EXPERIMENT NUMBER : 1
DATE : 25/03/2022 , FRIDAY

GPIO AND ADC PROGRAMMING USING ARDUINO

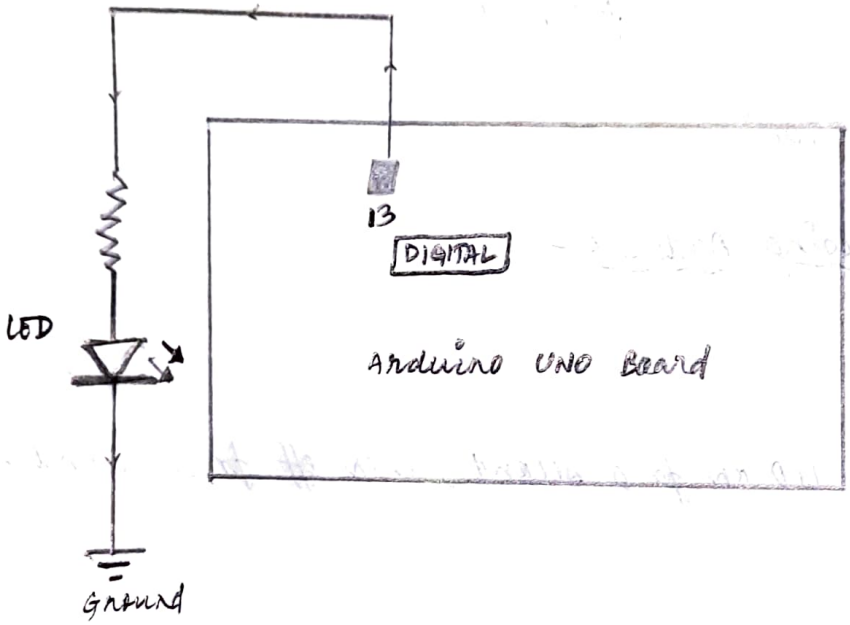
* AIM:
Understand both analog and digital sensor interfacing with a programmable hardware platform.

* CODE: (ARDUINO IDE)

(a) LED Blinking Using Arduino -

```
/*
  LED Blink:
  Turns on an LED on for a second, then off for a second,
  repeatedly.
*/

// The setup function runs once when you press reset or power the
board:
void setup()
{
  pinMode(13, OUTPUT); // Initialize digital pin 13 as an output
}

// The loop function runs over and over again forever:
void loop()
{
  digitalWrite(13, HIGH); // Turn the LED on (HIGH is the voltage
                          level)
  delay(1000); // Wait for a second
  digitalWrite(13, LOW); // Turn the LED off (LOW)
  delay(1000); // Wait for a second
}
```

→ PTO



(a) LED Blinking using Arduino

Components used:

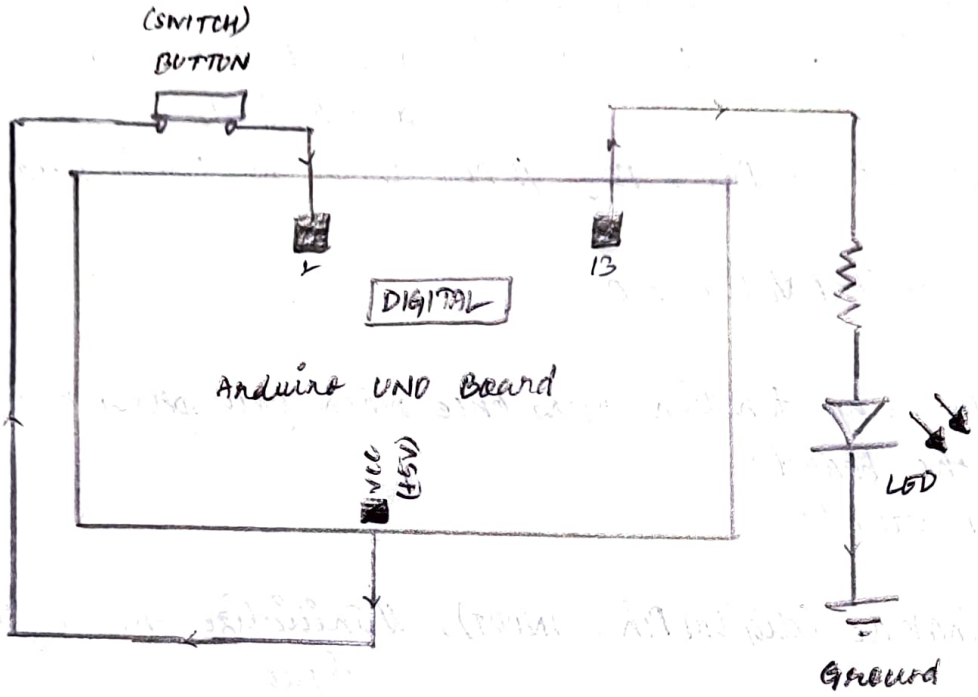
1. Arduino UNO Board
2. Yellow - LED
3. Resistor
4. Connecting wires
5. Power Supply Cable / Battery

? Built In

(b) Controlling LED With Switch -

```
/*
  Switch program to control LED:
*/

// This constant won't change:
const int digitalPin = 2; // Pin that the digital pin is
                             attached to
const int ledPin = 13;  // Pin that the LED is attached to

int digitalValue = 0

// The setup function runs once when you press reset or power
   the board:
void setup()
{
  pinMode(digitalPin, INPUT); // Initialize the digital pin as an
                                 input
  pinMode(  ledPin, OUTPUT); // Initialize the LED pin as an
                                output
}


// The loop function runs over and over again forever:
void loop()
{
  digitalValue = digitalRead(digitalPin); // Read the value

  // If the digital value is equal to 2, turn on the LED:
  if (digitalValue == HIGH)
  {
    digitalWrite(ledPin, HIGH);
  }
```

→ P.T.O.



(6) Controlling LED with Switch

Components Used:

1. Arduino UNO Board
2. Yellow LED
3. Resistor
4. Push Button
5. Connecting wires
6. Power supply cable / Battery

Built in

```
    else
    {
      digitalWrite (ledPin, LOW);
    }

}
```

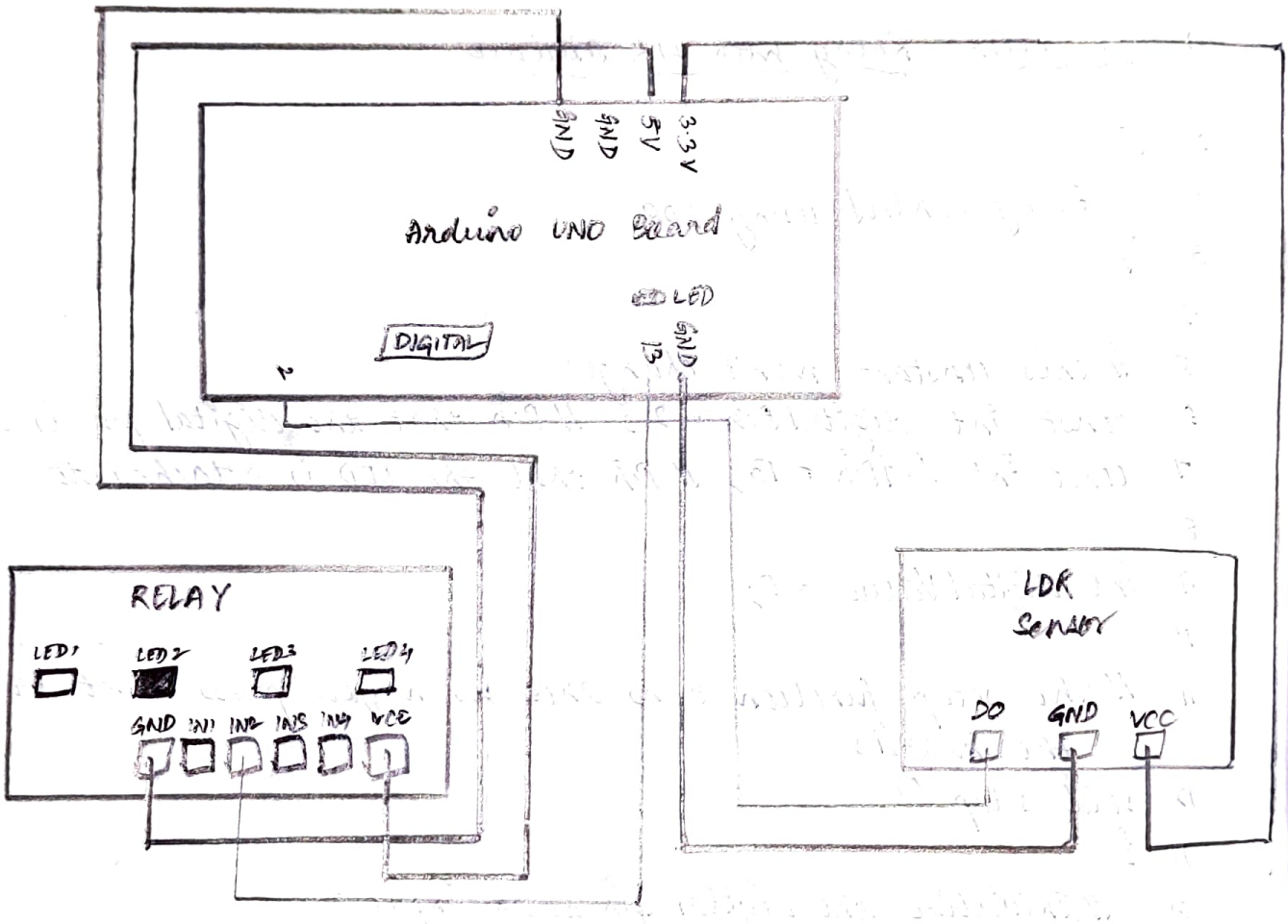
(c) Controlling Relay with LDR Module

```
/*
  Relay Control using LDR
*/

// This constant won't change:
const int digitalPin = 2; // Pin that the digital pin is attached to
const int ledPin = 13; // Pin that the LED is attached to

int digitalValue = 0;

// The setup function runs once when you press reset or power
   the board:
void setup ()
{
  // Initialize the digital pin as an input:
  pinMode (digitalPin, INPUT);
  // Initialize the LED pin as an output:
  pinMode (ledPin, OUTPUT);
}

// The loop function runs over and over again forever:
void loop ()
```

PTO

Components Used :

1. Arduino UNO Board
2. Relay
3. LDR Sensor
4. Connecting Wires
5. Power Supply cable / Battery



(c) Controlling Relay with LDR Module

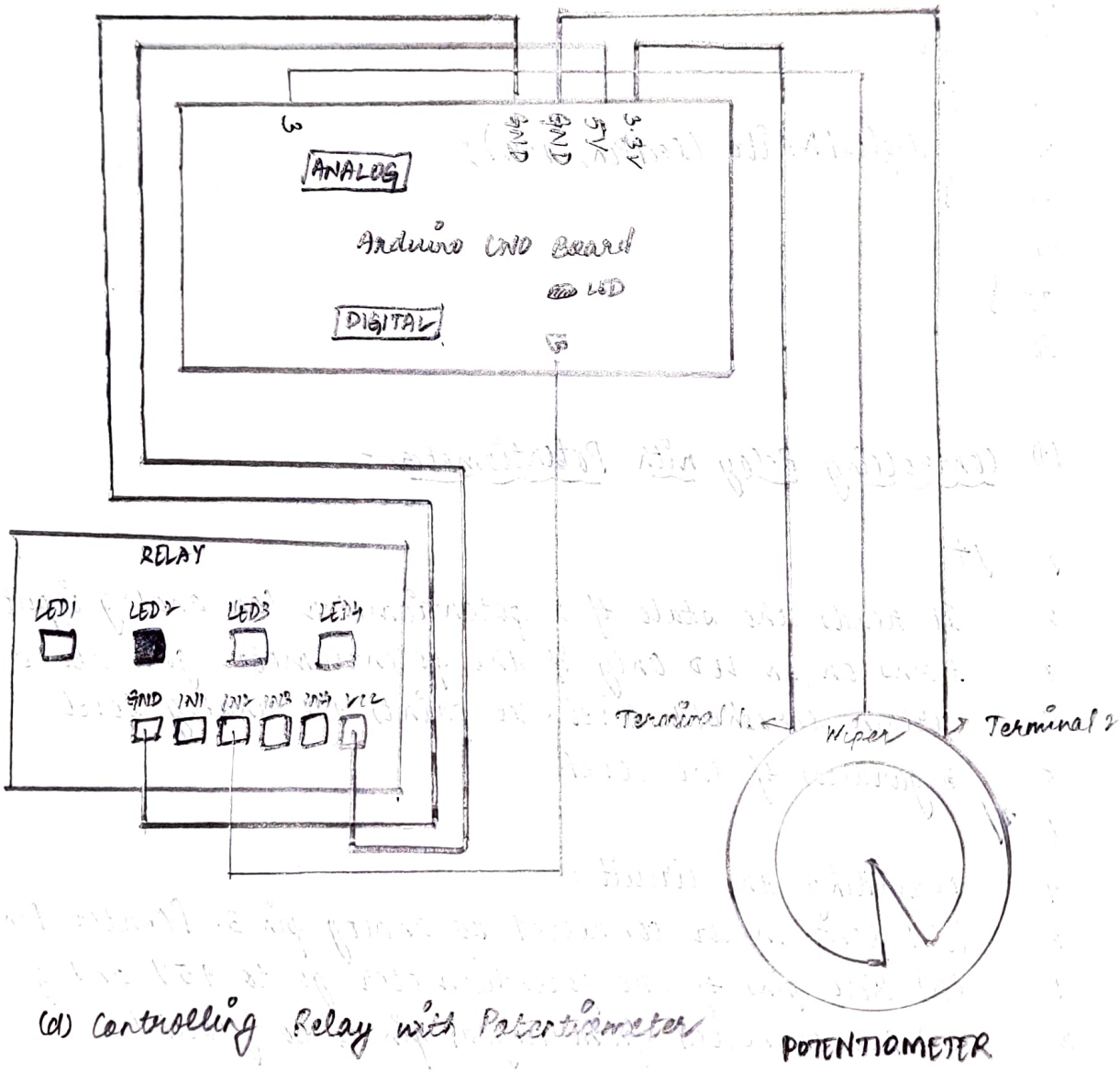
```
{
  digitalValue = digitalRead(digitalPin); //Read the value

  // If the digital value is equal to 2, turn on the LED:
  if (digitalValue == HIGH)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }

}
```

(d) Controlling Relay with Potentiometer -

```
/*
  It reads the state of a potentiometer (an analog input) and
  turns on an LED only if the potentiometer goes above a
  certain threshold level. It prints the analog level
  regardless of the level.

  Regarding the circuit:
  (i) Potentiometer connected to analog pin 3. [Center Pin]
  (ii) Side pins of the potentiometer go to +5V and ground.
  (iii) LED connected from digital pin 13 to ground.

  Note: On most Arduino boards, there is already an LED on
  the board connected to pin 13, so you don't need any extra
  components for this example.
*/
```

Components Required:

1. Arduino UNO Board
2. Relay
3. Potentiometer
4. Connecting Wires
5. Power Supply cable / Battery



(a) Controlling Relay with Potentiometer

```
16
17  // These constants won't change:
18  const int analogPin = A3; // Pin that the sensor is attached to
19  const int ledPin = 13; // Pin that the LED is attached to
20  const int threshold = 400; // An arbitrary threshold level that
    is in the range of the analog input
21
22  void setup()
23  {
24    pinMode(ledPin, OUTPUT); // Initialize the LED pin as an output
25    Serial.begin(9600); // Initialize serial communications
26  }
29
28  void loop()
29  {
30    int analogValue = analogRead(analogPin); // Read the value
      of the potentiometer
31
32    // If the analog value is high enough, turn on the LED:
33    if (analogValue > threshold)
34    {
35      digitalWrite(ledPin, HIGH);
36    }
37    else
38    {
39      digitalWrite(ledPin, LOW);
40    }
41
42    Serial.println(analogValue); // print the analog value
43    delay(1); // Delay in between reads for stability
44  }
45
```

PTO

* INFERENCE:

Analyzed both analog and digital sensor interfacing techniques with a programmable hardware platform and all simulation results were verified successfully.

EXPERIMENT NUMBER : 7

DATE : 26/03/2022 , SATURDAY

SERIAL COMMUNICATION PROTOCOLS (UART & SPI) USING ARDUINO ...

* AIM :

Analyze various communication protocols used in the design of portable devices.

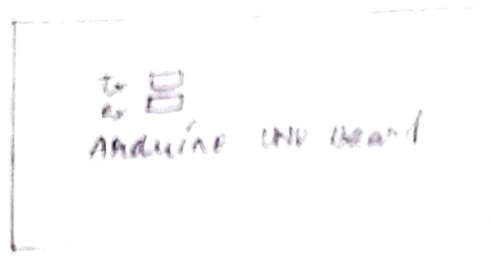
* CODE : (ARDUINO IDE)

(a) Serial Transmission using UART -

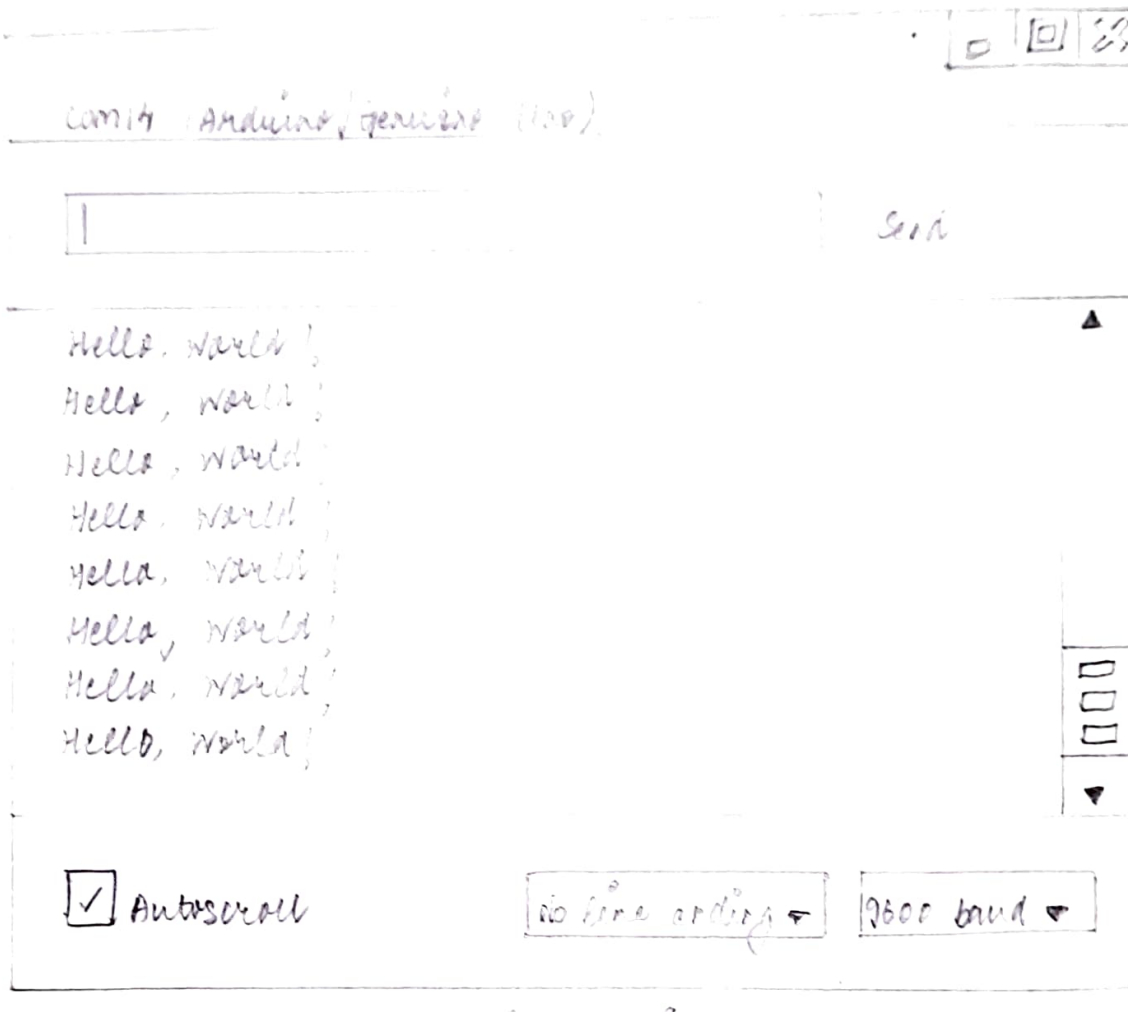
```
// The setup function runs once when you press reset or power
   the board:
void setup()
{
   Serial.begin(9600); // Initialize serial communications
}

// The loop function runs over and over again forever:
void loop()
{
   Serial.println("Hello, World!");
   delay(1); // Delay in between reads for stability
}
```

(b) Serial Reception and Transmission using UART -

```
char inputchar = ' '; // A character variable to hold
                      incoming data
```



Hardware



Output of Serial Monitor

Components Required:

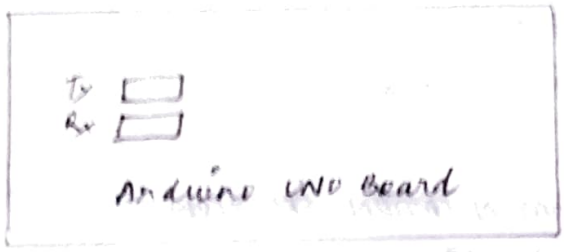
1. Arduino UNO Board
2. Battery / Power Supply Cable
3. connecting Wires

(a) Serial Transmission using UART

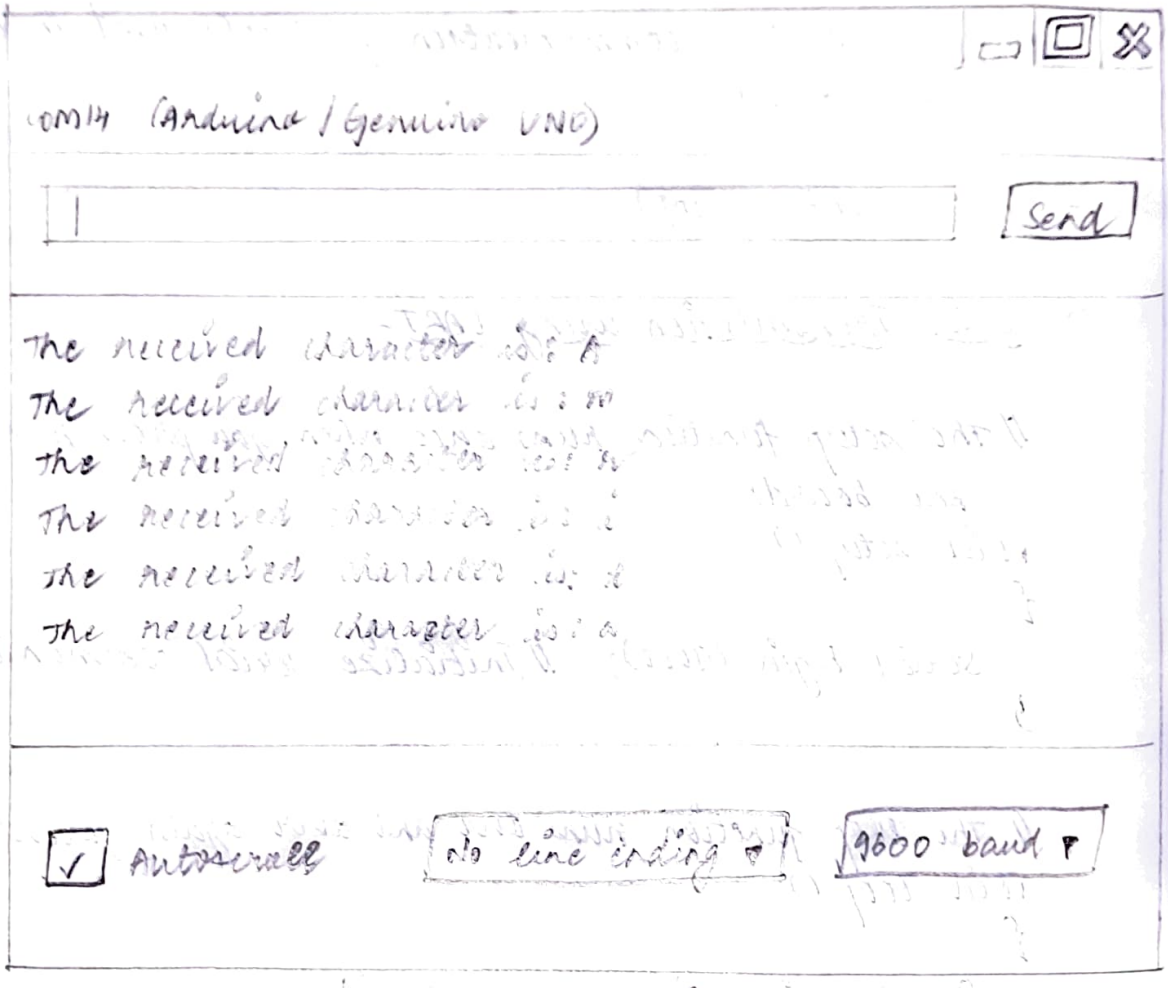
```
// The setup function runs once when you press reset or power
   the board:
void setup ()
{
  Serial. begin (9600);  // Initialize serial communications
}

// The loop function runs over and over again forever:
void loop ()
{
  if ((( Serial. available () ) > (0) ))
  {
    inputChar = Serial. read ();
    Serial. print ("The received character is: ");
    Serial. println (inputChar);
  }
}
```

(c) Device control using UART in Arduino -

```
// The setup function runs once when you press reset or power
   the board:
void setup ()
{
  Serial. begin (9600); // Initialize serial communications
  pinMode (13, OUTPUT); // Initialize digital pin 13 as an output
}

// The loop function runs over and over again forever:
void loop ()
{
  if ((( Serial. available () ) > (0) ))
  {
```



Hardware



Output of Serial Monitor

(c)

Components Required -

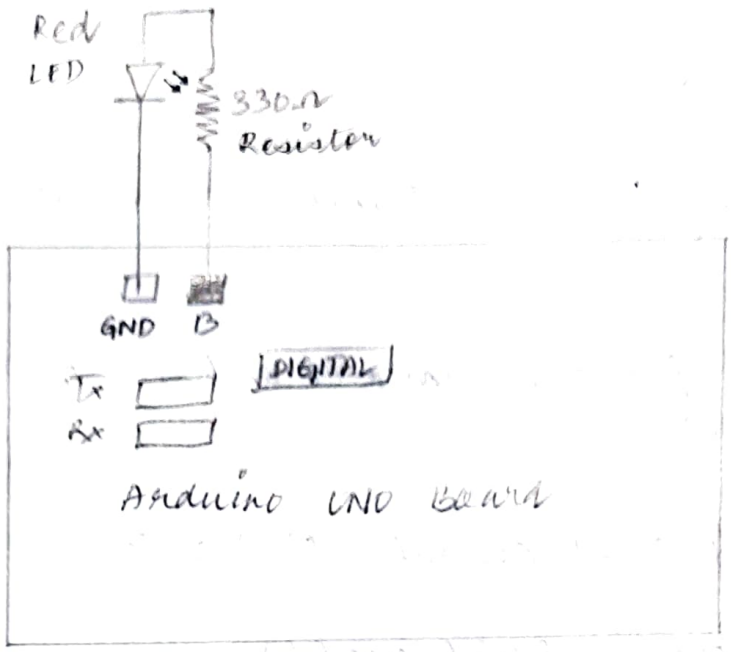
1. Arduino UNO Board
2. Battery / Power supply cable
3. Connecting wires

(b) Serial Reception and Transmission using USART

```
char inputChar = Serial.read();
Serial.print("The received character is: ");
Serial.println(inputChar);
if (inputChar == 'A')
{
  digitalWrite(13, HIGH); // Turn the LED on by making the
  delay(2000);                    voltage HIGH
  // Wait for two seconds
}
 else
 {
   digitalWrite(13, LOW); // Turn the LED off by making the
   delay(2000);                    voltage LOW
   // Wait for two seconds
 }
}
}
```

(d) Device control with Bluetooth Module using UART -

```
// These constants won't change:
const int ledPin 1 = 13; // Pin that the LED is attached to
const int ledPin 2 = 2; // Pin that the LED is attached to

// The setup function runs once when you press reset or power
   the board:
void setup()
{
  Serial.begin(9600); // Initialize serial communications
  pinMode(ledPin 1, OUTPUT); // Initialize the LED pin 1 as
                                an output
  pinMode(ledPin 2, OUTPUT); // Initialize the LED pin 2 as
                                an output
}
```



Hardware

Components Required -

1. Arduino UNO Board
2. Red LED
3. 330 Ω Resistor
4. Battery Power Supply
5. Connecting wires

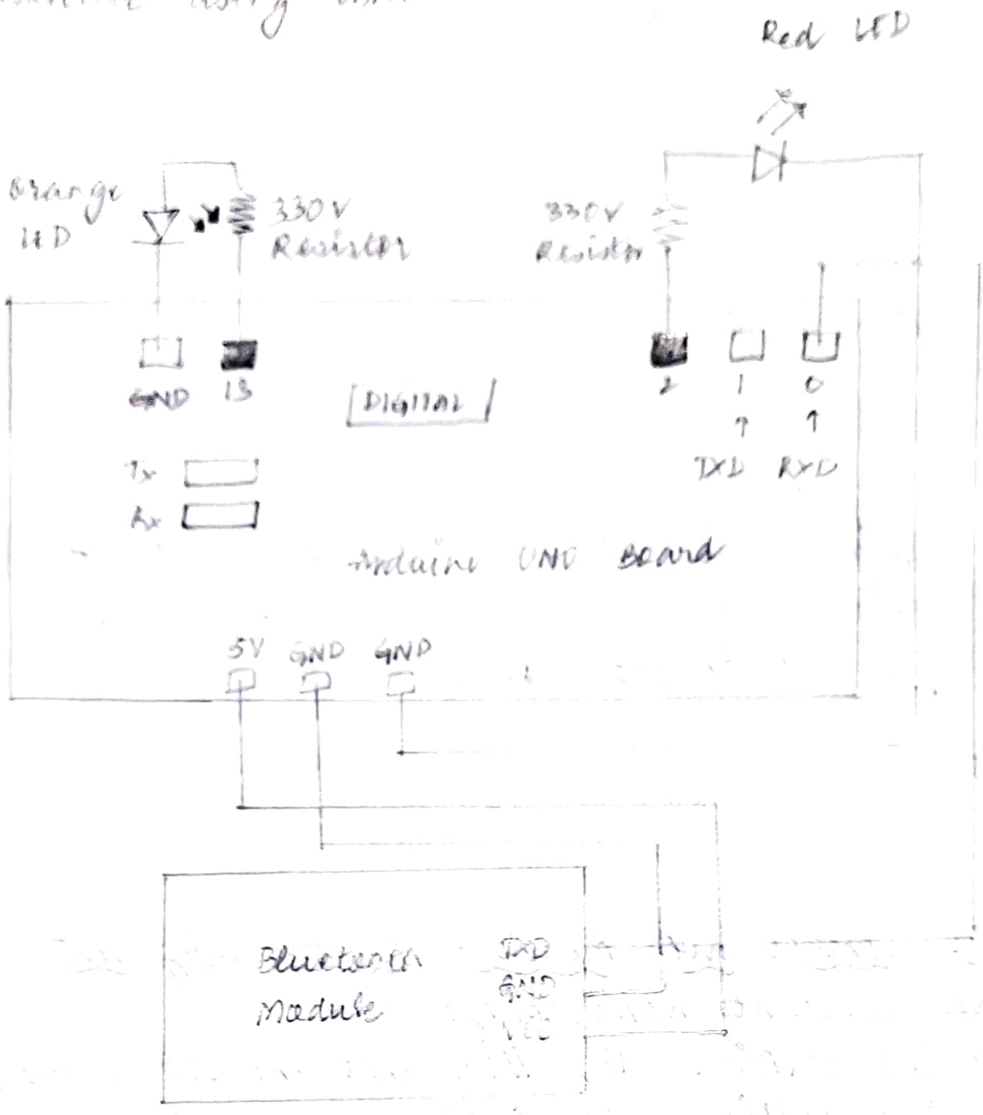
(c) Device control using UART in Arduino

```
// The loop function runs over and over again forever:
void loop()
{
  if ((Serial.available()) > (0))
  {
    char inputchar = Serial.read();
    Serial.print("the receiving character is: ");
    Serial.println(inputchar);
    if (inputChar == 'A')
    {
      digitalWrite(ledPin1, HIGH); // Turn the LED on by
                                   making the voltage HIGH
      digitalWrite(ledPin2, LOW); // Turn the LED off by
                                   making the voltage LOW
    }
    else if (inputchar == 'B')
    {
      digitalWrite(ledPin1, LOW); // Turn the LED off by
                                   making the voltage LOW
      digitalWrite(ledPin2, HIGH); Turn the LED on by
                                   making the voltage HIGH
    }
  }
}
```

* INFERENCE:

Analyze various communication protocols used in the design of portable devices and all simulation results were successfully verified.

(4) Device Control with Bluetooth Module using VDRT



Components Required

1. Arduino UNO Board
2. Orange LED
3. Red LED
4. 330 V Resistor - 2
5. Bluetooth Module HC-05
6. Battery / Power supply
7. Connecting Wires

EXPERIMENT NUMBER : 3
DATE : 28/03/2022, Monday

Webserver and IoT Cloud Communication Using Arduino ...

* AIM :

Analyze various communication protocals used in the design of portable devices.

* CODE (Arduino IDE) :

(a) Identification of Available WiFi Networks -

```
/*
  This code demonstrates how to scan WiFi networks. The API is
  almost the same as with the WiFi shield library, the most
  obvious difference being the different file you need to include.
*/

#include "ESP8266WiFi.h" // ESP8266 WiFi.h library

// The setup function runs once when you prepare for reset or
   power the board :
void setup()
{
  Serial.begin(9600); // Initialize serial communication (UART) with
                         baud rate of 9600 bps

  // Set WiFi to station mode and disconnect from an access
     point, if it was previously connected :
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
```

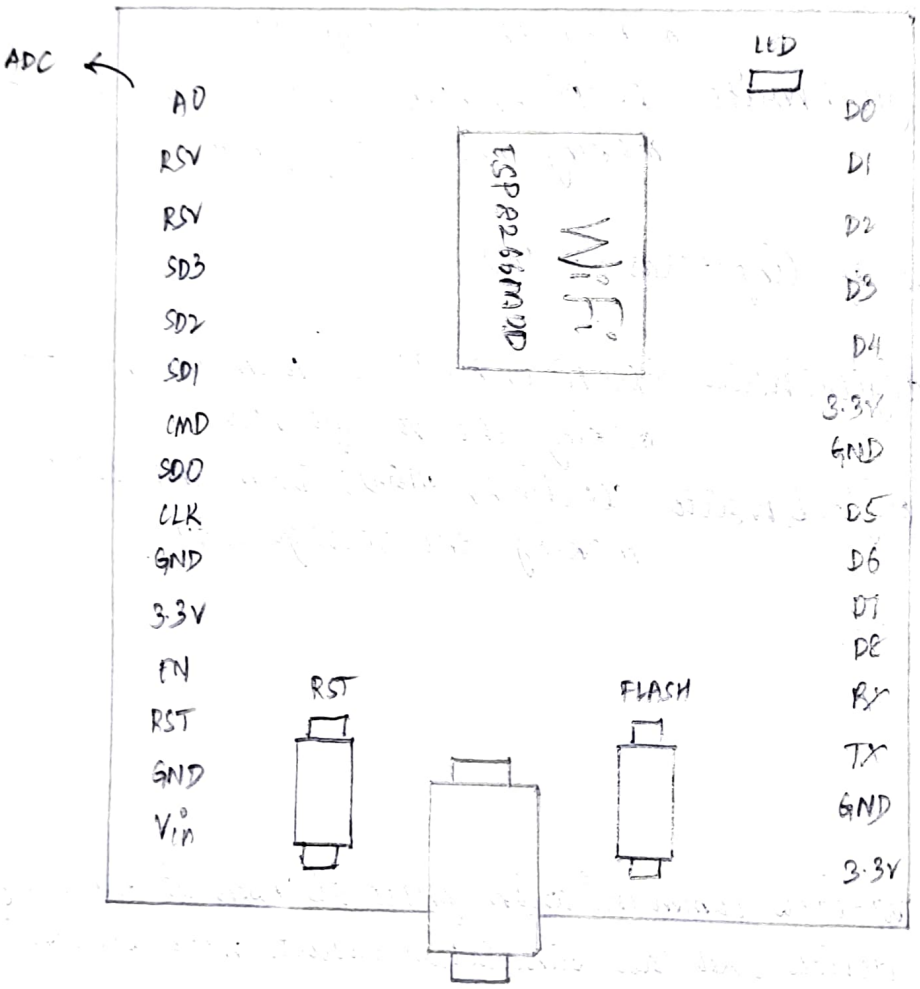
(a) Identification of Available WiFi Networks.

Components used :-

① ESP 8266 WiFi Module

② Power Supply Cable / Battery

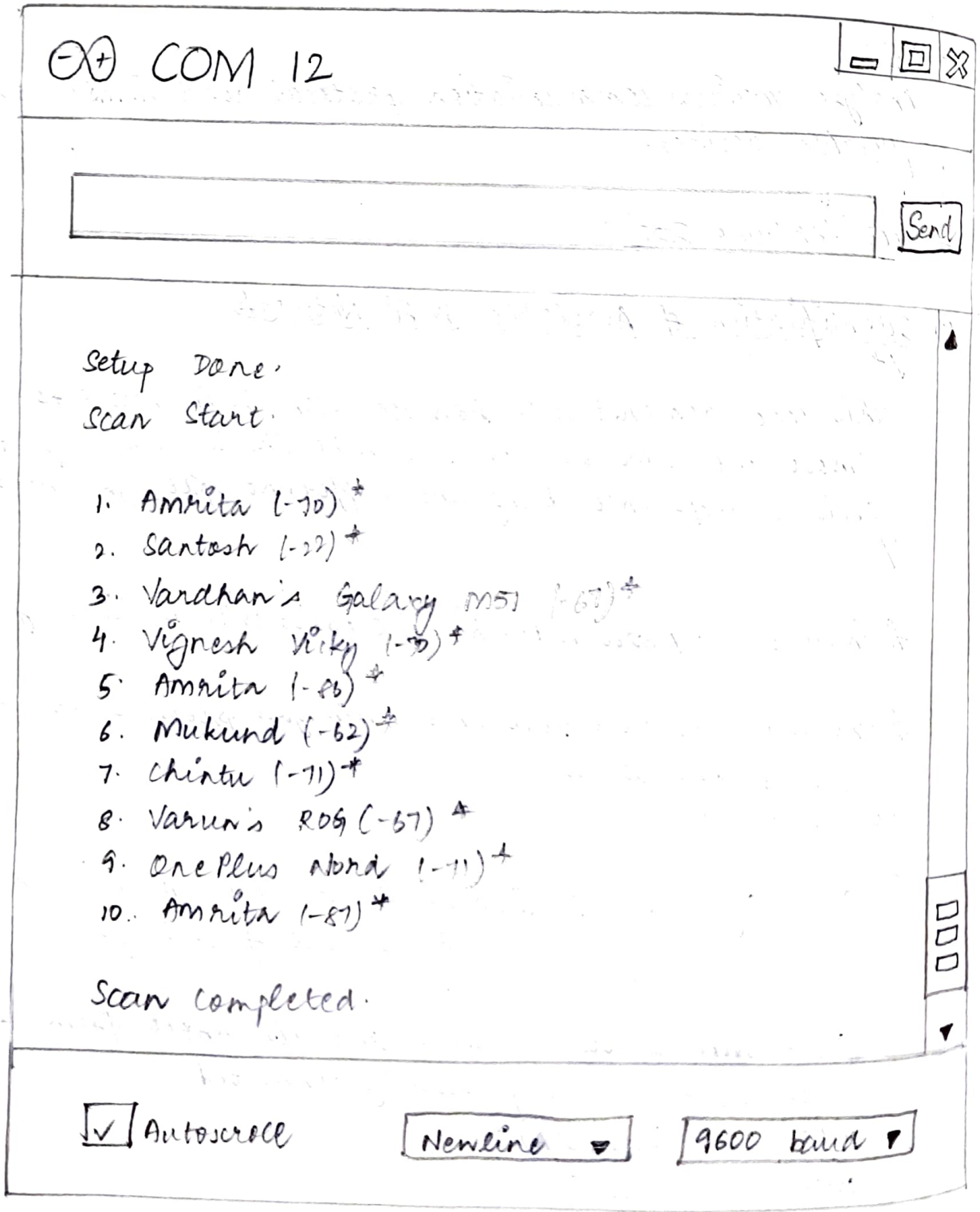
Hardware Labelled Diagram



```
    Serial.println("Setup Done.");
}

// The loop function runs over and over again forever:
void loop()
{
  Serial.println("scan start.");

  // WiFi.scanNetworks will return the number of network
     found:
  int n = WiFi.scanNetworks();
  Serial.println("scan Completed.");

  if (n==0)
    Serial.println("No networks found.");
  else
  {
     Serial.print(n);
     Serial.println(" Networks Found.");

     // Print the SSID (Service Set Identifier) and RSSI
        (Received Signal strength Indicator) for each networks:
     for (int i=0; i<n; ++i)
     {
        Serial.print(i+1);
        Serial.print(": ");

        Serial.print(WiFi.SSID(i));
        Serial.print(" (");

        Serial.print(WiFi.RSSI(i));
        Serial.print(") ");
```

(a) Identification of Available WiFi Networks -

Arduino 1.6.5 - Serial Monitor View



```
        Serial.println(WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " " : "*");
        delay(10);
    }
}

Serial.println("");
delay(5000); // Wait for five seconds before scanning again
}
```

(b) IoT Cloud Communication using ThingSpeak and NodeMCU -

```
#include <ESP8266WiFi.h>  // ESP8266WiFi.h library

const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
const char* host = "api.thingspeak.com";
const char* writeAPIKey = "0xxD4xxx07xxSWxx8";

int temp;
void setup()
{
  // Initialize the sensor:
  pinMode(2, OUTPUT);
  pinMode(14, INPUT);

  Serial.begin(115200); // Initialize serial communication (UART)
                        with baud rate of 1,15,200 bps

  delay(1000);

  // connect to WiFi network:
  WiFi.begin(ssid, password);
```

(b) IoT Cloud Communication using ThingSpeak and NodeMCU.

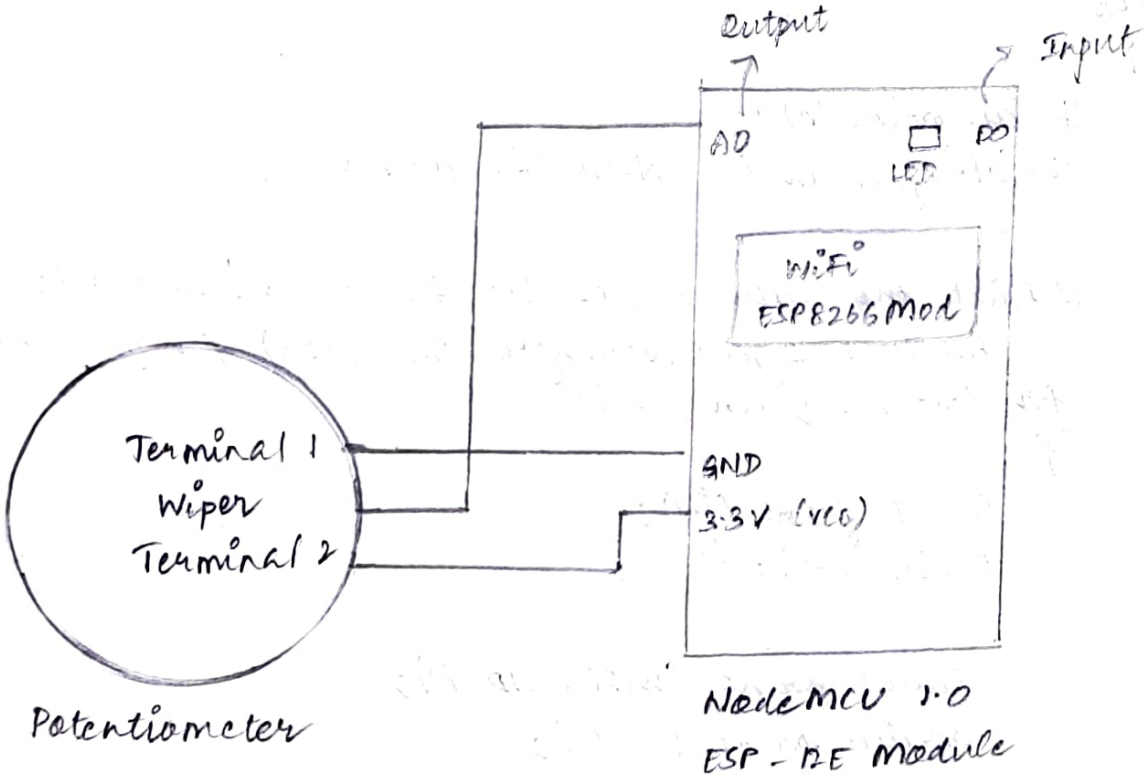
Components Required :-

1. WiFi ESP8266 Board → NodeMCU 1.0 (ESP-12E Module)
2. Potentiometer
3. Battery / Power Supply Cable
4. Connecting Wires

Software Required :-

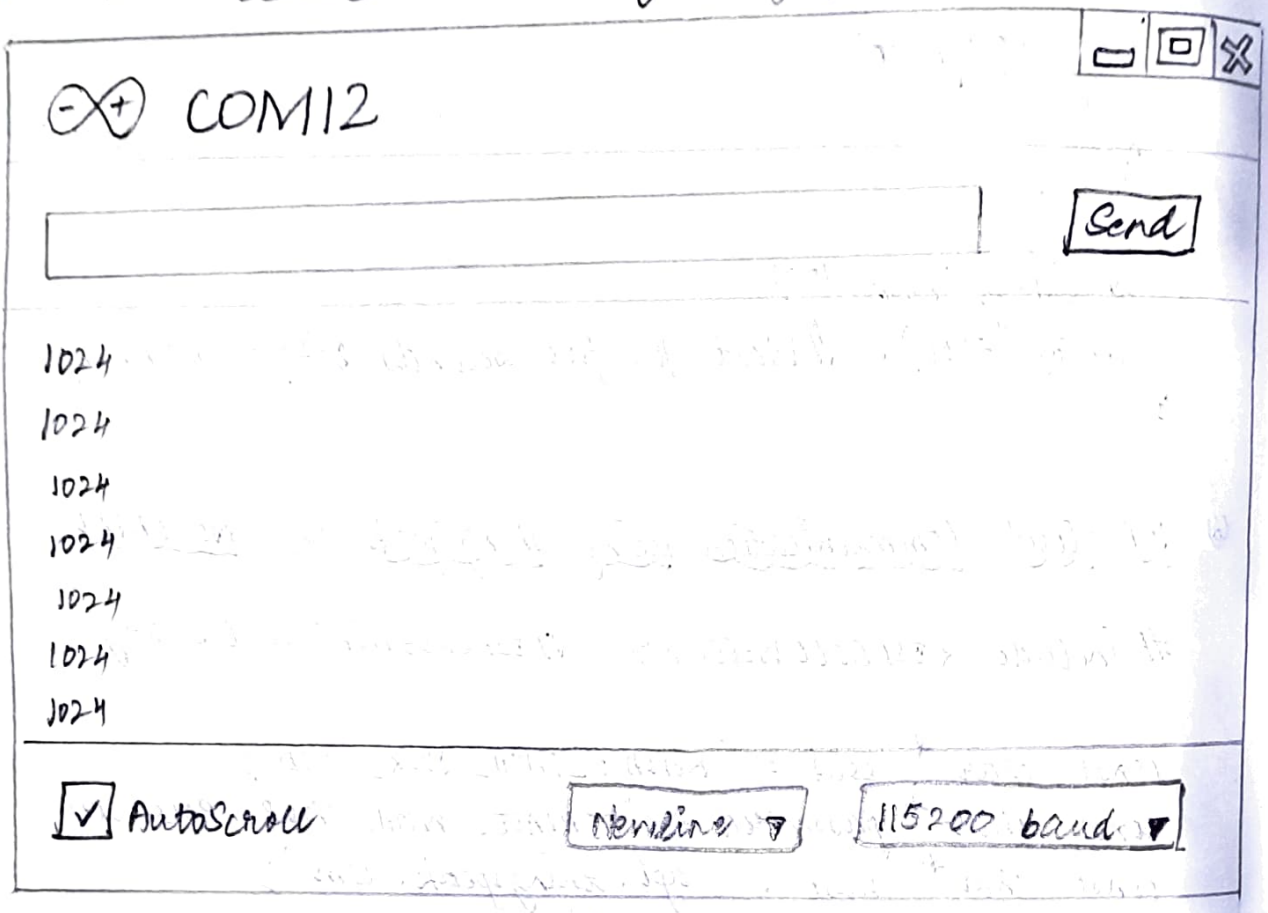
1. Arduino IDE (ESP8266 Board Installed)
2. ThingSpeak Internet of Things (IoT Analytics) (https://thingspeak.com)

Hardware Labelled Diagram

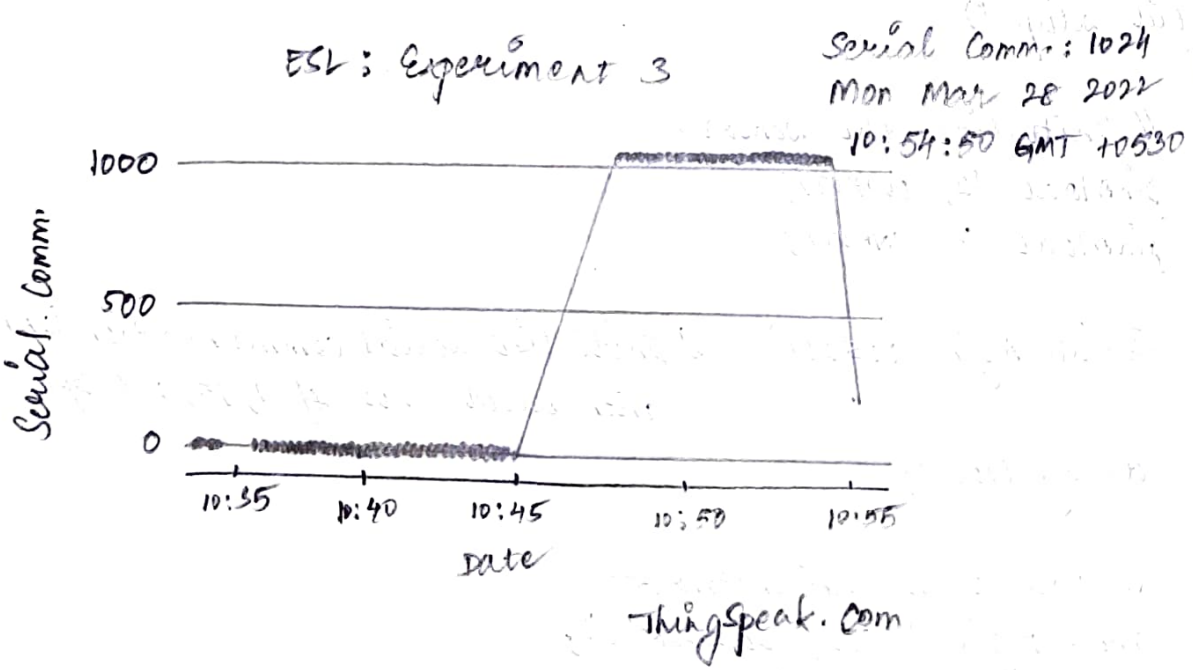


```
// Wait until WiFi connection is established:
while (WiFi.status() != WL_CONNECTED)
 {
  delay(500);
 }
}

// The loop function runs over and over again forever:
void loop()
{
  temp = analogRead(A0);  // Read ADC value from A0

  // Make TCP (Transmission Control Protocol) connections:
  WiFiClient client;
   const int httpPort = 80;

  // wait until the client (NodeMCU) is connected to the
     ThingSpeak Server:
  if (!client.connect(host, httpPort))
  {
    return;
  }

  // Request to the server:
  client.println(sendThingspeak());
  delay(1000);

  if (temp >= 700)
  {
    digitalWrite(2, HIGH);
  }
```

(b) IoT Cloud communication using ThingSpeak and Node MCU.



Arduino 1.6.5 - Serial Monitor View



```
  else
  {
    digitalWrite (2, LOW);
  }

   Serial.print (temp);
   Serial.print ("\n");
}

// Send the ADC value from Client (NodeMCU) to ThingSpeak
   Server in HTML Format:
String sendThingspeak ()
{
  String command =
          String ("GET ") +
           "/update?key=" +
          writeAPIKey +
          "&field1=" +
          String (temp) + "\r\n" +
          "HTTP/1.1 200 OK\r\n" +
          "Host: " + host + "\r\n" +
          "Connection: close\r\n\r\n";
       return command;
}
```

* **INFERENCE:**
Analyze various communication protocols used in the design of portable devices and all simulation results were successfully verified.

Experiment 3

a) Identification of available WiFi Networks

1. Initialize of Serial Communication (UART) with Baud Rate 9600 bps
2. Configure WiFi in “Station Mode” of operation
3. Scan the available WiFi Networks, which will return the no. of available WiFi networks (n).
4. If ‘n’ is zero, then print “ No Networks found”
5. If ‘n’ is non zero, print the SSID and RSSI (Received Signal Strength Indicator) for each networks.

- Initialize of Serial Communication (UART) with Baud Rate 9600 bps

```
Serial.begin(9600);
```

- Two modes available
  1. Station mode - Connect to available WiFi Network
  2. AP mode - Create its own WiFi Network & Devices can connect to this N/W

- Identification of available WiFi Networks

  - Scan the available WiFi Networks, which will return the no. of available WiFi networks (n).

```
int n = WiFi.scanNetworks();
```

Returns

```
n – No.of available Networks identified
```

  - If ‘n’ is zero, then print “ No Networks found” if (n == 0)

```
Serial.println("no networks found");
```

  - If ‘n’ is non zero, print the SSID and RSSI (Received Signal Strength Indicator) for each networks.

```
else
 {
for (int i = 0; i < n; ++i)
  {
        Serial.print(WiFi.SSID(i));
        Serial.print(WiFi.RSSI(i));
        delay(10);
  }
 }
```

b) **IoT Cloud Communication using Thingspeak and NodeMCU**

1. Initialize Serial Communication – UART
2. Initialize the WiFi Network settings and return the current status (Connected or not).
3. Wait until Wifi Connection is established.
4. Read ADC value from A0
5. Wait until the **Client (NodeMCU)** is connected to the Thingspeak Server
6. Send the **ADC value from Client (NodeMCU) to Thingspeak Server** in HTML Format.

EXPERIMENT NUMBER: 4
DATE: 30/08/2022, Wednesday

GRAPHICAL FRONT END DEVELOPMENT USING PROCESSING IDE ...

* **AIM:**
To develop graphical front end using prodessing IDE.

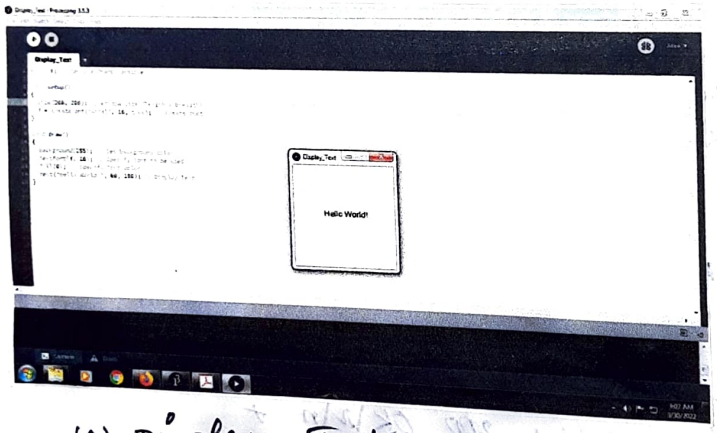
* **CODE (PROCESSING IDE):**

(a) Display Text.

```
PFont f; // Declare PFont variable

void setup()
{
  size(200, 200); // output window size (length x breadth)
  f = createFont("Arial", 16, true); // Create font
}

void draw()
{
  background(255); // set background color
  textFont(f, 16); // Specify font to be used
  fill(0); // Specify font color
  text("Hello, World!", 60, 100); // Display Text
}
```

(b) Slider control using ControlP5 Library.

```
import controlP5.*; // Import controlP5 library
ControlP5 cp5; // controlP5 object
```



(a) Display Test

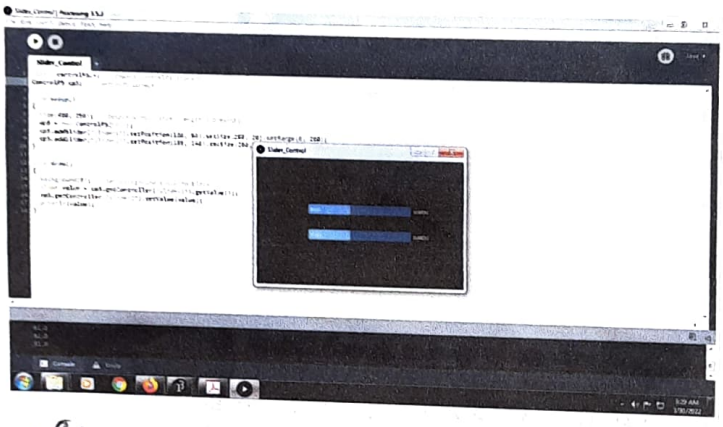
"Hello World!"

- 200 x 200 window size
- Arial Font
- Font size : 16

```
void setup ()
{
  size (400, 250); // output window size (length x breadth)
  cp5 = new ControlP5 (this);
  cp5.addSlider ("slider1"). setPosition (100, 90). setSize (200, 20).
    setRange (0, 200);
  cp5. addSlider ("slider2") . setPosition (100, 140). setSize (200, 20).
    setRange (0, 200);
}


void draw ()
{
  background (0); // Set background color to black
  float value = cp5. getController ("slider1"). getValue ();;
  cp5. getController ("slider 2"). setValue (value);
  println (value);
}
```

(c) Graphical Front End Development using ControlP5 library.

```
import controlP5. *; // Import controlP5 library
ControlP5 controlP5; // ControlP5 object
PFont f; // declare PFont variable

void setup ()
{
  size (360, 260); // output window size (length x breadth)
  smooth ();
  f = createFont ("Arial", 16, true); // create Font
  controlP5 = new ControlP5 (this);
  // Description : A bang controller triggers an event when pressed
  // Parameters : name, x, y, width, height
  controlP5. addBang ("bang1", 10, 10, 20, 20);
```

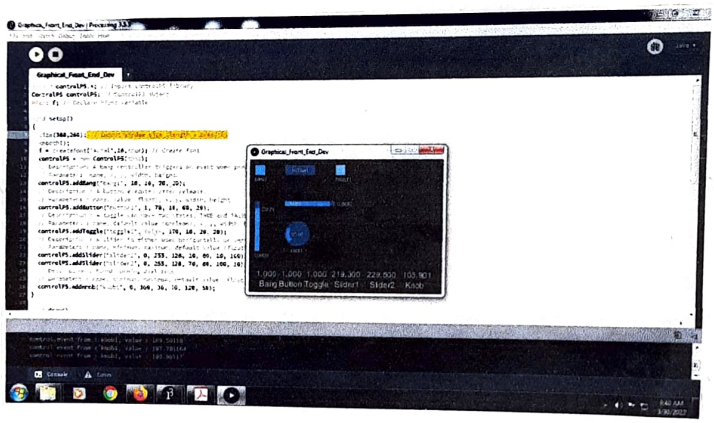


(b) Slider control using controlP5 library

- 400 x 250 window size
- 2 x sliders (Horizontal)
- Black Background
- Print slider value

```
// Description: A button executes after release.
// Parameters: name, value (float), x, y, width, height
controlP5.addButton("button", 1, 70, 10, 60, 20);

// Description: A toggle can have two states, TRUE and FALSE
   where TRUE has the value 1 and FALSE has the value 0.
// Parameters: name, default value (boolean), x, y,
   width, height
controlP5.addToggle("toggle1", false, 170, 10, 20, 20);

// Description: A slider is either used horizontally or vertically.
// Parameters: name, minimum, maximum, default value (float),
   x, y, width, height
controlP5.addSlider("slider1", 0, 255, 128, 10, 80, 10, 100);
  // Width is bigger, you get a horizontal slider.
controlP5.addSlider("seider2", 0, 255, 128, 70, 80, 100, 10);
  // Height is bigger, you get a vertical slider.

// Description: Round turning dial knob.
// Parameters: name, minimum, maximum, default value
   (float), x, y, diameter
controlP5.addKnob("knob1", 0, 360, 30, 70, 120, 50);
}


void draw()
{
  background(0); // Set background color to black
  textFont(f, 16); // specify font to be used
  fill(255); // specify font color

  text("Bang", 20, 250);
  text("Button", 60, 250);
  text("Toggle", 110, 250);
```

(1) Graphical Front end Development using controlP5 library



- 380 x 260 Window size
- Arial Font
- Font size : 16
- objects : Bang, Button, Toggle, Vertical slider, Horizontal slider, Knob
- Black Background

```
  text ("slider 1", 170, 250);
  text ("slider 2", 240, 250);
  text ("knob", 310, 250);

  text (controlP5.getController ("bang1").getValue(), 10, 230);
  text (controlP5.getController ("button1").getValue(), 60, 230);
  text (controlP5.getController ("toggle1").getValue(), 110, 230);
  text (controlP5.getController ("slider1").getValue(), 160, 230);
  text (controlP5.getController ("slider2").getValue(), 230, 230);
  text (controlP5.getController ("knob1").getValue(), 300, 230);
}

void controlEvent (ControlEvent theEvent)
{
  /* events triggered by controllers are automatically forwarded
  to the controlEvent method. By checking the name of a
  controller, one can distinguish which of the controllers
  has been changed. */

  /* check if the event is from a controller, otherwise you'll
  get an error when clicking other interface elements like radio
  button that doesn't support the controller() methods. */

  if (theEvent.isController())
  {
    print ("control event from : " + theEvent.getController().
    getName());
    println (", value: " + theEvent.getController().getValue());
  }
}
```

(d) Display Image -

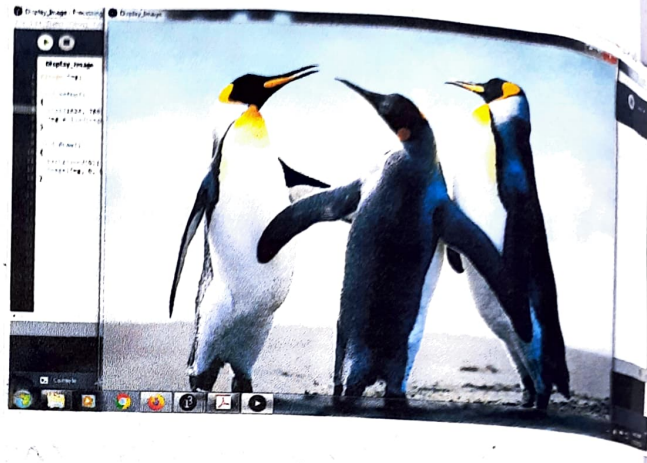
```
PImage img;

void setup()
{
  size(1024, 768); // output window size (length x breadth)
  img = loadImage("Penguins.jpg"); // make a new instance
   of a PImage by loading an image file.
}

void draw()
{
  background(0); // set background color to black
  image(img, 0, 0, width, height); // The image() function
    displays the image at a location, in this case the point (0,0)
}
```

(e) Display Image at Half of its Size -

```
PImage img;

void setup()
{
  size(1024, 768); // output window size (length x breadth)
  img = loadImage("Penguins.jpg"); // make a new instance
   of a PImage by loading an image file.
}

void draw()
{
  background(0); // set background color to black.
  image(img, 0, 0, width/2, height); // Displays the image at
    point (0,0) at half of its size.
```

(d) Display Image

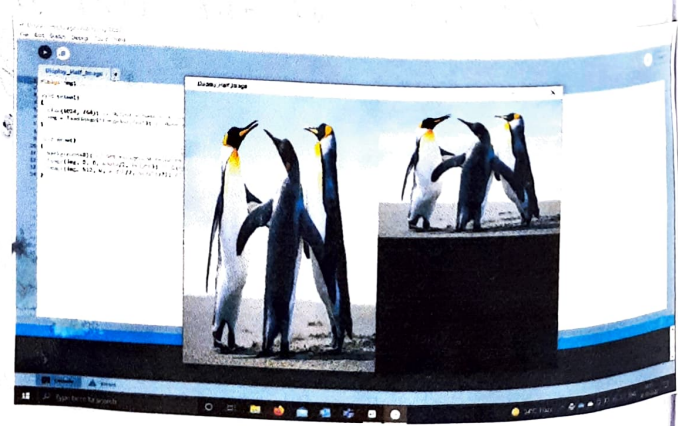
Original Image
size : 1024 x 768 pixels.



Displayed Image.
- Window size: 1024 x 768
- Black Background
- Point : (0, 0)

(e) Display Image at Half of its size

Original Image
size : 1024 x 768 pixels



Displayed Image
- Window size: 1024 x 768
- Black Background
- First point : 10, 0, w/2, h)
- Second point: 1512, 0, w/2, h/2)

```
    image(img, 512, 0, width/2, height/2); // The image()
    function displays the image at a location, in this case
    the point (512, 0);
}
```

(f) Display Image using Pixel Array -

```
PImage img;

void setup()
{
  size(1024, 768); // Output window size (length x breadth)
  img = loadImage("Tulips.jpg");
}

void draw()
{
  loadPixels(); // Call loadPixels() on the PImage to read
                its pixels.
  img.loadPixels();
  for(int x=0; x< width; x++)
  {
    for(int y=0; y< height; y++)
    {
      int loc = x + y * width;

      // The functions red(), green(), and blue() pull out the
        three color components from a pixel:
      float r = red(img.pixels[loc]);
      float g = green(img.pixels[loc]);
      float b = blue(img.pixels[loc]);

      pixels[loc] = color(r, g, b); // Set the digital pixel to the
        image pixel
```

(f) Display Image using Pixel Array

Original Image
Size: 1024 x 768 pixels

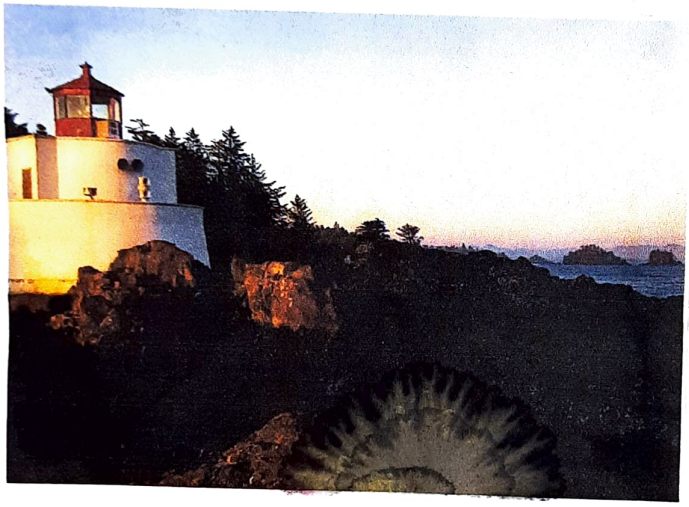
Displayed Image
window size: 1024 x 768

```
    }
  }
  updatePixels();
}
```

(9) Threshold Image using Pixel Array -

```
PImage source; // source image
PImage destination; // Destination image

void setup()
{
  size(1024, 768); // output window size (length x breadth)
  source = loadImage("Lighthouse.jpg");
  destination = createImage(source.width, source.height, RGB);
}

void draw()
{
  float threshold = 121;
  source.loadPixels();
  destination.loadPixels();
  for(int x=0; x < source.width; x++)
  {
    for(int y=0; y < source.height; y++)
    {
      int loc = x + y * source.width;

      // Test the brightness against the threshold:
      if(brightness(source.pixels[loc]) > threshold)
      {
        destination.pixels[loc] = color(255); // white
      }
```

(g) Threshold Image using Pixel Array



original Image
size: 1024 x 768 pixels

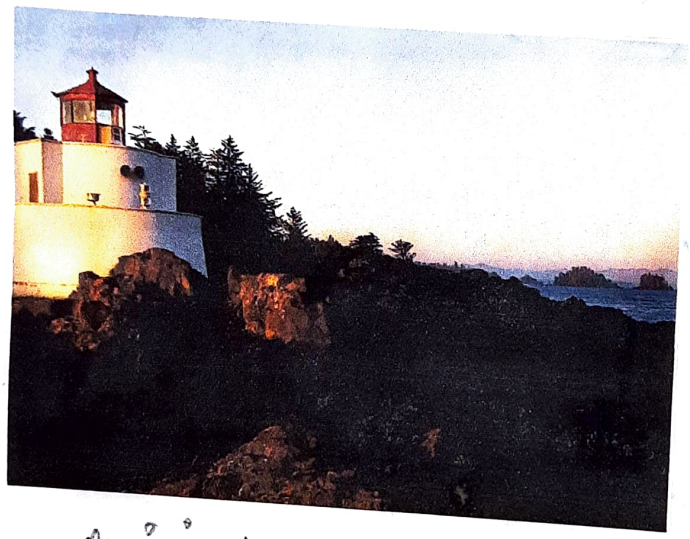
Threshold Image
- Window size: 1024 x 768
- Threshold Value: 127
- Destination point: (0, 0)

```
        else
        {
          destination.pixels[loc] = color(0); // Black
        }
      }
    }

    destination.updatePixels(); // Update the pixels in destination
    image(destination, 0, 0); // Display the destination
}
```

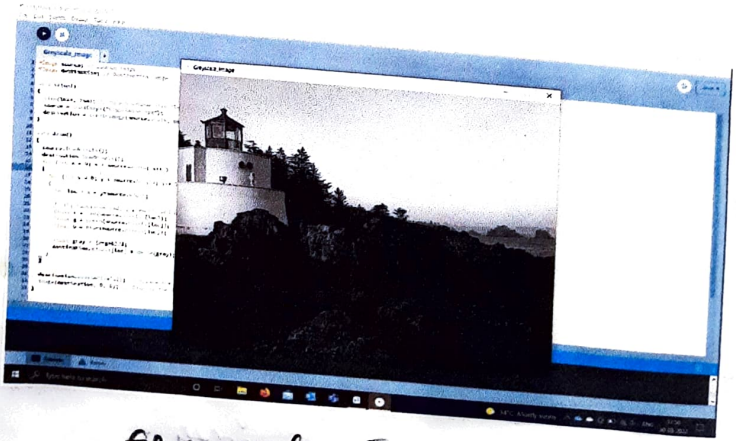
(2) Greyscale Image using Pixel Array -

```
PImage source; // Source image
PImage destination; // Destination image

void setup()
{
  size(1024, 768); // Output window size (length x breadth)
  source = loadImage("lighthouse.jpg");
  destination = createImage(source.width, source.height, RGB);
}

void draw()
{
  source.loadPixels();
  destination.loadPixels();
  for(int x=0; x < source.width; x++)
  {
    for(int y=0; y < source.height; y++)
    {
      int loc = x + y * source.width;
```

(A) Greyscale Image using Pixel Array



Original Image
Size: 1024 x 738 pixels



Greyscale Image
- Window size: 1024 x 768
- Gray = (Red + Green + Blue)/3
- Destination Point: (0,0)

```
      // The functions red(), green(), and blue() pull out
         the three components from a pixel:
      float r = red(source.pixels[loc]);
      float g = green(source.pixels[loc]);
      float b = blue(source.pixels[loc]);

      float gray = (r+g+b)/3;
      destination.pixels[loc] = color(gray);
    }
  }

  destination.updatePixels(); // Update the pixels in destination
  image(destination, 0, 0); // Display the destination
}
```

+ INFERENCE:

Analyzed and developed graphical front end using processing IDE and all simulation results were verified successfully.

EXPERIMENT NUMBER : 5

DATE : 05/04/2022, TUESDAY

ARDUINO INTERFACING WITH PROCESSING IDE

* **AIM :**

To design and develop embedded systems using Arduino and Processing IDE.

* **CODE :**

(A) **Serial Reception :-**

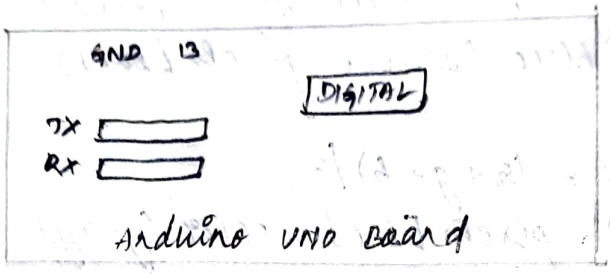
① **Arduino Source Code -**

```
// The setup function runs once when you press reset or power
the board :
void setup ()
{
  Serial.begin (9600); // start serial communications at 9,600
                          bits per second
}

// The loop function runs over and over again forever :
void loop ()
{
  Serial.println ("Hello, World!");
  delay (100); // Delay in between reads for stability
}
```

② **Processing source Code -**

```
import processing.serial.*;
Serial myPort ; // Create object from Serial class
```

(a) Serial Reception :



Processing IDE - Output

Hello, World!
Hello, World!
Hello, World!
Hello, World!

Components Required

① Arduino UNO Board
② Battery / Power Supply (adapter)
③ Connecting Wires as required

```
String val; // Variable to store the data received from
                the serial port

void setup ()
{
  // On Windows machine, this generally opens in COM1.
  println (Serial.list());
  String portName = Serial.list()[2]; // change the 0 to 1
  or 2 etc., to match your port
  myPort = new Serial (this, portName, 9600);
}


void draw ()
{
  if (myPort.available() > 0)
  {
    val = myPort.readStringUntil ('\n'); // Read the value
    and store it in a variable "val"
  }
  println (val); // Print it out in the console
}
```

(b) Device control using Button :-

(i) Arduino source code -

```
// The setup function runs once when you press reset or
   power the board :
void setup ()
{
  pinMode (13, OUTPUT); // Set pin 013 (RED-LED) as output
  Serial.begin (9600); // start serial communications at 9,600
                          bits per second

}
```

```
// The loop function runs over and over again forever:
void loop()
{
  if (serial.available())
  {
    // If data is available to read now:
    char val = serial.read();
    if (val == '1')
    {
      digitalWrite(13, HIGH); // Turn ON Arduino LED
    }
    else if (val == '0')
    {
      digitalWrite(13, LOW); // Turn OFF Arduino LED
    }
  }
}
```

(b) Processing source code -

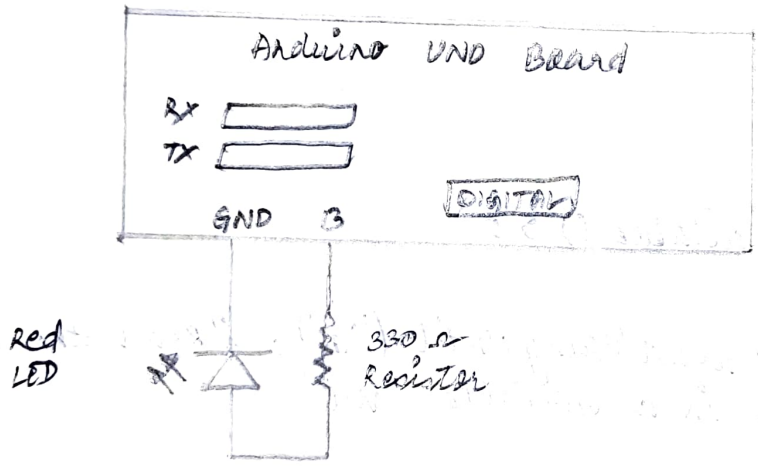
```
import controlP5.*; // Import controlP5 library
import processing.serial.*;
Serial myPort;
ControlP5 cp5; // Create controlP5 object
PFont font;

// same as setup in Arduino Program:
void setup()
{
  size(300, 250); // Windows size (Width, Height)
  printArray(Serial.list()); // Prints all the available
                             serial ports
  myPort = new Serial(this, "COM7", 9600);
```

(b) Device Control using Button :-

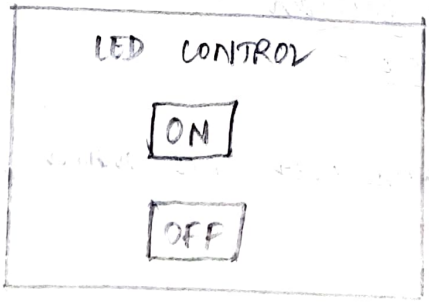
→ Components Required -

1. Arduino UNO Board
2. Red - LED
3. 330 Ω Resistor
4. Battery / Power Supply Cable
5. Connecting Wires, as required

→ Circuit Diagram -



→ Processing IDE Output -



1 → ON
0 → OFF
1 → ON
0 → OFF
1 → ON
0 → OFF

```
cp5 = new ControlP5(this); // Add button to empty window
font = createFont("calibri light bold", 20); // Custom fonts
  for buttons and title

cp5.addButton("on") // "blue" is the name of button
.setPosition(100, 50) // x and y coordinates of upper left corner
                                                  of button
.setSize(120, 70) // (width, height)
.setFont(font);

cp5.addButton("off") // "OFF" is the name of button
.setPosition(100, 150) // x and y coordinates of upper left corner
                                                  of button
.setSize(120, 70) // (width, height)
.setFont(font);
}

// Same as Loop in Arduino Program:
void draw()
{
  background(150, 0, 150); // Background color of window
      (r, g, b) or (0 to 255)

  // Title to the Window:
  fill(0, 255, 0); Text color (r, g, b)
  textFont(font);
  text("LED CONTROL", 80, 30); // ("TEXT", x-coordinate,
                                  y-coordinate)
}

// Add functions to buttons: when the button is pressed, it
  sends a particular char over serial port:
void on()
{ myPort.write('1');
  println("1");
}
```

```
void off ()
{
  myPort.write ('0');
  println ("0");
}
```

11) Device control using Mouse click :-

① Arduino source code -

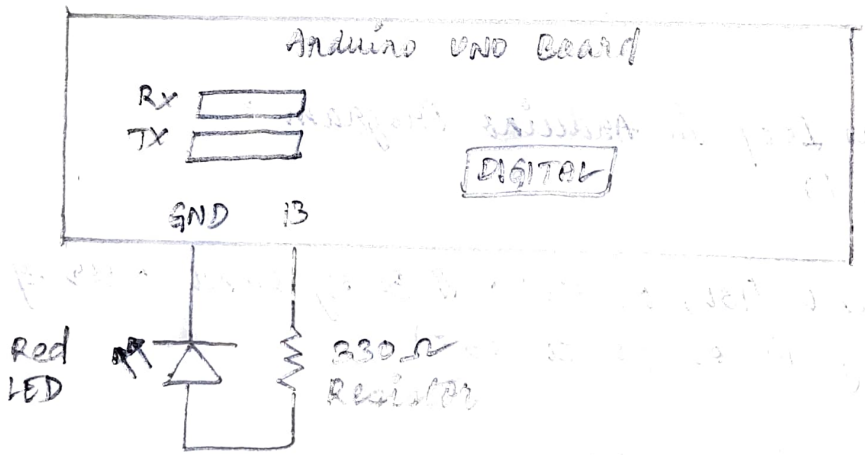
```
char val ; // Variable to store the data received from the
                serial port
int ledPin = 13; // LED Pin of Arduino

// The setup function runs once when you press reset or power
   the board:
void setup ()
{
  pinMode (ledPin, OUTPUT); // Set pin D13 (RED-LED) as output
  Serial.begin (9600); // start serial communications at 9,600
                          bits per second
}

// The loop function runs over and over again forever:
void loop ()
{
  if (Serial.available ())
  {
    // If data is available to read now:
    val = Serial.read (); // Read the value and store it in a
                             variable "val"
  }
```

(c) Device Control using Mouse Click :-

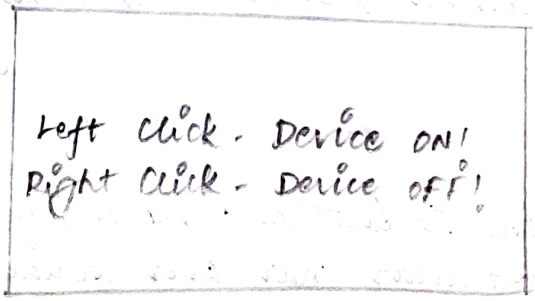
→ Components Required -

① Arduino UNO Board
② RED - LED
③ 330 Ω Resistor
④ Battery / Power supply cable
⑤ Connecting Wires, as required

→ Circuit Diagram -



→ Processing IDE Output -



1 → Left Click
0 → Right Click
1 → Left Click
0 → Right Click

```
  if (val == '0')
  {
    digitalWrite (ledPin, LOW); // Turn OFF Arduino LED
  }

  else if (val == '1')
  {
    digitalWrite (ledPin, HIGH); // Turn ON Arduino LED
  }

  delay (10); // Delay in between reads for stability
}
```

② Processing source code -

```
import processing.serial.*;
PFont f;
Serial myPort; // Create object from Serial class
String val; // Variable to store the data received from the
            serial port

void setup()
{
 size (200, 200); // output window size (width, height)
 f = createFont ("Arial", 16, true); // create Font
 String portName = Serial.list()[2]; // change the 0 to 1 or
  2 etc., to match your port
 myPort = new Serial (this, portName, 9600);
}

void draw()
{
  background (255); // set background color
  textFont (f, 14); // specify font to be used
  fill (0); // specify font color
```

```
text ("Left click - Device ON!", 20, 100); // Display Text
text ("Right click - Device OFF!", 200, 120); // Display Text

  if (mousePressed == true)
  {
    // If clicked in the window:
    if (mouseButton == LEFT)
    {
      myPort.write('1'); // Send a 1 (HIGH)
      println("1");
    }
    else if (mouseButton == RIGHT)
    {
      myPort.write('0'); // Send a 0 (low)
      println("0");
    }
  }
}
```

(d) Device Control using Keyboard-

① Arduino source code -

```
char val; // Variable to store the data received from the
                serial port
int ledPin = 13; // LED Pin of Arduino

// The setup function runs once when you press reset or power
   the board:
void setup()
{
  pinMode(ledPin, OUTPUT); // Set pin DB (RED - LED) as output
  Serial.begin(9600); // start serial communications at 9,600
                          bits per second
}
```

```
// The loop function runs over and over again forever:
void loop ()
{
  if (serial.available ())
  {
    // If data is available to read now:
    val = Serial.read (); // Read the value and store it in
                          a variable "val"
  }

  if (val == '1')
  {
   digitalWrite (ledPin, HIGH); // Turn ON Arduino LED
  }
   else if (val == '0')
   {
    digitalWrite (ledPin, LOW); // Turn OFF Arduino LED
   }

  delay (10); // Delay in between reads for stability
}
```

② Processing source code:

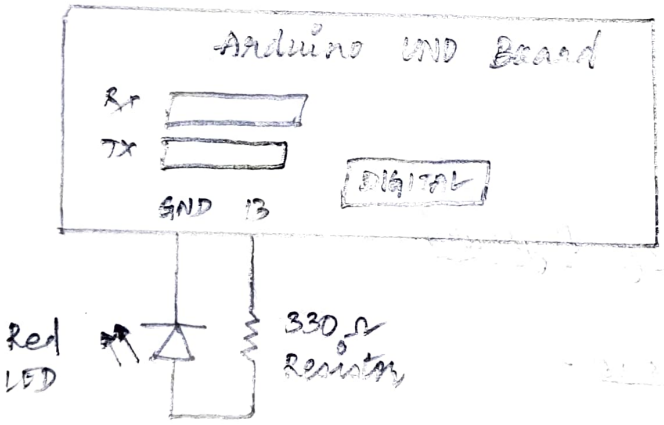
```
import processing.serial.*;
PFont f; // Declare PFont variable
Serial myPort; // Create object from Serial class
String val; // Variable to store the data received from the
            serial port

void setup ()
{
  size (220, 200); // output Window size (width, height)
  f = createFont ("Arial", 16, true); // Create font
```

(d) Device Control using Keyboard -

→ Components Required -

① Arduino UNO Board
② Red - LED
③ 330 Ω Resistor
④ Battery / Power Supply Cable
⑤ Connecting Wires, as required

→ Circuit Diagram.



→ Processing IDE Output -

'A' in Keyboard - Device ON!
'B' in Keyboard - Device OFF!

1 → 'A' in Keyboard
0 → 'B' in Keyboard
1 → 'A' in Keyboard

```
  String portName = Serial.list()[2]; // Change the 0 to 1 or 2 etc.
      to match your port
  myPort = new Serial(this, portName, 9600);
}

void draw()
{
  background(255); // set background color to black
  textFont(f, 14); // Specify font to be used
  fill(0); // Specify font color
  text("'A' in Keyboard - Device ON!", 20, 100); // Display Text
  text("'B' in keyboard - Device OFF!", 20, 120); // Display Text
}

void keyPressed()
{
  if (key == 'A' || key == 'a')
  {
    myPort.write('1'); // Send a 1 (HIGH)
    println("1");
  }
  else if (key == 'B' || key == 'b')
  {
    myPort.write('0'); // Send a 0 (LOW)
    println("0");
  }
}
```

(e) Servo Motor Control using Knob -

① Arduino Source Code -

```
#include <Servo.h>
Servo myservo; // create servo object to control a servo
```

```
int servoPin = 2; // Connect yellow servo wire to a digital GPIO
  pin 2 (D2 of Arduino), must be PWM (Pulse Width Modulation)
int val = 0; // Variable to store the data received from the
                serial port

// The setup function runs once when you press reset or power the
board:
void setup()
{
  myservo.attach(servoPin); // Attach the servo to the PWM pin
  Serial.begin(9600); // serial communications start at 9,600
                         bits per second
}

// The loop function runs over and over again forever:
void loop()
{
  if (Serial.available())
  {
    // ID data is available to read now:
    val = Serial.Read(); // Read the value and store it in a
                            variable "val"
  }

  myservo.write(val); // Set the servo position
  delay(15); // Wait for the servo to get there
}
```

② Processing source code -

```
import processing.serial.*;
Serial myPort; // Create object from serial class
import controlP5.*; // Import controlP5 library
```

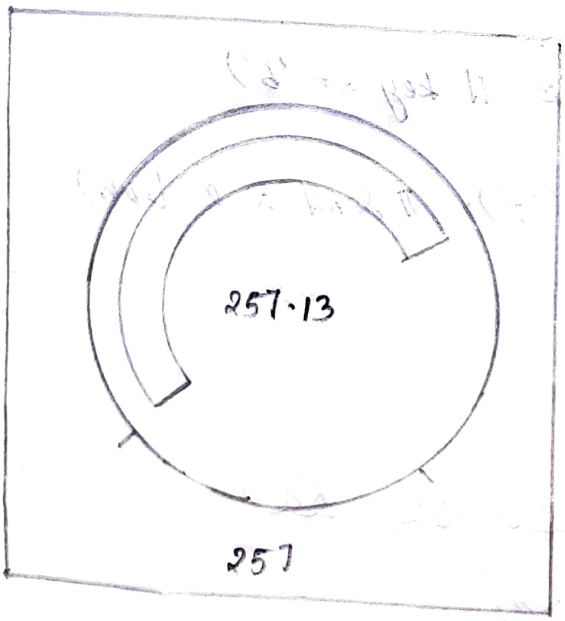
(c) Servo Motor Control using Knob:

→ Components Required -

① Arduino UNO Motor
② Servo Motor
③ Battery / Power Supply Cable
④ Connecting Wires as required

→ Circuit Diagram.

→ Processing IDE output -



Control event from : knob1, value : 257

```
ControlP5 controlP5; // ControlP5 object
PFont f; // Declare PFont variable

void setup ()
{
  size (200, 200); // Output Window Size (width, Height)
  String portName = serial.list () [2]; // change the 0 to 1 or 2
    etc., to match your port
  myPort = new Serial (this, portName, 9600);

  smooth ();
  f = createFont ("Arial", 16, true); // Create Font
  controlP5 = new ControlP5 (this);

  // Description: Round turning dial knob
  // Parameters: name, minimum, maximum, default value (float)
                    x, y, diameter
   controlP5.addKnob ("Knob 1", 0, 360, 30, 50, 40, 100);
}

void draw ()
{
  background (255); // Set background to black
  textFont (f, 10); // Specify font to be used
  fill (0); // Specify font color
  text ("Knob", 310, 260);
  text (int (controlP5.getController ("Knob 1").getValue ()), 90, 190);
}

void controlEvent (ControlEvent theEvent)
{
  if (theEvent.isController ())
  {
```

```
    print ("control event from: " + theEvent.getController().
      getName());
    println(", value: " + int(theEvent.getController().getValue())
      );
    myPort.write(int(theEvent.getController().getValue()));
  }
}
}
```

+ INFERENCE:

Designed and developed embedded systems using Arduino and Processing IDE and all simulation results were verified successfully.