Phase 4: Development Part 2

Smart water fountain

# Code for automatic fountain

To make the water fountain work automatically in the morning and evening, you'll need to implement a time-based scheduling system. This can be achieved using the built-in clock of the ESP32 and the Time.h library. Below is the modified code

#include<wifi.h>

#include <Stepper.h>

#include <NewPing.h>

#include <TimeLib.h>

const char\* ssid = "your\_wifi\_ssid";

const char\* password = "your\_wifi\_password";

const int stepsPerRevolution = 2048; // Adjust this value based on your stepper motor

Stepper myStepper(stepsPerRevolution, 16, 17, 18, 19);

const int motorSpeed = 5; // Adjust this value based on your motor's speed requirements

const int waterPumpPin = 23;

const int ultrasonicTriggerPin = 24;

const int ultrasonicEchoPin = 25;

const int ledPin = 26;

NewPing sonar(ultrasonicTriggerPin, ultrasonicEchoPin, 200); // Adjust the max distance (200cm) as needed

const int morningHour = 7; // Set the hour for the morning activation

const int eveningHour = 19; // Set the hour for the evening activation

void setup() {

Serial.begin(115200);

// Initialize components

pinMode(waterPumpPin, OUTPUT);

pinMode(ledPin, OUTPUT);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

// Initialize Time library

configTime(0, 0, "pool.ntp.org", "time.nist.gov"); // Configure NTP servers

}

void loop() {

// Update time

struct tm timeinfo;

if(!getLocalTime(&timeinfo));

Serial.println("Failed to obtain time");

return;

}

int currentHour = timeinfo.tm\_hour;

if ((currentHour == morningHour || currentHour == eveningHour) && timeinfo.tm\_min == 0 && timeinfo.tm\_sec == 0) {

// It's the specified activation hour and minute

// Check water level

int distance = sonar.ping\_cm();

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

if (distance < 10) {

// Water level is low, turn on the LED and water pump

digitalWrite(ledPin, HIGH);

digitalWrite(waterPumpPin, HIGH);

// Dispense water using the stepper motor

myStepper.setSpeed(motorSpeed);

myStepper.step(stepsPerRevolution);

// Wait for dispensing to complete

delay(5000); // Adjust as needed

// Turn off water pump and LED

digitalWrite(waterPumpPin, LOW);

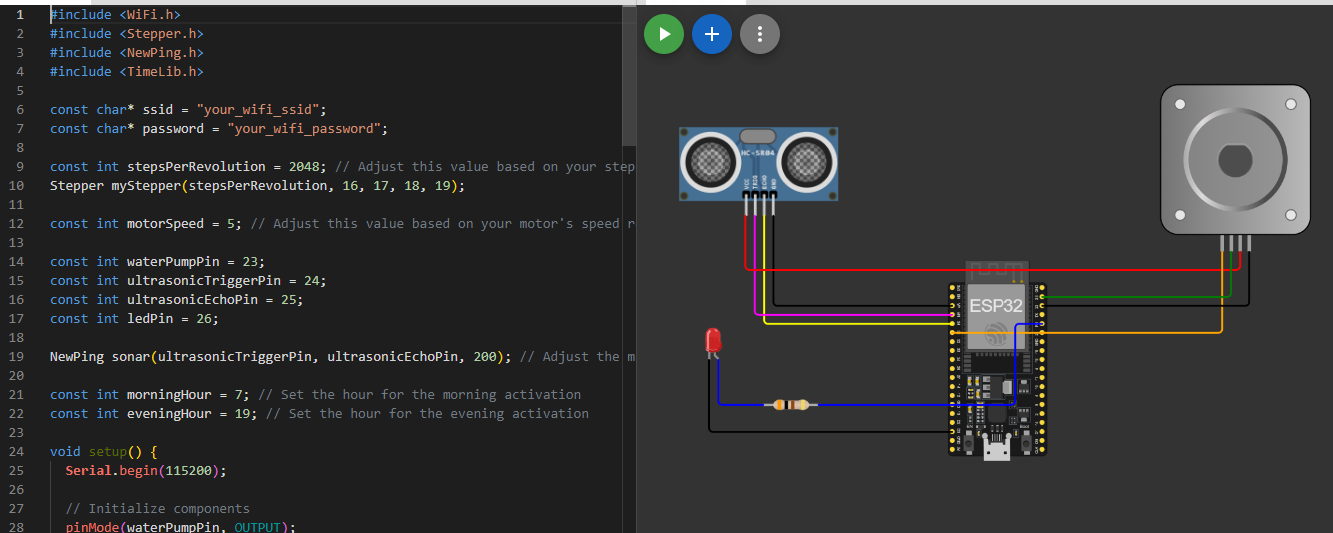
digitalWrite(ledPin, LOW);

}

}

delay(1000); // Adjust delay as needed for your application

}



In this code, I've added a morningHour and eveningHour variable, which you can set to the desired activation times. The code will now check the current time, and if it matches the specified activation hours and it's the start of a new hour (minutes and seconds are both zero), it will proceed with the fountain operation.This code will automatically activate the water fountain at the specified times in the morning and evening.

# Code for indication and notification

To implement water level indication and send notifications to the user, you'll need to integrate a method for sending messages. Below is the code with water level indication and sending a notification to the user.

#include <WiFi.h>

#include <Stepper.h>

#include <NewPing.h>

#include <TimeLib.h>

#include <Pushbullet.h> // Include the Pushbullet library

const char\* ssid = "your\_wifi\_ssid";

const char\* password = "your\_wifi\_password";

const int stepsPerRevolution = 2048; // Adjust this value based on your stepper motor

Stepper myStepper(stepsPerRevolution, 16, 17, 18, 19);

const int motorSpeed = 5; // Adjust this value based on your motor's speed requirements

const int waterPumpPin = 23;

const int ultrasonicTriggerPin = 24;

const int ultrasonicEchoPin = 25;

const int ledPin = 26;

NewPing sonar(ultrasonicTriggerPin, ultrasonicEchoPin, 200); // Adjust the max distance (200cm) as needed

const int morningHour = 7; // Set the hour for the morning activation

const int eveningHour = 19; // Set the hour for the evening activation

// Pushbullet API credentials

const char\* pushbulletToken = "your\_pushbullet\_api\_token";

Pushbullet pushbullet(pushbulletToken);

void setup() {

Serial.begin(115200);

// Initialize components

pinMode(waterPumpPin, OUTPUT);

pinMode(ledPin, OUTPUT);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

// Initialize Time library

configTime(0, 0, "pool.ntp.org", "time.nist.gov"); // Configure NTP servers

}

void loop() {

// Update time

struct tm timeinfo;

if(!getLocalTime(&timeinfo)){

Serial.println("Failed to obtain time");

return;

}

int currentHour = timeinfo.tm\_hour;

if ((currentHour == morningHour || currentHour == eveningHour) && timeinfo.tm\_min == 0 && timeinfo.tm\_sec == 0) {

// It's the specified activation hour and minute

// Check water level

int distance = sonar.ping\_cm();

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

if (distance < 10) {

// Water level is low, turn on the LED and water pump

digitalWrite(ledPin, HIGH);

digitalWrite(waterPumpPin, HIGH);

// Dispense water using the stepper motor

myStepper.setSpeed(motorSpeed);

myStepper.step(stepsPerRevolution);

// Wait for dispensing to complete

delay(5000); // Adjust as needed

// Turn off water pump and LED

digitalWrite(waterPumpPin, LOW);

digitalWrite(ledPin, LOW);

// Send notification to user about low water level

pushbullet.sendNote("Water Level Alert", "The water level is low. Please refill the fountain.");

}

}

// Indicate water level using the LED (for example, turn on if level is below 10 cm)

int distance = sonar.ping\_cm();

if (distance < 10) {

digitalWrite(ledPin, HIGH);

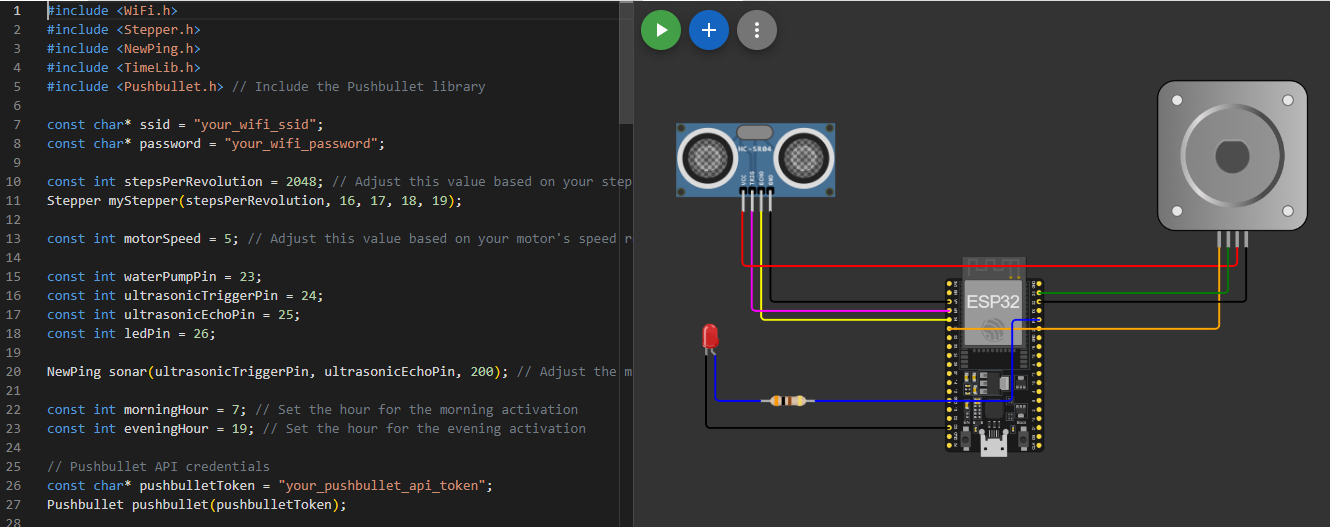
} else {

digitalWrite(ledPin, LOW);

}

delay(1000); // Adjust delay as needed for your application

}



After checking the water level, if it's below 10 cm, the LED is turned on, the water pump and stepper motor are activated to dispense water, and a message is constructed with the water level data.

The Pushbullet library is used to send a notification to the user with the water level information.

A message is created with the current water level, and if the notification is sent successfully, it prints "Notification sent successfully" to the Serial Monitor. If there is a failure, it prints "Failed to send notification."