Smart Pet Fountain

Tiancheng Lin
Santa Clara University
Santa Clara, CA
500 El Camino Real

Yujie Zhu
Santa Clara University
Santa Clara, CA
500 El Camino Real

Ziyang Wang
Santa Clara University
Santa Clara, CA
500 El Camino Real

***Abstract*—This Project is a fountain-type automatic cat water dispenser. We will use amazon ec2 to build a server, through this server, we will interact with the cat water dispenser to realize daily monitoring of vital signs, health status, water quality, and remote start and stop of the water dispenser.**

Keywords—IOT, EC2

I. Introduction

As a smart water dispenser, our product can be regarded as a part of the Internet of Things introduced in this course. We want to develop a pet water dispenser with network connection properties, and add water quality monitoring sensors, water level sensors, motion recognition sensors, etc. to it. On this product, it will include sensors, microcontroller, communication unit and power supply. It's battery powered, it's small and cheap, and it's not designed to perform complex tasks, all of which are the definitions of things in the Internet of Things that our product basically fits into.

As an IoT device, we need to let it interact with the IoT platform, and then interact with the IoT server. To interact with the hardware, we'll build a web server. On the server, we will display the current water quality and water level information in real time, and design a remote switch to allow users to control the water dispenser and understand the current status of the cat when they are not at home. We will design a restful interface, allowing the hardware device to upload data using put, and analyze and obtain the information of the cat's vital signs on the server side.

II. Project Objective

A. Purpose

As we know, cats are more interested in flowing water than steady water, so a pet fountain is good for cat. Since most of the water fountain can't be controlled remotely, we want to design product that can be switched on/off remotely and see the water level right now, in case the owner is not at home.

B. Background

1) Rational for the hardware and software architecture

For hardware, we are using 3D printing to get the fountain base, and we used channel relay module to power the water pump, and a water level sensor to get current water level.

For software, we are using EC2 to build the server. Javascript is used for client-side scripting, Python is used for server-side scripting. Everything front end is written in html5.

2) Related products

Most of the cat water fountain is powered by a fixed power source, and can not be switched on/off remotely. Also, yo can only see the water level when you are near the product. The figure below is one of the typical pet fountain.

III. Design

A. Our solution

Our solution consists of two parts:

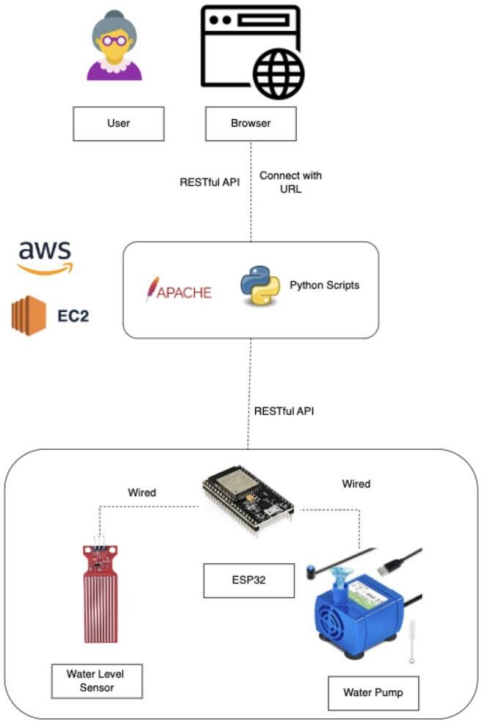
- A smart IoT device that can:
  - Be switched on/off remotely to feed pets
  - Record and post water level information periodically
- A web portal that allow user to:
  - Control pet water fountain switch from anywhere
  - Check water level history

The Iot Device connects with the web server using RESTful API (see Software Design section for detailed API information). It periodically gets the current status of the switch from the web server, and turns on/off the pump accordingly. Also it sends its current water level to the web server.

The web server provides API for iot to send over water level data, and stores them in sqlite3 database. It also provides API for iot device to query its current status (on/off) set by the user at web portal UI.

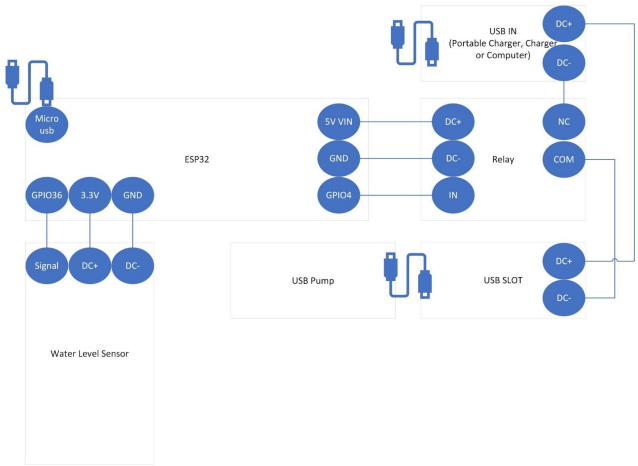
The web portal supports user authentication, user registration, user login, allows user to switch on/off the iot device and provides user a visualized data analytics chart for the water level variation history of the iot device.

The figure below is our high level architecture.



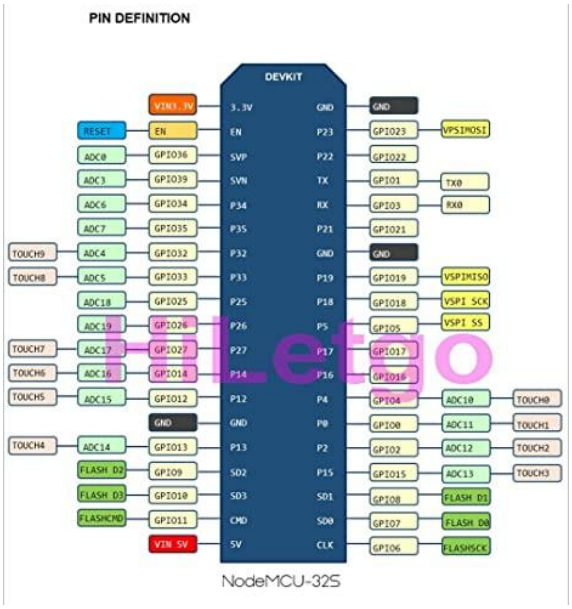
B. *Hardware Design*

We need an ESP32 board, a relay module, a water level sensor, a USB slot, a USB pump, and a 3D printed water tank to implement our design. This circuit diagram illustrates the details of our design.

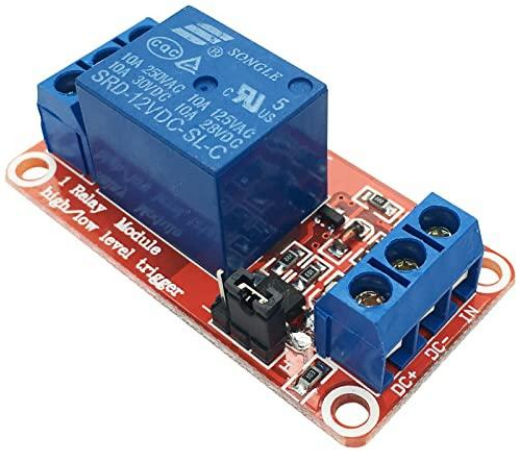


We will briefly introduce the electronic components we used in the following paragraphs.

First, we need a HiLetgo ESP-WROOM-32 development board and install Arduino 2.0 IDE. You may have noticed two development boards in the picture. This is because one of the ESP32 boards failed for unknown reasons and we were unable to remove it from the breadboard. We used the ESP32 built-in WIFI to connect the remote server through the Internet. Besides, we used the ESPAsyncWebServer library to provide a basic local web server. This provides us with failsafe options when we can't get access to the Internet. The following picture shows the pin definition of the ESP32 board. We used GPIO 36 to receive analog voltage and GPIO 4 to control the relay module.



Secondly, we need a relay module. The relay module needs a 5V power supply, so we used the VIN pin and GND pin of the ESP32 board to supply power. The other side of the relay module comprises COM pin, NC pin, NO pin where we connect the COM pin to the GND pin of the USB power supply and the NC pin to the GPIO 4 pin of the ESP32 board. We need to write high to GPIO 4, so the pump can get a power supply.

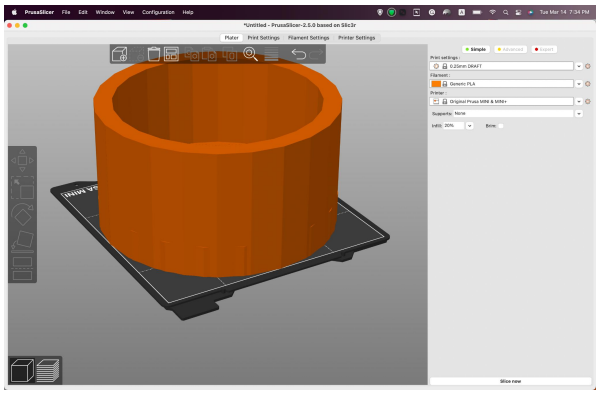


Thirdly, we need a water level sensor module. This module needs 3.3V power supply, so we connected the VIN 3.3V pin of the ESP32 board to the DC+ pin of the sensor, the GND pin of the ESP32 board to the DC- pin of the sensor, and GPIO 4 pin of the ESP32 board to the signal pin of the sensor. We measure the analog voltage of the sensor every two seconds, convert them into actual water-level values, and then report them to the remote server using RESTful style APIs.

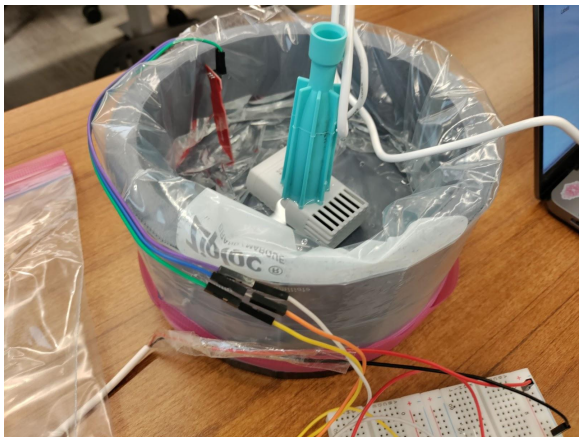
The following figure is Songhe rain water level sensor.

Fourthly, we need a USB pump and a 3D printed water tank. To prevent water leakage, we placed a plastic bag inside the water tank. We also need to connect the USB pump to the USB slot on the breadboard.

Finally, we 3D printed the fountain base, and the design graph is as follow.



The following figure is finished product.



C. *Software Architecture*

1) *Firmware design*

Firmware: Mac book pro
OS: macOS

2) *Front End Design and Implementation*

The frontend of this application consists of a few HTML files with CSS and Javascript embedded in it. Please see the following picture for the entire file architecture of the web portal application (frontend and backend).

Essentially, we have four html files responsible for different web pages that we support - user login page, user registration page, active user dashboard page and inactive user page.

In each html file, CSS is embedded for better visual effects and Javascript code is embedded to do some simple scripting including charts generating, calling backend API for user authentication and user registration.

3) *Back End Design and Implementation*

The domain for our web portal is https://iot.yujiezhu.net

The Apache2 server is configured to forward any request to this domain to the wsgi.py (see file structure figure from the above section) script. Any http request is redirected to https request with a HTTP code 301. [1]

wsgi.py is the entry point of the entire backend program. Since this project requires us not to use any framework to do routing, we configured Apache2 and created wsgi.py following PEP 333 standard [2] and mod_wsgi documentation [3] to connect Apache2 and our backend.

Apart from the entrypoint wsgi.py, the backend consists of three Python modules - api module, web module and db module.

The db module consists of a sqlite3 database file which stores all the data we have (see the following section for DB schema) and a controller that can be invoked by any other scripts to perform CRUD in DB.

The web module consists of a few controllers to respond to different web page requests. The controller will read the html file into memory, fetch related data from DB, inject data into html and send it back to wsgi.py. Then wsgi.py will encode the response (from normal string to bytes) and send back to Apache2 for response.

The api module consists of a few api controllers that take requests from both iot device and client-side JS scripts to perform DB CRUD. The response is either empty or a JSON format body.

4) *Exposed API list*

The web portal is configured to match and respond to multiple URL requests. Some of them are APIs from either the iot

device or from Javascript requests that are embedded in the HTML file. Others are URLs that users directly type in their browser or are being redirected to access web portal web pages.

I. APIs for IoT device to call:
https://iot.yujiezhu.net/api/water-level
https://iot.yujiezhu.net/api/status

II. APIs for client-side Javascript to call:
https://iot.yujiezhu.net/
https://iot.yujiezhu.net/web/register
https://iot.yujiezhu.net/web/login
https://iot.yujiezhu.net/web/dashboard

III. URLs for user to request web page:
https://iot.yujiezhu.net/api/new-user
https://iot.yujiezhu.net/api/auth
https://iot.yujiezhu.net/api/on
https://iot.yujiezhu.net/api/off

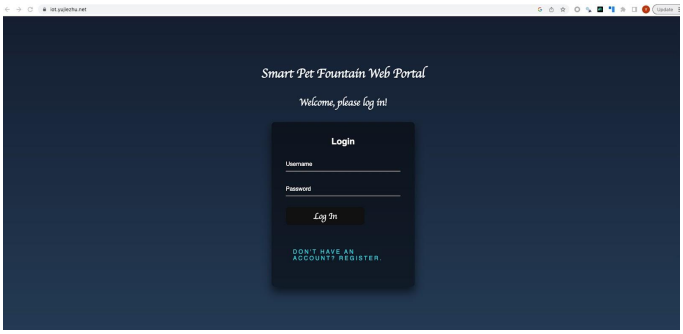
D. Database Schema

```
CREATE TABLE Usr(
        UID INTEGER PRIMARY KEY AUTOINCREMENT,
        username VARCHAR(255) NOT NULL,
        password VARCHAR(255) NOT NULL,
        activation INTEGER NOT NULL);

CREATE TABLE Level(
        LID INTEGER PRIMARY KEY AUTOINCREMENT,
        level VARCHAR(255) NOT NULL,
        time VARCHAR(255) NOT NULL,
        UID INTEGER,
        FOREIGN KEY (UID) REFERENCES Usr(UID) ON DELETE CASCADE
        );

CREATE TABLE Status(
        SID INTEGER PRIMARY KEY AUTOINCREMENT,
        status VARCHAR(255) NOT NULL,
        UID INTEGER,
        FOREIGN KEY (UID) REFERENCES Usr(UID) ON DELETE CASCADE
        );
```

E. User Interface

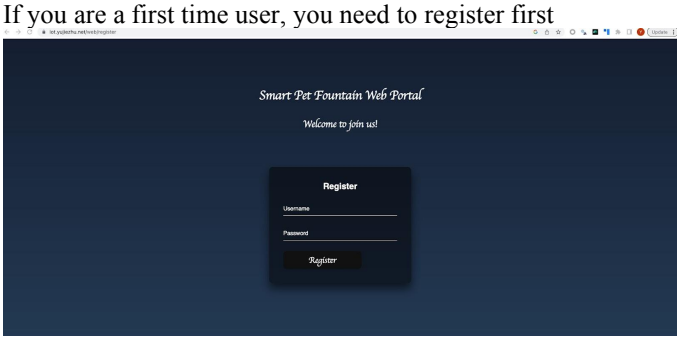
➢ Login page

In this page, you can login if you are a certified active user of our product.



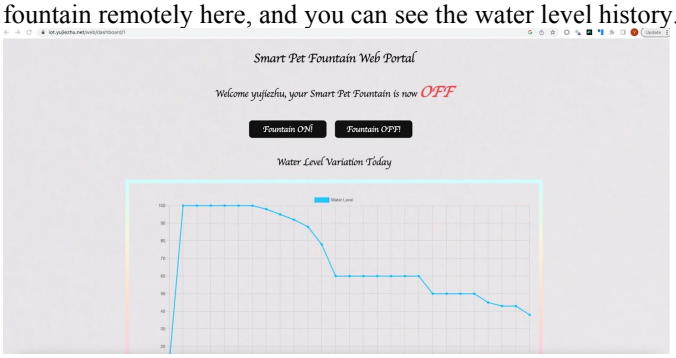
➢ Register page

If you are a first time user, you need to register first



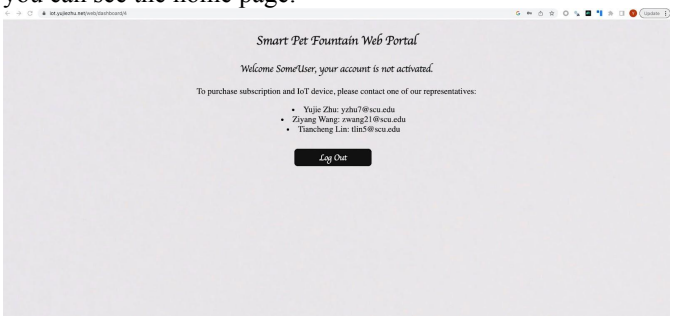
➢ Home page for user

This is the home page for certified user, you can control the fountain remotely here, and you can see the water level history.



➢ New user page

If you are a new user, you need to buy our service first so that you can see the home page.



IV. RESULTS

In this project, we finally get an IOT device that has following functions:

1) This dispenser can upload water level data to servers so that we can monitor when and how much the cat drinks water. It can provide essential data to vet when it is needed.
2) We build a front-end website to draw relative diagrams to get intuitive data.
3) We can control how much water the cat can drink even if we don't live at home because this dispenser can actively shut down or start when needed using websites.
4) This water dispenser saves energy to protect the environment because it only works when needed.
5) This dispenser uses a USB slot to power. It's very safe because the voltage of the USB slot is only 5V. Besides, we can use a charger, a computer, or a power bank to power it up.
6) This dispenser has a 3D-printed case containing more than 0.5-liter water. It is made of PLA, an environment-friendly material.

Our water dispensers can basically reach the top 3 of the 4 grades of IOT. It can collect information passively, or it can be actively turned off or turned on when needed. It is also possible to react proactively based on calculations, such as starting and stopping based on motion recognition sensors. but cannot make independent judgments.

This product faced many challenges, such as lifespan, scalability, maintainability, programmability, etc., but we all worked it out tenaciously.

V. References

[1] https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-ubuntu-20-04

[2] https://peps.python.org/pep-0333/#environ-variables

[3]https://modwsgi.readthedocs.io/en/develop/user-guides/quick-configuration-guide.html

[4] https://github.com/me-no-dev/ESPAsyncWebServer

VI. Appendices

- Hardware Inventory:

Songhe Rain Water Level Sensor * 1

10Gtek# 5V 1-Channel Relay Module * 1

10Gtek# USB-A Male to DIP Breakout Board Adapter * 1

Water Pump * 1

3D Printed Water Tank * 1

HiLetgo ESP32 Development Board * 1

USB Power Supply Line * 1

- Software Inventory:

Frontend: HTML5, CSS, JavaScript

Backend: Apache2 (added mod_wsgi package), Python, Sqlite3

No frameworks used

- Mobile App:

There is no special development for mobile devices. To access a web portal on a mobile device, simply use any browser on the device and go to https://iot.yujiezhu.net. UI is designed to work the best on laptop screens but has some optimizations for mobile screen so it is also usable on mobile devices.

- Execution Instructions:

Hardware Setup:

We need to connect the following pins according to the following table.

Components	Pins	Components	Pins
ESP32	GPIO 36	Water level sensor	S
ESP32	VIN 3.3V	Water level sensor	DC+
ESP32	GND	Water level sensor	DC-
ESP32	GPIO 4	Relay module	IN
ESP32	VIN 5V	Relay module	DC+
ESP32	GND	Relay module	DC-
USB power supply	GND	Relay module	NC
USB output slot	GND	Relay module	COM
USB power supply	VBUS	USB output slot	VBUS

Then we should connect the USB pump to the USB output slot, put the water level sensor in the water tank, upload the firmware to the ESP32 board, and connect the micro USB slot of the ESP32 board with the power supply.

- Frontend:

Front-end are just a few static files that do not need to be executed. They will be read by python scripts into memory and inject data into.

- Backend:

To execute, create a new directory on a server and put everything from web-code (including in the zip) into that directory.

An example path would be /var/www/iot.domain.com/

This directory would include wsgi.py, api module, web module, db module

Then we need to configure Apache2 to link to the backend script.

In Apache2 configuration file, add the following two lines:

ServerName iot.domian.com

WSGIScriptAlias /var/www/iot.domain.com/wsgi.py

Since the backend is essentially a bunch of scripts, we don't need to start any service in Linux to make it work. Every request comes in, Apache2 will invoke wsgi.py to pass over the request.