SmartGarden

An Internet of Things Senior Project
by Matt Lindly and Sam Lees

A 2017 Senior Project in conjunction with "Introduction to Internet of Things (CPE 458)" taught by David Maluf, Raghuram Sudhaakar, and Foaad Khosmood, a partnership between Cisco Systems, Inc. and California Polytechnic State University, San Luis Obispo in Fall 2016.

Significant structural modifications, performance improvements, and additional features were added as part of this product's continuation as a senior project. The work submitted for Senior Project credit is both a significant enhancement of the original code and a functionally superior product with substantial additional features.

The following documentation is updated to reflect the project's latest release.

Background

The problem that the SmartGarden project seeks to solve is the challenge of watering succulents correctly and maintaining consistent plant health. Too much water can cause root rot and drowning while too little water can cause dehydration and starvation.
The SmartGarden solves this common problem by configuring a Raspberry Pi to measure soil moisture, store the data to an AWS server in the cloud, and water the succulent when necessary, informing the user by providing both a web-interface and a variety of notifications.

Preliminary Project Proposal (PDF)

Phase 1 Presentation (PDF)

Functional Specifications

Some similar projects include the Parrot Flower Power, the Edyn Smart Garden Sensor, the GreenIQ Smart Garden Hub, and FarmBot. Research identified some strengths of these existing products; however, many of them are cost prohibitive or limit data accessibility to mobile applications, and none focus on succulents.

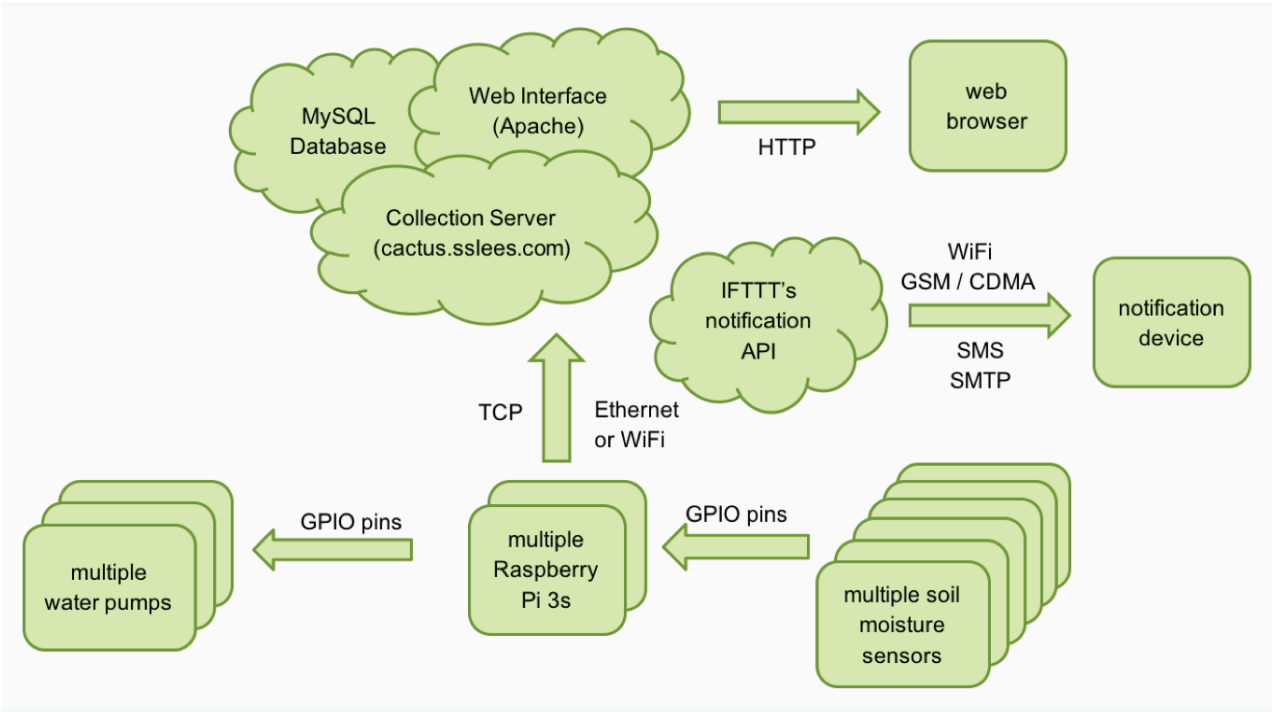
Requirement	Engineering Specification	Detailed Description
1) Moisture Sensor	YL-69 Sensor	Measures analog value of current moisture
2) Water Pump	RobotGeek Pumping Station	Pumps water from a nearby reservoir
3) Edge Node	Raspberry Pi 3	Provides a small form factor at the edge
4) Communication	TCP Protocol	Connects the client to the sever
5) Data Storage	MySQL	Stores historically collected data
6) DNS Resolution	No-IP	Streamlines GUI access
7) Presentation	Apache 2 Web Server	Serves GUI content as HTML/CSS/PHP/JS
8) Graphics	Google Charts	Presentation layer for data display
9) GUI Interaction	HTML Forms	Configures graph based on post requests
10) Notifications	IFTTT Integration	Sends various notifications to user

Technical Details

Some challenges of this project included handling the analog-to-digital conversion. This was overcome by purchasing a MCP3008 ADC to convert the output of the sensor and provide the Pi with compatible input. Designing a database schema also proved to be a challenge. The final solution utilizes a MySQL database for all data storage. Finally, ensuring data integrity and performing analytics on the data collected created the need for a few algorithms, such as a simple rolling average to reduce sensor noise.

Technical Implementation

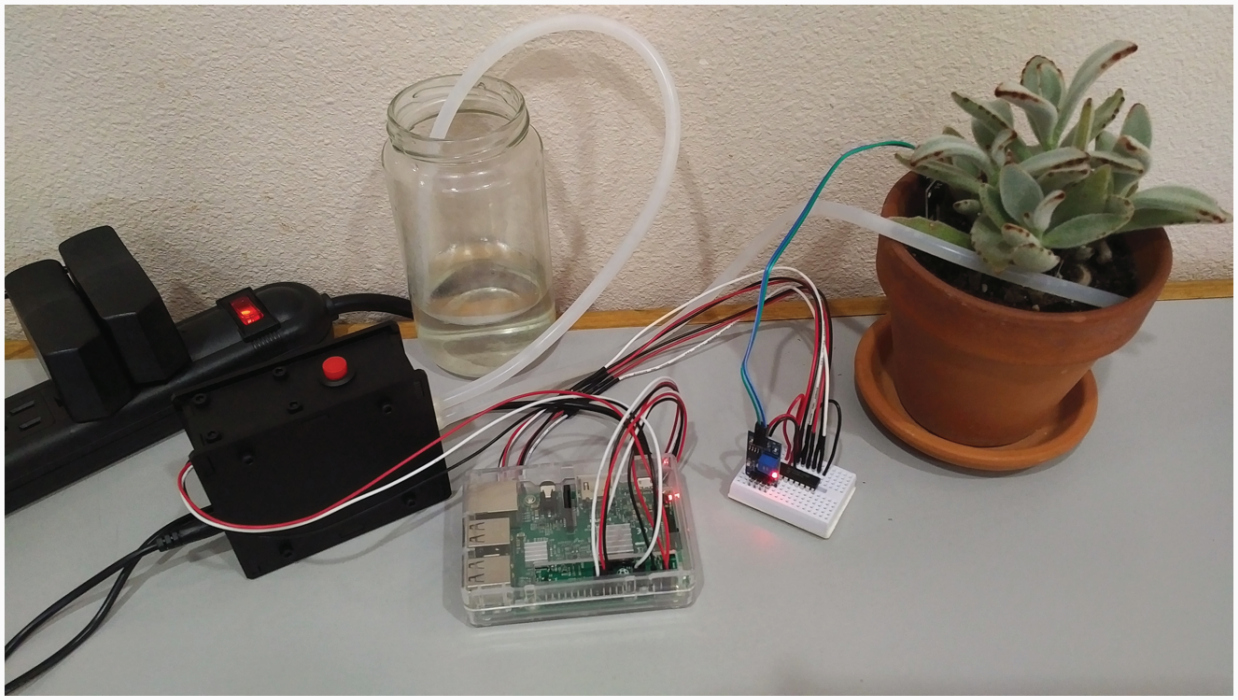
The project naturally lent itself to a two-phase approach. The first iteration of the project involved developing a solution to run on the node and provide sensor readings to the server, storing the data on a database and performing analytics to provide an accurate user-interface, automated watering, and notification platform. Once this phase was complete, the project was expanded to enable automated watering and accommodate multiple sensors at each node. These enhancements required extensive revision to the User Interface to allow for multi-user authentication.

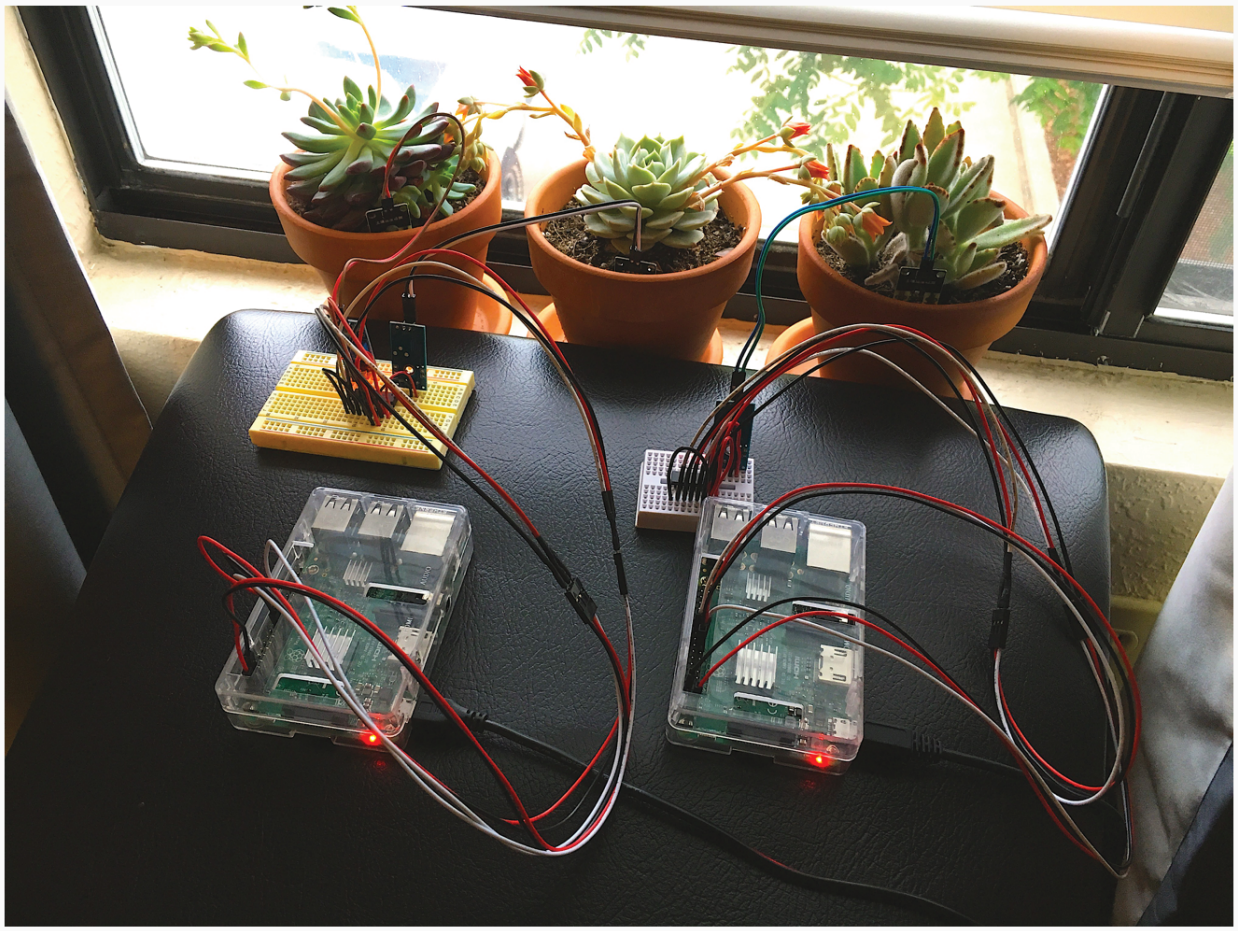


Demonstration

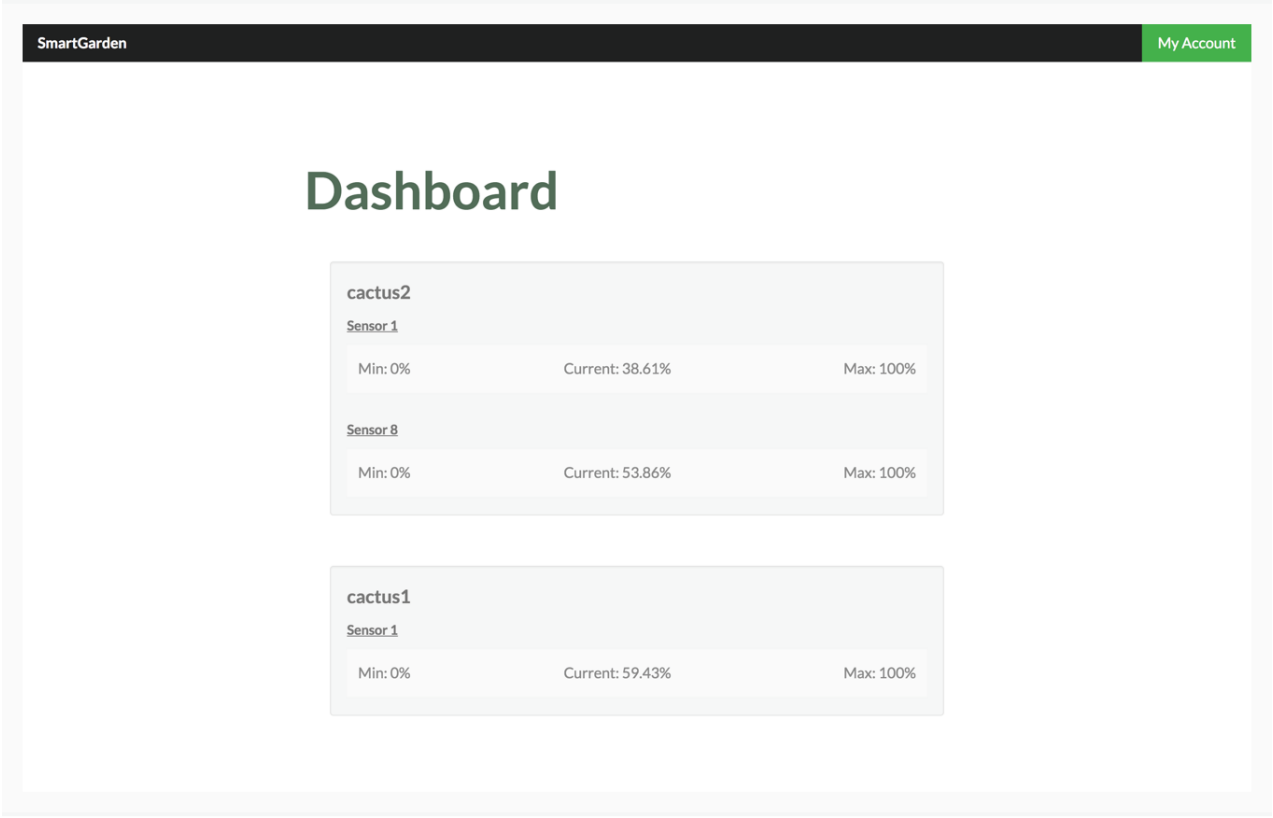
A successful demonstration of the product showcases the web-interface with real-time moisture gauges and graphs for historical data at various intervals. The IFTTT integration for notifications is also demonstrated showing that push notifications, emails, and phone calls are actuated when action is needed or performed on the system. The following section shows the significant hardware modifications, which provide additional features and functionality.

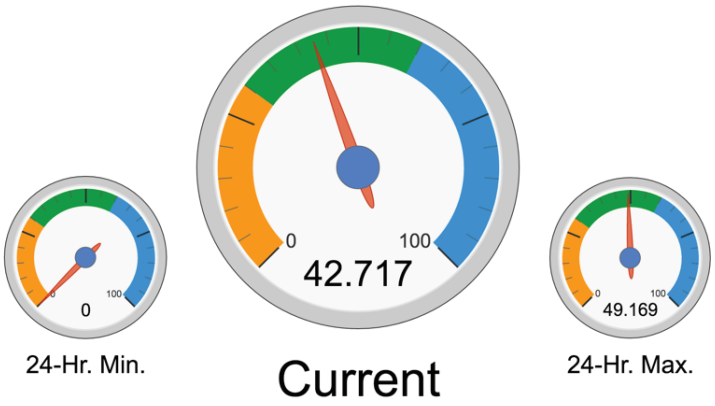
Hardware

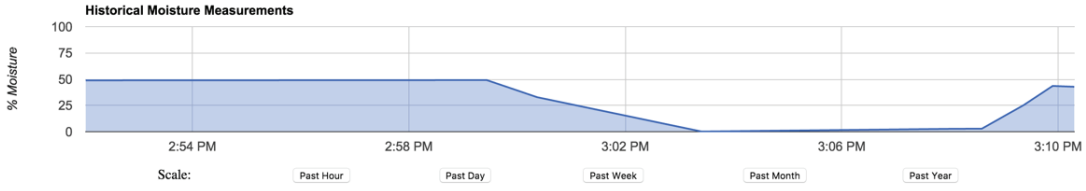




GUI


SmartGarden
My Account
Dashboard
cactus2
Sensor 1
Min: 0%
Current: 38.61%
Max: 100%
Sensor 8
Min: 0%
Current: 53.86%
Max: 100%
cactus1
Sensor 1
Min: 0%
Current: 59.43%
Max: 100%


0
100
0
42.717
0
100
49.169
24-Hr. Min.
Current
24-Hr. Max.


Historical Moisture Measurements
% Moisture
100
75
50
25
0
2:54 PM
2:58 PM
3:02 PM
3:06 PM
3:10 PM
Scale:
Past Hour
Past Day
Past Week
Past Month
Past Year

Conclusion

The latest release of the product include the implementation of enhancements beyond the scope of the original project. New features include automated watering made possible by integrating new hardware and software with the existing system to make the SmartGarden a more complete product. The system has also been scaled to accommodate up to eight sensors per node. The scaling of the existing product to multiple nodes required implementation of authentication protocols to secure each user's personal data and allow for a multi-user interface.

Presentation (PDF)

Source Code (GitHub)

Setup Guide

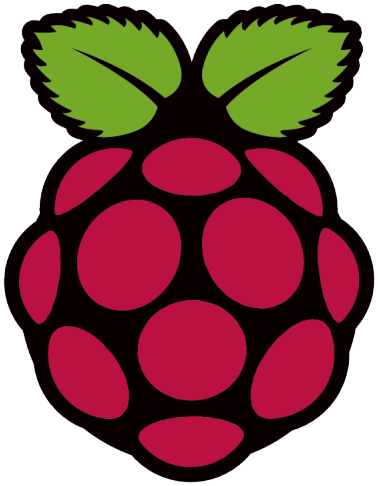
Design Analysis

References

Similar Products

- Parrot Flower Power
- Edyn Smart Garden System
- GreenIQ Smart Garden Hub
- FarmBot

Setup Guide



+

Setting up Raspberry Pi Client

Preliminary Setup

To get started, the Raspberry Pi Foundation has some excellent documentation for those new to using a Raspberry Pi.

https://www.raspberrypi.org/learning/software-guide/quickstart/

To download the latest version of Raspbian Lite (which is the software the Raspberry Pi runs on, use the link below:

https://downloads.raspberrypi.org/raspbian_lite_latest

To download the version of Raspbian we used:

http://director.downloads.raspberrypi.org/raspbian_lite/images/raspbian_lite-2017-03-03/2017-03-02-raspbian-jessie-lite.zip

```
SHA-1: 1778584c419208d919ca85e92a5cae16d1676090
```

To flash the operating system to the Raspberry Pi, we recommend using Etcher.

https://etcher.io/

SSH access will need to be enabled in order to install the software manually. A good guide on doing that can be found here:

https://www.raspberrypi.org/documentation/remote-access/ssh/README.md

If you have a keyboard and monitor, connect those and follow the first two steps in the guide above. If not, follow the third step.

Install the new SD card in your Pi and connect the Pi to power and internet (if not already done).

Wi-Fi Setup (Optional)

We recommend using an Ethernet connection, but it is possible to set up Wi-Fi. To do so, use

```
wpa_passphrase "ssid" "password"
```

to generate a wireless passkey like the following (`psk` is concealed and has been truncated in this example):

```
network={
        ssid="ssid"

        psk=0000000000000000000000000000000000000000000000000...
}
```

The output above must be added to

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

and you will need to run

```
sudo wpa_cli reconfigure
```

to reload the settings and connect to Wi-Fi.

If necessary, additional details and instructions can be found here:

https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md

Connect via SSH

SHH to the Pi. The default password should be `raspberry`.

```
ssh pi@raspberrypi.local
```

Depending on your router, you may need to use `ssh pi@<IP address>` instead.

Update the System

Run updates via:

```
sudo apt-get update &&
sudo apt-get upgrade -y &&
sudo apt-get autoremove &&
sudo reboot
```

Reconnect after reboot is finished.

System Configuration

Start the configuration tool via:

```
sudo raspi-config
```

The following lists the options that need to be changed from their default settings:

```
8 Update
1 Change User Password: c@c7u$
2 Hostname: cactus
3 Boot Options
     B1 Desktop / CLI: B1 Console
     B2 Wait for Network at Boot: Yes
4 Localisation Options
     I1 Change Locale
          [ ] en_GB.UTF-8 UTF-8
          [*] en_US.UTF-8 UTF-8
               en_US.UTF-8
     I2 Change Timezone: America: Los_Angeles
     I4 Change Wi-fi Country: US United States
5 Interfacing Options: P4 SPI: Yes

Finish: Yes (will reboot)
```

The last option (enabling SPI) is critical.

Dependency Installation

The following commands install some extra software that the client code needs to run:

```
sudo apt-get install -y python3 python3-pip
pip3 install spidev
sudo pip3 install RPi.GPIO
```

Install git with

```
sudo apt-get install -y git
```

Install Client Software

```
git clone https://github.com/sslees/cactus.git
```

Because the software is written in Python, it does not need to be compiled.

Hardware Setup

The site below has a great guide on setting up analog sensors on a Pi. It will document how to wire most of the electronic components, particularly the analog to digital converter (MCP3008).

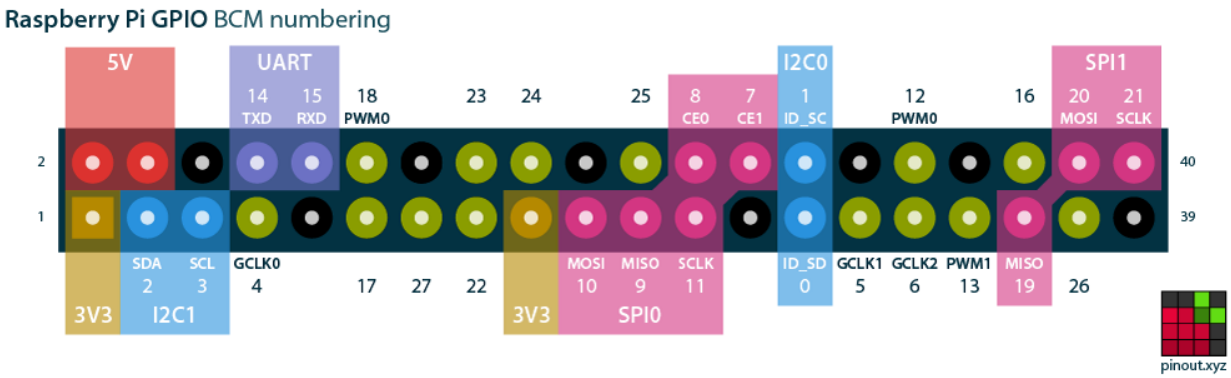
> http://www.raspberrypi-spy.co.uk/2013/10/analogue-sensors-on-the-raspberry-pi-using-an-mcp3008/

A link to the ADC is below:

> https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

The following diagram can assist with connecting everything to the Pi GPIO pins, including the pumps.

Note: The pin layout in the guide above is different than the layout in the diagram below. The diagram is consistent with the Pi version we used.



Start Client on Boot

Add

```
python3 /home/pi/cactus/client.py &
```

to

```
sudo nano /etc/rc.local
```

before

```
exit 0
```

to enable the client to start automatically at boot.

More information on startup scripts can be found here:

> https://www.raspberrypi.org/documentation/linux/usage/rc-local.md

Setting up Server (Advanced)

Note: This is only necessary if you are interested in running your own server.

For our project, we started with an Micro Instance server from Amazon Web Services.

Install the Python MySQL connector.

```
wget https://cdn.mysql.com//Downloads/Connector-
Python/mysql-connector-python-2.1.5.zip
unzip mysql-connector-python-2.1.5.zip
cd mysql-connector-python-2.1.5
sudo python3 setup.py install
```

After logging into a MySQL session, run the following commands to set create the necessary database and user:

```
CREATE DATABASE cactus;
CREATE USER 'cactus'@'localhost' IDENTIFIED BY 'c@c7u$';
GRANT ALL PRIVILEGES ON cactus.* TO 'cactus'@'localhost';
FLUSH PRIVILEGES;
```

Update httpd from version 2.2 to 2.4, and update php from version 5.4 to 5.6. (Steps vary by machine and are not listed here.)

If you want to process notifications on the server (as opposed to the client):

```
sudo yum install python35-pip
sudo pip-3.5 install requests
```

Database Setup

For setup simplicity, additional table creation steps are handled when the server is run for the first time.

For reference only, the following tables will be created automatically.

```
mysql> describe devices;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| uuid      | char(36)     | NO   | PRI | NULL    |       |
| ip        | varchar(39)  | YES  |     | NULL    |       |
| user      | varchar(254) | YES  | MUL | NULL    |       |
| nickname  | varchar(64)  | YES  |     | NULL    |       |
| automatic | tinyint(1)   | YES  |     | 0       |       |
+-----------+--------------+------+-----+---------+-------+

mysql> describe measurements;
+-----------+------------------+------+-----+---------+----------------+
| Field     | Type             | Null | Key | Default | Extra          |
+-----------+------------------+------+-----+---------+----------------+
| id        | int(11)          | NO   | PRI | NULL    | auto_increment |
| device    | char(36)         | NO   | MUL | NULL    |                |
| channel   | int(11)          | NO   |     | NULL    |                |
| timestamp | datetime         | NO   |     | NULL    |                |
| value     | int(10) unsigned | NO   |     | NULL    |                |
+-----------+------------------+------+-----+---------+----------------+

mysql> describe users;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| email         | varchar(254) | NO   | PRI | NULL    |       |
| first_name    | varchar(64)  | NO   |     | NULL    |       |
| last_name     | varchar(64)  | NO   |     | NULL    |       |
| password_hash | varchar(255) | NO   |     | NULL    |       |
| maker_key     | varchar(64)  | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```

SmartGarden
Analysis of Senior Project Design

Matthew Lindly and Samuel Lees
Spring 2017

Summary of Functional Requirements

- Describe the overall capabilities of functions of your project or design.

  We have designed an automated plant-watering system that can measure plant moisture values, water plants when dry, send data to a collection server, send texts, emails and push notifications to the user, graph historical measurement data, validate multiple users when logging in, and register devices and sensors as new users sign up.

- Describe what your project does.

  The SmartGarden system solves the common problem of improper plant watering by configuring a Raspberry Pi to measure soil moisture values, storing the data on an AWS server in the cloud, and watering the plant when necessary, informing the user by providing both a web-interface and a variety of notifications.

Primary Constraints

- Describe significant challenges or difficulties associated with your project or implementation.
  - For example, what were limiting factors or other issues that impacted your approach?

    The two primary limiting factors for this project were cost and indoor usability. When researching this project, we found a lot of similar products, but nearly all were either expensive or for outdoor use, if not both. We designed a system that a wide range of interested user can afford and use in their own home.

  - What made your project difficult?

    Part of what made our project difficult was that we started it before having finished taking databases, networks, or security. There were a few major changes we had to make along the way, like moving everything from a SQLite database to a SQL database. Another significant difficulty was handling multiple access by simultaneous users, which we solved by creating a multithreaded server.

- What parameters or specifications limited your options or directed your approach?

  As our project was designed as an Internet of Things project, we designed a system that could run on lightweight, low-cost devices (a Raspberry Pi). This caused us to implement our client code for that particular device, especially when it came to the sensor and pump wiring and control.

Economic

- Original estimated cost of component parts (as of the start of your project)

  As this project was a continuation from our IoT class, we started already had access to a Raspberry Pi for each system as well as sensors and cables for interconnection. We planned to spend approximately $200 on liquid pumps, mounting hardware, and water tubing.

- Actual final cost of component parts (at the end of your project)

  We ended up spending just over $180, which was within our planned budget. We ended up buying some additional wires and chargers, and the pumps ended up being less expensive than we planned for. Each system costs roughly $80 - $105 depending on whether the users is purchasing multiple systems together and which Raspberry Pi accessories are included.

- Original estimated development time (as of the start of your project)

  We estimated we would each be spending approximately 6 hours/wk.

- Actual development time (at the end of your project)

  Our estimate was fairly accurate in terms of hours, though it was slightly weighted towards the end of each quarter. Despite this, about half the work was completed each quarter, which is what we wanted, as opposed to the last quarter being rushed because of procrastination.

Environmental

- Describe any environmental impact associated with manufacturing or use.

  We do not anticipate any negative environmental impact. If anything, our device will raise awareness about water conservation and proper plant health.

Manufacturability

- Describe any issues or challenges associated with manufacturing.

  The hardware component of our project was made entirely of off-the-shelf parts, thus we experienced no challenges in manufacturing. We anticipate that any improvements in manufacturing processes will lower the cost of our system over time

Sustainability

- Describe any issues or challenges associated with maintaining the completed device or system.

  One serious sustainability issue that we noticed late in the quarter was that the moisture probes will oxidize if left in wet soil for long periods of time, and hydrolysis will leech the metal contacts form the probe, effectively destroying it over time. Besides simply replacing the probes as necessary, another metal (sacrificial donor) could be attached to the probe to slow this process. Higher quality probes would also last longer.

- Describe how the project impacts the sustainable use of resources.

  While this project was designed specifically for succulents, there is no reason it could not be adapted for home-grown herbs or produce. This device could take part in the rising push for home-grown and locally sourced produce, which promises better quality and reduction on the carbon footprint inherent in ground transportation.

- Describe any upgrades that would improve the design of the project.

  A few significant improvements can be made to our project:
  - The wires could be condensed into a single ribbon cable that plugs into the GPIO pins of the Raspberry Pi, eliminating some clutter.
  - The Raspberry Pi could be powered by the 12-V squid cable if we include an adapter.
  - The system needs to be made waterproof and placed in a more permanent housing. We had plans to mount the pump and sensors on a crate that enclosed the water supply.
  - The change password feature in the web interface should be implemented, but can be done manually for the time being.

- Describe any issues or challenges associated with upgrading the design.

  The issues above would be trivial to address, and our code has already been rewritten to greatly enhance scalability. The most difficult part of this project for anyone moving

forward would be to add to the web interface. While it suits our needs, it is all custom HTML and PHP, which would be very sensitive to any changes in backend design.

Ethical

- Describe ethical implications relating to the design, manufacture, use or misuse of the project.

  This device could potentially be used in the illegal production of controlled substances, though the device itself is not likely to motivate an otherwise law-abiding individual to do so.

Health and Safety

- Describe any health and safety concerns associated with design, manufacture or use.

  There are not any known health concerns with this system. Care should be taken not to allow water from the system to come in contact with any of the electronics or high-voltage power outlets.

Social and Political

- Describe any social and political concerns associated with design, manufacture or use.

  We do not foresee any concerns or issues of this nature.

Development

- Describe any new tools or techniques used for either development or analysis that you learned independently during the course of your project.

  A few new things we learned over the course of this project:
  - We learned how to communicate with IoT devices using CoAP, though we ultimately ended up using TCP instead.
  - We learned about managing and hosting a web server using Amazon Web Services.
  - We discovered that IFTTT is a fantastic service for reliably delivering a wide range of customizable notifications from text to email to phone calls.
  - As this was a multi-device project, Git was integral to keeping code up to date on all of our devices. We had used Git before, but never to synchronize software versions across devices.