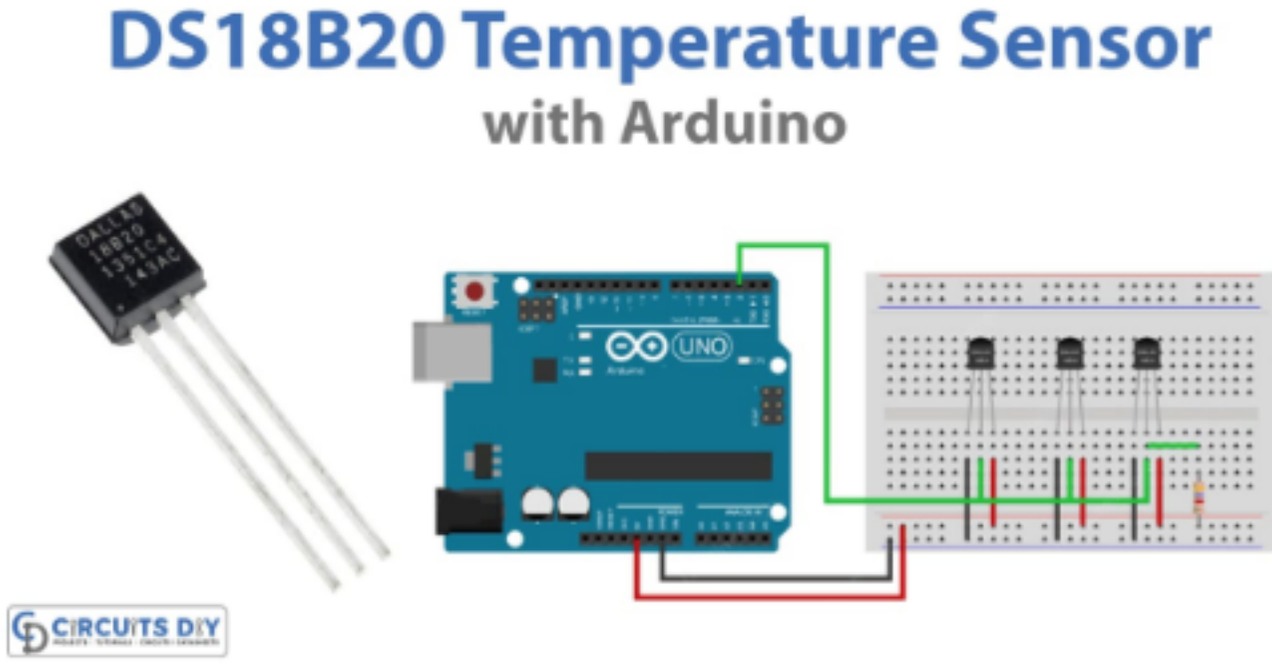
Smart water fountain

Sensor:



1.Temperature Sensor:

A water-proof temperature sensor is going to be used. Part number from is:

DS18B20 [6]. This temperature sensor is compatible with a relatively wide range of power

Supply from 3.0V to 5.5V. The measured temperature ranges from -55 to +125 C

Degrees. Between -10 to + 85 degrees, the accuracy is up to +-0.5 degrees. This sensor can

Fulfill all requirements needed for this project.

2 PH-sensor: PH value is a valued indicator of water quality. This PH-sensor[7] works with 5V voltage,

Which is also compatible with the temperature sensor. It can 6measure the PH value from 0

To 14 with an accuracy of +- 0.1 at the temperature of 25 degrees.

.3 Conductivity sensor: sensor is also part of the water quality assessment. The input voltage is from

3.0 to 5.0V. The error is small, +-5%F.S. The measurement value ranges from 0 to 20

Ms/cm which is enough for water quality monitoring. [8]

4 Liquid Level Sensor: This sensor [9] is responsible for reflecting how much freshwater is left in the water tank.

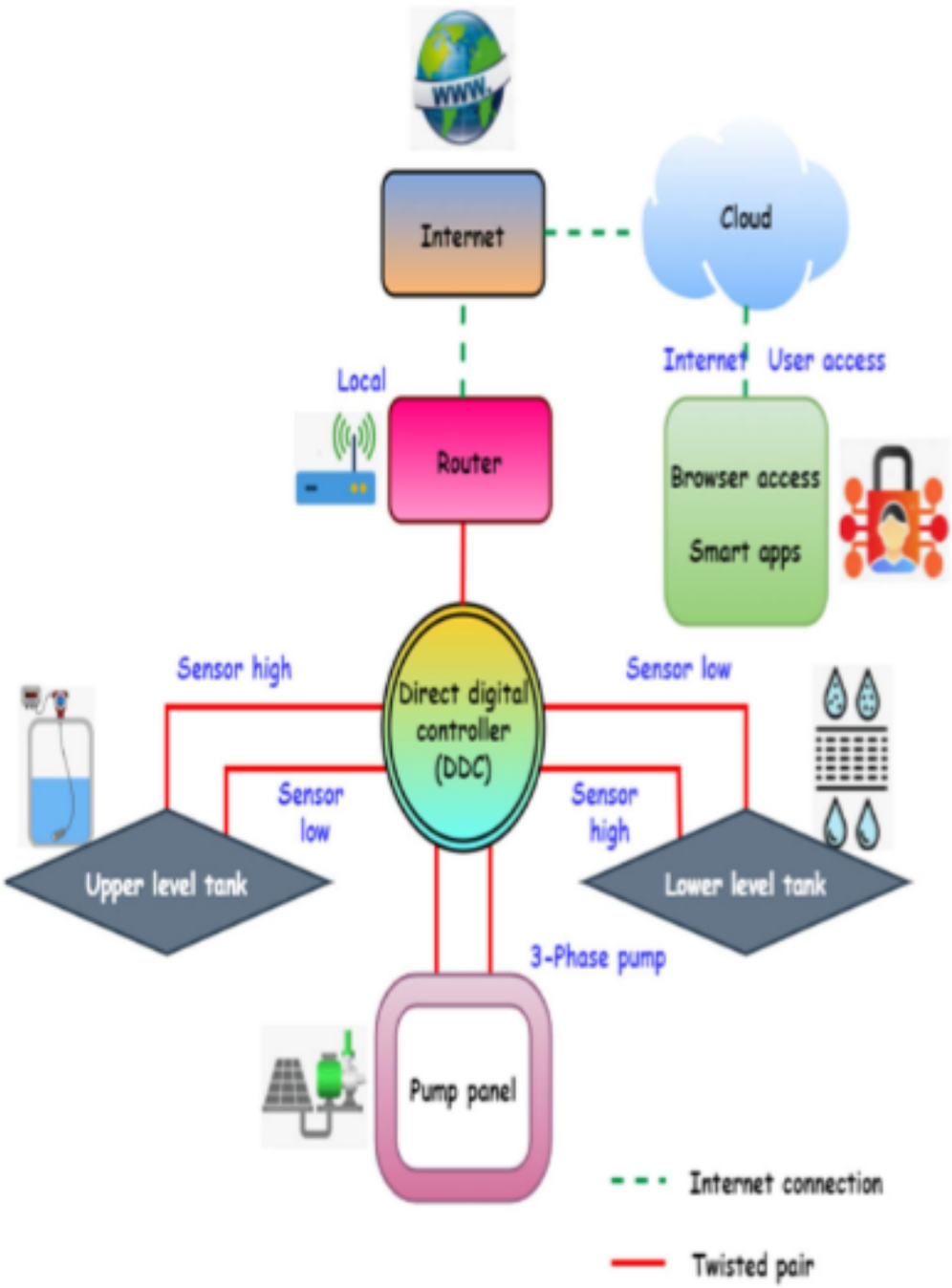
When the water level is low, fresh water will be pumped to the water tank to ensure the

Water fountain keeps running with freshwater. This sensor is 0.5 Watts. For water level from

0 to 9 inches, the corresponding sensor outputs readings from 0 to 1.6. From that, the

Quantity of freshwater left can be determined.

General Layout:



ALGORITHM:

1.Initialization:

Set up the water fountain with a proximity sensor (e.g., infrared or ultrasonic) and a water pump.

2.Start Loop:

Continuously monitor the proximity sensor to detect if there's someone nearby.

3.Detection:

If the proximity sensor detects someone within a predefined range (e.g., a few feet), proceed to the next step. If not, continue monitoring.

4.Dispense Water:

Activate the water pump to dispense water for a predefined duration (e.g., a few seconds) when someone is detected.

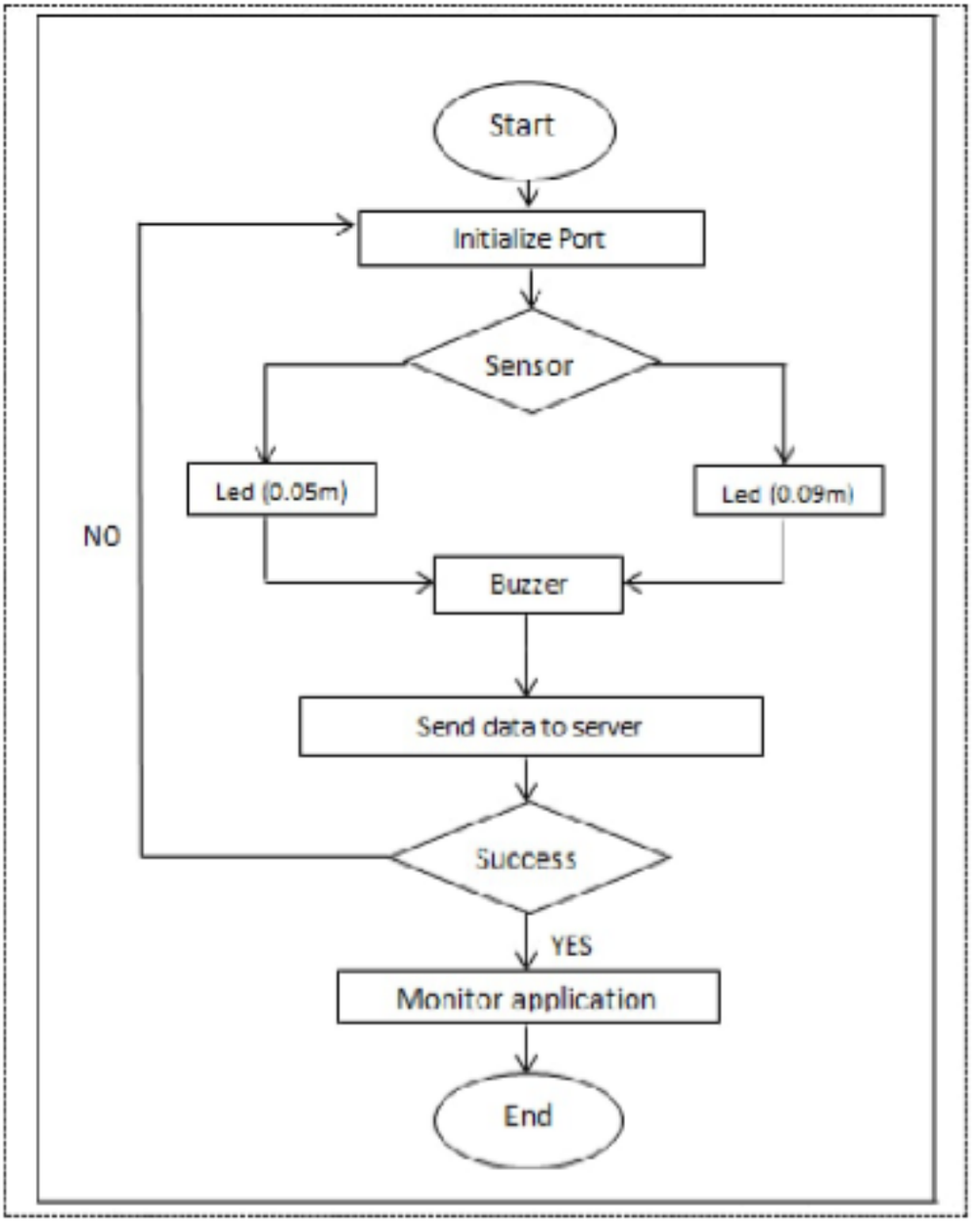
5.Delay:

Add a delay or cool down period to prevent continuous water dispensing. After dispensing, wait for a predefined time before checking for proximity again.

6.Repeat:

Go back to step 2 and continue monitoring for proximity.

Flowchart:


Start
Initialize Port
Sensor
Led (0.05m)
Led (0.09m)
NO
Buzzer
Send data to server
Success
YES
Monitor application
End

Program:

```
Def activate_fountains(fountains):
   N = len(fountains)
   Min_fountains = float('inf')
   For I in range(1, 2**n):
      Activated = []
      For j in range(n):
         If (I >> j) & 1:
            Activated.append(j)
      If is_covered(activated, fountains):
         Min_fountains = min(min_fountains, len(activated))
   Return min_fountains
Def is covered(activated, fountains):
   N = len(fountains)
   Coverage = [0] * 1
    For I in activated:
        Left = max(0, I – fountains[i]
         Right = min(n – 1, I + fountains[i])
    For j in range(left, right + 1):
    Coverage[j] = 1
   Return all(coverage)
    # example usage
   A1 = [1, 2, 1]
   A2 = [2, 1, 1, 2, 1]
   printf(activate_fountains(a1)) # output: 1
   printf(activate_fountains(a2))# output:2
```

OUTPUT:

1

2

2. // Java program to implement

// the above approach

Import java.util.*;

Class GFG {

// Function to find minimum

// number of fountains to be

// activated

Static int minCntFoun(int a[], int N)

// dp[i]: Stores the position of

// rightmost fountain that can

// be covered by water of leftmost

// fountain of the i-th fountain

Int[] dp = new int[N];

For(int i=0;i<N;i++)

{

Dp[i]=-1;

}

// Stores index of leftmost fountain

// in the range of i-th fountain

Int idxLeft;

// Stores index of rightmost fountain

// in the range of i-th fountain

Int idxRight;

// Traverse the array

For (int I = 0; I < N; i++)

{

```
// Stores count of fountain

// needed to be activated

Int cntfount = 1;

// Stores index of next fountain

// that needed to be activated

Int idxNext = 0;

idxRight = dp[0];

// Traverse dp[] array

For (int I = 0; I < N; i++)

{

   idxNext = Math.max(idxNext, dp[i]);

   // If left most fountain

   // cover all its range

   If (I == idxRight)

   {

      Cntfount++;

      idxRight = idxNext;

        }

     }

      Return cntfount;

   }

   // Driver Code

   Public static void main(String[] args)

   {

      Int a[] = { 1, 2, 1 };

      Int N = a.length;

    System.out.print(minCntFoun(a, N));

   }

}
```

Output:

1

3. // C# program to implement

// the above approach

Using System;

Class GFG {

```
    // Function to find minimum
    // number of fountains to be
    // activated
    Static int minCntFoun(int[] a, int N)
    {
        // dp[i]: Stores the position of
        // rightmost fountain that can
        // be covered by water of leftmost
        // fountain of the i-th fountain
        Int[] dp = new int[N];
        For (int I = 0; I < N; i++)
        {
            Dp[i] = -1;
        }
        // Stores index of leftmost
        // fountain in the range of
        // i-th fountain
        Int idxLeft;
        // Stores index of rightmost
        // fountain in the range of
```

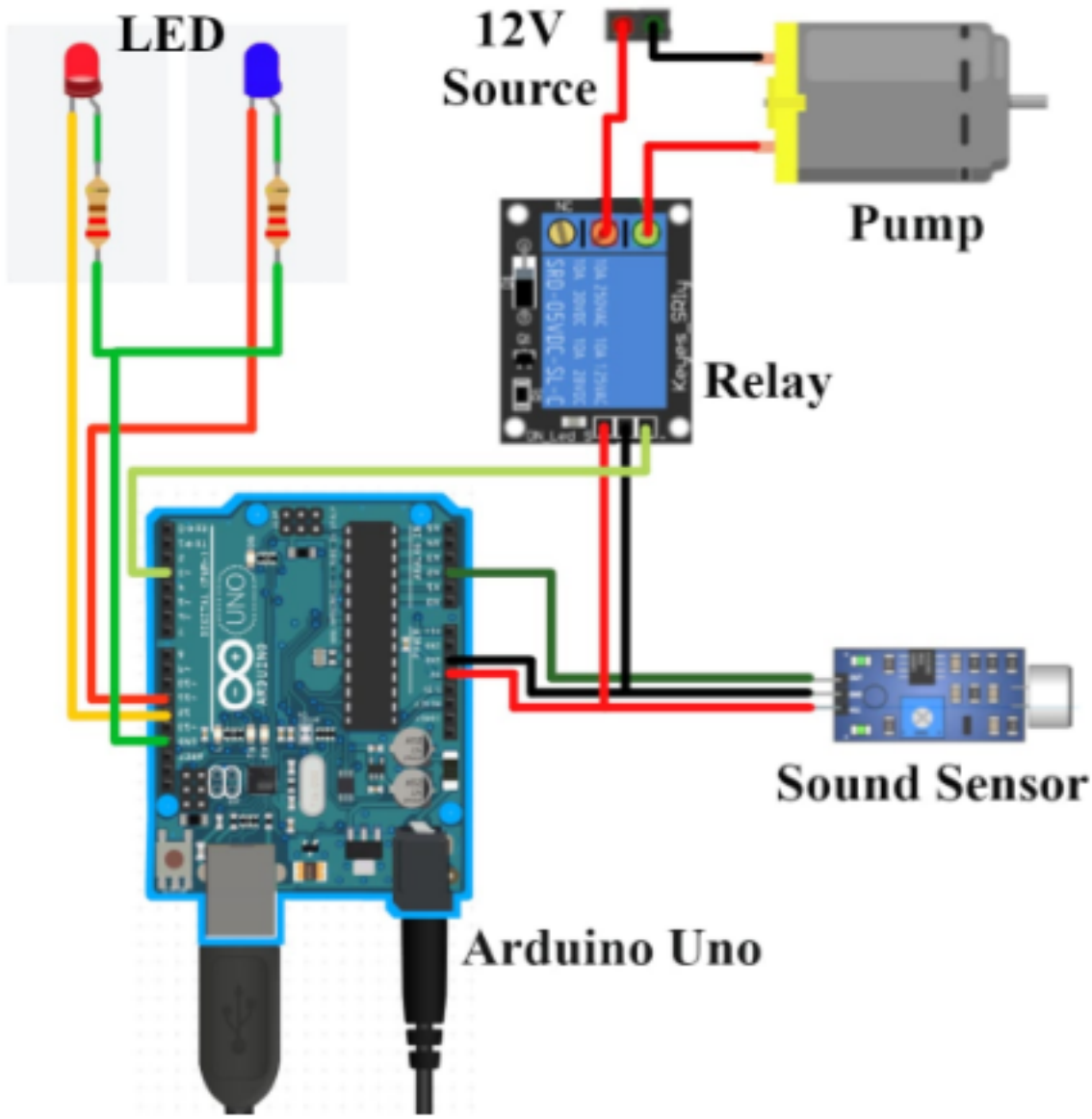
```
// i-th fountain
Int idxRight;
// Traverse the array
For (int I = 0; I < N; i++) {
   idxLeft = Math.Max(I – a[i], 0);
   idxRight = Math.Min(I + (a[i] + 1),
                       N);
   Dp[idxLeft] = Math.Max(dp[idxLeft],
                          idxRight);
}
// Stores count of fountains
// needed to be activated
Int cntfount = 1;
// Stores index of next
// fountain that needed
// to be activated
Int idxNext = 0;
idxRight = dp[0];
// Traverse []dp array
For (int I = 0; I < N; i++)
{
   idxNext = Math.Max(idxNext, dp[i]);
   // If left most fountain
   // cover all its range
   If (I == idxRight)
```

```
            If (I == idxRight)
            {
                Cntfount++;
                idxRight = idxNext;
            }
        }
        Return cntfount;
    }
    // Driver Code
    Public static void Main(String[] args)
    {
        Int[] a = { 1, 2, 1 };
        Int N = a.Length;
        Console.Write(minCntFoun(a, N));
    }
}
```

Output:

1

Circuit diagram:



Applications:

1.Public Spaces: They can be installed in parks, plazas, and other public areas to provide clean drinking water and reduce plastic waste from disposable bottles.

2.Schools and Universities: Smart fountains promote hydration on campuses and can track water consumption for health and sustainability initiatives.

3.Offices: In corporate settings, they encourage employees to stay hydrated and can integrate with wellness programs.

4.Healthcare Facilities: Hospitals and clinics can use them to ensure patients, visitors, and staff have access to safe drinking water.