AgroTile

COEN 243 - Matthew Findlay, Shachi Kakkar, Kevin Velcich, Esai Morales

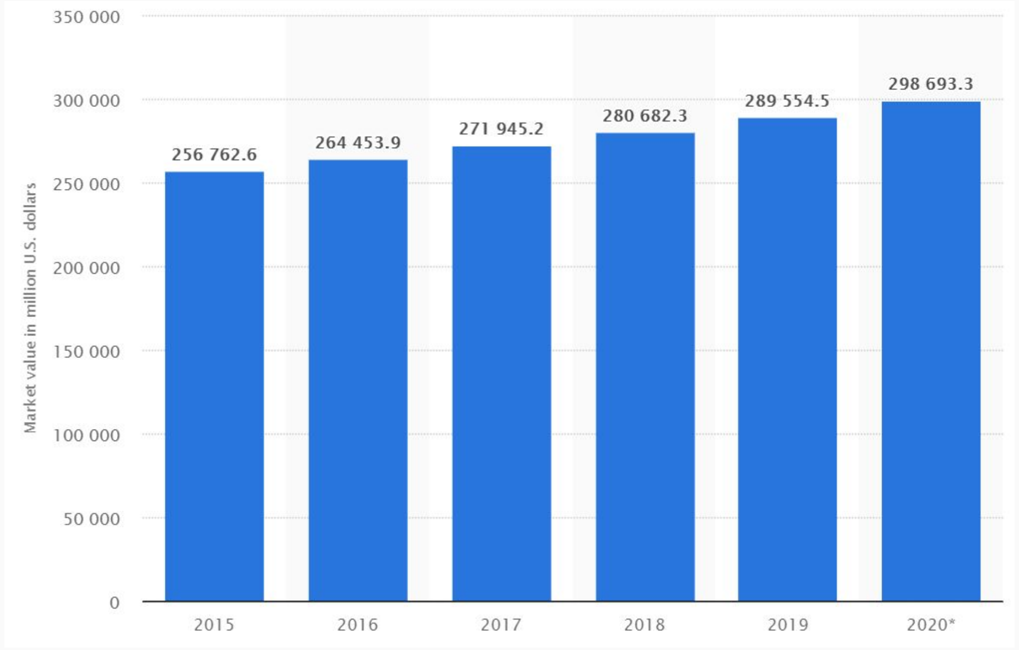
Motivation & Market

- Trend of indoor growing
  - expanding across the United States and the rest of the world
- All types of gardens
  - greenhouses
  - hoop houses
  - vertical farms
  - container farms
  - home growing systems

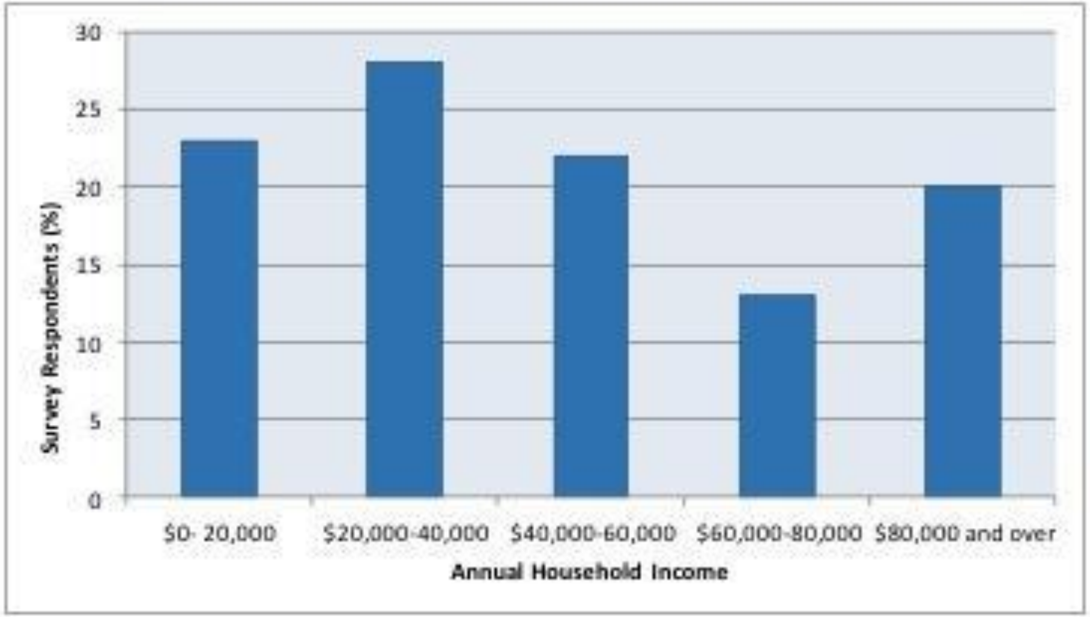
Motivation & Market *cont'd*

- Proportional rise in the personal gardening market
- Need for low-cost solutions to facilitate personal gardening needs

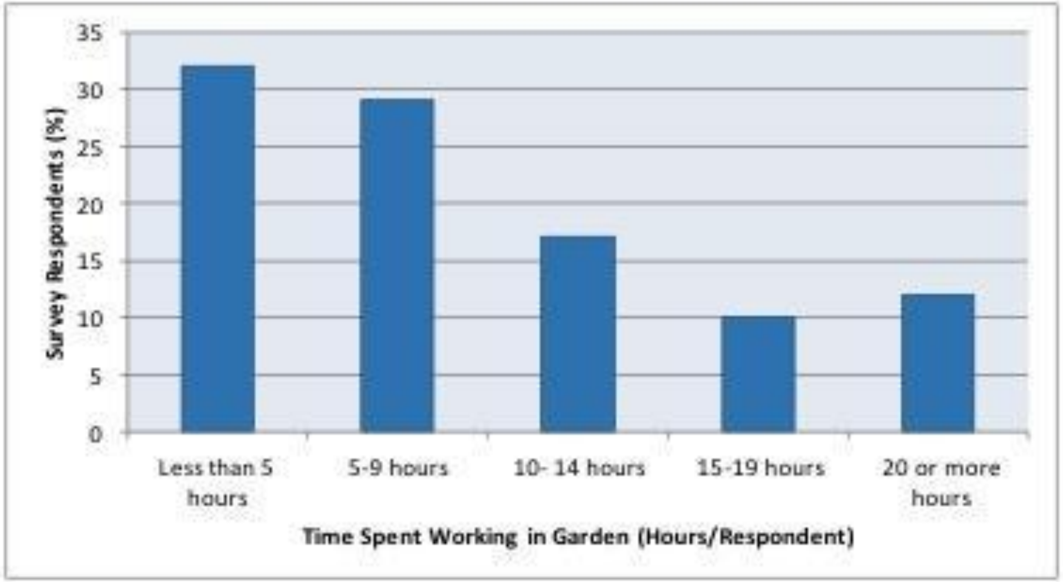
Growth of Garden Market



Income of People Who Garden



Average Time Spent Gardening



Current Solutions

- Current Solutions: offer not only soil moisture detection but also frost, humidity, leaf moisture, pesticide detection
    - Downside: built for large scale farms
        - Don’t cater to smaller gardens

Current Solutions *cont'd*

- Sensors displaying soil moisture on a small display
  - Do not post anything to website for tracking capabilities
- Sensor with a display averages $20 each
- Price point is too high for average gardener

Technologies Used

- Wifi Mesh
- Esp 32
- Soil Sensors
- HTTP/CSS/Javascript
- Arduino
- Async TCP Server (Library)

Our Contribution

- Simple and holistic device which allows someone to detect the moisture of the soil in which their plants are growing
- Processor attached to the sensor sends data to a web server which displays the information on a site so the user can determine when the plant went below the threshold soil moisture level
- By providing a mesh network, we are able to reduce the requirement of an access point which further reduces cost

Hardware

- Server and nodes are implemented on NodeMCU ESP-32S
- SenMod Moisture Sensors connected via GPIO pin
- 2000mAh batteries

Server Implementation

- It initializes mesh and web server
- Upon receiving a message
  - Verifies message is a moisture reading
  - Send acknowledgement to sending node
  - Dynamically store moisture value and current time in a unique list for node
  - If all nodes in mesh have sent messages and have waited a minimum waiting time, broadcast a sleep message
- When a client sends and HTML request
  - HTML structure with template values
  - Fill in template slots with JavaScript and HTML code to display a graph for each node

Node Sensor Implementation

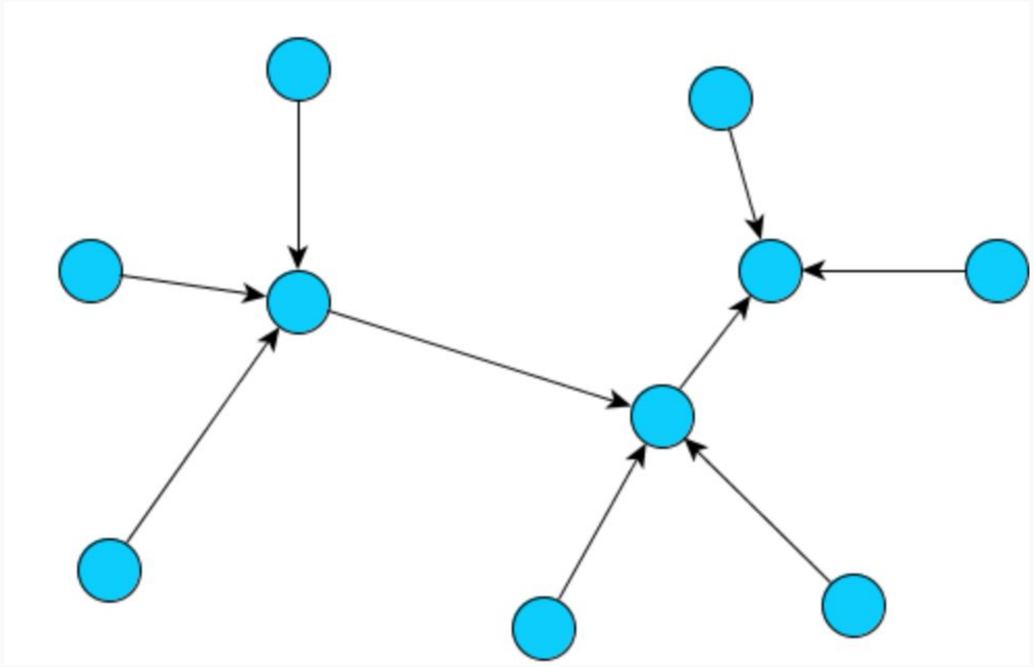
- On startup, joins the mesh network and begins sending the server it's moisture sensor readings every N seconds
- If we receive an acknowledgement message, we stop sending our moisture readings
- If we receive a sleep message, wait for broadcast to fully propagate throughout mesh and enter a deep sleep

Mesh Network

- Created a 802.11 mesh network using Painless Mesh library
- Layer 3 protocol
- All nodes act as an AP
- Creates a star mesh network by joining any unknown nodes using the same SSID.
- Will automatically join the node with the strongest signal

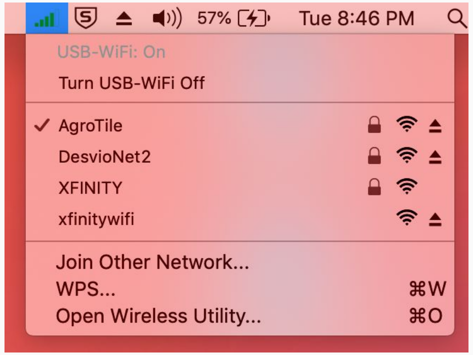
Mesh Routing

- TCP based connections between nodes
- Every node has a view of the entire mesh
- Mesh topology updated every 3 seconds

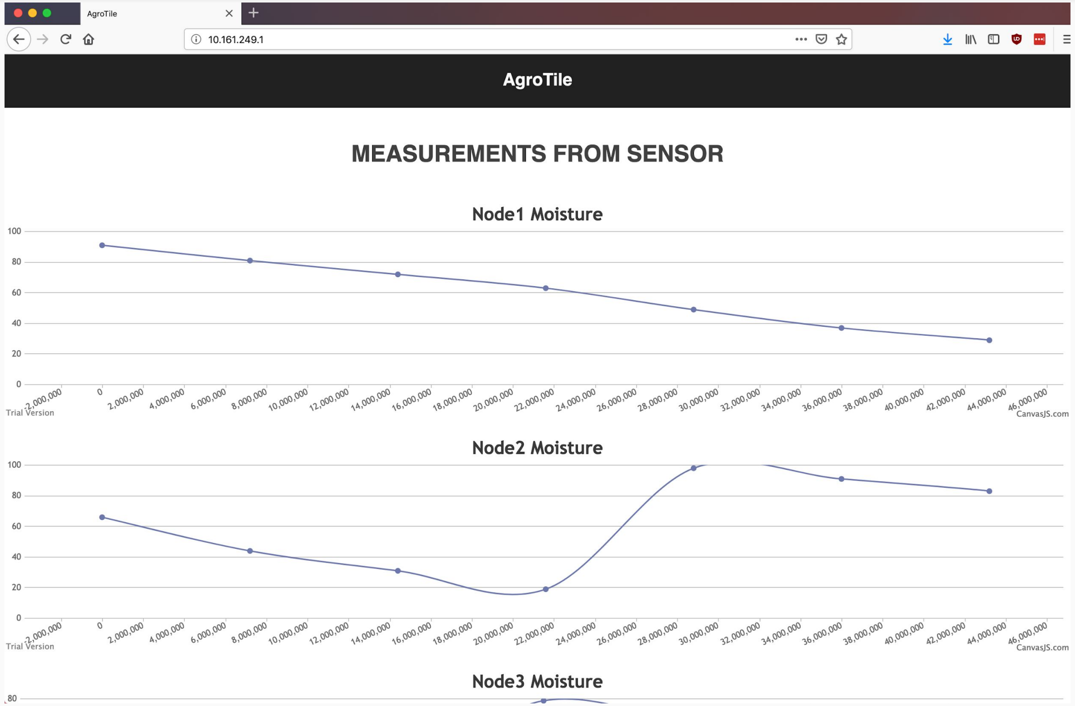


Accessing Measurements

- By connecting to the mesh WiFi network and sending an HTTP request to the server’s IP address the server will respond with an HTML page containing graphs of each node’s moisture history



Webpage


AgroTile
10.161.249.1
AgroTile
MEASUREMENTS FROM SENSOR
Node1 Moisture
Node2 Moisture
Node3 Moisture
Trial Version
CanvasJS.com

Results

- The root esp32 was able to maintain the mesh, quickly connect/update topology, store node messages, maintain a web server
- Battery was significantly saved by sending sleep signals to the nodes in the mesh. (10 hours constantly running to 125 hours with 2 hour sleep)
- The root esp32 was not able to maintain many HTTP connections concurrently.

Conclusion

- Mesh works well
  - Quickly connect and update topology
- Power consumption was an issue
  - Mitigated by sending sleep signals
- Difficult for node server to maintain TCP connections while also maintaining the mesh

Conclusion

- A single ESP32 can act as the root of a wireless mesh, store mesh node messages, and maintain an HTTP server
- Aim towards seeing how much power we could squeeze out of a single $10 esp32
- These results show that ad-hoc esp32 networks can serve as an entire IoT system by communicating, hosting, and storing messages without the need for more expensive equipment such as a Raspberry Pi or an AWS server

Future Work

- Areas of future work:
    - BLE (Bluetooth Low Energy) Mesh
    - Temperature & Humidity Detection
    - Machine Learning
    - Dedicated Server
    - Persistent Database
    - Lighterweight protocol than HTTP (CoAP)

Roles

- Web Server Node & Front-end
  - Kevin & Shachi
- Client Node, Mesh, Communication Interface
  - Matt & Esai
- Research, Paper, Presentation
  - Shachi, Esai, Kevin, Matt