EA076 – LABORATÓRIO DE SISTEMAS EMBARCADOS

Tópico: *Internet* das Coisas

Segundo Semestre de 2019

ROTEIRO 1: Infra-estrutura baseada no Protocolo MQTT

Prof. Antonio Quevedo e Profa. Wu Shin Ting

A característica mais marcante da Internet das Coisas (*Internet of Things* ou **IoT**) é a comunicação entre os objetos com uma certa autonomia de decisão seja em rede ou ponto-a-ponto. Por exemplo, através de Identificação por radiofrequência, ou **RFID** (*Radio-Frequency IDentification*), ou tecnologia de transmissão sem fio (*wireless*), os objetos inteligentes se identificam automaticamente e recuperam/armazenam os dados dispersos geograficamente. Para que tudo isto seja possível, os objetos necessitam de um **protocolo** de comunicação.

Um dos protocolos mais utilizados em IoT é o **MQTT** (*Message Queue Telemetry Transport*), com base no *stack* TCP/IP. Assim, este protocolo utiliza a estrutura usual da internet para a transmissão de mensagens, intermediando-as através de um servidor, neste caso chamado de **Broker**. Este servidor se conecta a múltiplos clientes. Os clientes enviam e recebem mensagens, e o *Broker* controla o fluxo das mesmas [1]. O modelo utilizado neste protocolo é o de **Publicação e Assinatura**. Cada cliente pode "assinar" um ou mais **tópicos**. Um tópico é a chave que identifica o canal de informações no qual as mensagens são "publicadas". Os assinantes usam o nome do tópico para identificar os canais de informações nos quais eles desejam receber mensagens publicadas. .Cada tópico pode ser assinado por mais de um cliente. Isto permite muita flexibilidade nas interações entre os clientes. Mais, é possível hierarquizar os tópicos em níveis, permitindo sub-tópicos aninhados e assinaturas em grupos de sub-tópicos [2].

Neste curso, exercitaremos alguns conceitos relevantes de IoT através de um projeto de uma rede local de sensores de temperatura, de motores CC 5V e de celulares que se comunicam entre si pelo protocolo MQTT. Os sensores e os motores serão integrados com os microcontroladores para ganharem autonomia na auto-identificação perante os outros objetos conectados à rede e na execução de algumas ações bem específicas. Além disso, teremos acoplados a cada estação (composta por um sensor, um motor e uma unidade de memória) um *display* e um teclado para configurações locais.

Nesta primeira parte do curso, vamos iniciar a implementação de um sistema microcontrolado para IoT. Começaremos por uma etapa fundamental do processo, que é a definição dos pinos do microcontrolador a serem utilizados, levando em conta as interfaces específicas disponíveis em determinados pinos. Depois, implementaremos um sistema básico de IoT, capaz de trocar mensagens de texto com outros clientes MQTT. É importante ressaltar que utilizaremos uma rede *WiFi* e um *Broker* dedicados e isolados da Internet, por questões de segurança. Entretanto, se fosse utilizada uma rede *WiFi* conectada à internet, poderíamos usar um *Broker* localizado na nuvem, adicionando um protocolo de criptografia (SSL). Assim, poderíamos comunicar com os objetos de forma segura estando em qualquer lugar,

A distribuição das aulas para cada tarefa é uma sugestão, para que os alunos tenham uma noção do andamento do trabalho.

Estrutura:

Primeira aula

- Alocação de pinos para o projeto e preparo da placa para o curso

Segunda aula

- Conexão física do Módulo ESP-01 com a placa FRDM-KL25Z
- Programação das interfaces seriais
- Comunicação do FRDM-KL25Z com a rede WIFI específica do LE-33

Terceira e quarta aulas

- Primeiro contato com o protocolo MQTT
- Uso do Terminal Serial USB para gerenciar as configurações e fazer testes básicos de comunicação
- Formatação e extração dos comandos na sintaxe do protocolo do *firmware* MQTT que roda no módulo ESP-01
- Troca de mensagens entre cliente MQTT em celular pessoal com a placa FRDM

Tarefas:

Análise das interfaces dos periféricos e alocação de pinos

1) Neste projeto são conectados os seguintes periféricos em cada estação controlada por um microcontrolador: 1 módulo ESP-01 [3] (serial), 1 *display* Nokia [4] (SPI com pinos MOSI e CLK, mais 4 GPIOs), 1 motor CC 5V interfaceado por uma ponte-H L293D [5] (canal de *timer* mais 2 GPIOs), 1 sensor de temperatura LM-61 [6] (uma entrada analógica), 1 teclado 4x3 [7] (4 GPIOs e 3 IRQs) e 1 unidade de memória Atmel 24C6 [8] (I2C). Utilizaremos o *plugin* Terminal já integrado no ambiente CodeWarrior com acesso serial alocado para UART0 (pinos PTA1 e PTA2). Em Anexo encontra-se um esquemático do projeto que completaremos ao longo do curso. Para o desenho do esquemático em Eagle [9], foram adicionados dois componentes novos ao repositório básico: o componente FRDM, FRDM.lbr [10] e a memória, atmel.lbr [11].

   Apresente uma proposta de alocação de pinos da FRDM-KL25, considerando os pinos disponíveis nos *headers*.

Obs: veja que algumas interfaces só estão disponíveis em alguns pinos. Cuide também para alocar UARTs distintas para as duas portas seriais. A Serial USB (via OpenSDA, UART0) já está conectada aos pinos PTA1 e PTA2 na placa FRDM. O módulo ESP-01 é conectado através de um conector DB-9. Assim, use o símbolo do conector DB-9 (*female*) e a seguinte pinagem:

- Vcc (5V): Pino 9; GND: Pino 5; Tx: Pino 3; Rx: Pino 2
- Tx e Rx são do ponto de vista do ESP8266.
- A alimentação do módulo é de 5V, da fonte externa. Há um regulador de tensão no módulo, e os níveis de tensão em Tx e Rx são de 3,3V.

Infra-estrutura de uma rede de estações

2) Consideramos como uma estação o conjunto formado pela placa de desenvolvimento com a placa FRDM-KL25 e os dispositivos a ela conectados. Faça a conexão da porta serial UART2 (usando os pinos PTE22 e PTE23) com o conector DB-9 da placa. Conecte também o GND e a alimentação externa de 5V, de acordo com a pinagem apresentada no item (1). Conecte o módulo ESP8266 no conector DB-9 da placa.

3) Programe as portas seriais (UART0 no OpenSDA e UART2 no módulo ESP) para que quando uma receba um caractere, o retransmita para a outra (se o caractere vier da UART0, retransmita para as 2 UARTs para que haja "eco" do caractere). Use interrupções nos RX para setar *flags*. Depois use o terminal serial no computador para conectar à UART0 e experimente os comandos de versão atual, de conexão com o WiFi, e os que fornecem o MAC Address e o IP [12].

Obs. 1: o ENTER do PC produz apenas o caracter CR (0x0D), e o módulo precisa dos caracteres CR+LF (0x0D – 0x0A) para saber que o comando encerrou. Assim, faça seu programa adicionar um caractere 0x0A logo após o 0x0D.

Conexão entre estações e celulares

4) Ainda com uma porta serial retransmitindo para a outra, experimente conectar ao *broker* usando como *clientID* a string do *MAC Address* e depois assine um tópico denominado "EA076/grupo*Xn*/celular", onde *X* é a letra da turma e *n* é o número do grupo. Usando um app cliente para *smartphone* (sugestão: *MyMQTT*), conecte ao *broker* e envie uma mensagem (*Publish*) ao tópico definido acima. Veja como ela é recebida pelo módulo ESP (sintaxe).

Obs. 2: Recomenda-se usar como *ClientID* o MAC Address, para assegurar a unicidade.

5) Agora, assine no app do *smartphone* um tópico denominado "EA076/grupo*Xn*/ESP", e envie uma mensagem para este tópico a partir do módulo ESP. Veja como a mensagem aparece.

6) Cancele a assinatura do tópico "EA076/grupo*Xn*/celular" no módulo ESP. Envie uma mensagem para este tópico através do *smartphone* e confirme que o módulo ESP não recebe mais a mensagem.

7) Faça um novo programa que execute as seguintes etapas automaticamente:

a) Conecte à rede WiFi, aguardando a conclusão da conexão

b) Conecte ao *broker* usando como *clientID* o MAC Address, aguardando a conclusão da conexão

c) Assine o tópico "EA076/grupo*Xn*/celular"

d) Escreva uma mensagem na UART0 dizendo que está tudo conectado.

e) Depois este programa deve entrar em um *loop* para repetir as seguintes tarefas:

e.1) Verifique se há mensagem recebida pelo módulo ESP. Separe as *strings* de tópico e de mensagem. Se for mensagem do tópico do item 7c, envie-a à UART0 (do OpenSDA), sem o tópico e sem os "[ ]", apenas a mensagem pura. Se for mensagem de erro, envie um comunicado de erro (incluindo o tipo) à UART0.

e.2) Verifique se há caracteres enviados pela UART0, faça o "eco" no terminal e os adicione a um *buffer*. Se o caractere de CR (0x0C, ou ENTER) chegar, publique o conteúdo do *buffer* no tópico "EA076/grupo*Xn*/ESP". Use o cliente no *smartphone* para receber a mensagem.

Apresentação:

Para os itens (1) e (2), apresente uma tabela com a função alocada para cada pino e o esquemático das ligações. Os itens (3) a (6) são para experimentação apenas. Para o item (7), apresente no relatório os códigos desenvolvidos (main.c e Events.c) com a documentação em *doxygen* (opcional por enquanto), além do funcionamento no vídeo postado. Recomendo não pular as etapas (3) a (6), pois elas ajudam muito a compreensão do funcionamento do protocolo.

Bibliografia:

[1] Conhecendo o MQTT (https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html)

[2] MQTT - Protocolos para IoT (https://www.embarcados.com.br/mqtt-protocolos-para-iot/)

[3] Instructables. Getting Started with the ESP8266 ESP-01 https://www.instructables.com/id/Getting-Started-With-the-ESP8266-ESP-01/

[4] ETT. User's Manual of Graphic LCD "ET-NOKIA LCD 5110" ftp://ftp.dca.fee.unicamp.br/pub/docs/ea076/complementos/User_Manual_ET_LCD5110.pdf

[5] Datasheet: L293D ftp://ftp.dca.fee.unicamp.br/pub/docs/ea076/datasheet/Half-H_Driver_L293.pdf

[6] Datasheet: LM-61. ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/datasheet/LM61.pdf

[7] Ali Behbehani.12-Key Keypad Connection to a Microcontroller.

https://www.egr.msu.edu/classes/ece480/capstone/spring12/group07/ECE480/Documents_files/Ali_Behbehani_Application_Notes.pdf

[8] Datasheet: AT24C. ftp://ftp.dca.fee.unicamp.br/pub/docs/ea076/datasheet/AT24C.pdf

[9] Rodrigo Krug. CadSoft Eagle 5.10: Uma Aplicação Prática. ftp://ftp.dca.fee.unicamp.br/pub/docs/ea076/manuais/Tutorial_Eagle_5_10.pdf

[10] Componente FRDM.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea076/complementos/Freedom.lbr

[11] Componente Memória Atmel 24C6.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea076/complementos/atmel.lbr

[12] Antonio Quevedo. ESP8266 Cliente MQTT.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea076/complementos/ESP8266_CLIENTE_MQTT.pdf

Anexo

ESP8266EX Standalone Device