ASHESI UNIVERSITY

DESIGN OF A RELIABLE BUT LOW-COST WIRELESS INDUSTRIAL NETWORK SYSTEM USING A BLUETOOTH LOW ENERGY MESH NETWORK

BSc. Electrical and Electronic Engineering

By

Nana Adum Kwabena Djan

2020

ASHESI UNIVERSITY COLLEGE

DESIGN OF A RELIABLE BUT LOW-COST WIRELESS INDUSTRIAL NETWORK SYSTEM USING BLUETOOTH LOW ENERGY MESH

THESIS

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Electrical and Electronic Engineering.

Nana Adum Kwabena Djan

2020

Declaration

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……………………………………………………..

Candidate's Name:

Nana Adum Kwabena Djan

Date:

29th May, 2020

I hereby declare that the preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

………………………………………………………..

Supervisor's Name:

Kofi Adu-Labi

Date:

29th May, 2020

Acknowledgement

I am grateful to everyone whose encouragement and intellectual advice contributed to the completion of this project. I am grateful to my family for their prayers and their support during the entire thesis period and supporting me emotionally and financially throughout my undergraduate studies. I would like to thank my supervisor, Kofi Adu-Labi (MSc.) for his support and guidance throughout this research period and helping me finetune this paper. He also purchased additional components for the project I could borrow and for that I am also very grateful. Lastly, I would like to thank the all Ashesi Staff and Faculty who have taught me and have imparted to me the skills and knowledge I have used to finish up this project.

Abstract

Many modern industrial plants have sensor actuator networks that allow operators on an industrial plant to receive data from sensors and send information to Actuators to operate plants efficiently. Most energy plants in Ghana such as Sony Asogli and Volta River Authorities Tema Thermal complex use guided media as a means of data communication. Guided media which is wired data communication has a couple of drawbacks. Problems with guided media include flexibility in re-arranging the sensor actuator topology in the plant. Another issue is the initial cost of wiring an entire plant. To make data communication between sensor actuators and operators on a plant more flexible in deployment the use of some suitable unguided media must be applied. Bluetooth Low Energy (BLE) mesh was the selected wireless media protocol due to its wide range, low power consumption constraints and interoperability due to a good number of devices having Bluetooth capability. My setup included multiple ESP 32 node MCU microcontrollers and a BLE compatible phone. Data from sensors could be received over long distance in real time using a flood mesh hopping algorithm. The experiment was a success and proved that BLE mesh can be used as a replacement for the traditional wired sensor networks.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background

Many industries are looking for cost-effective, reliable, and efficient ways to monitor, control and automate machinery and actuators on their plants. To monitor sensors and control actuators on a plant a communication protocols such as a machine to machine (M2M) protocol may be used [1].

Wiring a plant or factory that spans over kilometres of land will lead to increased cost [2]. Fortunately, there are a number of wireless communication protocols that can be used to communicate between sensors and machines on the plant but unfortunately, these wireless communications technologies have their limitations. In this thesis I will compare different wireless communication protocols and choose the one that is most suitable for Industrial Wireless Sensor Actuator Networks (IWSAN) [3].

1.2 Problem Statement

High performing industries and processing plants consist of hundreds of autonomous and connected machines and sensors which are in constant need of monitoring. Currently, these industries are looking for the most cost-effective way of providing Machine to Machine (M2M) communication between devices so that they can be monitored, controlled, and automated. There are many IoT protocols that can be used to wirelessly connect machines and sensors to each other, however, the key to finding the best M2M communication lies in finding a connection with maximum security, low energy and a wide range.

1.3 Objectives of the Project Work

The objective of this project is to test the effectiveness of the BLE mesh network in the control and monitoring of internal and external lighting together in addition to sensors carrying out preventive maintenance on a machine in the plant such as a motor. This paper seeks to fully understand the potential and the limitations of the BLE mesh system in the context of a factory set-up.

1.4 Expected Outcomes of the project work

The project is expected to prove that the Bluetooth Low Energy mesh protocol can seamlessly communicate between sensor nodes efficiently: i.e.

- Reliably
- At a low cost
- Across a wide range and
- With low latency.

1.5 Justifications and motivation for topic

BLE is a relatively new technology that promises many to many connections between Bluetooth embedded systems. It is also a very low-cost low energy consumption system. There has been the quest to find a suitable wireless protocol for machine to machine communication in the industrial setting. It turns out that BLE mesh can be used for various applications such as asset tracking, sensor monitoring, and light control. Also, the range of BLE mesh can be scaled up to as many devices as possible because of its mesh technology protocol. With the use case scenarios, meshes are very applicable in the industrial setting where sensor monitoring for machines and asset tracking is a large concern for business owners.

1.6 Scope of the project

This project seeks to build and test a wireless communication system model using Bluetooth Low Energy (BLE) for an industrial setting. The user should be able to monitor sensor data using BLE mesh and actuate a valve or a switch on the plant using the BLE mesh protocol. Not only do we want to receive and send data to machinery on the plant, but we would also want to communicate with these devices in real time.

Chapter 2

2.1 Literature review: What has been done?

Communication Technology used on Industrial plants include LoRa which has a communication range of 100m – 3km. We also have Ethernet-GPRS which has a range of 3km and greater [4]. A cluster of Zigbee nodes can also be used where the range of an individual Zigbee node is 100 meters [4]. RFID and Radar is another technology that has been for data acquisition on industrial plants [5]. A group of Engineers in their research findings were successfully able to monitor and control a biogas plant using GSM as their communication protocol [6].

2.2 About Bluetooth communication

The use of Bluetooth Classic (Bluetooth version 4.0) as a communication protocol has also been attempted. In this case the plant operator needs to take a device within Bluetooth classic range which is 10m or less to receive sensor data regarding a motor on the plant [5].

The special interest group further developed Bluetooth version 4.0 to Bluetooth version 5.0. This version of Bluetooth is also called Bluetooth Low energy. The upgrade to version 5.0 has twice the bandwidth of Bluetooth 4.0[7]. In comparison to 1Mbps speed in Bluetooth classic, Bluetooth Low Energy has speeds of up to 2Mbps [7]. The message capacity for BLE is 255 bytes per transmission which is significantly larger than Bluetooth which has 17 to 20 bytes for payload [7].

2.3 About Bluetooth Low Energy Mesh

A major setback of the BLE is the limiting range of about 10 meters. To solve this issue the Bluetooth SIG developed BLE mesh. A protocol that has multi-hop characteristics enabling data to jump from one BLE node to another until data packets reach their destination. BLE mesh uses radio frequency of 2.4GHz this frequency is known as the Industrial Scientific and Medical Band a free frequency for data communication [8].

Also, the BLE mesh protocol can be preventive maintenance. A study has been done on machine fault diagnosis by collecting data on the vibrational patterns of a machines using Fast Fourier Transform (FFT) analysis [9]. In this research we see the importance of having a wireless network to obtain necessary data from a machine especially when it is a mobile machine which makes traditional wiring inconvenient for use.

Another application of Bluetooth mesh in the industry is to help improve the wireless network range of a given location [10]. An advantage of the Bluetooth mesh network that has come in very handy in the Industrial Wireless Sensor Networks (IWSN) Is its self-healing characteristics [11]. This has given Bluetooth mesh greater reliability and flexibility amongst Industrial Wireless Sensor Network solutions. Mesh networks are self-healing because redundant nodes can re-route messages since it uses a flood messaging protocol. Also, new sensors in the form of nodes can be added to the network seamlessly. The network will self-adjust to take on the load.

Wireless technology for the internet of things must have low latency and high reliability and with the development of Bluetooth mesh protocol both parameters can be achieved [12].

2.4 Challenges

Industrial conditions tend to be unfavourable for wireless network systems because of the different electrical noises generated by machinery on the plant that can interfere with the chosen communication protocol [8]. Additionally, multiple communications protocols such as Wi-Fi and BlE mesh which are both operate in the 2.4GHz range can be running concurrently on a plant and when the two signals collide it may lead to signal interference.

2.5 Other potential applications of BLE mesh

Researchers have investigated and used the BLE mesh protocol for office lighting [13]. They found out that the managed flooding protocol used by BLE mesh makes it very reliable [13]. No routing tables in managed flooding lead to real time data transmission in smaller networks because no routing calculations occur before data transmission.

Chapter 3: Design

A lot of focus goes into the design of a wireless sensor network. In this chapter the primary focus is going to be the design decisions involved in setting up a wireless sensor actuator network. This chapter looks at pre-existing wireless sensor actuator networks and their advantages and disadvantages with the help of a Pugh matrix.

3.1 Design Objectives

The project seeks to develop a low-cost industrial control and monitoring system with a good wide area range and very little latency in transmitting information as well as low power consumption for these sensor nodes. The system must also be user-friendly and easy to learn.

3.2 Review of existing designs

Existing designs in Ghanaian Industries for example Tema Thermal Complex are wired communication systems that communicate data from sensors to the central system. Protocols used in the Industry are SCADA (Supervisory Control and Data Acquisition). Other communications used in the industrial sector are Wi-Fi, GSM and LoRa.

3.3 Design Decisions for wireless communication

A primary design decision for this project was choosing a wireless communication technology. The wireless media chosen after extensive research was BLE mesh. Its low latency and strong security features make this protocol very robust and resilient to external attackers and intruders.

Pugh Matrix

		Wi-Fi		Lora	GSM	Bluetooth mesh
criteria	concept	Baseline	weight			
1	Cost	0	2	+1	+1	+1
2	Range	0	2	+1	+1	+1
3	Reliability	0	2	0	-1	+1
4	power saving	0	1	+1	0	+1
5	responsiveness	0	2	0	0	+1
Total				5	2	11

Table 1: Pugh matrix of communication protocol

Design Decision for Microcontroller

		STM32WBx5		ESP 32	CYW20820	NRF51822
criteria	concept	Baseline	weight			
1	Cost	0	2	+1	0	0
2	Power consumption	0	1	+1	-1	-1
3	Flash memory	0	1	-1	+1	+1
4	Peripherals	0	1	+1	0	0
Total				3	0	0

Table 2: Pugh matric of microcontrollers

3.4 System Requirements

3.4.1 Functional Requirements

This section states what the system designed will do:

The system should be able to actuate lights and other industrial devices such as industrial motors.

i) The system should be able to gather temperature data from devices

ii) The system should have remote control via a phone app

iii) The system should have remote monitoring via a phone app

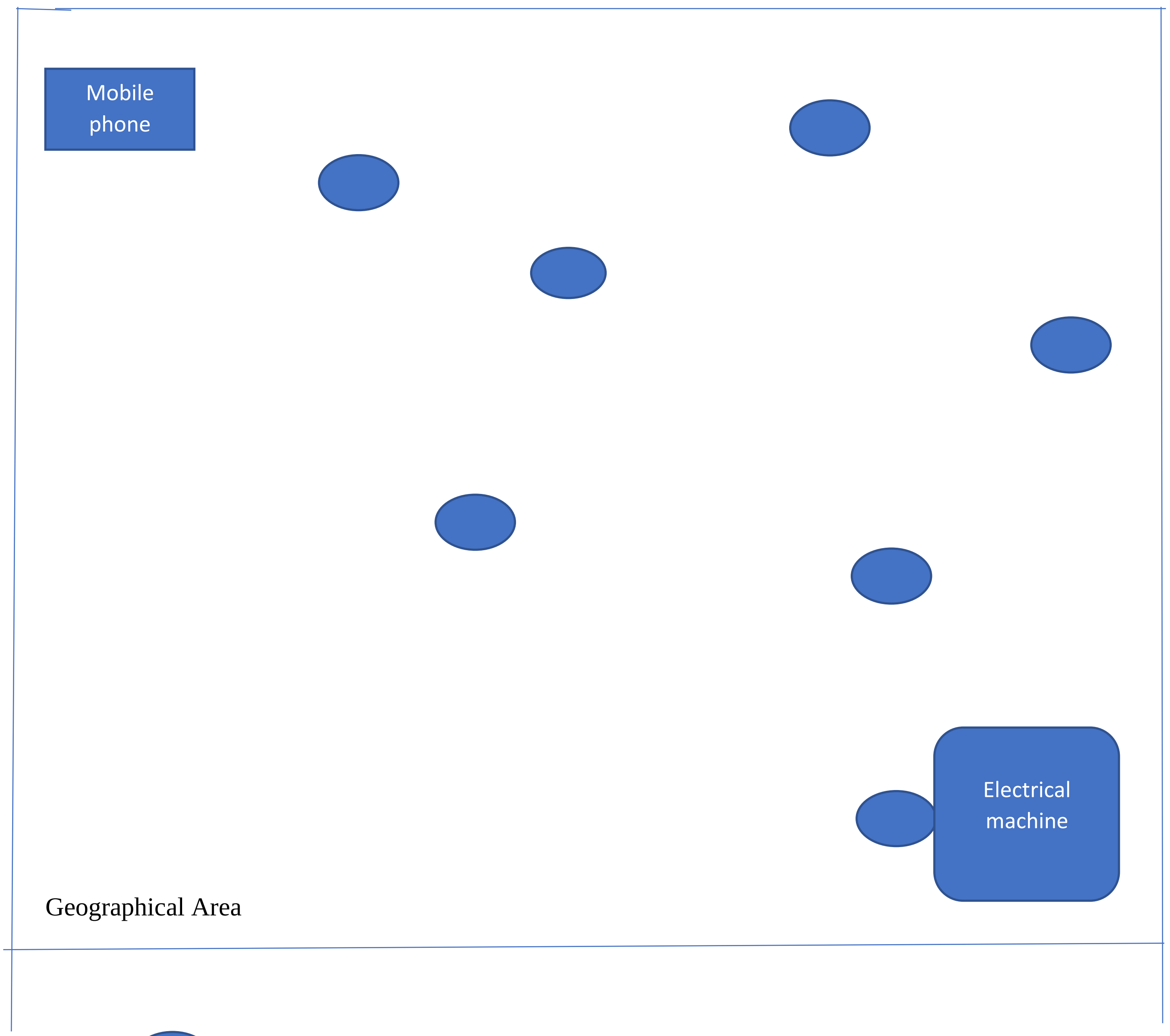
3.4.2 Non-functional Requirements

i) The system should be able to actuate a device at a 500-meter range or less.

ii) The system should be able to read sensor data or actuate a device in real-time.

iv) The system should have low power consumption capabilities. It should be able to run on a dry cell battery for at least a month.

3.5 System Block diagram



- ● - A relay node

Figure 1 : system block diagram

Figure 1 shows a simple sketch of the layout and in this diagram all nodes have relaying capabilities so data is being routed in the network from an electrical machine to a mobile device.

Chapter 4: Methodology and Implementation

4.1 Experimental Setup

This project seeks to prototype wireless Data communication in an industrial setting using BLE mesh. The system consisted of ESP 32 nodes and lightbulbs which will be actuated by relays for lighting control.

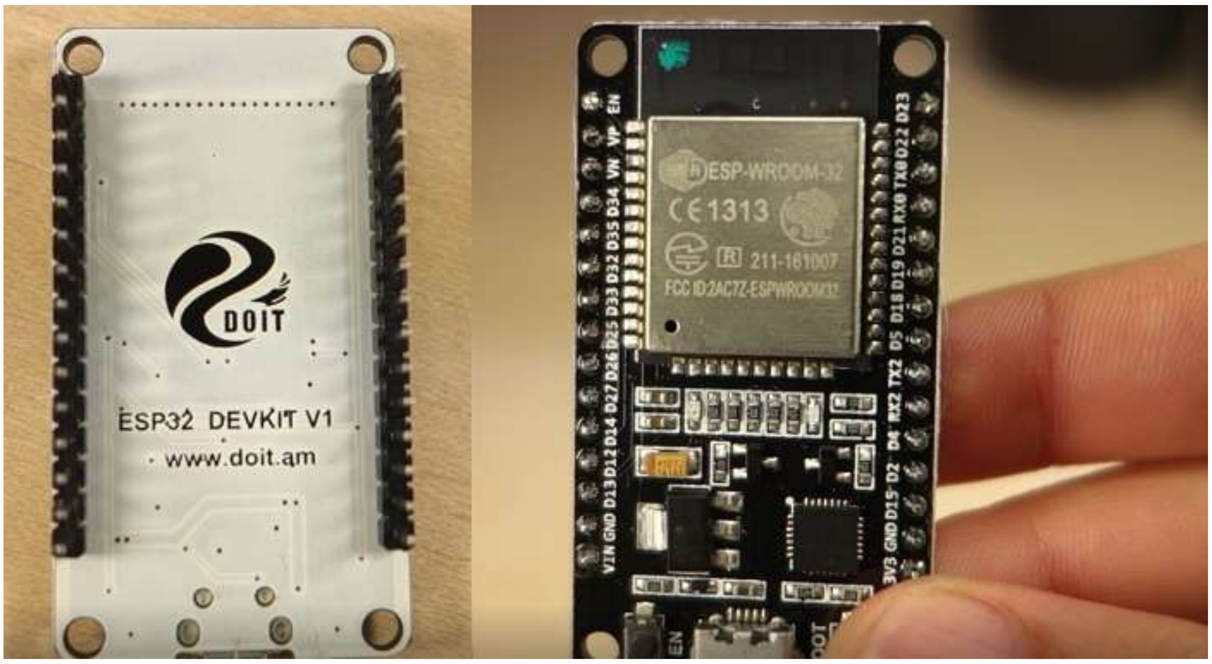


Figure 2: A picture of microcontrollers used (randomnerdtutorials,2016)

4.2 BLE Mesh Setup

The BLE mesh setup was implemented by using ESP 32 Node MCUs which have BLE mesh capabilities. Relays together with light bulbs were used to prototype the lighting control system. A client server model was used as a means of communication between the smart phone and the BLE Mesh node. The smartphone acted as a client issuing requests such as asking for the state of a light bulb i.e. high (on) or low (off). The client was also able to transmit signals in the form of a 1 (high) or 0 (low) to turn on and off a specific light bulb or motor in the plant. Step 1: Power BLE mesh node with a power source

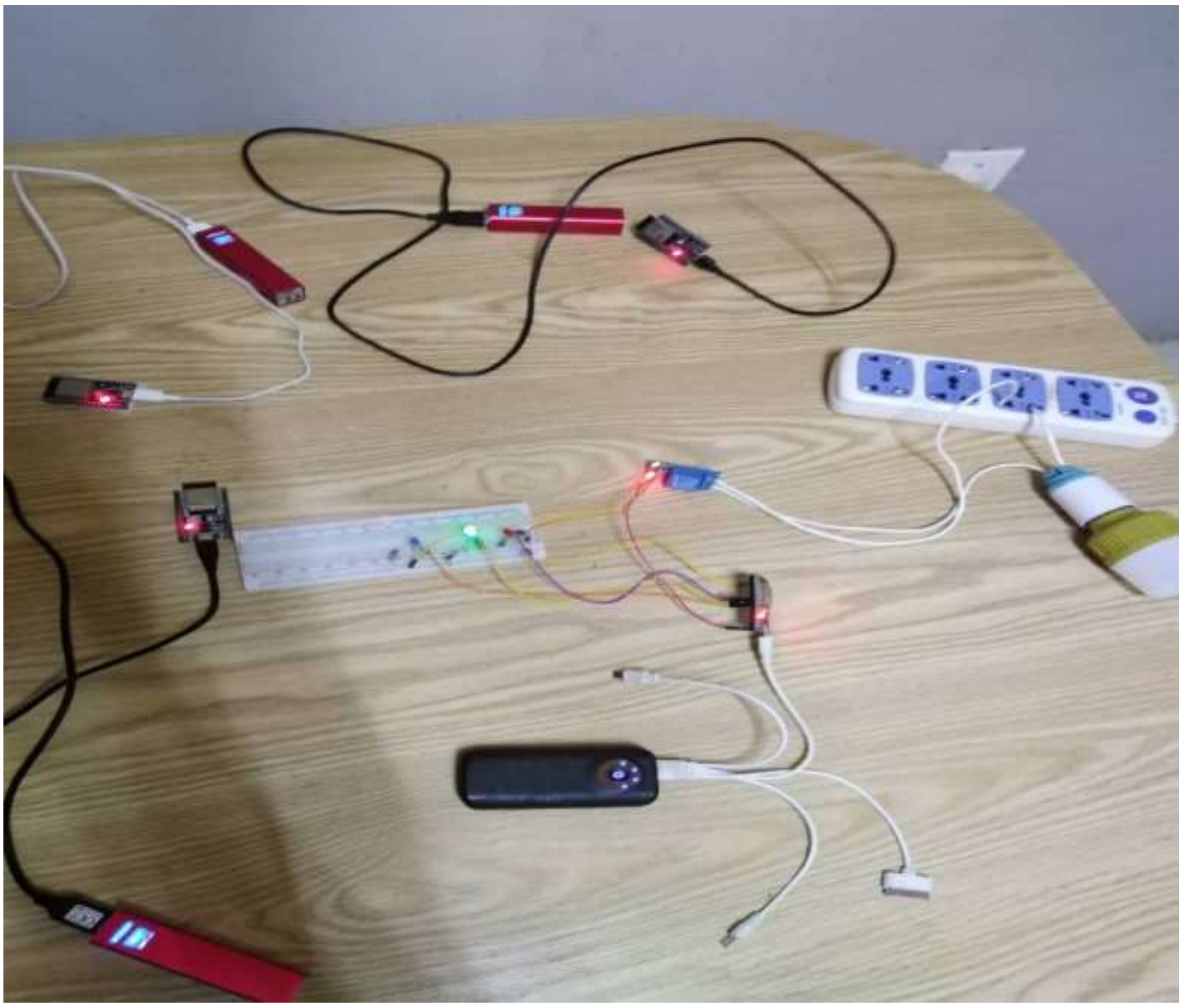


Figure 3: A picture of setup

Through the transmission of data to a node in the mesh network, we can actuate a machine connected the given node by sending a one to turn the machine on and a zero to it off.

Apart from relays serving as actuators to turn light bulbs on and off, the BLE mesh network also uses sensors to track assets and monitor industrial equipment. Resources can have tags that emit BLE energy helping operators in the industry know where every mobile device or machine is on the plant for safety purposes. Also, gyro sensors were fixed on motors to prototype how these sensors will sense data of vibrational patterns. This data can be sent to the cloud and analysed to check if that motor needs maintenance or not.

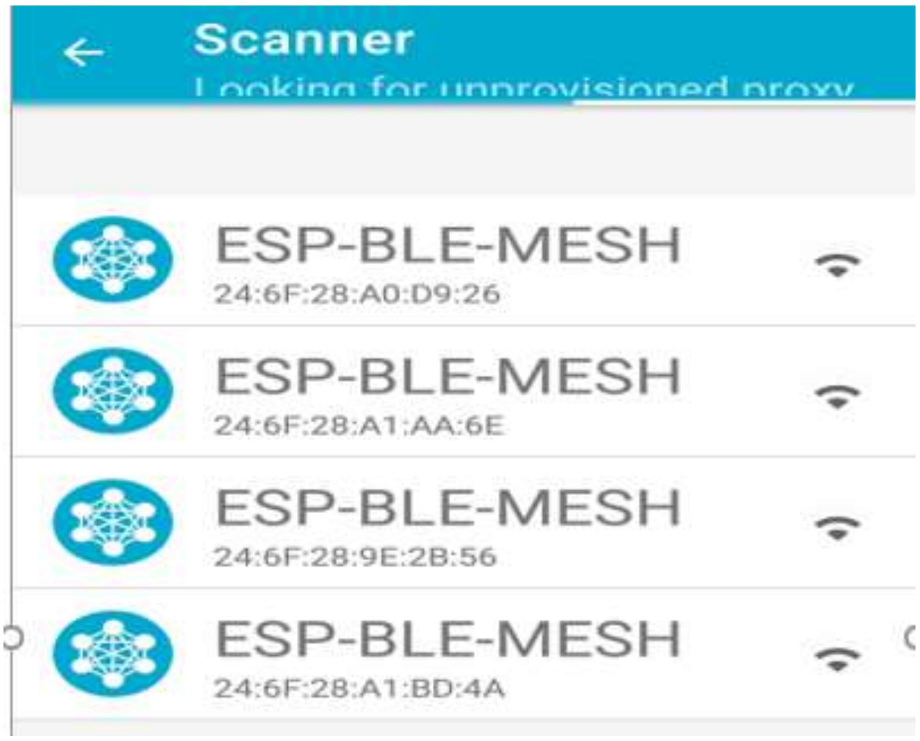
4.3 Software Setup

Software used to upload firmware unto the ESP 32 was C code programmed n Espressif's IoT Development Framework (IDF). See algorithm used the appendix. Using GitHub's repository called the On-Off server example code I was able to flash the code for switching lights on-off and read the state of light bulbs. An app called nrf-mesh from Nordic USA was used as a client to fire events such as turning lights off and on. The nrf-mesh app was also able to tell me the state of the BLE mesh nodes.

4.3.1 Phone application setup

Download the nRF mesh phone app

Scan for node

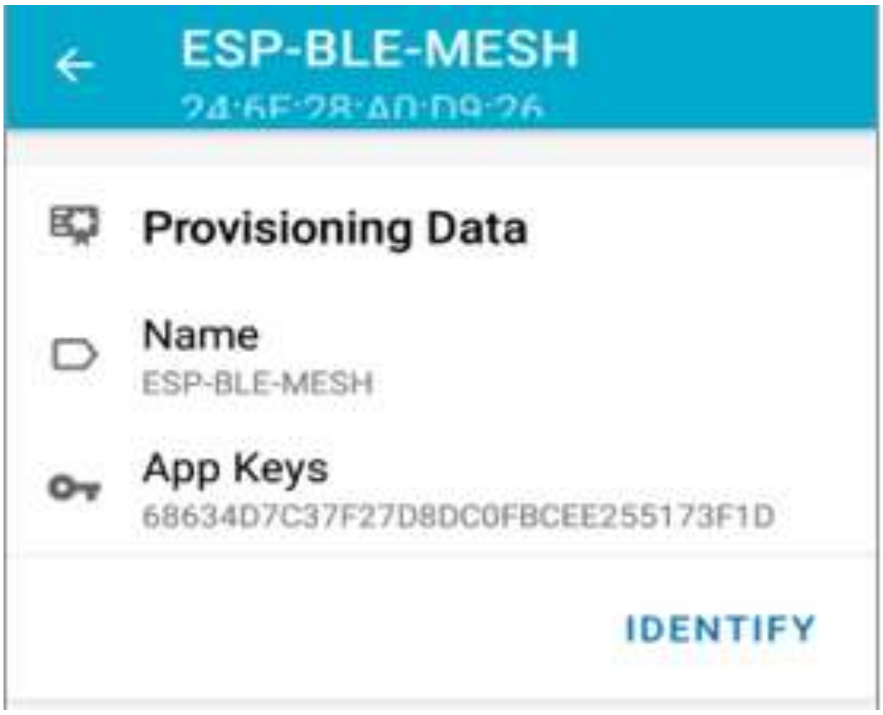


ESP-BLE-MESH
24:6F:28:A0:D9:26

ESP-BLE-MESH
24:6F:28:A1:AA:6E

ESP-BLE-MESH
24:6F:28:9E:2B:56

ESP-BLE-MESH
24:6F:28:A1:BD:4A

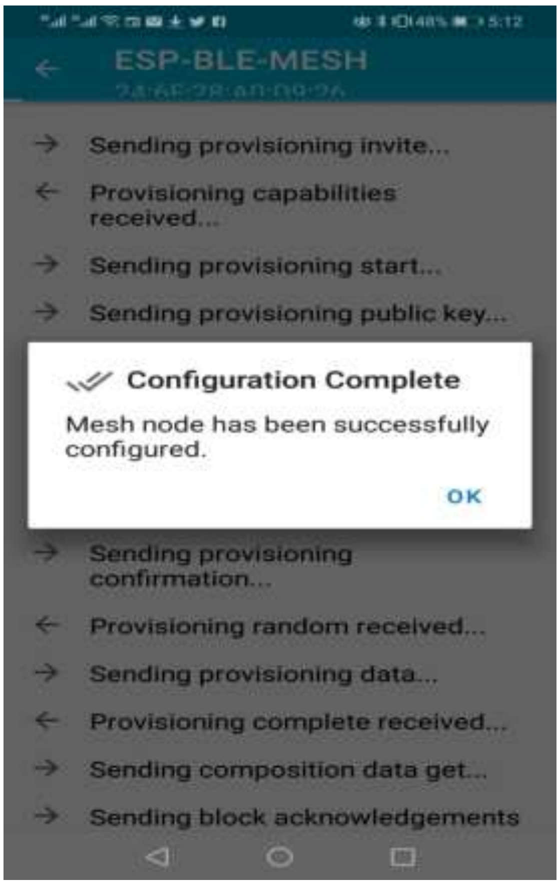

ESP-BLE-MESH
24:6F:28:A0:D9:26

Provisioning Data

Name
ESP-BLE-MESH

App Keys
68634D7C37F27D8DC0FBCEE255173F1D

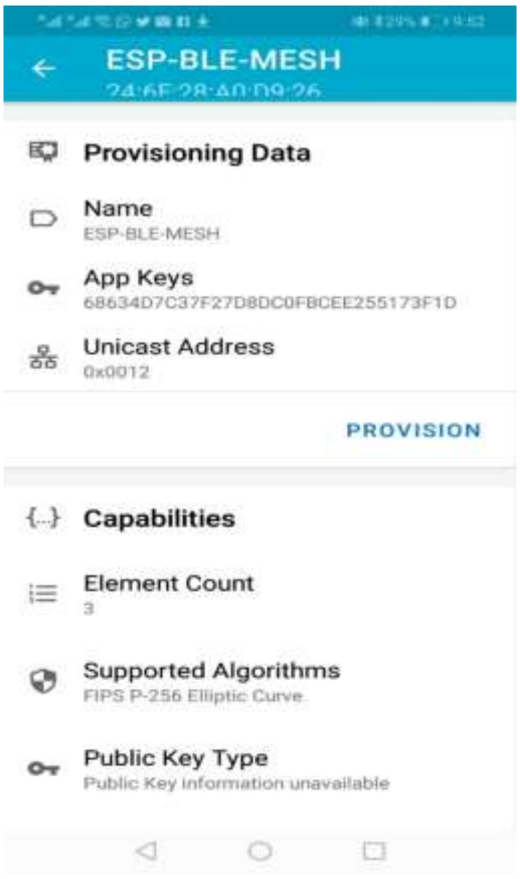
IDENTIFY


ESP-BLE-MESH
24:6F:28:A0:D9:26

→ Sending provisioning invite...
← Provisioning capabilities received...
→ Sending provisioning start...
→ Sending provisioning public key...

Configuration Complete
Mesh node has been successfully configured.
OK

→ Sending provisioning confirmation...
← Provisioning random received...
→ Sending provisioning data...
← Provisioning complete received...
→ Sending composition data get...
→ Sending block acknowledgements


ESP-BLE-MESH
24:6F:28:A0:D9:26

Provisioning Data

Name
ESP-BLE-MESH

App Keys
68634D7C37F27D8DC0FBCEE255173F1D

Unicast Address
0x0012

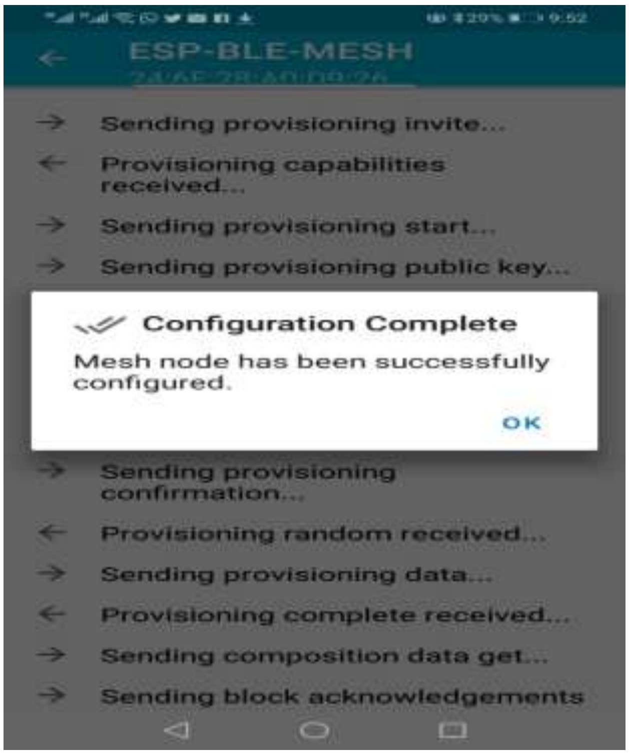
PROVISION

Capabilities

Element Count
3

Supported Algorithms
FIPS P-256 Elliptic Curve

Public Key Type
Public Key information unavailable



ESP-BLE-MESH

[illegible]

→ Sending provisioning invite...

← Provisioning capabilities received...

→ Sending provisioning start...

→ Sending provisioning public key...

Configuration Complete

Mesh node has been successfully configured.

OK

→ Sending provisioning confirmation...

← Provisioning random received...

→ Sending provisioning data...

← Provisioning complete received...

→ Sending composition data get...

→ Sending block acknowledgements

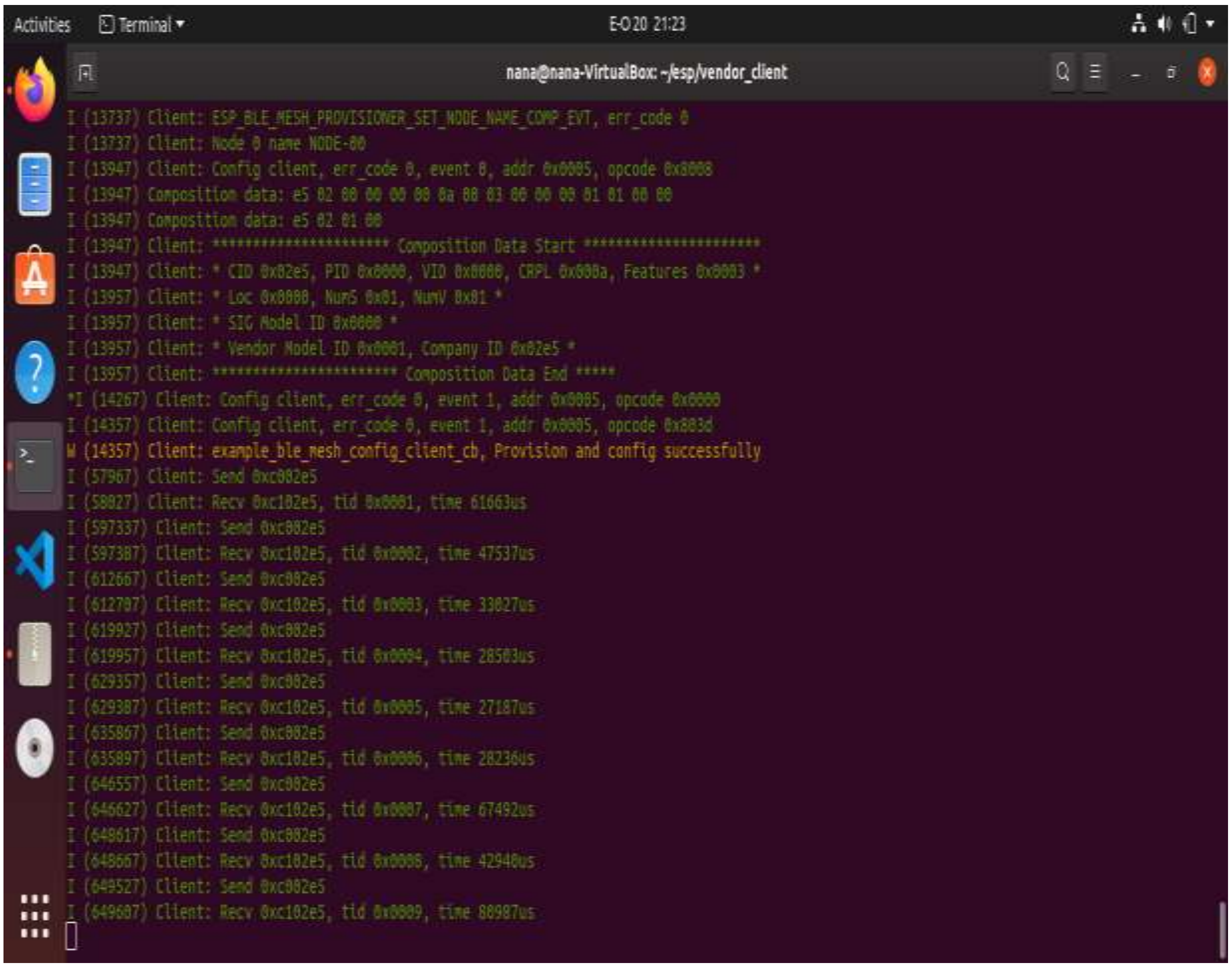
Figure 4 : A picture of mobile app features

The pictures in *Figure 4* show how the NRF-mesh app which is used to provision BLE nodes. Firstly, open app and scan for un-provisioned devices emitting BLE. Afterwards select the device and hit the identify button to provision the BLE node. After provisioning the node, it is given a unicast address. The node the network as an on-off server or a relay node depending on the functionality of the firmware uploaded unto the ESP-32 node. By choosing the unicast address of the node in the network the node can be accessed and controlled and in the case of the on-off server we can turn a GPIO pin high or low.

Chapter 5 testing and results

5.1 Latency test and results

Latency of signal transmission between two nodes using BLE mesh

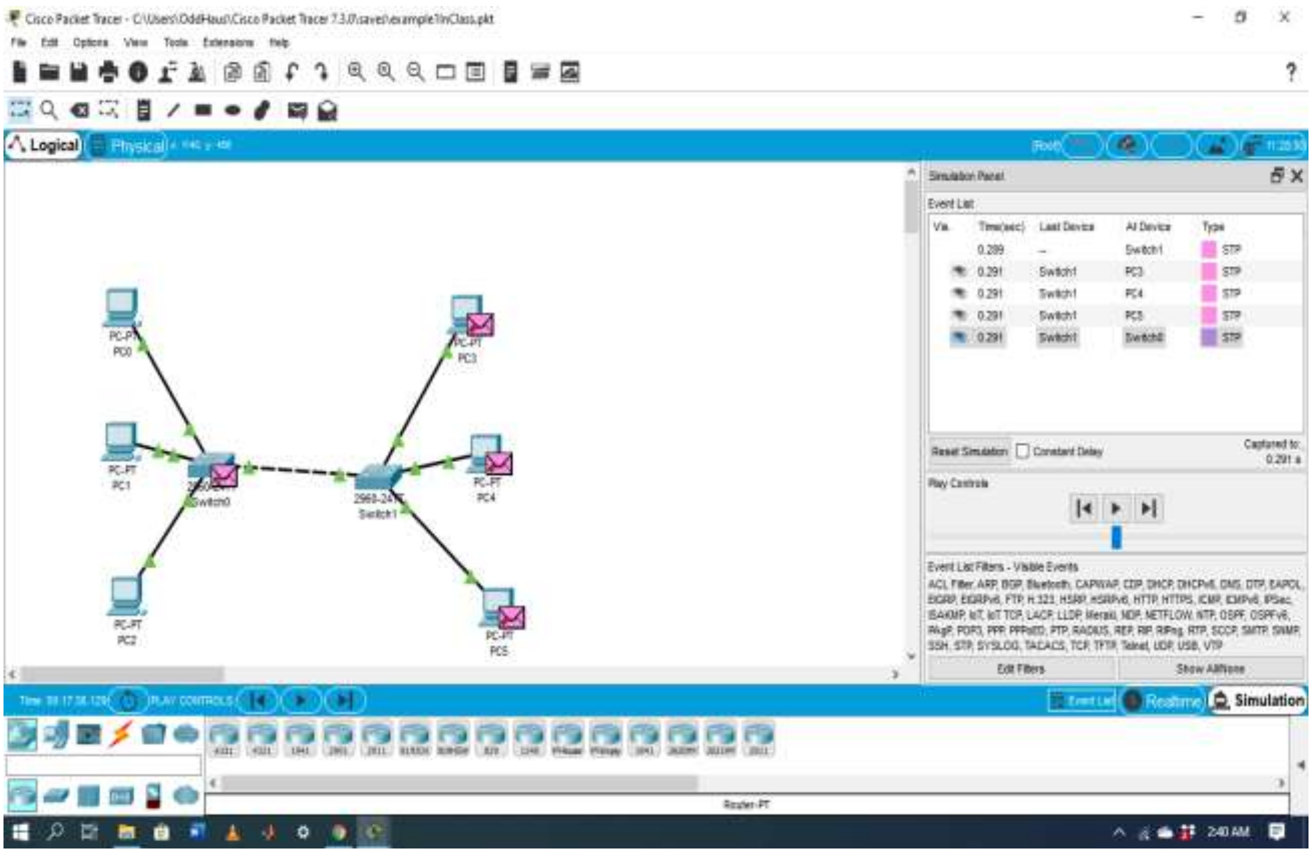


```
I (13737) Client: ESP_BLE_MESH_PROVISIONER_SET_NODE_NAME_COMP_EVT, err_code 0
I (13737) Client: Node 0 name NODE-00
I (13947) Client: Config client, err_code 0, event 0, addr 0x0005, opcode 0x8008
I (13947) Composition data: e5 02 00 00 00 00 0a 00 03 00 00 00 01 01 00 00
I (13947) Composition data: e5 02 01 00
I (13947) Client: ********************** Composition Data Start **********************
I (13947) Client: * CID 0x02e5, PID 0x0000, VID 0x0000, CRPL 0x000a, Features 0x0003 *
I (13957) Client: * Loc 0x0000, NumS 0x01, NumV 0x01 *
I (13957) Client: * SIG Model ID 0x0000 *
I (13957) Client: * Vendor Model ID 0x0001, Company ID 0x02e5 *
I (13957) Client: *********************** Composition Data End *****
*I (14267) Client: Config client, err_code 0, event 1, addr 0x0005, opcode 0x0000
I (14357) Client: Config client, err_code 0, event 1, addr 0x0005, opcode 0x803d
W (14357) Client: example_ble_mesh_config_client_cb, Provision and config successfully
I (57967) Client: Send 0xc002e5
I (58027) Client: Recv 0xc102e5, tid 0x0001, time 61663us
I (597337) Client: Send 0xc002e5
I (597387) Client: Recv 0xc102e5, tid 0x0002, time 47537us
I (612667) Client: Send 0xc002e5
I (612707) Client: Recv 0xc102e5, tid 0x0003, time 33027us
I (619927) Client: Send 0xc002e5
I (619957) Client: Recv 0xc102e5, tid 0x0004, time 28503us
I (629357) Client: Send 0xc002e5
I (629387) Client: Recv 0xc102e5, tid 0x0005, time 27187us
I (635867) Client: Send 0xc002e5
I (635897) Client: Recv 0xc102e5, tid 0x0006, time 28236us
I (646557) Client: Send 0xc002e5
I (646627) Client: Recv 0xc102e5, tid 0x0007, time 67492us
I (648617) Client: Send 0xc002e5
I (648667) Client: Recv 0xc102e5, tid 0x0008, time 42940us
I (649527) Client: Send 0xc002e5
I (649607) Client: Recv 0xc102e5, tid 0x0009, time 80987us
```

Figure 5: A picture of roundtrip time of data packets between two nodes

From the figure above we see that the maximum round-trip time it takes for data to move from a client node to the server is node is 80987 micro-seconds, which is 0.08 seconds.

Latency test in Wi-Fi using Cisco Packet Tracer



Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.289	--	Switch1	STP
	0.291	Switch1	PC3	STP
	0.291	Switch1	PC4	STP
	0.291	Switch1	PC5	STP
	0.291	Switch1	Switch0	STP

Reset Simulation ☐ Constant Delay Captured to: 0.291 s

Play Controls

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters | Show All/None

Figure 6 : simulation of data transmission period for Wi-Fi.

The average time it takes for messages to move from router to the node is 0.291 seconds.

Latency analysis between BLE mesh and Wi-Fi

BLE mesh had a latency period of 0.08 seconds. Wi-Fi has a latency of 0.3 seconds. From these results we can see BLE mesh transmission time is faster than that of Wi-Fi.

5.2 Range

Bluetooth classic (IEEE 802.15.1) has a nominal range of 10 meters while IEEE 802.15.11a/b/g (Wi-Fi) has a nominal range of about 100 meters [15]. However, BLE mesh has the capability of hoping data from one node to another using a managed flooding protocol [15]. This drastically increases the range of a BLE mesh network with comparison to the traditional mesh network.

Chapter 6: Conclusion

Conclusion

Quantitative analysis proves that Bluetooth Low Energy has an advantage over other forms of unguided media. Another reason BLE mesh will be recommend as an IWSN communication protocol for an Industrial setting is because of how ubiquitous BLE mesh compatible devices are especially because mobile phones have BLE. Bluetooth has a large ecosystem of devices that are Bluetooth compatible.

Limitations

A BLE mesh network in large areas is made up of many nodes and this might increase initial cost of running the network although long run cost is cheaper due to low energy consumption.

Future Works

The project sought to find out the viability of BLE mesh in the industry and future works could test data transmission in more rigorous environments. BLE mesh nodes could be tested close to heavy machinery producing electrical noise. The outcome could be analyzed by the reader to get a better understanding of the effect of interference on BLE.

References

[1] M. Taneja, "Lightweight protocols for LTE M2M networks," 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Noida, 2015, pp. 743-748.

[2] A. Varghese and D. Tandur, "Wireless requirements and challenges in Industry 4.0," *2014* International Conference on Contemporary Computing and Informatics (IC3I), Mysore, 2014, pp. 634-638.

[3] A. Potsch, H. Bernhard and A. Springer, "High resolution testbed for heterogenous industrial wireless sensor and actuator networks," 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, 2018, pp. 1-9.

[4] T. Addabbo et al., "An IoT Framework for the Pervasive Monitoring of Chemical Emissions in Industrial Plants," 2018 Workshop on Metrology for Industry 4.0 and IoT, Brescia, 2018, pp. 269-273, doi: 10.1109/METROI4.2018.8428325.

[5]R. N. Gore, H. Kour, M. Gandhi, D. Tandur and A. Varghese, "Bluetooth based Sensor Monitoring in Industrial IoT Plants," 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India, 2019, pp. 1-6, doi: 10.1109/IconDSC.2019.8816906.

[6] V. Acharya, V. V. Hegde, A. K. and M. K. M., "IoT (Internet of Things) Based Efficiency Monitoring System for Bio-Gas Plants," 2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), Bangalore, 2017, pp. 1-5, doi: 10.1109/CSITSS.2017.8447567.

[7] M. B. Yaakop, I. A. Abd Malik, Z. bin Suboh, A. F. Ramli and M. A. Abu, "Bluetooth 5.0 throughput comparison for internet of thing usability a survey," 2017 International Conference on Engineering Technology and Technopreneurship (ICE2T), Kuala Lumpur, 2017, pp. 1-6, doi: 10.1109/ICE2T.2017.8215995.

[8] C. Lu et al., "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," in Proceedings of the IEEE, vol. 104, no. 5, pp. 1013-1024, May 2016, doi: 10.1109/JPROC.2015.2497161.

[9] X. Wei, L. Hou and J. Hao, "Machine Fault Diagnosis Using IIoT, IWSNs, HHT, and SVM," 2018 IEEE 18th International Conference on Communication Technology (ICCT), Chongqing, 2018, pp. 978-982, doi: 10.1109/ICCT.2018.8600146.

[10] R. T. E. Dvinge, A. Stalmach and L. Nalpantidis, "Connection-Based Bluetooth Mesh Network as a Low Energy Solution for Off-Grid Data Networks," 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 2019, pp. 1-6, doi: 10.1109/MOCAST.2019.8741589.

[11] M. Raza, N. Aslam, H. Le-Minh, S. Hussain, Y. Cao and N. M. Khan, "A Critical Analysis of Research Potential, Challenges, and Future Directives in Industrial Wireless Sensor Networks," in IEEE Communications Surveys & Tutorials, vol. 20, no. 1, pp. 39-95, Firstquarter 2018, doi: 10.1109/COMST.2017.2759725.

[12] Y. Murillo, A. Chiumento, B. Reynders and S. Pollin, "SDN on BLE: Controlling Resource Constrained Mesh Networks," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-7, doi: 10.1109/ICC.2019.8761066.

[13] á. Hernández-Solana, D. Pérez-Díaz-De-Cerio, M. García-Lozano, A. V. Bardají and J. Valenzuela, "Bluetooth Mesh Analysis, Issues, and Challenges," in IEEE Access, vol. 8, pp. 53784-53800, 2020, doi: 10.1109/ACCESS.2020.2980795.

[14] randomnerdtutorials, 2016. Randomnerdtutorials.Com. [image] Available at: <https://randomnerdtutorials.com/getting-started-with-esp32/> [Accessed 7 May 2020].

[15] J. Lee, Y. Su and C. Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics *Society*, Taipei, 2007, pp. 46-51.

Appendix

Title: vendor_client.

Author: renzbagaporo

Availability: github

Date published: 22 March 2020

Date retrieved: 7 May 2020

```
/* main.c -
Application
main  entry
point */

              /*
               * Copyright (c) 2017 Intel Corporation
               * Additional Copyright (c) 2018 Espressif Systems (Shanghai) PTE LTD
               *
               * SPDX-License-Identifier: Apache-2.0
               */

              #include <stdio.h>
              #include <string.h>

              #include "esp_log.h"
              #include "nvs_flash.h"
              #include "esp_bt.h"

              #include "esp_ble_mesh_defs.h"
              #include "esp_ble_mesh_common_api.h"
              #include "esp_ble_mesh_networking_api.h"
              #include "esp_ble_mesh_provisioning_api.h"
              #include "esp_ble_mesh_config_model_api.h"
              #include "esp_ble_mesh_local_data_operation_api.h"

              #include "board.h"
              #include "ble_mesh_demo_init.h"

              #define CID_ESP     0x02E5
              #define CID_NVAL    0xFFFF

              #define ESP_BLE_MESH_VND_MODEL_ID_CLIENT    0x0000
              #define ESP_BLE_MESH_VND_MODEL_ID_SERVER    0x0001

              #define ESP_BLE_MESH_VND_MODEL_OP_SEND      ESP_BLE_MESH_MODEL_OP_3(0x00, CID_ESP)
```

```c
#define ESP_BLE_MESH_VND_MODEL_OP_STATUS    ESP_BLE_MESH_MODEL_OP_3(0x01, CID_ESP)

static uint8_t dev_uuid[ESP_BLE_MESH_OCTET16_LEN] = { 0x32, 0x10 };

static esp_ble_mesh_cfg_srv_t config_server = {
    .relay = ESP_BLE_MESH_RELAY_DISABLED,
    .beacon = ESP_BLE_MESH_BEACON_ENABLED,
#if defined(CONFIG_BLE_MESH_FRIEND)
    .friend_state = ESP_BLE_MESH_FRIEND_ENABLED,
#else
    .friend_state = ESP_BLE_MESH_FRIEND_NOT_SUPPORTED,
#endif
#if defined(CONFIG_BLE_MESH_GATT_PROXY_SERVER)
    .gatt_proxy = ESP_BLE_MESH_GATT_PROXY_ENABLED,
#else
    .gatt_proxy = ESP_BLE_MESH_GATT_PROXY_NOT_SUPPORTED,
#endif
    .default_ttl = 7,
    /* 3 transmissions with 20ms interval */
    .net_transmit = ESP_BLE_MESH_TRANSMIT(2, 20),
    .relay_retransmit = ESP_BLE_MESH_TRANSMIT(2, 20),
};

static esp_ble_mesh_model_t root_models[] = {
    ESP_BLE_MESH_MODEL_CFG_SRV(&config_server),
};

static esp_ble_mesh_model_op_t vnd_op[] = {
    ESP_BLE_MESH_MODEL_OP(ESP_BLE_MESH_VND_MODEL_OP_SEND, 2),
    ESP_BLE_MESH_MODEL_OP_END,
};

static esp_ble_mesh_model_t vnd_models[] = {
    ESP_BLE_MESH_VENDOR_MODEL(CID_ESP, ESP_BLE_MESH_VND_MODEL_ID_SERVER,
    vnd_op, NULL, NULL),
};

static esp_ble_mesh_elem_t elements[] = {
    ESP_BLE_MESH_ELEMENT(0, root_models, vnd_models),
};

static esp_ble_mesh_comp_t composition = {
    .cid = CID_ESP,
    .elements = elements,
    .element_count = ARRAY_SIZE(elements),
};
```

```
static esp_ble_mesh_prov_t provision = {
    .uuid = dev_uuid,
};

static void prov_complete(uint16_t net_idx, uint16_t addr, uint8_t flags, uint32_t
iv_index)
{
    ESP_LOGI(TAG, "net_idx 0x%03x, addr 0x%04x", net_idx, addr);
    ESP_LOGI(TAG, "flags 0x%02x, iv_index 0x%08x", flags, iv_index);
    board_led_operation(LED_G, LED_OFF);
}

static void example_ble_mesh_provisioning_cb(esp_ble_mesh_prov_cb_event_t event,
                                             esp_ble_mesh_prov_cb_param_t *param)
{
    switch (event) {
    case ESP_BLE_MESH_PROV_REGISTER_COMP_EVT:
        ESP_LOGI(TAG, "ESP_BLE_MESH_PROV_REGISTER_COMP_EVT, err_code %d", param-
>prov_register_comp.err_code);
        break;
    case ESP_BLE_MESH_NODE_PROV_ENABLE_COMP_EVT:
        ESP_LOGI(TAG,  "ESP_BLE_MESH_NODE_PROV_ENABLE_COMP_EVT,  err_code  %d",
param->node_prov_enable_comp.err_code);
        break;
    case ESP_BLE_MESH_NODE_PROV_LINK_OPEN_EVT:
        ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_PROV_LINK_OPEN_EVT, bearer %s",
            param->node_prov_link_open.bearer == ESP_BLE_MESH_PROV_ADV ? "PB-ADV"
: "PB-GATT");
        break;
    case ESP_BLE_MESH_NODE_PROV_LINK_CLOSE_EVT:
        ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_PROV_LINK_CLOSE_EVT, bearer %s",
            param->node_prov_link_close.bearer == ESP_BLE_MESH_PROV_ADV ? "PB-ADV"
: "PB-GATT");
        break;
    case ESP_BLE_MESH_NODE_PROV_COMPLETE_EVT:
        ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_PROV_COMPLETE_EVT");
        prov_complete(param->node_prov_complete.net_idx,                    param-
>node_prov_complete.addr,
            param->node_prov_complete.flags, param->node_prov_complete.iv_index);
        break;
    case ESP_BLE_MESH_NODE_PROV_RESET_EVT:
        ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_PROV_RESET_EVT");
        break;
    case ESP_BLE_MESH_NODE_SET_UNPROV_DEV_NAME_COMP_EVT:
        ESP_LOGI(TAG,  "ESP_BLE_MESH_NODE_SET_UNPROV_DEV_NAME_COMP_EVT,  err_code
%d", param->node_set_unprov_dev_name_comp.err_code);
        break;
    default:
        break;
    }
}

static  void  example_ble_mesh_config_server_cb(esp_ble_mesh_cfg_server_cb_event_t
event,
                                              esp_ble_mesh_cfg_server_cb_param_t
*param)
{
    if (event == ESP_BLE_MESH_CFG_SERVER_STATE_CHANGE_EVT) {
        switch (param->ctx.recv_op) {
        case ESP_BLE_MESH_MODEL_OP_APP_KEY_ADD:
            ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_APP_KEY_ADD");
            ESP_LOGI(TAG, "net_idx 0x%04x, app_idx 0x%04x",
                param->value.state_change.appkey_add.net_idx,
                param->value.state_change.appkey_add.app_idx);
            ESP_LOG_BUFFER_HEX("AppKey",                                    param-
>value.state_change.appkey_add.app_key, 16);
            break;
        case ESP_BLE_MESH_MODEL_OP_MODEL_APP_BIND:
            ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_MODEL_APP_BIND");
            ESP_LOGI(TAG,  "elem_addr  0x%04x,  app_idx  0x%04x,  cid  0x%04x,  mod_id
0x%04x",
                param->value.state_change.mod_app_bind.element_addr,
                param->value.state_change.mod_app_bind.app_idx,
                param->value.state_change.mod_app_bind.company_id,
                param->value.state_change.mod_app_bind.model_id);
            break;
        default:
            break;
        }
    }
}

static void example_ble_mesh_custom_model_cb(esp_ble_mesh_model_cb_event_t event,
                                             esp_ble_mesh_model_cb_param_t
*param)
{
    switch (event) {
    case ESP_BLE_MESH_MODEL_OPERATION_EVT:
        if (param->model_operation.opcode == ESP_BLE_MESH_VND_MODEL_OP_SEND) {
            uint16_t tid = *(uint16_t *)param->model_operation.msg;
            ESP_LOGI(TAG,     "Recv     0x%06x,     tid     0x%04x",     param-
>model_operation.opcode, tid);
            esp_err_t err = esp_ble_mesh_server_model_send_msg(&vnd_models[0],
                    param->model_operation.ctx, ESP_BLE_MESH_VND_MODEL_OP_STATUS,
                    sizeof(tid), (uint8_t *)&tid);
            if (err) {
                ESP_LOGE(TAG,     "Faild     to     send     message     0x%06x",
ESP_BLE_MESH_VND_MODEL_OP_STATUS);
            }
        }
        break;
    case ESP_BLE_MESH_MODEL_SEND_COMP_EVT:
        if (param->model_send_comp.err_code) {
            ESP_LOGE(TAG,    "Failed    to    send    message    0x%06x",    param-
>model_send_comp.opcode);
            break;
        }
        ESP_LOGI(TAG, "Send 0x%06x", param->model_send_comp.opcode);
        break;
    default:
        break;
    }
}

static esp_err_t ble_mesh_init(void)
{
    esp_err_t err;

    esp_ble_mesh_register_prov_callback(example_ble_mesh_provisioning_cb);

esp_ble_mesh_register_config_server_callback(example_ble_mesh_config_server_cb);

esp_ble_mesh_register_custom_model_callback(example_ble_mesh_custom_model_cb);

    err = esp_ble_mesh_init(&provision, &composition);
    if (err != ESP_OK) {
        ESP_LOGE(TAG, "Failed to initialize mesh stack");
        return err;
    }

    err       =       esp_ble_mesh_node_prov_enable(ESP_BLE_MESH_PROV_ADV       |
ESP_BLE_MESH_PROV_GATT);
    if (err != ESP_OK) {
        ESP_LOGE(TAG, "Failed to enable mesh node");
        return err;
    }
    board_led_operation(LED_G, LED_ON);

    ESP_LOGI(TAG, "BLE Mesh Node initialized");

    return ESP_OK;
}

void app_main(void)
{
    esp_err_t err;

    ESP_LOGI(TAG, "Initializing...");

    err = nvs_flash_init();
    if (err == ESP_ERR_NVS_NO_FREE_PAGES) {
        ESP_ERROR_CHECK(nvs_flash_erase());
        err = nvs_flash_init();
    }
    ESP_ERROR_CHECK(err);

    board_init();

    err = bluetooth_init();
    if (err) {
        ESP_LOGE(TAG, "esp32_bluetooth_init failed (err %d)", err);
        return;
    }

    ble_mesh_get_dev_uuid(dev_uuid);

    /* Initialize the Bluetooth Mesh Subsystem */
    err = ble_mesh_init();
    if (err) {
        ESP_LOGE(TAG, "Bluetooth mesh init failed (err %d)", err);
    }
}
```

Source code: https://github.com/espressif/esp-idf/tree/master/examples/bluetooth/esp_ble_mesh/ble_mesh_vendor_model/vendor_server/main