Korosh Moosavi
Fall 2021

Emerald Thumb Configuration

The Emerald Thumb system consists of a physical device, cloud resources, and a mobile app. Configuration is minimal for this system. The information included in this document is also available as individual ReadMe.txt files in the relevant code folders.

The Device:

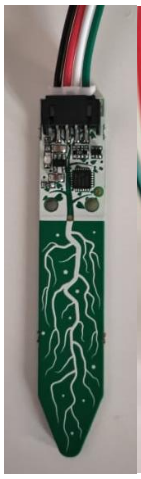
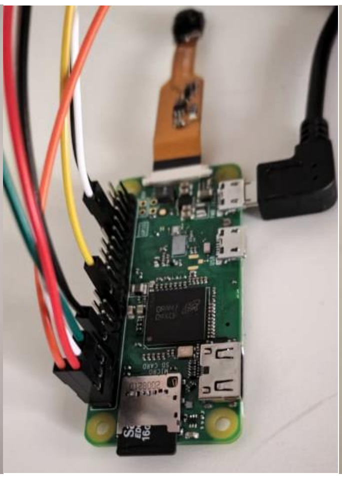
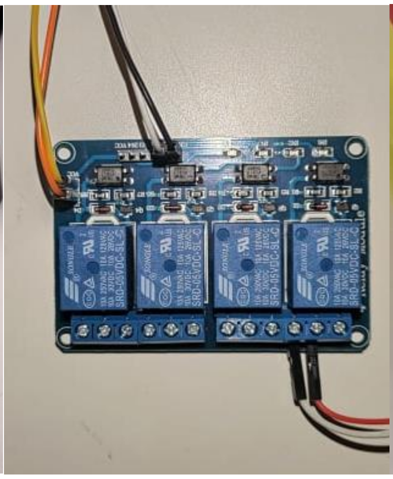
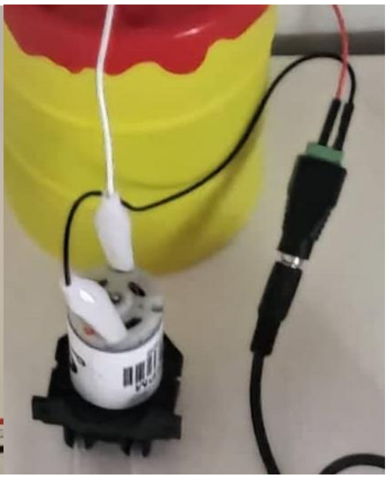
Hardware:

The device hardware isn't difficult to assemble. I'll include a brief guide here.

The soil sensor is connected to the Pi with a STEMMA cable. The color-coding connection guide for the STEMMA cable to the Raspberry Pi is [Sensor->Pi] black (GND) to GND, red (VIN) to 3V3, white (SDA) to SDA, green (SCL) to SCL.

The relay is connected to the Pi through two pairs of jumper cables. The first pair is an orange wire (JD_VCC) that connected to the Pi's 5V pin and a yellow wire (VCC) that connects to a second 3V3 on the Pi. The next pair is a black wire (GND) with connects to the Pi's GND, and a white wire (IN1) which was connected to Pi Pin 12.

The relay is then connected to the pump and power through red and white wires screwed into one channel. The 12V power supply is connected to a wire-screw adapter which connects to the relay and pump. The relay's red wire connects from the central output for the channel to the + end of the power supply adapter. A white alligator clip wire (male) connects the leftmost output of the channel to the pump (it doesn't matter which input on the pump, it will only affect flow direction). A similarly black alligator clip connects from the adapter's – end to the other pump input.

Software:

The device is configured with the crontab function in cmd to run the emeraldThumb.py script in the background 60 seconds after startup. The output of this script is redirected to ETlog.txt.

In the terminal:

```
sudo crontab -e

@reboot sleep 60 && cd /home/pi/Desktop/EmeraldThumb && python3 emeraldThumb.py
     > ETlog.txt 2>&1
```

The EmeraldThumb folder on the device contains the certs, keys, plant photo, log text file, and water level text file.

The Cloud:

Set up a new Thing in IoT Core for the device and attach a policy that allows the device to receive from the ‘rpi/input’ MQTT topic and publish to ‘rpi/output’ and ‘rpi/log’. It should also allow ‘iot:Connect’ for the client ID ‘EmeraldThumb’.

Download certs and keys into the ‘EmeraldThumb’ folder in the device, in the same directory as emeraldThumb.py

Create an S3 bucket ‘rpiet’ with 4 folders: ‘app’, ‘data’, ‘raw’, and ‘photo’

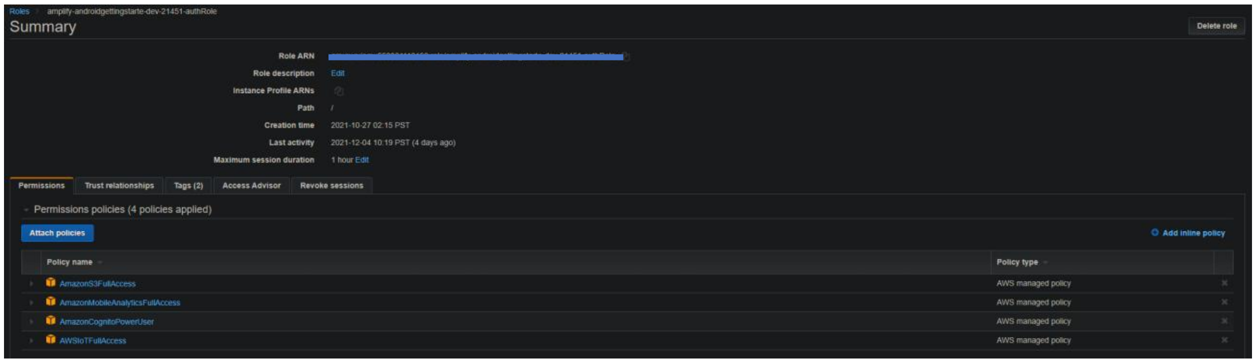
Create the first lambda function that processes MQTT data. Set a rule to trigger that uses the contents of a message published to ‘rpi/output’ as the lambda's input event. Give this function S3 permissions to put new files into the 'data' and 'raw' folders.

Create the second lambda function to republish S3 'app' folder uploads to ‘rpi/input’. Ensure this lambda has the appropriate S3 get permissions and IoT publish permissions.

Both lambda functions’ configurations and code can be seen in the included files for each lambda function (\cloudCode\Lambda Functions).

I used Amplify CLI to create a Cognito identity pool and federated identity info. This can be done manually, however your information will need to be updated into the mobile app's amplifyconfiguration.json and awsconfiguration.json files.

The auth role for your identities should include permissions to get and put objects in S3. More info on Amplify in the next section.


Roles > amplify-androidgettingstarte-dev-21451-authRole

Summary

Role ARN	[redacted]
Role description	Edit
Instance Profile ARNs	
Path	/
Creation time	2021-10-27 02:15 PST
Last activity	2021-12-04 10:19 PST (4 days ago)
Maximum session duration	1 hour Edit

Permissions policies (4 policies applied)

Policy name	Policy type
AmazonS3FullAccess	AWS managed policy
AmazonMobileAnalyticsFullAccess	AWS managed policy
AmazonCognitoPowerUser	AWS managed policy
AWSIoTFullAccess	AWS managed policy

Cognito identity pool Auth Role policies. IoT policies are not required for this implementation

The Mobile App:

If you just want to use the app, I have included a working debug build of the app as an installable .apk file. The following steps are if you want to configure it to use your own AWS resources with Android Studio.

The app was built and tested using the Pixel 2 API 26 environment in Android Studio. I used the Amplify CLI to configure my AWS account information, which is saved to the configuration json files. These files can also be configured manually by replacing the values I have with those of another AWS account. This will mainly include updating the stored Cognito credentials.

To configure with Amplify, first install the CLI, then use the 'amplify configure', 'amplify add auth', and 'amplify add storage'

```
curl -sL https://aws-amplify.github.io/amplify-cli/install-win -o install.cmd
                                                            && install.cmd
```

Each has on-screen instructions on what info it needs to continue, I used default settings for everything

```
amplify configure

      [ Provide AWS Account Information ]

amplify add auth

      [ Create a New Cognito Identity Pool for the Mobile App ]

      [ Use default settings where possible ]

amplify add storage

      [ Configure Amplify to Access the S3 ‘rpiet’ Bucket ]
```

End with 'amplify push' to publish your changes to the cloud. The environment is now set up

```
amplify push
```

After this Android Studio should be able to work on the project with your own AWS credentials.

The final step to opening the project is to rename a few of the included folders.

Korosh Moosavi
Fall 2021

At this path:

```
Emerald Thumb\emeraldthumb\build\intermediates
      \desugar_graph\release\out\currentProject
```

There are 4 folders named rename0 - rename3

Rename each of these folders with this string, ending with the corresponding number

Ex:

```
rename3 ->
jar_e52818b21966267abf4fa2fb4f7d323dfd884dcc2dd8a5fafb386f54d8f83b4a_bucket_3
```