



# Irrigation System Automation Using Home Assistant



Ali Boraqi  
Information Technology Project  
Management  
San Francisco State University  
May 27<sup>th</sup>, 2021

A cluster of hexagons in the top-left corner, including a large cyan one, a dark blue one, and several smaller ones in various shades of blue and cyan.

# Summary

- Design an affordable irrigation system
- Use automation for efficiency and avoid water waste
- Never forget to water plants
- Access remotely
- Analyze network hierarchy features



A cluster of hexagons in various shades of blue and green, some solid and some outlined, arranged in a honeycomb-like pattern in the top-left corner.

# Use Case

One of the primary reasons for plants failing to thrive is insufficient or excessive water maintenance. Insufficient water supply can result in inadequate hydration, while excessive watering can lead to root decay, causing the plants to perish.

How to use water less extensively and improve the crop growth and quality?

By incorporating IoT technology, this design introduces an automated irrigation system that not only saves time, money, and energy for farms and gardens but also enhances the growth and quality of crops.





# Home Assistant Background

Home Assistant is an open source home automation system. Originally, created by Paulus Schoutsen written in Python 3.8 and was first published on GitHub in November 2013. In July 2017, Home Assistant was introduced to an operating system called Hass.io to be installed on a Raspberry Pi in order to manage backups, updates, and extending the functionality options of the software with add-ons.





# Home Assistant Setup

1. Attach SD card to computer
2. Download and start [Balena Etcher](#)
3. Select “Flash from URL” for the Raspberry Pi
4. Paste the URL into Balena Etcher and click “OK”
5. Balena Etcher will download the image, when that is done click “Select target”
6. Select the SD card for the Raspberry Pi
7. Click on “Flash!” to start writing the image
8. Confirmation

\$85





# Welcome to Home Assistant

- Insert the installation media (SD card) into the Pi
- Attach the ethernet cable
- Attach the power cable
- Within a few minutes able to access Home Assistant on `homeassistant.local:8123`

The screenshot displays the Home Assistant web interface. On the left is a sidebar menu with the following items: 'Overzicht', 'Kaart', 'Logboek', 'Geschiedenis', 'Hass.io' (highlighted), 'Instellingen', and 'Ontwikkelaarstools'. The main content area is titled 'Hass.io: add-on details' and shows the 'Configurator' add-on. It includes a description: 'Browser-based configuration file editor for Home Assistant.. Visit [Configurator page](#) for details.' Below this is the 'Add-on Security Rating' section, which states: 'Hass.io provides a security rating to each of the add-ons, which indicates the risks involved when using this add-on. The more access an add-on requires on your system, the lower the score, thus raising the possible security risks.' The rating is visualized with three circular icons: a green circle with the number '5' labeled 'RATING', a blue circle with a house icon labeled 'HAAS', and a blue circle with a shield icon labeled 'APPROVAL'. Below these icons is the text 'default'. At the bottom of the configurator window is an 'INSTALL' button with a yellow play icon.



# Home Assistant Add ons

**ESPHome**

**Node-RED**

**File editor**

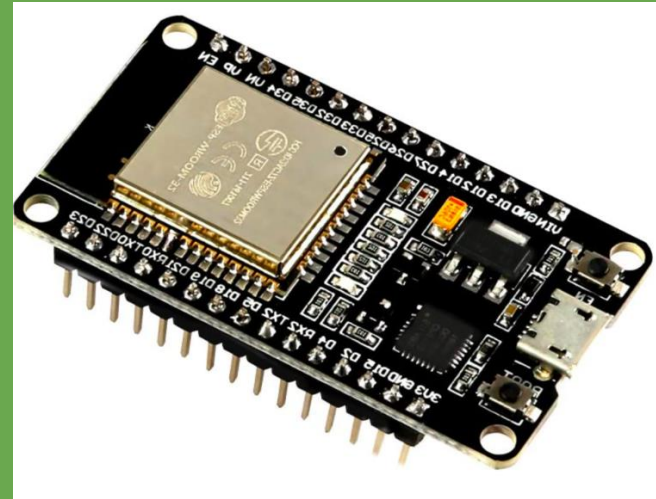
**SSH & Web  
Terminal**



# ESP32

- ★ ESP32 is a microcontroller chip with integrated Wi-Fi
- ★ It is a power amplifier, safe, reliable, and scalable to a variety of applications.
- ★ Easy to setup and integrate into home assistant
- ★ For the purpose of this project, ESP32 was used to power and manage data

\$14







# Additional Parts Irrigation System

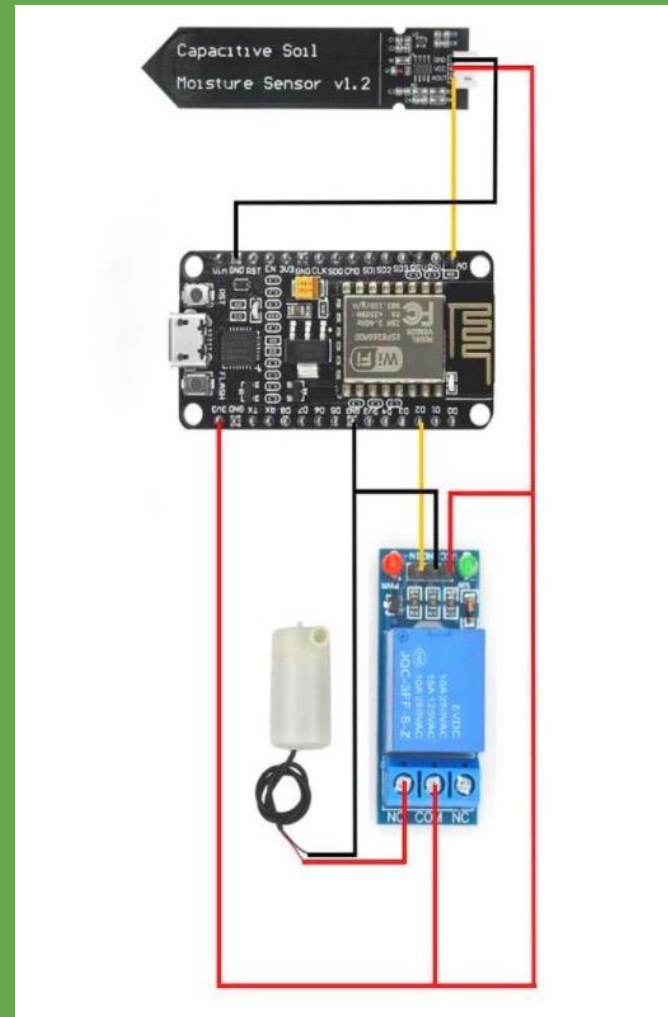
\$20

- Water Pump
- Soil Moisture Sensor
- Capacitive Soil Moisture Sensor
- 1 Channel 5V Relay
- Water Pump + 1M Vinyl Tubing
- DHT22 Temperature Humidity Sensor



# Wiring

- The ESP32 is powered via 3V3 and GND
- Moisture sensor has 3 wires, the black is connected to the GND, Red VCC is power the Yellow AOUT is for the Data
- Notice the relay and water pump also powered via the VCC Red Wire and GND is The Black Wire





# Yaml Code

```
esphome:
  name: node_one_plant
  platform: ESP32
  board: esp-wrover-kit
wifi:
  ssid: "Name"
  password: "passwor"
```

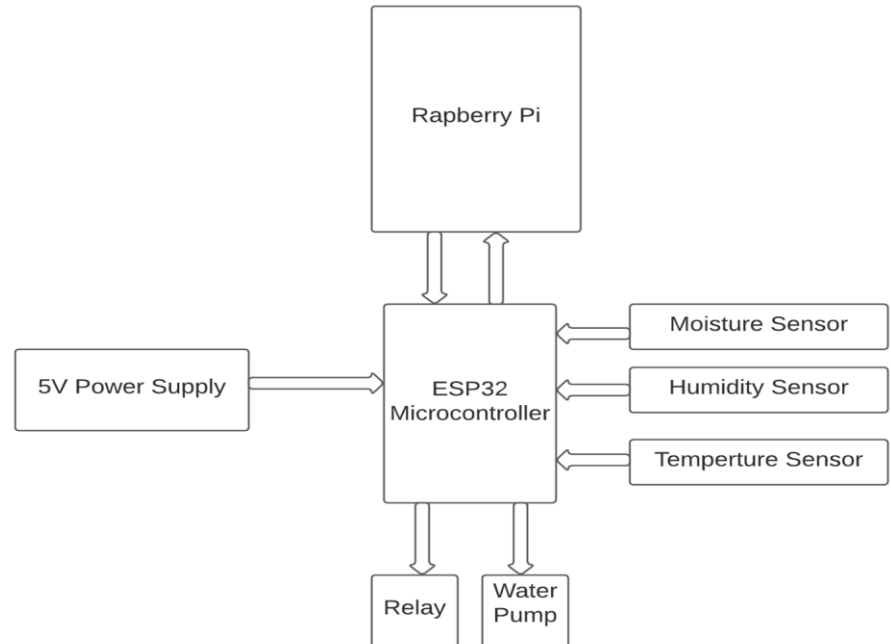
```
sensor:
  - platform: dht
    pin: 25
    model: dht22
    temperature:
      name: "Temperature"
    humidity:
      name: "Humidity"
    update_interval: 30s
```

```
switch:
  - platform: gpio
    pin: 23
    name: "Activate"
    inverted: yes
```

```
sensor:
  - platform: adc
    pin: 33
    name: "Soil Moisture Level"
    update_interval: 1s
    unit_of_measurement: "%"
    icon: "mdi:flower-outline"
    attenuation: 11db
    filters:
      - median:
          window_size: 7
          send_every: 2
          send_first_at: 1
      - calibrate_linear:
          - 1.25 -> 100.00 #dry
          - 2.8 -> 0.00 #wet
          - lambda: if (x < 0) return 0; else if (x > 100)
            return 100; else return (x);
```

# Block Diagram of the entire system

- ❖ The soil moisture, temperature & humidity sensors, and water pump, are in sync with the ESP32 updating the Pi with values periodically on Home Assistant.
- ❖ While the ESP32 is communicating with the Pi the Macbook is also engaged in the conversation with the Pi to display and send data.

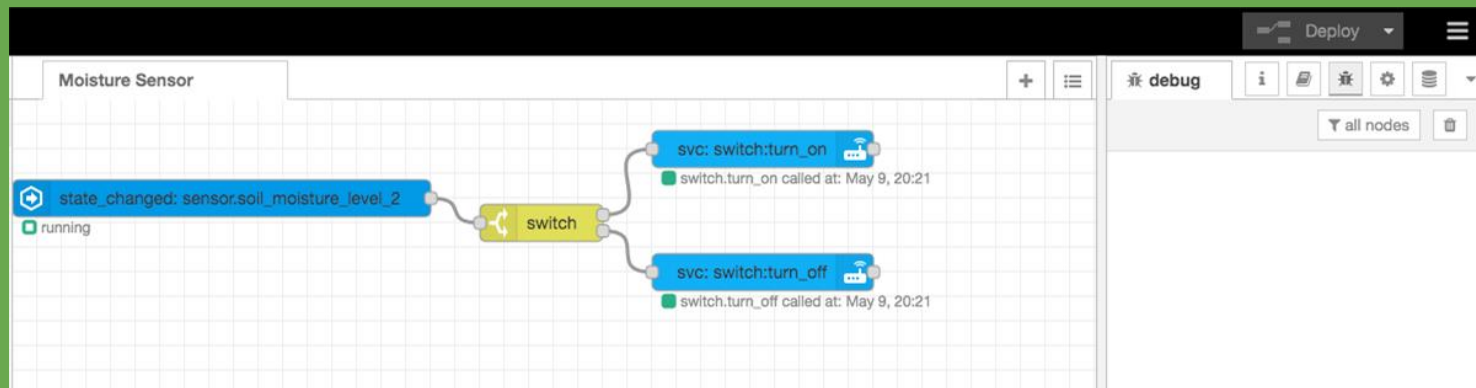


# Automation using Nod-RED

Create an event  
this case a sensor  
event

Connected  
to a switch

On/Off





# Prototype

The prototype is showing the microcontrollers one is connected to temperature & humidity sensor, and the other to soil moisture sensor, relay, and water pump.

These sensors sense the various parameter of the soil, temp & humidity, and the motor is used to provide water upon need, and finally the relay control the water pump.





# Communication Analysis

## TCPDUMP

- Install & run
- Captured data on the Pi
- Uploaded data onto  
Wireshark

## IP Addresses:

- Macbook == 192.168.0.158
- Pi HA == 192.168.0.103
- Gateway == 192.168.0.1
- Node1P == 192.168.0.158
- Node2H&T == 192.168.0.82



# Conversations between devices

Wireshark · Conversations · capture3.pcap

Ethernet IPv4 · 24 IPv6 · 3 TCP · 43 UDP · 28

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.0.158	224.0.0.251	2	222	2	222	0	0	82.193815	0.0994	17k	0
192.168.0.103	192.168.0.158	216	14k	114	6894	102	7300	0.073398	169.3957	325	344
192.168.0.103	192.168.0.104	180	21k	108	7530	72	14k	9.874274	150.0270	401	754
192.168.0.103	192.168.0.128	123	15k	73	5064	50	10k	9.874274	150.0248	270	548
192.168.0.103	239.255.255.250	3	414	3	414	0	0	44.925833	120.0043	27	0
192.168.0.103	224.0.0.22	12	720	12	720	0	0	44.935360	124.8119	46	0
192.168.0.103	192.168.0.140	15	9399	0	0	15	9399	46.120325	122.6685	0	612
192.168.0.103	224.0.0.251	7	702	7	702	0	0	81.960501	28.8065	194	0
192.168.0.103	192.168.0.103	61	3930	25	1734	36	2196	1.191444	165.6309	83	106
192.168.0.103	224.0.0.251	2	220	2	220	0	0	82.181716	0.1003	17k	0
192.168.0.103	192.168.0.103	361	49k	201	19k	160	29k	0.000000	170.0890	927	1406
192.168.0.103	224.0.0.251	11	1371	11	1371	0	0	3.058585	153.1885	71	0
192.168.0.103	192.168.0.255	5	460	5	460	0	0	26.722898	120.0220	30	0
192.168.0.55	224.0.0.251	16	5838	16	5838	0	0	109.706631	5.3331	8757	0
192.168.0.1	192.168.0.103	24	7854	24	7854	0	0	44.926431	120.0050	523	0
172.30.32.3	224.0.0.251	32	2976	32	2976	0	0	81.959938	10.0025	2380	0
172.30.32.3	224.0.0.22	48	2880	48	2880	0	0	81.971306	18.7842	1226	0
172.30.32.2	172.30.32.6	24	2170	14	1284	10	886	114.511835	15.1298	678	468
172.30.32.1	224.0.0.251	248	61k	248	61k	0	0	3.058975	153.1887	3196	0
172.30.32.1	172.30.32.2	628	244k	312	204k	316	39k	11.642268	157.8327	10k	1999
172.30.32.1	172.30.33.0	18	1320	12	876	6	444	42.171721	120.8751	57	29
172.30.32.1	172.30.32.3	8	800	4	360	4	440	130.001139	0.0008	—	—
169.254.239.55	239.255.255.250	1	203	1	203	0	0	165.878751	0.0000	—	—
127.0.0.1	127.0.0.1	6	435	6	435	0	0	42.171155	120.8749	28	0

Node -RED

Mac

Raspberry Pi





capture3.pcap

ip.addr == 192.168.0.82

No.	Time	Source	Destination	Protocol	Length	Info
10	1.244251	192.168.0.82	192.168.0.103	TCP	66	6053 → 47180 [ACK] Seq=1 Ack=4 Win=5359 Len=3 [TCP segment of a reassembled PDU]
24	3.451971	192.168.0.82	192.168.0.103	TCP	73	6053 → 47180 [PSH, ACK] Seq=4 Ack=4 Win=5359 Len=13 [TCP segment of a reassembled PDU]
26	3.454915	192.168.0.82	192.168.0.103	TCP	73	6053 → 47180 [PSH, ACK] Seq=17 Ack=4 Win=5359 Len=13 [TCP segment of a reassembled PDU]
142	16.297812	192.168.0.82	192.168.0.103	X11	66	Error: Success
362	31.346086	192.168.0.82	192.168.0.103	TCP	66	6053 → 47180 [ACK] Seq=33 Ack=10 Win=5353 Len=0
363	31.355044	192.168.0.82	192.168.0.103	TCP	66	6053 → 47180 [ACK] Seq=33 Ack=10 Win=5353 Len=3 [TCP segment of a reassembled PDU]
367	33.442772	192.168.0.82	192.168.0.103	TCP	73	6053 → 47180 [PSH, ACK] Seq=36 Ack=10 Win=5353 Len=13 [TCP segment of a reassembled PDU]
371	33.447858	192.168.0.82	192.168.0.103	TCP	73	6053 → 47180 [PSH, ACK] Seq=49 Ack=10 Win=5353 Len=13 [TCP segment of a reassembled PDU]
555	46.412134	192.168.0.82	192.168.0.103	X11	66	Error: Success
626	61.451792	192.168.0.82	192.168.0.103	TCP	66	6053 → 47180 [ACK] Seq=65 Ack=16 Win=5347 Len=3 [TCP segment of a reassembled PDU]
636	63.442107	192.168.0.82	192.168.0.103	TCP	73	6053 → 47180 [PSH, ACK] Seq=68 Ack=16 Win=5347 Len=13 [TCP segment of a reassembled PDU]
638	63.447485	192.168.0.82	192.168.0.103	TCP	73	6053 → 47180 [PSH, ACK] Seq=81 Ack=16 Win=5347 Len=13 [TCP segment of a reassembled PDU]
781	76.528404	192.168.0.82	192.168.0.103	X11	66	Error: Success
970	91.562053	192.168.0.82	192.168.0.103	TCP	66	6053 → 47180 [ACK] Seq=97 Ack=22 Win=5347 Len=0
1016	93.441254	192.168.0.82	192.168.0.103	TCP	73	6053 → 47180 [PSH, ACK] Seq=100 Ack=22 Win=5347 Len=13 [TCP segment of a reassembled PDU]
1018	93.448903	192.168.0.82	192.168.0.103	TCP	73	6053 → 47180 [PSH, ACK] Seq=113 Ack=22 Win=5347 Len=13 [TCP segment of a reassembled PDU]
1166	186.615663	192.168.0.82	192.168.0.103	X11	66	Error: Success

Frame 11: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on 0  
Linux cooked capture v2  
Internet Protocol Version 4, Src: 192.168.0.82, Dst: 192.168.0.82  
Transmission Control Protocol, Src Port: 47180, Dst Port: 6053, Seq: 4, Ack: 4, Len: 0  
Source Port: 47180  
Destination Port: 6053  
[Stream index: 2]  
[TCP Segment Len: 0]  
Sequence Number: 4 (relative sequence number)  
Sequence Number (raw): 1943116531  
[Next Sequence Number: 4 (relative sequence number)]  
Acknowledgment Number: 4 (relative ack number)

```

0000 00 00 00 00 00 00 03 00 01 04 06 dc a6 32 f9 .....2-
0010 8f 3d 00 00 45 00 00 28 8e 25 40 00 06 2a a1 ...E...@.*
0020 c0 a8 00 67 c0 a8 00 52 b8 4c 17 a5 73 d1 9a f3 ...g...R...
0030 00 00 21 c1 50 10 fa 0d 33 45 00 00 ...P...3E..

```

capture3.pcap

ip.addr == 192.168.0.158

No.	Time	Source	Destination	Protocol	Length	Info
5	0.113772	192.168.0.158	192.168.0.103	TCP	66	6053 → 34976 [ACK] Seq=1 Ack=4 Win=5674 Len=3 [TCP segment of a reassembled PDU]
12	1.472192	192.168.0.158	192.168.0.103	TCP	68	6053 → 34976 [PSH, ACK] Seq=4 Ack=4 Win=5674 Len=8 [TCP segment of a reassembled PDU]
28	3.470715	192.168.0.158	192.168.0.103	TCP	68	6053 → 34976 [PSH, ACK] Seq=12 Ack=4 Win=5674 Len=8 [TCP segment of a reassembled PDU]
30	5.472570	192.168.0.158	192.168.0.103	TCP	68	6053 → 34976 [PSH, ACK] Seq=20 Ack=4 Win=5674 Len=8 [TCP segment of a reassembled PDU]
32	7.472675	192.168.0.158	192.168.0.103	X11	68	Error: Success [TCP segment of a reassembled PDU]
34	9.469423	192.168.0.158	192.168.0.103	TCP	68	6053 → 34976 [PSH, ACK] Seq=36 Ack=4 Win=5674 Len=8 [TCP segment of a reassembled PDU]
76	11.472236	192.168.0.158	192.168.0.103	TCP	68	6053 → 34976 [PSH, ACK] Seq=44 Ack=4 Win=5674 Len=8 [TCP segment of a reassembled PDU]
84	13.471760	192.168.0.158	192.168.0.103	X11	73	Event: Sent-MotionNotify
106	15.166996	192.168.0.158	192.168.0.103	TCP	66	6053 → 34976 [ACK] Seq=65 Ack=17 Win=5661 Len=0
107	15.168817	192.168.0.158	192.168.0.103	TCP	70	6053 → 34976 [PSH, ACK] Seq=65 Ack=17 Win=5661 Len=10 [TCP segment of a reassembled PDU]
109	15.171550	192.168.0.158	192.168.0.103	TCP	66	6053 → 34976 [ACK] Seq=75 Ack=17 Win=5661 Len=3 [TCP segment of a reassembled PDU]
125	15.470471	192.168.0.158	192.168.0.103	TCP	73	6053 → 34976 [PSH, ACK] Seq=78 Ack=17 Win=5661 Len=13 [TCP segment of a reassembled PDU]
149	16.916704	192.168.0.158	192.168.0.103	X11	68	Error: BadFont [TCP segment of a reassembled PDU]
165	17.479180	192.168.0.158	192.168.0.103	TCP	73	6053 → 34976 [PSH, ACK] Seq=99 Ack=25 Win=5653 Len=13 [TCP segment of a reassembled PDU]
181	19.473678	192.168.0.158	192.168.0.103	TCP	73	6053 → 34976 [PSH, ACK] Seq=112 Ack=25 Win=5653 Len=13 [TCP segment of a reassembled PDU]
197	21.470826	192.168.0.158	192.168.0.103	X11	73	Event: <Unknown eventcode 86> [TCP segment of a reassembled PDU]
311	23.469329	192.168.0.158	192.168.0.103	TCP	73	6053 → 34976 [PSH, ACK] Seq=138 Ack=25 Win=5653 Len=13 [TCP segment of a reassembled PDU]

Frame 12: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on 0  
Linux cooked capture v2  
Internet Protocol Version 4, Src: 192.168.0.158, Dst: 192.168.0.103  
Transmission Control Protocol, Src Port: 6053, Dst Port: 34976, Seq: 4, Ack: 4, Len: 8  
Source Port: 6053  
Destination Port: 34976  
[Stream index: 1]  
[TCP Segment Len: 8]  
Sequence Number: 4 (relative sequence number)  
Sequence Number (raw): 7347  
[Next Sequence Number: 12 (relative sequence number)]  
Acknowledgment Number: 4 (relative ack number)

```

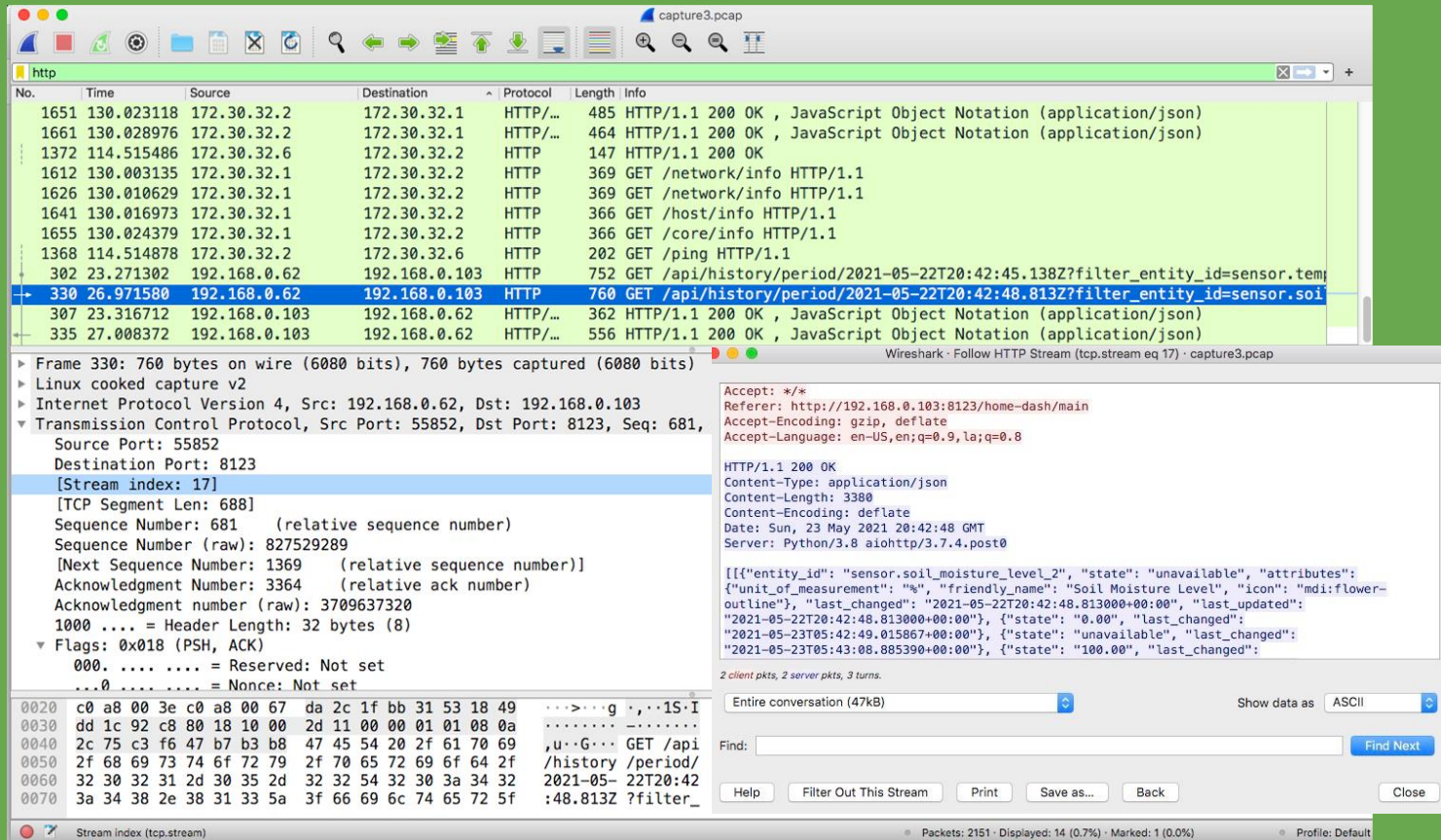
0000 00 00 00 00 00 00 03 00 01 00 06 00 3a f2 43 .....C
0010 e7 e4 00 00 45 00 00 30 00 59 00 0f 06 39 19 ...E...Y...9
0020 c0 a8 00 9e c0 a8 00 67 17 a5 88 a0 00 00 1c b3 ...g.....
0030 1b 24 a5 00 50 18 16 2a bf 1c 00 00 00 05 19 0d ...$.P.....
0040 46 86 7b 72 .....F{r

```

Frame 84: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on 0  
Linux cooked capture v2  
Internet Protocol Version 4, Src: 192.168.0.158, Dst: 192.168.0.103  
Transmission Control Protocol, Src Port: 6053, Dst Port: 34976, Seq: 52, Ack: 4, Len: 13  
Source Port: 6053  
Destination Port: 34976  
[Stream index: 1]  
[TCP Segment Len: 13]  
Sequence Number: 52 (relative sequence number)  
Sequence Number (raw): 7395  
[Next Sequence Number: 65 (relative sequence number)]  
Acknowledgment Number: 4 (relative ack number)  
Acknowledgment number (raw): 455386368  
0101 .... = Header Length: 20 bytes (5)  
Flags: 0x018 (PSH, ACK)  
000. .... = Reserved: Not set  
...0 .... = Nonce: Not set

# HTTP

Using method:  
GET/api



The image shows a Wireshark network packet capture of an HTTP GET request. The packet list on the left shows several HTTP requests, with packet 330 selected. The packet details pane on the right shows the structure of the selected packet, including the HTTP header and the JSON body. The packet bytes pane at the bottom shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
1651	130.023118	172.30.32.2	172.30.32.1	HTTP/...	485	HTTP/1.1 200 OK, JavaScript Object Notation (application/json)
1661	130.028976	172.30.32.2	172.30.32.1	HTTP/...	464	HTTP/1.1 200 OK, JavaScript Object Notation (application/json)
1372	114.515486	172.30.32.6	172.30.32.2	HTTP	147	HTTP/1.1 200 OK
1612	130.003135	172.30.32.1	172.30.32.2	HTTP	369	GET /network/info HTTP/1.1
1626	130.010629	172.30.32.1	172.30.32.2	HTTP	369	GET /network/info HTTP/1.1
1641	130.016973	172.30.32.1	172.30.32.2	HTTP	366	GET /host/info HTTP/1.1
1655	130.024379	172.30.32.1	172.30.32.2	HTTP	366	GET /core/info HTTP/1.1
1368	114.514878	172.30.32.2	172.30.32.6	HTTP	202	GET /ping HTTP/1.1
302	23.271302	192.168.0.62	192.168.0.103	HTTP	752	GET /api/history/period/2021-05-22T20:42:45.138Z?filter_entity_id=sensor.tem...
330	26.971580	192.168.0.62	192.168.0.103	HTTP	760	GET /api/history/period/2021-05-22T20:42:48.813Z?filter_entity_id=sensor.soil
307	23.316712	192.168.0.103	192.168.0.62	HTTP/...	362	HTTP/1.1 200 OK, JavaScript Object Notation (application/json)
335	27.008372	192.168.0.103	192.168.0.62	HTTP/...	556	HTTP/1.1 200 OK, JavaScript Object Notation (application/json)

Frame 330: 760 bytes on wire (6080 bits), 760 bytes captured (6080 bits) on Linux cooked capture v2  
Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.103  
Transmission Control Protocol, Src Port: 55852, Dst Port: 8123, Seq: 681, Source Port: 55852, Destination Port: 8123  
[Stream index: 17]  
[TCP Segment Len: 688]  
Sequence Number: 681 (relative sequence number)  
Sequence Number (raw): 827529289  
[Next Sequence Number: 1369 (relative sequence number)]  
Acknowledgment Number: 3364 (relative ack number)  
Acknowledgment number (raw): 3709637320  
1000 .... = Header Length: 32 bytes (8)  
Flags: 0x018 (PSH, ACK)  
000. .... = Reserved: Not set  
...0 .... = Nonce: Not set

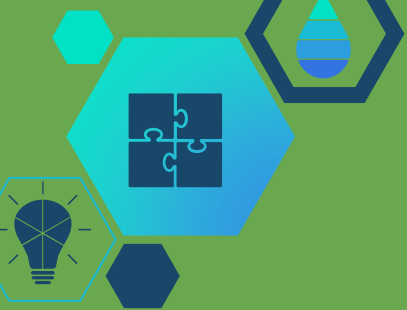
Accept: \*/\*  
Referer: http://192.168.0.103:8123/home-dash/main  
Accept-Encoding: gzip, deflate  
Accept-Language: en-US,en;q=0.9,la;q=0.8  
HTTP/1.1 200 OK  
Content-Type: application/json  
Content-Length: 3300  
Content-Encoding: deflate  
Date: Sun, 23 May 2021 20:42:48 GMT  
Server: Python/3.8 aiohttp/3.7.4.post0  
[{"entity\_id": "sensor.soil\_moisture\_level\_2", "state": "unavailable", "attributes": {"unit\_of\_measurement": "%", "friendly\_name": "Soil Moisture Level", "icon": "mdi:flower-outline"}, "last\_changed": "2021-05-22T20:42:48.813000+00:00", "last\_updated": "2021-05-22T20:42:48.813000+00:00"}, {"state": "0.00", "last\_changed": "2021-05-23T05:42:49.015867+00:00"}, {"state": "unavailable", "last\_changed": "2021-05-23T05:43:08.885390+00:00"}, {"state": "100.00", "last\_changed": "2021-05-23T05:43:08.885390+00:00"}]  
2 client pkts, 2 server pkts, 3 turns.  
Entire conversation (47kB)  
Find: [ ] Find Next  
Help Filter Out This Stream Print Save as... Back Close  
Packets: 2151 · Displayed: 14 (0.7%) · Marked: 1 (0.0%) Profile: Default

A decorative graphic on the left side of the slide consists of several hexagons in shades of blue and teal. Some hexagons contain icons: a lightbulb, a thumbs-up, a network of nodes, a smartphone, a magnifying glass, a gear, and a speech bubble. A large teal hexagon in the center of this cluster contains a white double quote symbol.

# Conclusion

This system ensures an adequate water supply that nourishes the soil and avoiding water waste while minimizing human intervention.

This project offers valuable hands-on experience with IoT devices, promoting the efficient and secure utilization of water resources in agricultural production. It showcases a cost-effective and efficient idea that paves the way for future advancements in agricultural development.



# Thanks!

