KKR & KSR INSTITUTE OF TECHNOLOGY AND SCIENCES

(Autonomous)

Vinjanampadu (V), Vatticherukuru (M) Guntur – 522017, Andhra Pradesh

REPORT ON
SKILL ORIENTED COURSE

Name :

Roll. No :

Course Name :

Year : **Semester :**

DEPARTMENT
OF
ELECTRONICS AND COMMUNICATION ENGINEERING

KKR & KSR INSTITUTE OF TECHNOLOGY & SCIENCES
AUTONOMOUS

CERTIFICATE

Cerified that this is the bonafide record of practical work done by

Mr/Ms…… a student of **Electronics and Communication Engineering Department** with Reg.No..... in the Skill oriented course on **EMBEDDED C USING IOT** during the year 2023.

Signature
Faculty In charge

Signature
Head of the Department

Submitted for the practical examination held on…….

Examiner

TABLE OF CONTENTS

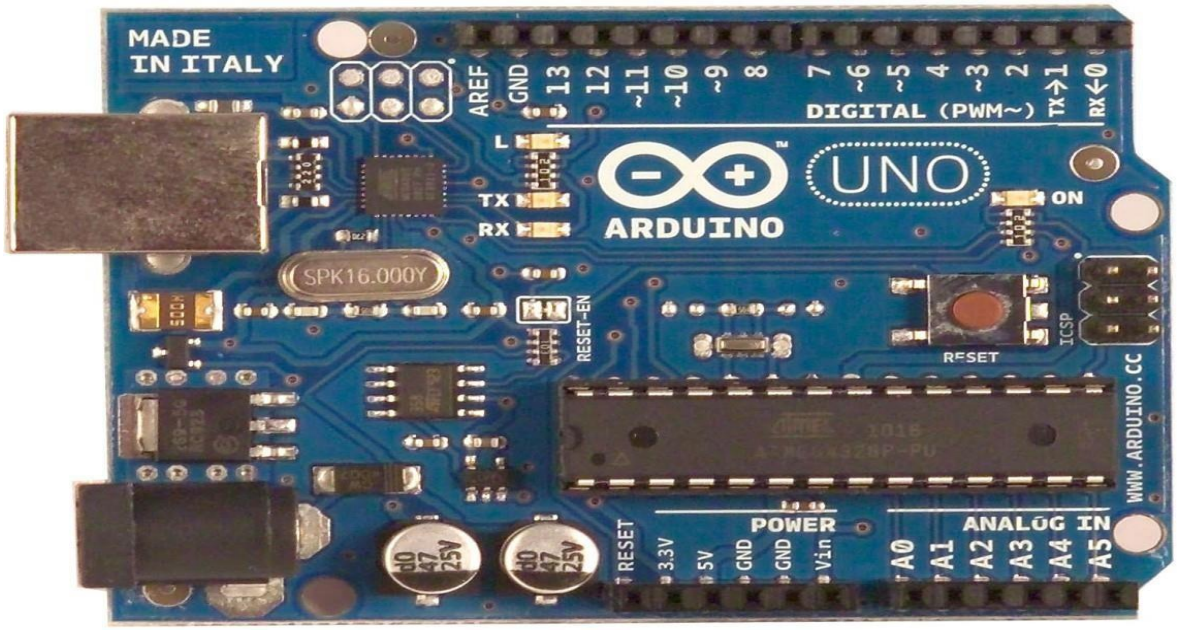
S. No	Date	Activity Name	Page No.	Marks	Signature
1		INTRODUCTION TO ARDUNIO AND ESP-32			
2		INTRODUCTION TO ARDUINO UNO PROGRAMMING			
3		DESIGN THE LED BLINKING USING ARDUINO UNO			
4		DESIGN THE CONTACTLESS DOOR BELL USING ARDUINO UNO			
5		DESIGN THE OBJECT THEFT DETECTOR USING ARDUINO UNO			
6		DESIGN THE ESTIMATING INDOOR POPULATION DENSITY USING ARDUINO UNO			
7		DESIGN THE AUTOMATIC STREET LIGHT SYSTEM USING ARDUINO UNO			
8		DESIGN THE FIRE DETECTION AND ALARM SYSTEM USING ARDUINO UNO			
9		DESIGN THE WEATHER MOINTORING SYSTEM USING ESP-32			
10		DESIGN THE AUTOMATIC SAFETY HOMEBELL SYSTEM WITH MESSAGE ENABLED FEAUTERS USING ESP-32			
11		DESIGN THE APPLICATION FOR ALZHEMER'S ON SMART MEDICAL REMAINDER USING ESP-32			
12		DESIGN THE APPLICATION FOR SMART BLIND STICK USING ESP-32			
13		DESIGN THE IOT BASED HOME AUTOMATION USING ESP-32			
14		DESIGN THE AGRICULTURE BASED WATER PUMP CONTROLLER USING ESP-32			
15		DESIGN THE VOICE CONTROL HOME AUTOMATION USING BLUETOOTH USING ESP-32			
16		DESIGN THE VOICE TO TEXT CONVERSION AND TEXT TO VOICE CONVERSION USING ESP-32			
17		DESIGN THE SMOKE DETECTION USING MQ-2 GAS SENSOR USING ARDUNIO UNO			
18		DESIGN THE INTERFACE ULTRASONIC SENSOR WITH ARDUINO UNO			

Activity:01

Date:

INTRODUCTION TO ARDUINO UNO AND ESP-32

Arduino UNO



Product Overview

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 versiwill 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; for a comparison with previous versions,see the index of Arduino boards.

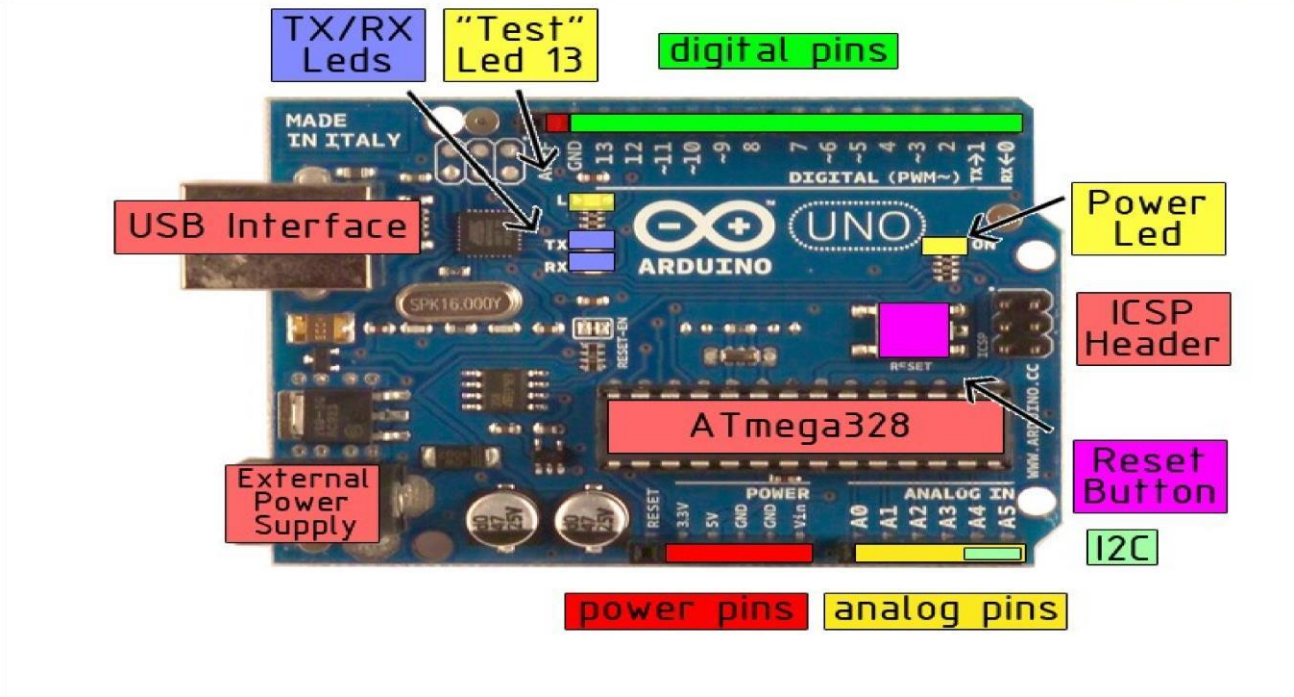
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

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 bybootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

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

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adaptercan 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 5Vpin 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 to5 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); Ithas also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and 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 areconnected 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, arising 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 ison, 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 analogReference() 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 analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button toshields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

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 USBand appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USBCOM 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-to-serial 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 (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new codeto it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit SerialProgramming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU

bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. 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).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in away that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 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 orso, 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.

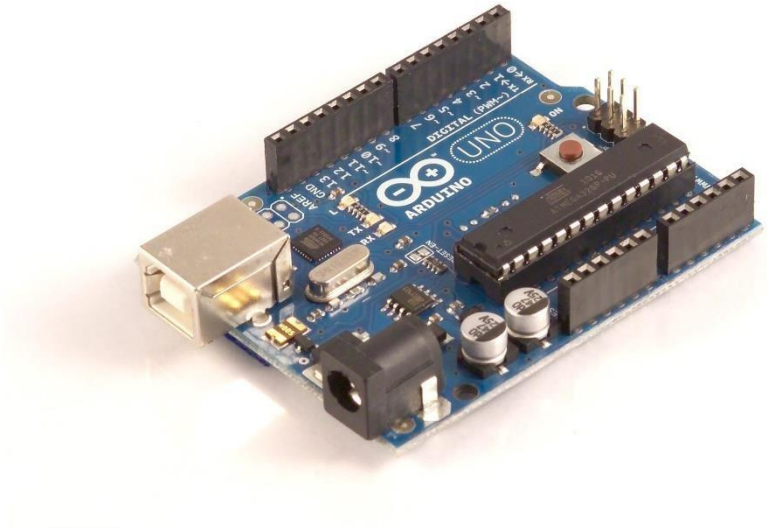
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-resetby connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

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 connectionuntil 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 connectorand power jack extending beyond the former dimension. Three screw holes allow the board to be attached toa surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multipleof the 100 mil spacing of the other pins.



How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

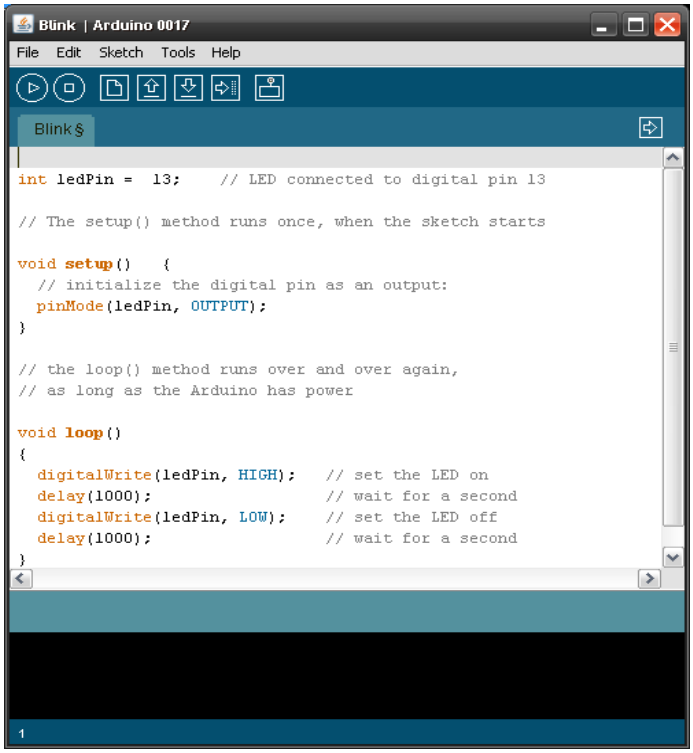
Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install Windows Install Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select



File>Sketchbook> Arduino-0017>Examples> Digital>Blink

Once you have your skecth you'llsee something very close to the screenshot on the right.

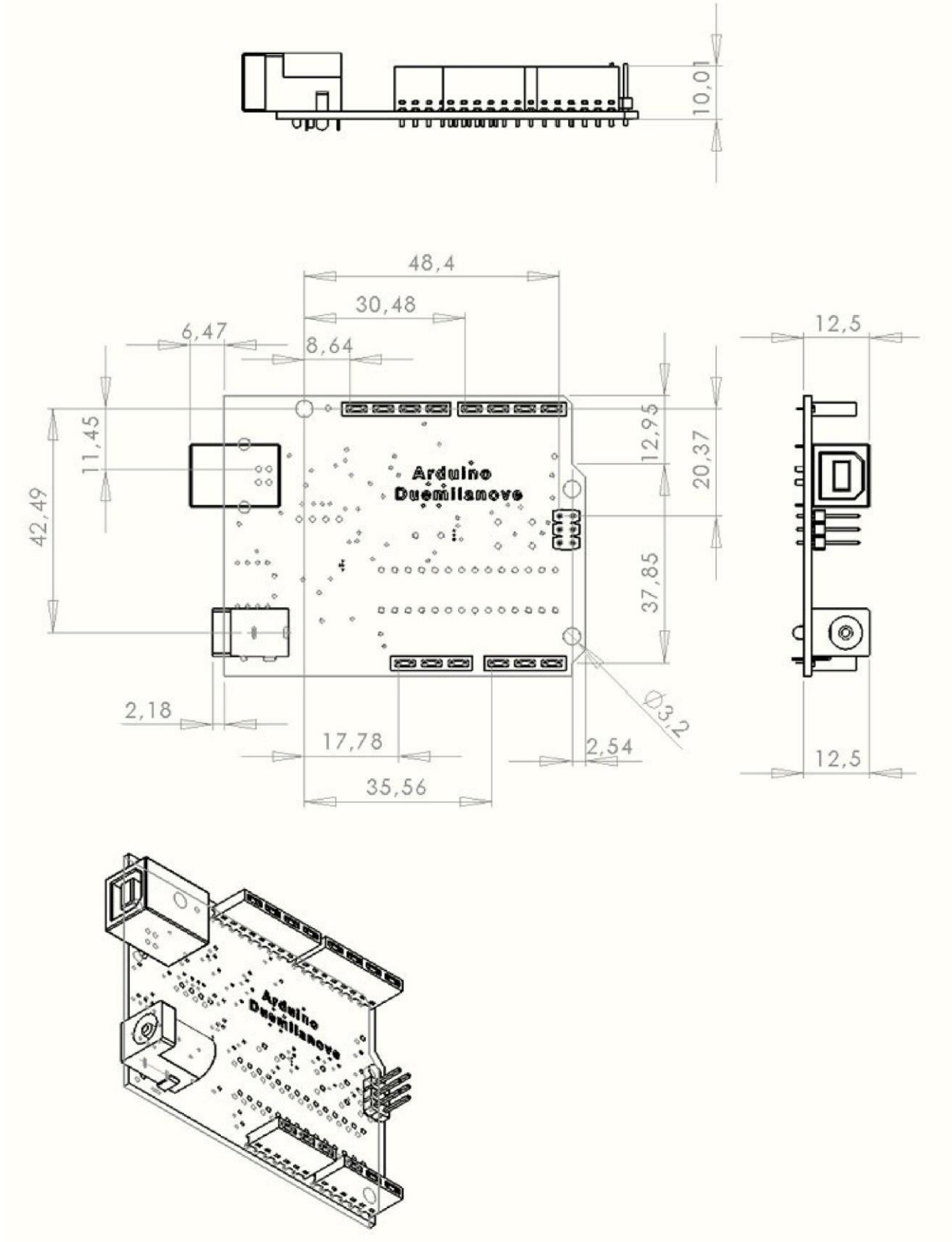
In **Tools>Board** select

Now you have to go to

Tools>SerialPort

and select the right serial port, theone arduino is attached to.

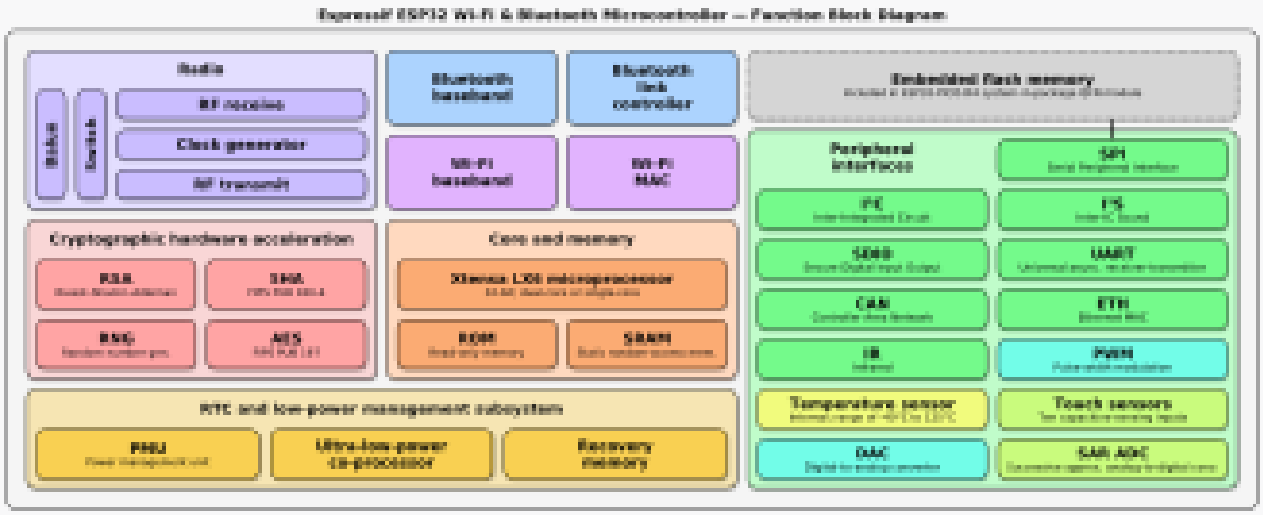
Dimensioned Drawing



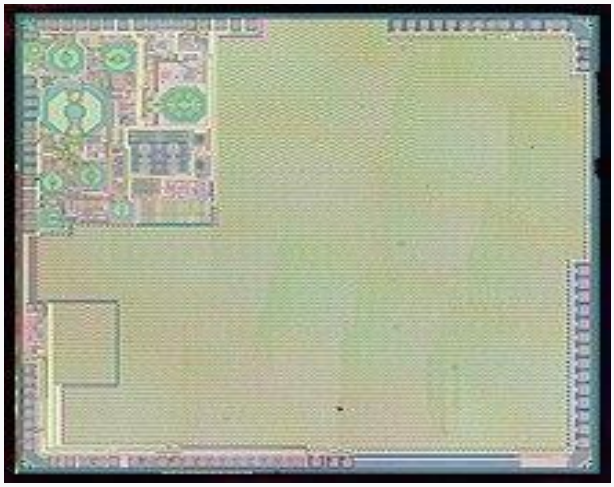
Introduction To ESP-32

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process.[2] It is a successor to the ESP8266 microcontroller.

Features:



ESP32 function block diagram.

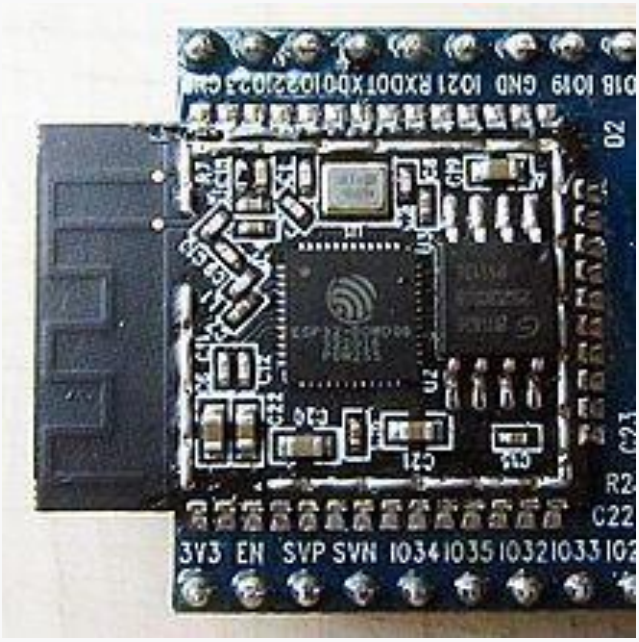


ESP32 Die shot

Features of the ESP32 include the following:[1]

- Processors:
  - CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
  - Ultra low power (ULP) co-processor
- Memory: 320 KiB RAM, 448 KiB ROM
- Wireless connectivity:
  - Wi-Fi: 802.11 b/g/n
  - Bluetooth: v4.2 BR/EDR and BLE (shares the radio with Wi-Fi)
- Peripheral interfaces:
  - 34 × programmable GPIOs
  - 12-bit SAR ADC up to 18 channels
  - 2 × 8-bit DACs
  - 10 × touch sensors (capacitive sensing GPIOs)
  - 4 × SPI
  - 2 × I²S interfaces
  - 2 × I²C interfaces
  - 3 × UART
  - SD/SDIO/CE-ATA/MMC/eMMC host controller
  - SDIO/SPI slave controller
  - Ethernet MAC interface with dedicated DMA and planned IEEE 1588 Precision Time Protocol support[4]
  - CAN bus 2.0
  - Infrared remote controller (TX/RX, up to 8 channels)
  - Pulse counter (capable of full quadrature decoding)
  - Motor PWM
  - LED PWM (up to 16 channels)
  - Ultra low power analog pre-amplifier
- Security:
  - IEEE 802.11 standard security features all supported, including WPA, WPA2, WPA3 (depending on version)[5] and WLAN Authentication and Privacy Infrastructure (WAPI)
  - Secure boot
  - Flash encryption
  - 1024-bit OTP, up to 768-bit for customers
  - Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)
- Power management:
  - Internal low-dropout regulator
  - Individual power domain for RTC
  - 5 µA deep sleep current
  - Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt

1. Processor: The ESP32 is based on a dual-core 32-bit Tensilica Xtensa LX6 microprocessor, which operates at up to 240 MHz. The microprocessor features a floating-point unit (FPU) for high-precision computing and a DMA controller for efficient data transfers.
2. Wireless connectivity: The ESP32 features built-in Wi-Fi and Bluetooth connectivity, which makes it ideal for IoT applications. It supports both 2.4 GHz Wi-Fi and Bluetooth Low Energy (BLE) 4.2, withsupport for BLE 5.0 available via software update.
3. Memory: The ESP32 has 520 KB of SRAM and 448 KB of ROM, as well as 4 MB of flash memory for program and data storage.
4. Peripherals: The ESP32 features a wide range of peripheral interfaces, including SPI, I2C, UART, PWM, ADC, DAC, and more. It also has a built-in hall sensor and temperature sensor, making it ideal for sensing applications.
5. Power consumption: The ESP32 has a range of low-power modes, which can significantly reduce power consumption in battery-powered applications. It also supports power management features such as dynamic voltage scaling and deep sleep mode.
6. Programming: The ESP32 can be programmed using a variety of programming languages, including C++, Arduino, MicroPython, and Lua. It also has a built-in bootloader for easy firmware updates.



ESP-WROOM-32 module with ESP32-D0WDQ6 chip

Manufacturer	Espressif Systems
Type	Microcontroller
Release date	September 6, 2016[1]

CPU	Tensilica Xtensa LX6 microprocessor @ 160 or 240 MHz
Memory	320 KiB SRAM
Power	3.3 V DC
Predecessor	ESP8266

Overall, the ESP32 is a powerful and versatile microcontroller that offers a wide range of features and capabilities for IoT and embedded applications. Its low cost and ease of use have made it a popular choice among developers and makers alike.

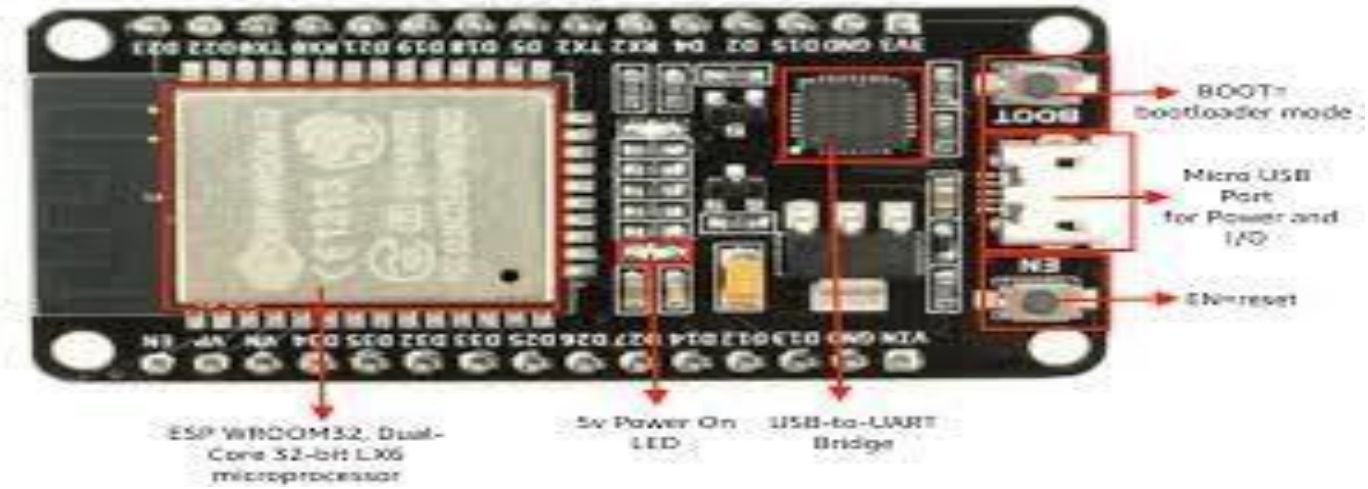
The ESP32 is a powerful microcontroller developed by Espressif Systems. It is a successor to the popular ESP8266 microcontroller and features a dual-core processor, wireless connectivity options, and a wide range of peripheral interfaces. The ESP32 has become popular among developers and makers due to its low cost, high performance, and ease of use.

Some of the key features of the ESP32 include:

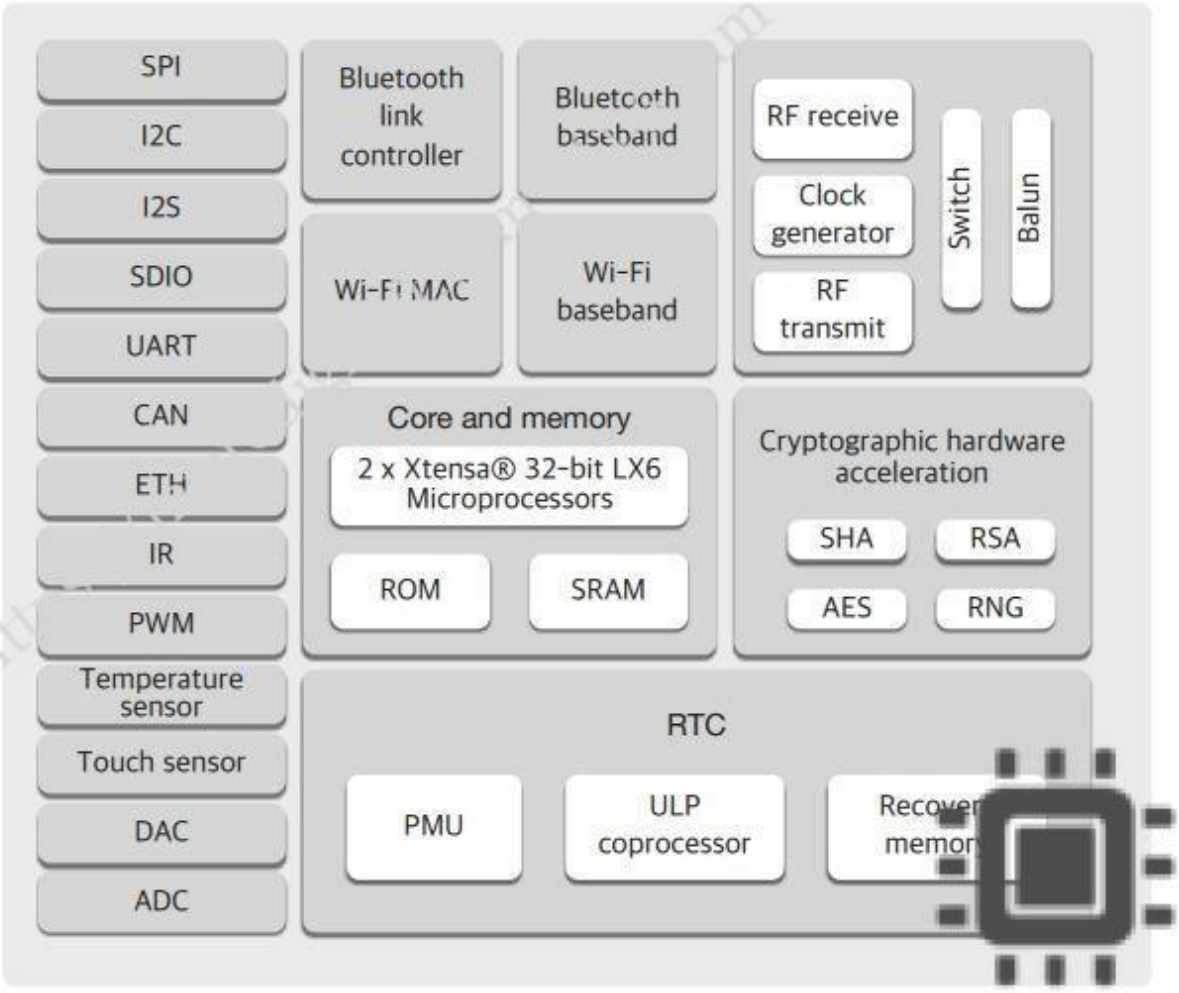
- Dual-core 32-bit processors: The ESP32 features two Tensilica Xtensa LX6 microprocessors running at up to 240 MHz.
- Wireless connectivity: The ESP32 supports both Wi-Fi and Bluetooth, making it ideal for Internet of Things (IoT) applications.
- Peripheral interfaces: The ESP32 has a wide range of peripheral interfaces, including I2C, SPI, UART, and more.
- Low power consumption: The ESP32 has a range of low power modes, making it ideal for battery-powered applications.

The ESP32 is compatible with the Arduino IDE and can be programmed using the Arduino language. It also supports other programming languages such as MicroPython and Lua.

Overall, the ESP32 is a versatile and powerful microcontroller that has become a popular choice for a wide range of applications, including home automation, robotics, and industrial automation.



BLOCK DIAGRAM OF ESP-32:



Activity:02

Date:

INTRODUCTION TO ARDUINO UNO PROGRAMMING

Functions

A function is a block of code that has a name and a block of statements that are executed when the function is called. The functions void setup() and void loop() havealready been discussed and other built-in functions will be discussed later.

Custom functions can be written to perform repetitive tasks and reduce clutter in a program. Functions are declared by first declaring the function type. This is the type of value to be returned by the function such as 'int' for an integer type function. If no value is to be returned the function type would be void. After type, declare the name given to the function and in parenthesis any parameters being passed to the function.

```
type functionName(parameters)
{
  statements;
}
```

The following integer type function delayVal() is used to set a delay value in a program by reading the value of a potentiometer. It first declares a local variable v, sets v to the value of the potentiometer which gives a number between 0-1023, thendivides that value by 4 for a final value between 0-255, and finally returns that valueback to the main program.

```
int delayVal()
{
  int v;                  // create temporary variable 'v'v  = analogRead(pot);   //   read
  potentiometer value
  v /= 4;                 // converts 0-1023 to 0-255
  return v;               // return final value
}
```

{} curly braces

Curly braces (also referred to as just "braces" or "curly brackets") define the beginning and end of function blocks and statement blocks such as the void loop()function and the for and if statements.

```
type function()
{
  statements;
}
```

An opening curly brace { must always be followed by a closing curly brace }. This is often referred to as the braces being balanced. Unbalanced braces can often lead tocryptic, impenetrable compiler errors that can sometimes be hard to track down in a large program.

The Arduino environment includes a convenient feature to check the balance of curlybraces. Just select a brace, or even click the insertion point immediately following a brace, and its logical companion will be highlighted.

; semicolon

A semicolon must be used to end a statement and separate elements of the program.A semicolon is also used to separate elements in a for loop.

```
int x = 13;          // declares variable 'x' as the integer 13
```

/*… */ block comments

Block comments, or multi-line comments, are areas of text ignored by the programand are used for large text descriptions of code or comments that help others understand parts of the program. They begin with /* and end with */ and can span multiple lines.

```
/* this is an enclosed block comment don't forget the closing comment -they
have to be balanced!
*/
```

Because comments are ignored by the program and take no memory space they should be used generously and can also be used to "comment out" blocks of code fordebugging purposes.

// line comments

Single line comments begin with // and end with the next line of code. Like blockcomments, they are ignored by the program and take no memory space.

```
// this is a single line comment
```

Single line comments are often used after a valid statement to provide more information about what the statement accomplishes or to provide a future reminder.

Variables

A variable is a way of naming and storing a numerical value for later use by the program. As their namesake suggests, variables are numbers that can be continuallychanged as opposed to constants whose value never changes. A variable needs to be declared and optionally assigned to the value needing to be stored. The followingcode declares a variable called inputVariable and then assigns it the value obtained on analog input pin 2:

```
int inputVariable = 0;          // declares a variable and
                                // assigns value of 0 inputVariable analogRead(2);          //
set variable to value of
                               // analog pin 2
```

'inputVariable' is the variable itself. The first line declares that it will contain an int, short for integer. The second line sets the variable to the value at analog pin 2. Thismakes the value of pin 2 accessible elsewhere in the code.

Once a variable has been assigned, or re-assigned, you can test its value to see if it meets certain

conditions, or you can use its value directly. As an example to illustratethree useful operations with variables, the following code tests whether the inputVariable is less than 100, if true it assigns the value 100 to inputVariable, and then sets a delay based on inputVariable which is now a minimum of 100:

```
if (inputVariable < 100) // tests variable if less than 100
{
  inputVariable = 100;     // if true assigns value of 100
}
delay(inputVariable);      // uses variable as delay
```

Variable declaration

All variables have to be declared before they can be used. Declaring a variable means defining its value type, as in int, long, float, etc., setting a specified name, and optionally assigning an initial value. This only needs to be done once in a program butthe value can be changed at any time using arithmetic and various assignments.

The following example declares that inputVariable is an int, or integer type, and that its initial value equals zero. This is called a simple assignment.

```
int inputVariable = 0;
```

A variable can be declared in a number of locations throughout the program and where this definition takes place determines what parts of the program can use thevariable.

Variable scope

A variable can be declared at the beginning of the program before void setup(), locally inside of functions, and sometimes within a statement block such as for loops. Where the variable is declared determines the variable scope, or the ability of certainparts of a program to make use of the variable.

A global variable is one that can be seen and used by every function and statement ina program. This variable is declared at the beginning of the program, before the setup() function.

A local variable is one that is defined inside a function or as part of a for loop. It is only visible and can only be used inside the function in which it was declared. It is therefore possible to have two or more variables of the same name in different parts of the same program that contain different values. Ensuring that only one function hasaccess to its variables simplifies the program and reduces the potential for programming errors.

The following example shows how to declare a few different types of variables and demonstrates each variable's visibility:

```
int value;                  // 'value' is visible
                            // to any function
void setup()
{
```

```
// no setup needed
}

void loop()
{
for (int i=0; i<20;)  // 'i' is only visible
{                                   // inside the for-loopi++;
}
float f;                    // 'f' is only visible
}                           // inside loop
```

Arrays

An array is a collection of values that are accessed with an index number. Any value in the array may be called upon by calling the name of the array and the index number of the value. Arrays are zero indexed, with the first value in the array beginning at index number 0. An array needs to be declared and optionally assignedvalues before they can be used.

int myArray[] = {value0, value1, value2...}

Likewise it is possible to declare an array by declaring the array type and size andlater assign values to an index position:

int myArray[5]; // declares integer array w/ 6 positionsmyArray[3] = 10; // assigns the 4th index the value 10

To retrieve a value from an array, assign a variable to the array and index position:

x = myArray[3]; // x now equals 10

Arrays are often used in for loops, where the increment counter is also used as the index position for each array value. The following example uses an array to flicker anLED. Using a for loop, the counter begins at 0, writes the value contained at index position 0 in the array flicker[], in this case 180, to the PWM pin 10, pauses for 200ms, then moves to the next index position.

```
int ledPin = 10;                    // LED on pin 10 byte flicker[] = {180, 30, 255, 200,
10, 90, 150, 60};
                                    // above array of 8
void setup()                        // different values
{
pinMode(ledPin, OUTPUT);            // sets OUTPUT pin
}

void loop()
{
for(int i=0; i<7; i++)              // loop equals number
{                                   // of values in array
analogWrite(ledPin, flicker[i]); // write index valuedelay(200);  // pause 200ms
}
}
```

True/false

These are Boolean constants that define logic levels. FALSE is easily defined as 0 (zero) while TRUE is often defined as 1, but can also be anything else except zero.So in a Boolean sense, -1, 2, and -200 are all also defined as TRUE.

```
if (b == TRUE);
{
 doSomething;
}
```

High/low

These constants define pin levels as HIGH or LOW and are used when reading orwriting to digital pins. HIGH is defined as logic level 1, ON, or 5 volts while LOW islogic level 0, OFF, or 0 volts.

digitalWrite(13, HIGH);

Input/output

Constants used with the pinMode() function to define the mode of a digital pin aseither INPUT or OUTPUT.

pinMode(13, OUTPUT);

If

if statements test whether a certain condition has been reached, such as an analogvalue being above a certain number, and executes any statements inside the brackets if the statement is true. If false the program skips over the statement. Theformat for an if test is:

```
if (someVariable ?? value)
{
   doSomething;
}
```

The above example compares someVariable to another value, which can be either avariable or constant. If the comparison, or condition in parentheses is true, the statements inside the brackets are run. If not, the program skips over them and continues on after the brackets.

if… else

if… else allows for 'either-or' decisions to be made. For example, if you wanted to testa digital input, and do one thing if the input went HIGH or instead do another thing if the input was LOW, you would write that this way:

```
if (inputPin == HIGH)
{
   doThingA;
}
else
{
   doThingB;
```

```
}
```

else can also precede another if test, so that multiple, mutually exclusive tests can berun at the same time. It is even possible to have an unlimited number of these else branches. Remember though, only one set of statements will be run depending on the condition tests:

```
if (inputPin < 500)
 {
   doThingA;
 }
 else if (inputPin >= 1000)
 {
   doThingB;
 }
 else
 {
   doThingC;
 }
```

For

The for statement is used to repeat a block of statements enclosed in curly braces a specified number of times. An increment counter is often used to increment and terminate the loop. There are three parts, separated by semicolons (;), to the for loopheader:

```
for (initialization; condition; expression)
{
   doSomething;
}
```

The initialization of a local variable, or increment counter, happens first and only once. Each time through the loop, the following condition is tested. If the condition remains true, the following statements and expression are executed and the conditionis tested again. When the condition becomes false, the loop ends.

The following example starts the integer i at 0, tests to see if i is still less than 20 andif true, increments i by 1 and executes the enclosed statements:

```
for (int i=0; i<20; i++)  // declares i, tests if less
{                                    // than 20, increments i by 1digitalWrite(13, HIGH); // turns
pin 13 on delay(250);
                          //      pauses for 1/4 second
 digitalWrite(13, LOW); // turns pin 13 off delay(250);
                           // pauses for 1/4 second
    }
```

While

while loops will loop continuously, and infinitely, until the expression inside the parenthesis becomes false. Something must change the tested variable, or the whileloop will never exit. This could be in your code, such as an incremented variable, or an external condition, such as testing a sensor.

```
while (someVariable ?? value)
{
doSomething;
}
```

The following example tests whether 'someVariable' is less than 200 and if trueexecutes the statements inside the brackets and will continue looping until 'someVariable' is no longer less than 200.

```
while (someVariable < 200) // tests if less than 200
{
doSomething;                // executes enclosed statementssomeVariable++;  //  increments variable by 1
}
```

do… while

The do loop is a bottom driven loop that works in the same manner as the while loop,with the exception that the condition is tested at the end of the loop, so the do loop will always run at least once.

```
do
{
doSomething;
} while (someVariable ?? value);
```

The following example assigns readSensors() to the variable 'x', pauses for 50 milliseconds, then loops indefinitely until 'x' is no longer less than 100:

```
do
{
x = readSensors();          // assigns the value of
                            // readSensors() to x
   delay (50);              // pauses 50 milliseconds
 } while (x < 100);         // loops if x is less than 100
```

pinMode(pin, mode)

Used in void setup()to configure a specified pin to behave either as an INPUT oran OUTPUT.

```
pinMode(pin, OUTPUT);       // sets 'pin' to output
```

Arduino digital pins default to inputs, so they don't need to be explicitly declared as inputs with pinMode(). Pins configured as INPUT are said to be in a high-impedancestate.

There are also convenient 20KΩ pullup resistors built into the Atmega chip that can be accessed from software. These built-in pullup resistors are accessed in the following manner:

```
pinMode(pin, INPUT);            // set 'pin' to input digitalWrite(pin, HIGH);    // turn on pullup resistors
```

Pullup resistors would normally be used for connecting inputs like switches. Notice inthe above example it does not convert pin to an output, it is merely a method for activating the internal pull-ups.

Pins configured as OUTPUT are said to be in a low-impedance state and can provide40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), but not enough current to runmost relays, solenoids, or motors.

Short circuits on Arduino pins and excessive current can damage or destroy the output pin, or damage the entire Atmega chip. It is often a good idea to connect anOUTPUT pin to an external device in series with a 470Ω or 1KΩ resistor.

digitalRead(pin)

Reads the value from a specified digital pin with the result either HIGH or LOW. Thepin can be specified as either a variable or constant (0-13).

```
value = digitalRead(Pin);          // sets 'value' equal to
                                   // the input pin
```

digitalWrite(pin, value)

Outputs either logic level HIGH or LOW at (turns on or off) a specified digital pin. Thepin can be specified as either a variable or constant (0-13).

```
digitalWrite(pin, HIGH);       // sets 'pin' to high
```

The following example reads a pushbutton connected to a digital input and turns onan LED connected to a digital output when the button has been pressed:

```
int led     = 13;    // connect LED to pin 13
int pin     =  7;    // connect pushbutton to pin 7
int value =  0;      // variable to store the read value

void setup()
{
  pinMode(led, OUTPUT);    // sets pin 13 as outputpinMode(pin, INPUT);        //      sets
  pin 7 as input
}

void loop()
{
  value = digitalRead(pin);  // sets 'value' equal to
                                    // the input pin digitalWrite(led, value);  // sets 'led' to the


}
```

analogRead(pin)

Reads the value from a specified analog pin with a 10-bit resolution. This function only works on the analog in pins (0-5). The resulting integer values range from 0 to1023.

value = analogRead(pin); // sets 'value' equal to 'pin

analogWrite(pin, value)

Writes a pseudo-analog value using hardware enabled pulse width modulation (PWM) to an output pin marked PWM. On newer Arduinos with the ATmega168 chip,this function works on pins 3, 5, 6, 9, 10, and 11. Older Arduinos with an ATmega8 only support pins 9, 10, and 11. The value can be specified as a variable or constantwith a value from 0-255.

analogWrite(pin, value); // writes 'value' to analog 'pin'

A value of 0 generates a steady 0 volts output at the specified pin; a value of 255 generates a steady 5 volts output at the specified pin. For values in between 0 and 255, the pin rapidly alternates between 0 and 5 volts - the higher the value, the moreoften the pin is HIGH (5 volts). For example, a value of 64 will be 0 volts three- quarters of the time, and 5 volts one quarter of the time; a value of 128 will be at 0 half the time and 255 half the time; and a value of 192 will be 0 volts one quarter of the time and 5 volts three-quarters of the time.

Because this is a hardware function, the pin will generate a steady wave after a call toanalogWrite in the background until the next call to analogWrite (or a call to digitalRead or digitalWrite on the same pin).

The following example reads an analog value from an analog input pin, converts thevalue by dividing by 4, and outputs a PWM signal on a PWM pin:

```
int led = 10;          // LED with 220 resistor on pin 10int pin =   0;        // potentiometer on
analog pin 0 int value;     // value for reading
void setup(){}       // no setup neededvoid loop()
{
value =   analogRead(pin);   // sets 'value' equal to 'pin'value /= 4;     // converts 0-1023 to 0-255
analogWrite(led, value);                  // outputs PWM signal to led
}
```

Serial.begin(rate)

Opens serial port and sets the baud rate for serial data transmission. The typical baud rate for communicating with the computer is 9600 although other speeds are supported.

```
void setup()
{
Serial.begin(9600); // opens serial port
} // sets data rate to 9600 bps
```

Note: When using serial communication, digital pins 0 (RX) and 1 (TX) cannot be used at the same time.

Serial.println(data)

Prints data to the serial port, followed by an automatic carriage return and line feed. This command takes the same form as Serial.print(), but is easier for reading data on the Serial Monitor. Serial.println(analogValue); // sends the value of // 'analogValue'

Note: For more information on the various permutations of the Serial.println() and Serial.print() functions please refer to the Arduino website.

The following simple example takes a reading from analog pin0 and sends this data to the computer every 1 second.

```
void setup()
{ Serial.begin(9600); // sets serial to 9600bps }
void loop()
{ Serial.println(analogRead(0)); // sends analog value
delay(1000); // pauses for 1 second
}
```

Activity:03

Date:

DESIGN THE LED BLINKING USING ARDUINO UNO

AIM:

To Design the CONTACTLESS DOOR BELL using arduino uno.

APPARATUS:

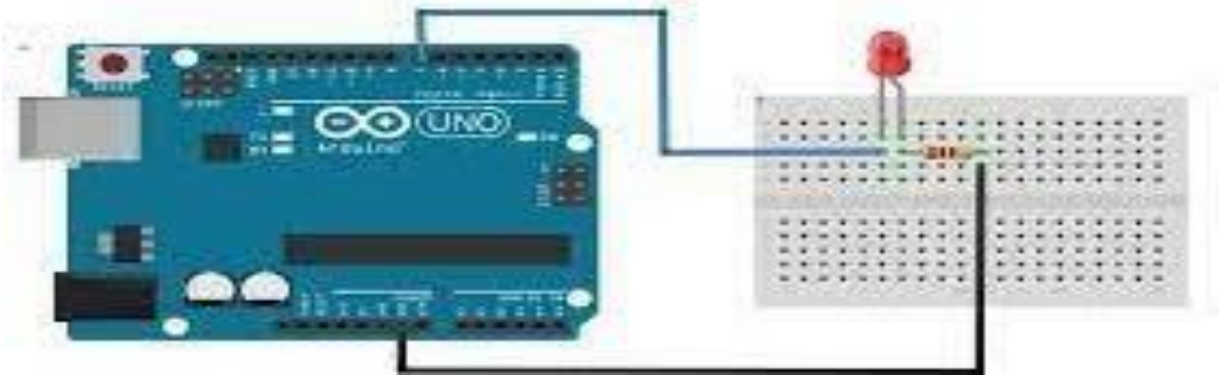
- Arduino uno
- Data cable
- Jumber cables
- Bread board
- Led

DESCRIPTION:

Here are the steps to make an LED blink:

1. Connect the positive leg of the LED to a digital output pin on your microcontroller (or any other digital device that can output voltage), using a jumper wire.
2. Connect the negative leg of the LED to the breadboard.
3. Connect one end of the resistor to the negative leg of the LED.
4. Connect the other end of the resistor to the ground rail of the breadboard.
5. Connect the digital output pin to a microcontroller or any other device capable of outputting voltage.
6. Write a program to toggle the digital output pin on and off at a desired rate (e.g. every half a second or one second).
7. Upload the program to the microcontroller and run it.
8. The LED should now be blinking at the desired rate.
9. Here is an example code in Arduino for making an LED blink:

CIRCUIT DIAGRAM:



CODE:

```
int ledPin = 13; // LED connected to digital pin 13
void setup() {
  pinMode(ledPin, OUTPUT); // set the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH); // turn the LED on
  delay(1000); // wait for a second
  digitalWrite(ledPin, LOW); // turn the LED off
  delay(1000); // wait for a second
}
```

This code sets the LED pin as an output and then toggles the LED on and off with a one-second delay between each state change.

ADVANTAGES:

1. Low Power Consumption: LEDs are very energy-efficient and consume very little power. Blinking an LED is a simple and effective way to conserve power in battery-powered devices.
2. Attention-Grabbing: Blinking LEDs are eye-catching and draw attention to themselves. They are commonly used in emergency situations, as well as in advertising and marketing campaigns.
3. Easy to Control: Blinking an LED is a straightforward process and can be easily controlled using a microcontroller or a simple circuit.
4. Long Lifespan: LEDs have a long lifespan compared to traditional light bulbs. Blinking LEDs can be used in a variety of applications, including traffic lights and warning signals, without the need for frequent replacement.

APPLICATIONS:

1. Signaling: LEDs are commonly used as signaling devices, and blinking can be an effective way to attract attention or convey information. For example, flashing LED lights can indicate a warning or signal for emergency situations.
2. Decoration: Blinking LEDs can also be used for decorative purposes. For instance, blinking lights on a Christmas tree or a disco ball can create a festive and entertaining atmosphere.
3. Advertising: Blinking LEDs can be used in advertising signs to make them more eye-catching and attractive. For instance, blinking lights can be used to highlight specific messages or products.
4. Security: Blinking LEDs can be used in security systems to indicate that the system is armed or disarmed.

c**CONCLUSION:**

Activity:04

Date:

DESIGN THE CONTACTLESS DOOR BELL USING ARDUINO UNO

AIM:

To Design the CONTACTLESS DOOR BELL using arduino uno.

APPARATUS:

- Arduino uno
- Data cable
- Jumber cables
- Bread board
- IR Sensor
- Buzzer

DESCRIPTION:

IR sensor:

An IR sensor, also known as an infrared sensor, is a type of electronic device that can detect and measure infrared radiation. Infrared radiation is a type of electromagnetic radiation with wavelengths longer than those of visible light, and it is commonly produced by heat or light sources.IR sensors are used in a wide range of applications, including temperature sensing, motion detection, and proximity sensing. They work by detecting the infrared radiation emitted by an object and converting it into an electrical signal that can be processed by a computer or other electronic device.

IR sensors are commonly used in security systems, such as motion detectors and burglar alarms, as well as in industrial applications, such as temperature monitoring in manufacturing processes. They are also used in consumer electronics, such as remote controls and touchless faucets.

BUZZER:

A buzzer is an electrical device that produces a buzzing sound when it is activated. It typically consists of an electromechanical oscillator, which generates the sound by rapidly vibrating a small diaphragm or armature. Buzzers can be used in a wide range of applications, such as alarms, timers, games, and other electronic devices that require an audible alert or warning signal. They come in various sizes and shapes, and can be powered by batteries or by an external power source. Some buzzers also have the capability to produce different tones or frequencies, allowing for more versatile use in different settings.

Here are the steps to design the contactless doorbell using IR sensor and Arduino Uno:

1.Connect the IR sensor module to the breadboard and connect the VCC and GND pins of the IR sensor module to the +5V and GND pins of the Arduino Uno respectively.

2.Connect the output pin of the IR sensor module to pin 5 of the Arduino Uno.

3.Connect the piezo buzzer to the breadboard and connect one of its pins to pin 7 of the Arduino Uno. Connect the other pin of the piezo buzzer to the GND pin of the Arduino Uno via a 220 ohm resistor.

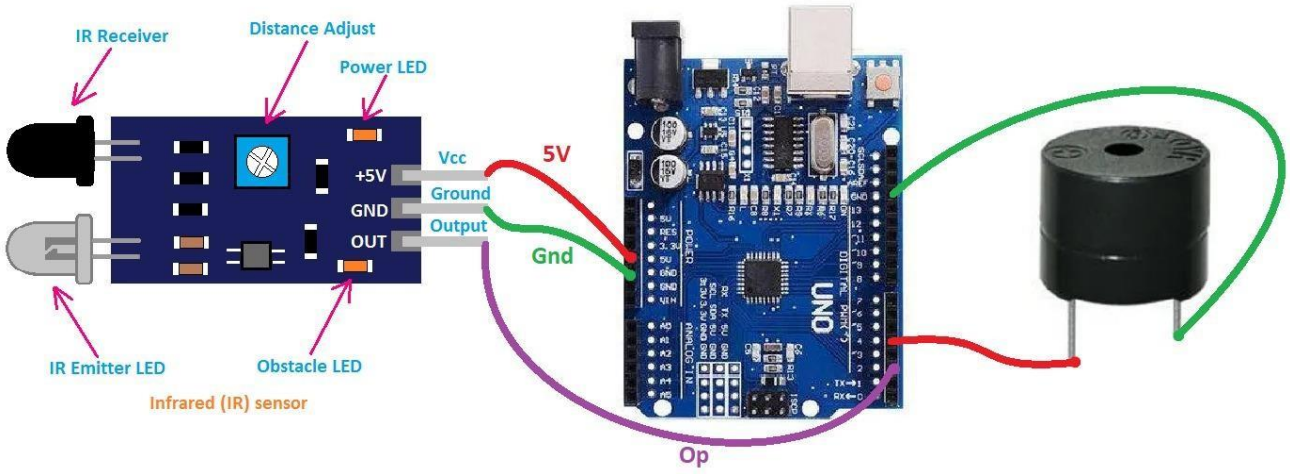
4Upload the following code to the Arduino Uno:

CODE:

```
#define irsensor 5
#define buzzer 7
void setup() {
  // put your setup code here, to run once:
pinMode(irsensor,INPUT);
pinMode(buzzer,OUTPUT);
Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
int raw = digitalRead(irsensor);
Serial.println(raw);
if(raw==0){
  digitalWrite(buzzer,1);
}else{
  digitalWrite(buzzer,0);
3
}
delay(500);
```

5.Once you have uploaded the code, your contactless doorbell is ready to use. When someone passes by the IR sensor, the buzzer will sound for 1 second and then stop for 1 second. You can adjust the delay time in the code as per your requirements**.**

CIRCUIT DIAGRAM:



ADVANTAGES:

1. No physical contact required: The contractless doorbell eliminates the need for physical contact, which is especially important in a post-COVID-19 world where contactless devices are becoming increasingly popular.

2. Cost-effective: Compared to traditional doorbells, contractless doorbells using Arduino Uno are cost-effective and easy to install.

3. Improved security: Contractless doorbells can be equipped with sensors and cameras that enhance security by allowing homeowners to see who is at the door before opening it.

APPLICATIONS:

1. Residential: Contractless doorbells using Arduino Uno are ideal for residential homes as they are easy to install and cost-effective.

2. Commercial: The contractless doorbell can be used in commercial settings, such as offices and stores, to enhance security and prevent unauthorized access.

3. Contractless doorbells can be used in healthcare settings to minimize the spread of germs and viruses.

4. Hospitality: Contractless doorbells can be used in hotels and other hospitality settings to provide guests with a convenient and hygienic way to request services.

CONCLUSION:

Activity:05

Date:

DESIGN THE OBJECT THEFT DETECTOR USING ARDUINO UNO

AIM:

To Design the using OBJECT THEFT DETECTOR arduino uno.

APPARATUS:

- Arduino uno
- Data cable
- Jumber cables
- Bread board
- IR Sensor
- Buzzer
- led

DESCRIPTION:

IR sensor:

An IR sensor, also known as an infrared sensor, is a type of electronic device that can detect and measure infrared radiation. Infrared radiation is a type of electromagnetic radiation with wavelengths longer than those of visible light, and it is commonly produced by heat or light sources.IR sensors are used in a wide range of applications, including temperature sensing, motion detection, and proximity sensing. They work by detecting the infrared radiation emitted by an object and converting it into an electrical signal that can be processed by a computer or other electronic device.

IR sensors are commonly used in security systems, such as motion detectors and burglar alarms, as well as in industrial applications, such as temperature monitoring in manufacturing processes. They are also used in consumer electronics, such as remote controls and touchless faucets.

BUZZER:

A buzzer is an electrical device that produces a buzzing sound when it is activated. It typically consists of an electromechanical oscillator, which generates the sound by rapidly vibrating a small diaphragm or armature. Buzzers can be used in a wide range of applications, such as alarms, timers, games, and other electronic devices that require an audible alert or warning signal. They come in various sizes and shapes, and can be powered by batteries or by an external power source. Some buzzers also have the capability to produce different tones or frequencies, allowing for more versatile use in different settings.

LED:

LED stands for Light Emitting Diode. It is a semiconductor device that emits light when an electric current is passed through it. LEDs are widely used in various electronic devices such as traffic lights, mobile phones, televisions, and computer monitors, as well as in lighting applications such as light bulbs, lamps, and decorative lighting.

LEDs have several advantages over traditional incandescent and fluorescent bulbs, including lower energy consumption, longer lifespan, and higher durability. They also have the ability to produce light of different colors, making them ideal for use in displays and signage.

LEDs come in different shapes and sizes, and can be used for a wide range of applications. They are also environmentally friendly as they do not contain toxic materials and are recyclable.

Here are the steps to design th**e object theft detector Arduino Uno:**

1..Connect the components: Connect the IR sensor module to the Arduino Uno using jumper wires. Connect the buzzer to the breadboard and connect one leg of the LED to the Arduino Uno via a resistor.

2.Write the code: You will need to write a code for the Arduino Uno that will read the input from the IR sensor module and turn on the buzzer and LED if the sensor detects any movement. Here is a sample code:

CODE:

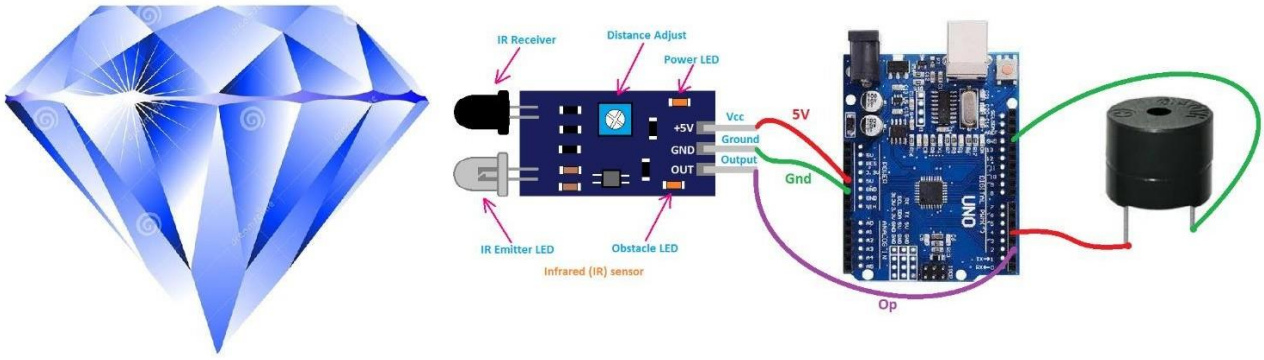
```
#define irsensor 4
#define buzzer 6
#define led 8
void setup() {
  // put your setup code here, to run once:
pinMode(irsensor,INPUT);
pinMode(buzzer,OUTPUT);
pinMode(led,OUTPUT);
Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
int raw = digitalRead(irsensor);
Serial.println(raw);
if(raw==1)
 {
```

```
 digitalWrite(buzzer,1);
 digitalWrite(led,1);
}
else
{
 digitalWrite(buzzer,0);
 digitalWrite(led,0);
}
delay(500);
}
```

3.Test the circuit: Connect the Arduino Uno to your computer and upload the code to the board. Then test the circuit by moving your hand in front of the IR sensor module. If everything is working correctly, the LED should light up and the buzzer should sound.

4.Finalize the design: Once you have tested the circuit, you can finalize the design by creating a case for the circuit and mounting it in the location where you want to use it as an object theft detector.

CIRCUIT DIAGRAM:



ADVANTAGES:

1.Low Cost: Arduino Uno microcontrollers are relatively inexpensive and readily available in the market, making the object theft detector using Arduino Uno an affordable solution.

2. Customizable: The Arduino Uno is an open-source platform that allows users to modify its firmware and hardware, making it highly customizable for specific needs and applications.

3. Easy to Use: The Arduino Uno is relatively easy to program and use, making it accessible to hobbyists and enthusiasts with little or no programming experience.

APPLICATIONS:

1. Home Security: An object theft detector using Arduino Uno can be used to detect and prevent theft in homes by monitoring the movement of objects and sending alerts when an unauthorized person is detected.

2. Asset Tracking: Companies can use an object theft detector using Arduino Uno to track their assets and prevent theft.

3. Retail Stores: Retail stores can use an object theft detector using Arduino Uno to monitor the movement of high-value items, such as jewelry or electronics, and alert security personnel in case of any theft attempt.

CONCLUSION:

Activity:06

Date:

DESIGN THE ESTIMATING INDOOR POPULATION DENSITY USING ARDUINO UNO

AIM:

To Design the using ESTIMATING INDOOR POPULATION DENSITY arduino uno.

APPARATUS:

- Arduino uno
- Data cable
- Jumber cables
- Bread board
- IR Sensor
- led

DESCRIPTION:

Determine the area to be monitored: Decide on the specific room or space to be monitored for population density.Choose appropriate sensors: Select the sensors that will be used to detect and measure the presence of people in the area. Options could include motion sensors, ultrasonic sensors, or infrared sensors.Connect sensors to the Arduino Uno: Connect the selected sensors to the appropriate input pins on the Arduino Uno.Write the code: Use the Arduino Integrated Development Environment (IDE) to write the code that will control the sensors and perform the population density calculations.Calibrate the sensors: Use a small group of people to calibrate the sensors and ensure that they are accurately detecting the presence of people in the area.
Test and refine: Test the system in real-world conditions and refine the code as necessary to improve accuracy and reliability.Determine population density: Use the data collected by the sensors and the calculations performed by the Arduino Uno to determine the population density in the monitored area.Note that the specific steps and sensors used may vary depending on the requirements of the project and the available resources.

Here are the steps to design:

1. Connect the infrared sensors to the Arduino Uno board as follows:
2. Connect the VCC pin of the infrared sensors to the 5V pin of the Arduino board.
3. Connect the GND pin of the infrared sensors to the GND pin of the Arduino board.
4. Connect the OUT pin of the infrared sensors to digital pins on the Arduino board (e.g. pins 2 and 3).
5. Connect the LED and resistor to the Arduino Uno board as follows:
6. Connect the longer leg of the LED to pin 13 of the Arduino board.
7. Connect the shorter leg of the LED to one end of the resistor.
8. Connect the other end of the resistor to the GND pin of the Arduino board.
9. Write a program in the Arduino IDE that reads the output of the infrared sensors and turns on the LED if the number of people in the room exceeds a certain threshold.

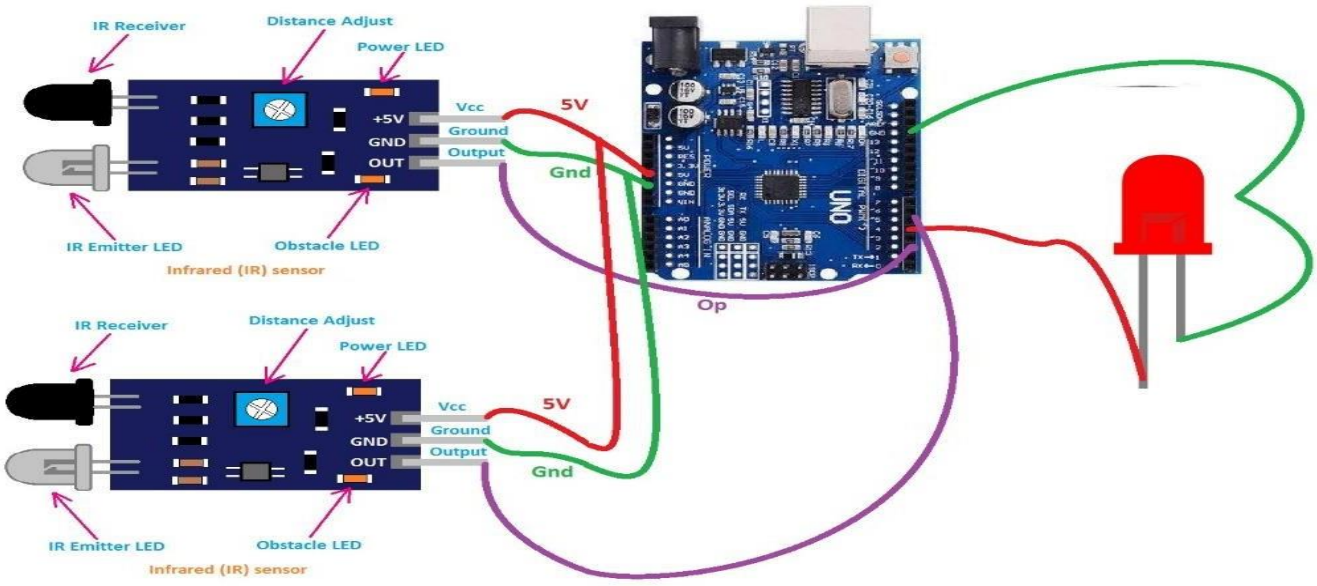
10. Upload the program to the Arduino board and test the system by moving around the room and observing the LED.
11. Sample Arduino Code:

CODE:

```
#define irin 2
#define irout 4
# define led 13
void setup() {
  // put your setup code here, to run once:
pinMode(irin,INPUT);
pinMode(irout,INPUT);
pinMode(led,OUTPUT);
digitalWrite(led,0);
Serial.begin(9600);
}
int count;
void loop() {
  // put your main code here, to run repeatedly:
if (digitalRead(irin)==0)
{
  count=count+1;
  Serial.print("count=");
  Serial.println(count);
  delay(1000);
}
if (digitalRead(irout)==0)
{
count=count-1;
Serial.print("count=");
Serial.println(count);
delay(1000);
```

```
}
if (count>1)
{
digitalWrite(led,1);
delay(1000);
}
else
{
  digitalWrite(led,0);
  delay(1000);
}
}
```

CIRCUIT DIAGRAM:



ADVANTAGES:

1.Low cost: Arduino Uno is a low-cost microcontroller board that can be easily purchased and programmed, making it an affordable option for estimating indoor population density.

2. Customizability: Arduino Uno can be easily programmed to suit the specific needs of a particular indoor environment. The flexibility of the device allows it to adapt to various sensor types and configurations, enabling accurate and customizable population density estimations.

3. Easy installation: Arduino Uno is small and easy to install, making it ideal for use in crowded indoor spaces where installing more cumbersome devices may be impractical.

APPLICATIONS:

1. Building Management: Monitoring the occupancy of rooms in buildings can help optimize energy consumption by adjusting heating, ventilation, and air conditioning (HVAC) systems according to the actual number of occupants.

2. Safety and Security: Estimating indoor population density can be useful for emergency response planning and evacuation procedures. It can also help detect overcrowding in buildings and prevent accidents.

3. Retail Analytics: Tracking the number of people visiting retail stores can help retailers optimize their marketing strategies and store layouts to increase sales and customer satisfaction.

CONCLUSION:

‘

Activity:07

Date:

DESIGN THE AUTOMATIC STREET LIGHT SYSTEM USING ARDUINO UNO

AIM:

To Design the AUTOMATIC STREET LIGHT SYSTEM using arduino uno.

APPARATUS:

- Arduino uno
- Data cable
- Jumber cables
- Bread board
- LDR
- led

DESCRIPTION

An automatic street light system is designed to automatically turn on and off the streetlights based on certain criteria such as the presence of vehicles or pedestrians on the road, the time of day, or the ambient light level. The underlying theory behind this system involves using sensors and controllers to detect and respond to changes in the environment.

One common approach to implementing an automatic street light system is to use a combination of motion sensors and light sensors. Motion sensors can detect the presence of vehicles or pedestrians on the road, while light sensors can measure the ambient light level. Based on this information, the system can turn on the streetlights when it detects motion and the ambient light level is below a certain threshold. Similarly, it can turn off the streetlights when there is no motion detected and the ambient light level is above a certain threshold.

Another approach is to use a centralized control system that monitors and manages the streetlights across an entire city or region. This system can use data from various sensors, including traffic flow sensors, weather sensors, and energy consumption sensors, to optimize the operation of the streetlights. For example, the system can adjust the brightness of the lights based on the level of traffic flow, or turn off lights in areas where there is low pedestrian activity during certain hours of the day to conserve energy.

Overall, the theory behind automatic street lights involves using sensors and controllers to detect and respond to changes in the environment, with the goal of optimizing the operation of the streetlights to improve safety, reduce energy consumption, and enhance the overall quality of life in urban areas.

Here are the steps to design:

Step 1:
First, connect the LDR to the breadboard and then connect one end of the 10kΩ resistor to the same row as the LDR's one end. Next, connect the other end of the resistor to the GND pin of the Arduino Uno.
Step 2:

Now, connect the LDR's other end to the analog pin A0 of the Arduino Uno.
Step 3:
Next, connect the positive (longer) leg of the LED to digital pin 13 of the Arduino Uno, and the negative (shorter) leg to GND.
Step 4:

Now, open the Arduino IDE on your computer and write the code to turn on the LED when it's dark and turn it off when it's bright. Here's an example code:

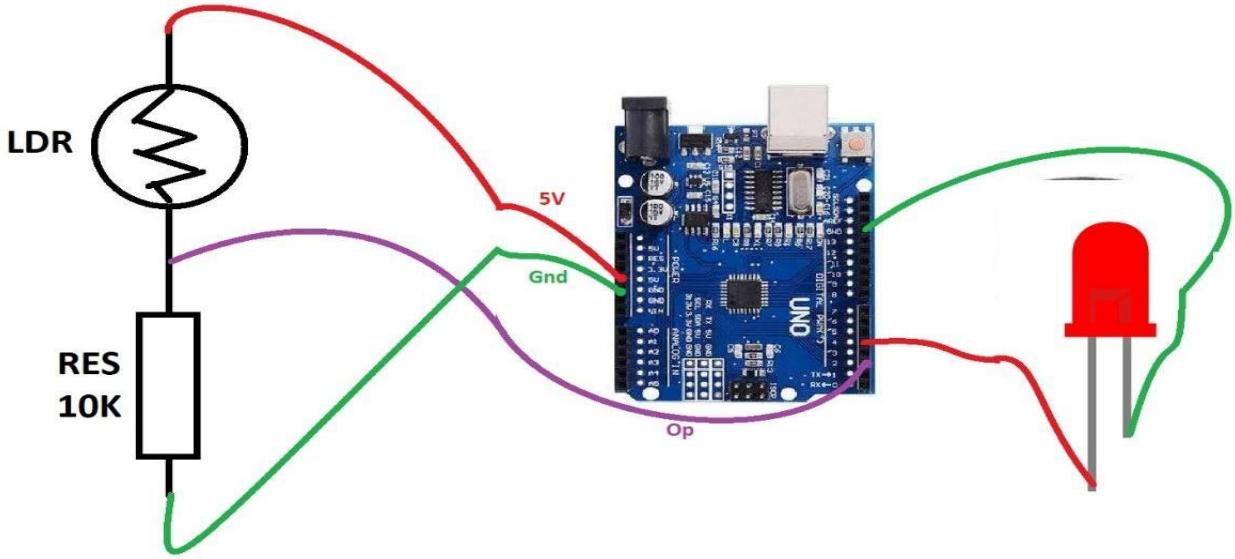
CODE:

```
#define ldr_pin A0

#define led_pin 7

void setup() {

  // put your setup code here, to run once:


pinMode(ldr_pin,INPUT);

pinMode(led_pin,INPUT);

Serial.begin(9600);

}


void loop() {

  // put your main code here, to run repeatedly:

int ldr_feed =analogRead(ldr_pin);

Serial.print("ldr feed=");

Serial.println(ldr_feed);

if(ldr_feed<=50)

digitalWrite(led_pin,1);

else

digitalWriteb(led_pin,0);

}
```

Step 5:
Upload the code to the Arduino Uno board by clicking the "Upload" button in the Arduino IDE.

Step 6: Now, power up the Arduino Uno board using a USB cable or a battery, and place the LDR in a dark environment (such as covering it with a cloth). You should see that the LED turns on automatically. When you remove the cloth and expose the LDR to light, the LED should turn off.

CIRCUIT DIAGRAM:



ADVANTAGES:

1. Energy Efficiency: Automatic street lights using Arduino Uno can help in reducing energy consumption as the lights are only turned on when required. This can lead to significant energy savings and lower electricity bills.

2. Cost Savings: By using an Arduino Uno-based system, the cost of street lighting can be reduced, as it eliminates the need for manual intervention, maintenance and supervision.

3. Improved Safety: Automatic street lights can improve safety on the roads by ensuring that there is always adequate lighting available. This can help to reduce accidents and other safety hazards.

4. Easy to Install: Arduino Uno-based street lighting systems are easy to install and can be implemented quickly, making it an ideal solution for cities and towns looking to upgrade their existing street lighting infrastructure.

APPLICATIONS:

1. Street Lighting: Automatic street lights using Arduino Uno can be used to light up streets, roads, highways, and other public areas.

2. Parking Lots: Arduino Uno-based street lighting systems can also be used in parking lots to provide adequate lighting for drivers and pedestrians.

3. Commercial Buildings: Automatic lighting systems can be installed in commercial buildings to provide adequate lighting for the building's exterior, parking lots, and walkways.

CONCLUSION:

Activity:08

Date:

DESIGN THE FIRE DETECTION AND ALARM SYSTEM USING ARDUINO UNO

AIM:

To Design the FIRE DETECTION AND ALARM SYSTEM using arduino uno.

APPARATUS:

- Arduino uno
- Data cable
- Jumber cables
- Bread board
- MQ-2 Gas sensor
- led

DESCRIPTION:

Fire detection and alarm systems are critical for protecting people, property, and assets in buildings. The theory behind fire detection and alarm systems is based on the principles of fire science and the behavior of fire.There are four essential components of a fire: heat, fuel, oxygen, and chemical chain reaction. Fire detection and alarm systems work by detecting one or more of these components and alerting people to the presence of a fire. The two primary types of fire detection systems are heat detectors and smoke detectors.Heat detectors operate by sensing the temperature in an area and triggering an alarm when the temperature rises above a predetermined threshold. Smoke detectors operate by sensing the presence of smoke in an area and triggering an alarm when the smoke density reaches a certain level. In addition to heat and smoke detectors, fire detection and alarm systems may also include flame detectors, which sense the presence of flames, and gas detectors, which detect the presence of gases that may indicate a fire. Once a fire is detected, the fire alarm system will sound an audible alarm and may also trigger visual alerts, such as flashing lights, to warn people in the building. The fire alarm system may also automatically notify the local fire department, which can dispatch firefighters to the scene. Fire detection and alarm systems are designed to be reliable and resilient, with redundant sensors and backup power supplies to ensure that they continue to function even in the event of a power outage or other failure. Regular testing and maintenance are essential toensure that these systems are operating correctly and can provide effective protection in the event of a fire. Overall, the theory behind fire detection and alarm systems is based on a deep understanding of fire science and the behavior of fire, combined with advanced technology and engineering to create a robust and reliable system for protecting people and property from the devastating effects of fire.

Here are the steps to design:

1.Connect the MQ-2 gas sensor module to the Arduino Uno board using jumper wires as follows:

2.VCC pin to the 5V pin on the Arduino Uno board

3.GND pin to the GND pin on the Arduino Uno board

4.D0 pin to digital pin 2 on the Arduino Uno board

5.A0 pin to analog pin 0 on the Arduino Uno board

7.Connect the buzzer to the Arduino Uno board using jumper wires as follows:

8. Positive pin to digital pin 3 on the Arduino Uno board

9. Negative pin to the GND pin on the Arduino Uno board

10. Connect the LED to the Arduino Uno board using jumper wires as follows:

11. Positive pin (longer leg) to digital pin 4 on the Arduino Uno board

12. Negative pin (shorter leg) to the GND pin on the Arduino Uno board

13. Write the code for the fire detection and alarm system in the Arduino Integrated Development Environment (IDE). Here's a basic code example

CODE:

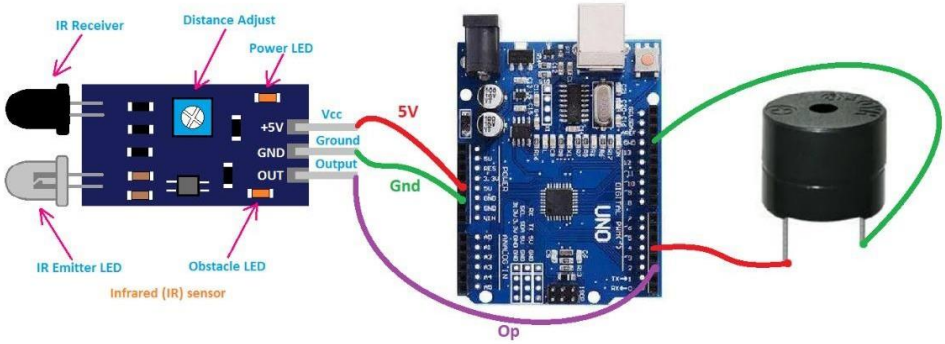
```
#define fire 2
#define led 13
#define buz 4
void setup() {
  // put your setup code here, to run once:
pinMode(fire,INPUT);
digitalWrite(led,OUTPUT);
pinMode(buz,OUTPUT);
digitalWrite(led,0);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(digitalRead(fire)==0)
{
  digitalWrite(buz,1);
  digitalWrite(led,1);
delay(1000);
}
else
{
  digitalWrite(buz,0);
```

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

14. This code reads the analog input from the MQ-2 gas sensor module and checks if the gas level exceeds the threshold value of 500 (which can be adjusted as needed). If the threshold is exceeded, the buzzer and LED are turned on to alert the user of a potential fire. If the gas level is below the threshold, the buzzer and LED are turned off.

15. Upload the code to the Arduino Uno board and test the fire detection and alarm system by exposing the MQ-2 gas sensor module to smoke or other combustible gases (such as a lighter or match). The buzzer and LED should turn on when the threshold gas level is reached.

CIRCUIT DIAGRAM:



ADVANTAGES:

1. Fire detection and alarm systems are crucial in protecting lives and property from fire-related incidents. Arduino Uno is a microcontroller board that can be used to build fire detection and alarm systems. Some advantages and applications of using Arduino Uno in fire detection and alarm systems include:

2. Low-cost: Arduino Uno is a low-cost microcontroller board that is easily accessible, making it an affordable option for building fire detection and alarm systems.

3. Easy to program: Arduino Uno is programmed using the Arduino Integrated Development Environment (IDE), which is easy to learn and use. This makes it easy for even novice programmers to build fire detection and alarm systems.

4. Customizable: Arduino Uno is highly customizable, allowing developers to tailor their fire detection and alarm systems to specific needs and requirements

APPLICATIONS:

1. Residential homes: Fire detection and alarm systems built using Arduino Uno can be installed in residential homes to provide early warning and protection against fire-related incidents.

2. Commercial buildings: Fire detection and alarm systems built using Arduino Uno can be installed in commercial buildings, such as offices, hospitals, and schools, to provide early warning and protection against fire-related incidents.

3. Industrial facilities: Fire detection and alarm systems built using Arduino Uno can be installed in industrial facilities, such as factories and warehouses, to provide early warning and protection against fire-related incidents.

CONCLUSION:

Activity:09

Date:

DESIGN THE WEATHER MOINTORING SYSTEM USING ESP-32

AIM:

To Design the WEATHER MOINTORING SYSTEM using Esp-32.

APPARATUS:

- ESP 32 board
- Temperature and humidity sensor
- Jumper wires
- Power supply
- Data cable
- Blynk app

DESCRIPTION:

1. Hardware Setup: Connect the ESP32 to the weather sensors (such as DHT11 or BME280) that measure the temperature, humidity, pressure, and other parameters. Also, connect the ESP32 to Wi-Fi or Ethernet for Internet connectivity.

2. Software Setup: Install the required libraries for ESP32 and Blynk, and create an account on the Blynk app to get the authentication token.

3. Code Development: Write the code for ESP32 in Arduino IDE, using the libraries such as Adafruit_Sensor.h, DHT.h, BME280.h, and Blynk.h. Define the pins for the sensors and configure the Wi-Fi or Ethernet connection. Also, define the Blynk virtual pins for each sensor parameter, and map them to the widgets on the Blynk app.

4. Data Transmission: Use the Blynk.run() function in the code to establish a connection between ESP32 and Blynk app. This allows the data to be transmitted from ESP32 to the Blynk server in real-time, and displayed on the widgets such as gauge, graph, or value display.

5. Data Analysis: Analyze the data on the Blynk app using the widgets or scripts. You can create alerts or notifications based on the threshold values for each parameter, or use the data to forecast the weather conditions.

6. User Interface: Customize the Blynk app interface by adding widgets or tabs for different sensors, and arranging them in a logical way. You can also add widgets for controlling the ESP32, such as turning on/off the sensors or adjusting the sampling frequency.

7. Testing and Deployment: Test the weather monitoring system in a real-world environment, and fine-tune the parameters or thresholds as needed. Deploy the system in the desired location, and monitor the weather conditions remotely using the Blynk app.

Here are the steps to design:

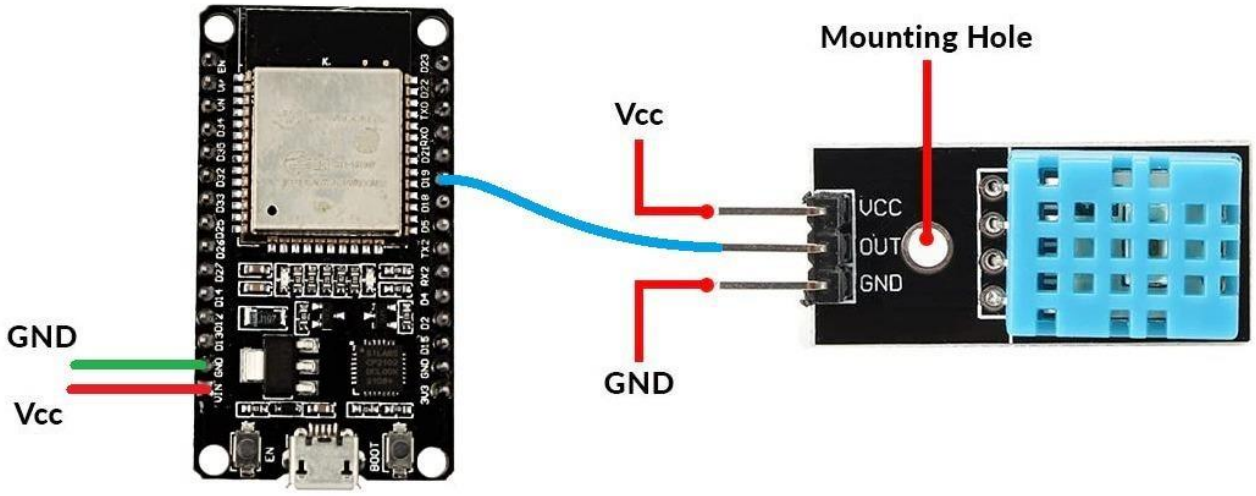
1. Connect the DHT11 sensor to the ESP32 board by connecting the VCC pin to 3.3V, GND to GND, and data pin to GPIO 4.
2. Connect the BMP280 sensor to the ESP32 board by connecting the VCC pin to 3.3V, GND to GND, SDA pin to GPIO 21, and SCL pin to GPIO 22.
3. Open the Arduino IDE and create a new sketch.
4. Add the required libraries for the DHT11 sensor (such as Adafruit_Sensor.h and DHT.h) and BMP280 sensor (such as Adafruit_BMP280.h).
5. Define the pins for the sensors and configure the Wi-Fi connection by adding the SSID and password for your Wi-Fi network.
6. Initialize the Blynk app by adding the authentication token and selecting the desired frequency of data transmission.
7. In the loop() function, read the data from the sensors and send it to the Blynk app using the virtual pins defined in the Blynk app.
8. Customize the Blynk app interface by adding widgets such as gauges, graphs, and value displays for each sensor parameter.
9. Save and upload the code to the ESP32 board.
10. Once the ESP32 is connected to the sensors and Wi-Fi network, and the Blynk app is initialized, the system will continuously read the data from the sensors and transmit it to the Blynk app in real-time. You can monitor the weather conditions remotely using the Blynk app.

CODE:

```
#define BLYNK_TEMPLATE_ID "TMPLnqfSk-af"
#define BLYNK_TEMPLATE_NAME "weathermonitering"
#define BLYNK_AUTH_TOKEN "l7kQWI0WDjuNBzASG-y-I5sPBHRnBJoK"
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>

char ssid[] = "LOKESH";
char pass[] = "7569121361";
```

```
#define DHTPIN 18          // What digital pin we're connected to
#define DHTTYPE DHT11      // DHT 11
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  Serial.begin(115200);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  dht.begin();
}

void loop()
{
  Blynk.run();
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
  Blynk.virtualWrite(V1, h);
  Blynk.virtualWrite(V2, t);
  delay(1000);
}
```

CIRCUIT DIAGRAM:



ADVANTAGES:

1.Early warning: Weather monitoring systems can provide early warnings for extreme weather events such as hurricanes, tornadoes, floods, and storms. This allows people to prepare and evacuate early, reducing the risk of loss of life and property.

2.Safety: Monitoring weather patterns can help to identify hazardous conditions such as lightning strikes, high winds, and heavy rain, and inform people to take necessary safety precautions.

3Agriculture: Weather monitoring systems provide farmers with critical information on rainfall patterns, temperature changes, and humidity levels that can help them optimize crop yields and increase productivity.

4.Cost-effective: IoT-based systems can be more cost-effective than traditional weather monitoring systems that require expensive equipment and maintenance.

APPLICATIONS:

1.Agriculture: Farmers rely heavily on weather monitoring to make informed decisions about planting, harvesting, and irrigation. They need to know the soil moisture, temperature, and humidity to ensure healthy crop growth.

2.Aviation: Weather monitoring is essential for safe aviation operations. Pilots need to know the current and forecasted weather conditions to determine flight paths and adjust the altitude, speed, and direction of the aircraft.

3.Energy: Weather monitoring is crucial for energy production and distribution. Energy companies use weather data to optimize the production of renewable energy, such as solar and wind power.

CONCLUSION:

Activity:10

Date:

DESIGN THE AUTOMATIC SAFETY HOMEBELL SYSTEM WITH MESSAGE ENABLED FEAUTERS USING ESP-32

AIM:

To Design the AUTOMATIC SAFETY HOMEBELL SYSTEM WITH MESSAGE ENABLED FEAUTERS using Esp-32.

APPARATUS:

- ESP 32 board
- Jumper wires
- Power supply
- Data cable
- Blynk app
- Wifi router

DESCRIPTION:

Motion sensors: These sensors would detect movement in the home and trigger the system to activate.Door and window sensors: These sensors would detect if a door or window is opened or closed and alert the system accordingly.Alarm system: This system would produce a loud, audible alert if any of the sensors are triggered.Message-enabled features: This would allow the system to send notifications to a homeowner's mobile device or computer, alerting them to any potential safety concerns or triggering events.Control panel: A control panel would allow the homeowner to arm and disarm the system as needed, as well as adjust settings for the motion sensors and message-enabled features.Backup power supply: This would ensure that the system remains functional even in the event of a power outage.Overall, this system would provide an effective way to monitor and secure a home, with both audible and message-enabled alerts to keep the homeowner informed of any potential safety concerns. The motion sensors and door/window sensors would work together to detect any potential intruders or other safety concerns, while the message-enabled features would allow the homeowner to receive notifications even when they are away from home. The control panel would provide a simple way for the homeowner to manage the system and adjust settings as needed. And the backup power supply would ensure that the system remains functional even in the event of a power outage.

Here are the steps to design:

1.Connect the ESP32 to the breadboard and connect the LED and resistor to the GPIO pin of the ESP32.

2.Connect the passive buzzer to the ESP32's GPIO pin.

3.Connect the push button to the ESP32's GPIO pin.

4.Connect the ESP32 to the Wi-Fi router using the Blynk app.

5.Set up a Blynk account and create a new project.

6. Add a notification widget in the Blynk app.

7. In the code, use the Blynk library to connect the ESP32 to the Blynk app and add the code to check for the button press.

8. When the button is pressed, turn on the LED and activate the passive buzzer to indicate that the system has been triggered.

9. Send a notification to the Blynk app to inform the user that the system has been triggered.

10. Use the message feature in the Blynk app to send a message to the user.

11. Finally, test the system and make sure it works properly.

Here's an example code

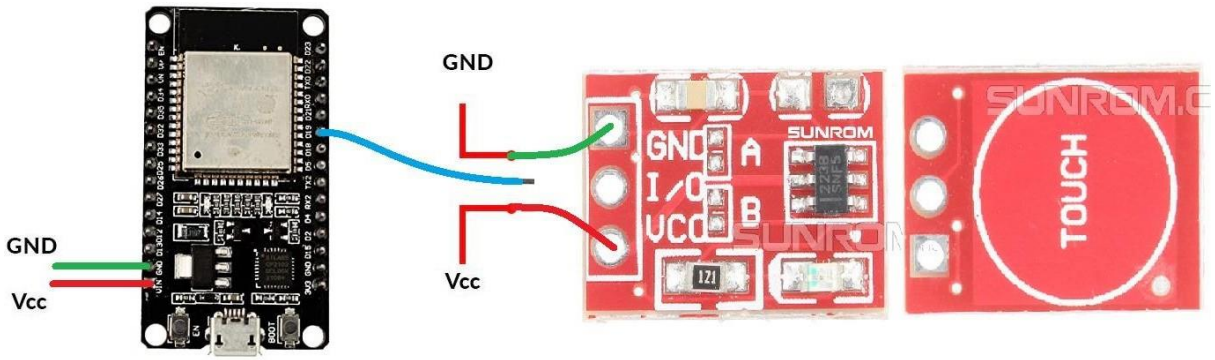
CODE:

```
#define BLYNK_TEMPLATE_ID           "TMPxxxxxx"
#define BLYNK_TEMPLATE_NAME         "Device"
#define BLYNK_AUTH_TOKEN            "YourAuthToken"


#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>


#define buz 16
#define capsen 18


long duration;
float distance;


char ssid[] = "your_hotspot_name";
char pass[] = "Your_Hotspot_Password";


void setup()
{
  // Debug console
  Serial.begin(9600);
```

```
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  pinMode(buz, OUTPUT);
  pinMode(cap,INPUT);
}

void loop()
{
  Blynk.run();

  if (digitalRead(capsen) = LOW)
  {
    digitalWrite(buz, HIGH);
    Blynk.logEvent("notify","Someone is at Door");
    while(digitalRead(cap) = LOW);
  }
  else
  {
    digitalWrite(buz, LOW);
  }
}
```

Note: Replace the your_template_id, your_device_name, your_auth_token, WIFI_SSID, and WIFI_PASSWORD with your own values.

CIRCUIT DIAGRAM:



ADVANTAGES:

1.Real-time updates: Automatic safety home bell systems can provide real-time updates about the status of your home. For example, you can receive notifications when your children arrive home from school or when a package is delivered.

2.Integration with other smart home devices: An automatic safety home bell system can be integrated with other smart home devices, such as security cameras, smart locks, and smart lights. This can provide a more comprehensive home security solution.

3.Easy to use: Automatic safety home bell systems are easy to install and use. They can be controlled using a smartphone app, and you can receive notifications and alerts directly on your phone.

APPLICATIONS:

1.Doorbell with messaging: The system can also have a feature where when someone rings the doorbell, the message-enabled feature can provide a pre-recorded or typed message of the homeowner to know who is at the door before answering it, enhancing safety from strangers or unwelcome visitors.

2.Child Safety: The system can also be set up to monitor children's movements and send a message to the homeowner's phone if they leave the house or go to areas of the house that are off-limits. This can help prevent accidents and ensure the safety of the children.

CONCLUSION:

Activity:11

Date:

DESIGN THE APPLICATION FOR ALZHEMER'S ON SMART MEDICAL REMAINDER USING ESP-32

AIM:

To Design the APPLICATION FOR ALZHEMER'S ON SMART MEDICAL REMAINDER using Esp-32.

APPARATUS:

- ESP 32 board
- Jumper wires
- Power supply
- Data cable
- Blynk app
- Real time clock module
- Lcd display

DESCRIPTION:

Medication dispensers: These are devices that can be programmed to dispense medications at specific times. Some dispensers are designed to hold multiple medications and can be programmed to dispense the right dose at the right time.

Sensors: These are devices that can be used to monitor the patient's health status, such as blood pressure, heart rate, or glucose levels. Sensors can also detect when the patient has taken their medication.

Connectivity: Smart medical reminders are connected to the internet, which enables them to send alerts to doctors or family members if there are any issues.

Mobile applications: Some smart medical reminders come with mobile applications that can be used to program the device, view medication schedules, and receive alerts.

The benefits of a smart medical reminder using IoT are numerous. It can help patients manage their medications more effectively, reducing the risk of missed doses or incorrect doses. It can also help doctors monitor their patients' health status more closely, allowing for earlier intervention if there are any issues.

Here are the steps to design:

Step 1: Gather materials

ESP32 board, Blynk app, Breadboard, Jumper wires, 5V power supply

Step 2: Connect ESP32 to Blynk app

Download and install the Blynk app on your smartphone

Create a new project and select ESP32 as the device

Get the authentication token and save it for later

Step 3: Set up the circuit

Connect the ESP32 board to the breadboard

Connect the VCC pin of the ESP32 to the positive rail of the breadboard

Connect the GND pin of the ESP32 to the negative rail of the breadboard

Connect the D7 pin of the ESP32 to the positive leg of an LED

Connect the negative leg of the LED to a 220 ohm resistor

Connect the other end of the resistor to the negative rail of the breadboard

Step 4: Code the ESP32

Open the Arduino IDE and create a new sketch

Install the Blynk library by going to Sketch > Include Library > Manage Libraries and searching for Blynk

Replace "your_template_id" with the ID of the project you created in the Blynk app

Replace "your_device_name" with a name for your device

Replace "your_auth_token" with the authentication token you obtained earlier

Replace "your_SSID" and "your_password" with the credentials for your Wi-Fi network

Upload the code to the ESP32 board

Step 5: Test the application

Open the Blynk app and go to the project you created

Add a button widget and set it to V0

Press the button and observe the LED on the breadboard blinking

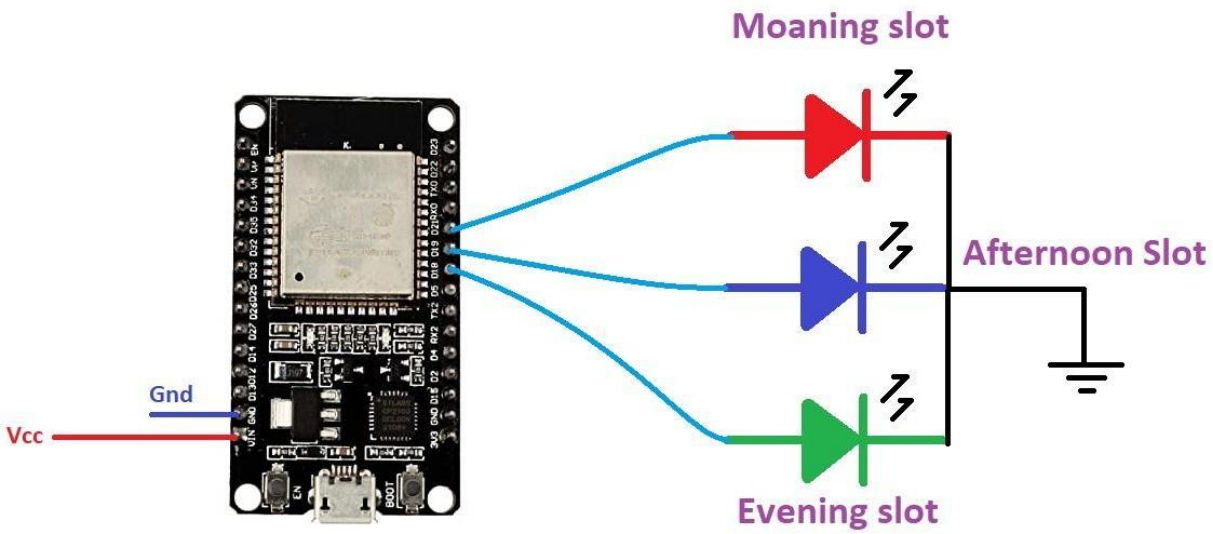
CODE:

```
#define BLYNK_TEMPLATE_ID "TMPLfyYrlPX8"
#define BLYNK_TEMPLATE_NAME "MEDICAL REMINDER"
#define BLYNK_AUTH_TOKEN "PC41Zk39KhZuOFb3MGscW4rIAOS-y_XI"


#define morn 19
#define after 18
#define night 17
#define buz 16
```

```
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "LOKESH";
char pass[] = "7569121361";

void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  pinMode(morn,OUTPUT);
  pinMode(after,OUTPUT);
  pinMode(night,OUTPUT);
}

void loop()
{
   Blynk.run();
   digitalWrite(morn,HIGH);
   digitalWrite(after,LOW);
   digitalWrite(after,LOW);
   digitalWrite(buz,1);
   delay(1000);
   digitalWrite(buz,0);
   Blynk.logEvent("notify","Morning Dosage :: Omeprazole-20mg... !!!!");
   delay(5000);
```

```
  digitalWrite(morn,LOW);
  digitalWrite(after,HIGH);
  digitalWrite(after,LOW);
  digitalWrite(buz,1);
  delay(1000);
  digitalWrite(buz,0);
  Blynk.logEvent("notify","Afternoon Dosage :: Atenolol-50mg ....!!!!");
  delay(5000);
  digitalWrite(morn,LOW);
  digitalWrite(after,LOW);
  digitalWrite(after,HIGH);
  digitalWrite(buz,1);
  delay(1000);
  digitalWrite(buz,0);
  Blynk.logEvent("notify","Evening Dosage :: Cetirizine-40mg.....!!!!");
  delay(5000);
}
```

CIRCUIT DIAGRAM:



ADVANTAGES:

1. Increased medication adherence: A smart medical reminder system can help patients to remember to take their medication on time and in the correct dosage. This can improve medication adherence and reduce the risk of complications associated with missed doses.

2. Real-time monitoring: IoT-enabled medical reminder systems can provide real-time monitoring of patients' medication adherence, vital signs, and other health data. This can help healthcare providers to detect and respond to health issues before they become serious.

3. Personalized reminders: Smart medical reminder systems can be personalized to individual patients' needs, including the type of medication, dosage, and timing of reminders. This can help patients to manage complex medication regimens and improve overall health outcomes.

APPLICATIONS:

1. Remote Patient Monitoring: Smart medical reminders can help doctors and caregivers monitor patients remotely, enabling them to identify potential health issues before they become more severe.

2. Elderly Care: Elderly individuals may require assistance in taking their medications on time. Smart medical reminders can assist in reminding them when it's time to take their medication and ensure they take the correct dosage.

3. Rehabilitation: Smart medical reminders can help in the rehabilitation process by reminding patients to perform their exercises at specific intervals.

CONCLUSION:

Activity:12

Date:

DESIGN THE APPLICATION FOR SMART BLIND STICK USING ESP-32

AIM:

To Design the APPLICATION FOR SMART BLIND STICK using Esp-32.

APPARATUS:

ESP32 board

IR Sensor

Blynk app

Buzzer

Breadboard

Jumper wires

5V power supply

DESCRIPTION:

A blind stick, also known as a white cane, is a mobility aid used by individuals who are visually impaired or blind. It is designed to detect obstacles in the user's path and provide tactile feedback through the use of a long, white cane that is swept back and forth in front of the user as they walk.The cane can help the user navigate their environment and avoid obstacles such as curbs, steps, and other hazards. It can also provide information about the texture of the surface being walked on, which can help the user to identify different types of surfaces, such as carpet, tile, or concrete.Blind sticks may also come equipped with additional features, such as sensors or GPS systems, to provide even more information about the user's surroundings. Overall, the blind stick is an important tool for individuals with visual impairments, allowing them to move around more safely and independently

Here are the steps to design:

1. **Here are the steps to create the blind stick using ESP32:**
2. **First, you need to install the ESP32 board in the Arduino IDE. You can follow the instructions provided by the manufacturer to do this.**
3. **Connect the ultrasonic sensor to the ESP32 board. The sensor will be used to detect obstacles in front of the user. The sensor should be connected to the 5V pin, GND pin, and two GPIO pins. One GPIO pin will be used for the trigger and the other for the echo.**
4. **Connect the vibrating motor to the ESP32 board. The motor will vibrate to alert the user of any obstacles. The motor should be connected to a GPIO pin.**
5. **Connect the buzzer to the ESP32 board. The buzzer will beep to alert the user of any obstacles. The buzzer should be connected to a GPIO pin.**

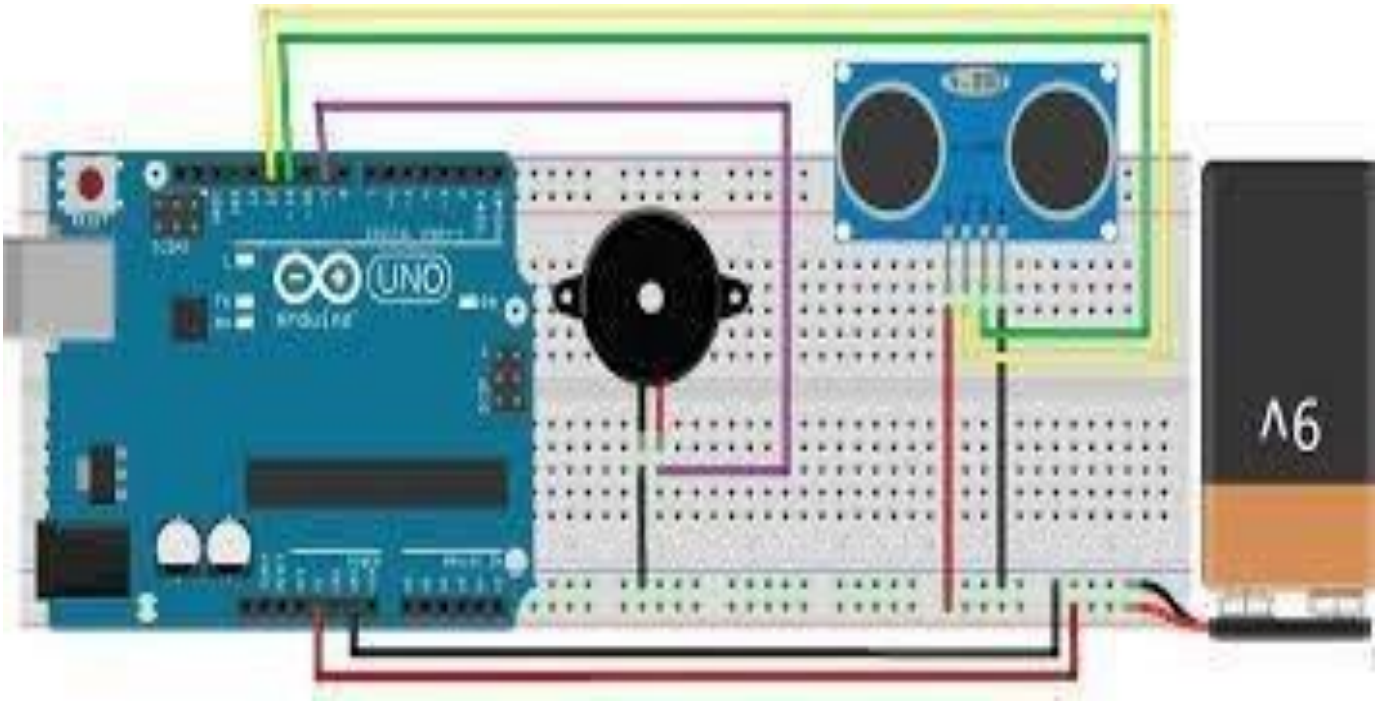
6. **Connect the Li-ion battery to the ESP32 board. The battery should be connected to the 5V pin and GND pin.**
7. **Connect the battery charger module to the ESP32 board. This will ensure that the battery is charged when needed.**
8. **Connect the LED to the ESP32 board. The LED will light up to indicate that the device is working. The LED should be connected to a GPIO pin.**
9. **Connect the switch to the ESP32 board. The switch will be used to turn on and off the device. The switch should be connected to a GPIO pin.**
10. **Once all the components are connected, you can write the code for the ESP32. The code should read the distance measured by the ultrasonic sensor, and if an obstacle is detected within a certain range, it should activate the vibrating motor and the buzzer to alert the user. The code should also turn on the LED to indicate that the device is working.**
11. **Test the device to make sure it is working properly.**

CODE:

```
#define BLYNK_TEMPLATE_ID "TMPLZrWLeWZe"
#define BLYNK_TEMPLATE_NAME "BLIND STICK"
#define BLYNK_AUTH_TOKEN "hmZ7BJZTVSqDmsqvIyi2T6IeFizuAwIh"

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

#define buz 16
#define irsensor 18
#define panic_sw 19

long duration;
float distance;

char ssid[] = "LOKESH";
char pass[] = "7569121361";
```

```
void setup()
{
  // Debug console
  Serial.begin(9600);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  pinMode(buz, OUTPUT);
  pinMode(irsensor,INPUT);
  pinMode(panic_sw,INPUT_PULLUP);
}

void loop()
{
  Blynk.run();

  if (digitalRead(irsensor) == LOW)
  {
    digitalWrite(buz, HIGH);
    while(digitalRead(irsensor) == LOW);
  }
  else
  {
    digitalWrite(buz, LOW);
  }

  if(digitalRead(panic_sw) == LOW)
  {
    Blynk.logEvent("notify","im WITH ADB");
    while(digitalRead(panic_sw) == LOW);
    }
}
```

CIRCUIT DIAGRAM:



ADVANTAGES:

1.Improved Navigation: Smart blind sticks using IoT can help blind individuals navigate more efficiently and safely by using GPS, sensors, and other technologies. These devices can provide real-time information on the surroundings, detect obstacles, and suggest alternative routes.

2.Connectivity: Smart blind sticks can be connected to other devices and networks through IoT, allowing for greater accessibility and ease of use. For example, they can be connected to smartphones or smart home devices, making it easier for blind individuals to control their environment.

APPLICATIONS:

1.Obstacle detection: Blind sticks can help users detect obstacles in their path. By feeling the texture and height of the obstacles, users can navigate around them more easily.

2.Terrain feedback: Blind sticks also provide feedback on the terrain. Users can feel changes in surface texture, such as from concrete to grass, which can help them navigate unfamiliar environments

CONCLUSION:

Activity:13

Date:

DESIGN THE IOT BASED HOME AUTOMATION USING ESP-32

AIM:

To Design the IOT BASED HOME AUTOMATION using Esp-32.

APPARATUS:

- ESP32 Board
- Relay Module
- Sensors
- Power supply
- Jumper wires
- Data cable

DESCRIPTION:

ESP32 is a powerful microcontroller that comes with built-in Wi-Fi and Bluetooth capabilities. This makes it an excellent choice for IoT-based home automation projects. You can use ESP32 to control a variety of devices in your home, including lights, fans, appliances, and more.The basic idea behind IoT-based home automation using ESP32 is to connect all the devices in your home to a central hub, which can be controlled using a smartphone app or a web interface. The ESP32 acts as the hub, and it connects to all the devices in your home using Wi-Fi or Bluetooth.To control the devices, you need to write code for the ESP32 that listens for commands from the app or web interface and sends those commands to the appropriate device. For example, if you want to turn on a light, you would send a command to the ESP32, which would then send a signal to the light switch to turn on the light.One of the key advantages of using ESP32 for home automation is its ability to communicate with other IoT devices. For example, you can use ESP32 to integrate with smart thermostats, smart locks, and other IoT devices in your home. This allows you to create a seamless home automation system that can be controlled from a single interface.In terms of the theory behind ESP32-based home automation, it is important to understand the basics of Wi-Fi and Bluetooth communication. Wi-Fi and Bluetooth are wireless communication protocols that allow devices to communicate with each other over a wireless network.ESP32 uses these protocols to connect to other devices in your home and receive commands. The ESP32 can also send data back to the app or web interface, allowing you to monitor the status of your devices and receive real-time notifications.

Overall, IoT-based home automation using ESP32 is an exciting field that has the potential to revolutionize the way we interact with our homes. With the right hardware and software, you can create a powerful home automation system that is easy to use and can save you time and energy.

Here are the steps to design:

1.Select the components: You'll need an ESP32 development board, a Wi-Fi module, a relay module, and a few other components depending on the features you want to implement in your home automation system.

2. Set up the development environment: You can use the Arduino IDE to program the ESP32. Install the ESP32 board in the Arduino IDE and download the Blynk library.

3. Create a Blynk account: Go to the Blynk website and create an account. Then create a new project and select the hardware device you'll be using (ESP32).

4. Connect the hardware: Connect the ESP32 to your Wi-Fi network, and then connect the relay module to the ESP32. You'll also need to connect the devices you want to control (e.g., lights, fans, etc.) to the relay module.

5. Program the ESP32: Use the Arduino IDE to program the ESP32. Write code to read input from the Blynk app and control the relay module. You can also add additional features like temperature sensors, motion sensors, and more.

6. Create a Blynk app: In the Blynk app, create a user interface that allows you to control the devices connected to the relay module. You can add buttons, sliders, and other widgets to the app to control the devices.

7. Test the system: Once everything is set up, test the system to make sure it's working properly. Use the Blynk app to control the devices and monitor any sensors you've added to the system.

8. Deploy the system: Once you're satisfied with the system, deploy it in your home. You can also add additional features or devices as needed

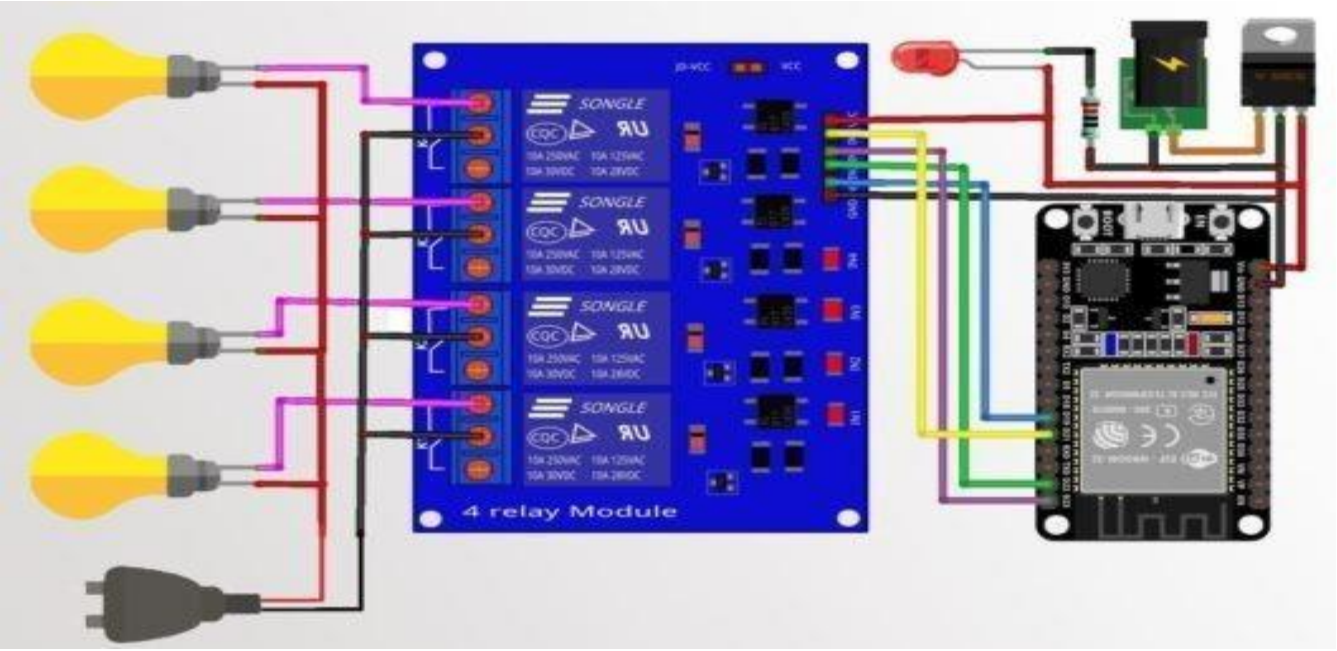
CODE:

```
#define BLYNK_TEMPLATE_ID "TMPL84iJXZ6o"
#define BLYNK_DEVICE_NAME "FireFigher Robot"
#define BLYNK_AUTH_TOKEN "d2u15efcykxoZGGAAXYGCHSto8iCJW67"

#define load1 19
#define load2 18
#define load3 17


#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
```

```
char ssid[] = "Robotics";
char pass[] = "makeindia";

int sw1, sw2, sw3;

BLYNK_WRITE(V0)
{
  sw1 = param.asInt();
  digitalWrite(load1, sw1);
}
BLYNK_WRITE(V1)
{
  sw2 = param.asInt();
  digitalWrite(load2, sw2);
}
BLYNK_WRITE(V2)
{
  sw3 = param.asInt();
  digitalWrite(load3, sw3);
}


void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  pinMode(load1, OUTPUT);
  pinMode(load2, OUTPUT);
  pinMode(load3, OUTPUT);
}
```

```
void loop()
{
  Blynk.run();
}
```

CIRCUIT DIAGRAM:



ADVANTAGES:

1.Convenience: IoT-based home automation systems make it easier to control various devices and systems in your home from a single interface, such as a smartphone app or voice command. This allows you to adjust lighting, temperature, security, and more from anywhere, making your life more convenient.

2.Energy efficiency: IoT-based home automation can help you save energy and money by optimizing the use of various systems in your home. For example, smart thermostats can automatically adjust the temperature based on your habits and preferences, reducing energy waste.

3.Safety and security: IoT-based home automation can provide you with added safety and security measures, such as smart locks, cameras, and alarms. These devices can be monitored and controlled remotely, making it easier to keep an eye on your home and protect it from intruders.

APPLICATIONS:

Here are some of the most common applications of IoT-based home automation:

1.Lighting control: IoT-based home automation can be used to control the lighting in your home automatically or remotely. This can help you save energy and make your home more secure by turning on lights when you're away.

2.Temperature control: Home automation systems can be used to control the temperature of your home remotely. You can adjust the temperature of your home from your smartphone or computer, ensuring that your home is always at the right temperature.

3.Security: Home automation systems can be used to monitor your home and alert you if there is any suspicious activity. You can also remotely control the locks on your doors and windows, ensuring that your home is always secure.

CONCLUSION:

Activity:14

Date:

DESIGN THE AGRICULTURE BASED WATER PUMP CONTROLLER USING ESP-32

AIM:

To Design the AGRICULTURE BASED WATER PUMP CONTROLLER using Esp-32.

APPARATUS:

- Esp-32 development board
- Data cable
- Jumber wires
- Power supply
- Water pump
- Water level sensor
- Relay module

SOFTWARE:

- Ardunio IDE
- Blynk IOT

DESCRIPTION:

An agricultural based water pump controller for farmers is a device that can automatically control the pumping of water in agricultural fields. It is designed to help farmers optimize their water usage and reduce wastage.The controller can be programmed to turn on and off the water pump at specific intervals, based on the specific crop and soil conditions. For instance, it may turn on the pump for a certain period in the morning and evening, when the crop needs the most water.The controller can also be configured to monitor the soil moisture levels and adjust the watering schedule accordingly. This ensures that the plants get the right amount of water without over- or under-watering.Another feature that can be included in the controller is the ability to detect faults in the pump or the water supply system. This helps farmers to identify problems early and take corrective action before it results in damage to crops or the pump. The controller can be integrated with a mobile app, allowing farmers to monitor the system remotely and receive alerts if there are any issues. It can also generate reports on water usage, pump performance, and crop health, helping farmers to make informed decisions about their farming practices.

Steps:

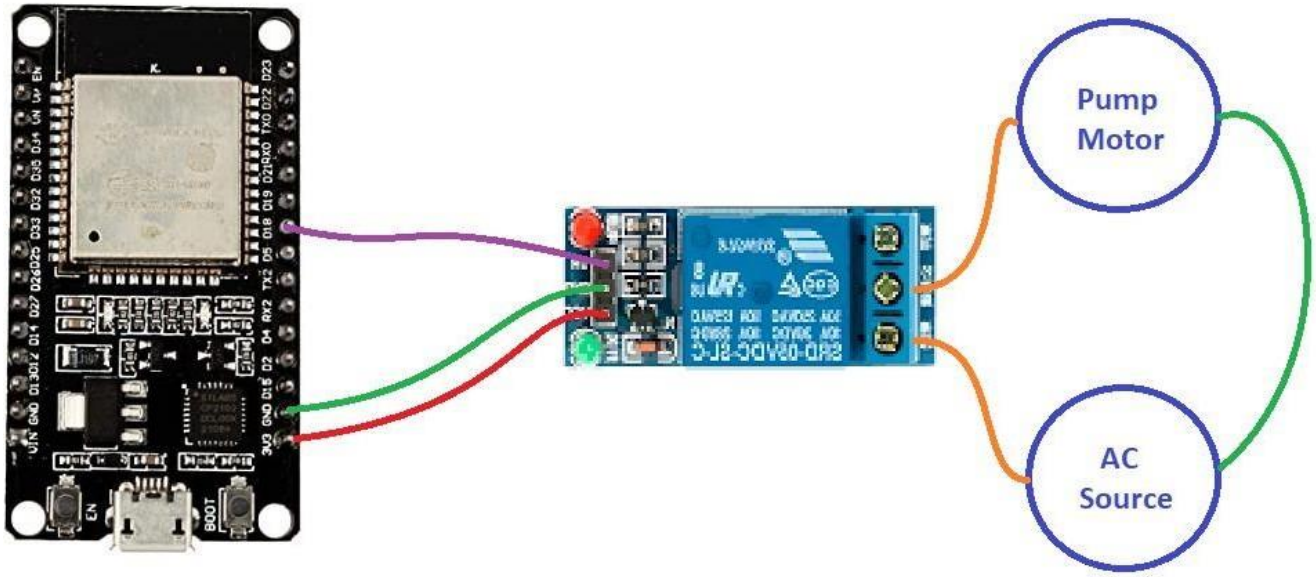
1. Set up the ESP32 Development Board by connecting it to your computer via USB cable and installing the necessary drivers.
2. Download the Blynk app on your mobile device and create an account.
3. Create a new project in the Blynk app and choose the ESP32 Development Board as the hardware model.
4. Add the necessary widgets to your Blynk app project, including buttons, sliders, and value displays, which you can use to control the water pump and monitor the soil moisture level.
5. Connect the soil moisture sensor to the ESP32 Development Board using jumper wires, ensuring that the power and ground pins are properly connected.
6. Connect the relay module to the ESP32 Development Board using jumper wires, ensuring that the power and ground pins are properly connected. Connect the water pump to the relay module.
7. Upload the Blynk code to the ESP32 Development Board using the Arduino IDE.
8. Open the Blynk app and connect to your ESP32 Development Board using your Wi-Fi network credentials.
9. Use the widgets on the Blynk app to control the water pump and monitor the soil moisture level. For example, you can use a button widget to turn on the water pump when the soil moisture level drops below a certain threshold, or you can use a slider widget to control the duration of the water pump operation.
10. Mount the components on a breadboard or PCB board and place them near your plants, ensuring that the soil moisture sensor is inserted into the soil.

CODE:

```
#define BLYNK_TEMPLATE_ID "TMPL84iJXZ6o"
#define BLYNK_DEVICE_NAME "FireFigher Robot"
#define BLYNK_AUTH_TOKEN "d2u15efcykxoZGGAAXYGCHSto8iCJW67"

#define load1 19

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Robotics";
char pass[] = "makeindia";
```

```
int sw1;

BLYNK_WRITE(V0)
{
  sw1 = param.asInt();
digitalWrite(load1, sw1);
Blynk.logEvent("notify"," motor is on");
}

void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  pinMode(load1, OUTPUT);
}

void loop()
{
  Blynk.run();
}
```

CIRCUIT DIAGRAM:



ADVANTAGES:

1. Remote monitoring: The ESP32-based water pump controllers can be remotely monitored and controlled using a smartphone or computer. This enables farmers to monitor the water pump's status and adjust the schedule or settings from anywhere, providing greater flexibility and convenience.

2. Cost-effective: The ESP32 microcontroller is relatively inexpensive, making it an affordable solution for small and medium-scale farmers. The low cost of the water pump controllers ensures that farmers can benefit from the latest technology without incurring significant expenses.

3. Improved crop yield: The ESP32-based water pump controllers ensure optimal water usage for the crops, resulting in improved crop yield and quality. The controllers can be programmed to provide the right amount of water at the right time, ensuring that the crops receive the required nutrients and moisture levels.

APPLICATIONS:

1. Irrigation System Control: An ESP32-based water pump controller can be used to automate the irrigation system in farms. Farmers can program the ESP32 to turn on/off the water pump at specific times, based on the soil moisture content or weather conditions.

2. Water Quality Monitoring: The ESP32 can be used to monitor the water quality in ponds, wells, or other water sources. It can measure parameters such as pH, temperature, and dissolved oxygen, and transmit the data to the farmers' mobile devices.

3. Remote Pump Control: Farmers can use an ESP32-based water pump controller to remotely monitor and control their water pump. They can turn the pump on/off, adjust the speed, or receive alerts if there are any issues with the pump.

CONCLUSION:

Activity:15

Date:

DESIGN THE VOICE CONTROL HOME AUTOMATION USING BLUETOOTH USING ESP-32

AIM:

To Design the VOICE CONTROL HOME AUTOMATION USING BLUETOOTH

using Esp-32.

APPARATUS:

- ESP32 board0
- Blynk app
- Jumber wires
- Data cable
- Microphone
- Led

DESCRIPTION:

Voice Recognition System: The voice recognition system is a crucial component of the voice control home automation system. It is responsible for converting the user's voice commands into digital signals that can be processed by the system. The voice recognition system can be implemented using various methods such as machine learning algorithms or speech recognition software.

Bluetooth Module: The Bluetooth module is responsible for establishing a wireless connection between the user's device and the home automation system. The module can be integrated into the home automation system or be a separate component that can be plugged into the system's USB port.

Home Automation System: The home automation system consists of various devices such as sensors, controllers, and actuators that are used to control the home's appliances, lighting, and temperature. The system can be programmed to respond to specific voice commands given by the user.

User Device: The user device is used to communicate with the home automation system through the Bluetooth module. The device can be a smartphone, tablet, or any other Bluetooth-enabled device.

Voice Commands: The user can give voice commands to the home automation system using the voice recognition system. The system can be programmed to respond to specific voice commands such as "turn off the lights," "increase the temperature," or "lock the doors."

System Response: Once the voice command is received, the home automation system responds accordingly. For example, if the user says "turn off the lights," the system will send a signal to the lighting controller to turn off the lights.

Security: Security is an essential aspect of voice control home automation using Bluetooth. The system must be secured against unauthorized access to prevent hackers from gaining control of the home automation system. Encryption and authentication mechanisms can be implemented to ensure the security of the system.

Here are the steps to follow:

Here are the steps to design a voice-controlled home automation system using Bluetooth and the ESP32 board with Blynk:

1. Hardware setup:
2. Connect the ESP32 board to your computer using a USB cable and open the Arduino IDE.
3. Install the Blynk library and the BluetoothSerial library in the Arduino IDE.
4. Connect the relays to the ESP32 board, which will control the home appliances such as lights, fans, and other electronic devices.
5. Power up the ESP32 board with a 5V power supply.
6. Create a Blynk project:
7. Download the Blynk app on your smartphone and create a new project.
8. Select the ESP32 board as your hardware device and choose the Bluetooth connection option.
9. Add the necessary widgets such as buttons, sliders, and labels to control the home appliances in the Blynk project.
10. Code the ESP32 board:
11. In the Arduino IDE, create a new sketch and copy the BluetoothSerial and Blynk libraries.
12. Create variables for the BluetoothSerial and Blynk objects.
13. Initialize the BluetoothSerial and Blynk objects.
14. Create variables for the relays and set their pin numbers.
15. Create a function to control the relays based on the received Bluetooth command from the Blynk app.
16. Create a function to receive the voice commands using the BluetoothSerial library.
17. Map the voice commands to the corresponding relay control functions.
18. Upload the code to the ESP32 board.
19. Test the system:
20. Open the Blynk app and connect to the ESP32 board via Bluetooth.
21. Use the voice commands to control the home appliances through the Blynk app and see if the system is working correctly.
22. You can also use the Blynk app to manually control the home appliances.

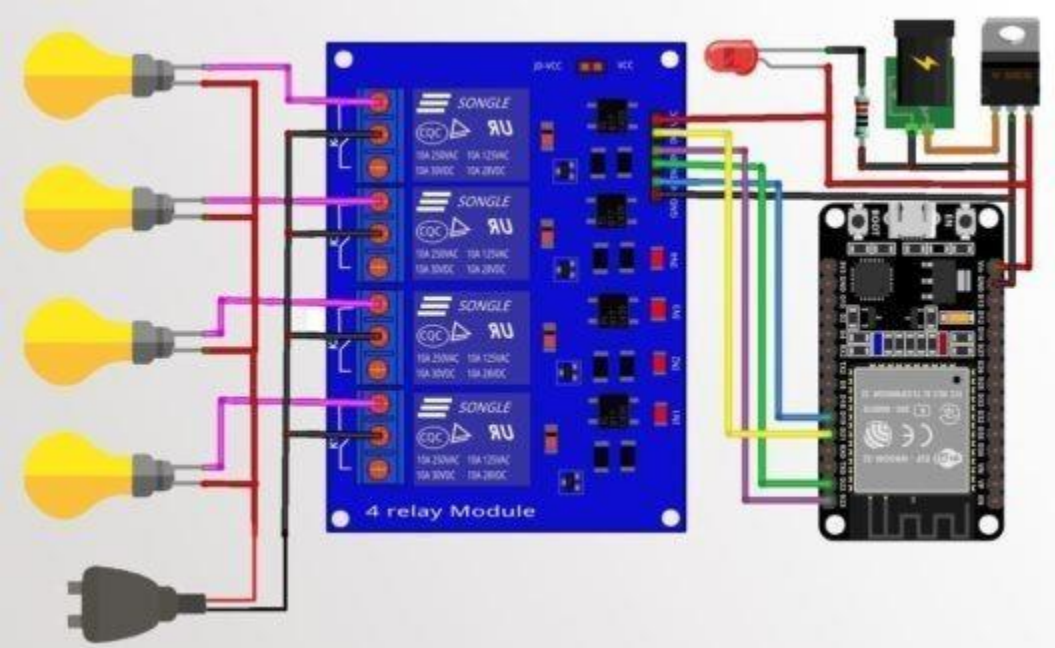
CODE:

```
#define load1 19

#define load2 18


#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
```

```
#error Bluetooth is not eneble!
#endif

BluetoothSerial mySerial;
String voice;  //to store the command
char c;       //to get characters from the command

void setup() {

  // put your setup code here, to run once:
  Serial.begin(9600);
  mySerial.begin("Vishishta");
  pinMode(load1, OUTPUT);
  pinMode(load2, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  while (mySerial.available())
  {
    delay(10);    //delay added to make it stable
    c = mySerial.read();  //get new command
    if ((c >= 65) && (c <= 90))
    {
      c += 32;
    }
    voice += c;  //to make a string of command
  }
  if (voice.length() > 0)
  {
```

```
  Serial.println(voice);  //print the command in Serial Terminal


  if (voice == "lighton")
  {
    Serial.println("your is light on");
    digitalWrite(load1, 1);
    voice = "";
  }
  else if (voice == "lightoff")
  {
    Serial.println("your is light off");
    digitalWrite(load1, 0);
    voice = "";
  }
  else if (voice == "fanon")
  {
    Serial.println("your is fan off");
    digitalWrite(load2, 1);
    voice = "";
  }
  else if (voice == "fanoff")
  {
    Serial.println("your is fan off");
    digitalWrite(load2, 0);
    voice = "";
  }
  voice = "";
 }
}
```

CIRCUIT DIAGRAM:



ADVANTAGES:

1. Hands-free operation: With voice control, you can operate your home automation system without the need for physical buttons or touchscreens. This can be especially useful for people with limited mobility or those who want to operate their devices from a distance.

2. Easy to set up: Bluetooth is a widely available and easy-to-use technology, and the ESP32 is a popular and versatile microcontroller that can be used for a variety of projects. This makes it relatively easy to set up a voice-controlled home automation system using Bluetooth and ESP32.

3. Customizability: With the ESP32, you can customize your home automation system to suit your specific needs and preferences. For example, you can create custom voice commands to control specific devices or create macros that perform multiple actions at once.

APPLICATIONS:

1. Home Entertainment System: You can also use voice control home automation to control your home entertainment system. With voice commands, you can play music, switch channels on your TV, and control other audio and video devices in your home.

2. Smart Home Appliances: With Bluetooth and ESP32 technology, you can control smart home appliances using your voice, such as coffee makers, refrigerators, and ovens. You can also set the temperature, cooking time, and other settings by using voice commands.

3. Voice-Activated Door Lock: You can also use voice-controlled home automation to lock or unlock your home's doors. This technology makes it easier to open or close the doors without touching them, making it more convenient and secure.

CONCLUSION:

Activity:16

Date:

DESIGN THE VOICE TO TEXT CONVERSION AND TEXT TO VOICE CONVERSION USING ESP-32

AIM:

To Design the VOICE TO TEXT CONVERSION AND TEXT TO VOICE CONVERSION using Esp-32.

APPARATUS:

ESP32 board

Blynk app

Jumber wires

Data cable

Microphone

Speaker

DESCRIPTION:

Voice to Text Conversion:

First, we need to record the voice input from the user. To do this, we can use an external microphone or a microphone module that can be connected to the ESP32. Next, we need to send the recorded audio file to a speech-to-text API for processing. There are several speech-to-text APIs available, such as Google Cloud Speech-to-Text, IBM Watson Speech-to-Text, and Amazon Transcribe. You will need to choose the one that best fits your project requirements. Once the audio file is sent to the API, it will convert the voice input into text. The converted text can be retrieved from the API and stored in a variable. Finally, we can output the converted text to a display or send it to a remote server for further processing.

Text to Voice Conversion:

First, we need to input the text that we want to convert into voice. This can be done through a text input module connected to the ESP32. Next, we need to send the text to a text-to-speech API for processing. There are several text-to-speech APIs available, such as Google Cloud Text-to-Speech, IBM Watson Text-toSpeech, and Amazon Polly. You will need to choose the one that best fits your project requirements. Once the text is sent to the API, it will convert the text into a voice audio file. The audio file can be retrieved from the API and stored in a variable.

To design voice to text conversion and text to voice conversion using ESP32 by Blynk, Here are the steps to follow:

- Setting up the Blynk app:

- Download the Blynk app from the app store or play store.
- Create a new project and select the ESP32 board.
- Add a button widget to control the voice to text conversion process.
- Add a display widget to show the converted text.
- Add a terminal widget to display the debugging messages.
- Setting up the ESP32:
- Install the ESP32 board and library in the Arduino IDE.
- Connect the microphone to the ESP32 board. The microphone's output pin should be connected to one of the ADC pins of the ESP32 board.
- Connect the speaker to the ESP32 board. The speaker should be connected to one of the PWM pins of the ESP32 board.
- Writing the code:
- Include the Blynk and ESP32 libraries in the Arduino IDE.
- Initialize the Blynk library with the auth token and Wi-Fi credentials.
- In the setup function, initialize the ADC and PWM pins of the ESP32 board.
- Write a function to convert voice to text using the Google Speech API.
- Write a function to convert text to voice using the Google Text-to-Speech API.
- In the loop function, read the microphone input and convert it to text using the voice to text function. Display the converted text on the display widget.
- When the button widget is pressed, convert the text to voice using the text to voice function and play it on the speaker.
- Here is a sample code to get you started

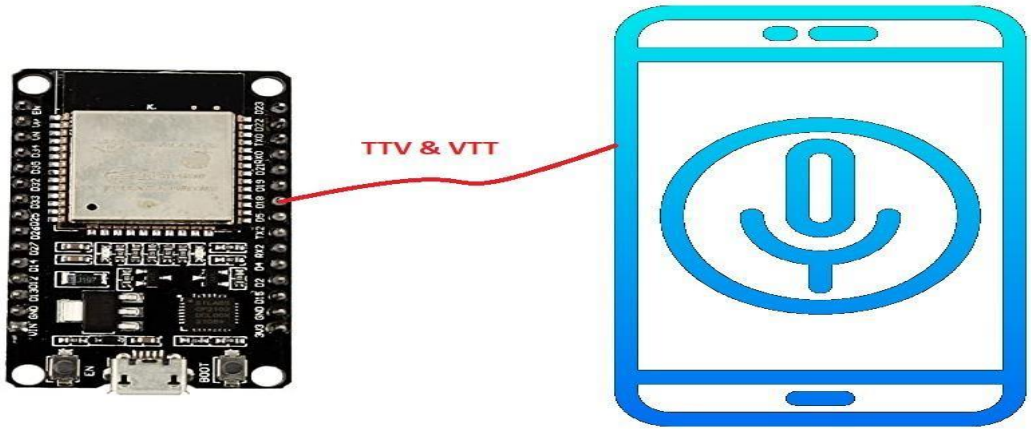
CODE:

```
#include<Arduino.h>


#include "BluetoothSerial.h"
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not eneble!
#endif


BluetoothSerial tts;
String voice;
char c;
```

```
void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600);
  tts.begin("Chintu..!");
}

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

  while (tts.available())
  {
    delay(10);    //delay added to make it stable
    c = tts.read();  //get new command
    if ((c >= 65) && (c <= 90) )
    {
      c += 32;
    }
    voice += c;  //to make a string of command
  }
  if (voice.length() > 0)
  {
    Serial.println(voice);

    if (voice == "hello esp how are you")  //1
    {
      delay(600);
      tts.println("I am fine , may I know who are you");
```

```
    }
    else if (voice == "introduce yourself") //2
    {
      tts.println("Hello world ,im esp32");
      delay(100);
     tts.println("Iam made up of Ten silica extensa LX 6 processor with a 512KB of RAM and 4MB of ROM.");
      delay(100);
    }
    else
    {
      delay(600);
      tts.println("i cant get you");
      delay(600);
      voice = "";
    }
    voice = "";
  }
}
```

CIRCUIT DIAGRAM:



ADVANTAGES:

Advantages of Voice to Text Conversion:

1. Improved Accessibility: Voice to text conversion makes it easier for individuals with disabilities to interact with technology. This feature allows individuals with limited mobility, visual or speech impairments to use technology with ease.

2. Time-Saving: Voice to text conversion can be a time-saving feature as it eliminates the need to type or write. It can also help to reduce transcription errors and enable users to quickly and accurately capture their thoughts.

3. Multilingual Support: Voice to text conversion can be useful in multilingual settings as it can recognize and transcribe different languages.

Advantages of Text to Voice Conversion:

1. Multilingual Support: Text to voice conversion can be useful in multilingual settings as it can read out text in different languages.

2. Personalization: Text to voice conversion can be personalized to match the user's preferences regarding voice type, speed, and tone. This feature can make the listening experience more enjoyable for users.

3. Cost-Effective: Using an ESP32 microcontroller for text to voice conversion can be cost-effective compared to using cloud-based text to voice services.

APPLICATIONS:

Applications of Voice to Text Conversion:

1. Transcription: Voice to text conversion is commonly used in transcription services, where spoken words are transcribed into text for record-keeping or research purposes.

2. Accessibility: Voice to text conversion can be used to make digital content more accessible for people with hearing impairments, as well as those who prefer to read text rather than listen to audio.

3. Dictation: Voice to text conversion can be used to create written documents, such as reports or emails, by dictating the content rather than typing it out.

Applications of Text to Voice Conversion:

1. Accessibility: Text to voice conversion is commonly used to make digital content more accessible for people with visual impairments, as well as those who prefer to listen to audio rather than read text.

2. Language learning: Text to voice conversion is used in language learning software to help users learn the correct pronunciation of words and phrases.

CONCLUSION:

Activity:17

Date:

Smoke Detection using MQ-2 GasSensor using ardunio uno

Aim: To detect the Smoke

Components Required:

Arduino Uno

Bread Board

MQ-2 Smoke Detection SensorJumper Wires

Jumper Wires

Led-Red, Grenn

Buzzer

Resistor-221 Ohms

Working Principle:

In this example, you will read the sensor analog output voltage and when the smokereaches a certain level, it will make sound a buzzer and a red LED will turn on.

When the output voltage is below that level, a green LED will be on.

MQ-2Smoke Sensor:

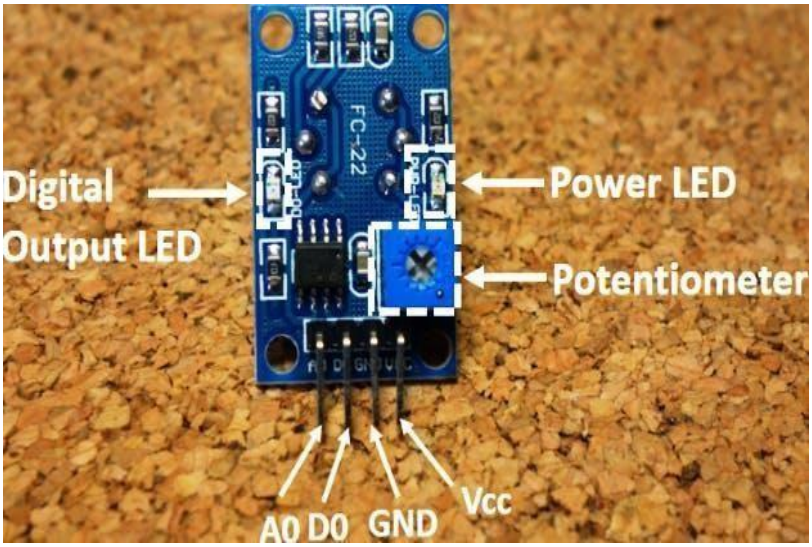
The MQ-2 smoke sensor is sensitive to smoke and to the following flammable gases:

- LPG
- Butane
- Propane
- Methane
- Alcohol
- Hydrogen

The resistance of the sensor is different depending on the type of the gas.

The smoke sensor has a built-in potentiometer that allows you to adjust the sensor sensitivity according to how accurate you want to detect gas.

MQ-2 sensor

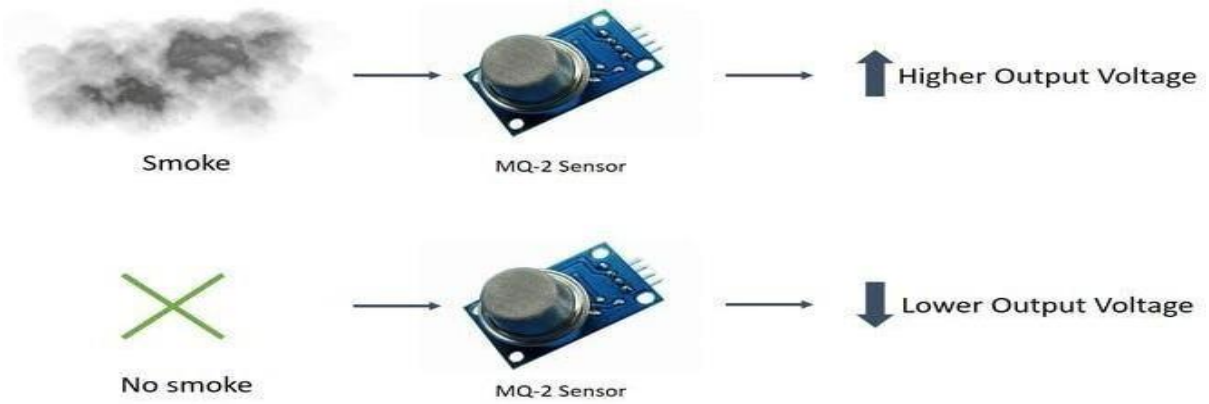


How does it Work?

The voltage that the sensor outputs changes accordingly to the smoke/gas level that exists in the atmosphere. The sensor outputs a voltage that is proportional to the concentration of smoke/gas.

In other words, the relationship between voltage and gas concentration is the following:

- The greater the gas concentration, the greater the output voltage.
- The lower the gas concentration, the lower the output voltage.



Working Mechanism

The output can be an analog signal (A0) that can be read with an analog input of the Arduino or a digital output (D0) that can be read with a digital input of the Arduino.

Pin Wiring:

The MQ-2 sensor has 4 pins.

Pin ---------- Wiring to Arduino Uno

A0..... Analog pins

D0..... Digital pins

GND..... GND

VCC..... V

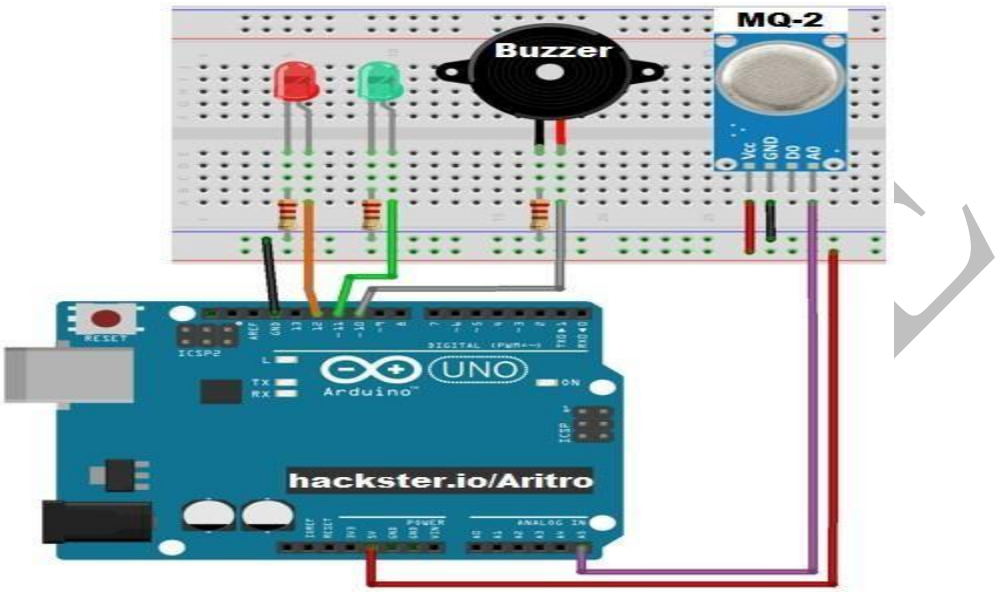
So, before jumping into the coding part, let's check whether we've assembled all t

Code:

```
int redLed =  12;
int greenLed = 11;
int  buzzer  = 10;
int smokeA0 = A5;
// Your threshold value
intsensorThres = 400;
```

```
void setup() {
pinMode(redLed, OUTPUT);
pinMode(greenLed, OUTPUT);
pinMode(buzzer,  OUTPUT);
pinMode(smokeA0, INPUT);
Serial.begin(9600);
      }
void loop() {
int analogSensor = analogRead(smokeA0);
Serial.print("Pin A0: ");
Serial.println(analogSensor);
if (analogSensor > sensorThres)
{
  digitalWrite(redLed, HIGH);
  digitalWrite(greenLed, LOW); tone(buzzer, 1000, 200);
}
else
{
  digitalWrite(redLed, LOW);
  digitalWrite(greenLed, HIGH);
  noTone(buzzer);
}
delay(100);
}
```

CIRCUIT DIAGRAM:



CONCLUSION:

Activity-18:

Date:

INTERFACE ULTRASONIC SENSOR WITH ARDUINOUNO

Aim: To interface ultrasonic sensor with Arduino Uno.

Requirements:

Arduino Uno

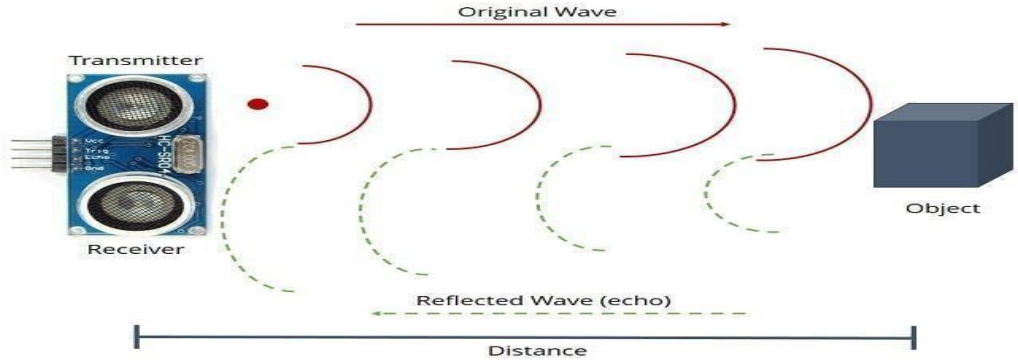
Ultrasound Sensor

Jumper Wires

Description:

Ultrasonic sensor : This sensor measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than audible sound (ultrasonic is the sound that humans cannot hear). Ultrasonic sensing is one of the best ways to sense proximityof obstacle and detect levels of substance or liquid with high reliability. An ultrasonic sensor module uses a transducer to send and receive ultrasonic pulses.The working principle of this module is simple. It sends an ultrasonic pulse out of trigger pin at 40kHz which travels through the air and if there is an obstacle orobject, it will bounce back to the sensor at echo pin. By calculating the time travelled by wave and the speed of sound, the distance of the object is calculated.

Ultrasonic sensor module has four pins namely Gnd, Vcc, Echo and Trigger. Gnd is considered as the negative pin and it is connected to the ground of the system. Vcc powers the sensor. It typically requires 3.3V. Trig (Trigger) pin is used to trigger the ultrasonic sound pulses. Echo pin produces a pulse when the reflected signal is received.



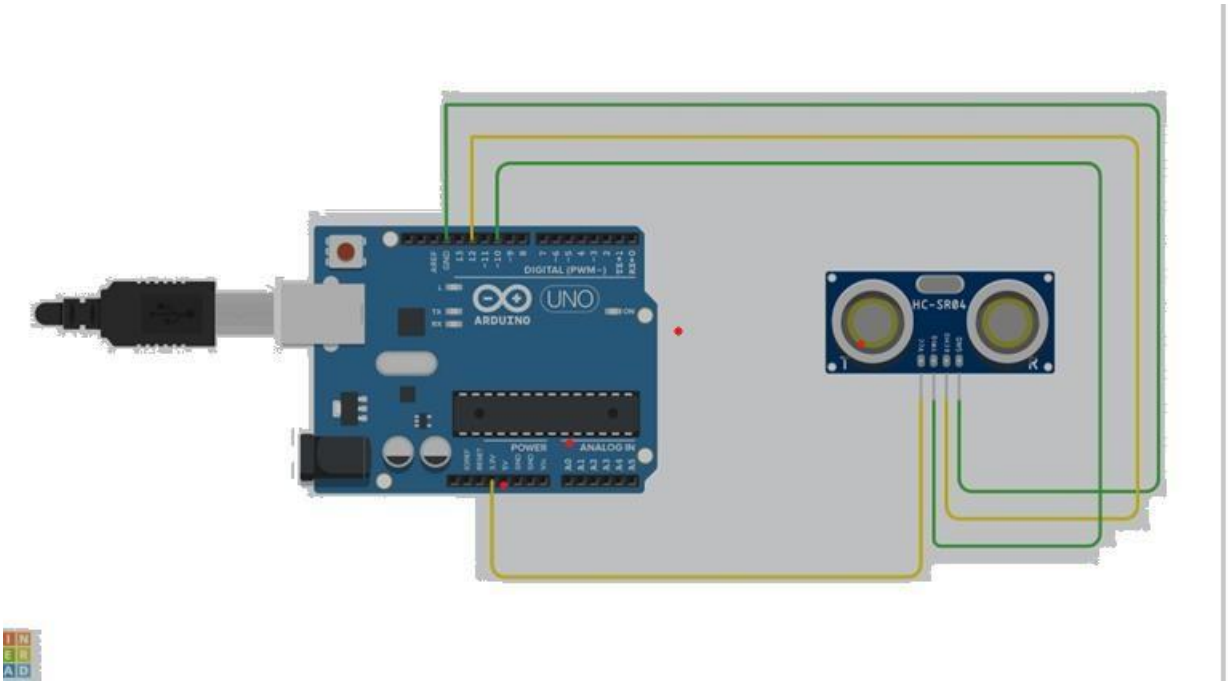
Arduino Uno is the brain of this system. It is microcontroller board based on the microcontroller ATmega328P. Arduino is capable of reading inputs, processing them and generating outputs. It has 14 digital input/output pins(of which 6 can be used as PWM outputs), 6 analog inputs, a USB connection, a power jack, an ICSP header and a reset button. Now as we know about ultrasonic sensor we can start building up our circuit. Connections are as follows : Gnd to Gnd of Arduino Echo to D12 of Arduino Trig to D10 of Arduino Vcc to 3.3v of Arduino.

Code:

```
// Interfacing Ultrasonic sensor with Arduino uno
 #define echoPin 12          //connect echo pin of ultrasonic sensor to D12 of Ardui no
#define trigPin 10          //connect trigger pin of ultrasonic sensor to D10 of Ar duino
 long duration;              // declare variables to hold duration and distance
int distance;
 void setup()               //setup() is used for initialization
 {
Serial.begin(9600);          //set the baud rate of serial communication to 9600
pinMode(trigPin,OUTPUT);        //set trigPin as output pin of Arduino
 pinMode(echoPin,INPUT);        //set echoPin as output pin of Arduino
}
void loop()
{
digitalWrite(trigPin,LOW);           //generate square wave at trigger pin
delayMicroseconds(2);
digitalWrite(trigPin,HIGH);
  delayMicroseconds(10);
digitalWrite(trigPin,LOW);
duration=pulseIn(echoPin,HIGH);       //calculation of distance of obstacle
distance=(duration*0.034/2);
  Serial.print("Distance : ");
Serial.print(distance);
 Serial.println(" cm ");
delay(1000);
}
```

After uploading the code check the right corner of the software you will find a magnifying glass icon and. Click on that option and you will get the values of ultrasonic sensor. Eg : Distance : 3 cm that means the object is 3 cm away from the sensor. This is called a serial monitor mainly used to display the values of sensors.

CIRCUIT DIAGRAM:



CONCLUSION: