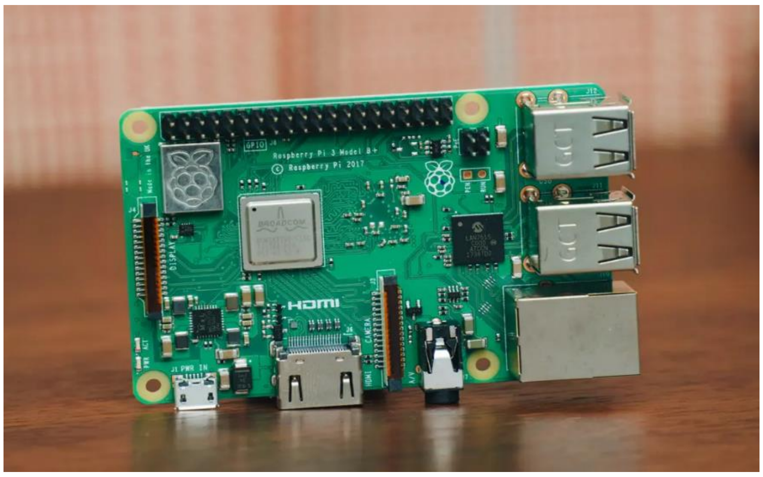
Project Title: Smart Water Fountain

RASPBERRY PI-3



INTRODUCTION:

Creating a smart water fountain IoT project using a Raspberry Pi 3 can be a fun and educational project. This project will allow you to remotely control and monitor a water fountain using the Raspberry Pi 3 and IoT technologies. Here's a step-by-step guide to get you started: Creating a smart water fountain IoT project using a Raspberry Pi 3 can be a fun and educational project. This project will allow you to remotely control and monitor a water fountain using the Raspberry Pi 3 and IoT technologies. Here's a step-by-step guide to get you started:

Components Needed:

1. Raspberry Pi 3 (or a newer model)
2. MicroSD card with Raspbian OS installed
3. Water pump
4. Water reservoir
5. Relay module
6. Ultrasonic distance sensor (HC-SR04)
7. Tubing and nozzle for the fountain
8. Jumper wires
9. Power supply for the pump
10. WiFi dongle (if your Raspberry Pi doesn't have built-in WiFi)
11. IoT platform (e.g., ThingSpeak, AWS IoT, or Google Cloud IoT)

12. Sensors for environmental data (optional)

Step 1: Set Up Your Raspberry Pi

1. Install Raspbian OS on the microSD card and boot up your Raspberry Pi.

2. Connect the Raspberry Pi to your local Wi-Fi network using the built-in Wi-Fi or a USB Wi-Fi dongle.

3. Update and upgrade your system:

 bash

  sudo apt-get update

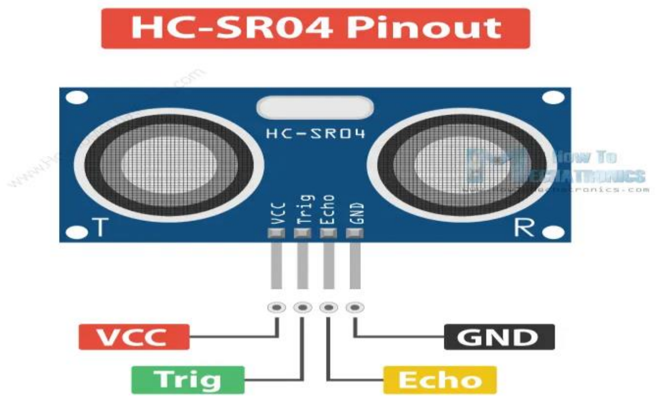
  sudo apt-get upgrade

  Step 2: Assemble the Hardware

1. Connect the water pump to the relay module. The relay module will be used to control the pump.

2. Connect the ultrasonic distance sensor to the Raspberry Pi using GPIO pins. Typically, you would connect the sensor's trigger pin to a GPIO output pin and the echo pin to a GPIO input pin.

3. Set up the water reservoir and fountain tubing. Make sure the water pump is submerged in the reservoir, and the nozzle is positioned where you want the water to flow.



Step 3: Write the Python Code

You'll need Python code to control the water pump and read data from the ultrasonic sensor. You can use libraries like RPi.GPIO for GPIO control and python-telegram-bot for remote control via Telegram.

Here's a simplified example to get you started:

python

import RPi.GPIO as GPIO

import time

from telegram.ext import Updater, CommandHandler

# Set up GPIO pins

```
GPIO.setmode(GPIO.BCM)

trigger_pin = 17

echo_pin = 18

relay_pin = 23

GPIO.setup(trigger_pin, GPIO.OUT)

GPIO.setup(echo_pin, GPIO.IN)

GPIO.setup(relay_pin, GPIO.OUT)

# Initialize the Telegram Bot (replace with your Telegram bot token)

bot_token = 'YOUR_BOT_TOKEN'

updater = Updater(token=bot_token, use_context=True)

dispatcher = updater.dispatcher

# Define command handlers (e.g., to turn on/off the fountain)

def start(update, context):

    context.bot.send_message(chat_id=update.message.chat_id, text="Welcome to the Smart
Fountain Bot!")

def turn_on(update, context):

    GPIO.output(relay_pin, GPIO.HIGH)

    context.bot.send_message(chat_id=update.message.chat_id, text="Fountain is on!")

def turn_off(update, context):

    GPIO.output(relay_pin, GPIO.LOW)

    context.bot.send_message(chat_id=update.message.chat_id, text="Fountain is off!")

# Register command handlers

dispatcher.add_handler(CommandHandler('start', start))

dispatcher.add_handler(CommandHandler('on', turn_on))

dispatcher.add_handler(CommandHandler('off', turn_off))

# Start the Telegram Bot

updater.start_polling()

# Add code for measuring distance and controlling pump based on it

try:

    while True:

        # Measure distance using the ultrasonic sensor
        # Add logic to turn on/off the pump based on the distance
        pass
except KeyboardInterrupt:
    GPIO.cleanup()
```

This is a basic template. You can extend it to add more features and control options.

Step 4: Set Up IoT Integration

To make your water fountain smart and remotely controllable, you can integrate it with an IoT platform like ThingSpeak, AWS IoT, or Google Cloud IoT. This will allow you to collect data, monitor fountain status, and even control it remotely.

For example, you can send data (such as water level, temperature, and humidity) to the IoT platform and receive commands to control the fountain through MQTT or HTTP requests.

Step 5: Test and Troubleshoot

Test your setup and make sure it works as expected. Troubleshoot any issues that may arise during testing.

Remember to ensure proper power management for the water pump and Raspberry Pi to prevent damage.

This project is a starting point, and you can expand on it by adding more sensors, features, and remote control options. It's a great way to learn about IoT, Raspberry Pi, and hardware integration.

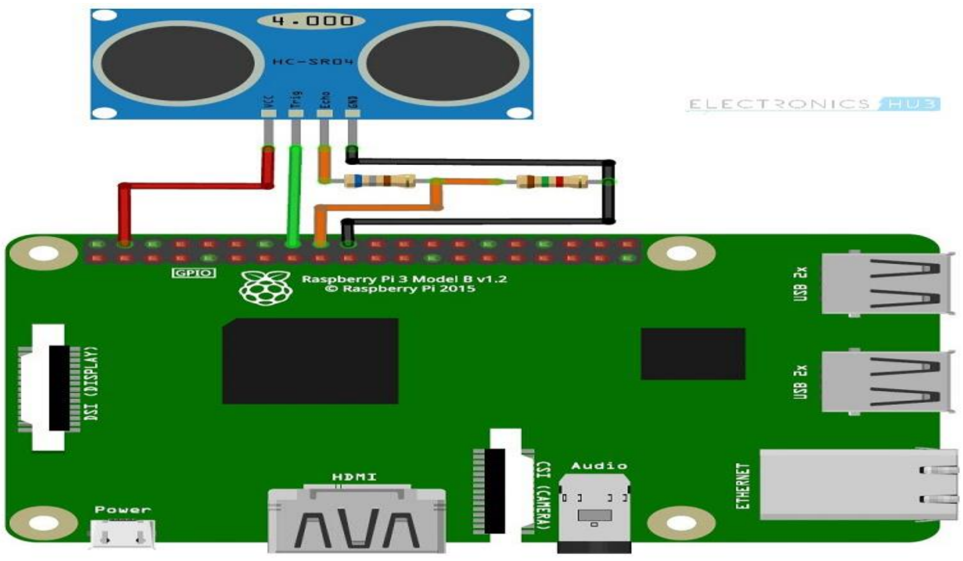
```
import RPi.GPIO as GPIO
import time
from telegram.ext import Updater, CommandHandler
# Set up GPIO pins
GPIO.setmode(GPIO.BCM)
trigger_pin = 17
echo_pin = 18
relay_pin = 23
GPIO.setup(trigger_pin, GPIO.OUT)
GPIO.setup(echo_pin, GPIO.IN)
GPIO.setup(relay_pin, GPIO.OUT)
# Initialize the Telegram Bot (replace with your Telegram bot token)
bot_token = 'YOUR_BOT_TOKEN'
updater = Updater(token=bot_token, use_context=True)
```

```
dispatcher = updater.dispatcher
# Define command handlers (e.g., to turn on/off the fountain)
def start(update, context):
    context.bot.send_message(chat_id=update.message.chat_id, text="Welcome to the Smart
Fountain Bot!")
def turn_on(update, context):
    GPIO.output(relay_pin, GPIO.HIGH)
    context.bot.send_message(chat_id=update.message.chat_id, text="Fountain is on!")
def turn_off(update, context):
    GPIO.output(relay_pin, GPIO.LOW)
    context.bot.send_message(chat_id=update.message.chat_id, text="Fountain is off!")
# Register command handlers
dispatcher.add_handler(CommandHandler('start', start))
dispatcher.add_handler(CommandHandler('on', turn_on))
dispatcher.add_handler(CommandHandler('off', turn_off))

# Start the Telegram Bot
updater.start_polling()
# Add code for measuring distance and controlling pump based on it
try:
    while True:
        # Measure distance using the ultrasonic sensor
        # Add logic to turn on/off the pump based on the distance
        pass

except KeyboardInterrupt:
    GPIO.cleanup()
   [Refill Water Tank]
```

CIRCUIT DIAGRAM:



APPLICATIONS OF SMART WATER FOUNTAIN:

Smart water fountains have a range of applications that can improve water accessibility, conservation, and user experience. Here are some common applications:

1. *Public Parks and Recreation Areas:*

   - Smart fountains in parks can provide convenient access to drinking water.

   - They can include features like bottle-filling stations, which promote reusable water bottles and reduce plastic waste.

2. *Educational Institutions:*

   - Schools, colleges, and universities can use smart fountains to provide safe and clean drinking water to students and staff.

   - These fountains can track water quality and usage, ensuring a healthy environment.

3. *Commercial Buildings:*

   - Office buildings and commercial complexes can install smart fountains for employees and visitors.

   - They can integrate with access cards for touchless dispensing.

4. *Healthcare Facilities:*

   - Hospitals and clinics can benefit from smart fountains that