

Universidade Católica de Pelotas (UCPel)
Centro de Ciências Sociais e Tecnológicas
Engenharia de Computação

Gerenciamento de Embarcados com Recurso de OTA Telegram

Aluno: Lucas Kerstner Penning
Professor: Adenauer Yamin

Pelotas
2020

Universidade Católica de Pelotas (UCPel)
Centro de Ciências Sociais e Tecnológicas
Engenharia de Computação

Gerenciamento de Embarcados com Recurso de OTA Telegram

Relatório do projeto apresentado ao Centro de Ciências Sociais e Tecnológicas do Curso de Engenharia de Computação da Universidade Católica de Pelotas (UCPel), como requisito para aproveitamento e conclusão da disciplina de Engenharia de Computação.

Aluno: Lucas Kerstner Penning

Professor: Adenauer Yamin

Pelotas
2020

Conteúdo

1	Introdução	1
2	Levantamento das Informações	1
3	Justificativa da Proposta	2
4	Funcionamento Detalhado	2
5	Diagrama UML	4
6	Entradas	4
7	Saídas	4
8	Especificações de Materiais	5
8.1	Microcontrolador(Embarcado):ESP32	5
8.2	Computador(Servidor API Telegram):PC	6
8.3	Alicativo Telegram	6
8.4	Broker MQTT	7
8.5	GitHub	7
8.6	UpyCraft	7
8.7	Sensor de temperatura e umidade: DHT11	8
8.8	Pyhton	9
8.9	MicroPyhton	9
9	Desenvolvimento	10
10	Conclusão	13
11	Referencias	13
A		13

1 Introdução

Neste relatório é apresentado um sistema que realiza o controle das ações da ESP32, que é um microcontrolador onde o projeto apresentado consiste em manter os códigos contidos sempre atualizados assim que surgirem novas atualizações, usufruindo do recurso OTA(Over-The-Air) juntamente com o Telegram que é um serviço de mensagens instantâneas, onde serão mandados comandos específicos por meio do Telegram e o embarcado irá responder conforme o comando solicitado ou retornará algum erro. Além disso será implementado rotina de atualizações OTA, após ser reiniciado ou caso o embarcado permaneça muito tempo ligado. Também irá ser empregado sensor para monitorar temperatura e umidade a fins de demonstração para funcionalidades.

O MQTT (Message Queue Telemetry Transport) é utilizado para a comunicação entre a API e embarcado, API que está presente em uma máquina que cumpre o papel de servidor através da medição do software DIoTY onde fica localizado o broker, além do NTP(Network Time Protocol) que auxilia para mantermos data e horário no tempo correto evitando problema de atualizações e sensoramento.

2 Levantamento das Informações

Após pesquisar sobre o assunto, foi realizado a descoberta de diversos projetos e documentos sobre a questão de atualizações OTA e também sobre a API Telegram, onde foram observados diversos aspectos entre esses assuntos, porém não se encontrou muito em questão da junção dos mesmo funcionando e cooperando entre si, então assim surgiu a ideia de implementar para funcionarem em sincronia, assim podendo verificar o status do embarcado em relação a atualizações a partir da aplicação Telegram.

Todos os códigos contidos no embarcado ESP32 foram desenvolvidos na linguagem de programação Micropython juntamente com o firmware da Micropython. Utilizando as portas do pino 19, GND e 3V3 se deu a implementação do sensoramento de temperatura e umidade com o sensor DHT11, ocorreu a utilização do pino 18 com GND para o LED amarelo de status de atualização, além da função WIFI contido no próprio embarcado para o funcionamento dos protocolos OTA, MQTT e NTP.

Já na parte do servidor o código foi desenvolvido inteiramente na linguagem de programação Pyhthon.

3 Justificativa da Proposta

Esse projeto tem como objetivo otimizar o acompanhamento das atualizações de códigos por meio do recurso OTA, assim como também a visualização do sensoramento de temperatura e umidade das últimas 24 horas ou do momento desejado, além de logs do embarcado e ajustes práticos sobre o tempo de hora e data, tudo isso por meio da aplicação Telegram.

4 Funcionamento Detalhado

O conjunto de software pode ser dividido em três partes: códigos em Mi-cropython programados para realizar comandos no embarcado ESP32, API Telegram implementada em Python presente em uma máquina que cumpre o papel de servidor para realizar a intermediação entre embarcado-Telegram, além do aplicativo Telegram que pode estar em um smartphone ou desktop por onde iremos enviar os comandos pré-definidos a serem cumpridos pelo embarcado ESP32.

No embarcado teremos presentes os protocolos mencionados acima(OTA, MQTT e NTP), ao ser ligado ou reiniciado o software irá automaticamente realizar o ajuste do tempo através do NTP, além de atualizar os códigos automaticamente por meio do repositório contido no GitHub caso ocorrer alguma atualização através do OTA. Foi também definido um determinado horário para o embarcado se reiniciar automaticamente evitando assim permanecer por muito tempo desatualizado ou com tempo de hora e data errado. A comunicação entre embarcado e API Telegram se dá por meio de um broker MQTT.

Na API Telegram temos presentes bibliotecas fornecidas pelo Telegram próprias para esse tipo de implementação na comunicação entre API e aplicação Telegram, além do protocolo MQTT mencionado anteriormente para comunicação com o embarcado.

Ao iniciar o aplicativo Telegram, entrando no chat pela primeira devemos nos registrar como um novo usuário, para podermos ter acesso ao embarcado assim podendo realizar solicitações de tarefas, iremos ter diversos botões com comando pré-definidos, no qual ao dar o click devemos aguardar 10 segundos para obter a resposta do comando enviado ou de algum possível erro.

Será realizado o monitoramento do embarcado pelo Telegram através das funcionalidades descritas abaixo, além de comandos com tarefas para serem realizados pelo embarcado obtendo retorno de valores e status avisando se ocorreu tudo bem nas realizações das mesmas. Com a implementação do OTA sempre que um nova atualização sair, ao ser reiniciado o embarcado irá

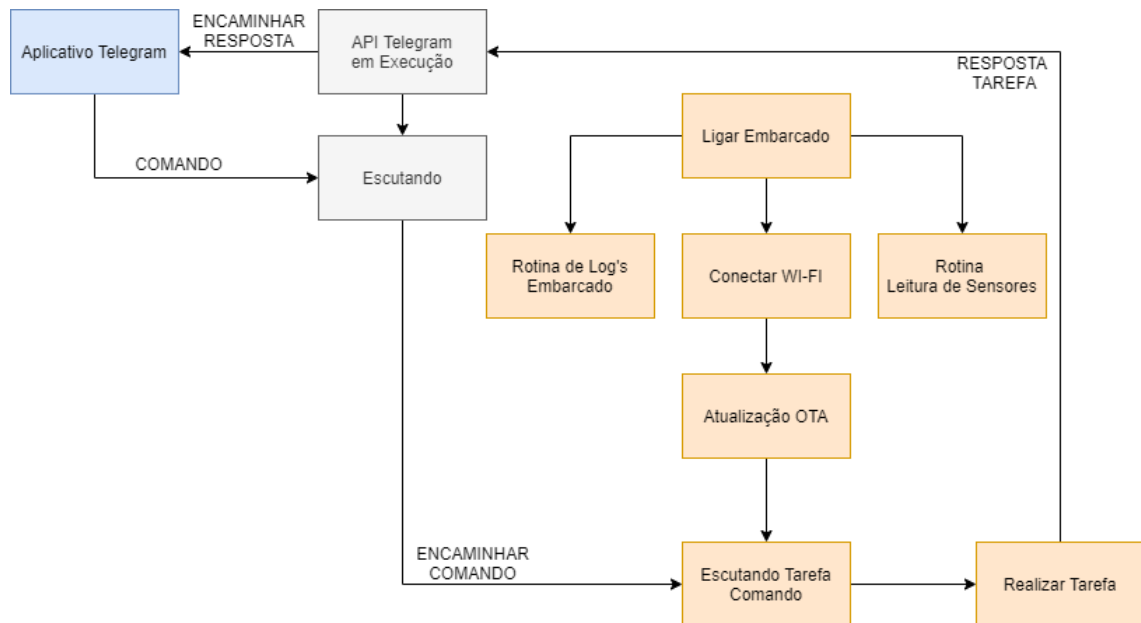
se encarregar de fazer automaticamente, contamos também com uma rotina de verificações que manterá o embarcado sempre atualizado caso se faça a necessidade de seu funcionamento constantemente, ou também disparando um comando de atualização pelo Telegram. Teremos os valores do sensor pretendido sempre que desejado rapidamente.

Operações:

- Uptime: tempo em atividade do embarcado;
- Version: verifica a versão do software do embarcado;
- Sensores: sensores instalados no embarcado;
- Horário: retorna a hora que está no embarcado;
- Temperatura: retorna temperatura ambiente em graus;
- Umidade: retorna à Umidade ambiente;
- Atualizar: embarcado irá se atualizar imediatamente;
- Reiniciar: embarcado será reiniciado;
- Serão enviadas a cada X minutos as medições do sensores para o BoT Telegram
- Por padrão $X = 30$ minutos, podendo ser configurado um novo valor por mensagem
- Registrar no embarcado o histórico das atualizações: versão e data
- Registrar no embarcado um log das últimas 24h das medições feitas pelos sensores
- Por solicitação será enviado o log das últimas 24h das medições feitas
- Por solicitação será enviado o histórico das atualizações

5 Diagrama UML

Figura 1



6 Entradas

- Sensor de temperatura;
- Sensor de umidade;
- Comandos por Telegram;

7 Saídas

- Realização de rotinas de atualização;
- Realização das atividades através de comandos por Telegram;
- Logs do embarcado;
- Envio de dados solicitados.

8 Especificações de Materiais

8.1 Microcontrolador(Embarcado):ESP32

- CPU: Xtensa® Dual-Core 32-bit LX6;
- Memória ROM: 448 KBytes;
- Clock máximo: 240MHz;
- Memória RAM: 520 Kbytes;
- Memória Flash: 4 MB;
- Wireless padrão 802.11 b/g/n;
- Conexão Wifi de 2.4Ghz (máximo de 150 Mbps);
- Antena embutida na placa;
- Conector micro USB para comunicação e alimentação;
- Wi-Fi Direct (P2P), P2P Discovery, P2P Group Owner mode e P2P Power Management;
- Modos de operação: STA/AP/STA+AP;
- Bluetooth BLE 4.2;
- Portas GPIO: 11;
- GPIO com funções de PWM, I2C, SPI, etc;
- Tensão de operação: 4,5 – 9V;
- Conversor analógico digital (ADC).



[EPS32]

8.2 Computador(Servidor API Telegram):PC

- Windows 10;
- Processador quad core;
- 4 GB RAM;
- 500 GB HDD;



[PC]

8.3 Alicativo Telegram

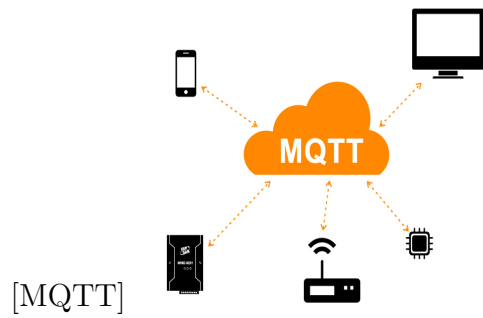
- Desktop ou Mobile;



[Telegram]

8.4 Broker MQTT

- DIoTY, Mosquitto, etc.;



8.5 GitHub

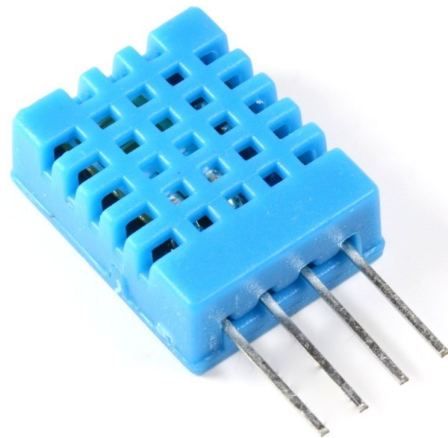


8.6 UpyCraft



8.7 Sensor de temperatura e umidade: DHT11

- Dimensões: 23mm x 12mm x 5mm (incluindo terminais)
- Alimentação: 3,0 a 5,0 VDC (5,5 Vdc máximo)
- Corrente: 200uA a 500mA, em stand by de 100uA a 150 uA
- Faixa de medição de umidade: 20 a 90 por cento
- Faixa de medição de temperatura: 0° a 50°C
- Precisão de umidade de medição: $\pm 5,0$ por cento
- Precisão de medição de temperatura: ± 2.0 °C
- Tempo de resposta: menor que 5s
- Pino 1: Alimentação - 3,0 a 5,0 VDC;
- Pino 2: Saída Data;
- Pino 3: Não é utilizado;
- Pino 4: GND – 0V.



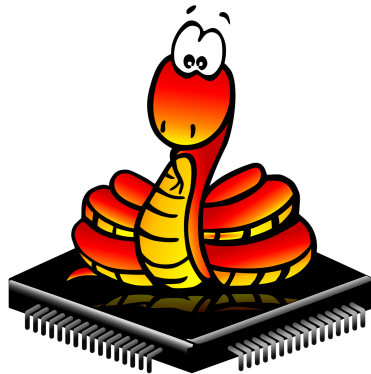
[DHT11]

8.8 Pyhton



[Pyhton]

8.9 MicroPyhton



[MicroPyhton]

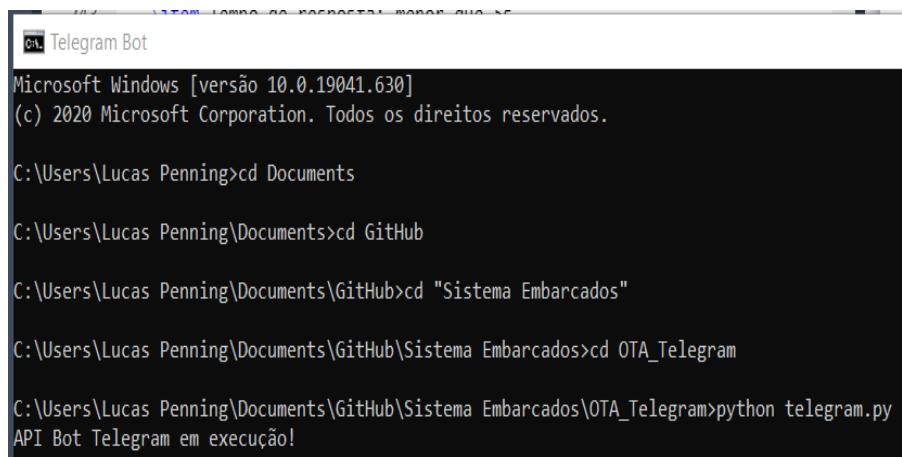
9 Desenvolvimento

Inicialmente devemos abrir o aplicativo Telegram e entrar em contato com o BotFather, onde iremos fazer a configuração do chat, após terminar as mesmas iremos ter um "TOKEN" do chat, que será usado para configurar o arquivo "config.ini" conforme figura 1:

```
[TELEGRAM]
API_TOKEN = SEU_TOKEN_TELEGRAM
ADMIN = SEU_ID_USUARIO_TELEGRAM
TRUSTED_USERS = ID_USUARIO_TELEGRAM
```

[Figura 1]

Logo após devemos configurar as informações do MQTTClient presentes no arquivo "telegram.py" para o destino Broker MQTT configurado por você. Após isso podemos colocar o servidor em execução, para isso é necessário ter o Python instalado na máquina que terá o papel de servidor, abrindo o prompt de comando, indo até o local onde está o arquivo "telegram.py" e executando o seguinte comando: "python telegram.py", obtendo como resultado a mensagem "API Bot Telegram em execução" como na figura 2:



```
Microsoft Windows [versão 10.0.19041.630]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Lucas Penning>cd Documents

C:\Users\Lucas Penning\Documents>cd GitHub

C:\Users\Lucas Penning\Documents\GitHub>cd "Sistema Embarcados"

C:\Users\Lucas Penning\Documents\GitHub\Sistema Embarcados>cd OTA_Telegram

C:\Users\Lucas Penning\Documents\GitHub\Sistema Embarcados\OTA_Telegram>python telegram.py
API Bot Telegram em execução!
```

[Figura 2]

No GitHub devemos criar um repositório com versionamento ou "releases", passando para o embarcado o arquivo que deve ser configurado é o arquivo JSON, sendo alterado o SSID e PASSWORD da rede, além do link do repositório do GitHub desejado, conforme figura 3:

```

1  {
2    "network": {
3      "ssid": "SSID",
4      "password": "SENHA"
5    },
6    "git": {
7      "url": "LINK_REPOSITORIO_GITHUB",
8      "dir": "src"
9    }
10  }
11
12 }

```

[Figura 3]

Próximo passo será alterar as informações do MQTTClient presentes no arquivo boot.py para o destino Broker MQTT configurado por você, conforme figura 4:

```

#Configurar o Broker MQTT
mq = MQTTClient("AtualizacaoOTA","broker",1883,"usuario","senha")
mq.set_callback(sub_cb)

```

[Figura 4]

Além de alterar o tópico para o seu tópico em todas as linhas com a característica da figura 5, no arquivo "boot.py":

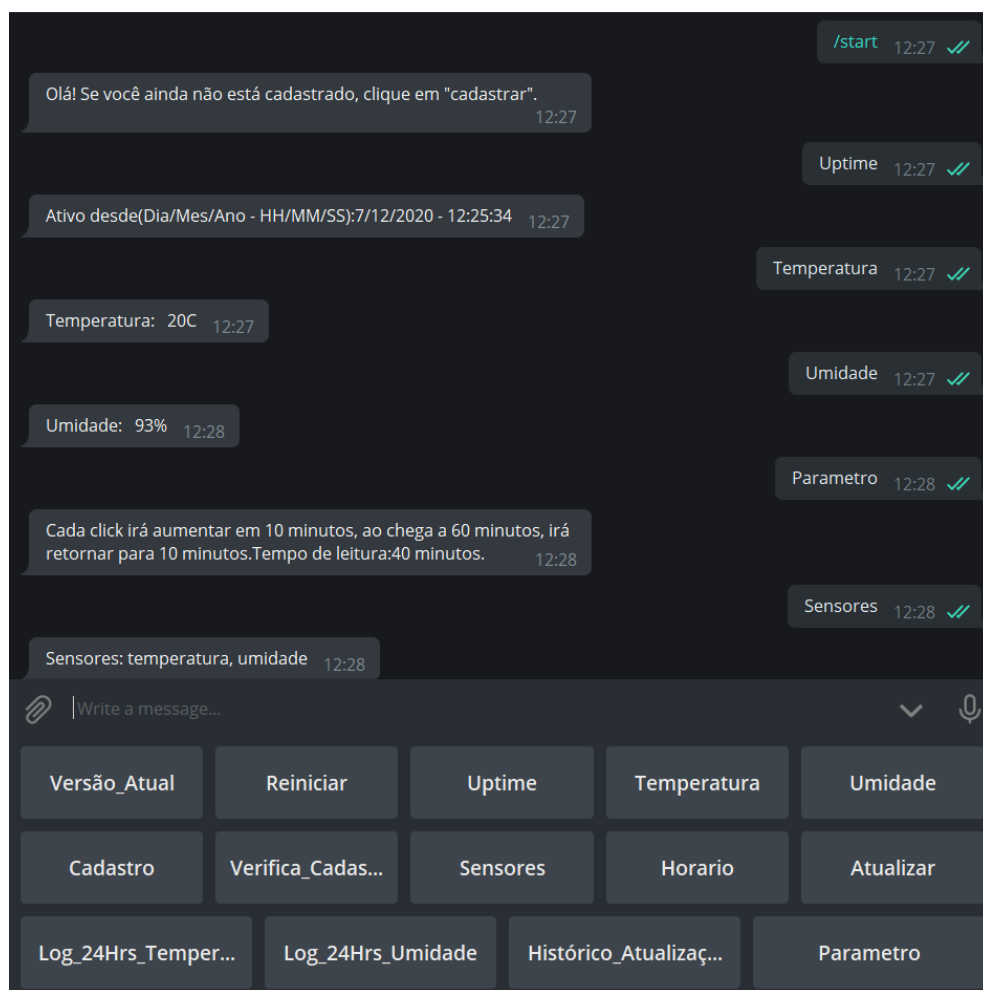
```

mq.connect()
mq.publish(b"Topico",b""+str(version_v))

```

[Figura 5]

Após todas essas etapas concluídas corretamente, podemos então solicitar registro de usuário na API para emitir comandos tarefas pré-definidas em botões para o embarcado por meio do aplicativo Telegram com a API em execução, veja alguns exemplos na imagem abaixo:



[Figura 6]

10 Conclusão

Conclui-se que é possível implementar atualização OTA juntamente com o aplicativo Telegram facilitando o acompanhamento e execução das mesmas, além da execução de algumas outras funcionalidades de forma mais prática tornando a aplicação de comandos, assim como, a visualização de determinadas tarefas através de um celular ou computador que disponha do aplicativo Telegram, muito mais simples e otimizada para o usuário final.

11 Referencias

Apêndice A

Repositório GitHub usado como base para API Telegram: <https://github.com/nconnector/iot-garage-door-telegram>.

Repositorio GitHub para Download do projeto apresentado neste relatório: https://github.com/lucaspennning/Telegram_OTA.