UNIVERSIDADE FEDERAL DO PARANÁ

ENGENHARIA ELÉTRICA

CAROLINE CRISTINA BAGATIN ANDRADE

RENATA SAES BUSATO

PROJETO DE DETECTOR DE INTRUSÃO PARA APLICAÇÃO EM CONTÊINERES DE CARGA

CURITIBA

2019

CAROLINE CRISTINA BAGATIN ANDRADE

RENATA SAES BUSATO

PROJETO DE DETECTOR DE INTRUSÃO PARA APLICAÇÃO EM CONTÊINERES DE CARGA

Trabalho de Conclusão de Curso de Engenharia Elétrica, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná.

Orientador: Prof. Ph.D. André Augusto Mariano

CURITIBA

2019

CAROLINE CRISTINA BAGATIN ANDRADE

RENATA SAES BUSATO

PROJETO DE DETECTOR DE INTRUSÃO PARA APLICAÇÃO EM CONTÊINERES DE CARGA

TRABALHO APRESENTADO AO CURSO DE ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DO PARANÁ COMO REQUISITO À OBTENÇÃO DO TÍTULO DE GRADUAÇÃO.

COMISSÃO EXAMINADORA

PROF. PH.D. ANDRÉ AUGUSTO MARIANO

PROF. DR. JOÃO DA SILVA DIAS

MESTRE MARIANE GAVIOLI BERGAMINI

CURITIBA, DEZEMBRO DE 2019.

AGRADECIMENTOS

Gostaríamos de agradecer primeiramente a Deus que sempre nos suportou e nos auxiliou e com sua graça e bondade nos concedeu a oportunidade de estar onde nunca imaginávamos. Deus é bom o tempo todo!

À nossa família e aos que amamos que oraram por nós, nunca mediram esforços para nos auxiliar em tudo que necessitamos e estiveram presentes em cada passo que decidimos dar.

Aos nossos amigos que se tornaram nossa família, estavam presentes nos momentos de descontração em meio às dificuldades. Ao laço que nos tornou irmãs.

Aos nossos gestores, líderes e colegas da empresa Brado Logística que nos guiaram e deram suporte durante o desenvolvimento desse projeto.

Agradecemos nosso professor orientador e todos os professores do DELT por toda a paciência e ensinamentos durante todos esses anos.

“Cada pessoa deve trabalhar para o seu aperfeiçoamento e, ao mesmo tempo, participar da responsabilidade coletiva por toda a humanidade”.

Marie Curie, 1921

Não só isso, mas também nos gloriamos nas tribulações, porque sabemos que a tribulação produz perseverança; a perseverança, um caráter aprovado; e o caráter aprovado, esperança. E a esperança não nos decepciona, porque Deus derramou seu amor em nossos corações, por meio do Espírito Santo que ele nos concedeu.

Romanos 5:3-5 (NVI)

RESUMO

A engenharia elétrica possui múltiplas áreas e aplicações em diversos locais do mundo, sendo utilizada para resolução de problemas de áreas que não são diretamente ligadas à eletricidade e eletrônica. A internet das coisas tem aumentado ainda mais a presença de dispositivos de monitoramento ao longo do século XXI. Nesse trabalho é desenvolvida uma solução para a área logística a partir de um sistema de sensoriamento para a detecção de intrusão em contêineres. Esse projeto tem como objetivo o desenvolvimento de um sistema e um protótipo de identificação de violação de contêineres. O projeto ainda visa alcançar um baixo custo para ser viável, mesmo nos contêineres que são enviados para o exterior sem previsão de retorno ao Brasil. Os resultados mostram que as soluções propostas neste trabalho de hardware, de firmware, bem como do aplicativo foram alcançados, e futuras melhorias já são identificadas. Conclui-se que o sistema e protótipo desenvolvidos têm grande potencial para contribuir na identificação de furtos em contêineres para melhoria no serviço prestado pelas empresas que fazem transporte com contêineres.

Palavras-chave: Contêiner, sensoriamento, microcontrolador.

ABSTRACT

Electrical engineering has multiple areas and applications in various locations around the world and is used to troubleshoot areas that are not directly connected to electricity and electronics. The internet of things has further increased the presence of monitoring devices throughout the 21st century. In this work, a solution for the logistics area is developed as a sensing system to detect container intrusion. This project aims to present a prototype and system for container breach identification. The project still looks to achieve a low cost to be viable, even in containers that are shipped abroad without return forecast to Brazil. The results show that the proposed solutions in the hardware, firmware, and system were achieved, and future improvements are already identified. It is concluded that the developed system and prototype has great potential to contribute in the identification of container thefts and to improve the service provided by the companies that transport containers.

Keywords: Container, sensing, microcontroller.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ISO - *International Organization for Standartization*

TEU – *Twenty-foot Equivalent Unit*

Wi-Fi – *Wireless Fidelity*

ROM – *Ready Only Memory*

RAM – *Random Access Memory*

SRAM – *Static Random Access Memory*

µC - microcontrolador

GPIO – General Purpose Input/Output

GND – *Ground*

PWM – *Pulse Width Modulation*

IP – *Internet Protocol*

KiB – *Kilo Binary Digit*

LDR – *Light Dependent Resistor*

IEEE – Instituto de Engenheiros Elétricistas e Eletrônicos

PAN – Personal Area Networks

USB – Universal Serial Bus

SPIFFS – *SPI Flash File System*

IDE - *Integrated Development Environment*

SUMÁRIO

1. INTRODUÇÃO

Na década de 30, um jovem americano chamado Malcom Mc Lean que havia acabado de se formar, ao observar o transporte lento dos fardos de algodão diretamente no porto de Nova Iorque, teve a ideia de transportá-los em grandes caixas de aço, que seriam então transportadas pelos navios. Em 1986, os padrões ISO mundiais foram oficializados. Em se tratando de transporte marítimo, esses padrões normatizaram os procedimentos que deveriam ser utilizados pelos países que realizavam o comércio internacional. (CORREA; Rosana, 2018)

Em 2012, já havia mais de 20 milhões de contêineres no mundo. E algo entre cinco e seis milhões estavam sendo transportados ao redor do mundo em navios, caminhões e trens. No total eles realizam cerca de 200 milhões de viagens por ano. (BILLING; Jane, 2012).

A empresa para qual o estudo foi desenvolvido realiza transporte multimodal nacional, ou seja, utiliza o modal ferroviário e rodoviário. O processo é personalizado, com foco no cliente, para que o modelo desenhado seja mais vantajoso do que o modal mais tradicional brasileiro, o rodoviário. Além de ser vantajoso financeiramente, a utilização da ferrovia para transporte de contêineres reduz em até 93% a emissão de gás carbônico a cada 1000km rodados em comparação com o transporte rodoviário, transformando a execução de forma muito sustentável. (BRADO, 2019).

A unidade padrão mundial utilizada para calcular volume de contêineres é TEU *(*do inglês *Twenty-foot Equivalent Unit)*, um TEU é equivalente ao volume de um contêiner de 20 pés. De acordo com o artigo de publicidade legal publicado pela empresa Brado Logística no ano de 2018 foram movimentados 268 mil TEU’s, o que em número de contêineres de 40 pés (tamanho mais usual de contêineres), representa 134 mil contêineres transportados. Comparando com o ano de 2017 houve um aumento de 20%. A previsão para 2019 é um aumento de 20% frente ao ano de 2018, ocasionando uma movimentação de 320 mil TEU’s, que representa 160 mil contêineres de 40 pés.

1.1. PROBLEMA E MOTIVAÇÃO

Durante o processo de transporte contêineres em ferrovia há grande incidência de roubos ou tentativa de roubos de carga. Os trens possuem em média 50 vagões, que carregam dois contêineres cada, ou seja, cada composição tem, em geral, 100 contêineres enfileirados, de forma que o maquinista não consegue ter total visão da composição. Em geral as violações são reportadas às empresas quando o trem chega ao terminal sem nenhuma informação sobre onde o evento aconteceu, gerando perda parcial ou total do produto com prejuízo a transportadora multimodal, além de falta de informação para melhoria da segurança nos trechos com maior incidência de violações. Dentro dos modais brasileiros foram identificados muitos furtos em contêineres e dificuldade em identificar a localização do ocorrido para se obter a melhoria do serviço prestado pelas empresas que fazem essa movimentação. Este projeto visa à criação de um hardware de baixo custo que identificará uma tentativa ou furto de carga no contêiner e será capaz de armazenar as informações que serão coletadas no próximo terminal que o contêiner for verificado.

1.2. OBJETIVOS

1.2.1. OBJETIVO GERAL

Este trabalho tem foco no desenvolvimento de um sistema e um protótipo de identificação de violação de contêineres.

1.2.2. OBJETIVOS ESPECÍFICOS

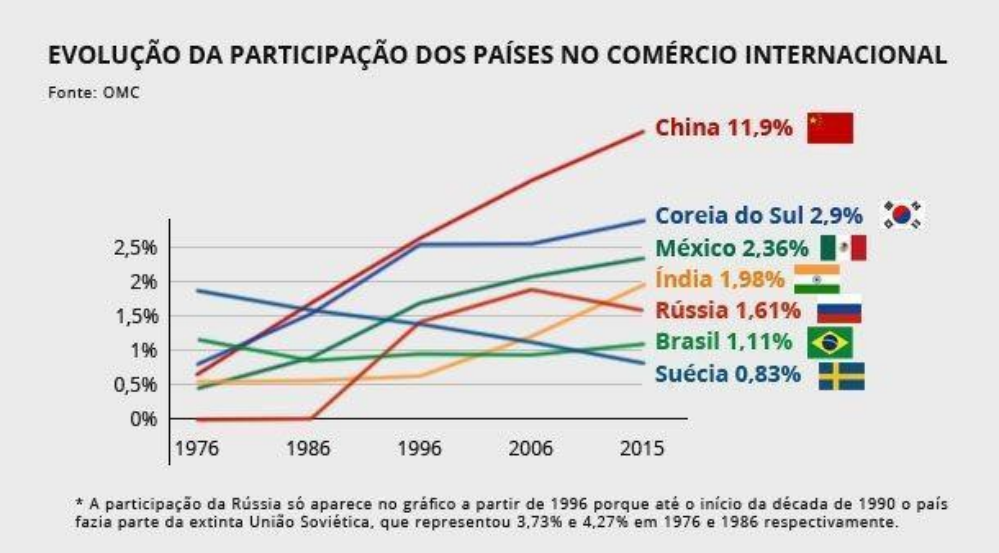
- Desenvolver um hardware que seja capaz de identificar a ocorrência de um evento dentro do contentor de carga;
- Desenvolver um software que seja capaz de armazenar as informações identificadas de forma inteligente;
- Desenvolver um aplicativo que seja capaz de gerar uma informação que possa ser analisada pelo operacional da empresa.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. O MODAL DE TRANSPORTE POR CONTÊINERES

Após a padronização dos tamanhos de contêineres, na década de 80, a participação dos países no comércio mundial aumentou de forma significativa, conforme é possível observar na FIGURA 1. Aumento esse que possibilitou o transporte de novos tipos de cargas e demandou novos tipos de contêiner. (CORREA; Rosana, 2018).

FIGURA 1 - EVOLUÇÃO DA PARTICIPAÇÃO DOS PAÍSES NO COMÉRCIO INTERNACIONAL



FONTE: CBN, 2016

Os contêineres tradicionais com portas ao fundo podem ser do tipo *dry*, isolante ou refrigerado. Os do tipo *dry* podem ser ventilados, ou não, e servem para cargas secas, como: alimentos, roupas, móveis e produtos químicos. Os contêineres do tipo isolante servem para cargas que não podem ser expostas a mudanças rápidas ou bruscas de temperatura. Os refrigerados, também conhecidos como *reefer* são isolantes e equipados com um sistema de refrigeração embutido, são utilizados primariamente para alimentos ou outros artigos que requerem temperatura controlada de meio ambiente, quando em funcionamento normal costumam perder em média 1ºC de temperatura por dia. Os contêineres *tank* (i*sotank*) são utilizados

no transporte de líquidos e gases. Eles exigem um grande aparato de segurança, porque na maioria das vezes a sua carga tem um alto grau de periculosidade. E, por fim, o contêiner de tipo *bulk*, é um contêiner graneleiro, ele tem uma pequena porta na parte de baixo da parede frontal, que tem como objetivo o despejo da carga, facilitando a desova do contêiner. (ICONTAINERS, 2019).

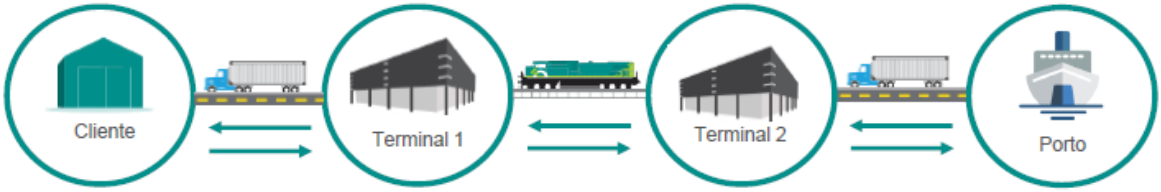
FIGURA 2 - TIPOS DE CONTÊINERES

FONTE: https://yata.ostr.locaweb.com.br/

O processo da operação de exportação funciona da seguinte maneira: primeiramente o contêiner vazio, de posse de algum armador, é retirado do *depot* próximo ao porto via modal rodoviário, esse contêiner é carregado no trem e segue até o terminal da empresa que é mais próximo a planta do cliente. Após a chegada, esse contêiner é descarregado do trem e é estufado com a carga, ou no pátio da empresa ou na planta do cliente que é dono do produto. Após o contêiner carregado, ele é faturado, o que é chamado de *gate* e é novamente carregado no trem para que siga em direção a um terminal próximo ao porto.

Para que ocorra a importação, os contêineres que chegam ao porto, de posse de algum armador, são levados até a próxima parada do trem através do modal rodoviário, para que seja carregado no trem e transportado até ao terminal da empresa que seja mais próximo a planta do cliente, para que seja encaminhado para ao destino final por meio de um caminhão.

FIGURA 3 - MOVIMENTAÇÃO DE CONTÊINERES DE EXPORTAÇÃO E IMPORTAÇÃO



FONTE: Autoras, 2019

Por fim, o transporte na operação do mercado interno é focado na movimentação de cargas dentro do país. Acontece entre dois terminais ferroviários no corredor norte, os contêineres dessa modalidade são próprios da empresa fornecedora do serviço. A partir de um contêiner vazio no terminal A, o processo é iniciado na estufagem dentro da empresa ou na planta do cliente. Em seguida, ele é carregado no trem e segue viagem até o terminal B onde é desovado e aguarda, vazio, o início do próximo ciclo do que será do terminal B em direção ao terminal A.

FIGURA 4 - MOVIMENTAÇÃO DE CONTÊINERES DE MERCADO INTERNO

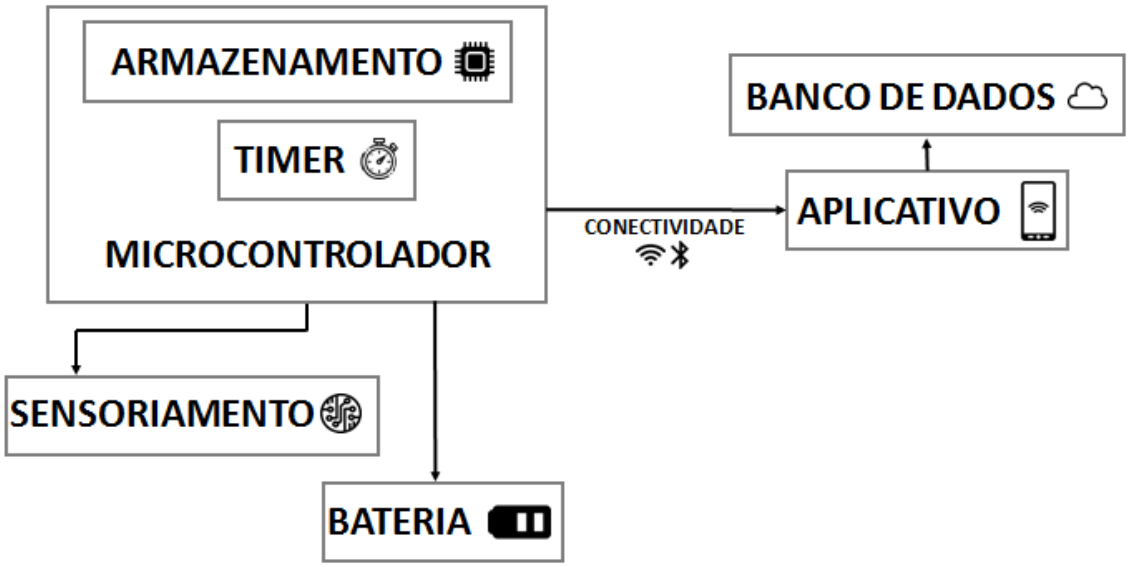
FONTE: Autoras, 2019

Nos próximos anos novos corredores ferroviários receberão o transporte de contêineres, focados no mercado interno, aumentando o volume de maneira significativa. Dessa maneira o foco atual da empresa é a boa manutenção e melhores controles dos ativos, bem como a identificação dos pontos falhos de segurança na malha ferroviária.

2.2. ESTRUTURA DO PROJETO

O projeto foi desenhado em formato de diagrama de blocos, mostrado na FIGURA 5, para estruturar o trabalho que será desenvolvido. O centro do projeto será o microcontrolador que irá controlar os elementos periféricos e terá armazenamento suficiente para todos os dados que servirão para identificação da intrusão dos contêiners. A identificação será realizada pelo sensoriamento e atrelada a uma data e hora do relógio interno do µC. Haverá uma bateria que será capaz de manter o projeto autônomo durante o tempo de operação dos contêineres. Por meio de um sistema de comunicação sem fio o microcontrolador transmitirá os dados armazenados para um smartphone que estará portando um aplicativo e enviará os dados para um banco de dados para análise do corporativo da empresa.

FIGURA 5 - DIAGRAMA DE BLOCOS DO PROJETO



FONTE: Autoras, 2019

2.3. MICROCONTROLADOR

O microcontrolador é o sistema central do projeto, considerado o cérebro do sistema de hardware. Tem capacidade de armazenar o código de programação para controlar os sistemas periféricos, com portas analógicas, digitais, conversores analógico-digital, memória interna e conectividade. (AURELIANO; Andre, 2017)

O ESP32 é um microcontrolador produzido pela Espressif Systems, que possui dois núcleos, ou seja, é dual-core e tem uma frequência de clock de até 240MHz. Tem um módulo Wi-Fi 802.11n de 2.4GHz até 150 Mbit/seg e Bluetooth v4.2 BR/EDR e Bluetooth low energy integrados. Possui memória interna ROM de 448 KiB que é utilizada para boot e funções principais do hardware, SRAM de 520 KiB, utilizada para os programas com dados e instruções, RTC slow SRAM de 8 KiB para acessar o co-processador no modo deep sleep, RTC fast SRAM de de 8 KiB para armazenamento de dados e utilização de relógio de tempo-real no modo *deep-sleep*, e eFuse de 1Kbit, onde 256 bits são utilizados pelo sistema (endereço IP e configurações do chip) e 768 bits são para aplicações de criptografia da Flash e Chip-ID. Com relação as portas, o ESP32 possui 36 GPIOs, sendo 16 canais PWM de 12 bits. Opera com tensão entre -0.3V e 3.6V e temperatura entre -40ºC e 150ºC. (ESP32 WROOM Datasheet, 2019).

O microcontrolador Arduíno Mega também possui características interessantes para o tipo de processamento necessário e módulo pronto para o incremento atingir o resultado proposto do protótipo, porém a solução como todo seria custosa por conta dos módulos de conectividade e armazenamento necessários. (SOUZA; Fábio, 2014).

Além dos dispositivos do Arduino, também se levou em consideração o microcontrolador ultra *lowpower* MSP430, porém, da mesma forma não atende todos os requisitos necessários, o que resultaria em um aumento do escopo do projeto de conclusão de curso proposto. (BERNARDO; Jader, 2015).

Considerando o baixo custo do microcontrolador ESP32, o módulo Wi-Fi e Bluetooth já incluídos, o sistema RTC para relógio em *deep-sleep* e a faixa de temperatura em que ele é capaz de operar, ele foi escolhido como o microcontrolador ideal para o projeto proposto.

2.4. SENSORIAMENTO

Sensor é um dispositivo capaz de identificar um estímulo do ambiente e convertê-lo a um sinal elétrico compatível com os sistemas que estão acoplados a ele. Para o projeto, foram escolhidos dois sensores, para que exista uma redundância de detecção de variação do sistema interno do contêiner, sendo eles: o

LDR, para avaliar a variação de luminosidade internamente e o LM35 para avaliar a variação de temperatura interna do contêiner. (SILVEIRA; Cristiano Bertulucci, 2018).

O LDR é um sensor que funciona como uma resistência que tem seu valor alterado conforme a variação de luminosidade. Opera na tensão máxima de 150V, tem uma potência máxima de 100 mW e funciona na faixa de temperatura de -60ºC até aproximadamente 75ºC. (RS Datasheet LDR, 1997).

O LM35 é um sensor que apresenta na saída, uma tensão proporcional a temperatura do ambiente em que ele se encontra, com um sinal de 10mV para cada grau Celsius. Pode ser alimentado com uma tensão entre 4V e 30V e possui temperatura de trabalho de -55ºC até 150ºC. (Texas Instruments Datasheet LM35, 1999, revisado em 2015).

Outros sensores foram avaliados, como os sensores de microondas, que funcionam emitindo ondas para o ambiente e validam se houve alguma mudança na onda emitida e os sensores acústicos que emitem ondas sonoras em uma frequência não audível por humanos e detectam alterações nas ondas emitidas. Esses sensores foram descartados porque os contêineres são bruscamente movimentados durante o trajeto na ferrovia, balançados de forma irregular o que também causa muito barulho. A grande movimentação das cargas e o intenso ruído sonoro durante o transporte ferroviário convencional contribuíram para a exclusão da possibilidade de utilizar estes outros dois formatos de sensoriamento. (AUSEC; Redação, 2016).

2.5. COMUNICAÇÃO

A comunicação de um dispositivo é o meio pelo qual as informações são transmitidas a outras plataformas. A transmissão de informação precisa de um meio, e isto a define como uma transmissão com ou sem fio. A comunicação e suas características são padronizadas de acordo com a tecnologia e protocolos utilizados, dependendo da largura de banda, taxa de transmissão, frequências, entre outros. (CORRÊA; Underléa *et al*., 2006).

Nas comunicações com fio, o meio da comunicação é físico, ou seja, utiliza fios. Em geral, este tipo de comunicação é ponto - ponto, bidirecional e segura. Já

nas comunicações sem fio, o meio da comunicação é o ar. Geralmente, ponto - multiponto e largamente utilizada. Pode ser via Wireless, Bluetooth, Infravermelho, etc. Rede Wireless utiliza transmissão por rádio frequência e são regidas pelos padrões IEEE e são categorizadas como redes de larga distância. A rede Bluetooth, utilizada como rede pessoal (Personal Area Networks - PANs), utiliza rádio frequência com ondas de pequeno comprimento, ou seja, de altas frequências ocasionando um raio de cobertura pequeno. (CORRÊA; Underléa *et al*., 2006).

O microcotrolador ESP32, escolhido para o projeto possui as duas modalidades de comunicação, via cabo micro USB, módulo WI-FI e Bluetooth. Como o contêiner não fica parado e o operador da empresa pode não ter acesso manual ao local que será colocado o projeto na parte interna do contêiner, uma solução cabeada não seria a ideal, por isso, a conectividade sem fio foi a escolhida.

A comunicação Wi-Fi possibilita a criação de WebServices para que dados estejam na rede acessada pelos usuários. WebService é uma integração entre sistemas que estão conectados a uma mesma rede e possibilita enviar e receber dados. Podem ser desenvolvidas em diversas plataformas e linguagens diferentes para que os desenvolvedores possam escolher de acordo com a aplicação de cada projeto. (SOAWebServices, 2012)

2.6. ARMAZENAMENTO

Os três tipos de memória mais comuns em chips de armazenamento são a memória RAM, a ROM e a Flash. A memória RAM (*Random Access Memory*), memória de acesso aleatório, não armazena dados de forma permanente, ela funciona para leitura de dados quando requeridos, não necessariamente de forma seqüencial, ela armazena os dados que estão sendo utilizados no momento. Quanto maior a memória RAM, mais atividades pode se executar ao mesmo tempo no dispositivo. A memória SRAM (*Static Random Access Memory*) é um tipo de memória RAM, é uma memória de acesso aleatório estática, ou seja, ela consegue armazenar os dados, mesmo sem atualização contínua. A memória ROM (*Ready Only Memory*), memória para apenas leitura, serve para guardar dados que sempre devem estar armazenados, como as configurações de sistema, e dados que precisem ser apenas executados sem serem alterados. A memória FLASH, pode

receber e armazenar dados mesmo se estiver desligada e o armazenamento dos dados é feito em células. Para que os dados da memória FLASH sejam apagados, não é necessário que novos dados sejam reescritos no lugar dos dados antigos, o que diferencia ela de outros tipos de memória e a torna mais rápida na atualização do que as outras. (RODRIGUES; Gabriela, 2014).

Devido à capacidade de apagar os dados em blocos inteiros, a memória FLASH é perfeita para aplicações como cartões de memória e *pendrives*, função esta parecida com a objetivada nesse projeto, portanto, será utilizada nesse projeto.

3. MATERIAS E MÉTODOS

O desenvolvimento desse projeto iniciou com a identificação do problema e da solução técnica mais adequada. Foi efetuada uma pesquisa bibliográfica e uma pesquisa de mercado para saber quais produtos já existiam com funcionalidades para resolução do problema identificado. Com o problema identificado e a solução determinada, foi construído um diagrama de blocos, como ilustrado na FIGURA 5 com todos os elementos do circuito proposto. Na programação do microcontrolador estará o armazenamento dos dados de quando o contêiner for violado, juntamente com o timer que dirá a data e o horário em que o fenômeno aconteceu. O sensoriamento servirá para identificar quando houver variação no sistema normal interno do contêiner. A bateria alimentará todo o sistema para que seja autônomo por pelo menos 10 dias. Com a conectividade com o celular, através de um aplicativo os dados armazenados sobre a data da intrusão serão baixados e armazenados em um banco de dados para futuras análises preventivas da empresa possuidora dos contêineres. E por fim, foram determinados todos os componentes que seriam necessários para a construção da solução.

O desenvolvimento do projeto foi dividido em três grandes blocos, o desenvolvimento de hardware, o desenvolvimento de firmware e o desenvolvimento do aplicativo.

3.1. HARDWARE

Inicialmente foi feito um estudo do circuito que seria necessário para integração dos blocos funcionais de bateria, sensoriamento e microcontrolador, que compõem o hardware do projeto.

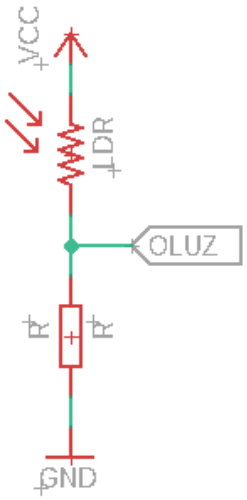
A bateria aplicada ao projeto foi dimensionada a partir do circuito desenvolvido a fim de que o protótipo seja autossuficiente durante o trajeto, que em média dura 240 horas. A corrente que percorre o circuito foi medida e então a bateria foi definida. Juntamente com isso foi necessária a inserção de um circuito de regulação de tensão para que a alimentação do microcontrolador e dos sensores fosse de 5 V.

O sensoriamento será feito a partir de dois sensores, para que exista uma redundância entre as informações coletadas e a intrusão seja identificada por

qualquer uma das grandezas identificadas. Considerando todas as situações normais dos contêineres, ou seja, se encontram em situações barulhentas, são bruscamente movimentados, os sensores de luminosidade e de temperatura foram considerados os mais indicados porque não possuem aberturas suficientes para tornar o ambiente interno iluminado e nem um ambiente com temperatura igual ao do meio que se encontra, de forma que quando o contentor de carga for violado, a luminosidade invadirá o contêiner, provocando um delta suficiente para que o sensor verifique, e a temperatura irá mudar, o contêiner *dry* irá resfriar e o contêiner *reefer* se tornará cada vez mais quente.

O sensor de luminosidade, LDR, funciona como um resistor e por isso foi inserido a um divisor de tensão, como mostrado na FIGURA 6, para que a variação de luminosidade acarretasse em uma variação de tensão e posteriormente será codificada pelo microcontrolador.

FIGURA 6 - DIVISOR DE TENSÃO DO LDR



FONTE: Autoras, 2019

O sensor de temperatura, LM35, foi ligado à fonte de energia e diretamente no microcontrolador, pois o dispositivo já realiza uma variação de tensão de acordo com a temperatura ambiente.

O microcontrolador escolhido, ESP32, possui internamente um cristal que funciona como um relógio de tempo real, memória para o armazenamento de dados e tem integrado o módulo de Wi-Fi e o modulo de Bluetooth.

O armazenamento disponível no ESP32 para arquivos externos está presente no *SPI FLASH FILE SYSTEM* (SPIFFS), que contempla 16MB de memória. Para que a solução seja eficaz é preciso que o dispositivo tenha um armazenamento suficiente para captar as medições ao longo da vida útil do protótipo. A mensagem armazenada possui data, hora, índices de luminosidade, identificação de possibilidade de intrusão e a temperatura. A mensagem será da seguinte forma:

DATA-HORA; IDENTIFICAÇÃO DE INTRUSÃO; LUMINOSIDADE; TEMPERATURA

Verificando o tamanho em bytes do que o vetor proposto ocuparia identificou-se que o mesmo tem 77 byte, ou 7.7 e-5 MB, conforme ilustrado na FIGURA 8, representação da porta serial da função que foi utilizada para medir o tamanho do vetor, testFileIO(), conforme mostrado na FIGURA 7.

FIGURA 7 - FUNÇÃO testFileIO

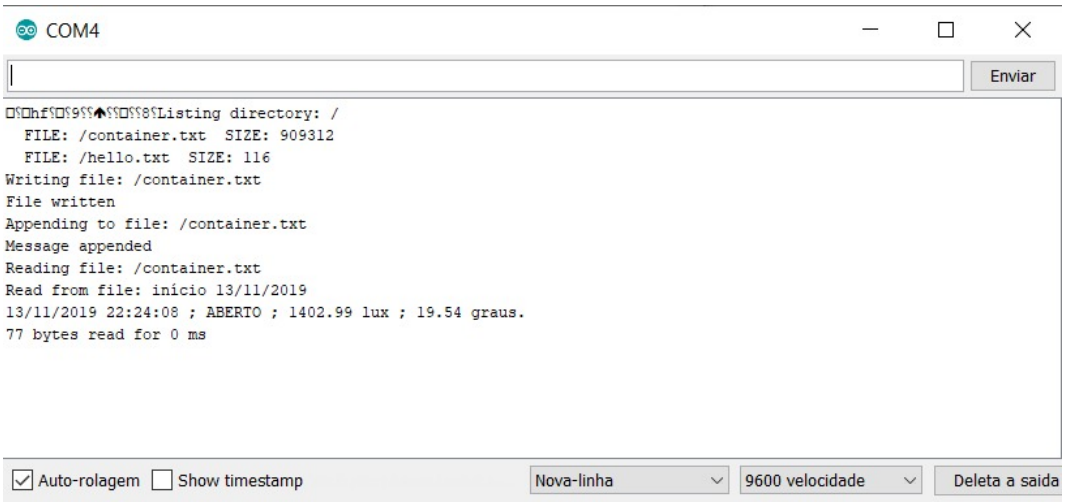
```
void testFileIO(fs::FS &fs, const char * path){
    File file = fs.open(path);
    static uint8_t buf[512];
    size_t len = 0;
    uint32_t start = millis();
    uint32_t end = start;
    if(file && !file.isDirectory()){
        len = file.size();
        size_t flen = len;
        start = millis();
        while(len){
            size_t toRead = len;
            if(toRead > 512){
                toRead = 512;
            }
            file.read(buf, toRead);
            len -= toRead;
        }
        end = millis() - start;
        Serial.printf("%u bytes read for %u ms\n", flen, end);
        file.close();
    } else {
        Serial.println("Failed to open file for reading");
    }


    file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }

    size_t i;
    start = millis();
    for(i=0; i<2048; i++){
        file.write(buf, 512);
    }
    end = millis() - start;
    Serial.printf("%u bytes written for %u ms\n", 2048 * 512, end);
    file.close();
}
```

FONTE: SUHANKO; Djames, 2019.

FIGURA 8 - PORTA SERIAL COM TAMANHO DO VETOR ARMAZENADO



FONTE: Autoras, 2019

Portanto, considerando os 77 bytes por mensagem, poderíamos obter aproximadamente 207.792 registros. Neste caso não é preciso buscar outros métodos de armazenamento.

Como já citado anteriormente, o ESP32 possui um RTC interno, ou seja, ele possui um relógio de tempo real (*real time clock*, em inglês) que é capaz de manter a data atualizada enquanto conectado a bateria. Para configurar a hora do relógio, foi utilizado o *Unix Time,* um conversor de data para sistemas de programação. (*Epoch Unix Time Stamp Converter*).

O módulo de Wi-Fi interno do ESP32 foi o escolhido para fazer a comunicação com o aplicativo desenvolvido para celular. Então, o ESP32 criará uma rede wireless e funcionará como um ponto de acesso, onde o celular que possuir o aplicativo será conectado e extrairá as informações a partir dos dados que foram armazenados no microcontrolador.

Uma placa de circuito impresso (PCB) foi construída a partir do layout do circuito que foi impresso em papel couchê para que, com uma prensa hidráulica, a tinta fosse transpassada para uma placa de fenolite, o papel remanescente foi retirado e pôr fim a placa foi imersa em percloreto para que a trilhas do circuito fossem corroídas, os furos dos componentes foram feitos com uma furadeira de bancada e os componentes foram soldados com auxílio de um ferro de solda e estanho. Este método de construção da placa de circuito impresso foi escolhido, pois

há disponível a prensa hidráulica no laboratório de eletrônica do curso de engenharia elétrica da Universidade Federal do Paraná.

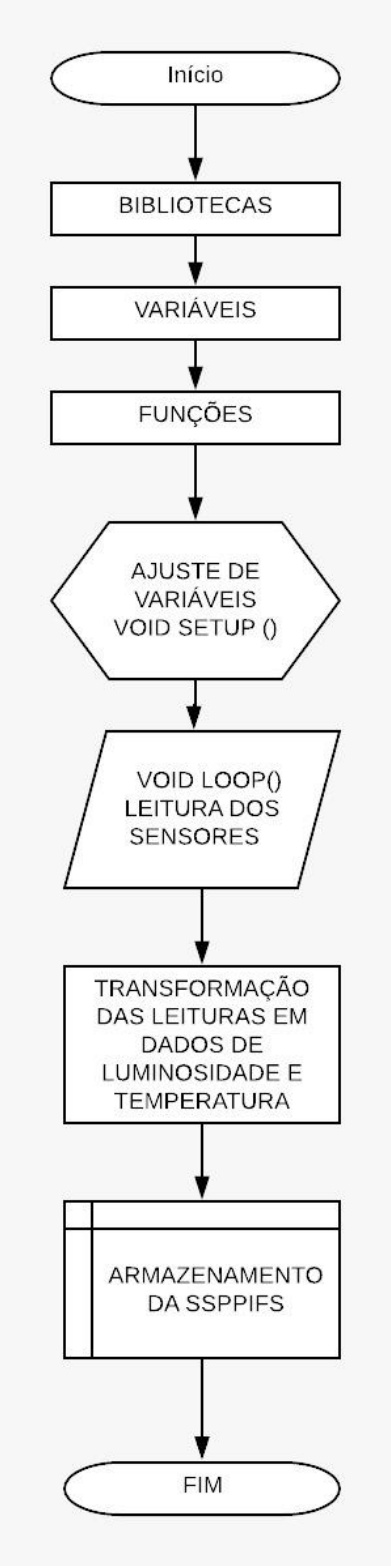
3.2. FIRMWARE

O firmware do projeto compõe a programação do microcontrolador.

O microcontrolador precisa ser programado para ser capaz de receber os sinais dos sensores e transformá-los em informação útil que será armazenado para posterior acesso via comunicação sem fio. O armazenamento e o timer também foram desenvolvidos dentro das linhas de comando. Dentro da plataforma IDE do Arduino o ESP32 foi programado desde a declaração de bibliotecas necessárias, declaração de variáveis, recebimento de dados dos sensores, gravação na memória, geração de informação de data até a conectividade com o aplicativo.

O desenvolvimento do código do firmware foi conforme o diagrama de blocos da FIGURA 9, onde inicialmente foram inseridas as bibliotecas, foram declaradas as variáveis e definidas as funções. Dentro do *void setup()*, as variáveis e portas foram configuradas e, por fim, no *void loop()* foram colocadas as rotinas de leitura e armazenamento.

FIGURA 9 - DIAGRAMA DE BLOCOS DO FIRMWARE



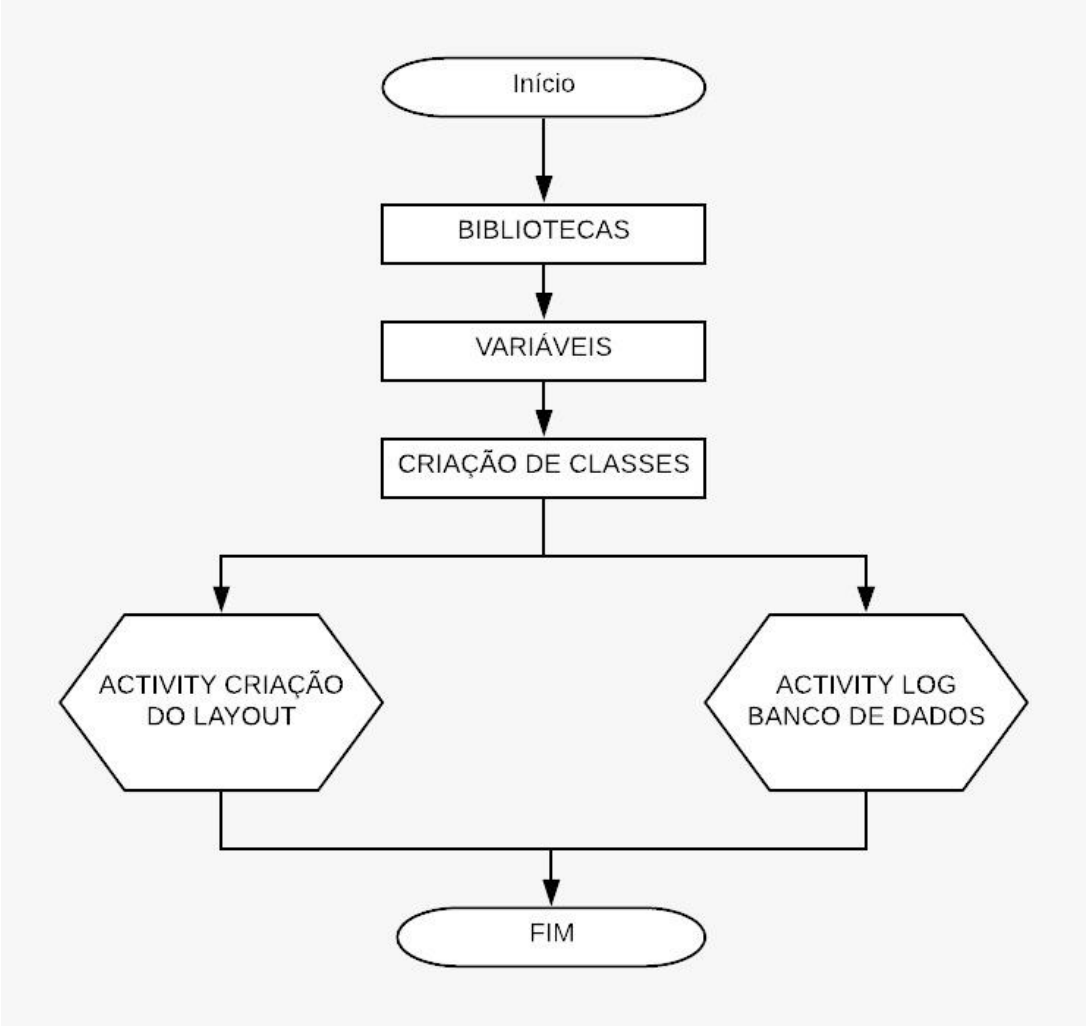
FONTE: Autoras, 2019

3.3. APLICATIVO

O aplicativo focou-se em criar um ambiente agradável ao usuário e que se conectando ao hardware gera informação com base nos dados armazenados e envia para a rede. Dessa forma foi desenvolvida uma programação em Java desde a declaração de variáveis, criação de layout, conectividade com o hardware, geração de informação e envio das informações para a rede.

O desenvolvimento se deu conforme o diagrama de blocos da FIGURA 10, que inicialmente foram incluídas as bibliotecas, definidas as variáveis e criadas as classes, para que depois fosse feito o *input* no banco de dados.

FIGURA 10 - DIAGRAMA DE BLOCOS DO APLICATIVO



FONTE: Autoras, 2019

3.4. MATERIAIS

Para o desenvolvimento do protótipo, foram utilizados os seguintes componentes:

- 1 microcontrolador ESP32 WROOM
- 2 capacitores de 100nF;
- 2 capacitores de 10µF;
- 1 regulador de tensão LM7805;
- 1 conector DC P4 fêmea;
- Cabo Adaptador Bateria 9v
- 2 resistores de 330Ω;
- 1 resistor de 220Ω;
- 1 resistor de 1kΩ;
- 1 sensor LDR;
- 1 capacitor de 2.2µF;
- 1 sensor LM35.

A calibração do sensor LDR foi feita com a utilização de um luxímetro e a análise desses dados foi feita com o software Origin. Para o desenvolvimento do firmware foi utilizado o sofware Arduino, com a linguagem de programação própria do Arduino. O esquemático do circuito e o layout da placa foram desenvolvidos com o software Eagle. O aplicativo foi feito com o software Android Studio, com a linguagem de programação Java.

4. DESENVOLVIMENTO DO PROTÓTIPO

4.1. HARDWARE

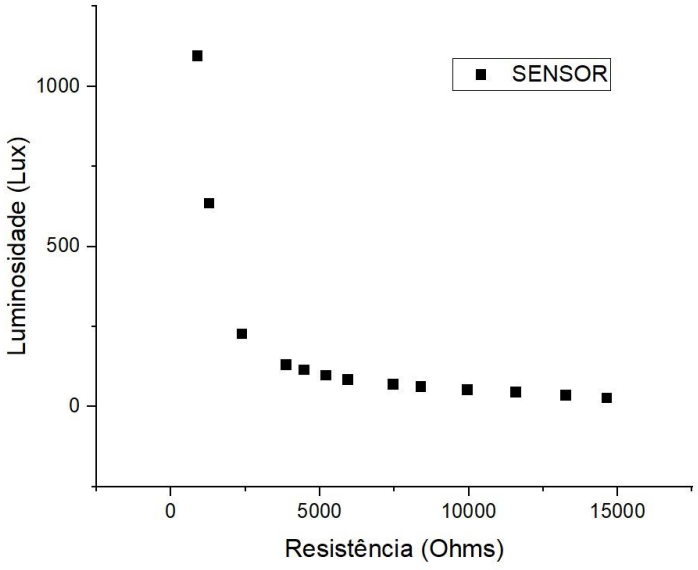
O primeiro sensor, LDR, conforme comentado anteriormente, é um resistor que varia com a luminosidade. Foi necessária a calibração do sensor, variando a luminosidade e medindo com um multímetro a resistência resultante e com um luxímetro a luminosidade aplicada em Lux. Obteve-se a tabela TABELA 1 na caracterização do LDR e em seguida foi utilizado o software ORIGIN para a plotagem dos dados e geração da curva.

TABELA 1 - CARACTERIZAÇÃO SENSOR LDR

Luminosidade (Lux)	26.7	35.1	46	53.6	62.8	69.8	84.6	97.8	115.1	131.6	227	635	1095
Resistência (kΩ)	14.63	13.25	11.57	9.94	8.39	7.45	5.94	5.20	4.46	3.86	2.37	1.28	0.89

FONTE: Autoras, 2019

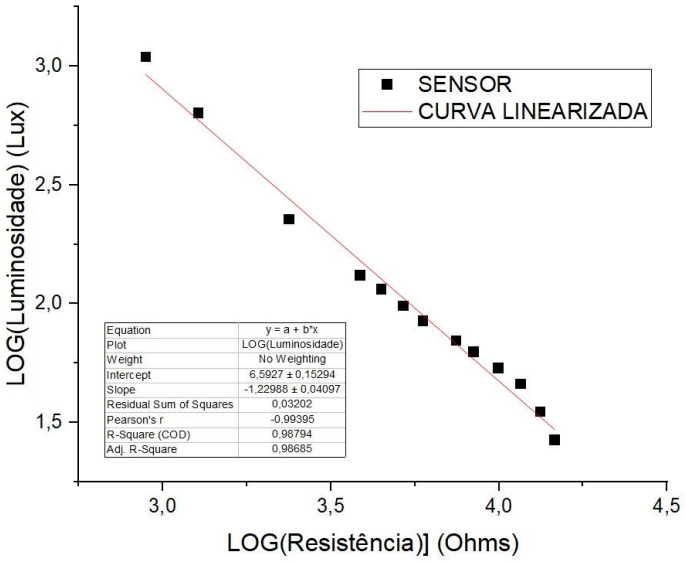
FIGURA 11 - CARACTEIZAÇÃO DO SENSOR LDR



FONTE: Autoras, 2019

Com o objetivo de obter a equação de calibração, foi necessário realizar uma linearização da FIGURA 11. Para isso, foi utilizada a linearização exponencial, que dentre as linearizações realizadas se mostrou a mais adequada para o sistema, que resultou na equação de calibração do sensor:

FIGURA 12 - CALIBRAÇÃO DO SENSOR LDR LINEARIZADO



FONTE: Autoras, 2019

A partir da linearização da curva os parâmetros da equação ax+b foram apresentados e substituídos a fim de obter a equação 1. Para a linearização foi preciso aplicar o log da luminosidade e da resistência, que pode ser observado na FIGURA 12.

$$log_{10}(I) = -1.22988 \cdot log_{10}(R_{LDR}) + 6.5927 \quad (1)$$

$$log_{10}(R_{LDR}) = \frac{log_{10}(I)}{-1.22988} - \frac{6.5927}{-1.22988} \quad (2)$$

$$log_{10}(R_{LDR}) = log_{10}\left(I^{\frac{1}{-1.22988}}\right) + 5.36044 \quad (3)$$

$$R_{LDR} = I^{\frac{1}{-1.22988}} * 10^{5.36044} \quad (4)$$

$$R_{LDR} = I^{-0.813087} * 10^{5.36044} \quad (5)$$

Para a implementação do circuito do sensor de luminosidade, foi utilizado um divisor de tensão para que a partir da variação de luminosidade houvesse uma variação de tensão. Para que a tensão de saída seja metade da entrada de 5V é preciso que os valores das resistências sejam iguais.

Em laboratório, foram realizados testes empíricos e identificou-se que a luminosidade média que caracteriza uma intrusão é de 380 lux. Esses testes aconteceram de forma que o protótipo foi colocado sob diversas luminosidades dentro de uma caixa, para simular o contentor de carga. Portanto, na equação (6) foi substituída a variável I por 380 e resultou no valor da resistência que foi aplicado ao divisor de tensão para que tensões médias já identificassem uma violação nos contêineres.

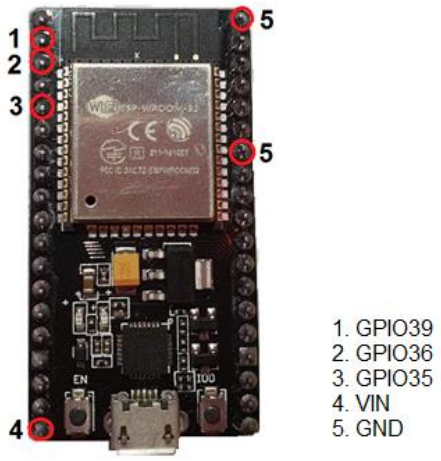
$$R_{LDR} = 380^{-0.813087} * 10^{5.36044} = 1830.69\Omega \qquad (6)$$

Na construção real do circuito, foram utilizados dois resistores de 330Ω, um resistor de 220Ω e um resistor de 1kΩ de forma a obter 1880Ω, valor real de resistência mais próximo ao valor calculado na equação (6) de 1830.69Ω.

O sensor de temperatura LM35 foi adicionado ao sistema para promover uma redundância de dados, de maneira que a intrusão do contêiner seja identificada mesmo que haja falha no sensor de luminosidade. Ele foi colocado no circuito com o seu pino de saída diretamente conectado a GPIO36 (vide FIGURA 13), pois a sua saída indica a temperatura em V. O resultante da medição é apresentado em graus Celsius.

O microcontrolador escolhido, ESP32, possui um botão de *boot*, que precisa ser pressionado para que o código de programação seja carregado. Para que esse botão não precisasse mais ser pressionado, foi adicionado ao sistema um capacitor de 2.2µF entre a GPIO39, que é o pino de *ENABLE*, e o GND. Além disso, na entrada do circuito foi colocado um regulador de tensão 7805, com dois capacitores de 100nF e dois capacitores de 10µF para transformar os 9V da bateria para os 5V utilizados no circuito. O ESP32, com as portas que foram utilizadas, está ilustrado na FIGURA 13.

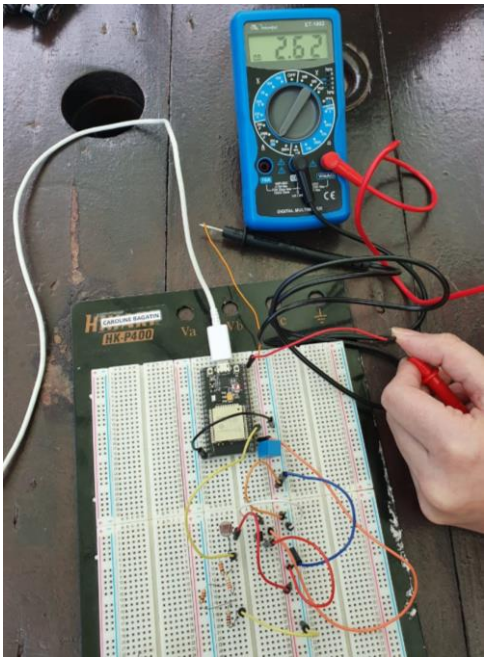
FIGURA 13 - ESP32 E PORTAS UTILIZADAS



FONTE: Autoras, 2019

Para o funcionamento de forma autônoma, foi medida corrente do circuito completo para o dimensionamento da bateria. A corrente foi de 2.62 mA e para que tenhamos o mínimo de 240h de autonomia precisamos de uma bateria de 628.8 mAh, calculo mostrado nas equações (7) e (8).

FIGURA 14 - MEDIÇÃO DA CORRENTE DO CIRCUITO



FONTE: Autoras, 2019

$$Capacidade\ (mAh) = I_{total}(mA) * horas \quad (7)$$

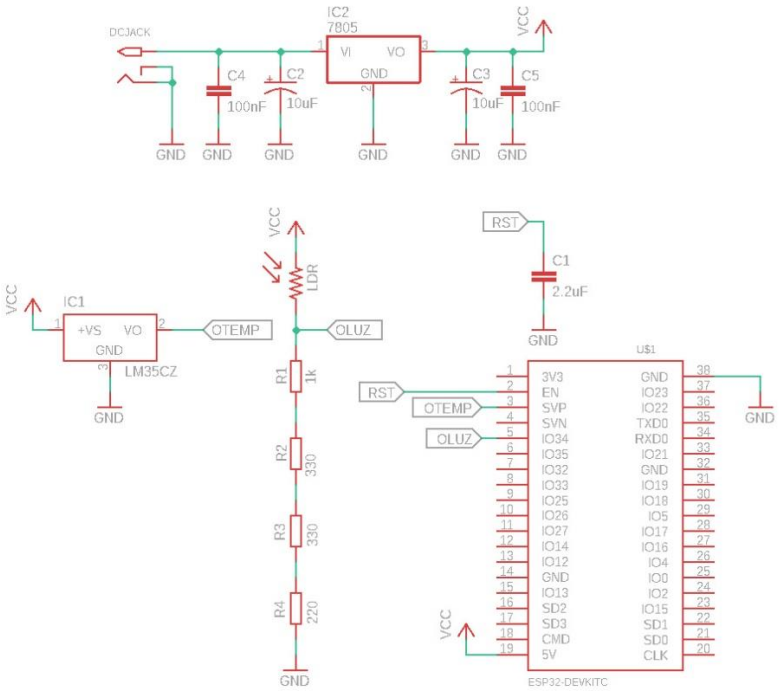
$$Capacidade\ (mAh) = 2.62(mA) * 240 \quad (8)$$

$$Capacidade\ (mAh) = 628.8\ mAh \quad (9)$$

Uma bateria de 9V foi escolhida para atender os requisitos do projeto, porque o circuito foi projetado para uma alimentação de 5V e a bateria de tensão mais próxima é a de 9V, além da necessidade de 10 dias de autonomia do circuito, que requer uma bateria com uma grande capacidade de mAh. Ela foi colocada com o regulador de tensão na entrada do circuito. O regulador de tensão 7805, com dois capacitores de 100nF e dois capacitores de 10µF foram utilizados para transformar os 9V da bateria em 5V para que tanto o microcontrolador quanto os sensores fossem alimentados com a tensão necessária. O protótipo possui uma autonomia de 221.3 h, ou seja, 9.2 dias realizando medições a cada 2 segundos e gravando os dados.

Por fim, o esquemático do circuito implementado está ilustrado na FIGURA 15.

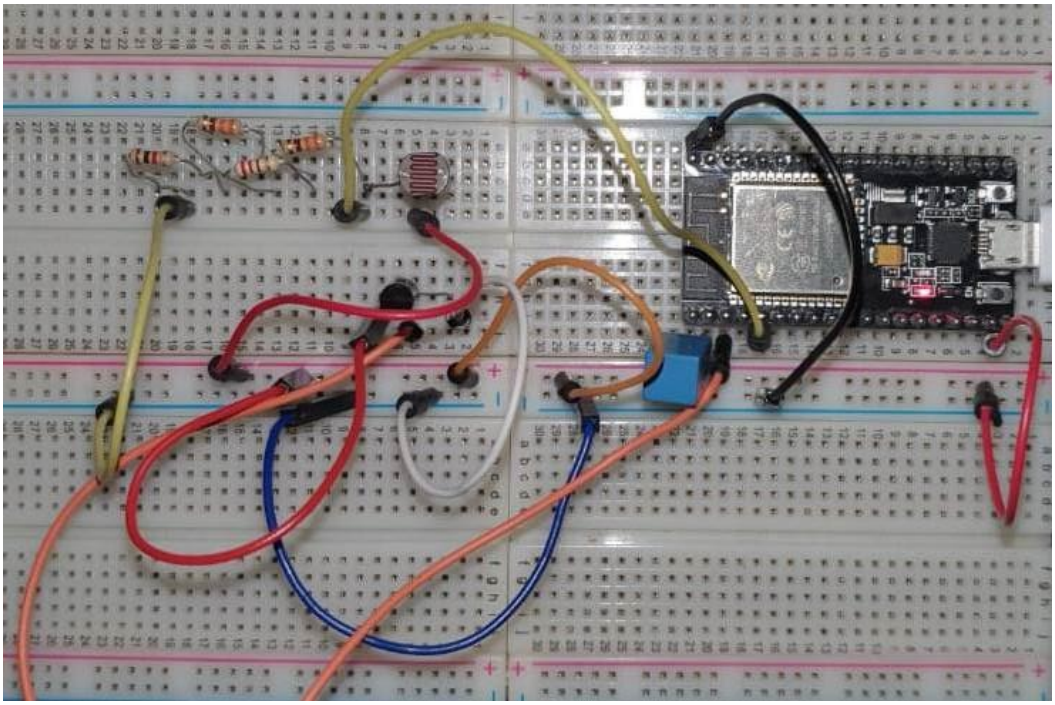
FIGURA 15 - ESQUEMÁTICO DO CIRCUITO IMPLEMENTADO



FONTE: Autoras, 2019

O esquemático foi testado na protoboard como ilustrado na FIGURA 16, para que o layout da placa do protótipo fosse desenhado.

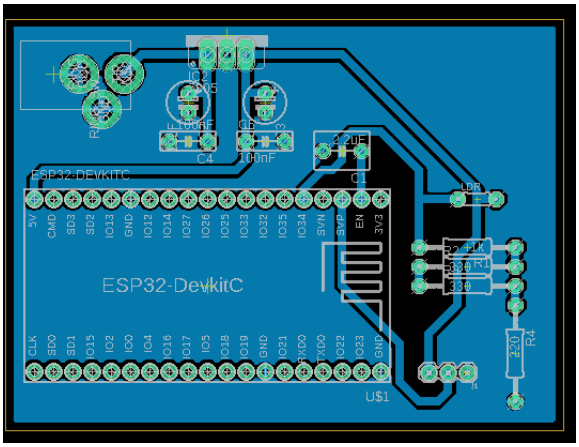
FIGURA 16 - CIRCUITO NA PROTOBOARD



FONTE: Autoras, 2019

O desenvolvimento do esquemático e layout foi feito com o software EAGLE, o layout pode ser observado na FIGURA 17.

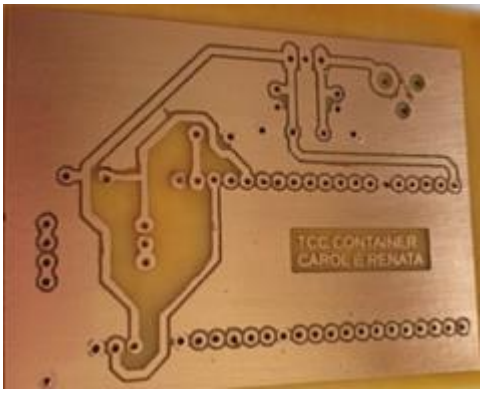
FIGURA 17 - LAYOUT DO CIRCUITO



FONTE: Autoras, 2019

Com o layout pronto, a placa do circuito foi confeccionada, conforme ilustrado na FIGURA 18.

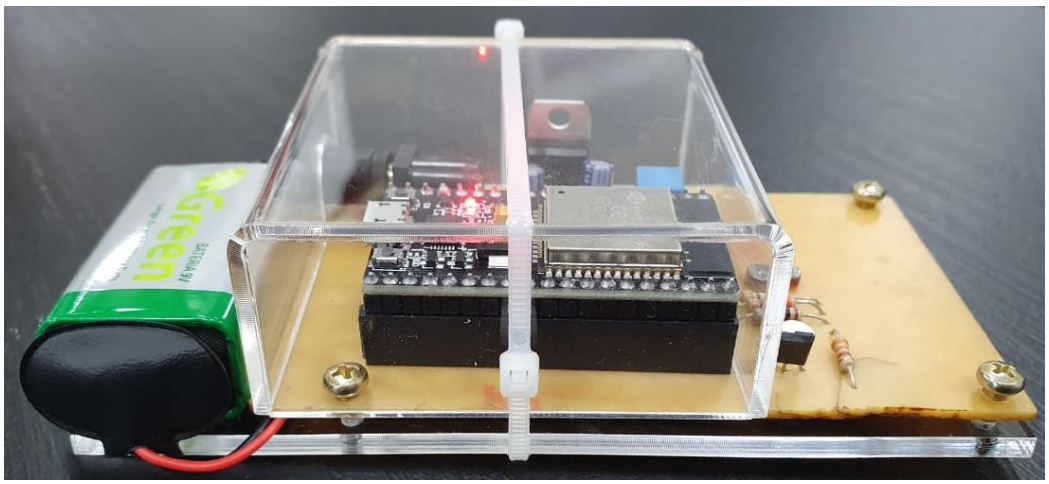
FIGURA 18 - PCB CONFECCIONADA



FONTE: Autoras, 2019

Os componentes do protótipo foram soldados à placa e uma caixa de acrílico foi customizada para o encapsulamento do protótipo, conforme ilustrado na FIGURA 19.

FIGURA 19 - PROTÓTIPO COM ENCAPSULAMENTO ACRÍLICO



FONTE: Autoras, 2019

5.5.1. ANÁLISE DE CUSTO

O preço final do protótipo detector de intrusão de contêineres ficou em torno de R$69,64, preço de varejo, custeado pelas autoras. O valor calculado diz respeito apenas aos custos dos componentes, sem levar em conta as horas-engenheiro necessárias para o seu desenvolvimento. Desta forma, o objetivo de baixo custo foi atingido considerando que um contêiner refrigerado pode levar em média 28 toneladas de carne, com o preço do kg da carne de R$20,00, o preço da carga que está sendo protegida é de R$560.000,00, ou seja, o custo do projeto representa 0,012% do valor da carga.

TABELA 2 - ANÁLISE DE CUSTO

Componente	Custo unitário	U	Custo Final
Placa ESP-32 - AC000203	R$ 43,90	1	R$ 43,90
Capacitor 100nF	R$ 0,08	2	R$ 0,16
Capacitor 10µF	R$ 0,09	2	R$ 0,18
Capacitor 2,2µF	R$ 2,03	1	R$ 2,03
Regulador de tensão LM7805	R$ 0,95	1	R$ 0,95
Conector DC P4 fêmea	R$ 0,90	1	R$ 0,90
Cabo Adaptador Bateria 9v	R$ 8,90	1	R$ 8,90
Resistor 330Ω	R$ 0,07	2	R$ 0,14
Resistor 220Ω	R$ 0,07	1	R$ 0,07
Resistor 1kΩ	R$ 0,07	1	R$ 0,07
Sensor LDR	R$ 0,30	1	R$ 0,30
Sensor LM35	R$ 8,44	1	R$ 8,44
Placa Fenolite	R$ 3,60	1	R$ 3,60
TOTAL			R$ 69,64

FONTE: Autoras, 2019

4.2. FIRMWARE

O código inicia com a inclusão das bibliotecas que serão utilizadas. Foram declaradas oito bibliotecas, que estão mostradas na FIGURA 20, as bibliotecas <*freertos/FreeRTOS.h*>, <*freertos/task.h*> e <*esp_system.h*> servem para o reconhecimento do ESP32, a biblioteca <*time.h*> e <*sys/time.h*> tem como utilidade o set do relógio de tempo real, as bibliotecas <*Wi-Fi.h*> e <*Wire.h*> foram utilizadas

para programação do Wi-Fi do ESP32 e a biblioteca <*SPI.h*> foi utilizada para o armazenamento.

FIGURA 20 - BIBLIOTECAS UTILIZADAS

```
2 #include <freertos/FreeRTOS.h>
3 #include <freertos/task.h>
4 #include <esp_system.h>
5 #include <time.h>
6 #include <sys/time.h>
7 #include <WiFi.h>
8 #include <Wire.h>
9 #include <SPI.h>
```

FONTE: Autoras, 2019

Logo após a inclusão das bibliotecas, as variáveis foram declaradas e, o código pode ser observado na FIGURA 21. As variáveis *integralPartSize* e *decimalPartSize* servem para especificar o tamanho do inteiro e do decimal dos vetores de armazenamento, a variável *sizeOfRecord* determina o tamanho da linha em caracteres de cada variável do armazenamento. Para definir as portas do microcontrolador que receberiam os sinais dos sensores de luminosidade e temperatura foram utilizadas as variáveis *portaLDR* e *LM35*, respectivamente. A variável *temperatura* armazena a temperatura medida. Para criar a estrutura que contém as informações de data foi utilizada a *tm data*. A rede *wirelles* criada pelo microcontrolador foi definida com uma *ssid, "TCC Contêiner"* e uma senha (*password*), *123456789*. O servidor do Wi-Fi foi setado para a porta 80 e a *string header* armazena o HTTP requerido.

FIGURA 21 - DECLARAÇÃO DE VARIÁVEIS

```
13 const int integralPartSize = 5, decimalPartSize = 2;
14 const int sizeOfRecord = 13;
15 const int portaLDR = 34;
16 const int LM35 = 36;
17 float temperatura;
18 struct tm data;
19 const char* ssid     = "TCC Contêiner";
20 const char* password = "123456789";
21 String header;
22 WiFiServer server(80);
```

FONTE: Autoras, 2019

A função void setup() é executada quando o programa começa e serve para configurar os pinos da placa e iniciar a comunicação serial com o computador. Está mostrada na FIGURA 22. A velocidade determinada para iniciar a serial foi determinada na linha 179. O pino declarado para a leitura do sensor LDR foi definido como pino de entrada com a função pinMode na linha 180. As funções escritas nas linhas 182 a 184 configuram as SPIFFS que serão utilizadas no armazenamento, como já explicado anteriormente. As funções que estão escritas nas linhas 187 a 189 configuram o RTC, de forma que a linha 187 cria a estrutura que é utilizada na linha 188 que insere o tempo no formato Unix Time e a linha 189 configura o RTC para manter a data atualizada. As linhas de 191 a 196 estão relacionadas ao Wi-Fi. A linha 191 mostra que o Wi-Fi está sendo conectado e a 191 entra com a ssid e password da rede criada. As linhas 194 a 196 setam o IP que está sendo utilizado e o imprimem na serial. A linha 199 lista todos os diretórios que estão armazenados nas SPIFFS, e a 200 cria o arquivo onde os dados serão armazenados e escreve o início do arquivo.

FIGURA 22 - FUNÇÃO *void setup()*

```
177 void setup() {
178
179   Serial.begin(9600);
180   pinMode(portaLDR, INPUT);
181
182    if(!SPIFFS.begin(true)){
183         Serial.println("SPIFFS Mount Failed");
184         return;
185     }
186
187   timeval tv;
188   tv.tv_sec = 1574260580;
189   settimeofday(&tv, NULL);
190
191   Serial.print("Setting AP (Access Point)...");
192   WiFi.softAP(ssid, password);
193
194   IPAddress IP = WiFi.softAPIP();
195   Serial.print("AP IP address: ");
196   Serial.println(IP);
197
198
199   listDir(SPIFFS, "/", 0);
200   writeFile(SPIFFS, "/container.txt", "início 13/11/2019\n");
201 }
```

FONTE: Autoras, 2019

A função void loop() está nas linhas 206 a 283, executa as operações que estão dentro dela em *loops* consecutivos e será explicada em partes.

A FIGURA 23 mostra a primeira parte do loop. A primeira linha dessa função, 207, é referente ao Wi-Fi e serve para iniciar o servidor. As linhas 209 a 215, formatam a data e imprimem a data formatada.

Na linha 217 a leitura do sensor LDR é realizada. Para a conversão da leitura dos sensores de temperatura e luminosidade, foi utilizado o número 5, que é a máxima tensão de saída, para multiplicar, e o valor 4095, que representa a resolução da porta analógica, para dividir. A conversão da leitura do LDR foi feita na linhas 219 a 220 e do LM35 foi feita na linha 221. As linhas 222 e 223 configuram o formato das variáveis que contem a luminosidade e a temperatura. A linha 225 possui um delay para realizar uma leitura a cada 2 segundos e a linha 226 possui uma função para esperar 1 segundo. Por fim, as linhas 228 e 229, armazenam as grandezas de luminosidade, temperatura e data e o arquivo txt é criado.

FIGURA 23 - FUNÇÃO *void loop()* parte 1

```
206 void loop() {
207   server.begin();
208 
209     time_t tt = time(NULL);
210     data = *gmtime(&tt);
211 
212     char data_formatada[64];
213     strftime(data_formatada, 64, "%d/%m/%Y %H:%M:%S", &data);
214     printf("\nUnix Time: %d\n", int32_t(tt));
215     printf("Data formatada: %s\n", data_formatada);
216 
217   int leitura = analogRead(portaLDR);
218   float V = leitura*(5/4095.0);
219   float x = (0.0004735978*(4095-leitura)/leitura);
220   float Lux = pow(x,-0.81308746);
221   temperatura = (float(analogRead(LM35))*5/(4095))/0.01;
222   char luxchar[100];
223   char tempchar[100];
224 
225   delay(1000);
226   vTaskDelay(pdMS_TO_TICKS(1000));//Espera 1 seg
227 
228     armazena(Lux,temperatura,data_formatada);
229     readFile(SPIFFS, "/container.txt");
```

FONTE: Autoras, 2019

A FIGURA 24 mostra a segunda parte do loop, onde foi configurada a página que aparece no celular quando o contêiner é violado, com as grandezas de luminosidade, temperatura e a data, este envio é mostrado nas linhas 250 a 262. Como código base de configuração do Wi-Fi foi utlizado o código do autor SANTOS, Rui, 2017. No código completo, ainda foi utilizada uma função para armazenamento, cuja autoria é de SUHANKO, Djames, 2019.

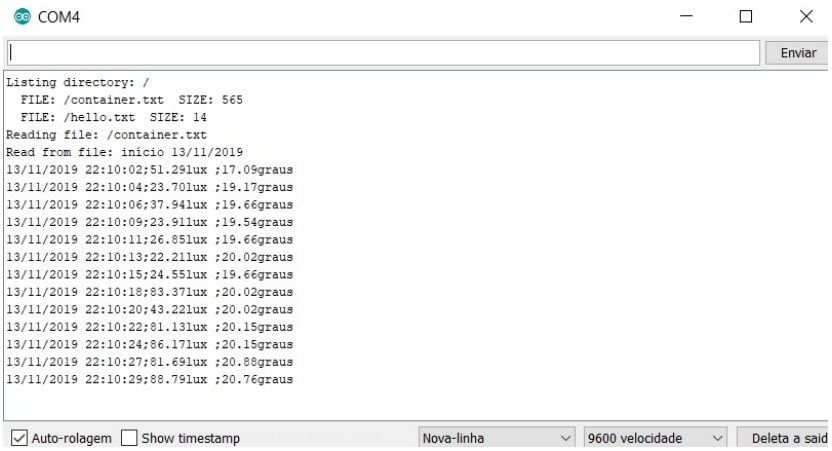
FIGURA 24 - FUNÇÃO void loop() parte 2

```
231     WiFiClient client = server.available();
232       //TESTE WEB SERVICE
233     if (client) {
234     Serial.println("New Client.");
235     String currentLine = "";
236     while (client.connected()) {
237       if (client.available()) {
238         char c = client.read();
239         Serial.write(c);
240         header += c;
241         if (c == '\n') {
242           // if the current line is blank, you got two newline characters in a row.
243           // that's the end of the client HTTP request, so send a response:
244           if (currentLine.length() == 0) {
245             client.println("HTTP/1.1 200 OK");
246             client.println("Content-type:text/html");
247             client.println("Connection: close");
248             client.println();
249
250             // Mostra a pagina HTML
251             client.println("<!DOCTYPE html><html>");
252             client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
253             client.println("<link rel=\"icon\" href=\"data:,\">");
254
255             // Começo da pagina Web
256               client.println("<body><h1>TCC CONTAINER 2019</h1>");
257               client.print(data_formatada);
258               client.print(" ; ");
259               client.print(Lux);
260               client.print(" lux ; ");
261               client.print(temperatura);
262               client.print(" graus ; ");
263
264             client.println();
265             break;
266           } else { // if you got a newline, then clear currentLine
267             currentLine = "";
268           }
269         } else if (c != '\r') {  // if you got anything else but a carriage return character,
270           currentLine += c;      // add it to the end of the currentLine
271         }
272       }
273     }
274     header = "";
275     client.stop();
276     Serial.println("Client disconnected.");
277     Serial.println("");
278   }
279
280      header = "";
281     client.stop();
282     Serial.println("");
283 }
```

FONTE: Autoras, 2019

O resultado da programação do ESP32 é visualizado e acompanhado pela porta serial ao qual foi salvo e é ilustrado na FIGURA 25 os resultados parciais do protótipo que foram extraídos em laboratório.

FIGURA 25 - PORTA SERIAL COM RESULTADO PARCIAL



FONTE: Autoras, 2019

Os dados apresentados na porta serial são apresentados e enviados à um Web Service de IP: 192.168.4.1 definido na programação do ESP32 e o resultado apresentado é apresentado na FIGURA 26.

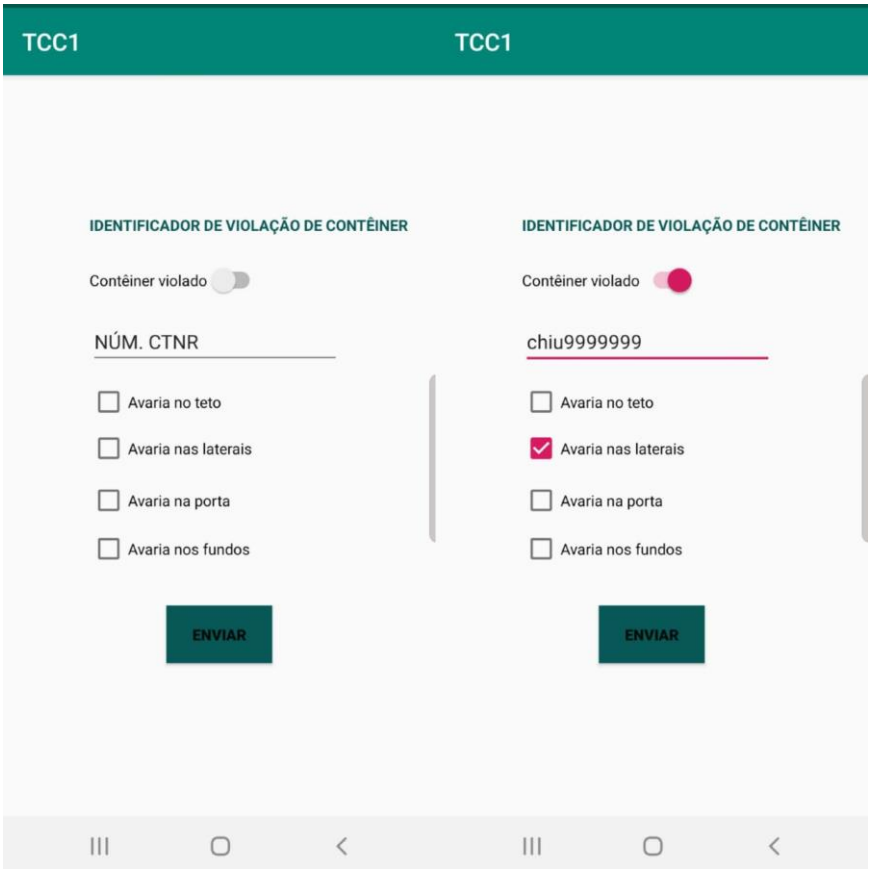
FIGURA 26 - WEBSERVICE

FONTE: Autoras, 2019

Os dados são atualizados em tempo real e por este WebService o aplicativo tem acesso às informações da intrusão do contêiner.

O aplicativo funciona de forma que o usuário se conecta ao Wi-Fi do protótipo, insere o código do contêiner avariado e marca a opção que identifica em qual parte do contêiner está o estrago. A tela do celular com o aplicativo está na FIGURA 27, a parte da esquerda mostra o aplicativo antes de inserir o dado do contêiner e na direita o aplicativo com os dados já inseridos.

FIGURA 27 - APLICATIVO NO CELULAR



FONTE: Autoras, 2019

4.3. TESTES

O protótipo foi levado para um terminal na região metropolitana de Curitiba e testado dentro de um contêiner, conforme mostra a FIGURA 28. Na foto o computador aparece com brilho alto, porém para o teste ilustrado o abaixo o

computador foi fechado, com o computador fechado o contêiner fica totalmente escuro, por isso não é possível mostrar com fotos.

FIGURA 28 - PROTÓTIPO DENTRO DO CONTÊINER

FONTE: Autoras, 2019

Com o protótipo dentro do contêiner, foi possível obter dados reais que comprovaram que 380 Lux é a luminosidade média de contêiner aberto, os dados são mostrados abaixo, na FIGURA 29. Os dados iniciam com a luminosidade zerada, quando o contêiner estava com a porta fechada, e aumenta gradativamente com a abertura da porta do contêiner, até chegar à luminosidade determinada como “inf”, ou seja, infinita, com a porta aberta.

FIGURA 29 - DADOS ARMAZENADOS TESTE NO CONTÊINER

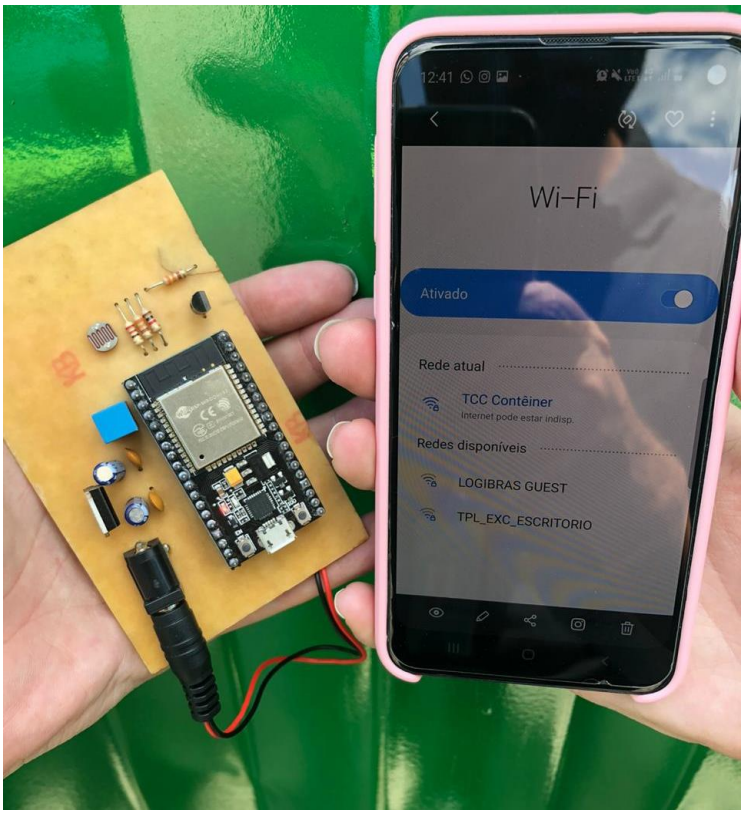
```
Unix Time: 1574260941
Data formatada: 20/11/2019 14:42:21

Appending to file: /container.txt
Appending to file: /container.txt
Appending to file: /container.txt
Appending to file: /container.txt
Appending to file: /container.txt
Appending to file: /container.txt
Reading file: /container.txt
Read from file: início 13/11/2019
20/11/2019 14:38:23 ; 0.00 lux ; 26.37 graus.
20/11/2019 14:38:27 ; 0.00 lux ; 26.50 graus.
20/11/2019 14:38:31 ; 0.00 lux ; 26.62 graus.
20/11/2019 14:38:35 ; 0.00 lux ; 27.11 graus.
20/11/2019 14:38:40 ; 30.54 lux ; 27.11 graus.
20/11/2019 14:38:44 ; 167.22 lux ; 26.86 graus.
20/11/2019 14:38:49 ; 144.95 lux ; 27.23 graus.
20/11/2019 14:38:53 ; 548.17 lux ; 26.86 graus.
20/11/2019 14:38:58 ; inf lux ; 25.52 graus.
20/11/2019 14:39:02 ; inf lux ; 500.00 graus.
```

FONTE: Autoras, 2019

O *access point* criado pelo protótipo conectado no celular é possível observar na FIGURA 30.

FIGURA 30 - Wi-Fi CONECTADO



FONTE: Autoras, 2019

5. CONCLUSÃO

Este trabalho foi desenvolvido de modo a construir um sistema para resolver um problema real, ou seja, a intrusão de contêineres para roubo de cargas, de maneira que os conhecimentos teóricos aprendidos durante o curso fossem consolidados.

O hardware atingiu o objetivo proposto de identificar um fenômeno estranho dentro do contentor de carga, com duas informações diferentes, garantindo a redundância de informações para que a intrusão seja identificada mesmo que haja alguma falha em um dos sensores.

O firmware teve por objetivo o armazenamento das informações de forma inteligente, para não ocorrer o esgotamento de memória do microcontrolador com informações desnecessárias e de forma que a data e hora do acontecimento de intrusão ficassem armazenados, e o objetivo foi atendido.

O aplicativo, por sua vez, objetivava obter os dados a partir da comunicação com o microcontrolador e gerar uma informação que fosse útil para que o operacional da empresa pudesse analisar e agir nos locais mais problemáticos, e o objetivo foram atingidos com a geração de um e-mail através do aplicativo.

5.1. MELHORIAS FUTURAS

a. Aviso em tempo real: encontrar alguma antena que possibilite o aviso de intrusão do contêiner em tempo real. O obstáculo encontrado para este desenvolvimento dentro do trabalho de conclusão de curso foi o tempo disponível para o projeto, porque como o contêiner funciona como uma gaiola de Faraday o projeto da antena precisa ser robusto.
b. Encapsulamento robusto: o encapsulamento precisa resguardar melhor o circuito, conseguir ser de fácil manuseio e acoplamento no contêiner.

REFERÊNCIAS

CORREA, Rosana. **Transporte de Container: O grande protagonista da globalização.** 2018. Disponível em: <https://brasilmaxi.com.br/>. Acesso em: 16 de agosto de 2019.

CBN. **Nos últimos 40 anos, Brasil foi ultrapassado por 12 países no ranking de participação no comércio internacional.** 2016. Disponível em: <https://cbn.globoradio.globo.com/>. Acesso: 16 de agosto de 2019.

BILLING, Jane. **Facts about Shipping containers.** 2012. Disponível em: <https://www.billiebox.co.uk/>. Acesso em: 16 de agosto de 2019.

BRADO, 2019. Disponível em: <http://www.brado.com.br/>. Acesso em: 18 de agosto de 2019.

ICONTAINERS. **A short history to shipping container dimensions**. 2019. Disponível em: <https://www.icontainers.com/the-different-types-of-containers/>. Acesso: 18 de agosto de 2019

AURELIANO, Andre. **Microcontroladores.** 2017. Disponível em: <https://www.fiozera.com.br/>. Acesso em: 20 de agosto de 2019.

ESPRESSIF SYSTEMS. **ESP32-WROOM-32 Datasheet.** 2019. Disponível em: < https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>. Acesso em: 20 de agosto de 2019.

RS COMPONENTS. **LIGHT DEPENDENT RESISTORS.** 1997. Disponível em: <http://www.bilimteknik.tubitak.gov.tr/sites/default/files/gelisim/elektronik/dosyalar/40/LDR_NSL19_M51.pdf>. Acessado em: 20 de agosto de 2019.

SOAWebServices. 2012. **Como funcionam os WebServices**. Disponível em: <https://www.soawebservices.com.br/como-funciona.aspx>. Acesso em: 24 setembro de 2019.

PUBLICIDADE LEGAL Nº21. BRADO LOGÍSTICA.2019. Disposível em: <https://www.bemparana.com.br/edital/page/2#.XdgLTOhKjDd>. Acesso: 20 de agosto de 2019.

SILVEIRA; Cristiano Bertulucci. **Sensor: Você Sabe o Que é Quais os Tipos?.** 2018. Disponível em: <https://www.citisystems.com.br/>. Acesso em: 24 de agosto de 2019.

AUSEC. **Conheça os tipos de sensores de presença mais indicados para sua empresa,** 2016. Disponível em: < http://blog.ausec.com.br/>. Acesso em: 25 de agosto de 2019.

SOUZA, Fábio. **Arduino MEGA 2560.** 2014. Disponível em: <https://www.embarcados.com.br/>. Acesso em: 24 de agosto de 2019.

BERNARDO, Jader. **MSP430 - INTRODUÇÃO.** 2015. Disponível em: <http://eletronworld.com.br/>. Acesso em: 30 de agosto de 2019.

RODRIGUES, Gabriela. Memórias RAM, ROM e Flash, 2014. Disponível em: <https://prezi.com/>. Acesso em: 30 de agosto de 2019.

TEXAS INSTRUMENTS. **LM35 Precision Centigrade Temperature Sensors.** 1999, revisado em 2017. Disponível em: <http://www.redrok.com/TemperatureSensor_LM35_10mVperC.pdf>. Acesso em: 24 de setembro de 2019.

CORRÊA, Underléa et al. Redes Locais sem Fio: Conceito e Aplicações. Florianópolis: Universidade Federal de Santa Catarina, 2006.

Epoch Unix Time Stamp Converter. 2014. Disponível em: <https://www.unixtimestamp.com/>.

SUHANKO, Djames. **Como escrever arquivos no SPIFFS com ESP32**. Disponível em: <https://www.dobitaobyte.com.br/>. Acesso em: 10/10/2019.

SANTOS, Rui. ESP32 **Web Server – Arduino IDE.** 2017. Disponível em: <https://randomnerdtutorials.com/>. Acesso em: 20/10/2019.

APENDICE A - FIRMWARE DO DISPOSITIVO

```
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <esp_system.h>
#include <time.h>
#include <sys/time.h>
#include <WiFi.h>
#include <Wire.h>
#include <SPI.h>

const int integralPartSize = 5, decimalPartSize = 2;
const int sizeOfRecord = 13;
const int portaLDR = 34;
const int LM35 = 36;
float temperatura;
struct tm data;
const char* ssid     = "TCC Contêiner";
const char* password = "123456789";
WiFiServer server(80);
String header;

//ARMAZENAMENTO
#include "FS.h"
#include "SPIFFS.h"
//função que lista ps diretórios dentro no spiffs
void listDir(fs::FS &fs, const char * dirname, uint8_t levels){
    Serial.printf("Listing directory: %s\n", dirname);

    File root = fs.open(dirname);
    if(!root){
        Serial.println("Failed to open directory");
        return;
    }
    if(!root.isDirectory()){
        Serial.println("Not a directory");
        return;
    }

    File file = root.openNextFile();
    while(file){
        if(file.isDirectory()){
            Serial.print("  DIR : ");
            Serial.println(file.name());
            if(levels){
                listDir(fs, file.name(), levels -1);
            }
        } else {
            Serial.print("  FILE: ");
```

```
            Serial.print(file.name());
            Serial.print("  SIZE: ");
            Serial.println(file.size());
        }
        file = root.openNextFile();
    }
}
//função que lê os arquivos
void readFile(fs::FS &fs, const char * path){
  WiFiClient client = server.available();
    Serial.printf("Reading file: %s\n", path);

    File file = fs.open(path);
    if(!file || file.isDirectory()){
        Serial.println("Failed to open file for reading");
        return;
    }

     Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
    }
}

//função que escreve arquivos no sistema
void writeFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Writing file: %s\n", path);

    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)){
        Serial.println("File written");
    } else {
        Serial.println("Write failed");
    }
}
//função que add um conteúdo à um arquivo existente
void appendFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Appending to file: %s\n", path);

    File file = fs.open(path, FILE_APPEND);
    if(!file){
        Serial.println("Failed to open file for appending");
        return;
    }
```

```
    if(file.print(message)){
        //Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
}

//função que deleta um arquivo existente
void deleteFile(fs::FS &fs, const char * path){
    Serial.printf("Deleting file: %s\n", path);
    if(fs.remove(path)){
        Serial.println("File deleted");
    } else {
        Serial.println("Delete failed");
    }
}
//função que diz o tamanho do arquivo
void testFileIO(fs::FS &fs, const char * path){
    File file = fs.open(path);
    static uint8_t buf[512];
    size_t len = 0;
    uint32_t start = millis();
    uint32_t end = start;
    if(file && !file.isDirectory()){
        len = file.size();
        size_t flen = len;
        start = millis();
        while(len){
            size_t toRead = len;
            if(toRead > 512){
                toRead = 512;
            }
            file.read(buf, toRead);
            len -= toRead;
        }
        end = millis() - start;
        Serial.printf("%u bytes read for %u ms\n", flen, end);
        file.close();
    } else {
        Serial.println("Failed to open file for reading");
    }


    file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
```

```
    size_t i;
    start = millis();
    for(i=0; i<2048; i++){
        file.write(buf, 512);
    }
    end = millis() - start;
    Serial.printf("%u bytes written for %u ms\n", 2048 * 512, end);
    file.close();
}



char armazena(float lux, float temp, char* dat)
{
    char luxchar[100];
    char tempchar[100];
    dtostrf(lux,4,2,luxchar);
    dtostrf(temp,4,2,tempchar);
    appendFile(SPIFFS, "/container.txt", dat);
    appendFile(SPIFFS, "/container.txt", " ; ");
    appendFile(SPIFFS, "/container.txt",luxchar);
    appendFile(SPIFFS, "/container.txt", " lux ; ");
    appendFile(SPIFFS, "/container.txt", tempchar);
    appendFile(SPIFFS, "/container.txt", " graus. \n");
}


//SETUP

void setup() {

  Serial.begin(9600);
  pinMode(portaLDR, INPUT);

   if(!SPIFFS.begin(true)){
        Serial.println("SPIFFS Mount Failed");
        return;
    }

  timeval tv;
  tv.tv_sec = 1574260580;
  settimeofday(&tv, NULL);

  Serial.print("Setting AP (Access Point)…");
  WiFi.softAP(ssid, password);

  IPAddress IP = WiFi.softAPIP();
```

```
  Serial.print("AP IP address: ");
  Serial.println(IP);


  listDir(SPIFFS, "/", 0);
  writeFile(SPIFFS, "/container.txt", "início 13/11/2019\n");
}


//LOOP

void loop() {
  server.begin();

    time_t tt = time(NULL);
    data = *gmtime(&tt);

    char data_formatada[64];
    strftime(data_formatada, 64, "%d/%m/%Y %H:%M:%S", &data);
    printf("\nUnix Time: %d\n", int32_t(tt));
    printf("Data formatada: %s\n", data_formatada);

  int leitura = analogRead(portaLDR);
  float V = leitura*(5/4095.0);
  float x = (0.0004735978*(4095-leitura)/leitura);
  float Lux = pow(x,-0.81308746);
  temperatura = (float(analogRead(LM35))*5/(4095))/0.01;
 char luxchar[100];
 char tempchar[100];

  delay(1000);

    vTaskDelay(pdMS_TO_TICKS(1000));//Espera 1 seg


      armazena(Lux,temperatura,data_formatada);
      readFile(SPIFFS, "/container.txt");

    WiFiClient client = server.available();
      //TESTE WEB SERVICE
    if (client) {
    Serial.println("New Client.");
    String currentLine = "";
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        header += c;
```

```
        if (c == '\n') {
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Mostra a pagina HTML
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");

            // Começo da pagina Web
              client.println("<body><h1>TCC CONTAINER 2019</h1>");
              client.print(data_formatada);
              client.print(" ; ");
              client.print(Lux);
              client.print(" lux ; ");
              client.print(temperatura);
              client.print(" graus ; ");

            client.println();
            break;
          } else { // if you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    header = "";
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }

     header = "";
    client.stop();
    Serial.println("");
}
```

APENDICE B - CÓDIGO DO APLICATIVO

// layout APP //

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textTITULO"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_marginStart="72dp"
        android:layout_marginLeft="72dp"
        android:layout_marginTop="116dp"
        android:text="@string/identificador_de_viola_o_de_cont_iner"
        android:textColor="#075856"
        android:textColorLink="#046B61"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/numeroConteiner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25dp"
        android:autofillHints=""
        android:ems="11"
        android:inputType="textPersonName"
        android:labelFor="@+id/numeroConteiner"
        android:maxLength="11"
        android:text="NÚM. CTNR"
        app:layout_constraintBottom_toTopOf="@+id/checkTeto"
        app:layout_constraintStart_toStartOf="@+id/swtVioacao"
        app:layout_constraintTop_toBottomOf="@+id/swtVioacao" />

    <Switch
        android:id="@+id/swtVioacao"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25dp"
        android:text="@string/cont_iner_violado"
        android:textOff="@string/n_o"
```

```
    android:textOn="@string/sim"
    app:layout_constraintBottom_toTopOf="@+id/numeroConteiner"
    app:layout_constraintStart_toStartOf="@+id/textTITULO"
    app:layout_constraintTop_toBottomOf="@+id/textTITULO" />

<Button
    android:id="@+id/btnEnviar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="136dp"
    android:background="#075856"
    android:text="@string/enviar"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/checkFundos" />

<CheckBox
    android:id="@+id/checkLaterais"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp"
    android:text="@string/avaria_nas_laterais"
    app:layout_constraintBottom_toTopOf="@+id/checkPortas"
    app:layout_constraintStart_toStartOf="@+id/checkTeto"
    app:layout_constraintTop_toBottomOf="@+id/checkTeto" />

<CheckBox
    android:id="@+id/checkFundos"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="40dp"
    android:text="@string/avaria_nos_fundos"
    app:layout_constraintBottom_toTopOf="@+id/btnEnviar"
    app:layout_constraintStart_toStartOf="@+id/checkPortas"
    app:layout_constraintTop_toBottomOf="@+id/checkPortas" />

<CheckBox
    android:id="@+id/checkPortas"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="17dp"
    android:text="@string/avaria_na_porta"
    app:layout_constraintBottom_toTopOf="@+id/checkFundos"
    app:layout_constraintStart_toStartOf="@+id/checkLaterais"
    app:layout_constraintTop_toBottomOf="@+id/checkLaterais" />
```

```
    <CheckBox
        android:id="@+id/checkTeto"
        android:layout_width="wrap_content"
        android:layout_height="24dp"
        android:layout_marginBottom="19dp"
        android:text="@string/avaria_no_teto"
        app:layout_constraintBottom_toTopOf="@+id/checkLaterais"
        app:layout_constraintStart_toStartOf="@+id/numeroConteiner"
        app:layout_constraintTop_toBottomOf="@+id/numeroConteiner" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_begin="20dp" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_begin="16dp" />
```

//ACTIVITY MAIN //

```
package com.example.tcc1;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    TextView textTITULO;
    EditText numeroConteiner;
    Switch swtVioacao;
    CheckBox checkTeto;
    CheckBox checkLaterais;
    CheckBox checkPortas;
    CheckBox checkFundos;
    Button btnEnviar;
```

```
    boolean dadosValidados = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textTITULO = findViewById(R.id.textTITULO);
        numeroConteiner = findViewById(R.id.numeroConteiner);
        swtVioacao = findViewById(R.id.swtVioacao);
        checkTeto = findViewById(R.id.checkTeto);
        checkLaterais = findViewById(R.id.checkLaterais);
        checkPortas =  findViewById(R.id.checkPortas);
        checkFundos = findViewById(R.id.checkFundos);
        btnEnviar = findViewById(R.id.btnEnviar);

        btnEnviar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(getApplicationContext(), "Numero Conteiner:
"+numeroConteiner.getText(),Toast.LENGTH_LONG).show();
                Toast.makeText(getApplicationContext(), "Avaria:
"+swtVioacao.getText(),Toast.LENGTH_LONG).show();

            }

        });

    }
}
```

ANEXO A - DATASHEET ESP32

ESP32-WROOM-32

Datasheet

Version 2.9
Espressif Systems

About This Document

This document provides the specifications for the ESP32-WROOM-32 module.

Revision History

For revision history of this document, please refer to the last page.

Documentation Change Notification

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

Certification

Download certificates for Espressif products from www.espressif.com/en/certificates.

Disclaimer and Copyright Notice

Contents

List of Tables

List of Figures

1. Overview

ESP32-WROOM-32 is a powerful, generic Wi-Fi+BT+BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

At the core of this module is the ESP32-D0WDQ6 chip*. The chip embedded is designed to be scalable and adaptive. There are two CPU cores that can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. The user may also power off the CPU and make use of the low-power co-processor to constantly monitor the peripherals for changes or crossing of thresholds. ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I²S and I²C.

Note:

* For details on the part numbers of the ESP32 family of chips, please refer to the document *ESP32 Datasheet*.

The integration of Bluetooth, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is all-around: using Wi-Fi allows a large physical range and direct connection to the Internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 μA, making it suitable for battery powered and wearable electronics applications. The module supports a data rate of up to 150 Mbps, and 20 dBm output power at the antenna to ensure the widest physical range. As such the module does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that users can upgrade their products even after their release, at minimum cost and effort.

Table 1 provides the specifications of ESP32-WROOM-32.

Table 1: ESP32-WROOM-32 Specifications

Categories	Items	Specifications
Certification	RF certification	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Test	Reliablity	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH

Categories	Items	Specifications
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I²C, LED PWM, Motor PWM, I²S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range	–40 °C ~ +85 °C
	Package size	(18.00±0.10) mm × (25.50±0.10) mm × (3.10±0.10) mm
	Moisture sensitivity level (MSL)	Level 3

2. Pin Definitions

2.1 Pin Layout

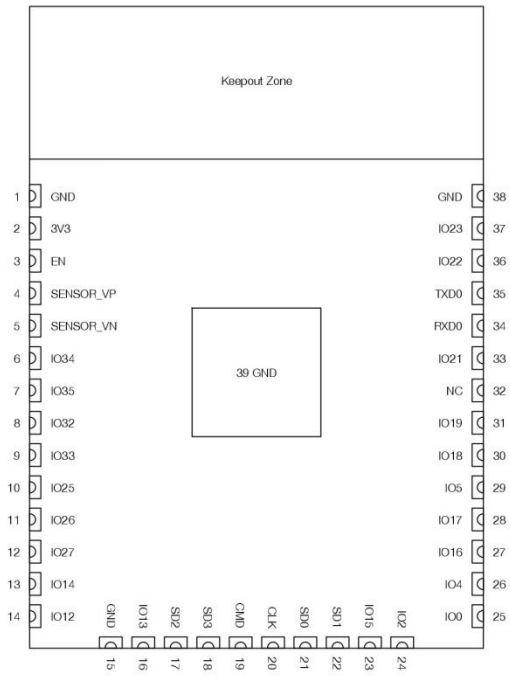


Figure 1: ESP32-WROOM-32 Pin Layout (Top View)

2.2 Pin Description

ESP32-WROOM-32 has 38 pins. See pin definitions in Table 2.

Table 2: Pin Definitions

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	Module-enable signal. Active high.
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8

Name	No.	Type	Function
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

Notice:

* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on the module and are not recommended for other uses.

2.3 Strapping Pins

ESP32 has five strapping pins, which can be seen in Chapter 6 Schematics:

- MTDI
- GPIO0
- GPIO2
- MTDO
- GPIO5

Software can read the values of these five bits from register "GPIO_STRAPPING".

During the chip's system reset release (power-on-reset, RTC watchdog reset and brownout reset), the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device's boot mode, the operating voltage of VDD_SDIO and other initial system settings.

Each strapping pin is connected to its internal pull-up/pull-down during the chip reset. Consequently, if a strapping pin is unconnected or the connected external circuit is high-impedance, the internal weak pull-up/pull-down will determine the default input level of the strapping pins.

To change the strapping bit values, users can apply the external pull-down/pull-up resistances, or use the host MCU's GPIOs to control the voltage level of these pins when powering on ESP32.

After reset release, the strapping pins work as normal-function pins.

Refer to Table 3 for a detailed boot-mode configuration by strapping pins.

Table 3: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3 V		1.8 V	
MTDI	Pull-down	0		1	
Booting Mode					
Pin	Default	SPI Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Enabling/Disabling Debugging Log Print over U0TXD During Booting					
Pin	Default	U0TXD Active		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	Falling-edge Sampling Falling-edge Output	Falling-edge Sampling Rising-edge Output	Rising-edge Sampling Falling-edge Output	Rising-edge Sampling Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

Note:

- Firmware can configure register bits to change the settings of "Voltage of Internal LDO (VDD_SDIO)" and "Timing of SDIO Slave" after booting.
- The module integrates a 3.3 V SPI flash, so the pin MTDI cannot be set to 1 when the module is powered up.

The strapping pins need a setup and hold time before and after the EN signal goes high. For details please refer to Section Strapping Pins in *ESP32 Datasheet*.

3. Functional Description

This chapter describes the modules and functions integrated in ESP32-WROOM-32.

3.1 CPU and Internal Memory

ESP32-D0WDQ6 contains two low-power Xtensa® 32-bit LX6 microprocessors. The internal memory includes:

- 448 KB of ROM for booting and core functions.
- 520 KB of on-chip SRAM for data and instructions.
- 8 KB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 KB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 Kbit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID.

3.2 External Flash and SRAM

ESP32 supports multiple external QSPI flash and SRAM chips. More details can be found in Chapter SPI in the *ESP32 Technical Reference Manual*. ESP32 also supports hardware encryption/decryption based on AES to protect developers' programs and data in flash.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- The external flash can be mapped into CPU instruction memory space and read-only memory space simultaneously.
  - When external flash is mapped into CPU instruction memory space, up to 11 MB + 248 KB can be mapped at a time. Note that if more than 3 MB + 248 KB are mapped, cache performance will be reduced due to speculative reads by the CPU.
  - When external flash is mapped into read-only data memory space, up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads are supported.
- External SRAM can be mapped into CPU data memory space. Up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads and writes are supported.

ESP32-WROOM-32 integrates a 4 MB SPI flash, which is connected to GPIO6, GPIO7, GPIO8, GPIO9, GPIO10 and GPIO11. These six pins cannot be used as regular GPIOs.

3.3 Crystal Oscillators

The module uses a 40-MHz crystal oscillator.

3.4 RTC and Low-Power Management

With the use of advanced power-management technologies, ESP32 can switch between different power modes.

For details on ESP32's power consumption in different power modes, please refer to section "RTC and Low-Power Management" in *ESP32 Datasheet*.

4. Peripherals and Sensors

Please refer to Section Peripherals and Sensors in *ESP32 Datasheet*.

Note:
External connections can be made to any GPIO except for GPIOs in the range 6-11. These six GPIOs are connected to the module's integrated SPI flash. For details, please see Section 6 Schematics.

5. Electrical Characteristics

5.1 Absolute Maximum Ratings

Stresses beyond the absolute maximum ratings listed in Table 4 below may cause permanent damage to the device. These are stress ratings only, and do not refer to the functional operation of the device that should follow the recommended operating conditions.

Table 4: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDD33	Power supply voltage	–0.3	3.6	V
I_{output} [1]	Cumulative IO output current	-	1,100	mA
T_{store}	Storage temperature	–40	150	°C

1. The module worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground. Please note that pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.
2. Please see Appendix IO_MUX of *ESP32 Datasheet* for IO's power domain.

5.2 Recommended Operating Conditions

Table 5: Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power supply voltage	3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T	Operating temperature	–40	-	85	°C

5.3 DC Characteristics (3.3 V, 25 °C)

Table 6: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA

Symbol	Parameter	Min	Typ	Max	Unit
I_{OL}	Low-level sink current (VDD[1] = 3.3 V, V_{OL} = 0.495 V, output drive strength set to the maximum)	-	28	-	mA
R_{PU}	Resistance of internal pull-up resistor	-	45	-	kΩ
R_{PD}	Resistance of internal pull-down resistor	-	45	-	kΩ
V_{IL_nRST}	Low-level input voltage of CHIP_PU to power off the chip	-	-	0.6	V

Notes:

1. Please see Appendix IO_MUX of *ESP32 Datasheet* for IO's power domain. VDD is the I/O voltage for a particular power domain of pins.
2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.
3. Pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.

5.4 Wi-Fi Radio

Table 7: Wi-Fi Radio Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Operating frequency range ^{note1}	-	2412	-	2484	MHz
Output impedance ^{note2}	-	-	*note 2*	-	Ω
TX power ^{note3}	11n, MCS7	12	13	14	dBm
	11b mode	17.5	18.5	20	dBm
Sensitivity	11b, 1 Mbps	-	–98	-	dBm
	11b, 11 Mbps	-	–89	-	dBm
	11g, 6 Mbps	-	–92	-	dBm
	11g, 54 Mbps	-	–74	-	dBm
	11n, HT20, MCS0	-	–91	-	dBm
	11n, HT20, MCS7	-	–71	-	dBm
	11n, HT40, MCS0	-	–89	-	dBm
	11n, HT40, MCS7	-	–69	-	dBm
Adjacent channel rejection	11g, 6 Mbps	-	31	-	dB
	11g, 54 Mbps	-	14	-	dB
	11n, HT20, MCS0	-	31	-	dB
	11n, HT20, MCS7	-	13	-	dB

1. Device should operate in the frequency range allocated by regional regulatory authorities. Target operating frequency range is configurable by software.
2. For the modules that use IPEX antennas, the output impedance is 50 Ω. For other modules without IPEX antennas, users do not need to concern about the output impedance.
3. Target TX power is configurable based on device or certification requirements.

5.5 BLE Radio

5.5.1 Receiver

Table 8: Receiver Characteristics – BLE

Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @30.8% PER	-	-	–97	-	dBm
Maximum received signal @30.8% PER	-	0	-	-	dBm
Co-channel C/I	-	-	+10	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	–5	-	dB
	F = F0 – 1 MHz	-	–5	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–35	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–45	-	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	–10	-	-	dBm
	2000 MHz ~ 2400 MHz	–27	-	-	dBm
	2500 MHz ~ 3000 MHz	–27	-	-	dBm
	3000 MHz ~ 12.5 GHz	–10	-	-	dBm
Intermodulation	-	–36	-	-	dBm

5.5.2 Transmitter

Table 9: Transmitter Characteristics – BLE

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	0	-	dBm
Gain control step	-	-	3	-	dBm
RF power control range	-	–12	-	+9	dBm
Adjacent channel transmit power	F = F0 ± 2 MHz	-	–52	-	dBm
	F = F0 ± 3 MHz	-	–58	-	dBm
	F = F0 ± > 3 MHz	-	–60	-	dBm
$\Delta f1_{avg}$	-	-	-	265	kHz
$\Delta f2_{max}$	-	247	-	-	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	-	-	–0.92	-	-
ICFT	-	-	–10	-	kHz
Drift rate	-	-	0.7	-	kHz/50 μs
Drift	-	-	2	-	kHz

5.6 Reflow Profile

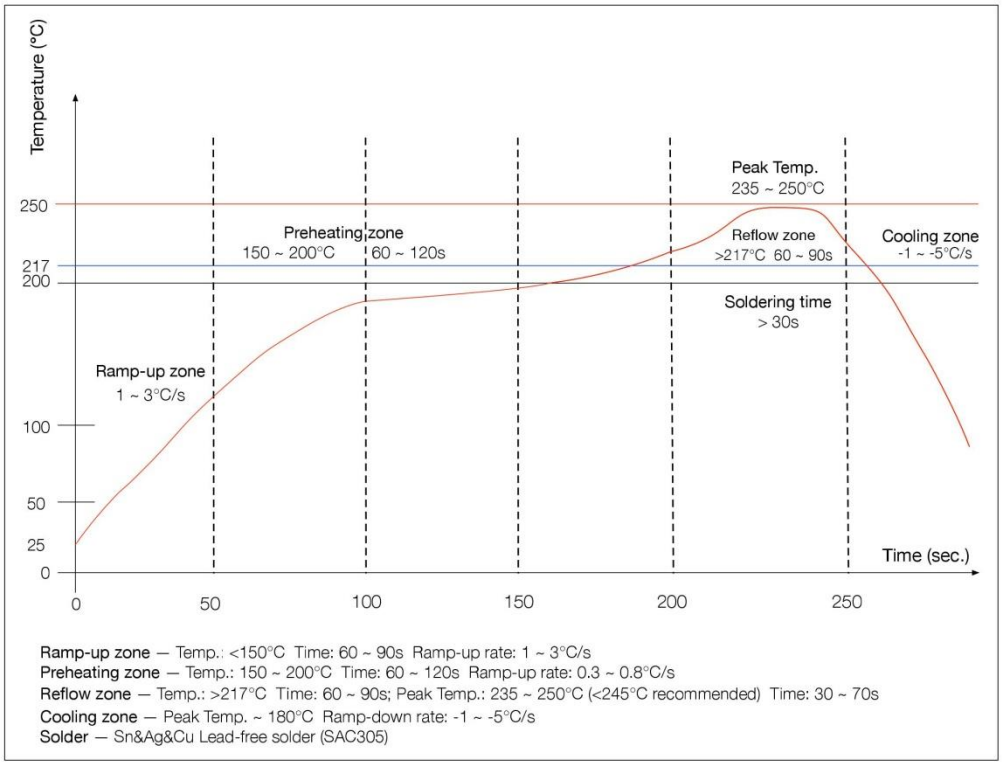


Figure 2: Reflow Profile

6. Schematics

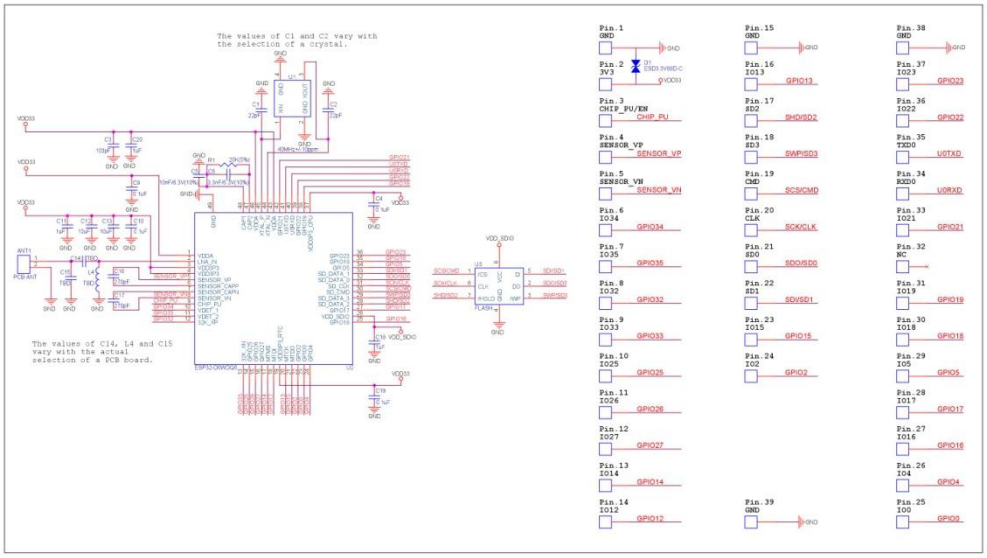


Figure 3: ESP32-WROOM-32 Schematics

7. Peripheral Schematics

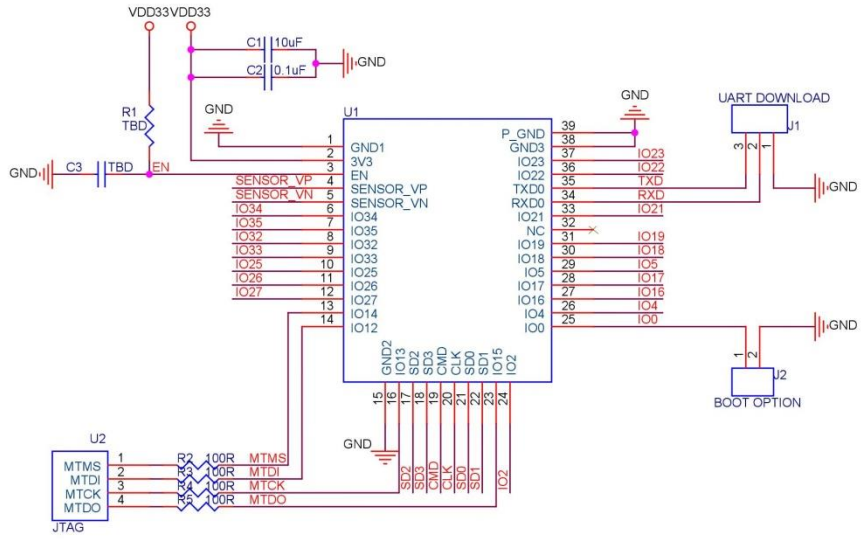


Figure 4: ESP32-WROOM-32 Peripheral Schematics

Note:

- Soldering Pad 39 to the Ground of the base board is not necessary for a satisfactory thermal performance. If users do want to solder it, they need to ensure that the correct quantity of soldering paste is applied.
- To ensure the power supply to the ESP32 chip during power-up, it is advised to add an RC delay circuit at the EN pin. The recommended setting for the RC delay circuit is usually R = 10 kΩ and C = 0.1 μF. However, specific parameters should be adjusted based on the power-up timing of the module and the power-up and reset sequence timing of the chip. For ESP32's power-up and reset sequence timing diagram, please refer to Section *Power Scheme* in *ESP32 Datasheet*.

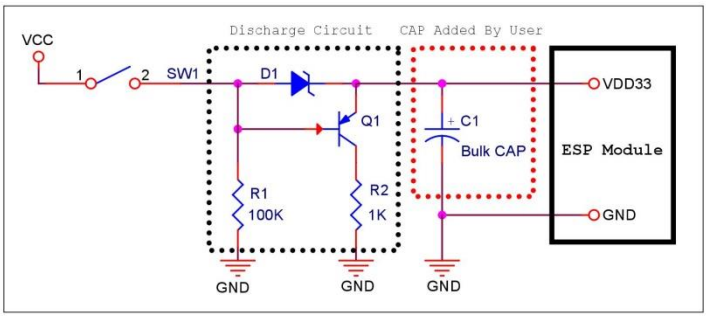


Figure 5: Discharge Circuit for VDD33 Rail

Note:
The discharge circuit can be applied in scenarios where ESP32 is powered on and off repeatedly by switching the power rails, and there is a large capacitor on the VDD33 rail. For details, please refer to Section *Power Scheme* in *ESP32 Datasheet*.

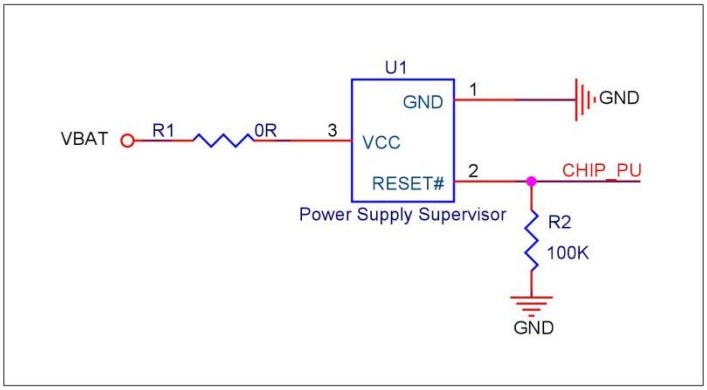


Figure 6: Reset Circuit

Note:
When battery is used as the power supply for ESP32 series of chips and modules, a supply voltage supervisor is recommended to avoid boot failure due to low voltage. Users are recommended to pull CHIP_PU low if the power supply for ESP32 is below 2.3 V.

8. Physical Dimensions

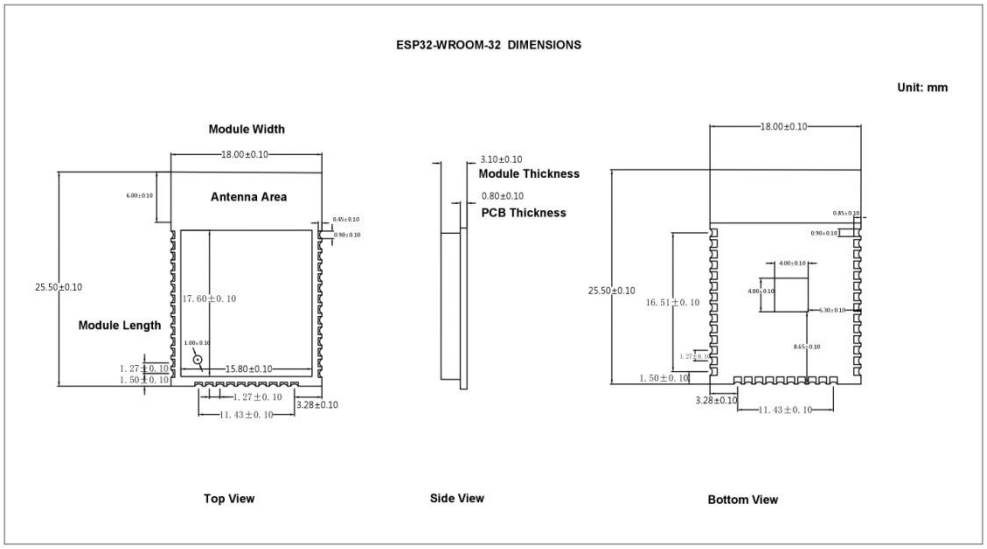


Figure 7: Physical Dimensions of ESP32-WROOM-32

9. Recommended PCB Land Pattern

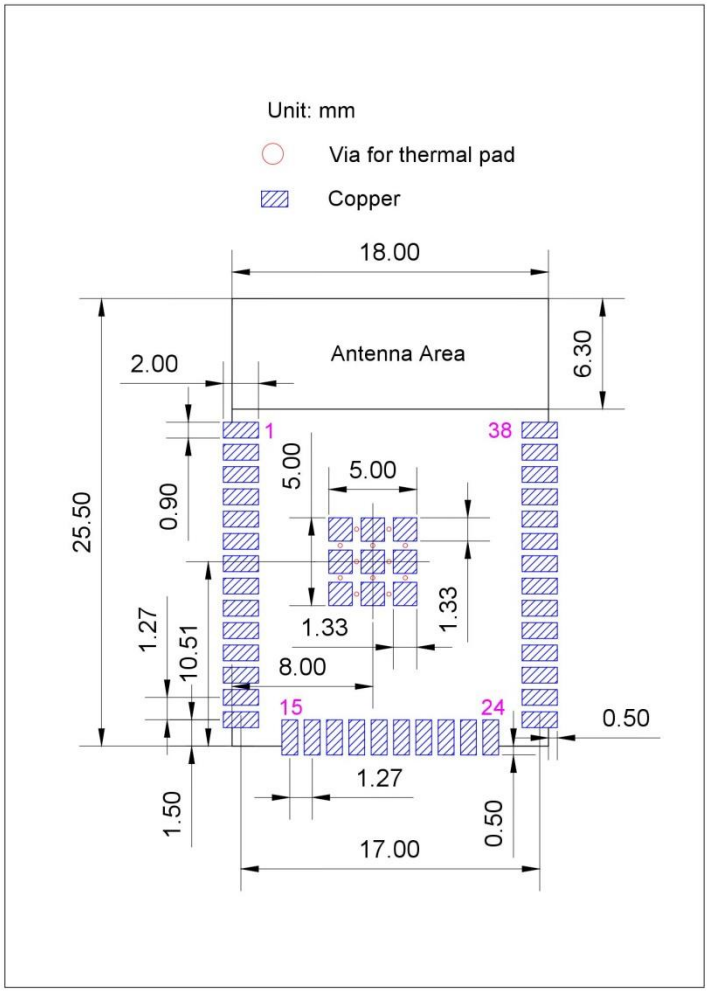


Figure 8: Recommended PCB Land Pattern

10. Learning Resources

10.1 Must-Read Documents

The following link provides documents related to ESP32.

- *ESP32 Datasheet*
  This document provides an introduction to the specifications of the ESP32 hardware, including overview, pin definitions, functional description, peripheral interface, electrical characteristics, etc.
- *ESP-IDF Programming Guide*
  It hosts extensive documentation for ESP-IDF ranging from hardware guides to API reference.
- *ESP32 Technical Reference Manual*
  The manual provides detailed information on how to use the ESP32 memory and peripherals.
- ESP32 Hardware Resources
  The zip files include the schematics, PCB layout, Gerber and BOM list of ESP32 modules and development boards.
- *ESP32 Hardware Design Guidelines*
  The guidelines outline recommended design practices when developing standalone or add-on systems based on the ESP32 series of products, including the ESP32 chip, the ESP32 modules and development boards.
- *ESP32 AT Instruction Set and Examples*
  This document introduces the ESP32 AT commands, explains how to use them, and provides examples of several common AT commands.
- *Espressif Products Ordering Information*

10.2 Must-Have Resources

Here are the ESP32-related must-have resources.

- ESP32 BBS
  This is an Engineer-to-Engineer (E2E) Community for ESP32 where you can post questions, share knowledge, explore ideas, and help solve problems with fellow engineers.
- ESP32 GitHub
  ESP32 development projects are freely distributed under Espressif's MIT license on GitHub. It is established to help developers get started with ESP32 and foster innovation and the growth of general knowledge about the hardware and software surrounding ESP32 devices.
- ESP32 Tools
  This is a webpage where users can download ESP32 Flash Download Tools and the zip file "ESP32 Certification and Test".
- ESP-IDF
  This webpage links users to the official IoT development framework for ESP32.
- ESP32 Resources
  This webpage provides the links to all available ESP32 documents, SDK and tools.

Revision History

Date	Version	Release notes
2019.09	V2.9	• Changed the supply voltage range from 2.7 V ~ 3.6 V to 3.0 V ~ 3.6 V; • Added Moisture sensitivity level (MSL) 3 in Table 1 *ESP32-WROOM-32 Specifications*; • Added notes about "Operating frequency range" and "TX power" under Table 7 *Wi-Fi Radio Characteristics*; • Updated Section 7 *Peripheral Schematics* and added a note about RC delay circuit under it; • Updated Figure 8 *Recommended PCB Land Pattern*.
2019.01	V2.8	Changed the RF power control range in Table 9 from –12 ~ +12 to –12 ~ +9 dBm.
2018.10	V2.7	Added "Cumulative IO output current" entry to Table 4: Absolute Maximum Ratings; Added more parameters to Table 6: DC Characteristics.
2018.08	V2.6	• Added reliability test items the module has passed in Table 1: ESP32-WROOM-32 Specifications, and removed software-specific information; • Updated section 3.4: RTC and Low-Power Management; • Changed the module's dimensions from (18±0.2) mm x (25.5 ±0.2) mm x (3.1±0.15) mm to (18.00±0.10) mm x (25.50±0.10) mm x (3.10±0.10) mm; • Updated Figure 8: Physical Dimensions; • Updated Table 7: Wi-Fi Radio.
2018.06	V2.5	• Changed the module name to ESP32-WROOM-32; • Deleted Temperature Sensor in Table 1: ESP32-WROOM-32 Specifications; • Updated Chapter 3: Functional Description; • Added Chapter 8: Recommended PCB Land Pattern; Changes to electrical characteristics: • Updated Table 4: Absolute Maximum Ratings; • Added Table 5: Recommended Operating Conditions; • Added Table 6: DC Characteristics; • Updated the values of "Gain control step", "Adjacent channel transmit power" in Table 9: Transmitter Characteristics - BLE.
2018.03	V2.4	Updated Table 1 in Chapter 1.
2018.01	V2.3	Deleted information on LNA pre-amplifier; Updated section 3.4 RTC and Low-Power Management; Added reset circuit in Chapter 7 and a note to it.
2017.10	V2.2	Updated the description of the chip's system reset in Section 2.3 Strapping Pins; Deleted "Association sleep pattern" in Table "Power Consumption by Power Modes" and added notes to Active sleep and Modem-sleep; Updated the note to Figure 4 Peripheral Schematics; Added discharge circuit for VDD33 rail in Chapter 7 and a note to it.
2017.09	V2.1	Updated operating voltage/power supply range updated to 2.7 ~ 3.6V; Updated Chapter 7.
2017.08	V2.0	Changed the sensitivity of NZIF receiver to -97 dBm in Table 1; Updated the dimensions of the module; Updated Table "Power Consumption by Power Modes" Power Consumption by Power Modes, and added two notes to it;

Date	Version	Release notes
		Updated Table 4, 7, 8, 9; Added Chapter 8; Added the link to certification download.
2017.06	V1.9	Added a note to Section 2.1 Pin Layout; Updated Section 3.3 Crystal Oscillators; Updated Figure 3 ESP-WROOM-32 Schematics; Added Documentation Change Notification.
2017.05	V1.8	Updated Figure 1 Top and Side View of ESP32-WROOM-32 (ESP-WROOM-32).
2017.04	V1.7	Added the module's dimensional tolerance; Changed the input impedance value of 50Ω in Table 7 Wi-Fi Radio Characteristics to output impedance value of 30+j10 Ω.
2017.04	V1.6	Added Figure 2 Reflow Profile.
2017.03	V1.5	Updated Section 2.2 Pin Description; Updated Section 3.2 External Flash and SRAM; Updated Section 4 Peripherals and Sensors Description.
2017.03	V1.4	Updated Chapter 1 Preface; Updated Chapter 2 Pin Definitions; Updated Chapter 3 Functional Description; Updated Table Recommended Operating Conditions; Updated Table 7 Wi-Fi Radio Characteristics; Updated Section 5.6 Reflow Profile; Added Chapter 10 Learning Resources.
2016.12	V1.3	Updated Section 2.1 Pin Layout.
2016.11	V1.2	Added Figure 7 Peripheral Schematics.
2016.11	V1.1	Updated Chapter 6 Schematics.
2016.08	V1.0	First release.

ANEXO B - DATASHEET LDR

Data pack F *Issued March 1997* **232-3816**

Data Sheet

Light dependent resistors

NORP12 RS stock number 651-507
NSL19-M51 RS stock number 596-141

Two cadmium sulphide (cdS) photoconductive cells with spectral responses similar to that of the human eye. The cell resistance falls with increasing light intensity. Applications include smoke detection, automatic lighting control, batch counting and burglar alarm systems.

Guide to source illuminations

Light source	Illumination (Lux)
Moonlight	0.1
60W bulb at 1m	50
1W MES bulb at 0.1m	100
Fluorescent lighting	500
Bright sunlight	30,000

Circuit symbol

Light memory characteristics

Light dependent resistors have a particular property in that they remember the lighting conditions in which they have been stored. This memory effect can be minimised by storing the LDRs in light prior to use. Light storage reduces equilibrium time to reach steady resistance values.

NORP12 (**RS** stock no. 651-507)

Absolute maximum ratings

Voltage, ac or dc peak	320V
Current	75mA
Power dissipation at 30°C	250mW
Operating temperature range	-60°C to +75°C

Electrical characteristics

T_A = 25°C. 2854°K tungsten light source

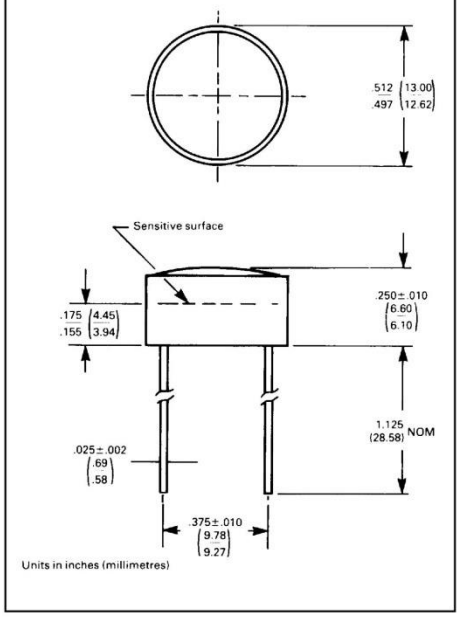
Parameter	Conditions	Min.	Typ.	Max.	Units
Cell resistance	1000 lux 10 lux	- -	400 9	- -	Ω kΩ
Dark resistance	-	1.0	-	-	MΩ
Dark capacitance	-	-	3.5	-	pF
Rise time 1	1000 lux 10 lux	- -	2.8 18	- -	ms ms
Fall time 2	1000 lux 10 lux	- -	48 120	- -	ms ms

1. Dark to 110% R_L
2. To 10 × R_L
R_L = photocell resistance under given illumination.

Features

- Wide spectral response
- Low cost
- Wide ambient temperature range.

Dimensions



232-3816

Figure 1 Power dissipation derating

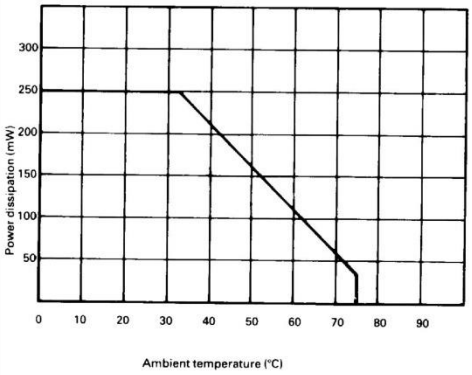
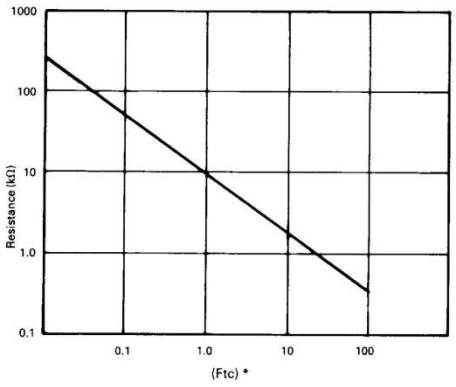
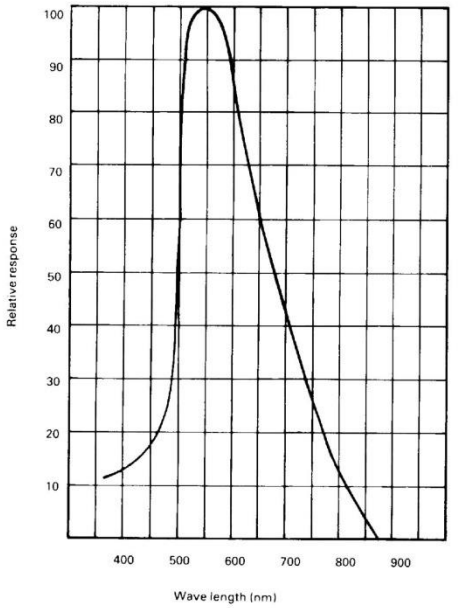


Figure 3 Resistance as a function of illumination



*1Ftc=10.764 lumens

Figure 2 Spectral response



232-3816

Absolute maximum ratings

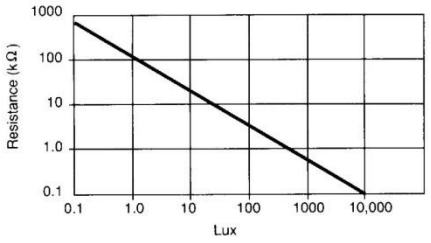
Voltage, ac or dc peak _____ 100V
Current _____ 5mA
Power dissipation at 25°C _____ 50mW*
Operating temperature range _____ -25°C +75°C

*Derate linearly from 50mW at 25°C to 0W at 75°C.

Electrical characteristics

Parameter	Conditions	Min.	Typ.	Max.	Units
Cell resistance	10 lux 100 lux	20 -	- 5	100 -	kΩ kΩ
Dark resistance	10 lux after 10 sec	20	-	-	MΩ
Spectral response	-	-	550	-	nm
Rise time	10ftc	-	45	-	ms
Fall time	10ftc	-	55	-	ms

Figure 4 Resistance as a function illumination



Dimensions

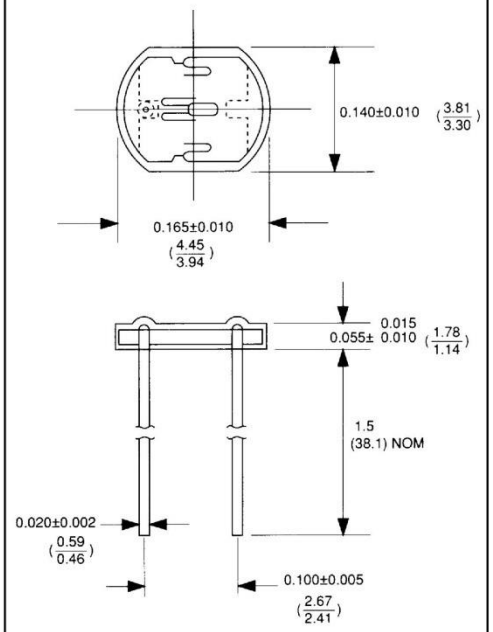
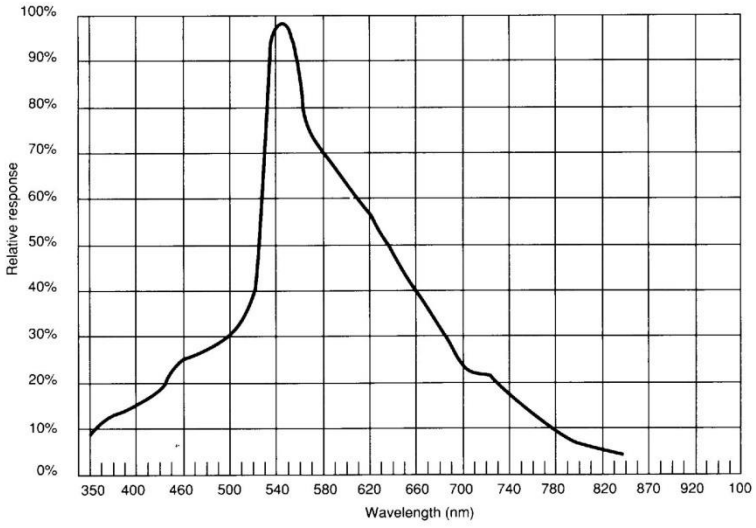


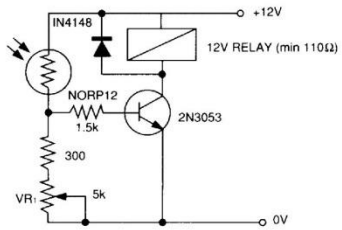
Figure 5 Spectral response



232-3816

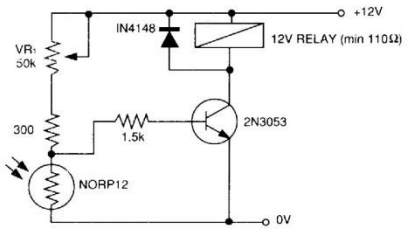
Typical application circuits

Figure 6 **Sensitive light operated relay**



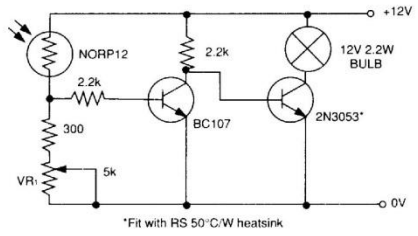
Relay energised when light level increases above the level set by VR_1

Figure 7 **Light interruption detector**



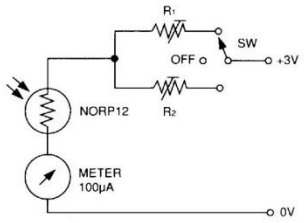
As Figure 6 relay energised when light level drops below the level set by VR_1

Figure 8 **Automatic light circuit**



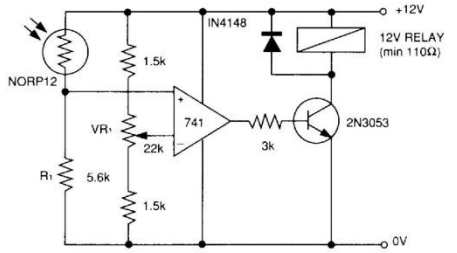
Adjust turn-on point with VR_1

Figure 9 **Logarithmic law photographic light meter**



Typical value R^1 = 100k**Ω**
R^2 = 200k**Ω** preset to give two overlapping ranges.
(Calibration should be made against an accurate meter.)

Figure 10 **Extremely sensitive light operated relay**



(Relay energised when light exceeds preset level.)
Incorporates a balancing bridge and op-amp. R_1 and NORP12 may be interchanged for the reverse function.

ANEXO C - DATASHEET LM35

LM35

LM35 Precision Centigrade Temperature Sensors

Literature Number: SNIS159B

November 2000

National Semiconductor

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

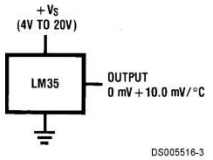
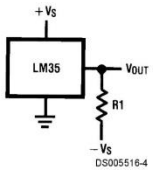


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)

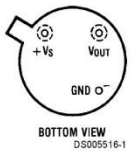


Choose $R_1 = -V_S/50\ \mu A$
V_{OUT} = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

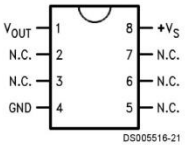
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

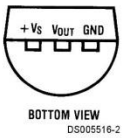
SO-8
Small Outline Molded Package



N.C. = No Connection

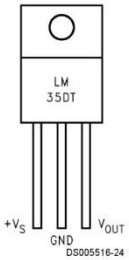
Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package



Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60˚C to +180˚C
TO-92 Package,	−60˚C to +150˚C
SO-8 Package,	−65˚C to +150˚C
TO-220 Package,	−65˚C to +150˚C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300˚C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260˚C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215˚C
Infrared (15 seconds)	220˚C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55˚C to +150˚C
LM35C, LM35CA	−40˚C to +110˚C
LM35D	0˚C to +100˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy	T_A=+25˚C	±0.2	±0.5		±0.2	±0.5		˚C
(Note 7)	T_A=−10˚C	±0.3			±0.3		±1.0	˚C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		˚C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	˚C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	˚C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/˚C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25˚C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25˚C	56	67		56	67		µA
	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25˚C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25˚C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy,	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		°C
LM35, LM35C	T_A=−10°C	±0.5			±0.5		±1.5	°C
(Note 7)	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	°C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D	T_A=+25°C				±0.6	±1.5		°C
(Note 7)	T_A=T_{MAX}				±0.9		±2.0	°C
	T_A=T_{MIN}				±0.9		±2.0	°C
Nonlinearity	T_{MIN}≤T_A≤T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
(Note 8)								
Sensor Gain	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°C
(Average Slope)								
Load Regulation	T_A=+25°C	±0.4	±2.0		±0.4	±2.0		mV/mA
(Note 3) 0≤I_L≤1 mA	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation	T_A=+25°C	±0.01	±0.1		±0.01	±0.1		mV/V
(Note 3)	4V≤V_S≤30V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current	V_S=+5V, +25°C	56	80		56	80		µA
(Note 9)	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25°C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of	4V≤V_S≤30V, +25°C	0.2	2.0		0.2	2.0		µA
Quiescent Current	4V≤V_S≤30V	**0.5**		**3.0**	**0.5**		**3.0**	µA
(Note 3)								
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Note 1: Unless otherwise noted, these specifications apply: −55°C≤T_J≤+150°C for the LM35 and LM35A; −40°≤T_J≤+110°C for the LM35C and LM35CA; and 0°≤T_J≤+100°C for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2°C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

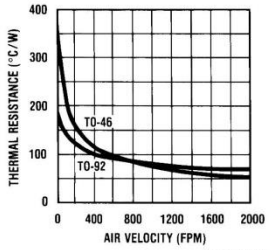
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

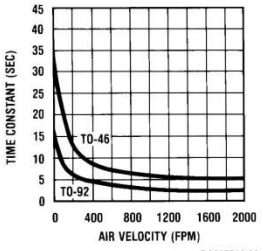
Typical Performance Characteristics

Thermal Resistance Junction to Air



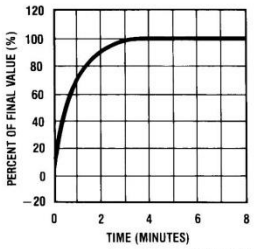
DS005516-25

Thermal Time Constant



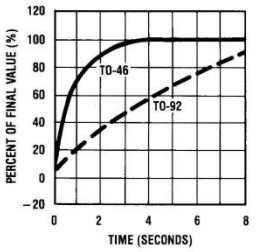
DS005516-26

Thermal Response in Still Air



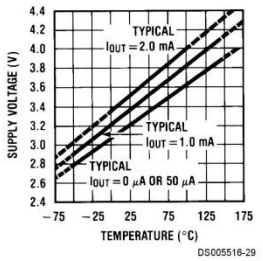
DS005516-27

Thermal Response in Stirred Oil Bath



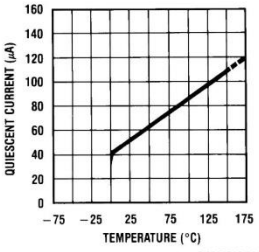
DS005516-28

Minimum Supply Voltage vs. Temperature



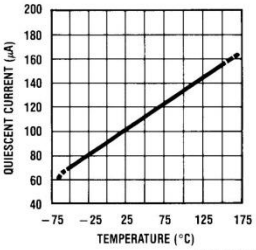
DS005516-29

Quiescent Current vs. Temperature (In Circuit of *Figure 1*.)



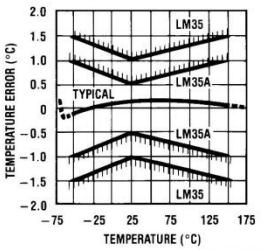
DS005516-30

Quiescent Current vs. Temperature (In Circuit of *Figure 2*.)



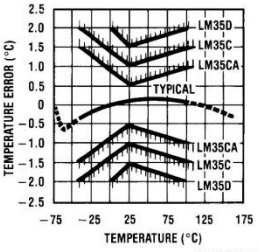
DS005516-31

Accuracy vs. Temperature (Guaranteed)



DS005516-32

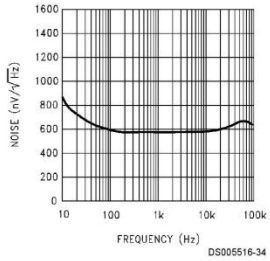
Accuracy vs. Temperature (Guaranteed)



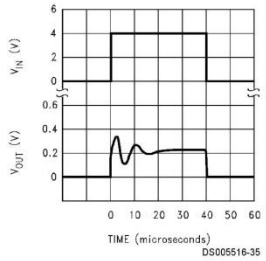
DS005516-33

Typical Performance Characteristics (Continued)

Noise Voltage



Start-Up Response



Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01˚C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is expecially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			
(Clamped to metal, Infinite heat sink)	(24°C/W)				(55°C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

Typical Applications

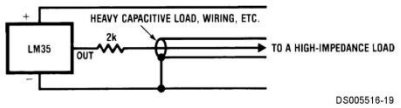


FIGURE 3. LM35 with Decoupling from Capacitive Load

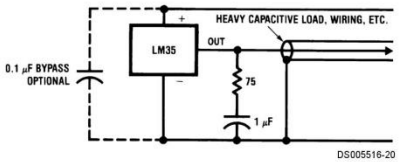


FIGURE 4. LM35 with R-C Damper

CAPACITIVE LOADS

Like most micropower circuits, the LM35 has a limited ability to drive heavy capacitive loads. The LM35 by itself is able to drive 50 pf without special precautions. If heavier loads are anticipated, it is easy to isolate or decouple the load with a resistor; see *Figure 3*. Or you can improve the tolerance of capacitance with a series R-C damper from output to ground; see *Figure 4*.

When the LM35 is applied with a 200Ω load resistor as shown in *Figure 5*, *Figure 6* or *Figure 8* it is relatively immune to wiring capacitance because the capacitance forms a bypass from ground to input, not on the output. However, as with any linear circuit connected to wires in a hostile environment, its performance can be affected adversely by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, SCR transients, etc, as its wiring can act as a receiving antenna and its internal junctions can act as rectifiers. For best results in such cases, a bypass capacitor from V_{IN} to ground and a series R-C damper such as 75Ω in series with 0.2 or 1 μF from output to ground are often useful. These are shown in *Figure 13*, *Figure 14*, and *Figure 16*.

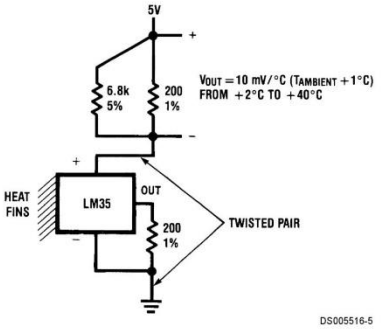


FIGURE 5. Two-Wire Remote Temperature Sensor (Grounded Sensor)

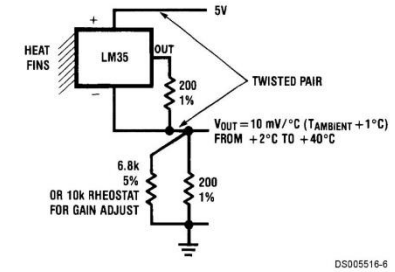


FIGURE 6. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

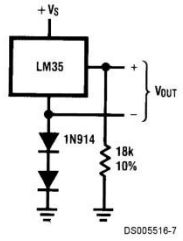


FIGURE 7. Temperature Sensor, Single Supply, −55˚ to +150˚C

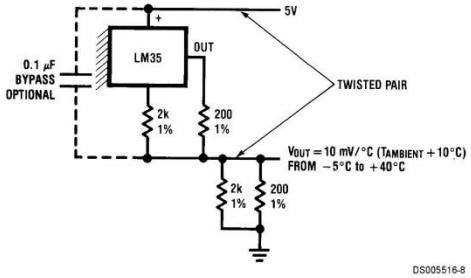


FIGURE 8. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

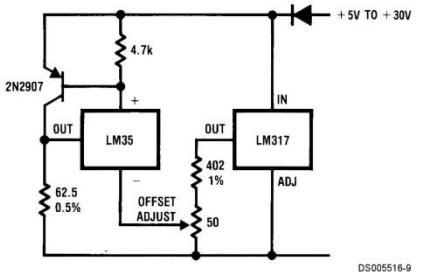


FIGURE 9. 4-To-20 mA Current Source (0˚C to +100˚C)

Typical Applications (Continued)

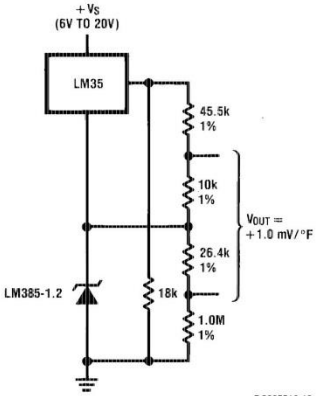


FIGURE 10. Fahrenheit Thermometer

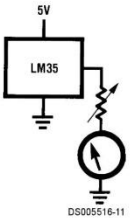


FIGURE 11. Centigrade Thermometer (Analog Meter)

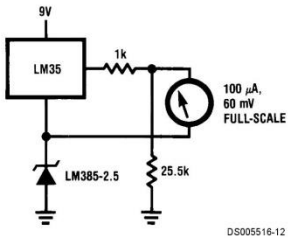


FIGURE 12. Fahrenheit ThermometerExpanded Scale Thermometer
(50˚ to 80˚ Fahrenheit, for Example Shown)

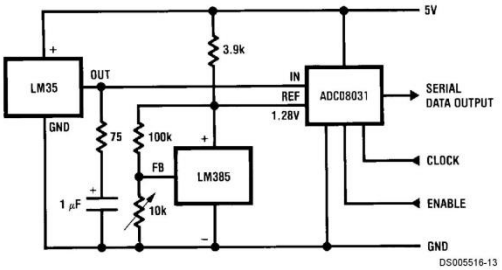


FIGURE 13. Temperature To Digital Converter (Serial Output) (+128˚C Full Scale)

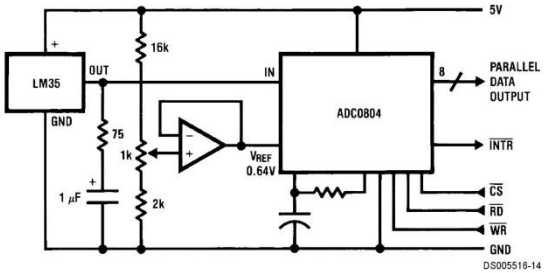
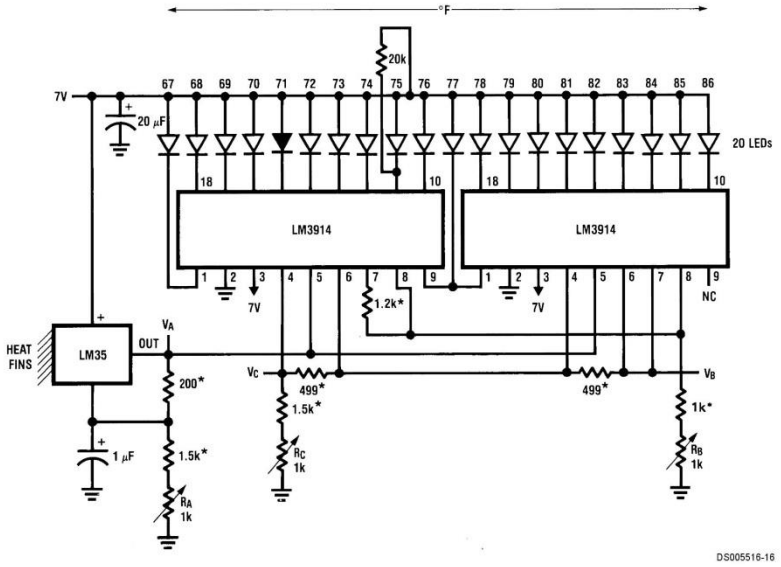


FIGURE 14. Temperature To Digital Converter (Parallel TRI-STATE™ Outputs for Standard Data Bus to µP Interface) (128˚C Full Scale)

Typical Applications (Continued)



*=1% or 2% film resistor
Trim R_B for V_B=3.075V
Trim R_C for V_C=1.955V
Trim R_A for V_A=0.075V + 100mV/°C x $T_{ambient}$
Example, V_A=2.275V at 22°C

FIGURE 15. Bar-Graph Temperature Display (Dot Mode)

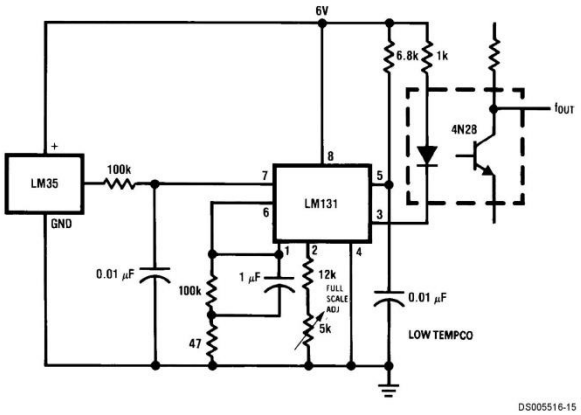
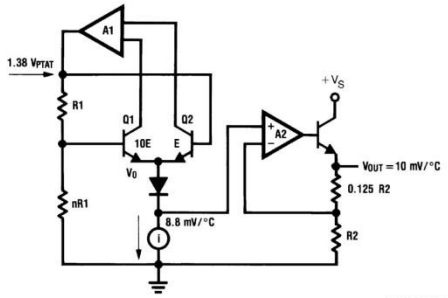


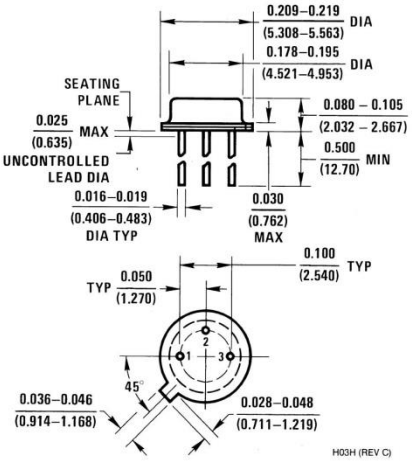
FIGURE 16. LM35 With Voltage-To-Frequency Converter And Isolated Output (2°C to +150°C; 20 Hz to 1500 Hz)

Block Diagram

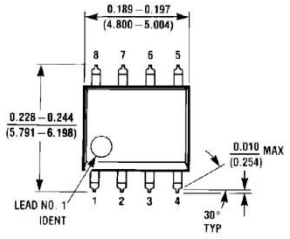


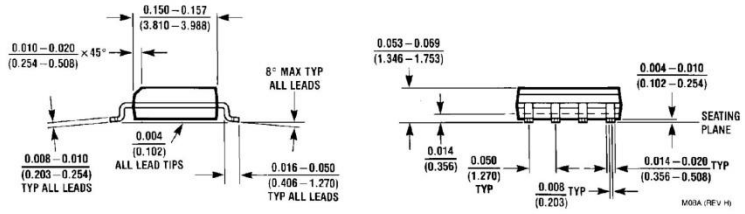
DS005516-23

Physical Dimensions inches (millimeters) unless otherwise noted



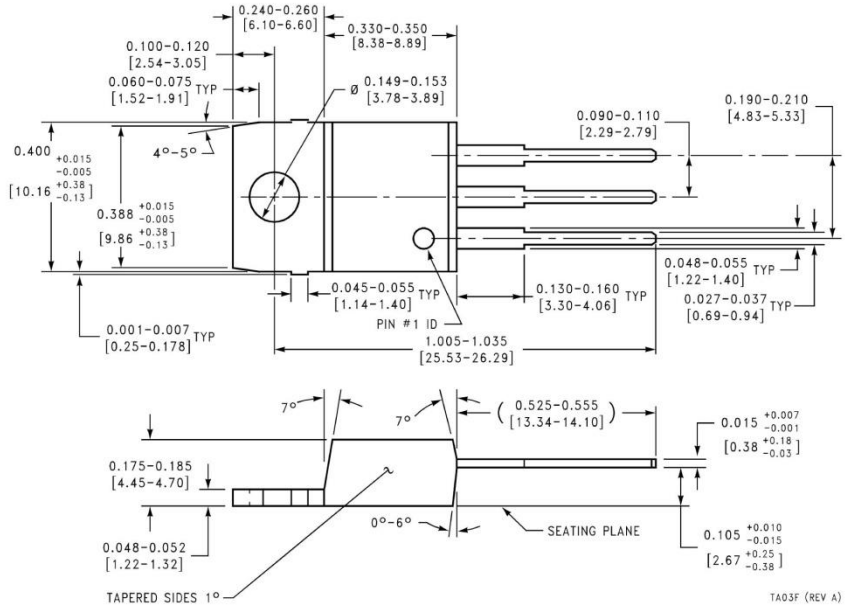
TO-46 Metal Can Package (H)
Order Number LM35H, LM35AH, LM35CH, LM35CAH, or LM35DH
NS Package Number H03H





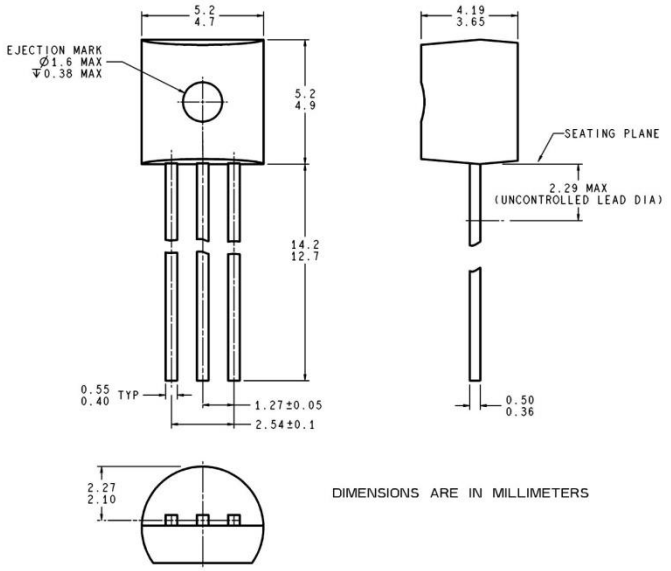
SO-8 Molded Small Outline Package (M)
Order Number LM35DM
NS Package Number M08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



Power Package TO-220 (T)
Order Number LM35DT
NS Package Number TA03F

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



TO-92 Plastic Package (Z)
Order Number LM35CZ, LM35CAZ or LM35DZ
NS Package Number Z03A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265