PROJECT SUBMITTED BY

Name :S.Deepan

Reg.no:713921106009

Email ID:srdeepan2110@gmail.com

Nm Id :au713921106009

College code:7139

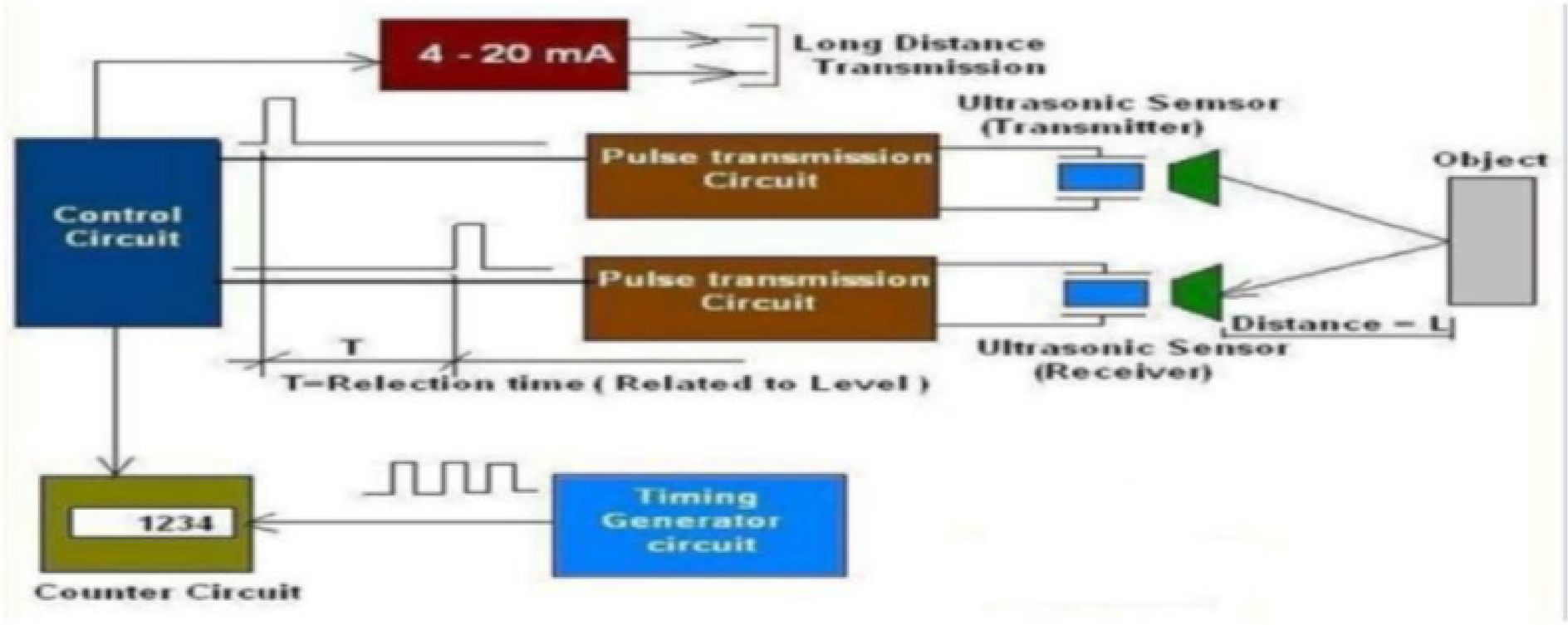
Iot based smart water management

MAJOR COMPONENT SPICTURE USED IN THESE PROJECT:

1. RASPBERRY PI:

2. ULTRASONIC SENSOR:

ULTRASONIC TRANSMITTER FUNCTIONAL BLOCK DIAGRAM



3.RELAY:

Features of 5-Pin 5V Relay:

Trigger Voltage (Voltage across coil) : 5V DC

Trigger Current (Nominal current) : 70mA

Maximum AC load current: 10A@ 250/125V AC

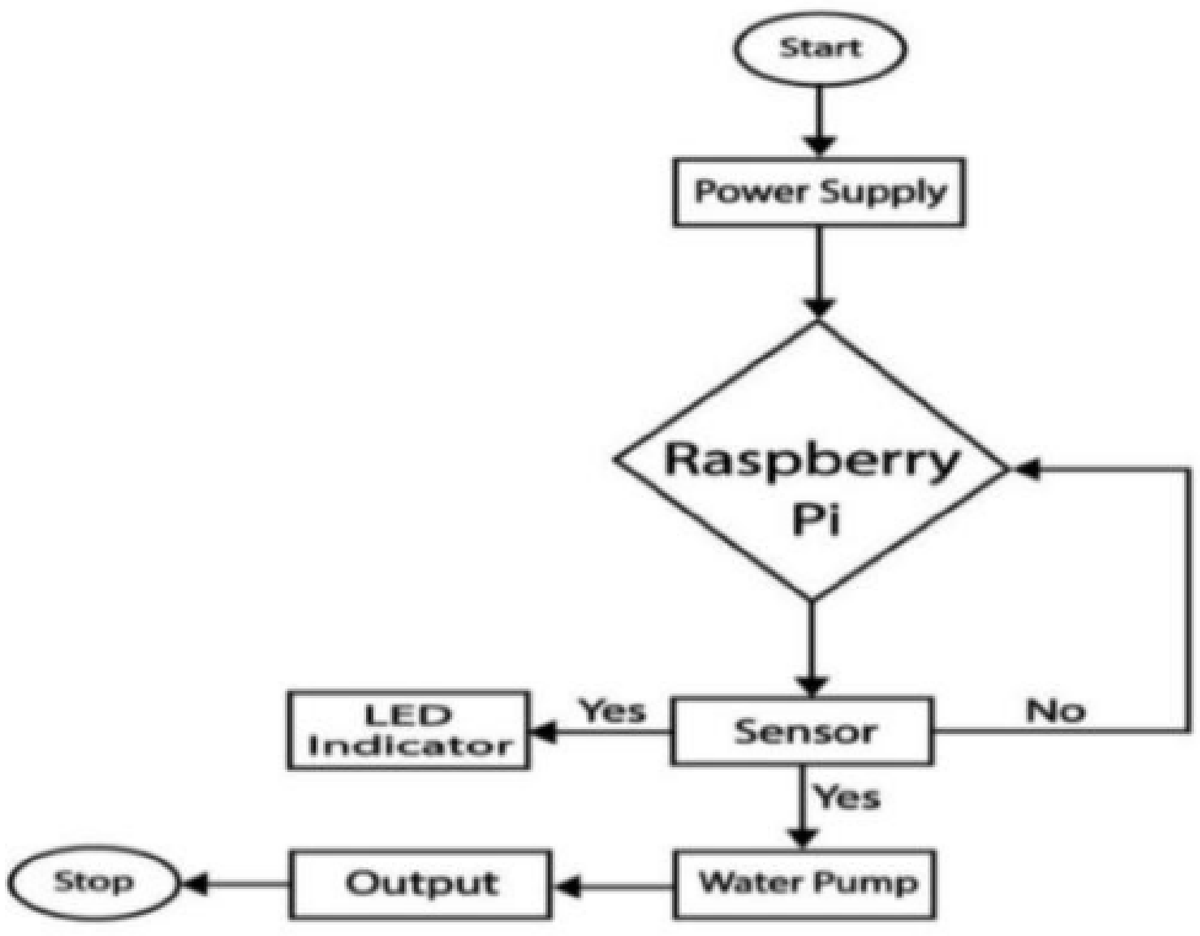
Maximum DC load current: 10A@ 30/28V DC

Compact 5-pin configuration with plastic moulding

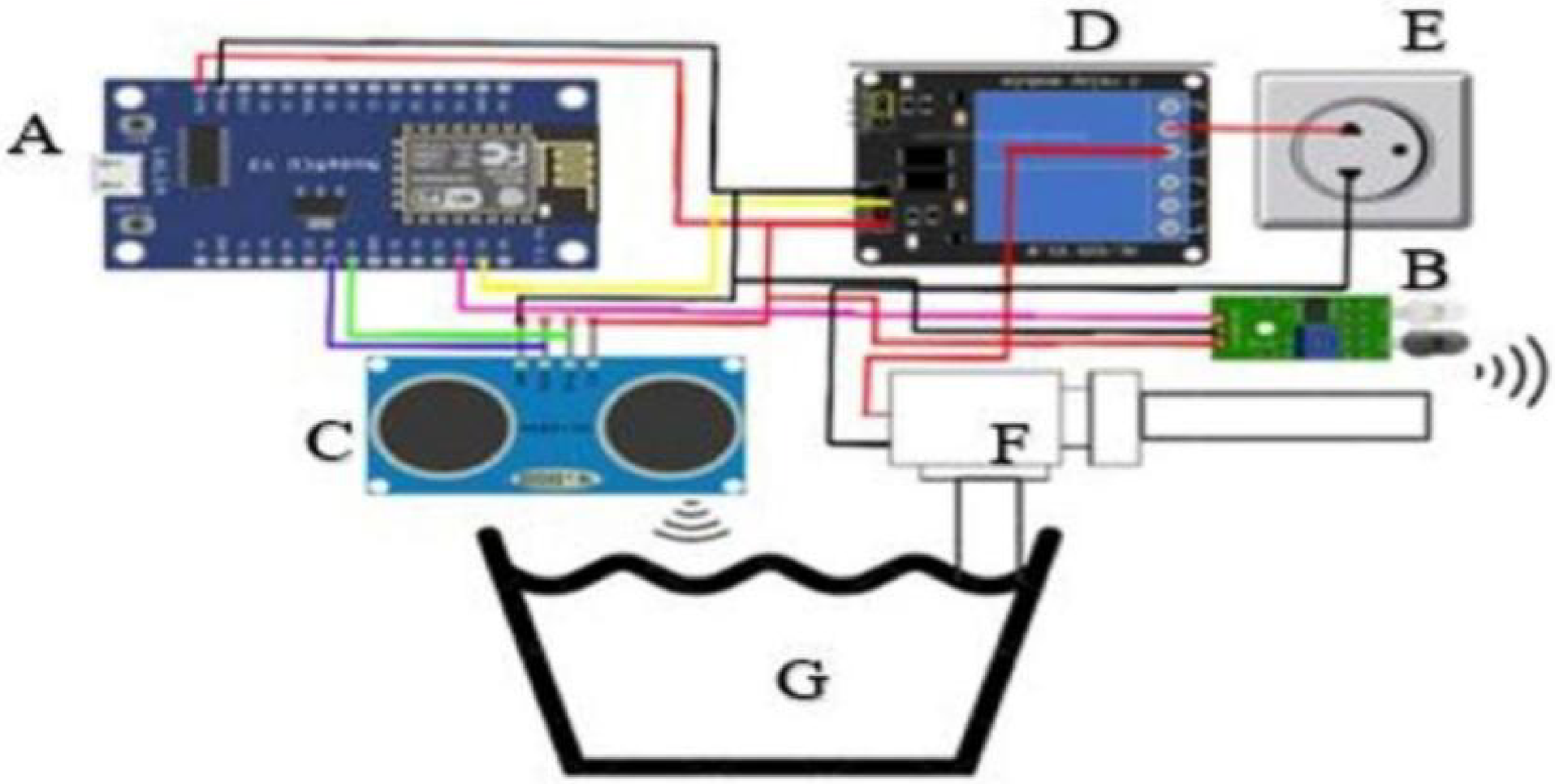
Operating time: 10 msec Release time: 5 msec

Maximum switching: 300 operating/ minute (mechanically)

FLOWCHART FOR SMART FAUCET SYSTEM:



OVERALL SERIES OF TOOLS:



WORKING OF THE SYSTEM:

The Concept behind the Smart Tap System is very simple. I will use a HCSR04 Ultrasonic Sensor to check if any object such that the glass is placed before the tap (dispenser). A solenoid Valve will be used to control the flow of water, which is when energised the water will flow out And when de energised the water will be stopped. So I will write a python program which always Checks if any object is placed near the tap, if yes then the solenoid will be turned on and wait till The object is removed, once the object is removed the solenoid will turn off automatically thus Closing the supply of water. The solenoid valve used in this project is a 12V valve with a Maximum current rating of 1.2A and a continuous current rating of 700mA. That is when the Valve is turned on it will consume about 700mA to keep the valve turned on. As we know an Arduino is a Development board which operates with 5V and hence we need a switching driver Circuit for the Solenoid to turn it on and off. The Ultrasonic Sensor is powered by the +5V and Ground pins of the GPIO pin. The Echo and Trigger pin is connected to the GPIO pin 8 and pin 9 Respectively. I can then program the Raspberry pi to use the

Ultrasonic sensor to measure the Distance and turn on the MOSFET when an object is detect. The whole circuit is simple and Hence can be easily build on top of a breadboard. Mine looked something like this below after Making the connections.

Web development:

Designing a website platform to receive and display water consumption data from Io T sensors for an Io F based smart public restroom is a significant project. Below is a high level example of how you might structure the website using Python with a web framework like Flask. This example focuses on a simple web interface for displaying water consumption data and promoting water conservation efforts.

Requirements:

Python (with Flask)

HTML, CSS

JavaScript (for interactive elements)

Io T data source (simulated for this example)

A database (SQLite for simplicity)

Charts.js library for data visualization (optional)

Here' s a simplified example of such a website:

Python (Flask) Backend:

```
From flask import Flask, render_template, request, jsonify
From flask_sqlalchemy import SQLAlchemy
App = Flask(__name__)
App.config[' SQLALCHEMY_DATABASE_URI ']=' sqlite:///water_data.db'
Db = SQLAlchemy(app)
Class WaterConsumption(db.Model):
    Id= db.Column(db.Integer, primary_key= True)
    Timestamp = db.Column(db.String(50) )
```

```
    Consumption= db.Column(db.Float)

@app.route(‘/’)

Def index():

    Data= WaterConsumption.query.all()

    Return render_template(‘index.html’, data= data)

@app.route(‘/api/water_data’, methods=[‘POST’])

Def receive_water_data():

    Data= request.get_json()

    Timestamp= data[‘timestamp’]

    Consumption= data[‘consumption’]

    New_entry= WaterConsumption(timestamp= timestamp, consumption= consumption)

    Db.session.add(new_entry)

    Db.session.commit()

    Return jsonify({“message”: “Data received”})

If __name__ == ‘__main__’:

    Db.create_all()

    App.run(debug= True)
```

HTML (index.html) Template:

```
<!DOCTYPE html>

<html>

<head>

    <title> Water Consumption Data</ title>

</head>

<body>

    <h1> Water Consumption Data for Smart Public Restroom</ h1>

    <table>

        <tr>
```

```
        <th>Timestamp</th>
        <th>Consumption (liters)</th>
    </tr>
    {% for entry in data %}
    <tr>
        <td>{{ entry.timestamp }}</td>
        <td>{{ entry.consumption }}</td>
    </tr>
    {% endfor %}
</table>
<div id="promotion">
    <h2>Water Conservation Efforts</h2>
    <!-- Add information and tips for water conservation -->
</div>
</body>
</html>
```

This example is simplified and provides a basic web interface to display water consumption data received from IoT sensors. The website allows you to visualize data and provide information and tips for water conservation efforts.

In a real-world scenario, you would need to consider the security of the data transmission, incorporate data visualization libraries like Charts.js for graphical representation, and add more extensive content and features to promote water conservation in public restrooms.