USO DE MICROPYTHON EM MICROCONTROLADORES ESP

Carlos Henrique Gomes Taborda Lopes (UEPG), Dierone César Foltran Júnior, hlopes.ec@gmail.com: dcfoltran@gmail.com.

Universidade Estadual de Ponta Grossa/Departamento de Ciências Agrárias e de Tecnologia.

Ciência da Computação, Sistemas de Computação, Software Básica.

Palavras-chave: Micropython, ESP, microcontrolador, ESP8266.

Resumo

A tecnologia atual nos proporciona a utilização de microcontroladores, dispositivos pequenos com alto desempenho de processamento utilizados para automação. Entretanto para sua utilização é necessário que todo conjunto de instruções seja previamente criada pelo projetista, desta forma com a utilização do *framework* MicroPython e microcontroladores ESP busca-se a criação de um sistema que realize a leitura de sensores e os disponibilize para acesso de uma API.

Introdução

Desde o início da utilização de sistemas robóticos buscamos tecnologias com alto desempenho, baixo custo e pequeno tamanho. Atualmente nos deparamos com inúmeros microcontroladores em nosso cotidiano, tais como: controles remotos, controles de automóveis, eletrodomésticos, brinquedos, entre outros.
Um microcontrolador consiste de um único circuito integrado contendo um núcleo de processamento, memória (RAM, PROM, Flash) e periféricos programáveis de entrada e saída, que executam aplicações embarcadas.
O microcontrolador teve sua origem em meados da década de 70 quando a empresa Intel iniciou sua linha de produção de microprocessadores de baixo custo, alguns exemplos são os modelos 4004, 4040 e 8051.



Figura 1 - Microcontroladores Intel 4004, 4040 e 8051

A partir de sua utilização ocorreu substituição da mão de obra manual pela automatização de processos, onde um mesmo dispositivo passou a agregar cada vez mais mais funcionalidades com o passar do tempo, passando então ser chamado de SoC (*System On Chip*) e atualmente de microcontroladores.
Dentre os microcontroladores disponiveis no mercado os modelos ESP apresentam um se destacam por sua versatilidade, sendo bastante utilizados para aplicações de IoT (*Internet of Things*) e automação.
Diferente da desenvolvimento de aplicações para microprocessadores, que dispõem de sistema operacional e BIOS, o desenvolvimento para microcontroladores

permitem uma maior liberdade para a criação de sistemas semelhantes aos utilizados em computadores.

Materiais e Métodos

Na busca da definição do modelo ideal a ser utilizado foi realizado uma prospecção de diferentes modelos de microcontroladores ESP, dentre as escolhas se destacaram os modelos ESP32 e ESP8266.
O modelo ESP32 apresentou melhor desempenho de hardware entretanto, a documentação não fornecia detalhes sobre o microcontrolador, além disso, a versão do firmware disponibilizada para o *framework* MicroPython era inferior à versão disponivel para o modelo ESP8266, esse fator foi determinante fazendo com o modelo ESP8266 fosse escolhido para utilização.
A utilização da *framework* possibilita um menor tempo de desenvolvimento pois funções bases já estão disponibilizadas nas suas bibliotecas.
O framework MicroPyhton, é uma implementação da linguagem Python 3 com otimizações para execução em microcontroladores, para que isso ocorra parte dos módulos que a linguagem oferece são removidos caso não sejam úteis aos microcontroladores, assim, o limite de memória oferecido não é ultrapassado.(2)
Devido á esse fato a linguagem trabalha com a gerencia de memoria atraves de *garbage collection* enquanto a versão normal trabalha como *reference-counting.*
Isso posto, o ESP8266 é um microcontrolador de processador RISC Tensilica L106 de 32bits, tempo de *clock* de 160MHz e Wi-fi integrado, isso possibilita um cachê de alta velocidade aumentando a performance de sistema e de memória.
A arquitetura RISC (*Reduced Instruction Set Computer*) oferece um conjunto pequeno de instruções simples que levam aproximadamente o mesmo tempo para serem executadas.

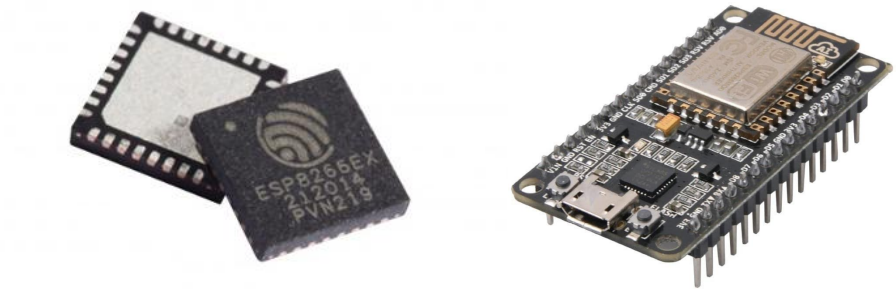


Figura 2 - Microcontrolador ESP8266 e módulo de desenvolvimento NodeMCU

Para o desenvolvimento da pesquisa um fator de grande importância era a maneira que os dados seriam disponibilizados, então foi optado pela disponibilização dos dados através da rede para uma API.
Para isso a conexão do dispositivo pode ocorrer de dois modos, o modo *station* é usado quando o ESP se comunica a rede e o modo *ponto de acesso* é usado quando o ESP é conectado por outros dispositivos.
O desenvolvimento ocorreu utilizando as bibliotecas disponibilizadas pela *framework* e implementações baseadas na necessidade do uso com a utilização de Orientação a Objetos.
Utilizando Orientação a Objetos foi possivel fazer com que a configuração necessária para utilização da classe ocorresse no momento em que era necessário o uso fazendo com que não fosse necessário o carregamento de toda configuração no momento de inicialização do dispositivo.

O armazenamento dos logs era um fator importante para controle, as informações serão salvas em um arquivo de *log*, entretanto devido ao tamanho de armazenamento do dispositivo ser de 4MB era necessário que o log ocorresse da maneira a economizar memória.
Para isso antes de armazenar o dado no *log* era primeiramente verificado na função *callback* se estava fora da sua margem de erro para então sobrescrever o dado anterior caso estivesse.
Para conexões de sensores analógicos é necessário que eles sejam multiplexados pois o ESP8266 apenas possui uma porta analógica, entretanto possui 17 portas GPIOs (*General Purpose Input/Output*).(3)
GPIOs são portas usadas para interação com periféricos externos, geralmente sensores, e podem ser definidas como entrada e saída de sinais digitais com sinal alto (1) e baixo (0) usadas para funcionalidades como UART, I2C, SDIO, PWM e etc.
Seu consumo de energia é entre 2.5V e 3.3V com uma corrente de 12mA, os valores variam dependendo do seu modo de funcionamento.(4)

Resultados e Discussão

O microcontrolador utilizado apresentou bom resultado para utilização de sensores digitais porém devido ao fato de possuir somente uma porta analógica exigindo que seja multiplexado para o uso de mais sensores criando então um aumento na complexidade do sistema.
Seu suporte a linguagens C e Python e a maneira que ocorre o gerenciamento dos arquivos possibilita uma fácil manutenção de código.

Conclusões

Com a utilização do *framework* Micropython é possível criar aplicações de automação com maior facilidade.
O microcontrolador ESP8266 apresenta algumas limitações sendo a principal delas o fato de possuir somente uma porta analógica.
Um dos principais problemas encontrados durante o desenvolvimento foi a execução de algumas funções que exigiam um tempo *sleep* caso contrario não apresentariam um bom funcionamento.

Agradecimentos

Agradeço ao professor Dierone César Foltran Júnior pela possibilidade de pesquisa e pelo auxílio durante sua realização e a UEPG pelo auxílio financeiro durante a realização da pesquisa.

Referências

1. Internet das Coisas com ESP8266, Arduino e Raspberry Pi. Disponivel em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=kdQnDwAAQBAJ&oi=fnd&pg=PA17&dq=ESP8266&ots=zLxivb32WL&sig=Q4jptiV55TJiDt1EVdKF-c4pJoo#v=onepage&q=ESP8266&f=false>. Acesso em 23 de Agosto de 2018.
2. MicroPython. Disponível em: <https://micropython.org/>. Acesso em 23 de Agosto de 2018.

MicroPython Tutorial. Disponível em: <https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/intro.html>. Acesso em 23 de Agosto de 2018.
3. Differences to CPython. Disponível em <https://github.com/micropython/micropython/wiki/Differences>. Acesso em 28 de Agosto de 2018.
4. ESP8266X Datasheet - Expressif System. Disponível em < https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. Acesso em 28 de Agosto de 2018.