

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIA E SAÚDE**

Luiz Antônio Kuhnen Ronsani

**UM PROTÓTIPO DE SISTEMA INTELIGENTE PARA
MONITORAMENTO DE VAGAS DE
ESTACIONAMENTOS - UMA APLICAÇÃO BASEADA
EM INTERNET DAS COISAS**

Araranguá

2018

Luiz Antônio Kuhnen Ronsani

**UM PROTÓTIPO DE SISTEMA INTELIGENTE PARA
MONITORAMENTO DE VAGAS DE
ESTACIONAMENTOS - UMA APLICAÇÃO BASEADA
EM INTERNET DAS COISAS**

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina, como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Fábio Rodrigues de La Rocha

Araranguá

2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Kuhnen Ronsani, Luiz Antonio
UM PROTÓTIPO DE SISTEMA INTELIGENTE PARA
MONITORAMENTO DE VAGAS DE ESTACIONAMENTOS - UMA
APLICAÇÃO BASEADA EM INTERNET DAS COISAS / Luiz
Antonio Kuhnen Ronsani ; orientador, Fábio
Rodrigues de La Rocha, 2018.
88 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Campus
Araranguá, Graduação em Engenharia de Computação,
Araranguá, 2018.

Inclui referências.

1. Engenharia de Computação. 2. estacionamento
inteligente. 3. cidades inteligentes. 4. internet
das coisas. 5. sistemas embarcados. I. Rodrigues de
La Rocha, Fábio. II. Universidade Federal de Santa
Catarina. Graduação em Engenharia de Computação. III.
Título.

Luiz Antônio Kuhnen Ronsani

**UM PROTÓTIPO DE SISTEMA INTELIGENTE PARA
MONITORAMENTO DE VAGAS DE
ESTACIONAMENTOS - UMA APLICAÇÃO BASEADA
EM INTERNET DAS COISAS**


Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de “Bacharel em Engenharia de Computação”, e aprovado em sua forma final pela Universidade Federal de Santa Catarina.

Araranguá, 28 de Novembro 2018.

Prof^a. Dr^a. Eliane Pozzebon
Coordenador do Curso

Prof. Dr. Fábio Rodrigues de La Rocha
Orientador

Banca Examinadora:



Prof. Dr. Fabrício de Oliveira Ourique

Prof. Dr. Roderval Marcelino

Aos meus pais, que nunca mediram esforços para que eu chegasse até aqui.

AGRADECIMENTOS

À Deus, por iluminar meu caminho nessa caminhada. Aos meus pais, Maria Kuhnen Ronsani e Luiz Carlos Ronsani, por todo apoio, suporte financeiro, e por nunca terem desistido desse sonho apesar das dificuldades. À Ana Lígia Martins, pelo apoio e incentivo para a realização deste trabalho, nunca me deixando esquecer-lo. Aos amigos que fiz durante esta jornada, por todo conhecimento e experiências trocadas. Ao professor Fábio Rodrigues de La Rocha, por ter aceitado me orientar nesta etapa importante da graduação. Mas principalmente por toda ajuda, ensinamentos e conhecimentos compartilhados através de várias ligações por Skype. À UFSC, pela oportunidade de ter estudado em uma Universidade de excelência. À todo corpo docente do curso de Engenharia de Computação, por todo conhecimento passado ao longo destes anos. E a todos aqueles que de alguma forma contribuíram para a realização deste trabalho.

*Procure ser uma pessoa de valor, em vez
de procurar ser uma pessoa de sucesso. O
sucesso é consequência*

Albert Einstein

RESUMO

A quantidade de veículos nas ruas tem aumentado substancialmente nas últimas décadas, em consequência disso também têm aumentado o tempo despendido no momento de procurar por uma vaga em um estacionamento. Diante disso, algumas empresas têm desenvolvido algumas soluções para facilitar esta tarefa, entretanto estas soluções ainda apresentam um alto custo e uma grande infraestrutura física. Além disso, estas alternativas ainda não resolvem este problema de uma maneira eficiente, fazendo com que o condutor esteja atento à painéis e ainda gaste tempo percorrendo o estacionamento. Desta forma, neste trabalho foi desenvolvido um sistema de gerenciamento de vagas, baseado nos conceitos de Internet das Coisas (IoT), utilizando redes sem fio. Para isto, um aplicativo móvel foi desenvolvido, o qual exibe um mapa do estacionamento mostrando as vagas ocupadas e livres, permitindo que o cliente escolha a vaga mais conveniente. Além disso, o aplicativo mostra informações relevantes sobre o estado atual do estacionamento, permitindo que o cliente tome decisões previamente e desta forma evitar a perda de tempo.

Palavras-chave: estacionamento inteligente, cidades inteligentes, internet das coisas, redes sem fio, sistemas embarcados, aplicação móvel

ABSTRACT

The number of vehicles on the streets has increased substantially in recent decades, as a result of this, the time spent searching for a parking space has also increased. In consequence, some companies have developed solutions to facilitate this task, however these solutions still present a high cost and requires a big physical infrastructure. In addition, these alternatives still do not solve this problem in an efficient way, which the driver needs to pay attention at displays and still spend time traveling the entire parking lot. In this way, this work aims to develop a system of parking spaces management, based on the concepts of Internet of Things (IoT), using wireless networks. For this, a mobile application was developed which will display a parking map showing the spaces occupied and free, allowing the client to choose the most convenient place. In addition, the application shows relevant information about the current state of parking, allowing the customer to make decisions in advance and thus avoid wasting time

Keywords: smart parking, smart cities, internet of things, IoT, wireless networks, embedded systems, mobile app

LISTA DE FIGURAS

Figura 1	Fatia de mercado projetada das principais aplicações de IoT para 2025.....	29
Figura 2	Arquiteturas IoT: (a) Três camadas, (b) Baseada em Middleware, (c) Orientada a Serviços, (d) Cinco camadas.....	30
Figura 3	Os elementos de IoT.....	31
Figura 4	Exemplo de Microcontrolador Microchip.....	34
Figura 5	Núcleo do ESP-8266 (E) e ESP-32 (D).....	37
Figura 6	ESP8266 NodeMCU.....	38
Figura 7	ESP32 NodeMCU.....	38
Figura 8	Arquitetura de uma rede <i>mesh</i> de infra estrutura.....	40
Figura 9	Arquitetura de uma rede <i>mesh</i> de clientes.....	41
Figura 10	Arquitetura de uma rede <i>mesh</i> híbrida.....	41
Figura 11	Topologia estrela e topologia <i>mesh</i>	42
Figura 12	Estrutura do sistema desenvolvido.....	44
Figura 13	ESP-8266 (E) e ESP-32 (D).....	45
Figura 14	Sensor ultrassônico HC-SR04P.....	46
Figura 15	Ilustração do funcionamento do sensor ultrassônico....	46
Figura 16	Protótipo do sensor.....	47
Figura 17	Esquemático de ligação do sensor.....	47
Figura 18	Interface <i>web</i> versão <i>desktop</i>	48
Figura 19	Interface <i>web</i> versão <i>mobile</i>	49

LISTA DE TABELAS

Tabela 1	Algumas interrupções disponíveis em microcontroladores PIC30F	36
Tabela 2	Tabela dos nós cadastrados no banco de dados	50
Tabela 3	Tabela do banco de dados utilizado nos testes	52

LISTA DE ABREVIATURAS E SIGLAS

IoT	Internet of Things
M2M	Machine to Machine
RFID	Radio-Frequency Identification
3G	Third Generation
GSM	Global System for Mobile
UMTS	Universal Mobile Telecommunications System
IEEE	Institute of Electrical and Electronics Engineers
LTE	Long-Term Evolution
NFC	Near-Field communication
FPGA	Field Programmable Gate Array
ASIC	Field Programmable Gate Array)
LED	Light Emitting Diode
SoC	System on a Chip
USB	Universal Serial Bus
WMN	Wireless Mesh Network
P2P	Peer to Peer
PC	Personal Computer
JSON	JavaScript Object Notation
XML	Extensible Markup Language
SVG	Scalable Vector Graphics

SUMÁRIO

1 INTRODUÇÃO	23
1.1 JUSTIFICATIVA	24
1.2 OBJETIVOS	25
1.2.1 Objetivo Geral	25
1.2.2 Objetivos Específicos	25
1.3 METODOLOGIA	26
1.4 ORGANIZAÇÃO DO TRABALHO	26
2 INTERNET DAS COISAS	28
2.1 ARQUITETURA	28
2.2 ELEMENTOS DA <i>IOT</i>	30
3 SISTEMAS EMBARCADOS	33
3.1 MICROCONTROLADORES	33
3.1.1 Componentes Integrados ao Microcontrolador	34
3.1.1.1 <i>Timers</i>	34
3.1.1.2 <i>Pinos de Entrada e Saída</i>	35
3.1.1.3 Interrupção	35
3.1.1.4 Comunicação	35
3.2 MICROCONTROLADOR ESP	36
4 REDES MESH SEM FIOS	39
4.1 A ARQUITETURA DA REDE	39
4.1.1 Redes de infraestrutura	40
4.1.2 Redes de clientes	40
4.1.3 Redes híbridas	41
4.2 VANTAGENS DAS REDES <i>MESH</i>	42
5 DESENVOLVIMENTO	43
5.1 DESCRIÇÃO	43
5.1.1 Requisitos do Sistema	44
5.1.1.1 Requisitos Funcionais	44
5.1.1.2 Requisitos não funcionais	45
5.2 HARDWARE	45
5.2.1 Protótipo	46
5.3 SOFTWARE	47
5.3.1 Interface WEB do Cliente	48
5.3.2 Servidor Web	49
5.3.3 Banco de dados	50
5.3.4 Firmware	51
5.3.4.1 Firmware nó ordinário	51

5.3.4.2 Firmware nó mestre	51
6 TESTES E RESULTADOS	52
6.1 TESTES E VALIDAÇÃO DA REDE <i>MESH</i>	52
6.2 TESTES DA APLICAÇÃO	53
7 CONSIDERAÇÕES FINAIS.....	54
7.1 CONCLUSÃO.....	54
7.2 TRABALHOS FUTUROS	54
REFERÊNCIAS	55

1 INTRODUÇÃO

Com o grande crescimento da indústria dos semicondutores e consequentemente na constante diminuição na escala de tamanho dos transistores, o desenvolvimento de dispositivos cada vez menores e com a capacidade de estarem conectados à uma rede de internet têm transformado o modo em que vivemos (LIU; TONG, 2017). Através disso, em 1999 surgiu o termo *Internet das Coisas (IoT)*, que de acordo com Rose (2015), é o cenário onde a conectividade à rede e a capacidade computacional se estende aos objetos, sensores e itens do cotidiano antes não considerados computadores, permitindo a estes dispositivos gerar, trocar e consumir dados sem nenhuma intervenção humana. Segundo Al-Fuqaha (2015), é esperado que 50 bilhões de dispositivos inteligentes sejam implementados no mercado global até 2020. Além disso, é esperado que até 2022 o tráfego de informações machine-to-machine (M2M na sigla em inglês) seja equivalente à 45% do tráfego total da internet.

Inúmeras são as aplicações para a IoT que podem ser utilizadas para solucionar problemas do dia-a-dia. Um problema bem presente nos dias atuais é a mobilidade urbana. Supermercados, shopping centers e grandes centros comerciais disponibilizam áreas de estacionamento destinadas aos seus clientes. Entretanto, a quantidade de veículos nas ruas têm aumentado substancialmente nas últimas décadas, em consequência disso, temos um dos nossos bens mais preciosos, o tempo, sendo desperdiçado a cada vez que procuramos por uma vaga em um estacionamento. Diante deste cenário, muitas empresas têm buscado alternativas inteligentes para reduzir este tempo gasto, no objetivo de atender seus clientes de forma mais eficiente.

Este é um problema bem conhecido para o qual algumas empresas já utilizam tecnologias para facilitar o condutor a encontrar uma vaga livre, porém, essas tecnologias possuem alguns inconvenientes: elas têm alto custo e precisam de infraestrutura planejada no local. O que vemos hoje é que apenas grandes estacionamentos são capazes de lidar com esses inconvenientes, inviabilizando o acesso aos de pequeno porte. Além disso, quando estas tecnologias foram desenvolvidas não existia o acesso praticamente universal aos dispositivos móveis como temos atualmente, permitindo que o mesmo problema seja revisto abrindo possibilidades para soluções mais vantajosas.

A utilização de redes sem fio é uma alternativa eficiente e que junto com o crescimento da Internet das Coisas têm se tornado bastante acessíveis e baratas. A topologia de rede mais empregada em aplica-

ções de *IoT* são as redes *mesh* sem fio, que é uma rede de comunicação em que cada nó da rede é um transmissor de rádio frequência organizados em uma topologia tipo malha, onde todos os nós da rede estão interconectados. Esta topologia possui algumas vantagens, dentre elas podemos citar: sua auto-configuração, fácil acesso à rede, e baixo custo (LIU; TONG, 2017).

Neste trabalho, apresenta-se a proposta de um sistema para o controle e automação de um estacionamento fechado, realizando a detecção de vagas livres e ocupadas e apresentando essas informações ao condutor através de um mapa interativo. Nesta visão, cada vaga é um nó dentro do sistema, que através de comunicação sem fios se comunicam entre si a fim de manter o sistema atualizado. Além disso, através de um aplicativo móvel o usuário poderá conhecer a situação global do estacionamento que pretende utilizar e assim rapidamente encontrar uma vaga livre ou ainda deixar o estacionamento quando lotado, evitando transtornos e perda de tempo.

1.1 JUSTIFICATIVA

Hoje em dia cada vez mais o automóvel, que até então era sinônimo de conforto e praticidade, vem se tornando motivo de estresse pelo motorista frente à dificuldade de encontrar uma vaga em um estacionamento. Mesmo os grandes centros comerciais disponibilizando um espaço próprio para o estacionamento de seus clientes, ainda há uma perda de tempo considerável despendida na busca por uma vaga livre.

O problema que pretendemos atacar já é bem conhecido na literatura e para este já existem soluções tecnológicas. A motivação para retornar ao problema é que uma nova tecnologia foi desenvolvida e esta permite repensar o problema através de uma nova abordagem com ganhos em custos e benefícios. Além do acesso aos *smartphones* por praticamente toda a população, o que facilita uma solução através de um aplicativo, o desenvolvimento de novos microcontroladores, menores, mais baratos e portadores de comunicação sem fios, são motivadores para buscar uma nova solução para este problema.

Atualmente este problema é resolvido de duas maneiras: a forma tradicional e o sistema automatizado. Na forma tradicional, o cliente deve circular pelo estacionamento até encontrar uma vaga livre. Em alguns casos, a empresa disponibiliza um funcionário para auxiliar o cliente indicando uma vaga livre ou indicando onde provavelmente ele irá encontrá-la. Neste caso, o cliente pode acabar não encontrando uma

vaga e desistir, ocasionando prejuízos ao estabelecimento. No sistema automatizado, cada vaga em um estacionamento possui um sensor que detecta se existe um veículo ali estacionado ou não, indicando através de um sinal verde para vaga livre ou vermelho para vaga ocupada. Além disso, este modelo possui alguns painéis espalhados pelo estacionamento indicando quantas vagas disponíveis existem naquele corredor. Neste tipo solução, apesar de ser mais eficiente que a tradicional, ainda apresenta alguns problemas. Um deles é o alto custo, pois a comunicação é realizada através de fios, o que exige uma infraestrutura do prédio para ser instalado (cabearamento e painéis eletrônicos). Outro problema é que o cliente ainda terá que percorrer o estacionamento observando os painéis ou sinal luminoso. Além disso, em muitos casos o estacionamento é composto por mais de um andar, desta forma, dificulta ainda mais o condutor.

1.2 OBJETIVOS

Esta seção apresenta os objetivos desejados descritos abaixo para o desenvolvimento deste trabalho.

1.2.1 Objetivo Geral

Utilizar a *IoT* para propor uma nova solução para o problema dos estacionamentos. Criar uma solução tecnológica integrada *hardware/software* para demonstrar a viabilidade técnica e os benefícios da solução e com plena capacidade de transformar-se num produto a ser lançado no mercado.

1.2.2 Objetivos Específicos

1. Propor um sistema embarcado para realizar o sensoriamento das vagas de estacionamento utilizando eletrônica embarcada de baixo custo;
2. Desenvolver a infraestrutura de alto nível (*back-end*) a ser executado num computador servidor para integrar as informações provenientes de todos os sensores;
3. Usar tecnologia de transmissão de dados entre clientes e o servidor para que os clientes do estacionamento possam acompanhar

através de uma interface móvel (*front-end*) o *status* das vagas do estacionamento;

4. Desenvolver um protótipo funcional capaz de ser apresentado como prova de conceito.

1.3 METODOLOGIA

Este trabalho é uma pesquisa tecnológica aplicada que tem como um objetivo final desenvolver um protótipo funcional. O trabalho em questão utiliza diferentes tecnologias tais como microcontroladores, transmissão *WiFi*, sensores de ultrassom bem como desenvolvimento de sistemas *Web* (aplicações *back-end* e *front-end*).

O desenvolvimento do *hardware* é baseado no microcontrolador *ESP8266* e o *software* embarcado foi desenvolvido utilizando o Ambiente *Arduino* em linguagem de programação *C++*.

O desenvolvimento de *software Web* (*back-end*) foi realizado em *NodeJS* em função da capacidade de escalabilidade, grande quantidade de bibliotecas disponíveis que implementam comunicação por *sockets*, *http*, etc. Outra grande vantagem é que o *NodeJS* é uma extensão de *JavaScript*, o que nos permite trabalhar utilizando uma mesma linguagem do lado do servidor e também do lado do cliente (*front-end*). Desta forma, obtendo ganhos em produtividade ao reaproveitar código entre as partes.

A troca de mensagens entre os nós sensores é realizada através de uma topologia *MESH* em virtude da capacidade desta em propiciar que as mensagens enviadas por um nó sejam capturadas e propagadas para outros nós até que eventualmente atinjam o destino. Durante o desenvolvimento são produzidos arquivos de *log* para verificar a correta troca de mensagens entre os participantes da rede, bem como as mensagens recebidas no servidor.

1.4 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está organizado em 7 (sete) capítulos, contando com a introdução.

O **Capítulo 2** apresenta importantes conceitos sobre Internet das Coisas. O surgimento do termo e suas aplicações.

O **Capítulo 3** apresenta o conceito de um Sistema Embarcado. O que é, suas características e aplicações.

O **Capítulo 4** apresenta um detalhamento sobre as redes mesh sem fios, apresentando suas características e explicando seu funcionamento.

O **Capítulo 5** apresenta o sistema proposto. Dividido em duas etapas, *hardware* e *software*, este capítulo mostra com detalhes o protótipo desenvolvido para testes e a aplicação móvel criada para este fim.

O **Capítulo 6** apresenta os testes realizados e os resultados obtidos.

O **Capítulo 7** apresenta as considerações finais sobre este trabalho e algumas propostas para trabalhos futuros.

2 INTERNET DAS COISAS

O termo Internet das Coisas foi primeiramente utilizado em 1999 pelo britânico pioneiro em tecnologia Kevin Ashton para descrever um sistema no qual objetos no mundo físico poderiam estar conectados à internet através de sensores (ROSE; ELDRIDGE; CHAPIN, 2015).

Até então pensava-se na Internet como uma rede que conectava apenas pessoas. Hoje o cenário é tal que existem mais dispositivos eletrônicos do dia a dia acessando a rede do que pessoas. Um bom exemplo disso é a constatação que em 2011 a população da terra atingiu 7 bilhões de pessoas e o número de dispositivos na rede atingiu 13 bilhões. A expectativa para 2015 era de três vezes mais dispositivos conectados na Internet do que a população mundial e calcula-se que em 2020 existirão 50 bilhões de dispositivos na rede e a expectativa é que a população mundial esteja em apenas 7,6 bilhões (OLSON, 2018) (MIORANDI et al., 2012) (ATZORI; IERA; MORABITO, 2010).

O mercado para *IoT* oferece grandes oportunidades, de acordo com Al-Fuqaha et al. (2015), o impacto econômico anual causado pela *IoT* é estimado entre 2,7 e 6,2 trilhões de dólares para o ano de 2015. Na figura 1 podemos ver os principais mercados impactados pela Internet das Coisas.

2.1 ARQUITETURA

A *IoT* deve ser capaz de interconectar bilhões ou trilhões de dispositivos heterogêneos através da internet, e para que isso aconteça, existe uma necessidade enorme por uma arquitetura de camadas flexíveis.

Nos modelos de arquitetura existentes, o modelo básico é composto por três camadas: camada de aplicação, camada de rede e camada de percepção. Entretanto, na literatura recente, foram propostos novos modelos de arquiteturas que apresentam maior abstração. Na figura 2 podemos ver algumas arquiteturas propostas, entre elas a arquitetura de cinco camadas, que de acordo com Al-Fuqaha et al. (2015), tem sido muito utilizada atualmente.

A primeira camada é a Camada de Objetos, que representa os sensores nas nas aplicações de *IoT*, responsáveis por coletar e processar as informações. Esta camada inclui sensores e atuadores, executando diferentes funcionalidades, como leitura de temperatura, peso, veloci-

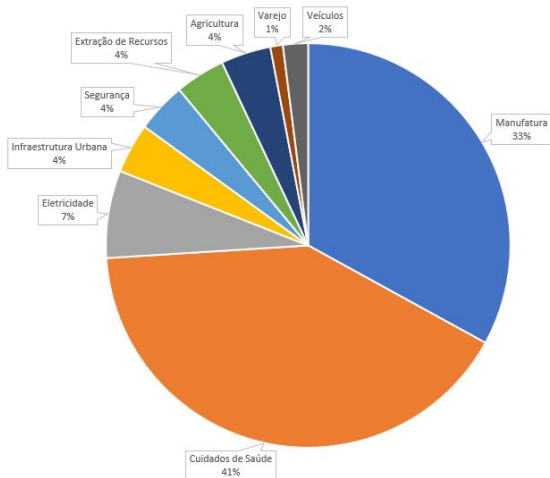


Figura 1 – Fatia de mercado projetada das principais aplicações de IoT para 2025

Fonte: Adaptado de Al-Fuqaha et al. (2015)

dade, etc. A camada de objetos digitaliza e transfere os dados para a próxima camada, a camada de abstração de objetos.

A segunda camada é a de Abstração de Objetos, que é responsável por transmitir os dados recebidos da camada de objetos para a camada de gerenciamento de serviços através de canais seguros. A transferência dos dados pode ocorrer através de diversas tecnologias, como RFID, 3G, GSM, UMTS, WiFi, Bluetooth, infravermelho, Zig-Bee, etc (AL-FUQAHA et al., 2015).

A terceira camada, chamada de Gerenciamento de Serviços, é responsável por parear os serviços e quem os requisitou baseado em nomes e endereços. Esta camada permite os programadores a trabalhar com objetos diferentes sem se preocupar com plataformas de *hardware* específicas. Além disso, processa dados recebidos, toma decisões e entrega serviços requisitados através de protocolos de rede.

A Camada de Aplicação é a quarta camada desta arquitetura e é responsável por prover os serviços requisitados pelos clientes. A importância desta camada é que ela tem a capacidade de prover serviços de inteligentes de alta qualidade para atender as necessidades dos clientes. A camada de aplicação engloba diversas vertentes do mercado de *IoT*, como *smart homes*, *smart buildings*, transportes, entre outros (AL-FUQAHA et al., 2015).

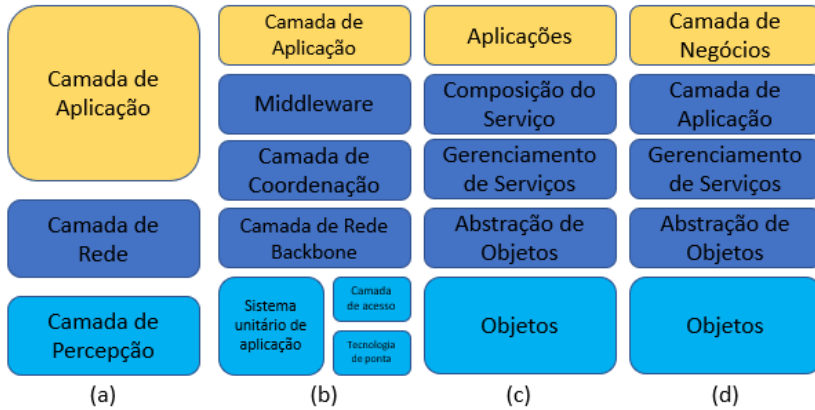


Figura 2 – Arquiteturas IoT: (a) Três camadas, (b) Baseada em Middleware, (c) Orientada a Serviços, (d) Cinco camadas

Fonte: Adaptado de Al-Fuqaha et al. (2015)

A última camada desta arquitetura é a Camada de Negócio, que gerencia globalmente os serviços e atividades das aplicações de *IoT*. Ela é responsável pela criação de modelos de negócio, gráficos, fluxogramas, etc, baseados nos dados recebidos da camada de aplicação. Esta camada permite a tomada de decisão baseada em análises de *Big Data*. Além disso, o monitoramento e gerenciamento das quatro camadas inferiores é alcançada na camada de negócio.

2.2 ELEMENTOS DA IOT

Alguns elementos são essenciais para entregar o funcionamento das aplicações de *IoT*. Segundo Al-Fuqaha et al. (2015), os seis principais elementos são: identificação, sensoriamento, comunicação, processamento, serviços e semântica. Na figura 3 podemos ver uma ilustração destes elementos.

O primeiro elemento da lista é a identificação, e é essencial para a *IoT* para nomear e combinar os serviços com sua demanda. De acordo com Atzori, Iera e Morabito (2010), os componentes-chave das aplicações de Internet das Coisas serão *RFID* (Radio-Frequency Identification, ou Identificação por Rádio Frequência) por possuírem um identificador único (ATZORI; IERA; MORABITO, 2010).

O elemento de sensoriamento é onde os dados são coletados dos



Figura 3 – Os elementos de IoT

Fonte: Adaptado de Al-Fuqaha et al. (2015)

objetos dentro da rede e enviados para o "armazém de dados". A partir deste momento, os dados podem ser analisados e servir para tomada de decisão pertinente à aplicação. Os sensores nas aplicações de *IoT* podem ser de vários tipos, como sensores inteligentes e atuadores, ou os chamados *wearable devices* (dispositivos vestíveis), como os *smartwatches*.

O terceiro elemento, o elemento de comunicação, é onde os dispositivos heterogêneos são interconectados para entregar serviços inteligentes específicos. Os nodos de *IoT* devem, geralmente, operar utilizando baixo consumo na presença de ruídos no canal de comunicação. Exemplos de protocolos de comunicação usados nas aplicações de *IoT* são *WiFi*, *Bluetooth*, *IEEE 802.15.4*, *Z-wave*, *LTE*, *NFC*, *RFID* e *M2M* (AL-FUQAHA et al., 2015).

Processamento é o quarto elemento da lista, onde unidades de processamento como microprocessadores, microcontroladores e FPGAs (Field Programmable Gate Array), são o cérebro e a habilidade computacional da *IoT*. Existem várias plataformas de *hardware* que foram desenvolvidas para executarem aplicações de *IoT*, como *Arduino*, *Intel Galileo*, *Raspberry PI* e *ESP*. Além disso, existem as plataformas em nuvem para *IoT*, que oferecem instalações para objetos inteligentes enviarem seus dados para a nuvem, oferecendo o conceito de *Big Data*, além de processamento em tempo real (AL-FUQAHA et al., 2015).

O elemento de serviços é onde os serviços de *IoT* são providos, como serviços relacionados à identidade, serviços de agregação de informação, serviços de colaboração e serviços ubíquos (AL-FUQAHA et al., 2015). Aqui são providos serviços para todas aplicações como as descritas na figura 1.

O último elemento da lista é a semântica, que, de acordo com Atzori, Iera e Morabito (2010), é onde está o maior desafio, pois o endereçamento único dos objetos e a representação e armazenamento das informações trocadas são tarefas difíceis. Neste contexto, semântica é a habilidade de extrair conhecimento de forma inteligente através de

dispositivos diferentes para prover os serviços requeridos pela aplicação (AL-FUQAHA et al., 2015).

3 SISTEMAS EMBARCADOS

De acordo com KAMAL (2014), um sistema embarcado é um sistema que possui *software* embarcado e *hardware* computacional, o qual o faz um sistema dedicado para uma aplicação ou uma parte específica de uma aplicação, ou produto, ou uma parte de um grande sistema. Para KUMAR et al. (2011), sistema embarcado são sistemas específicos de aplicações que contêm *hardware* e *software* sob medida para uma tarefa específica e são geralmente parte de um sistema maior.

Processadores de propósito geral, processadores de propósito especial e ASIC (Application-Specific Integrated Circuit, ou Circuito Integrado de Aplicação Específica), estão entre os muitos componentes utilizados para implementar estes sistemas. Sistemas embarcados geralmente estão dentro da categoria de sistemas reativos, sistemas os quais continuamente interagem com seres humanos e o meio ambiente, com sistemas em tempo real que devem encontrar alguma restrição de tempo (KUMAR et al., 2011).

Sistemas embarcados estão sempre presentes em nosso dia-a-dia, podemos citar exemplos como: a unidade de controle de um motor de automóvel, o piloto automático de uma aeronave, o sistema de controle de um micro-ondas e até o controle de uma cafeteira.

Neste capítulo, abordaremos os principais componentes que fazem parte de um sistema embarcado e suas características.

3.1 MICROCONTROLADORES

Kamal (2014) define microcontrolador como um chip integrado que possui processador, memória e diversas outras unidades de *hardware* em si. Apesar de possuir vários periféricos como memórias, barramentos, *timers*, portas de comunicação, conversores de sinal analógico para digital, etc, os microcontroladores são componentes pequenos, ideais para aplicações que necessitam de menores dimensões e custos (ANDRADE, 2010).

Primeiramente foram criados os microprocessadores, os quais foram responsáveis por substituir milhões de transistores nos computadores, que eram enormes e podiam ocupar o tamanho de uma casa. Os microprocessadores diminuíram drasticamente o tamanho dos computadores, e atualmente estão presentes em muitos aparelhos eletrônicos (ANDRADE, 2010). Na figura 4 podemos ver um exemplo de micro-

controlador da fabricante *Microchip*, uma das maiores fabricantes de microcontroladores do mundo.

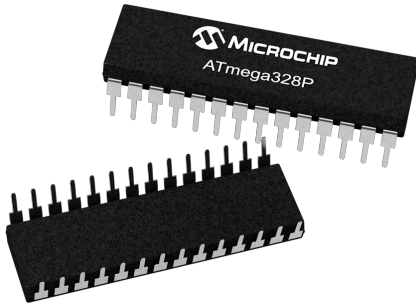


Figura 4 – Exemplo de Microcontrolador Microchip

Fonte: Microchip (2018)

3.1.1 Componentes Integrados ao Microcontrolador

Os microcontroladores possuem normalmente diversos circuitos e recursos auxiliares integrados em sua construção. Estes circuitos auxiliares geralmente logicamente separados do processador, sendo construídos no mesmo encapsulamento do microcontrolador. Isso traz a vantagem de disponibilizar o programador de ter recursos no mesmo componente sem a necessidade de adicionar periféricos externos ao sistema (SIMON, 1999).

3.1.1.1 *Timers*

Os temporizadores (*Timers*) são componentes que têm a função de fornecer noção de tempo para as aplicações. É comum os microcontroladores possuírem mais de um *timer*. Basicamente sua função é contar os ciclos de *clock* do processador e chamar uma interrupção quando a contagem expirar. Uma das características dos *timers* é o *pre-scaler*, que divide o sinal de *clock* do processador por algum valor constante, com isso, a cada determinada quantidade de ciclos de *clock* o contador incrementa uma unidade (SIMON, 1999) (ANDRADE, 2010).

Os tipos mais comum de temporizadores são de 16 e 32 *bits*. Isso significa que a contagem máxima realizada é de:

$$2^{16} - 1 = 65.535 \quad (3.1) \quad 2^{32} - 1 = 4.294.967.295 \quad (3.2)$$

3.1.1.2 Pinos de Entrada e Saída

Em microcontroladores destinados à sistemas embarcados é comum existirem diversos pinos de entrada e saída (I/O *Input* e *output*). Estes pinos podem ser configurados como de saída, onde o programador pode escrever um valor alto ou baixo (0 ou 1), escrevendo em um registrador, ou como pinos de entrada, quando é realizada a leitura de algum sensor, como por exemplo um sensor de ultrassom.

Estes pinos podem ser usados para diversos propósitos, como ligar e desligar um *LED* (diodo emissor de luz, na sigla em ingles), resetar o *watchdog timer*, ler dados de uma memória externa, etc. (SIMON, 1999).

3.1.1.3 Interrupção

De acordo com Andrade (2010), interrupção é um evento disparado por algum acontecimento externo ao processador, que faz com que ele pare a execução do *software* e desvie a execução para um bloco de código alternativo. Ao final da execução desta operação, ele retorna exatamente à instrução seguinte ao ponto onde foi interrompido.

As interrupções podem ser feitas de diversas formas, na tabela 1 podemos ver alguns exemplos para a família de microcontroladores *PIC30F*.

3.1.1.4 Comunicação

Existem dois tipos básicos de comunicação entre dispositivos computacionais, a comunicação paralela e a comunicação serial.

Na comunicação paralela, os dados são transmitidos simultaneamente pelo barramento, no caso de um barramento de 8 bits, 8 dados podem ser transmitidos ao mesmo tempo. Este tipo de comunicação é muito presente em computadores pessoais, principalmente nas conexões de impressoras.

Já na comunicação serial, os dados são transmitidos em fila, um após o outro, por uma única via. Este tipo de comunicação de dados é o

Número da interrupção	Origem da Interrupção
0	INT0 - Interrupção Externa 0
1	IC1 - Entrada no Capture 1
2	OC1 - Saída do Compare 1
3	T1 - Timer 1
4	IC2 - Entrada no Capture 2
5	OC2 - Saída do Compare 2
6	T2 - Timer 2
7	T3 - Timer 3
8	SPI 1
16	INT1 - Interrupção Externa 1
23	INT2 - Interrupção Externa 2

Tabela 1 – Algumas interrupções disponíveis em microcontroladores PIC30F

Fonte: Adaptado de Andrade (2010)

mais comum entre os sistemas embarcados, pois ocupa menor espaço no *chip*. Aqui, existem dois canais de transmissão, um para enviar dados (Tx) e outro para o recebimento dos dados (Rx). O controle dos dados trafegados é feito pela inclusão de alguns *bits* para cada dado enviado. Os *bits* de controle mais comuns são o *bit* de início, de parada, e de paridade. O *bit* de paridade é configurado em 0 ou 1 para garantir que o total de *bits* 1 no campo de dados é par ou ímpar, como desejado pelo programador (ANDRADE, 2010).

3.2 MICROCONTROLADOR ESP

No trabalho desenvolvido, foram utilizados microcontroladores modelos *ESP8266* e *ESP-32*. Estes microcontroladores são fabricados pela chinesa *Espressif Systems* e são muito utilizados em aplicações de *IoT*. Na figura 5 podemos ver o núcleo do *ESP8266* e do *ESP32*.

Estes microcontroladores são *SoC* (*System on a Chip*), o que significa que possui todos os componentes de um computador em um único chip. As versões utilizadas foram as chamadas *NodeMCU*, o que podemos chamar de plataformas de desenvolvimento, pois possuem diversos componentes para facilitar seu uso, como porta de configuração *USB* (*Universal Serial Bus*), pinos de comunicação, *WiFi*, e no caso do *ESP32*, até *bluetooth*.

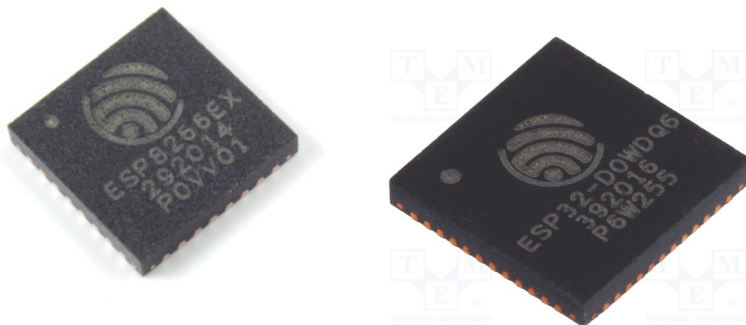


Figura 5 – Núcleo do ESP-8266 (E) e ESP-32 (D)

O microcontrolador ESP8266EX integra um processador *RISC Tensilica L106* de 32 bits, que possui *clock* máximo de 160 MHz e baixíssimo consumo de energia. O sistema operacional em tempo real (RTOS) e a pilha Wi-Fi permitem que cerca de 80% da capacidade de processamento esteja disponível para programação e desenvolvimento de aplicativos do usuário (ESPRESSIF, 2018).

Quanto ao consumo de energia, como são projetados para dispositivos móveis, eletrônicos vestíveis e aplicativos IoT, o ESP8266EX apresenta três modos de operação: modo ativo, modo de suspensão e modo de suspensão profunda. Isso permite que projetos alimentados por baterias não necessitem troca constante das mesmas (ESPRESSIF, 2018).

Na Figura 6 e Figura 7, podemos ver os modelos *ESP8266 NodeMCU* e na temos o modelo *ESP32 NodeMCU*.

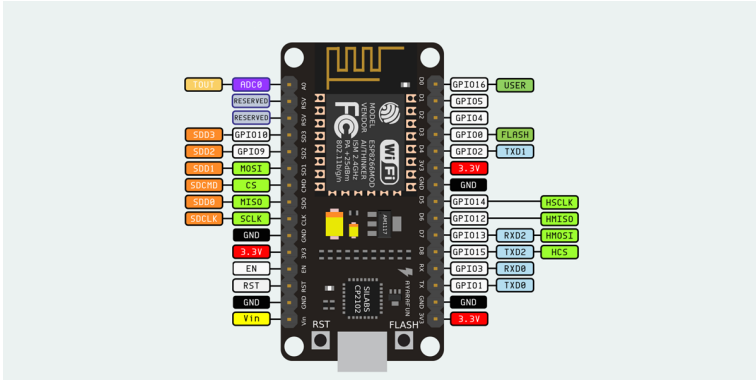


Figura 6 – ESP8266 NodeMCU
 Fonte: Google Imagens

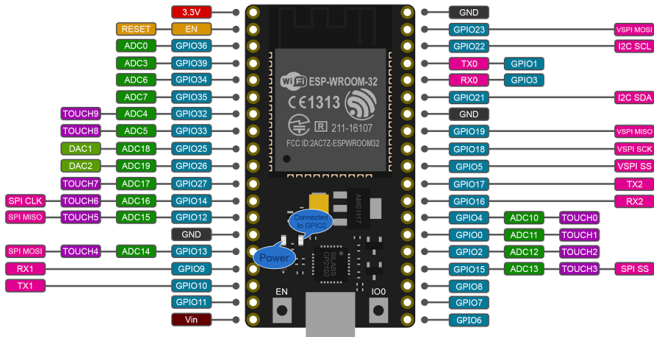


Figura 7 – ESP32 NodeMCU
 Fonte: Google Imagens

4 REDES *MESH* SEM FIOS

Assim como muitos tipos de redes sem fios têm sido desenvolvidas para oferecerem melhores serviços, *Wireless Mesh Networks* (WMN) surgiu recentemente. Em WMNs, os nós são compostos por roteadores e clientes. Cada nó opera como *host* e como roteador, encaminhando pacotes de outros nós que não possuem caminho direto ao destinatário. Uma rede *mesh* é auto-organizada e auto-configurada, com os nós da rede estabelecendo e mantendo a conexão automaticamente entre eles.

Redes *mesh* sem fios podem ser utilizadas nas mais diversas aplicações, mas tem se apresentado como a principal topologia de rede para aplicações de Internet das Coisas, onde o número de nós nestas aplicações é muito grande e dinâmico. Com a capacidade de auto-organização e auto-configuração, as redes *mesh* podem ser implantadas gradualmente, adicionando nós conforme necessário. Quanto mais nós são adicionados à rede, maior a sua confiabilidade e, conseqüentemente, sua conectividade (AKYILDIZ; WANG, 2009).

4.1 A ARQUITETURA DA REDE

Redes *mesh* consistem em dois tipos de nós: roteadores e clientes. Além da capacidade de roteamento para funções de *gateway/repetidor*, como em um roteador convencional sem fio, um roteador *wireless mesh* contém funções de roteamento adicionais para suportar redes *mesh*. Para aprimorar ainda mais a flexibilidade da rede, um roteador de *mesh* geralmente é equipado com múltiplas interfaces sem fio construídas nas mesmas ou diferentes tecnologias de acesso sem fio. Comparado com um roteador sem fio convencional, um roteador *mesh* sem fio pode alcançar a mesma cobertura com um poder de transmissão muito menor através de comunicações multihop. Opcionalmente, o protocolo de controle de acesso médio em um roteador *mesh* é aprimorado com uma melhor escalabilidade em um ambiente *mesh* multihop (AKYILDIZ; WANG, 2009).

Clientes *mesh* também possuem as funções necessárias para a interconexão *mesh*, e em alguns casos podem funcionar como roteadores. Nós clientes possuem apenas uma interface de conexão sem fios, ou seja, se conectam a apenas um roteador.

A arquitetura das redes *mesh* pode ser classificadas em três categorias: redes de infra estrutura, redes clientes, e redes híbridas.

4.1.1 Redes de infraestrutura

Neste tipo de arquitetura, os nós roteadores formam a rede *mesh* para clientes se conectarem a eles. Nesta abordagem, um ou mais nós roteadores se conectam a internet, e através deles os clientes também possuem acesso à rede. É o tipo de arquitetura mais utilizado, sendo aplicado em soluções residenciais e corporativas, como por exemplo: automação residencial, *smart homes*, *smart buildings*, e a maioria das aplicações de Internet das Coisas. Na figura 8, temos uma representação de como é a arquitetura do tipo infraestrutura.

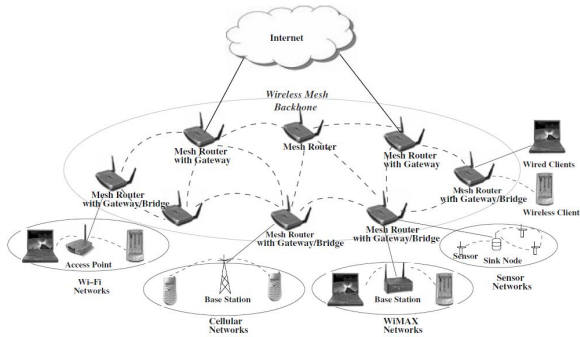


Figura 8 – Arquitetura de uma rede *mesh* de infra estrutura
 Fonte: Adaptado de Akyildiz; Wang (2009)

4.1.2 Redes de clientes

Nas redes *mesh* clientes temos uma rede *peer-to-peer* (P2P) entre os dispositivos. Nesta configuração, os nós clientes constituem a rede e fazem o roteamento e configuração da mesma. Um nó roteador não é necessário neste tipo de rede, pois o acesso às redes externas se fazem através dos próprios dispositivos. Na figura 9 podemos visualizar como esta arquitetura é formada, onde os pacotes de informação são enviados e passam por vários nós até chegarem ao destino final.

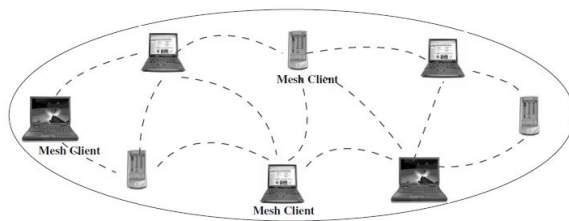


Figura 9 – Arquitetura de uma rede *mesh* de clientes
 Fonte: Adaptado de Akyildiz; Wang (2009)

4.1.3 Redes híbridas

Nesta configuração temos uma mistura de rede de infraestrutura e rede de clientes. Neste caso, os clientes podem acessar a rede através dos nós roteadores ou se comunicarem diretamente com outros nós clientes. Esta arquitetura também é muito utilizada, pois oferece conexão à internet e a comunicação entre os clientes dentro da própria rede. Na figura 10 temos um modelo de rede híbrida.

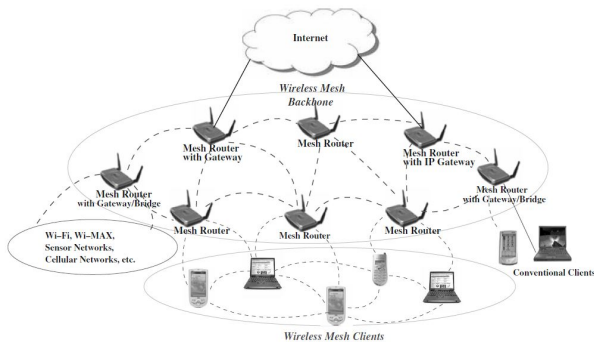


Figura 10 – Arquitetura de uma rede *mesh* híbrida
 Fonte: Adaptado de Akyildiz; Wang (2009)

4.2 VANTAGENS DAS REDES *MESH*

As redes *mesh* sem fios têm sido amplamente utilizadas em aplicações de *IoT* por apresentar grandes vantagens em relação às redes tipo *estrela*. Algumas dessas vantagens é a sua auto-organização e sua auto-configuração, o que faz com que cada nó adicionado na rede seja automaticamente reconhecido e passe a fazer parte dela. Enquanto que na topologia *estrela* todos os nós estão conectados a um único nó central, sendo que somente este nó tem acesso à camada superior da rede. Na figura 11 podemos ver uma comparação na arquitetura entre as duas topologias.

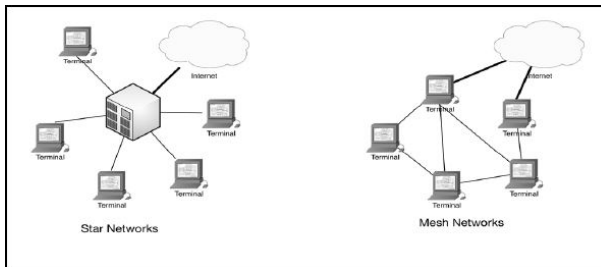


Figura 11 – Topologia estrela e topologia *mesh*

Fonte: Adaptado de Akyildiz; Wang (2009)

Além destas principais vantagens, outros pontos importantes trazem benefícios às redes *mesh*, como:

- escalabilidade, onde existe a possibilidade de aumentar a rede sem comprometer a capacidade do canal, além de poder ser incrementada durante sua existência.
- comunicação entre nós fora do ponto de visão, onde a conexão se faz através de nó-a-nó.
- mobilidade, onde os nós podem ser estacionários ou móveis, facilitando muitas aplicações de *IoT*.
- baixo consumo energético, onde principalmente clientes requerem protocolos de baixo consumo. Um exemplo é uma rede de sensores, onde cada nó depende de baterias para funcionar.
- interoperabilidade, capaz de se comunicar com outras redes existentes, como *WiMAX* e *ZigBee* (AKYILDIZ; WANG, 2009).

5 DESENVOLVIMENTO

Neste capítulo será apresentado o detalhamento do sistema desenvolvido para o controle de vagas em um estacionamento, bem como suas partes de hardware e software.

5.1 DESCRIÇÃO

Um sistema embarcado foi projetado e desenvolvido utilizando as tecnologias descritas nos capítulos anteriores, como: microcontroladores, comunicação sem fios e sensores ultrassônicos. Neste sistema, o usuário acessa uma página através do navegador, que é responsiva e pode ser acessada por qualquer dispositivo (*PC, smartphone, tablet*, etc) e pode saber onde existe uma vaga livre e seu tipo, que pode ser normal, para idosos ou para deficientes.

O sistema é composto por nós ordinários, que são os nós vagas e monitoram o status da vaga correspondente, um nó mestre, que, conectado à rede *WiFi*, recebe as mensagens dos nós ordinários e as envia para o servidor, e um servidor que atende às requisições dos clientes e do nó mestre. Na figura 12 podemos ver como o sistema está dividido:

Ao serem energizados, os nós ordinários se conectam automaticamente ao nó mestre através de uma rede sem fios *WiFi* formando uma rede *mesh*, conforme descrita no capítulo 4.

O nó ordinário monitora a ocupação da vaga através do sensor de ultrassom medindo a distância entre o sensor e o objeto mais próximo. Comparando com um valor pré-estabelecido, ele define se está livre ou ocupada e envia uma mensagem para o nó mestre com sua identificação e o status. Ao receber a mensagem do nó ordinário, o nó mestre envia ao servidor *web*, via *socket*, uma mensagem no protocolo *JSON* contendo o tipo da mensagem, o ID da vaga e o status.

Ao receber a mensagem, o servidor a interpreta de acordo com seu tipo faz o tratamento necessário. No caso de uma mensagem do tipo "marcaVaga", o servidor consulta um banco de dados, com os nós pré cadastrados, para saber qual o número da vaga correspondente ao ID do nó. Neste momento, o servidor envia uma mensagem *broadcast* para todos os clientes conectados (aplicativos móveis) informando a vaga a ser marcada, com seu tipo e status.

Neste momento, o cliente tem uma visão atualizada das condições do estacionamento, que pode ser acessada de qualquer lugar com

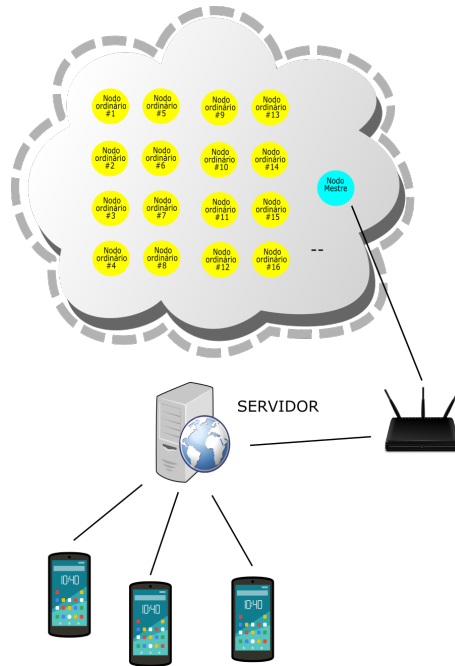


Figura 12 – Estrutura do sistema desenvolvido
 Fonte: O autor

acesso à internet, facilitando a busca por uma vaga. Além disso, o usuário pode, a partir das condições em sua tela, decidir se entra no estacionamento ou procura um outro estabelecimento.

5.1.1 Requisitos do Sistema

Os requisitos definidos para o desenvolvimento do sistema proposto são divididos em:

5.1.1.1 Requisitos Funcionais

1. Ser totalmente sem-fios e utilizar da infraestrutura de redes sem fios existente;
2. Atualização em tempo real do status das vagas do estaciona-

mento;

3. Permitir o crescimento incremental dos nós instalados, ou seja, não é necessário implantar todos os nós sensores de uma vez;
4. Possuir auto-configuração ao adicionar novas vagas ou substituir o nó sensor em caso de problema;

5.1.1.2 Requisitos não funcionais

1. Interface amigável com o cliente (*PC, smartphone e tablet*);
2. baixo custo;
3. Período de atualização do status de cada vaga não maior do que 10 segundos;

5.2 HARDWARE

O projeto de *hardware* foi pensado buscando o menor custo, menor tamanho e conexão sem-fios. Para isto, foi escolhido o microcontrolador modelo ESP-8266 (figura 13), que possui módulo *WiFi* para conexão sem fio e baixo consumo de energia. Para os testes também foi utilizado o modelo ESP-32, que é semelhante ao ESP-8266, pela sua disponibilidade no momento.

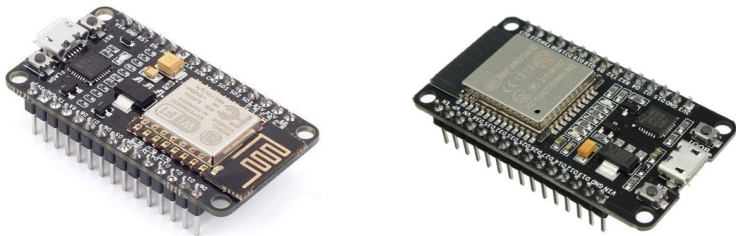


Figura 13 – ESP-8266 (E) e ESP-32 (D)

Fonte: Google imagens

O sensor ultrassônico escolhido foi o modelo HC-SR04P (Figura 14), que tem um ângulo de captação de 15° , o suficiente para o monitoramento do tamanho de um carro na vaga. Sua tensão de funcionamento é de 3 V a 5 V, permitindo trabalhar com o microcontrolador ESP. Quanto ao consumo, a corrente em modo de espera é menor que 2 mA, e de trabalho é de 2,8 mA. Na Figura 15 podemos ver o diagrama de funcionamento do sensor.



Figura 14 – Sensor ultrassônico HC-SR04P
Fonte: Google imagens

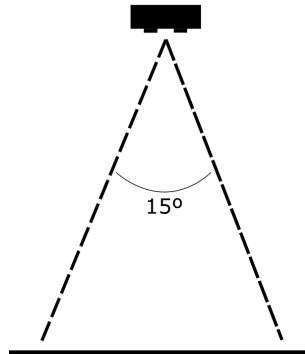


Figura 15 – Ilustração do funcionamento do sensor ultrassônico
Fonte: O autor

5.2.1 Protótipo

Um protótipo do sensor foi montado para ilustrar a possibilidade de um produto. Foi pensado a fácil instalação e manutenção, tamanho reduzido, e visual simples. Na Figura 16 podemos ver protótipo:

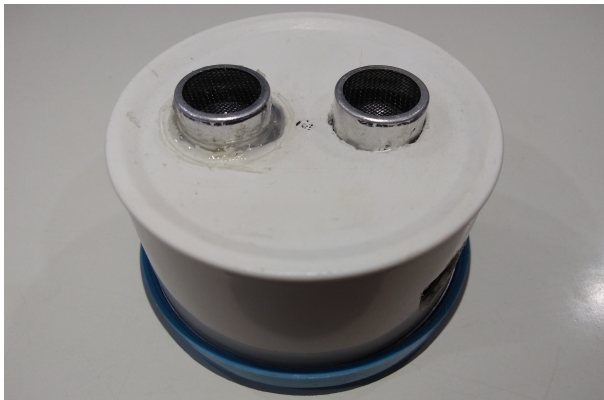


Figura 16 – Protótipo do sensor
Fonte: O autor

Dentro deste nó existe um microcontrolador e um sensor ultrasom, o esquema eletrônico dentro do nó está representado na Figura 17.

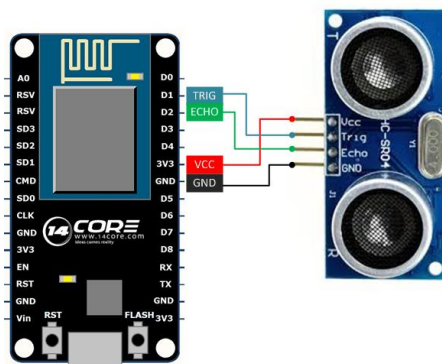


Figura 17 – Esquemático de ligação do sensor
Fonte: O autor

5.3 SOFTWARE

O projeto de *software* é composto por três partes principais: uma interface *web* onde o usuário acessa para verificar o status do estacionamento, um servidor para atender as requisições dos clientes e do

nó mestre, e um *firmware* desenvolvido para o microcontrolador. Nas seções seguintes são detalhados cada parte dos *softwares* apresentados anteriormente. Todo o código-fonte desenvolvido está disponível e pode ser acessado no repositório GitLab ¹.

5.3.1 Interface WEB do Cliente

A interface do cliente foi desenvolvida utilizando uma imagem do tipo *SVG*, que nada mais é do que um arquivo *XML* que contém *tags* específicas para gerar uma imagem vetorizada para a aplicação. Este tipo de arquivo possui algumas vantagens, como: facilidade na manipulação das imagens através da linguagem de programação *Javascript*, utilizada no desenvolvimento do servidor; por ser uma imagem vetorizada, é responsiva à diferentes formatos de leitura (ARAÚJO, 2011). O arquivo foi gerado através do *software Inkscape* e o resultado pode ser visualizado em sua versão PC na Figura 18 e mobile na Figura 19.



Figura 18 – Interface *web* versão *desktop*

Fonte: O autor

¹<https://gitlab.com/projetosfabiorocha/estacionamento>



Figura 19 – Interface *web* versão *mobile*

Fonte: O autor

Nesta interface o cliente pode visualizar o *layout* do estacionamento, bem como a situação atual de cada vaga e seu tipo.

5.3.2 Servidor Web

Um servidor foi desenvolvido para atender as mensagens enviadas pelo mestre e atender as requisições do cliente. O servidor foi escrito em linguagem de programação *JavaScript* utilizando a plataforma *Node.js* e a *framework Express*. A plataforma *Node.js* foi escolhida devido sua grande capacidade de escalabilidade (NODEBR, 2016a) e a *framework Express* devido sua flexibilidade e por fornecer um conjunto robusto de recursos para aplicativos web e móvel (NODEBR, 2016b).

O mecanismo de comunicação entre o nó mestre e o servidor é realizada através de *socket*, e o protocolo de comunicação utilizado foi o *JSON*. De acordo com (PINTO, 2012), *socket* é um mecanismo de comunicação, usado normalmente para implementar um modelo cli-

ente/servidor, que permite a troca de mensagens entre os processos de uma máquina/aplicação servidor e de uma máquina/aplicação cliente. *JSON* (JavaScript Object Notation) é um modelo para armazenamento e transmissão de informações no formato texto, que de acordo com (GONCALVES, 2012), tem sido bastante utilizado por aplicações Web devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo *XML*, tornando mais rápido o *parsing* dessas informações.

As funcionalidades do servidor incluem verificar se todos os nós vaga estão operantes, para isso, uma função chamada *checkAlive* é invocada a cada 10 segundos para fazer uma varredura em todos os nós conectados. Neste momento é comparado a data e hora atual com a data e hora da última mensagem recebida pelo nó, se for maior que um determinado valor seu *status* é marcado como *offline* e a vaga pintada de branco.

5.3.3 Banco de dados

Um banco de dados foi desenvolvido para armazenar as informações dos nós, como: ID da vaga, ID do microcontrolador e o tipo da vaga. O banco de dados foi desenvolvido utilizando o sistema *MongoDB*, que é um sistema de banco de dados do tipo *NoSQL*, ou seja, sem esquemas e orientado à documentos (JUNIOR, 2017). Para o gerenciamento do banco de dados foi utilizado o *Robo 3T* na sua versão 1.2, que é uma forma mais amigável de criar e gerenciar bancos de dados.

O banco de dados foi pensado para que antes da implantação do sistema proposto, possa ser cadastrado todas as vagas existentes no estacionamento. Na tabela 2 podemos ver a lista dos nós cadastrados no banco de dados.

<u>_id</u>	<u>VAGA_ID</u>	<u>TIPO_VAGA</u>
2786281334	8	deficiente
2131099496	14	normal
2751770441	1	idoso
3289213305	161	normal

Tabela 2 – Tabela dos nós cadastrados no banco de dados

5.3.4 Firmware

Para o sistema proposto, foram desenvolvidos dois *firmwares*, um para o nó ordinário e outro para o nó mestre. Como os microcontroladores utilizados, ESP-8266 e ESP-32, podem ser programados através da IDE do *arduino*, a linguagem de programação utilizada para escrever os códigos foi o *C++*.

5.3.4.1 Firmware nó ordinário

O sistema do nó ordinário tem a função de fazer a leitura da distância aferida pelo sensor de ultrassom e enviar uma mensagem com este valor ao nó mestre. A fim de diminuir o volume de mensagens enviadas, foi determinado um período de 5 segundos entre os envios, podendo ser alterado de acordo com a configuração desejada.

O sistema converte o valor lido em dois tipos de *status*: *ON* ou *OFF*. Onde *ON* significa que a vaga está ocupada e *OFF* que a vaga está desocupada. O valor determinado para este parâmetro foi de 200 cm, ou seja, se o valor lido for inferior à 200 cm a mensagem enviada será *ON*, caso contrário, a mensagem será *OFF*. Este parâmetro também pode ser alterado conforme a estrutura do estacionamento a ser instalado.

5.3.4.2 Firmware nó mestre

O sistema desenvolvido para o nó mestre implementa todas as funcionalidades necessárias para a criação e manutenção da rede *mesh*. Para isto, foi utilizado a biblioteca *painlessMesh*, e para a comunicação *wireless* foi utilizada a biblioteca *ESPAsyncWebServer*, ambas de código aberto e disponíveis pela comunidade. Com isso, o nó se conecta à rede *WiFi* local e simultaneamente cria uma rede *mesh* para os nós ordinários se conectarem.

O sistema principal recebe as mensagens enviadas pelos nós ordinários e as encaminham para o servidor via *socket* utilizando a biblioteca *WebSocketsClient*, uma biblioteca de código aberta e disponível para *download*. A mensagem enviada para o servidor utiliza o protocolo *JSON* e contém os atributos: "*type*", "*espID*" e "*status*".

6 TESTES E RESULTADOS

Após o desenvolvimento do sistema, foram realizados testes a fim de validar a proposta e hipótese. Foram realizados apenas testes funcionais, a fim de verificar se o sistema se comporta da forma esperada. Para validar os testes, foi utilizado um arquivo de *log*, no qual foi gravado cada mensagem trocada pela aplicação, possibilitando verificar a consistência das mensagens. Dividiremos os testes em duas etapas, primeiro testes para validar o funcionamento da rede *mesh* pela biblioteca *painlessMesh*, e os testes das aplicações desenvolvidas, como o servidor e a página *web*.

Os testes foram realizados através do protótipo desenvolvido, e por disponibilizar de cinco microcontroladores *ESP*, um foi definido como nó raiz e quatro representando as vagas, que foram distribuídas e cadastradas no banco de dados conforme a tabela 3.

ESP ID	VAGA ID	TIPO VAGA
2751770441	1	idoso
2786281334	8	deficiente
2131099496	14	normal
3289213305	161	normal

Tabela 3 – Tabela do banco de dados utilizado nos testes

6.1 TESTES E VALIDAÇÃO DA REDE *MESH*

O primeiro teste realizado foi para validar o funcionamento da rede *mesh*. Uma das características desta topologia é a auto-configuração e auto-organização, desta maneira, a cada nó adicionado, este deve se conectar ao nó mais próximo pertencente à rede. Os testes foram realizados adicionando os em diferentes distâncias e a rede se comportou como esperado.

Outro teste realizado para validar a rede *mesh* é quanto ao envio de mensagens entre nós que não estão no mesmo campo de visão, ou seja, não possuem conexão direta. Neste teste todas as mensagens enviadas por nós sem conexão direta com o nó raiz foram recebidas com sucesso.

6.2 TESTES DA APLICAÇÃO

Os testes realizados para validar a aplicação foram aplicados nos *firmwares*, no servidor e na interface *web*.

Nos *firmwares*, os testes foram feitos para validar se a mensagem enviada pelo nó vaga ao nó raiz, *ON* ou *OFF*, está de acordo com o valor de distância lida pelo sensor ultrassônico. Nos testes realizados a assertividade foi de 100%, portanto foi validado.

Para testar o servidor, foi simulado o funcionamento de um estacionamento com os nós existentes. Durante a simulação, todas mensagens enviadas foram atendidas pelo servidor e corretamente enviadas aos clientes. Não foi constatado nenhuma inconsistência na troca de mensagens e nos tratamentos efetuados pelo servidor.

Não foram realizados testes a fim de verificar a capacidade máxima, tanto de nós vaga quanto de clientes que o sistema suportaria, pois na literatura se sabe que os valores seriam muito superiores aos valores possíveis nos testes.

7 CONSIDERAÇÕES FINAIS

7.1 CONCLUSÃO

Com o desenvolvimento deste trabalho vimos que os conceitos de IoT podem ser aplicados para resolver o problema dos estacionamentos. Os sistemas de controle de vagas estão presentes no mercado atualmente, porém o apresentado neste trabalho oferece alguns recursos diferenciais, como a aplicação *web* onde o cliente pode verificar a situação do estacionamento. Além disso, o protótipo desenvolvido mostra o grande potencial para virar um produto comercial, podendo ser um sistema único e completo ou sendo adicionado à sistemas de automação já existentes.

As tecnologias utilizadas, como o microcontrolador *ESP* e o sensor ultrassônico se mostraram eficientes na captura e processamento das informações. A topologia de rede *mesh*, através da biblioteca *pythonlessMesh*, mostrou ser eficiente e robusta para aplicações de IoT, onde existem centenas ou até milhares de nós conectados, possibilitando a comunicação entre nós que estão fora do alcance direto.

Além disso, quanto aos *softwares* desenvolvidos, a aplicação móvel do cliente, responsiva e de fácil utilização, oferece uma visão completa do estacionamento e sua situação em tempo real, o que possibilita a tomada de decisão antes mesmo de sair de casa. E o servidor *web*, que com as tecnologias utilizadas, como *NodeJS* e *MongoDB*, demonstraram robustez e escalabilidade necessária para a implementação do sistema em larga escala.

7.2 TRABALHOS FUTUROS

Uma sugestão para trabalhos futuros é a adição da funcionalidade de reserva de vagas, a qual utilizaria GPS para levar o condutor até a vaga desejada. Junto disso, um sistema de cadastro e para que o controle de pagamentos fosse realizado diretamente na aplicação.

Outra sugestão é o desenvolvimento de sistema de alimentação por baterias, o qual dispensaria a necessidade de alimentação por cabos, facilitando ainda mais a instalação e manutenção dos equipamentos.

REFERÊNCIAS

- AKYILDIZ, I. F.; WANG, X. *Wireless Mesh Networks*. [S.l.]: Wiley, 2009. ISBN 0470032561.
- AL-FUQAHA, A. et al. Internet of Things : A Survey on Enabling. v. 17, n. 4, p. 2347–2376, 2015.
- ANDRADE, A. S. de Oliveira e Fernando Souza de. *Sistemas Embarcados. Hardware e Firmware na Prática*. [S.l.]: Érica, 2010. ISBN 8536501057.
- ARAÚJO, H. *Entendendo e usando o SVG*. 2011. [Online; Acessado em 15 de Outubro de 2018]. <<https://goo.gl/DEuDVp>>.
- ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things : A survey. *Computer Networks*, Elsevier B.V., v. 54, n. 15, p. 2787–2805, 2010. ISSN 1389-1286. <<http://dx.doi.org/10.1016/j.comnet.2010.05.010>>.
- ESPRESSIF. 2018. <<https://www.espressif.com/en/products/hardware/esp8266ex/overview>>.
- GONCALVES, E. C. *JSON Tutorial*. 2012. [Online; Acessado em 15 de Outubro de 2018]. <<https://goo.gl/cHUL6v>>.
- JUNIOR, L. F. D. *MongoDB para iniciantes em NoSQL*. 2017. [Online; Acessado em 15 de Outubro de 2018]. <<https://goo.gl/JbfTV1>>.
- KUMAR, S. et al. *The Codesign of Embedded Systems: A Unified Hardware/Software Representation*. [S.l.]: Springer, 2011. ISBN 9781461285533.
- LIU, Y.; TONG, K.-f. Wireless Mesh Networks in IoT Networks. p. 183–185, 2017.
- MIORANDI, D. et al. Ad Hoc Networks Internet of things : Vision , applications and research challenges. *Ad Hoc Networks*, Elsevier B.V., v. 10, n. 7, p. 1497–1516, 2012. ISSN 1570-8705. <<http://dx.doi.org/10.1016/j.adhoc.2012.02.016>>.
- NODEBR. *O que é Node.js?* 2016. [Online; Acessado em 15 de Outubro de 2018]. <<https://goo.gl/uN5gAx>>.

NODEBR. *Primeiros passos com Express em Node.js*. 2016. [Online; Acessado em 15 de Outubro de 2018]. <<https://goo.gl/WpfgXZ>>.


OLSON, N. The internet of things. *New Media & Society*, v. 20, n. 8, p. 3091–3092, 2018. <<https://doi.org/10.1177/1461444818776939c>>.

PINTO, P. *Redes – Sabe o que são sockets de comunicação? (Parte I)*. 2012. [Online; Acessado em 15 de Outubro de 2018]. <<https://goo.gl/G6oKkP>>.

ROSE, K.; ELDRIDGE, S.; CHAPIN, L. THE INTERNET OF THINGS: AN OVERVIEW. Understanding the Issues and Challenges of a More Connected World. *The Internet Society*, n. October, p. 80, 2015. ISSN 1536-5026. <<http://electronicdesign.com/communications/internet-things-needs-firewalls-too>>.

SIMON, D. E. *An Embedded Software Primer*. [S.l.]: Addison-Wesley Professional, 1999. ISBN 020161569X.

ANEXO A - Datasheet ESP 8266




ESP8266EX

Datasheet



Version 6.0
Espressif Systems
Copyright © 2018



About This Guide

This document introduces the specifications of ESP8266EX.

Release Notes

Date	Version	Release Notes
2015.12	V4.6	Updated Chapter 3.
2016.02	V4.7	Updated Section 3.6 and Section 4.1.
2016.04	V4.8	Updated Chapter 1.
2016.08	V4.9	Updated Chapter 1.
2016.11	V5.0	Added Appendix II "Learning Resources".
2016.11	V5.1	Changed the power consumption during Deep-sleep from 10 μ A to 20 μ A in Table 5-2.
2016.11	V5.2	Changed the crystal frequency range from "26 MHz to 52 MHz" to "24 MHz to 52 MHz" in Section 3.3.
2016.12	V5.3	Changed the minimum working voltage from 3.0V to 2.5V.
2017.04	V5.4	Changed chip input and output impedance from 50 Ω to 39+j6 Ω .
2017.10	V5.5	Updated Chapter 3 regarding the range of clock amplitude to 0.8 ~ 1.5V.
2017.11	V5.6	Updated VDDPST from 1.8V ~ 3.3V to 1.8V ~ 3.6V.
2017.11	V5.7	<ul style="list-style-type: none">• Corrected a typo in the description of SDIO_DATA_0 in Table 2-1;• Added the testing conditions for the data in Table 5-2.

Date	Version	Release Notes
2018.02	V5.8	<ul style="list-style-type: none"> • Updated Wi-Fi protocols in Section 1.1; • Updated description of the integrated Tensilica processor in 3.1.
2018.09	V5.9	<ul style="list-style-type: none"> • Update document cover; • Added a note for Table 1-1; • Updated Wi-Fi key features in Section 1.1; • Updated description of the Wi-Fi function in 3.5; • Updated pin layout diagram; • Fixed a typo in Table 2-1; • Removed Section AHB and AHB module; • Restructured Section Power Management; • Fixed a typo in Section UART; • Removed description of transmission angle in Section IR Remote Control; • Other optimization (wording).
2018.11	V6.0	<ul style="list-style-type: none"> • Added an SPI pin in Table 4-2; • Updated the diagram of packing information.

Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe at <https://www.espressif.com/en/subscribe>.

Certification

Download certificates for Espressif products from <https://www.espressif.com/en/certificates>.

Table of Contents

1. Overview	1
1.1. Wi-Fi Key Features	1
1.2. Specifications	2
1.3. Applications	3
2. Pin Definitions	4
3. Functional Description	6
3.1. CPU, Memory, and Flash	6
3.1.1. CPU	6
3.1.2. Memory	6
3.1.3. External Flash	7
3.2. Clock	7
3.2.1. High Frequency Clock	7
3.2.2. External Clock Requirements	8
3.3. Radio	8
3.3.1. Channel Frequencies	8
3.3.2. 2.4 GHz Receiver	9
3.3.3. 2.4 GHz Transmitter	9
3.3.4. Clock Generator	9
3.4. Wi-Fi	9
3.4.1. Wi-Fi Radio and Baseband	9
3.4.2. Wi-Fi MAC	10
3.5. Power Management	10
4. Peripheral Interface	12
4.1. General Purpose Input/Output Interface (GPIO)	12
4.2. Secure Digital Input/Output Interface (SDIO)	12
4.3. Serial Peripheral Interface (SPI/HSPI)	13
4.3.1. General SPI (Master/Slave)	13
4.3.2. HSPI (Slave)	13
4.4. I2C Interface	14
4.5. I2S Interface	14
4.6. Universal Asynchronous Receiver Transmitter (UART)	14
4.7. Pulse-Width Modulation (PWM)	15
4.8. IR Remote Control	16
4.9. ADC (Analog-to-Digital Converter)	16

5. Electrical Specifications	18
5.1. Electrical Characteristics.....	18
5.2. RF Power Consumption	18
5.3. Wi-Fi Radio Characteristics	19
6. Package Information	20
I. Appendix - Pin List	21
II. Appendix - Learning Resources	22
II.1. Must-Read Documents	22
II.2. Must-Have Resources.....	22



1.

Overview

Espressif's ESP8266EX delivers highly integrated Wi-Fi SoC solution to meet users' continuous demands for efficient power usage, compact design and reliable performance in the Internet of Things industry.

With the complete and self-contained Wi-Fi networking capabilities, ESP8266EX can perform either as a standalone application or as the slave to a host MCU. When ESP8266EX hosts the application, it promptly boots up from the flash. The integrated high-speed cache helps to increase the system performance and optimize the system memory. Also, ESP8266EX can be applied to any microcontroller design as a Wi-Fi adaptor through SPI/SDIO or UART interfaces.

ESP8266EX integrates antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules. The compact design minimizes the PCB size and requires minimal external circuitries.

Besides the Wi-Fi functionalities, ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor and on-chip SRAM. It can be interfaced with external sensors and other devices through the GPIOs. Software Development Kit (SDK) provides sample codes for various applications.

Espressif Systems' Smart Connectivity Platform (ESCP) enables sophisticated features including:

- Fast switch between sleep and wakeup mode for energy-efficient purpose;
- Adaptive radio biasing for low-power operation
- Advance signal processing
- Spur cancellation and RF co-existence mechanisms for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation

1.1. Wi-Fi Key Features

- 802.11 b/g/n support
- 802.11n support (2.4 GHz), up to 72.2 Mbps
- Defragmentation
- 2 x virtual Wi-Fi interface
- Automatic beacon monitoring (hardware TSF)
- Support Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode
- Antenna diversity



1.2. Specifications

Table 1-1. Specifications

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4G ~ 2.5G (2400M ~ 2483.5M)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
802.11 g: -75 dbm (54 Mbps)		
802.11 n: -72 dbm (MCS7)		
Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip	
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5V ~ 3.6V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	-40°C ~ 125°C
	Package Size	QFN32-pin (5 mm x 5 mm)
External Interface	-	
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Note:

The TX power can be configured based on the actual user scenarios.



1.3. Applications

- Home appliances
- Home automation
- Smart plugs and lights
- Industrial wireless control
- Baby monitors
- IP cameras
- Sensor networks
- Wearable electronics
- Wi-Fi location-aware devices
- Security ID tags
- Wi-Fi position system beacons



2. Pin Definitions

Figure 2-1 shows the pin layout for 32-pin QFN package.

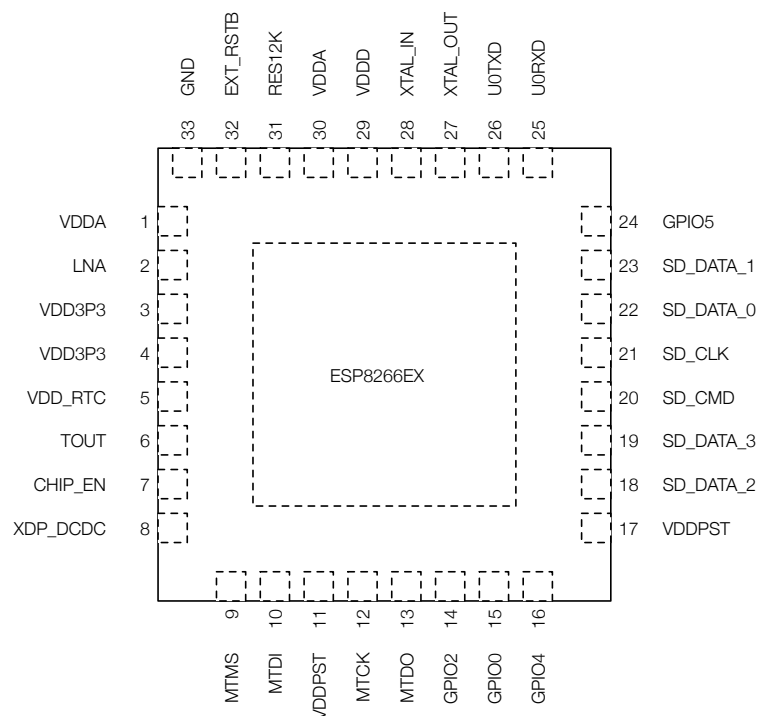


Figure 2-1. Pin Layout (Top View)

Table 2-1 lists the definitions and functions of each pin.

Table 2-1. ESP8266EX Pin Definitions

Pin	Name	Type	Function
1	VDDA	P	Analog Power 2.5V ~ 3.6V
2	LNA	I/O	RF antenna interface Chip output impedance= $-39+j6 \Omega$. It is suggested to retain the π -type matching network to match the antenna.
3	VDD3P3	P	Amplifier Power 2.5V ~ 3.6V
4	VDD3P3	P	Amplifier Power 2.5V ~ 3.6V
5	VDD_RTC	P	NC (1.1V)
6	TOUT	I	ADC pin. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously.



Pin	Name	Type	Function
7	CHIP_EN	I	Chip Enable High: On, chip works properly Low: Off, small current consumed
8	XPD_DCDC	I/O	Deep-sleep wakeup (need to be connected to EXT_RSTB); GPIO16
9	MTMS	I/O	GPIO 14; HSPI_CLK
10	MTDI	I/O	GPIO 12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8V ~ 3.6V)
12	MTCK	I/O	GPIO 13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO 15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART TX during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8V ~ 3.6V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 200Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200Ω); SPI_MISO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART TX during flash programming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 2.5V ~ 3.6V
30	VDDA	P	Analog Power 2.5V ~ 3.6V
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: active)

Note:

1. GPIO2, GPIO0, and MTDO are used to select booting mode and the SDIO mode;
2. U0TXD should not be pulled externally to a low logic level during the powering-up.



3. Functional Description

The functional diagram of ESP8266EX is shown as in Figure 3-1.

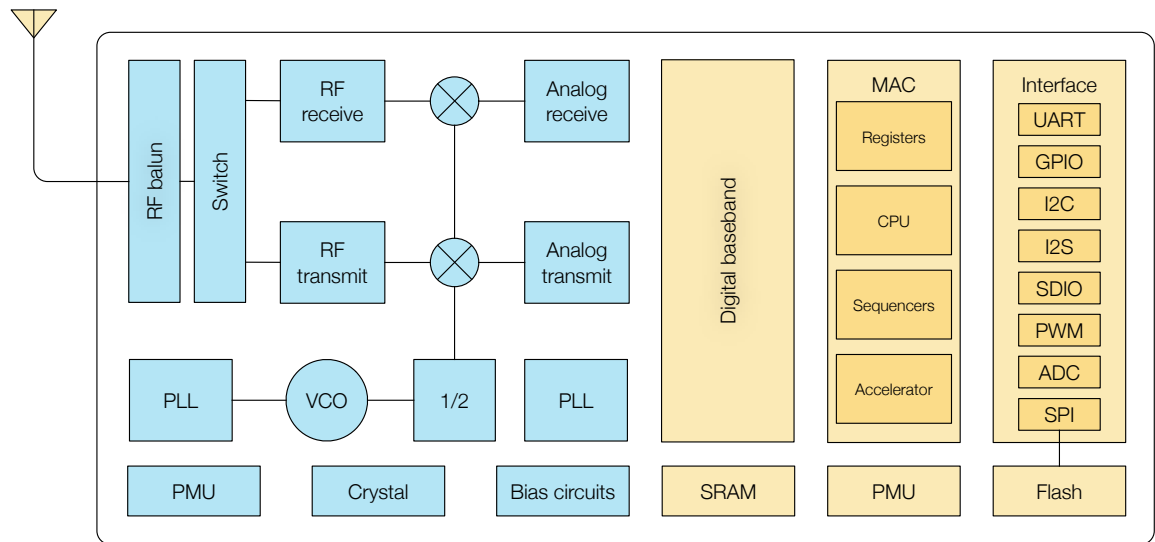


Figure 3-1. Functional Block Diagram

3.1. CPU, Memory, and Flash

3.1.1. CPU

The ESP8266EX integrates a Tensilica L106 32-bit RISC processor, which achieves extra-low power consumption and reaches a maximum clock speed of 160 MHz. The Real-Time Operating System (RTOS) and Wi-Fi stack allow 80% of the processing power to be available for user application programming and development. The CPU includes the interfaces as below:

- Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can also be used to visit flash.
- Data RAM interface (dBus), which can be connected with memory controller.
- AHB interface which can be used to visit the register.

3.1.2. Memory

ESP8266EX Wi-Fi SoC integrates memory controller and memory units including SRAM and ROM. MCU can access the memory units through iBus, dBus, and AHB interfaces. All memory units can be accessed upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK, SRAM space available to users is assigned as below.



- RAM size < 50 kB, that is, when ESP8266EX is working under the Station mode and connects to the router, the maximum programmable space accessible in Heap + Data section is around 50 kB.
- There is no programmable ROM in the SoC. Therefore, user program must be stored in an external SPI flash.

3.1.3. External Flash

ESP8266EX uses external SPI flash to store user programs, and supports up to 16 MB memory capacity theoretically.

The minimum flash memory of ESP8266EX is shown below:

- OTA disabled: 512 kB at least
- OTA enabled: 1 MB at least

! Notice:

SPI mode supported: Standard SPI, Dual SPI and Quad SPI. The correct SPI mode should be selected when flashing bin files to ESP8266. Otherwise, the downloaded firmware/program may not be working properly.

3.2. Clock

3.2.1. High Frequency Clock

The high frequency clock on ESP8266EX is used to drive both transmit and receive mixers. This clock is generated from internal crystal oscillator and external crystal. The crystal frequency ranges from 24 MHz to 52 MHz.

The internal calibration inside the crystal oscillator ensures that a wide range of crystals can be used, nevertheless the quality of the crystal is still a factor to consider to have reasonable phase noise and good Wi-Fi sensitivity. Refer to Table 3-1 to measure the frequency offset.

Table 3-1. High Frequency Clock Specifications

Parameter	Symbol	Min	Max	Unit
Frequency	FXO	24	52	MHz
Loading capacitance	CL	-	32	pF
Motional capacitance	CM	2	5	pF
Series resistance	RS	0	65	Ω
Frequency tolerance	Δ FXO	-15	15	ppm
Frequency vs temperature (-25°C ~ 75°C)	Δ FXO,Temp	-15	15	ppm



3.2.2. External Clock Requirements

An externally generated clock is available with the frequency ranging from 24 MHz to 52 MHz. The following characteristics are expected to achieve good performance of radio.

Table 3-2. External Clock Reference

Parameter	Symbol	Min	Max	Unit
Clock amplitude	VXO	0.8	1.5	V _{pp}
External clock accuracy	Δ FXO,EXT	-15	15	ppm
Phase noise @1-kHz offset, 40-MHz clock	-	-	-120	dBc/Hz
Phase noise @10-kHz offset, 40-MHz clock	-	-	-130	dBc/Hz
Phase noise @100-kHz offset, 40-MHz clock	-	-	-138	dBc/Hz

3.3. Radio

ESP8266EX radio consists of the following blocks.

- 2.4 GHz receiver
- 2.4 GHz transmitter
- High speed clock generators and crystal oscillator
- Bias and regulators
- Power management

3.3.1. Channel Frequencies

The RF transceiver supports the following channels according to IEEE802.11b/g/n standards.

Table 3-3. Frequency Channel

Channel No.	Frequency (MHz)	Channel No.	Frequency (MHz)
1	2412	8	2447
2	2417	9	2452
3	2422	10	2457
4	2427	11	2462
5	2432	12	2467
6	2437	13	2472
7	2442	14	2484



3.3.2. 2.4 GHz Receiver

The 2.4 GHz receiver down-converts the RF signals to quadrature baseband signals and converts them to the digital domain with 2 high resolution high speed ADCs. To adapt to varying signal channel conditions, RF filters, automatic gain control (AGC), DC offset cancelation circuits and baseband filters are integrated within ESP8266EX.

3.3.3. 2.4 GHz Transmitter

The 2.4 GHz transmitter up-converts the quadrature baseband signals to 2.4 GHz, and drives the antenna with a high-power CMOS power amplifier. The function of digital calibration further improves the linearity of the power amplifier, enabling a state of art performance of delivering +19.5 dBm average TX power for 802.11b transmission and +18 dBm for 802.11n (MCS0) transmission.

Additional calibrations are integrated to offset any imperfections of the radio, such as:

- Carrier leakage
- I/Q phase matching
- Baseband nonlinearities

These built-in calibration functions reduce the product test time and make the test equipment unnecessary.

3.3.4. Clock Generator

The clock generator generates quadrature 2.4 GHz clock signals for the receiver and transmitter. All components of the clock generator are integrated on the chip, including all inductors, varactors, loop filters, linear voltage regulators and dividers.

The clock generator has built-in calibration and self test circuits. Quadrature clock phases and phase noise are optimized on-chip with patented calibration algorithms to ensure the best performance of the receiver and transmitter.

3.4. Wi-Fi

ESP8266EX implements TCP/IP and full 802.11 b/g/n WLAN MAC protocol. It supports Basic Service Set (BSS) STA and SoftAP operations under the Distributed Control Function (DCF). Power management is handled with minimum host interaction to minimize active-duty period.

3.4.1. Wi-Fi Radio and Baseband

The ESP8266EX Wi-Fi Radio and Baseband support the following features:

- 802.11b and 802.11g
- 802.11n MCS0-7 in 20 MHz bandwidth
- 802.11n 0.4 μ s guard-interval
- up to 72.2 Mbps of data rate



- Receiving STBC 2x1
- Up to 20.5 dBm of transmitting power
- Adjustable transmitting power
- Antenna diversity

3.4.2. Wi-Fi MAC

The ESP8266EX Wi-Fi MAC applies low-level protocol functions automatically, as follows:

- 2 × virtual Wi-Fi interfaces
- Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode
- Request To Send (RTS), Clear To Send (CTS) and Immediate Block ACK
- Defragmentation
- CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WEP (RC4) and CRC
- Automatic beacon monitoring (hardware TSF)
- Dual and single antenna Bluetooth co-existence support with optional simultaneous receive (Wi-Fi/Bluetooth) capability

3.5. Power Management

ESP8266EX is designed with advanced power management technologies and intended for mobile devices, wearable electronics and the Internet of Things applications.

The low-power architecture operates in the following modes:

- Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.
- Modem-sleep mode: The CPU is operational. The Wi-Fi and radio are disabled.
- Light-sleep mode: The CPU and all peripherals are paused. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
- Deep-sleep mode: Only the RTC is operational and all other part of the chip are powered off.

Table 3-4. Power Consumption by Power Modes

Power Mode	Description	Power Consumption
Active (RF working)	Wi-Fi TX packet	Please refer to 5-2.
	Wi-Fi RX packet	
Modem-sleep ^①	CPU is working	15 mA
Light-sleep ^②	-	0.9 mA
Deep-sleep ^③	Only RTC is working	20 uA
Shut down	-	0.5 uA

**Notes:**

- ① **Modem-sleep** mode is used in the applications that require the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it shuts down the Wi-Fi Modem circuit while maintaining a Wi-Fi connection with no data transmission to optimize power consumption. E.g. in DTIM3, maintaining a sleep of 300 ms with a wakeup of 3 ms cycle to receive AP's Beacon packages at interval requires about 15 mA current.
- ② During **Light-sleep** mode, the CPU may be suspended in applications like Wi-Fi switch. Without data transmission, the Wi-Fi Modem circuit can be turned off and CPU suspended to save power consumption according to the 802.11 standards (U-APSD). E.g. in DTIM3, maintaining a sleep of 300 ms with a wakeup of 3ms to receive AP's Beacon packages at interval requires about 0.9 mA current.
- ③ During **Deep-sleep** mode, Wi-Fi is turned off. For applications with long time lags between data transmission, e.g. a temperature sensor that detects the temperature every 100s, sleeps for 300s and wakes up to connect to the AP (taking about 0.3 ~ 1s), the overall average current is less than 1mA. The current of 20 μ A is acquired at the voltage of 2.5V.



4. Peripheral Interface

4.1. General Purpose Input/Output Interface (GPIO)

ESP8266EX has 17 GPIO pins which can be assigned to various functions by programming the appropriate registers.

Each GPIO PAD can be configured with internal pull-up or pull-down (XPD_DCDC can only be configured with internal pull-down, other GPIO PAD can only be configured with internal pull-up), or set to high impedance. When configured as an input, the data are stored in software registers; the input can also be set to edge-trigger or level trigger CPU interrupts. In short, the IO pads are bi-directional, non-inverting and tristate, which includes input and output buffer with tristate control inputs.

These pins, when working as GPIOs, can be multiplexed with other functions such as I2C, I2S, UART, PWM, and IR Remote Control, etc.

For low power operations, the GPIOs can also be set to hold their state. For instance, when the IOs are not driven by internal and external circuits, all outputs will hold their states before the chip entered the low power modes.

The required drive strength is small— 5 μ A or more is enough to pull apart the latch.

4.2. Secure Digital Input/Output Interface (SDIO)

ESP8266EX has one Slave SDIO, the definitions of which are described as Table 4-1, which supports 25 MHz SDIO v1.1 and 50 MHz SDIO v2.0, and 1 bit/4 bit SD mode and SPI mode.

Table 4-1. Pin Definitions of SDIOs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SDIO_CLK
SDIO_DATA0	22	IO7	SDIO_DATA0
SDIO_DATA1	23	IO8	SDIO_DATA1
SDIO_DATA_2	18	IO9	SDIO_DATA_2
SDIO_DATA_3	19	IO10	SDIO_DATA_3
SDIO_CMD	20	IO11	SDIO_CMD



4.3. Serial Peripheral Interface (SPI/HSPI)

ESP8266EX has two SPIs.

- One general Slave/Master SPI
- One general Slave HSPI

Functions of all these pins can be implemented via hardware.

4.3.1. General SPI (Master/Slave)

Table 4-2. Pin Definitions of SPIs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SPICLK
SDIO_DATA0	22	IO7	SPIQ/MISO
SDIO_DATA1	23	IO8	SPID/MOSI
SDIO_DATA_2	18	IO9	SPIHD
SDIO_DATA_3	19	IO10	SPIWP
U0TXD	26	IO1	SPICS1
GPIO0	15	IO0	SPICS2
SDIO_CMD	20	IO11	SPICS0

Note:

SPI mode can be implemented via software programming. The clock frequency is 80 MHz at maximum when working as a master, 20 MHz at maximum when working as a slave.

4.3.2. HSPI (Slave)

Table 4-3. Pin Definitions of HSPI (Slave)

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	HSPICLK
MTDI	10	IO12	HSPIQ/MISO
MTCK	12	IO13	HSPID/MOSI
MTDO	13	IO15	HPSICS

Note:

SPI mode can be implemented via software programming. The clock frequency is 20 MHz at maximum.



4.4. I2C Interface

ESP8266EX has one I2C, which is realized via software programming, used to connect with other microcontrollers and other peripheral equipments such as sensors. The pin definition of I2C is as below.

Table 4-4. Pin Definitions of I2C

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	I2C_SCL
GPIO2	14	IO2	I2C_SDA

Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized via software programming, and the clock frequency is 100 kHz at maximum.

4.5. I2S Interface

ESP8266EX has one I2S data input interface and one I2S data output interface, and supports the linked list DMA. I2S interfaces are mainly used in applications such as data collection, processing, and transmission of audio data, as well as the input and output of serial data. For example, LED lights (WS2812 series) are supported. The pin definition of I2S is shown in Table 4-5.

Table 4-5. Pin Definitions of I2S

I2S Data Input			
Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	I2SI_DATA
MTCK	12	IO13	I2SI_BCK
MTMS	9	IO14	I2SI_WS
MTDO	13	IO15	I2SO_BCK
U0RXD	25	IO3	I2SO_DATA
GPIO2	14	IO2	I2SO_WS

4.6. Universal Asynchronous Receiver Transmitter (UART)

ESP8266EX has two UART interfaces UART0 and UART1, the definitions are shown in Table 4-6.



Table 4-6. Pin Definitions of UART

Pin Type	Pin Name	Pin Num	IO	Function Name
UART0	U0RXD	25	IO3	U0RXD
	U0TXD	26	IO1	U0TXD
	MTDO	13	IO15	U0RTS
	MTCK	12	IO13	U0CTS
UART1	GPIO2	14	IO2	U1TXD
	SD_D1	23	IO8	U1RXD

Data transfers to/from UART interfaces can be implemented via hardware. The data transmission speed via UART interfaces reaches 115200 x 40 (4.5 Mbps).

UART0 can be used for communication. It supports flow control. Since UART1 features only data transmit signal (TX), it is usually used for printing log.

Note:

By default, UART0 outputs some printed information when the device is powered on and booting up. The baud rate of the printed information is relevant to the frequency of the external crystal oscillator. If the frequency of the crystal oscillator is 40 MHz, then the baud rate for printing is 115200; if the frequency of the crystal oscillator is 26 MHz, then the baud rate for printing is 74880. If the printed information exerts any influence on the functionality of the device, it is suggested to block the printing during the power-on period by changing (U0TXD, U0RXD) to (MTDO, MTCK).

4.7. Pulse-Width Modulation (PWM)

ESP8266EX has four PWM output interfaces. They can be extended by users themselves. The pin definitions of the PWM interfaces are defined as below.

Table 4-7. Pin Definitions of PWM

Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	PWM0
MTDO	13	IO15	PWM1
MTMS	9	IO14	PWM2
GPIO4	16	IO4	PWM3

The functionality of PWM interfaces can be implemented via software programming. For example, in the LED smart light demo, the function of PWM is realized by interruption of the timer, the minimum resolution reaches as high as 44 ns. PWM frequency range is adjustable from 1000 μ s to 10000 μ s, i.e., between 100 Hz and 1 kHz. When the PWM frequency is 1 kHz, the duty ratio will be 1/22727, and a resolution of over 14 bits will be achieved at 1 kHz refresh rate.



4.8. IR Remote Control

ESP8266EX currently supports one infrared remote control interface. For detailed pin definitions, please see Table 4-8 below.

Table 4-8. Pin Definitions of IR Remote Control

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	IR TX
GPIO5	24	IO 5	IR Rx

The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are supported by this interface. The frequency of modulated carrier signal is 38 kHz, while the duty ratio of the square wave is 1/3. The transmission range is around 1m which is determined by two factors: one is the maximum current drive output, the other is internal current-limiting resistance value in the infrared receiver. The larger the resistance value, the lower the current, so is the power, and vice versa.

4.9. ADC (Analog-to-Digital Converter)

ESP8266EX is embedded with a 10-bit precision SAR ADC. TOUT (Pin6) is defined as below:

Table 4-9. Pin Definition of ADC

Pin Name	Pin Num	Function Name
TOUT	6	ADC Interface

The following two measurements can be implemented using ADC (Pin6). However, they cannot be implemented at the same time.

- Measure the power supply voltage of VDD3P3 (Pin3 and Pin4).

Hardware Design	TOUT must be floating.
RF Initialization Parameter	The 107th byte of <i>esp_init_data_default.bin</i> (0 ~ 127 bytes), <i>vdd33_const</i> must be set to <code>0xFF</code> .
RF Calibration Process	Optimize the RF circuit conditions based on the testing results of VDD3P3 (Pin3 and Pin4).
User Programming	Use <code>system_get_vdd33</code> instead of <code>system_adc_read</code> .

- Measure the input voltage of TOUT (Pin6).

Hardware Design	The input voltage range is 0 to 1.0V when TOUT is connected to external circuit.
-----------------	--



RF Initialization Parameter	The value of the 107th byte of esp_init_data_default.bin (0 ~ 127 bytes), vdd33_const must be set to the real power supply voltage of Pin3 and Pin4. The unit and effective value range of vdd33_const is 0.1V and 18 to 36, respectively, thus making the working power voltage range of ESP8266EX between 1.8V and 3.6V,
RF Calibration Process	Optimize the RF circuit conditions based on the value of vdd33_const . The permissible error is $\pm 0.2V$.
User Programming	Use <code>system_adc_read</code> instead of <code>system_get_vdd33</code> .

Notes:

esp_init_data_default.bin is provided in SDK package which contains RF initialization parameters (0 ~ 127 bytes). The name of the 107th byte in **esp_init_data_default.bin** is **vdd33_const**, which is defined as below:

- When **vdd33_const** = 0xff, the power voltage of Pin3 and Pin4 will be tested by the internal self-calibration process of ESP8266EX itself. RF circuit conditions should be optimized according to the testing results.
- When $18 \leq \text{vdd33_const} \leq 36$, ESP8266EX RF Calibration and optimization process is implemented via ($\text{vdd33_const}/10$).
- When **vdd33_const** < 18 or $36 < \text{vdd33_const} < 255$, **vdd33_const** is invalid. ESP8266EX RF Calibration and optimization process is implemented via the default value 3.3V.



5. Electrical Specifications

5.1. Electrical Characteristics

Table 5-1. Electrical Characteristics

Parameters	Conditions	Min	Typical	Max	Unit
Operating Temperature Range	-	-40	Normal	125	°C
Maximum Soldering Temperature	IPC/JEDEC J-STD-020	-	-	260	°C
Working Voltage Value	-	2.5	3.3	3.6	V
I/O	V_{IL}	-	-0.3	-	$0.25V_{IO}$
	V_{IH}	-	$0.75V_{IO}$	-	3.6
	V_{OL}	-	-	-	$0.1V_{IO}$
	V_{OH}	-	$0.8V_{IO}$	-	-
	I_{MAX}	-	-	-	12
Electrostatic Discharge (HBM)	TAMB=25°C	-	-	2	KV
Electrostatic Discharge (CDM)	TAMB=25°C	-	-	0.5	KV

5.2. RF Power Consumption

Unless otherwise specified, the power consumption measurements are taken with a 3.0V supply at 25°C of ambient temperature. All transmitters' measurements are based on a 50% duty cycle.

Table 5-2. Power Consumption

Parameters	Min	Typical	Max	Unit
TX 802.11b, CCK 11Mbps, $P_{OUT}=+17$ dBm	-	170	-	mA
TX 802.11g, OFDM 54Mbps, $P_{OUT}=+15$ dBm	-	140	-	mA
TX 802.11n, MCS7, $P_{OUT}=+13$ dBm	-	120	-	mA
Rx 802.11b, 1024 bytes packet length, -80 dBm	-	50	-	mA
Rx 802.11g, 1024 bytes packet length, -70 dBm	-	56	-	mA
Rx 802.11n, 1024 bytes packet length, -65 dBm	-	56	-	mA



5.3. Wi-Fi Radio Characteristics

The following data are from tests conducted at room temperature, with a 3.3V power supply.

Table 5-3. Wi-Fi Radio Characteristics

Parameters	Min	Typical	Max	Unit
Input frequency	2412	-	2484	MHz
Output impedance	-	39+j6	-	Ω
Output power of PA for 72.2 Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1 Mbps	-	-98	-	dBm
CCK, 11 Mbps	-	-91	-	dBm
6 Mbps (1/2 BPSK)	-	-93	-	dBm
54 Mbps (3/4 64-QAM)	-	-75	-	dBm
HT20, MCS7 (65 Mbps, 72.2 Mbps)	-	-72	-	dBm
Adjacent Channel Rejection				
OFDM, 6 Mbps	-	37	-	dB
OFDM, 54 Mbps	-	21	-	dB
HT20, MCS0	-	37	-	dB
HT20, MCS7	-	20	-	dB



6. Package Information

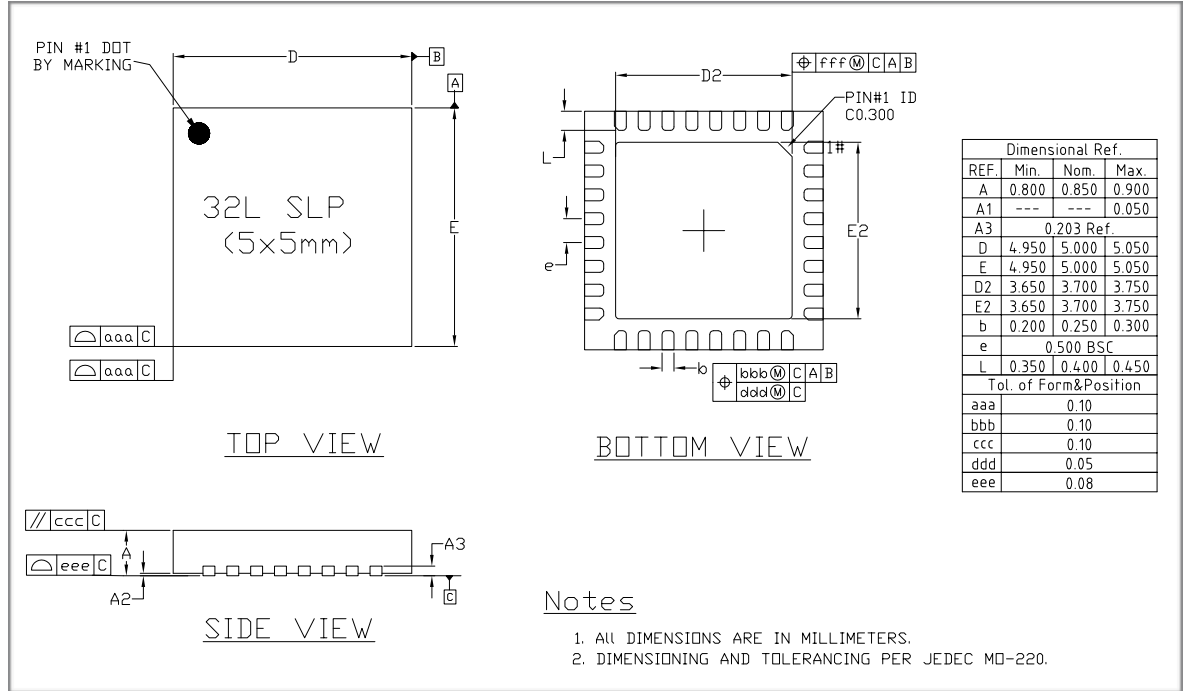


Figure 6-1. ESP8266EX Package



I. Appendix - Pin List

For detailed pin information, please see [ESP8266 Pin List](#).

- Digital Die Pin List
- Buffer Sheet
- Register List
- Strapping List

Notes:

- *INST_NAME* refers to the *IO_MUX REGISTER* defined in **eagle_soc.h**, for example *MTDI_U* refers to *PERIPHS_IO_MUX_MTDI_U*.
- *Net Name* refers to the pin name in schematic.
- *Function* refers to the multifunction of each pin pad.
- *Function number 1 ~ 5* correspond to *FUNCTION 0 ~ 4* in SDK. For example, set *MTDI* to *GPIO12* as follows.
 - `#define FUNC_GPIO12 3 //defined in eagle_soc.h`
 - `PIN_FUNC_SELECT(PERIPHS_IO_MUX_MTDI_U, FUNC_GPIO12)`



II. Appendix - Learning Resources

II.1. Must-Read Documents

- [ESP8266 Quick Start Guide](#)

Description: This document is a quick user guide to getting started with ESP8266. It includes an introduction to the ESP-LAUNCHER, instructions on how to download firmware to the board and run it, how to compile the AT application, as well as the structure and debugging method of RTOS SDK. Basic documentation and other related resources for the ESP8266 are also provided.
- [ESP8266 SDK Getting Started Guide](#)

Description: This document takes ESP-LAUNCHER and ESP-WROOM-02 as examples of how to use the ESP8266 SDK. The contents include preparations before compilation, SDK compilation and firmware download.
- [ESP8266 Pin List](#)

Description: This link directs you to a list containing the type and function of every ESP8266 pin.
- [ESP8266 Hardware Design Guideline](#)

Description: This document provides a technical description of the ESP8266 series of products, including ESP8266EX, ESP-LAUNCHER and ESP-WROOM.
- [ESP8266 Hardware Matching Guide](#)

Description: This document introduces the frequency offset tuning and antenna impedance matching for ESP8266 in order to achieve optimal RF performance.
- [ESP8266 Technical Reference](#)

Description: This document provides an introduction to the interfaces integrated on ESP8266. Functional overview, parameter configuration, function description, application demos and other pieces of information are included.
- [ESP8266 Hardware Resources](#)

Description: This zip package includes manufacturing BOMs, schematics and PCB layouts of ESP8266 boards and modules.
- [FAQ](#)

II.2. Must-Have Resources

- [ESP8266 SDKs](#)



Description: This webpage provides links both to the latest version of the ESP8266 SDK and the older ones.

- [ESP8266 Tools](#)

Description: This webpage provides links to both the ESP8266 flash download tools and the ESP8266 performance evaluation tools.

- [ESP8266 Apps](#)
- [ESP8266 Certification and Test Guide](#)
- [ESP8266 BBS](#)
- [ESP8266 Resources](#)



Espressif IOT Team
www.espressif.com

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member logo is a trademark of the Wi-Fi Alliance. The Bluetooth logo is a registered trademark of Bluetooth SIG.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Copyright © 2018 Espressif Inc. All rights reserved.