Curso: Bacharelado em Engenharia da Computação

Disciplina: Técnicas de prototipagem

Professor: Alexandre Sales Vasconcelos

Grupo 6: Amanda Moura Camilo (201811250004)

Josehylton Ricardo Pereira Neves (201921250021)

Rafael Victor Cordeiro Muniz (201921250022)

Projeto 06 - Sistemas de automação energia solar

Campina Grande, Agosto/2022

Sumário

Introdução 3

Objetivo 3

Descrição 3

Figura 1: Esquema de comunicação ESP - Telegram 4

Requisitos 4

Materiais e Métodos 4

Atividades Executadas 4

Lista de Materiais 4

Components: 5

Sensor de temperatura (DS18B20) 5

Figura 2: Sensor de temperatura DS18B20 5

Figura 3: Diagrama de bloco sensor de temperatura 6

Driver para comando elétrico (JQC3F) 6

Figura 4: Driver JQC3F 6

Sensor de tensão AC (4N25) 7

Figura 5: Sensor de tensão AC (4N25) 7

Sensor de tensão e corrente DC (ACS712) 7

Figura 6: Sensor de tensão e corrente DC (ACS712) 8

Diagrama de bloco: 9

Figura 7: Diagrama de bloco sensor de corrente DC (ACS712) 9

ESP32 9

Figura 8: ESP32 9

Conversor Nível Lógico (CJMCU TXS0108E) 11

Figura 10: Conversor nível lógico (CJMCU TXS0108E) 11

Figura 11: Diagrama de blocos do conversor nível lógico (CJMCU TXS0108E) 12

Bibliotecas Utilizadas 12

Resultados 13

Conclusão 14

Repositório 14

Links dos Datasheets 14

1. Introdução

Atualmente, o mundo tem se voltado para a percepção de sustentabilidade e a utilização de energias renováveis têm um papel fundamental no processo de mudança para um mundo cada vez mais sustentável.

Assim sendo, sistemas como os de energia solar têm sido amplamente utilizados, pelo fato de servir-se de um recurso natural, que abastece grande parte do globo terrestre para a geração de energia elétrica de maneira abundante.

Assim sendo, os sistemas de energia solar off-grid têm como característica principal o “autossustento”, sendo capazes de produzir energia elétrica através de sua exposição aos raios solares, e de armazenar a energia solar excedente em baterias para que possam ser utilizadas em dias que não há produção de energia.

2. Objetivo

Visando o funcionamento e características do sistema de energia solar off-grid, nosso projeto tem como objetivo a realização de um sistema de automação que irá enviar mensagens através da plataforma do Telegram, indicando o status das baterias que armazenam a energia excedente e a porcentagem de energia armazenada nelas.

3. Descrição

Como o objetivo principal do nosso projeto é o envio de mensagens indicando o status e porcentagem de energia armazenada na bateria acoplada ao sistema de produção de energia solar, utilizaremos para esses fins, além dos componentes de hardware que serão listados no item posterior, o software NodeRed e a plataforma AWS.

Os softwares estão linkados da seguinte maneira em nosso projeto:

O microcontrolador utilizado no projeto, está conectado ao Topic do Broken MQTT que é um cliente AWS - MQTT é um protocolo de comunicação entre o publisher e o subscriber do ESP 32 que faz a comunicação entre o ESP 32 e o NodeRed.

O Node-RED tem a função de intermediar a comunicação entre emissor ESP 32 e receptor Telegram (ou vice-versa) direcionando as mensagens. Ele está conectado ao Broken MQTT e direciona as mensagens do Topic MQTT para o Telegram ou vice-versa, auxiliando a troca de mensagens entre ESP e Telegram, como mostrado na imagem abaixo.

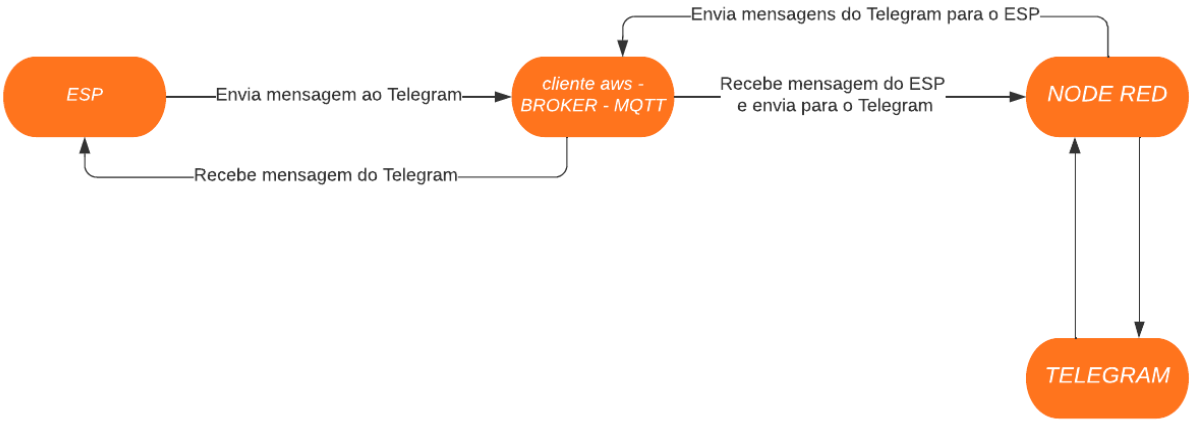


Figura 1: Esquema de comunicação ESP - Telegram

4. Requisitos

- Informações do estado da rede elétrica (falta de energia) e da carga da bateria;
- Gera alarmes e enviar pelo Telegram quando ocorrer falta de energia na rede elétrica e carga baixa das baterias;
- Controle remoto do contator (Telegram -> Sistema embarcado).

5. Materiais e Métodos

5.1. Atividades Executadas

Primeiramente foi feita a compra dos componentes necessários para realização do projeto. Após a chegada de todos eles, foi feita a testagem de cada componente utilizando a IDE Visual Studio Code e por meio de bibliotecas feitas e já existentes na linguagem C. Depois da confirmação de como seria a utilização de cada componente individualmente, foram feitas estratégias para a junção das funções dos componentes em busca de atingir o objetivo proposto para o projeto. Para a função de recebimento e envio de mensagens também foi utilizado o aplicativo mensageiro Telegram e o auxílio do Node-RED, que serviu como um comunicador/intermediador entre o ESP32 e o Telegram.

5.2. Lista de Materiais

- Sensor de falta de energia (rede elétrica) (cross zero);

- Sensor de tensão DC (divisor de tensão)
- Sensor de corrente DC (acs712)
- Sensor de temperatura para baterias (DS18B20),
- Jumpers macho-macho,
- Relé,
- Conversor de sinal bidirecional,
- Resistores 4k7, 200k
- Fonte de alimentação 5V,
- ESP32.

5.3. Components:

- Sensor de temperatura (DS18B20)

Figura 2: Sensor de temperatura DS18B20

Para o sensor de temperatura foi utilizado um termômetro DS18B20. É um sensor de temperatura digital indicado para a aplicações com faixa de temperatura entre -55°C até +125°C (-67°F até +257°F) com precisão de ±0.5°C em -10°C até +85°C que atende os requisitos de temperatura do sistema de automação energia solar off-grid, além de ser um componente de baixo custo, e de fácil disponibilidade no mercado.

Fonte de alimentação	3V a 5,5V
Consumo de corrente	1mA
Faixa de temperatura	-55 a 125°C
Precisão	±0,5°C
Resolução	9 a 12 bits (selecionável)
Tempo de conversão	< 750 ms

Diagrama de bloco:

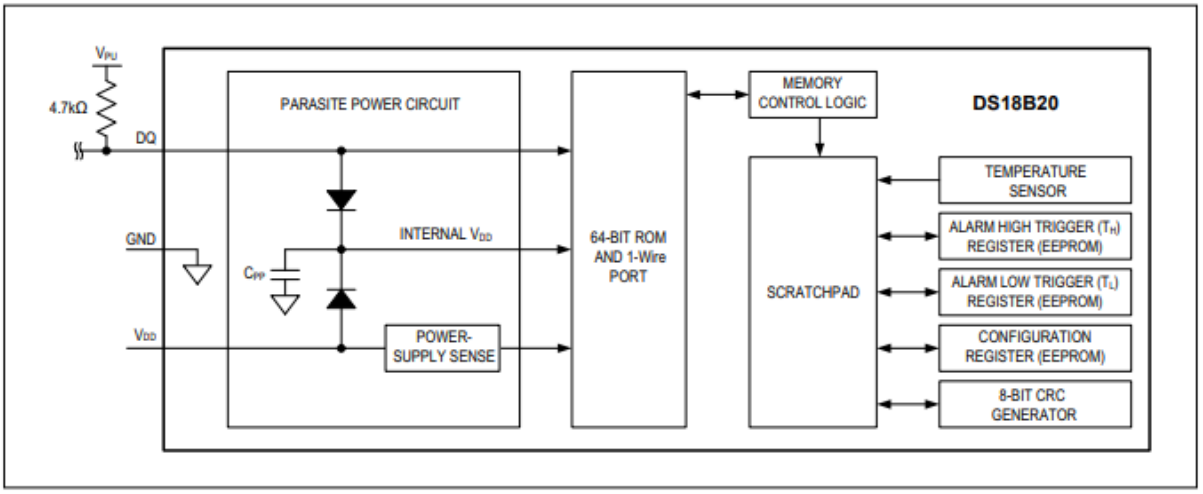


Figura 3: Diagrama de bloco sensor de temperatura

- Driver para comando elétrico (JQC3F)

Figura 4: Driver JQC3F

O Módulo Relé 5V JQC3F é um compacto módulo de acionamento que permite integração com um grande número de sistemas microcontroladores, dentre estes: Arduino, AVR, PIC, ARM, Raspberry PI, etc.

Por meio desta placa de acionamento (módulo relé 5V) é possível controlar 1 dispositivo de corrente alternada, de até 10A, como, por exemplo, lâmpadas, portões eletrônicos, ventiladores, etc.

Modelo	JQC3F-C
Pinos	SIG/VCC/GND

Carga nominal	10A 250VAC/ 10A 125VAC/ 10A 30VDC/ 10A 28VDC
Tensão de operação	5VDC (VCC)
Tensão de sinal	TTL 5VCD
Corrente por canal	até 10A

- Sensor de tensão AC (4N25)

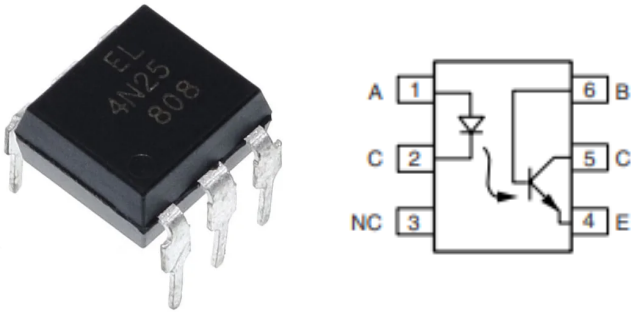
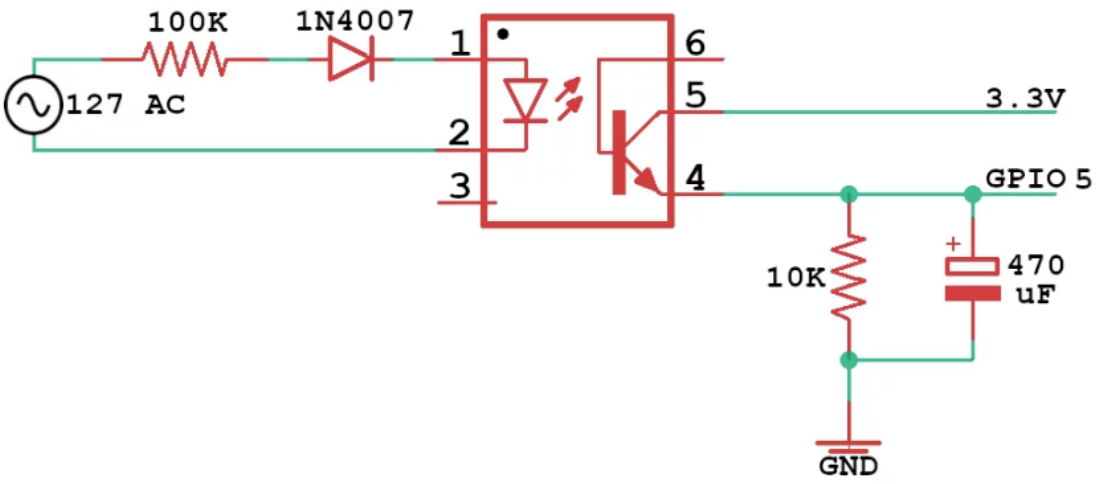


Figura 5: Sensor de tensão AC (4N25)

A família 4N25 é um acoplador de fototransistor de canal único padrão da indústria. Esta família inclui o 4N25, 4N26, 4N27, 4N28. Cada optoacoplador consiste em um LED infravermelho de arsenieto de gálio e um fototransistor NPN de silício. Uma de suas aplicações é detecção de rede AC, e é para o propósito de detecção de falta de energia que é utilizado nesse projeto.

Temperatura de operação	-55°C até 100°C
Voltagem direta máxima	1,5 V
Corrente reversa máxima	100 µA

Diagrama eletrico:



- Sensor de tensão e corrente DC (ACS712)

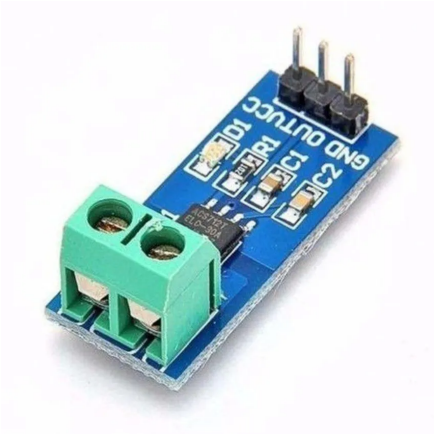


Figura 6: Sensor de tensão e corrente DC (ACS712)

O ACS712 oferece soluções econômicas e precisas para detecção de corrente AC ou DC em sistemas industriais, comerciais e de comunicação. No projeto ele tem a utilização de detectar o tipo de tensão e corrente de entrada no momento.

Temperatura de operação	-40°C até 85°C
Máxima temperatura de junção	165°C
Tensão de Alimentação máxima	8V

Tensão de Alimentação comum	5.5V

Diagrama de bloco:

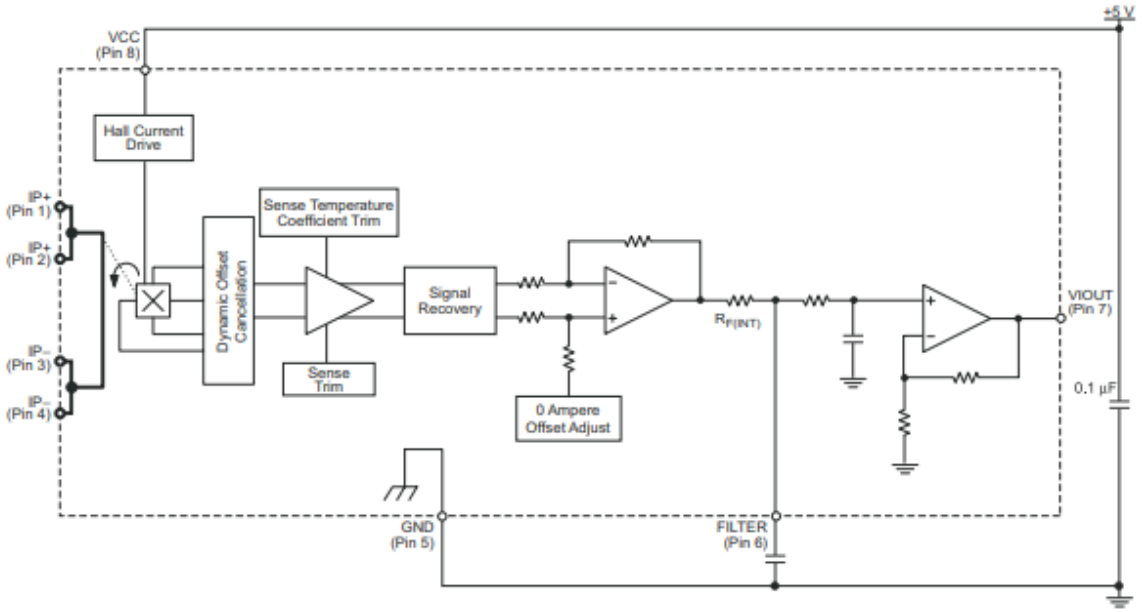


Figura 7: Diagrama de bloco sensor de corrente DC (ACS712)

- ESP32

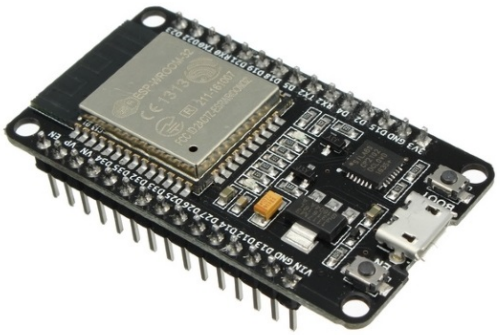


Figura 8: ESP32

O ESP32 é um dispositivo ioT (Internet das Coisas) que consiste de um microprocessador de baixa potência dual core Tensilica Xtensa 32-bit LX6 com suporte embutido à rede WiFi, Bluetooth v4.2 e memória flash integrada.

Essa arquitetura permite que ele possa ser programado de forma independente, sem a necessidade de outras placas microcontroladoras como o Arduino, por exemplo.

Dentre as principais características deste dispositivo, podemos citar: baixo consumo de energia, alto desempenho de potência, amplificador de baixo ruído, robustez, versatilidade e confiabilidade.

MCU	Dual core Tensilica Xtensa 32-bit LX6
Bluetooth	4.2
Frequência típica	160MHz
SRAM	512kBytes
GPIO	36
Hardware/Software PWM	1/16 Canais
ADC	12-bit
Temperatura de trabalho	-40°C até 125°C

Diagrama de blocos:

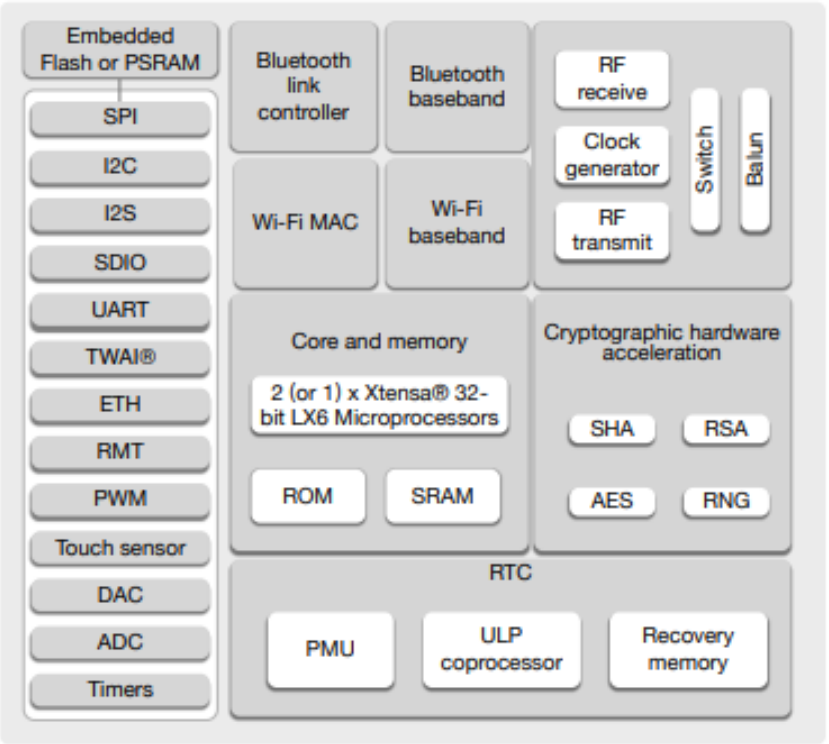


Figura 9: Diagrama de blocos ESP32

5.4. Componentes criados/utilizados

- all_components_inteface
- ac_voltage_sensor
- acs712
- dc_voltage_sensor
- ds18b20
- onewire_ds18b20
- ds18b20_interface
- mqtt
- mqtt_implement
- power_supply_driver
- wifi

5.5. Especificações dos components

- **all_components_interface** = é um componente criado para uso do “main.c” para todas as bibliotecas.

- **ac_voltage_sensor** = componente criado para implementar a verificação de tensão elétrica da rede e seu métodos acessíveis são:

1. void init_ac_volltage_sensor() inicia as configurações do mesmo;
2. bool read_ac_voltage_sensor() verifica se existe tensão na rede;

- **acs712** = componente criado para verificar a intensidade da corrente da bateria, seus métodos são:
    1. void init_acs712_sensor() inicia as configurações do mesmo;
    2. float read_acs712_sensor() realiza a medição da intensidade da corrente de elétrica que circula no circuito da bateria;
    3. void calibrate_sensor() utilizado para calibrar o valor da corrente eletrica (método privado);

- **dc_voltage_sensor** = componente criado para medir a tensão da bateria, seus metodos são:
    1. void init_dc_voltage_sensor() inicia as configurações do mesmo;
    2. float read_dc_voltage_sensor() realização a medição da tensão da bateria;

- **ds18b20** = componente externo utilizado para realizar a leitura/conversão do sensor de temperatura.

- **onewire_ds18b20** = componente externo utilizado para implementar o protocolo de comunicação do sensor ds18b20;

- **ds18b20_interface** = componente criado para utilizar os metodos necessários para realizar a medição de temperatura utilizando os componentes ds18b20 e onewire_ds18b20, seus métodos são:
    1. void init_ds18b20_sensor() implementa o protocolo onewire e inicia o sensor ds18b20;
    2. float read_ds18b20_sensor() realiza a leitura do sensor ds18b20;

- **mqtt** = componente modulo do framework esp-idf.

- **mqtt_implement** = componente criado para implementar o protocolo de comunicação mqtt para realizar trocas de mensagens (publisher/consumer), seus métodos são:
    1. void mqtt_start() inicializa o mqtt;
    2. void send_message(char* message) recebe uma mensagem para ser publicada
    3. void send_message_to_topic(char* topic, char* message) método para enviar mensagem para um topic (método privado);
    4. static void mqtt_event_handler() metodo criado para tratar eventos;
    5. static esp_err_t mqtt_event_handler_cb() método criado apra aulizar o método tratador de eventos;

- **wifi** = módulo ajustado para conectar no wifi, seus métodos são.
    1. void wifi_start() metodo realizado para conectar no wifi;
    2. static void event_handler() método criado para tratar eventos;

5.6. Configurações do projeto

Os sensores e as conexões de MQTT e WIFI podem ser configuradas no menuconfig do esp “idf.py menuconfig” digite o comando sem aspas no esp-idf terminal ou selecione a opção “menuconfig” caso esteja usando o VSCode

6. Resultados

Ao final do nosso projeto, conseguimos concluir nosso principal objetivo que é a troca de informações entre o sistema embarcado e outros servidores, através da intermediação de servidores remotos.

Não conseguimos finalizar a interface de controle, entretanto, o objetivo principal, fazer o controle da temperatura e a transmissão da mensagem, foram concluídos com êxito.

7. Conclusão

Concluímos que, ao final do projeto, aprendemos mais acerca do uso do microcontrolador ESP 32, uso de sensores analógicos e digitais, tratamento de dados, envio e recebimento de mensagens direcionadas por outro software, além do uso de serviços remotos de cliente e servidor.

8. Repositório

Para mais consultas, segue o link do repositório do nosso projeto de sistema de automação energia solar off-grid no GitHub:

https://github.com/riccckneves/projeto-embarcados.git

9. Links dos Datasheets

- Sensor de temperatura (DS18B20)
- Driver para comando elétrico - contator
- Sensor de falta de energia (4N25)
- Sensor de corrente DC (ACS712)
- ESP32