Multi-Hop communication using ESP32

01 December 2018

Abhinav Khanna (2017csb1061) , Abhishek Yadav (2017csb1063) , Amritpal Singh (2017csb1068) , Divyanshu (2017csb1074)

Indian Institute of Technology Ropar
Punjab
India

Overview

In our project, we have successfully implemented a static network of ESP32 modules, which collect varied types of data, such as temperature and humidity, which is then collected and displayed on a central server. Since boards will be spread across a large area, each one will have to function as a router and data collector. This project can be used extensively in large buildings, so that the presence/absence of people in a particular room can be detected without having to manually check the room. This will save a lot of resources and manpower.

Goals

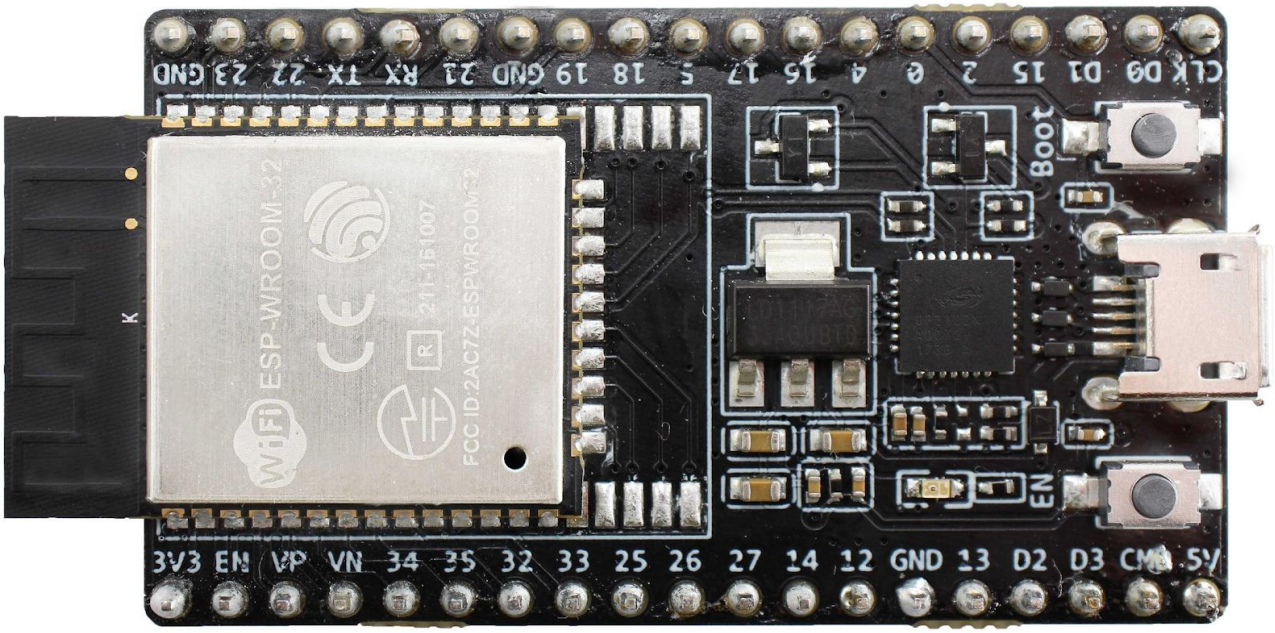
1. To send data from one ESP32 module to another using master slave communication schematic.
2. Ensure that each ESP32 switches timely from to master mode to slave mode, and vice versa.
3. To implement a mesh network of ESP32 modules, so as to collect data of each node is collected on a central server.
4. To ensure that the data on the server is updated consistently without fail, after a fixed amount of time.

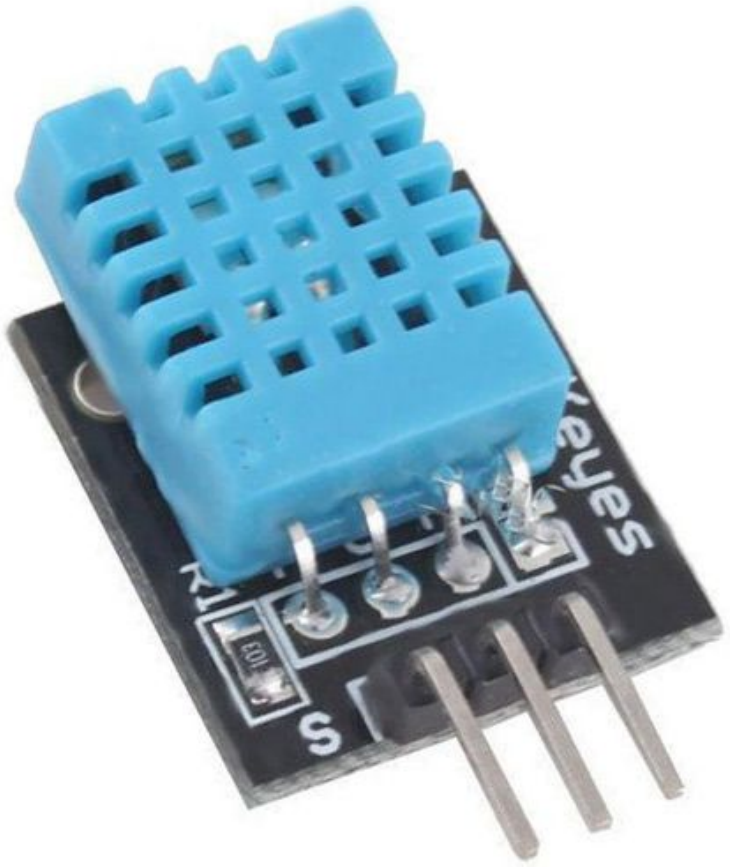
Specifications

We have used an ESP32 module as a base node. The ESP32 module was preferred over other arduino boards such as Arduino uno because it provides both bluetooth and WIFi access.

We have also used temperature and humidity sensor (DHT11) as data to be sent through our network.

Note: In the following text, we will be using the master slave terminology quite often. It must be noted that each master node refers to the node which is sending data, and slave refers to the node which receives data. This can often seem counter-intuitive.

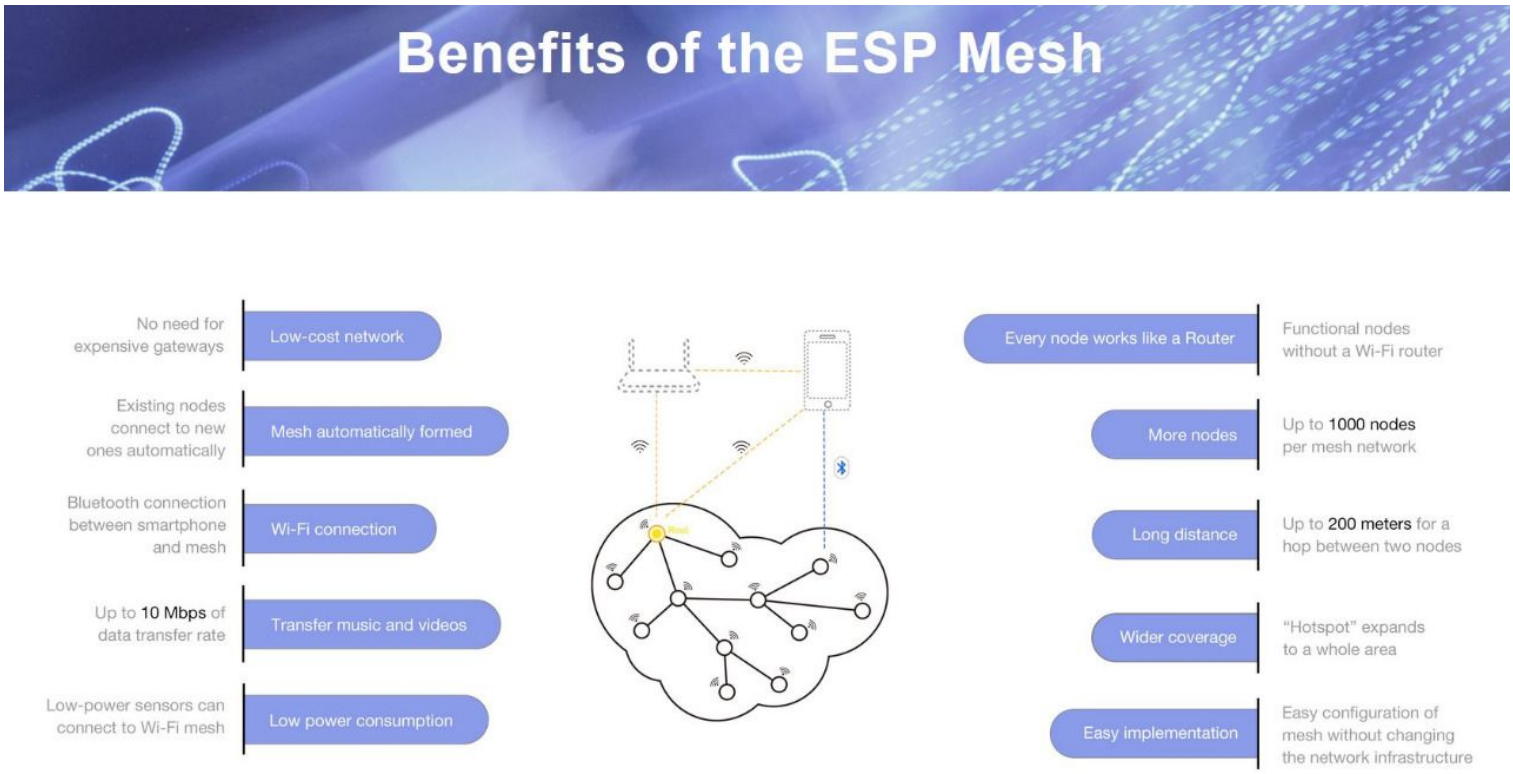




Milestones

I. Mesh network

The network is set-up as a mesh/tree, such that each node of the tree sends data. This basically implies that the mesh is not just any random collection of nodes sending data to one another. It is, in fact a tree type network. What this means is that each node is actually on a certain level, in reference to the root node. Data is sent from one node to it's parent node, and this process repeats itself until data reaches the root node and then the server itself.



II. Multi-Hop communication

Since data from, say, the root passes through multiple nodes, hence the name Multi-Hop communication. The underlying theme is that every node in the mesh, except the root node, acts both as a master and a slave. This requires that the node first act as a slave i.e. receive data from it's children, and then turn into a master, and send data to its own parents. This requires that the programme of both master and slave be loaded into the module, and that the module itself alternates between the two states.

III. Establishing webserver using ESP32

We use one ESP32 module as the root node, and in our case this also happens to be the server. The data which is collected in the root node is displayed on a device with which the root node is connected, and the serial monitor. Please note that we are not using computer as a server and thus we are not storing any data since we have not yet made a database for storing the data, but only displaying the currently acquired value. The same can be done by writing python script which collects data from a webpage and stores it.

IV. Establishing ESP32 as a Soft AP (Software enabled access point)

SoftAP is an abbreviated term for "software enabled access point". This is software enabling a computer/device which hasn't been specifically made to be a router into a wireless **access point**. It is often used interchangeably with the term "virtual router". ESP32 is not a router, but we coded it in such a manner that it behaves as an access point.

V. Static Implementation

The structure of tree network is pre decided and does not change. Once set, we cannot change the network structure until we change the code. We managed to achieve this implementation of ESP32 boards and we managed to collect data (in our case, temperature, humidity, path, name of ESP32 board from where data is coming and location of node).

Project Extension

Dynamic Implementation

The structure of tree network is not decided beforehand. ESP-MESH is self-organizing and self-healing meaning the network can be built and maintained autonomously. In the given time frame, we could only achieve static implementation where we ourselves set the level number of the nodes (ESP32). But in our implementation, the node chooses its parent on it's own, but the level of each node is fixed, hence making it a static network. Just like the node chooses its own parent, the code can be modified such that each node chooses its own level, thus making it a dynamic network.