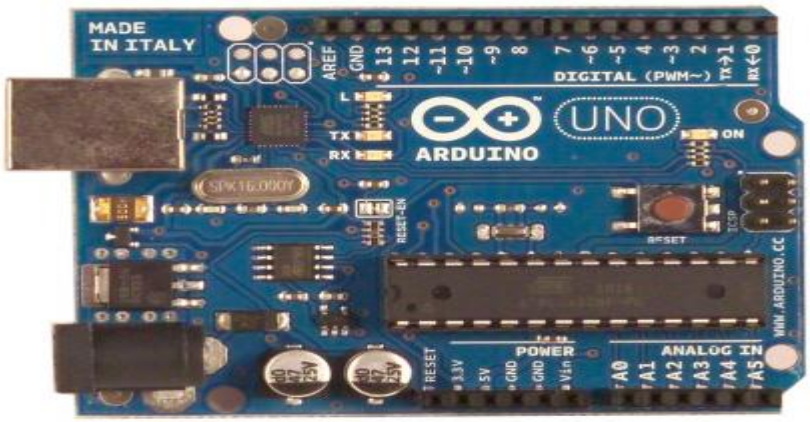
•Open source and extensible programming. The Arduino program language is available as open source, available for development by experienced engineers. The lingo can be reached out through C++ libraries, and people expecting to understand the specific purposes of different interests can make the leap 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.

3.1.2 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;



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

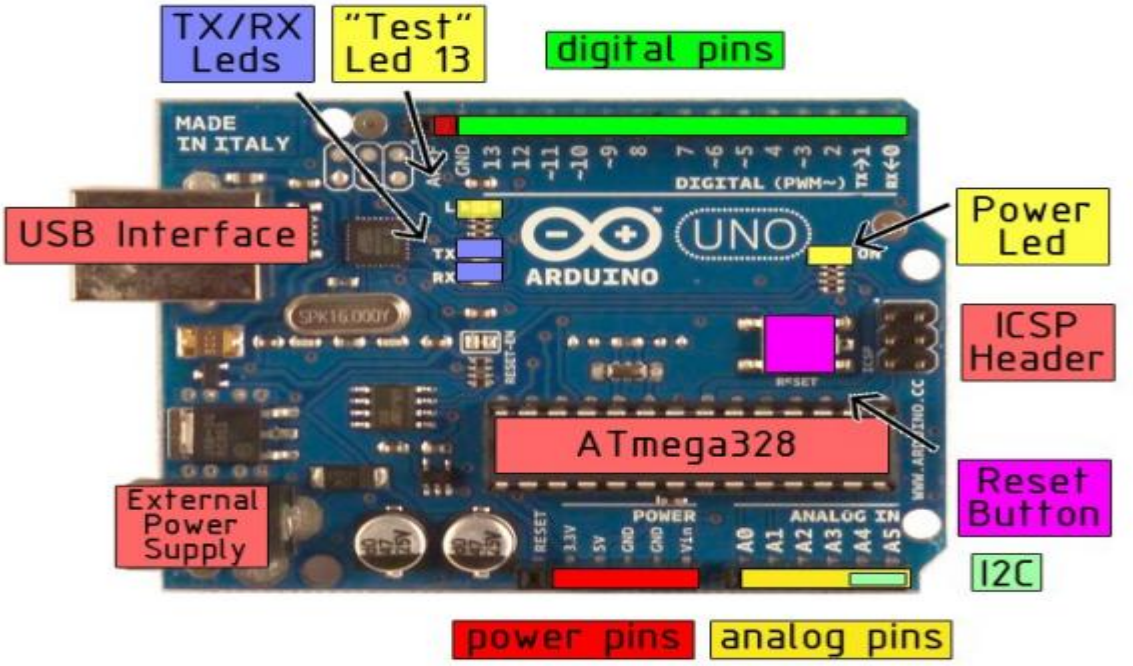
32 KB of which 0.5 KB used by

bootloader

SRAM 2 KB

EEPROM 1 KB

Clock Speed 16 MHz



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.

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).

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).

• I 2C: 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 analogReference().

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

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 SoftwareSerial 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.

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.

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

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.

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.

DFPlayer Mini MP3 Player

The **DFPlayer Mini MP3 Player For Arduino** is a small and low cost MP3 module with an simplified output directly to the speaker. The module can be used as a stand alone module with attached battery, speaker and push buttons or used in combination with microcontrollers such as Arduino, ESP32, Raspberry Pi and any microcontrollers with Uart.

Specification

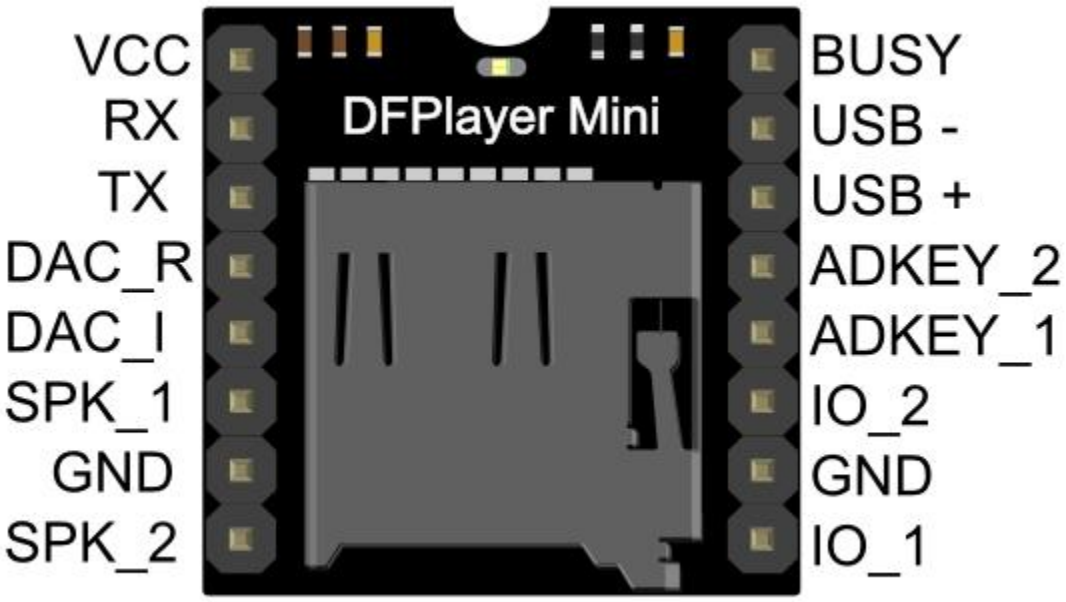
- Sampling rates (kHz): 8/11.025/12/16/22.05/24/32/44.1/48
- 24 -bit DAC output, support for dynamic range 90dB , SNR support 85dB
- Fully supports FAT16 , FAT32 file system, maximum support 32G of the TF card, support 32G of U disk, 64M bytes NORFLASH
- A variety of control modes, I/O control mode, serial mode, AD button control mode
- advertising sound waiting function, the music can be suspended. when advertising is over in the music continue to play

- audio data sorted by folder, supports up to 100 folders, every folder can hold up to 255 songs
- 30 level adjustable volume, 6 -level EQ adjustable

Application

- Car navigation voice broadcast;
- Road transport inspectors, toll stations voice prompts;
- Railway station, bus safety inspection voice prompts;
- Electricity, communications, financial business hall voice prompts;
- Vehicle into and out of the channel verify that the voice prompts;
- The public security border control channel voice prompts;
- Multi-channel voice alarm or equipment operating guide voice;
- The electric tourist car safe driving voice notices;
- Electromechanical equipment failure alarm;
- Fire alarm voice prompts;
- The automatic broadcast equipment, regular broadcast.

Pin Map



Pin	Description	Note
VCC	Input Voltage	DC3.2~5.0V;Type: DC4.2V
RX	UART serial input	
TX	UART serial output	
DAC_R	Audio output right channel	Drive earphone and amplifier
DAC_L	Audio output left channel	Drive earphone and amplifier
SPK2	Speaker-	Drive speaker less than 3W
GND	Ground	Power GND
SPK1	Speaker+	Drive speaker less than 3W
IO1	Trigger port 1	Short press to play previous（long press to decrease volume）
GND	Ground	Power GND
IO2	Trigger port 2	Short press to play next（long press to increase volume）
ADKEY1	AD Port 1	Trigger play first segment
ADKEY2	AD Port 2	Trigger play fifth segment
USB+	USB+ DP	USB Port
USB-	USB- DM	USB Port
BUSY	Playing Status	Low means playing \High means no

3.3 Work Mode

1. Serial Mode

Support for asynchronous serial communication mode via PC serial sending commands
Communication Standard:9600 bps Data bits :1 Checkout :none Flow Control :none

- Instruction Description

Format: $S VER Len CMD Feedback para1 para2 checksum $O		
$S	Start bit 0x7E	Each command feedback begin with $, that is 0x7E
VER	Version	Version Information
Len	the number of bytes after "Len"	Checksums are not counted
CMD	Commands	Indicate the specific operations, such as play / pause, etc.
Feedback	Command feedback	If need for feedback, 1: feedback, 0: no feedback
para1	Parameter 1	Query high data byte
para2	Parameter 2	Query low data byte
checksum	Checksum	Accumulation and verification [not include start bit $]
$O	End bit	End bit 0xEF

For example, if we specify play NORFLASH, you need to send: 7E FF 06 09 00 00 04 FE EE EF
Data length is 6, which are 6 bytes [FF 06 09 00 00 04]. Not counting the start, end, and verification.

- Serial Control Cmd

CMD	Function Description	Parameters(16 bit)
0x01	Next	
0x02	Previous	
0x03	Specify tracking(NUM)	0-2999
0x04	Increase volume	
0x05	Decrease volume	
0x06	Specify volume	0-30
0x07	Specify EQ(0/1/2/3/4/5)	Normal/Pop/Rock/Jazz/Classic/Base
0x08	Specify playback mode (0/1/2/3)	Repeat/folder repeat/single repeat/ random
0x09	Specify playback source(0/1/2/3/4)	U/TF/AUX/SLEEP/FLASH
0x0A	Enter into standby – low power loss	
0x0B	Normal working	
0x0C	Reset module	
0x0D	Playback	
0x0E	Pause	
0x0F	Specify folder to playback	1~10(need to set by user)
0x10	Volume adjust set	{DH＝1:Open volume adjust }{DL: set volume gain 0~31}
0x11	Repeat play	{1:start repeat play}{0:stop play}

- Serial Query Cmd

Commands	Function Description	Parameters(16 bit)
0x3C	STAY	
0x3D	STAY	
0x3E	STAY	
0x3F	Send initialization parameters	0 - 0x0F(each bit represent one device of the low-four bits)
0x40	Returns an error, request retransmission	
0x41	Reply	
0x42	Query the current status	
0x43	Query the current volume	
0x44	Query the current EQ	
0x45	Query the current playback mode	
0x46	Query the current software version	
0x47	Query the total number of TF card files	
0x48	Query the total number of U-disk files	
0x49	Query the total number of flash files	
0x4A	Keep on	
0x4B	Queries the current track of TF card	
0x4C	Queries the current track of U-Disk	
0x4D	Queries the current track of Flash	

3.4 NeoPixel LEDs

NeoPixel LED Strip
NeoPixel LED Pinout

The **NeoPixel LEDs** are RGB LED lights with a built driver IC that makes these lights addressable and programmable. The idea was originally coined by Adafruit and since then there are many types of Neo pixels of varying sizes and shapes available in the market. In this article

we will focus only on the strip type flexible Neo Pixels which are commonly available and most used. The **picture and pinouts of a NeoPixel LED** is shown above.

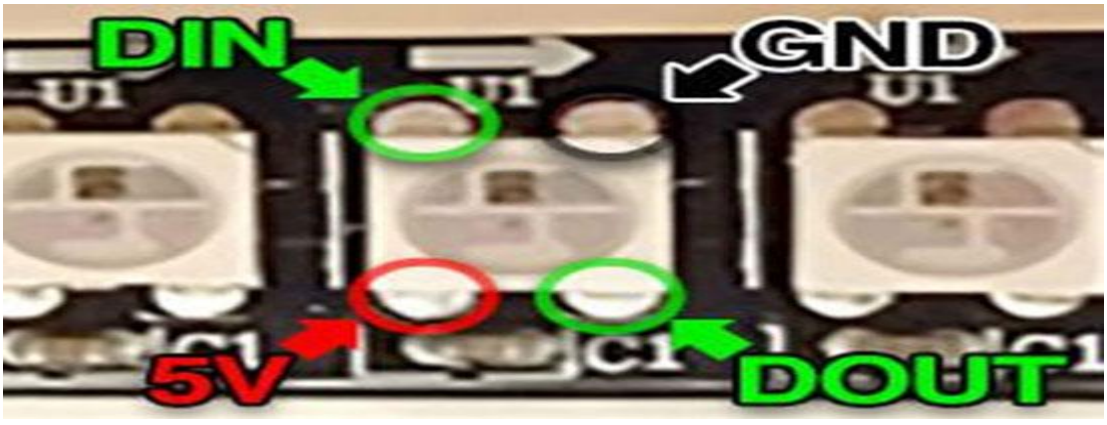
NeoPixel Pin Configuration

Normally the pin names are mentioned on the **NeoPixel LED strip** itself and the same is also listed below, but while making the connection make sure the arrow is faced upwards as shown in picture

Pin Number	Pin Name	Description
1	Ground	Connected to the ground of the circuit.
2	Data (Din)	This is the Data in pin which is provided with PWM signal
3	+5V	This powers the LED with 5V

Note: Due to different vendors the order of the pins can vary, hence verify it using the silk screen

Some older versions of LED did not have markings on the slick screen, in that case you have to solder to the LED terminals directly as shown below or use a multi-meter in connectivity mode to identify the pins



NeoPixel LED Features

- Individually addressable and programmable RGB LEDs
- Flexible and available in different form factors
- Operating voltage: 3.3V to 5V
- Power consumption: 60mA per LED at full brightness
- Communication: PWM through data pin
- Driver IC: WS2812
- Available in many different packages and form factors

Where to use Neo Pixels?

The Neo pixels are small in size with less circuitry and almost no messy wires since the driver IC is embedded into each LED. Each LED has a minimum of RGB light and hence they can be combined to get almost any colour of your choice. This makes it a very good choice for wearable electronics and other decorative lights.

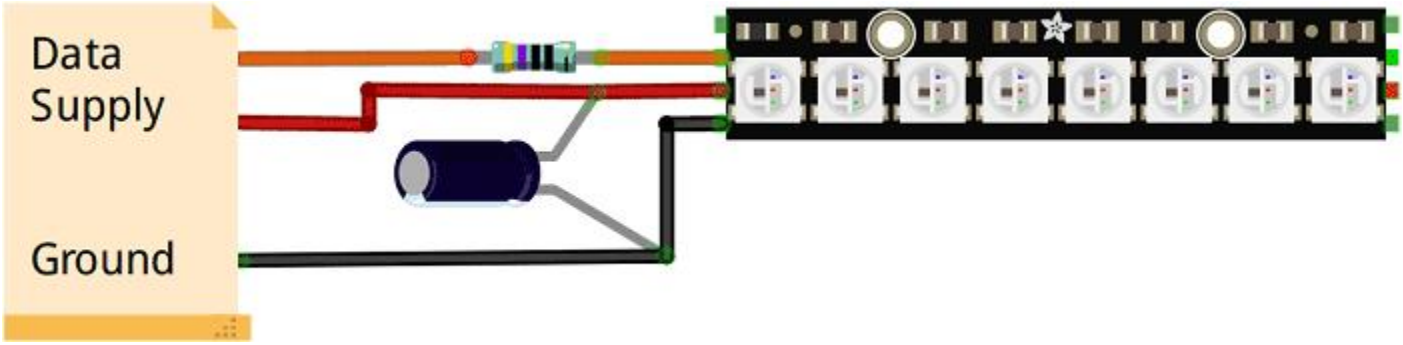
Each LED on this strip can be individually addressed and programmed, meaning each LED can be set to glow in a different colour. Also you can add as many leds in series as you like at control all of them with a single controller. I personally have seen LED strips having upto 200+ LEDS, the only constrain is the power and RAM of your controller. If you are using a controller like Arduino, then Adafruit has a library readily available using which these LED can be easily programmed. So if you are looking for easy to use programmable LED light then NeoPixels are what you are looking for.

How to use NeoPixel Strips?

The Neo Pixel Strips can be purchased in meters or as small ones with just 5 LEDs on it. They can be operated with a 5V power supply or form a 3.3V supply or even form li-po batteries of 3.7V since they can tolerate from 3.3V to 5V. But the data pin can read only 5V so a logic level shifter like 74HC245 IC has to be used. Each LED consume around 60mA so make sure your power supply can source enough current for all the LED.

Let us assume you are using a 5 LED RBG strip neo pixel with Arduino, and then in that case the LEDs can be directly powered by the 5V pin of Arduino since they can source upto 500mA.

Keep in mind that these LEDs are very sensitive in nature and hence have to be protected from voltage spikes and ESDs. A sample circuit diagram is shown below



A resistor of value 470 ohms should be used in series with data line (orange colour) and a capacitor of 1000uF and 6.3V should be added across the power rails (red and black) as shown above. These components are not mandatory to use but it is a good practise to protect the LED when working with more number of LEDs.

3.5 Servo Motor

Servo Motor SG90
Servo Motor Pinout (Wires)

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

TowerPro SG-90 Features

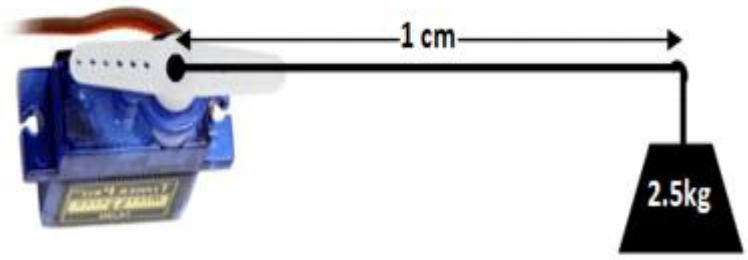
- Operating Voltage is +5V typically
- Torque: 2.5kg/cm
- Operating speed is 0.1s/60°
- Gear Type: Plastic
- Rotation : 0°-180°
- Weight of motor : 9gm
- Package includes gear horns and screws

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.

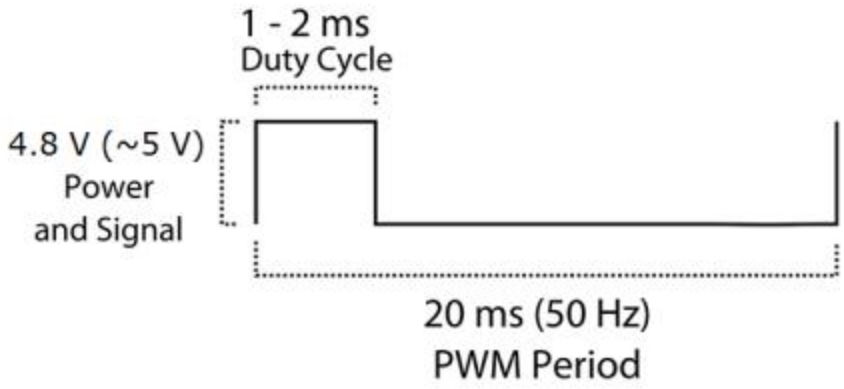
Most of the hobby Servo motors operates from 4.8V to 6.5V, the higher the voltage higher the torque we can achieve, but most commonly they are operated at +5V. Almost all hobby servo motors can rotate only from 0° to 180° due to their gear arrangement so make sure you project can live with the half circle 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.



How to use a 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.



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°

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

CHAPTER-4

4.1 SOFTWARE DESCRIPTION

4.1.1ARDUINO 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.

Writing 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.

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

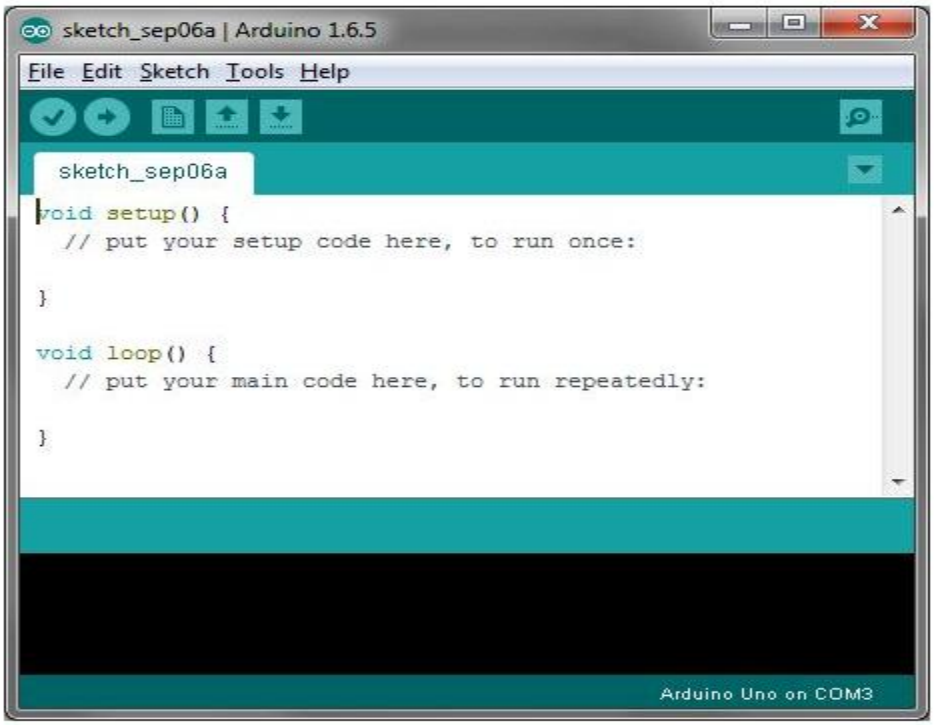
Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online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

4.1.2Arduino IDE: Initial Setup

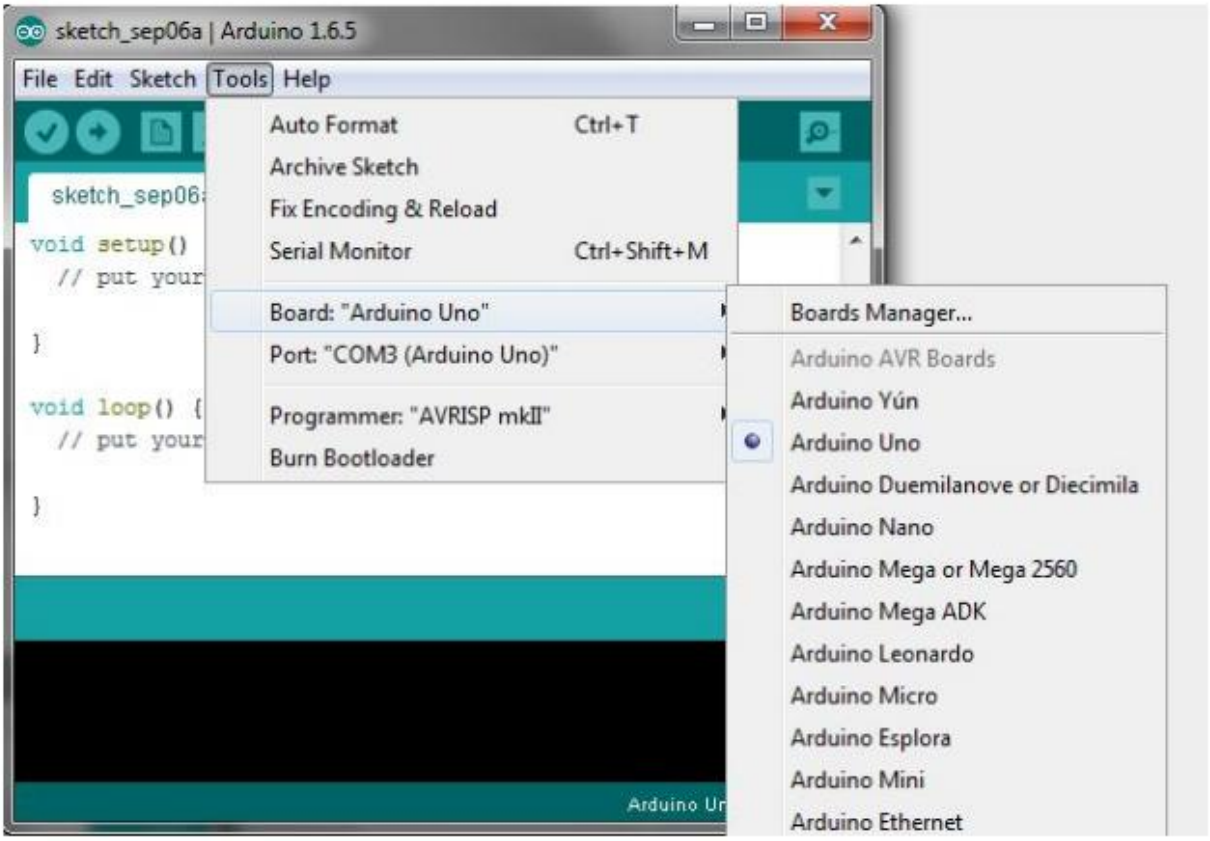
This is the Arduino IDE once it's been opened. It opens into a blank sketch where you can start programming immediately. First, we should configure the board and port settings to allow us to upload code. Connect your Arduino board to the PC via the USB cable.



Arduino IDE Default Window

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.

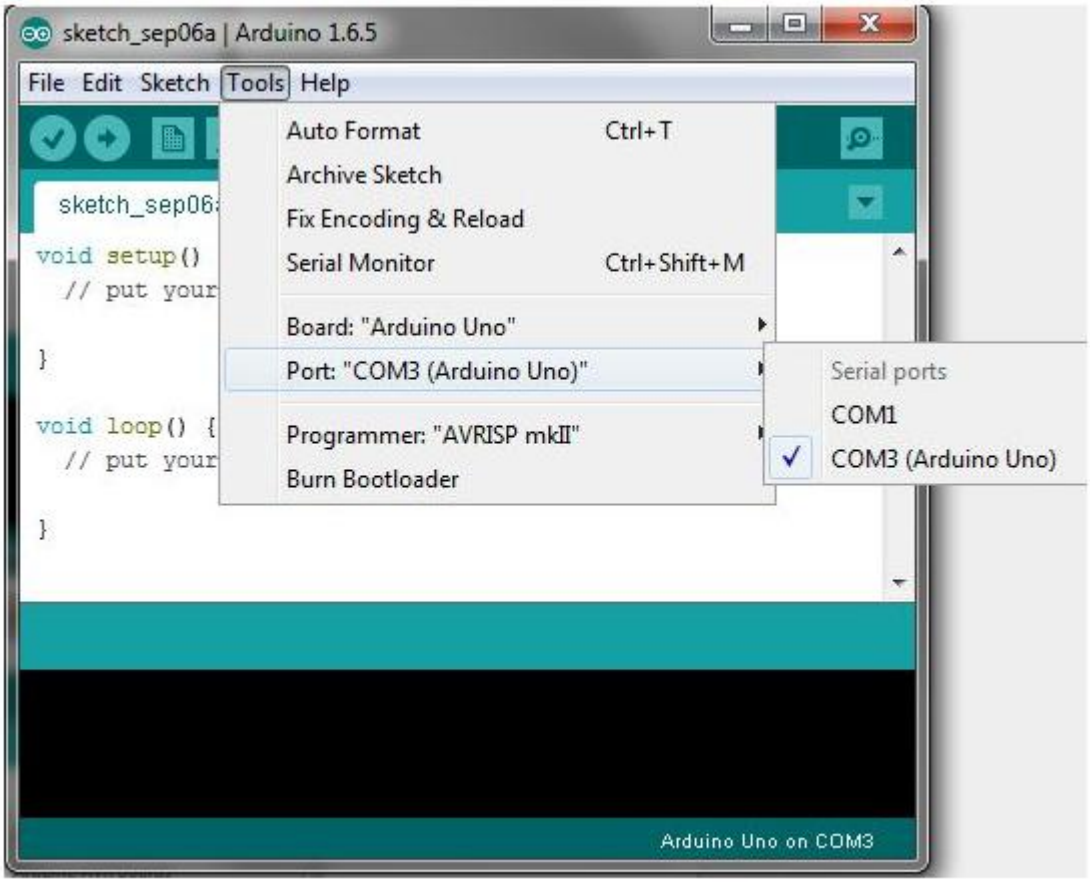


Arduino IDE: Board Setup Procedure

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.

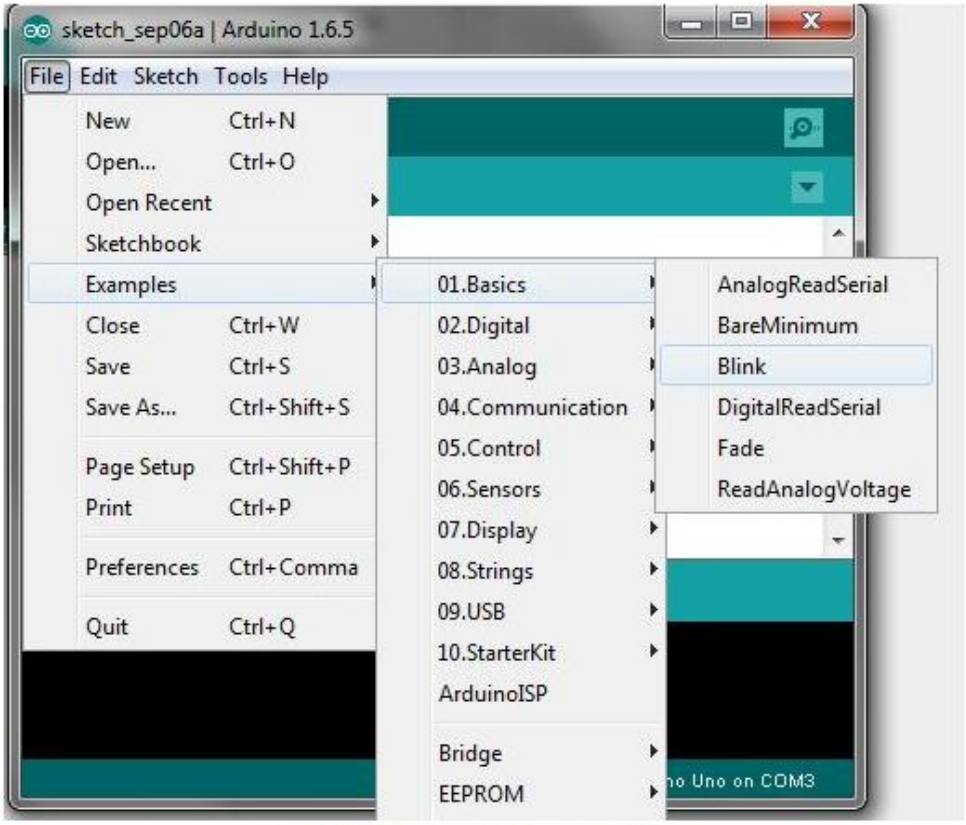


Arduino IDE: COM Port Setup

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.

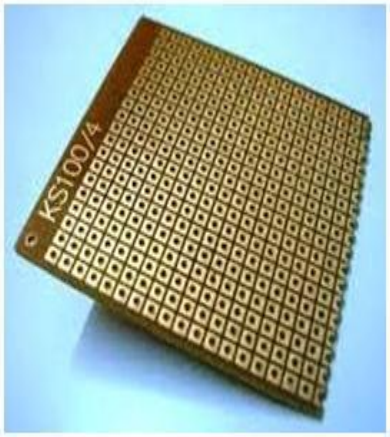
Upload Button: 

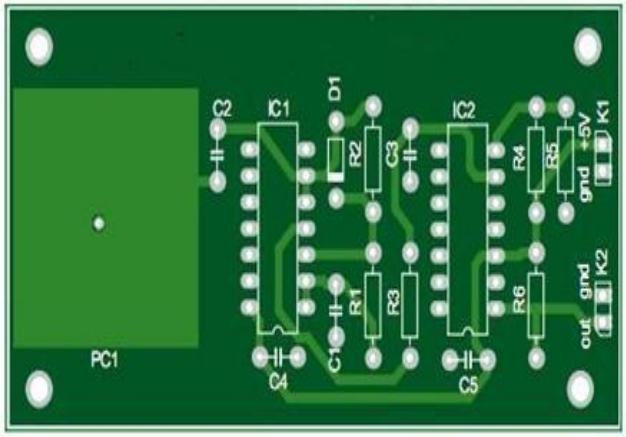


Arduino IDE: Loading Blink Sketch

4.2 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.





Dotted PCB

Layout PCB

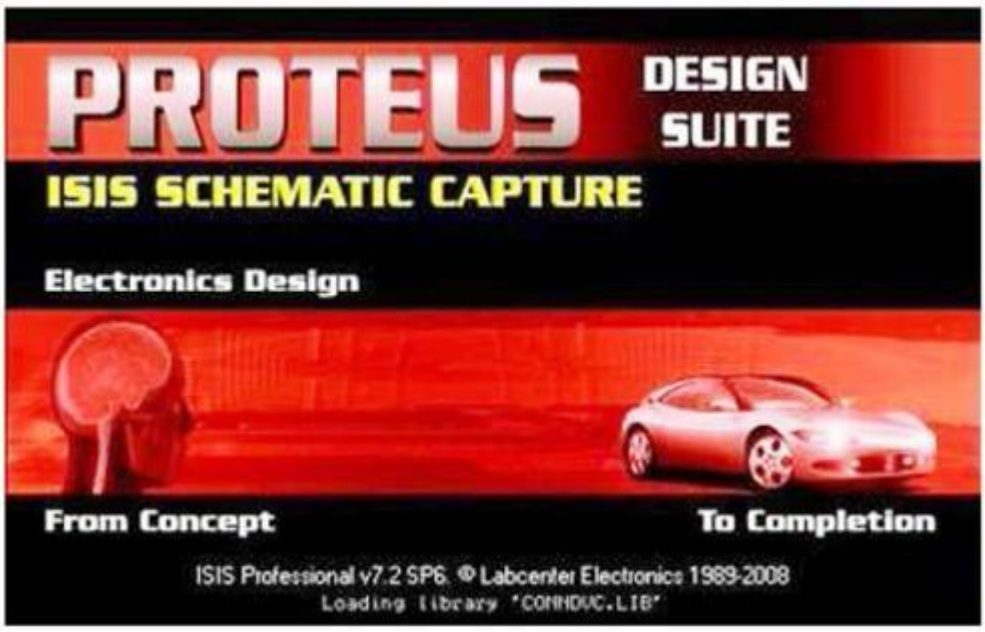
What is the main difference between the dotted PCB and layout PCB? In dotted PCB board only dots are available. According to our requirement we can place or insert the components in those holes and attach the components with wires and soldering lid. In this dotted PCB we can make the circuit as out wish 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.

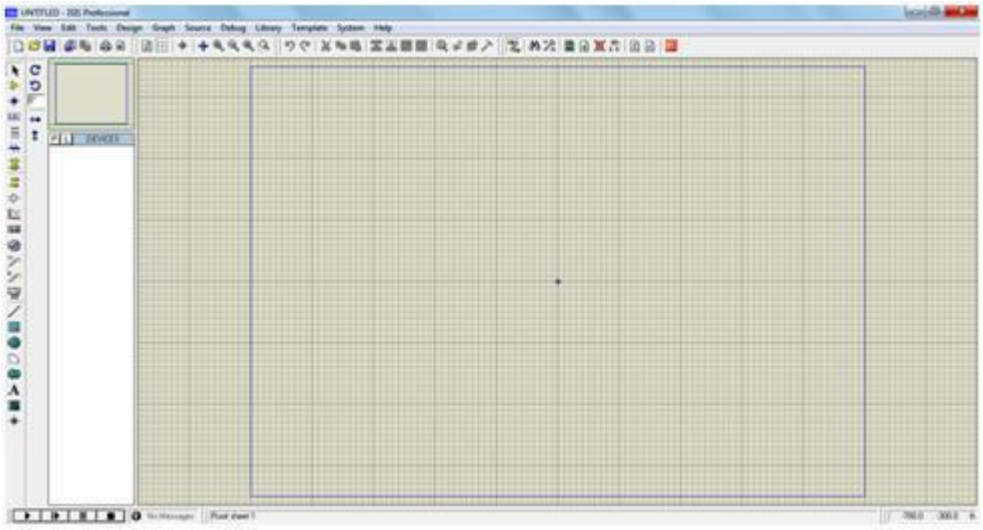
Introduction to Proteus:

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.

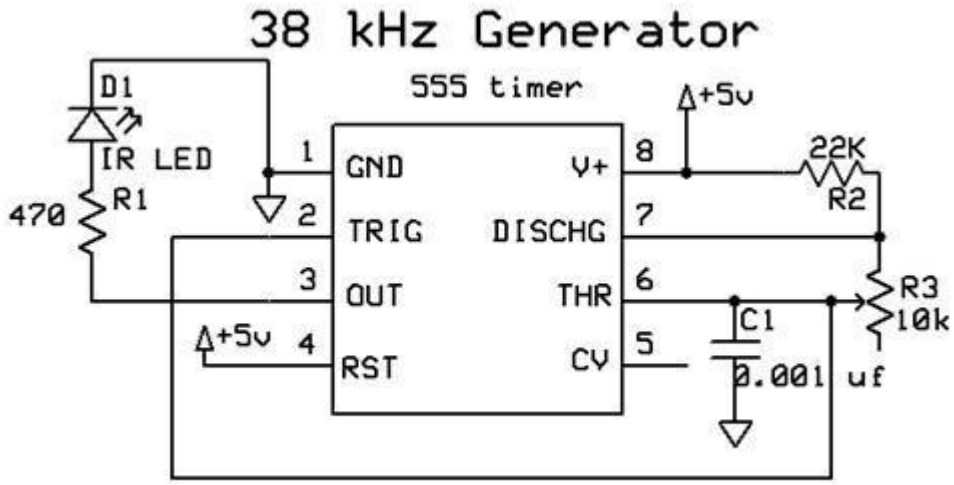
Up to know we have discussed about the basics and software description. Now we are entering into the designing section. Run the ISIS professional program by clicking the icon on the desktop, then this splash screen will appear.



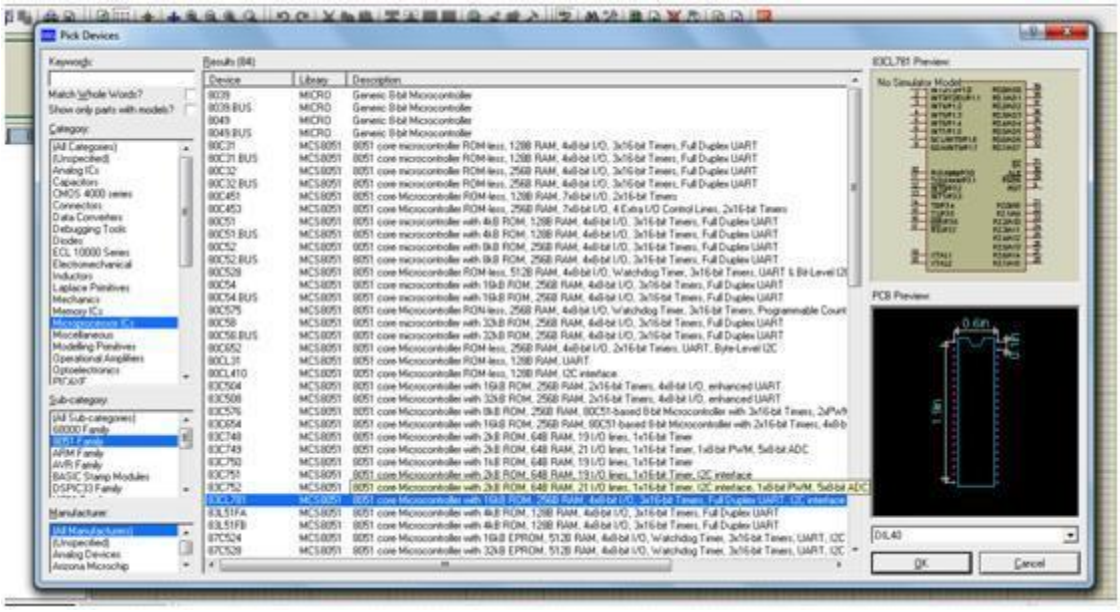
Next, a work space with interface buttons for designing circuit will appear as shown in figure below. Note that there is a blue rectangular line in the workspace; make sure that whole circuit is designed inside the rectangular space.



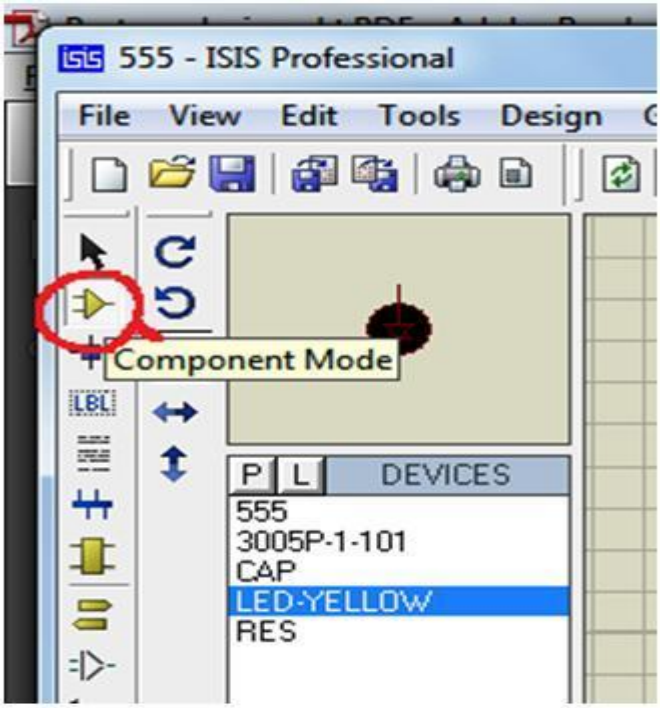
Next step is selecting the components to our required circuit. Let us take one example is designing of 38 kHz frequency generator by using 555 timer IC. The circuit diagram is shown in below image.



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.

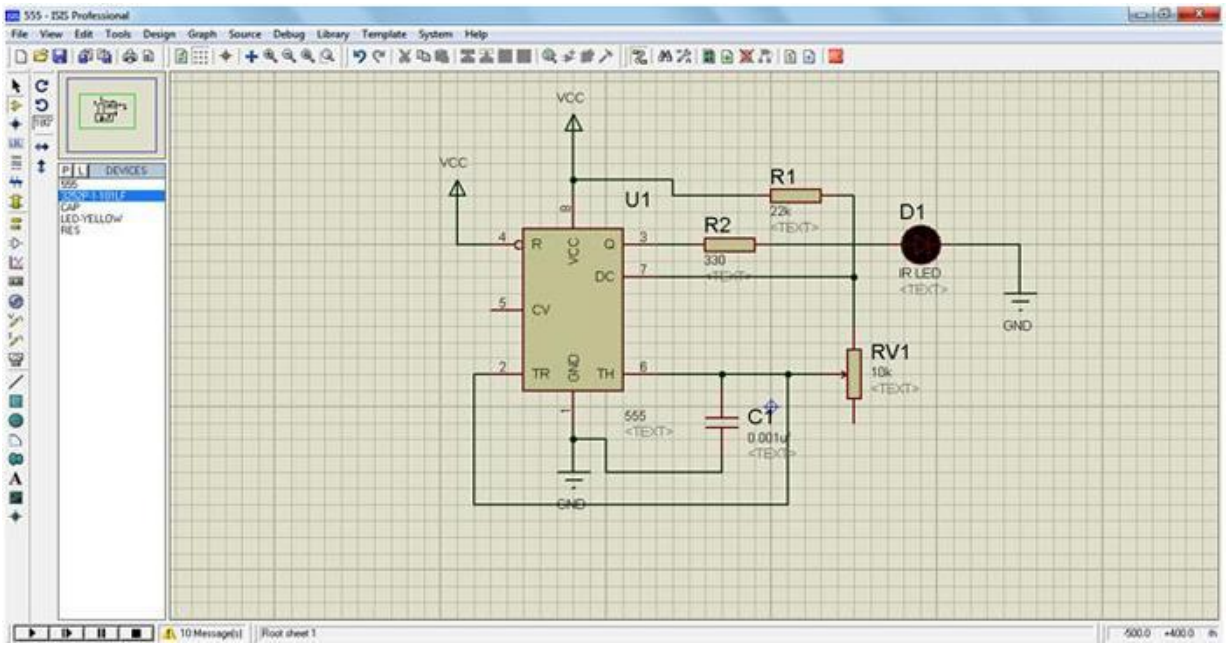


There is another way to select the components. In work space left side there is a tool bar. In that tool bar click the component mode button or pick from library.

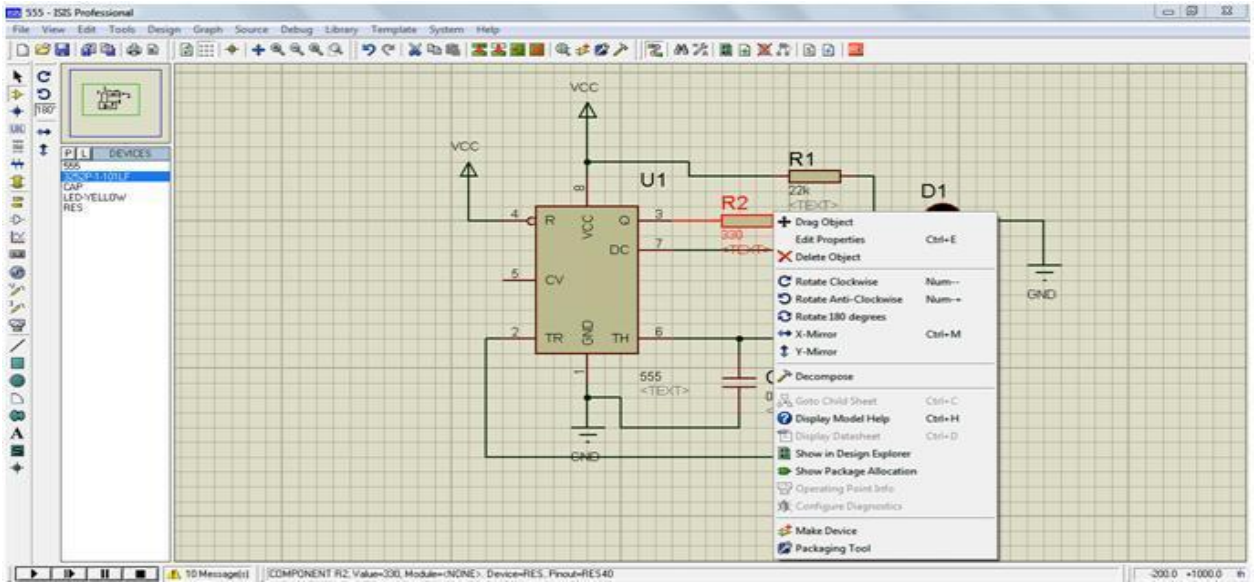


Select the all components from library, that components are added to devices list. Click on the device and change the angle of the device by using rotate buttons. Then click in the work space then the selected component is placed in work space.

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.



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.



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.

YOLO V5 Animal Detection:

YOLOv5 algorithm The characteristic of the YOLO algorithm is that it can maintain a certain accuracy rate at the same time with high speed, and has a relatively low probability of treating the background as an object to be recognized[6]. It can present good classification results for various objects and has strong versatility. The most basic idea used by the YOLO series algorithm is to divide a picture into several grids of the same size [6]. Those grid will be responsible for predicting one target which falls into the grid, and each grid contains several parameters, including the location (x and y) of the object to be detected, the shape (length and width) of the candidate box, the probability level, and the type of object to be detected. Its neural network will output these parameters as the target and extract features around this target. The subsequent versions of YOLO is based on the original YOLO, actively adopting popular and useful new ideas and methods [7, 8], and finally becoming the YOLO The Head part accepts the characteristic diagrams of three different scales. Because the target detection algorithm is generally difficult to detect small targets, the corresponding feature map with larger scale has greater weight in calculation. In YOLOv5 , there are nine candidate boxes, three under each scale, which are used to predict objects of different sizes [3, 9]. The specific size of candidate boxes is provided in yolov5s. yaml. YOLOv5 also allows two grids around the grid where the center of the detected target is located to participate in the prediction process to increase the number of positive samples. At the same time, the predicted values of position is allowed to fluctuate between -0.5-1.5. In YOLOv5 , the matching method of the candidate box has also changed. When the aspect ratio of the bounding box to the ground truth is within 4 times, the bounding box will be selected. In this way, more positive samples can also be obtained for training. So YOLOv5 allows the predicted values of w and h to fluctuate between 0-4v5 model with stable performance.

Model training

In the actual training process, the data set for training needs to be prepared. First, we need to select a portion of the data set as the training set and the remaining portion as the verification set,

then put them into the training folder and verification folder under the images folder respectively. In addition, it is also necessary to give appropriate YOLO format labels to the positive samples for each pictures. Each label has five parameters of type, x, y, w, and h relative to the size of the picture. It is saved as a txt file with the same name as the picture, and also placed in the train folder and verification folder under the labels folder. This time, the data set is taken from African Wildlife in Kaggle, which includes four animals: buffalo, elephant, zebra and rhino. Each animal has nearly 400 pictures. Here, We selected 300 pictures from each animal in data set for training, and other pictures will be used for verification. In addition, three documents need to be prepared in advance. The first is the weight file of YOLOv5 . Here, yolov5s.pt is used. It has a relatively small size, but it can ensure good accuracy, which is very suitable for this case. Next, YOLOv5 needs a yaml file for the samples. The file contents include the path of the training and verification data set, the total number and names of possible classes. Furthermore, the parameter in yolov5s.yaml also needs to be changed. The parameter “nc” should be set as the total number of possible classes. In the train.py file used for training, several parameters need to be changed: set “weights ” as the path of weight file , set “cfg ” as the path of yolov5s.yaml file, set “data ” as the path of the data set yaml file, set “epochs” to 20, set “batch-size” to 4, and set “workers ” to 4. The “batch-size” and “workers ” are set to 4 because of the limitation of the equipment used during training.

CHAPTER-5

CODING

5.1.HARDWARE CODING:

```
#include<Servo.h>

#include "DFRobotDFPlayerMini.h"

#include<SoftwareSerial.h>

#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 10, 11, 12, 13);

#include "DHT.h"

#define DHTPIN A0

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

#ifdef __AVR__

 #include <avr/power.h>

#endif

int cnt=0;

DFRobotDFPlayerMini player;

SoftwareSerial mySerial(2,3);

int led1=4;

int led2=5;
```

```
int led3=6;
int ms=A2;
int pmp1=A4;
int pmp2=A5;
Servo motor;
int buz=A1;
void setup()
{
  Serial.begin(9600);   // Initiate a serial communication
 mySerial.begin(9600);
  lcd.begin(16,2);
  lcd.clear();
  lcd.print("WELCOME");
dht.begin();
delay(1000);
player.begin(mySerial);
    delay(200);
    player.volume(30);
    player.playMp3Folder(1);
  motor.attach(7);
  motor.write(90);
  pinMode(buz,OUTPUT);
  pinMode(led1,OUTPUT);
```

```
 pinMode(led2,OUTPUT);
 pinMode(led3,OUTPUT);
pinMode(pmp1,OUTPUT);
 pinMode(pmp2,OUTPUT);
 digitalWrite(led1,0);
 digitalWrite(led2,0);
 digitalWrite(led3,0);
}
void loop()
{

 int tval=dht.readTemperature();
 int hval=dht.readHumidity();
 int mval=100-analogRead(ms)/10.23;
 lcd.clear();
 lcd.print("M:"+String(mval)+ " T:"+String(tval));
 lcd.setCursor(0,1);
 lcd.print("H:"+String(hval) + " PS:" + String(digitalRead(pmp1)));
 delay(1000);
  if(mval<30)
 {
  digitalWrite(pmp1,1);
  digitalWrite(pmp2,0);
```

```
}
else
{
   digitalWrite(pmp1,0);
  digitalWrite(pmp2,0);
}
cnt=cnt+1;
if(cnt>15)
{
Serial.println("M:"+String(mval)+ "T:"+String(tval)+"H:"+String(hval));
cnt=0;
}
if(Serial.available())
{
  int x=Serial.read();
if(x=='1')
{
 digitalWrite(buz,1);
 player.playMp3Folder(2);
  for(int i=5;i>=0;i--)
  {
  digitalWrite(led1,1);
digitalWrite(led2,1);
```

```
digitalWrite(led3,1);
  for(int ii=0;ii<=180;ii++)
  {
    motor.write(ii);
    delay(15);
  }
     digitalWrite(led1,0);
digitalWrite(led2,0);
digitalWrite(led3,0);
  for(int ii=180;ii>=0;ii--)
  {
    motor.write(ii);
    delay(15);
  }
  digitalWrite(led1,1);
digitalWrite(led2,1);
digitalWrite(led3,1);
delay(200);
digitalWrite(led1,0);
digitalWrite(led2,0);
digitalWrite(led3,0);
  }
  digitalWrite(buz,0);
```

```
  delay(500);
}
if(x=='2')
 {
  digitalWrite(buz,1);
  player.playMp3Folder(3);
   for(int i=5;i>=0;i--)
   {
      digitalWrite(led1,1);
 digitalWrite(led2,1);
 digitalWrite(led3,1);
   for(int ii=0;ii<=180;ii++)
   {
    motor.write(ii);
    delay(15);
   }
     digitalWrite(led1,0);
 digitalWrite(led2,0);
 digitalWrite(led3,0);
   for(int ii=180;ii>=0;ii--)
   {
    motor.write(ii);
    delay(15);
```

```
    }
     digitalWrite(led1,1);
   digitalWrite(led2,1);
   digitalWrite(led3,1);
   delay(200);
   digitalWrite(led1,0);
   digitalWrite(led2,0);
   digitalWrite(led3,0);
     }
     digitalWrite(buz,0);
     delay(500);
   }
}
}
```

5.2 SOFTWARE CODING:

```
import argparse

import os

import sys

from pathlib import Path

import serial

import torch

import torch.backends.cudnn as cudnn

import telepot

FILE = Path(_file_).resolve()

ROOT = FILE.parents[0]  # YOLOv5 root directory

if str(ROOT) not in sys.path:

    sys.path.append(str(ROOT))  # add ROOT to PATH

ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from models.common import DetectMultiBackend

from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadStreams

from utils.general import (LOGGER, check_file, check_img_size, check_imshow,
check_requirements, colorstr, cv2,

 increment_path, non_max_suppression, print_args, scale_coords, strip_optimizer, xyxy2xywh)

from utils.plots import Annotator, colors, save_one_box

from utils.torch_utils import select_device, time_sync
```

```
ser=serial.Serial('com6',9600)
import time
def handle(msg):
  global telegramText
  global chat_id
  global receiveTelegramMessage
    chat_id = msg['chat']['id']
  telegramText = msg['text']
  print("Message received from " + str(chat_id))
  def capture():
   print("Sending photo to " + str(chat_id))
    bot.sendPhoto(chat_id, photo = open('./image.jpg', 'rb'))
bot = telepot.Bot('7143572692:AAHs4Wym9TQ2i4h25Edi63m5dALMAy8V6Nk')
chat_id='5180186601'
bot.message_loop(handle)
print("Telegram bot is ready")
bot.sendMessage(chat_id, 'BOT STARTED')
time.sleep(2)
@torch.no_grad()
def run(
        weights=ROOT / 'best.pt',  # model.pt path(s)
        source=ROOT / 'data/images',  # file/dir/URL/glob, 0 for webcam
        data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
```

```
imgsz=(640, 640),  # inference size (height, width)
conf_thres=0.25,  # confidence threshold
iou_thres=0.45,  # NMS IOU threshold
max_det=1000,  # maximum detections per image
device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
view_img=False,  # show results
save_txt=False,  # save results to *.txt
save_conf=False,  # save confidences in --save-txt labels
save_crop=False,  # save cropped prediction boxes
nosave=False,  # do not save images/videos
classes=15,  # filter by class: --class 0, or --class 0 2 3
agnostic_nms=False,  # class-agnostic NMS
augment=False,  # augmented inference
visualize=False,  # visualize features
update=False,  # update all models
project=ROOT / 'runs/detect',  # save results to project/name
name='exp',  # save results to project/name
exist_ok=False,  # existing project/name ok, do not increment
line_thickness=3,  # bounding box thickness (pixels)
hide_labels=False,  # hide labels
hide_conf=False,  # hide confidences
half=False,  # use FP16 half-precision inference
dnn=False,  # use OpenCV DNN for ONNX inference
```

```
):
    source = str(source)
    save_img = not nosave and not source.endswith('.txt')  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
    webcam = source.isnumeric() or source.endswith('.txt') or (is_url and not is_file)
    if is_url and is_file:
        source = check_file(source)  # download
    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir
    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size
    # Dataloader
    if webcam:
        view_img = check_imshow()
        cudnn.benchmark = True  # set True to speed up constant image size inference
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt)
        bs = len(dataset)  # batch_size
    else:
```

```
    dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt)
    bs = 1  # batch_size
vid_path, vid_writer = [None] * bs, [None] * bs
# Run inference
model.warmup(imgsz=(1 if pt else bs, 3, *imgsz))  # warmup
dt, seen = [0.0, 0.0, 0.0], 0
for path, im, im0s, vid_cap, s in dataset:
    t1 = time_sync()
    im = torch.from_numpy(im).to(device)
    im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
    im /= 255  # 0 - 255 to 0.0 - 1.0
    if len(im.shape) == 3:
        im = im[None]  # expand for batch dim
    t2 = time_sync()
    dt[0] += t2 - t1
    # Inference
    visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
    pred = model(im, augment=augment, visualize=visualize)
    t3 = time_sync()
    dt[1] += t3 - t2
    # NMS
    pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)
```

```
        dt[2] += time_sync() - t3


        # Second-stage classifier (optional)
        # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)
        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f'{i}: '
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)
            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'_{frame}')
# im.txt
            s += '%gx%g ' % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness, example=str(names))
            sts=0
            if len(det):
                sts=1
```

```
            # Rescale boxes from img_size to im0 size
            det[:, :4] = scale_coords(im.shape[2:], det[:, :4], im0.shape).round()
            # Print results
            for c in det[:, -1].unique():
                n = (det[:, -1] == c).sum()  # detections per class
                s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string
            # Write results
            for *xyxy, conf, cls in reversed(det):
                if save_txt:  # Write to file
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  # normalized xywh
                    line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label format
                    with open(f'{txt_path}.txt', 'a') as f:
                        f.write(('%g ' * len(line)).rstrip() % line + '\n')

                if save_img or save_crop or view_img:  # Add bbox to image
                    c = int(cls)  # integer class
                    label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
                    annotator.box_label(xyxy, label, color=colors(c, True))

                if save_crop:
```

```
                save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg', BGR=True)


        # Stream results
        im0 = annotator.result()
        if view_img:
            cv2.imshow(str(p), im0)
            cv2.waitKey(1)  # 1 millisecond
            if(sts==1):
                ser.write('1'.encode())
                bot.sendMessage(chat_id, 'BOT STARTED')
                cv2.imwrite('image.jpg',im0)
                cv2.waitKey(1)
                capture()
                time.sleep(5)


        # Save results (image with detections)
        if save_img:
            if dataset.mode == 'image':
                cv2.imwrite(save_path, im0)
            else:  # 'video' or 'stream'
                if vid_path[i] != save_path:  # new video
                    vid_path[i] = save_path
```

```
                if isinstance(vid_writer[i], cv2.VideoWriter):
                    vid_writer[i].release()  # release previous video writer
                if vid_cap:  # video
                    fps = vid_cap.get(cv2.CAP_PROP_FPS)
                    w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                    h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                else:  # stream
                    fps, w, h = 30, im0.shape[1], im0.shape[0]
                save_path = str(Path(save_path).with_suffix('.mp4'))  # force *.mp4 suffix on results videos
                vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
            vid_writer[i].write(im0)
        # Print time (inference-only)
        LOGGER.info(f'{s}Done. ({t3 - t2:.3f}s)')
    # Print results
    t = tuple(x / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at shape {(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt else ''
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
```

```
    strip_optimizer(weights)  # update model (to fix SourceChangeWarning)

def parse_opt():

    parser = argparse.ArgumentParser()

    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'best.pt', help='model path(s)')

    parser.add_argument('--source', type=str, default=ROOT / '0', help='file/dir/URL/glob, 0 for webcam')

    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='(optional) dataset.yaml path')

    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640], help='inference size h,w')

    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence threshold')

    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')

    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections per image')

    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')

    parser.add_argument('--view-img', action='store_true', help='show results')

    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')

    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt labels')

    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction boxes')

    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')

    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0, or --classes 0 2 3')

    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
```

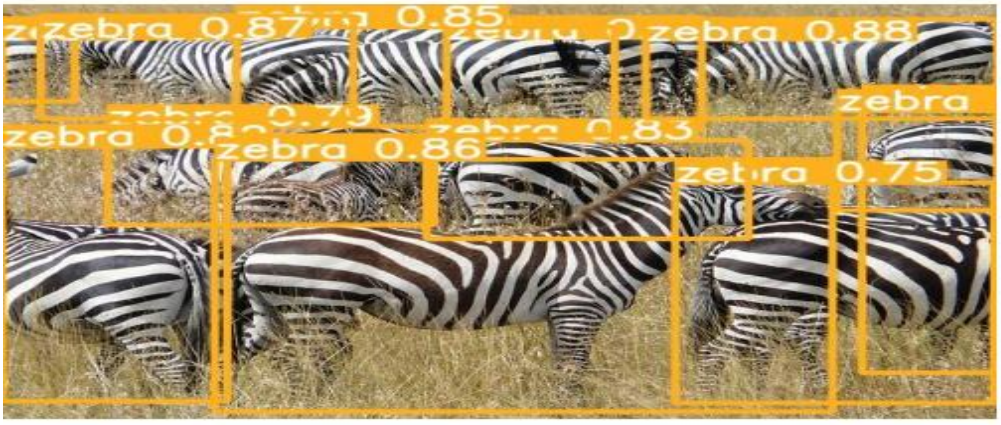
```
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save results to project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box thickness (pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for ONNX inference')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt
def main(opt):
    check_requirements(exclude=('tensorboard', 'thop'))
    run(**vars(opt))
if _name_ == "_main_":
    opt = parse_opt()
```

main(opt)

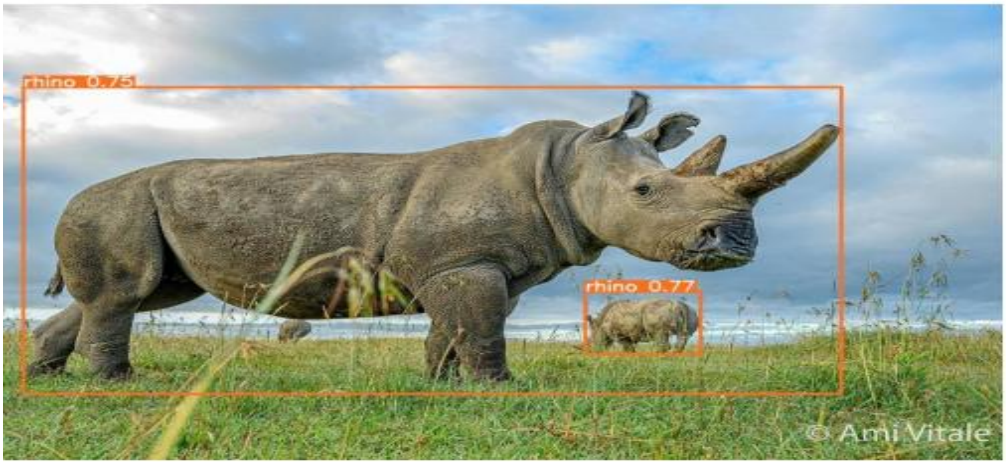
CHAPTER-6

RESULT

(a)



(b)



CHAPTER-7

CONCLUSION:

In conclusion, the proposed system presents a sophisticated and effective solution to the persistent challenge of animal intrusion in agricultural settings. By integrating advanced technology, including the YOLOv5 algorithm for real-time animal detection, with non-lethal deterrent measures, such as Neopixel lights, scaring sounds, and servo motor movements, the system offers a humane, efficient, and environmentally sustainable approach to crop protection.

The system's ability to detect intruding animals in real-time and send timely alerts to farmers via Telegram ensures swift response and intervention, minimizing crop damage and financial losses. Furthermore, the use of non-lethal deterrents prioritizes the safety and well-being of both farmers and wildlife, fostering coexistence and harmony between humans and the natural environment.

With its cost-effectiveness, scalability, and adaptability to various agricultural and environmental settings, the proposed system has wide-ranging applications beyond crop protection, including wildlife conservation, urban gardening, forestry management, and research.

REFFERENCES: