Universidade Federal Fluminense - UFF
Escola de Engenharia - TCE
Curso de Engenharia de Telecomunicações - TGT

Programa de Educação Tutorial – PET
Grupo PET-Tele

Tutoriais PET-Tele

Implementação de um sistema de telemetria de vazão de água utilizando Arduino e *Water Flow Sensor*

(Versão: A2017M08D14)

Autor: Lucas Pontes Siqueira
Thiago Chequer Coelho

Tutor: Alexandre Santos de la Vega

Niterói – RJ
Agosto / 2017

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução e Motivação

O grupo PET-Tele trabalha há algum tempo desenvolvendo com a plataforma de prototipagem *Arduino*, e assim vem resultando na criação de materiais didáticos, cursos práticos, projetos e até uma optativa ministrada na Universidade Federal Fluminense. Isso tudo para, de alguma forma, repassar conhecimento e auxiliar primariamente os alunos de Engenharia de Telecomunicações e outros cursos.

O *Arduino* em sua arquitetura permite a associação com diversos outros periféricos, deixando assim mais ampla a gama de possibilidades a serem construídas. Essas que nos auxiliam na solução de problemas nas mais diversas vertentes de engenharia.

Dentre as várias dificuldades que atualmente encontramos no dia-a-dia, podemos dizer que uma das mais significativa delas é a quantização ou, em outras palavras, a obtenção de dados estatísticos.

Entretanto, não apenas a aquisição desses dados compõe a lista de "problemas" que nos acompanha na vertente de quantização de informações. Os instrumentos que nós utilizamos são fundamentais para nos garantir medições plausíveis e, além disso, nos fornecer esses dados de forma simples e otimizada.

Já não bastam, por exemplo, os problemas que encontraremos ao desenvolver cálculos com os dados obtidos, enfrentar dificuldades na obtenção desses dados atrasaria desnecessariamente o progresso da pesquisa. Dessa forma, é de suma importância que utilizemos os instrumentos e as técnicas corretas para obtenção dos dados.

Avançando para o campo tecnológico, a técnica que permite a medição e envio de dados do objeto de estudo até o utilizador dessas informações é chamada *Telemetria*. Da morfologia da palavra temos "tele" que significa distância e "metria" que remete ao significado de medida. Dessa forma, concluímos que trata-se de uma tecnologia de medição remota.

No presente documento, buscamos apresentar um tutorial que é fruto de uma implementação telemétrica utilizando um sensor de medição de vazão d'água juntamente com a plataforma de prototipagem *Arduino*. Resumidamente, trata-se de uma solução que o grupo implantou no Laboratório de Drenagem, Irrigação e Saneamento Ambiental (LaDISan) da Universidade Federal Fluminense para otimizar a medição de vazão de entrada e saída de água num reservatório.

2 Um pouco sobre Arduino

O Arduino faz parte do conceito de hardware e software livre e está aberto para uso e contribuição de toda sociedade. O conceito Arduino surgiu na Itália, em 2005, com o objetivo de criar um dispositivo que fosse utilizado em projetos/protótipos construídos de uma forma menos dispendiosa do que outros sistemas disponíveis no mercado.

Ele pode ser usado para desenvolver artefatos interativos stand-alone ou conectados ao computador, utilizando diversos aplicativos, tais como: Adobe Flash, Processing, Max/MSP, Pure Data ou SuperCollider.

O Arduino foi projetado com a finalidade de ser de fácil entendimento, de fácil programação e de fácil aplicação, além de ser multiplataforma, podendo ser configurado em ambientes Linux, Mac OS e Windows. Além disso, um grande diferencial deste dispositivo é ser mantido por uma comunidade que trabalha na filosofia open-source, desenvolvendo e divulgando gratuitamente seus projetos.

O equipamento é uma plataforma de computação física: são sistemas digitais ligados a sensores e atuadores, que permitem construir sistemas que percebam a realidade e respondem com ações físicas. Ele é baseado em uma placa microcontrolada, com acessos de Entrada/-Saída (I/O), sobre a qual foram desenvolvidas bibliotecas com funções que simplificam a sua programação, por meio de uma sintaxe similar à das linguagens C e C++.

O Arduino utiliza o microcontrolador Atmega. Um microcontrolador (também denominado MCU) é um computador em um chip, que contém um microprocessador, memória e periféricos de entrada/saída. Ele pode ser embarcado no interior de algum outro dispositivo, que, neste caso, é o Arduino, para que possa controlar suas funções ou ações.

Em resumo, o Arduino é um kit de desenvolvimento, que pode ser visto como uma unidade de processamento capaz de mensurar variáveis do ambiente externo, transformadas em um sinal elétrico correspondente, através de sensores ligados aos seus terminais de entrada. De posse da informação, ele pode processá-la computacionalmente. Por fim, ele pode ainda atuar no controle ou no acionamento de algum outro elemento eletro-eletrônico conectado ao terminal de saída.

Uma vez que o Arduino é baseado em um microcontrolador e, portanto, é programável, torna-se possível criar diversas aplicações diferentes com uma certa facilidade. Além disso, o próprio equipamento pode ser reutilizado, através de uma nova programação. Por sua vez, a sua programação é simplificada pela existência de diversas funções que controlam o dispositivo, com uma sintaxe similar a de linguagens de programação comumente utilizadas (C e C++).

Assim sendo, em um ambiente profissional, as características do Arduino fazem dele uma boa ferramenta de prototipagem rápida e de projeto simplificado. Por outro lado, em um ambiente acadêmico, ele pode ser perfeitamente utilizado como ferramenta educacional, uma vez que não requer do usuário conhecimentos profundos de eletrônica digital nem da programação de dispositivos digitais específicos.

3 Componentes do sistema e prototipagem

Ao passo que discutiremos um pouco da técnica de programação e as linhas de código utilizadas no desenvolvimento do sistema, vamos discutir inicialmente o que tange a montagem do sistema.

Na montagem do sistema foi usada uma protoboard para auxiliar nas conexões entre os sensores e ao Arduino. Os terminais de Vcc e GND dos sensores foram conectados respectivamente à saída de 5 V do Arduino e ao GND. Já as saídas dos sinais foram ligadas a portas digitais 2 e 3.

Para fazer a leitura de um sinal na forma de onda quadrada, o Arduino UNO possui um método de 4 tipos diferentes de detecção dos pulsos que iriam chamar um procedimento de interrupção. Os tipos de detecção acontecem quando o pino está em nível baixo, quando é detectada uma mudança de valor, quando o pino lê uma transição de subida e quando lê uma transição de descida. Esse tipo de leitura somente pode ser feita nos pinos 2 e 3.

No projeto foi usada a transição de subida portanto, toda vez que o Arduino detectasse uma subida no valor lógico do sinal, a função de interrupção seria chamada.

No primeiro momento falaremos um pouco sobre o sensor que foi utilizado para medição da vazão.

3.1 Sensor *Water Flow*

Figura 1: G1/2 Water Flow sensor

Em poucas palavras, o sensor de fluxo em questão é feito de plástico e apresenta, no seu interior, uma hélice para fluidos e um sensor de efeito hall.

Rapidamente pontuando, o efeito hall pode ser caracterizado pelo surgimento de uma diferença de potencial, transversal ao fluxo de corrente e um campo magnético perpendicular à corrente, em outras palavras, há um ímã em uma das hélices presentes no sensor que a cada rotação interage com um outro ímã fixado no corpo do sensor gerando um estímulo eletromagnético.

Quando a água flui pelo corpo do sensor, a hélice adquire uma velocidade de acordo com a velocidade da água, que é traduzida na quantidade de interações magnéticas resultantes do efeito hall num determinado intervalo de tempo.

Com essa informações é possível calcular (veremos no código) a quantidade de vazão d'água em metros cúbicos por segundo, por exemplo.

4 Programação

4.1 Pinagem de interrupção em cada placa

Cada versão da placa do Arduino conta com uma quantidade diferente de pinos digitais que, por *default*, são configurados para serem operados com uma rotina de interrupção como a necessária nesse projeto.

Vamos diagramar numa tabela as placas e os pinos digitais que já vem de fábrica com opção de trabalhar com interrupção:

Placa	Pinos digitais
Uno, Nano, Mini, other 328-based	2, 3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	Todos, exceto o 4
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due e 101	Todos

Com "configurado de fábrica" podemos entender como um pino que foi preparado para estar apto a operar com rotinas de interrupção.

Entretanto, como visto na tabela anterior, as placas mais acessíveis, bem como as mais utilizadas pela comunidade (Uno e Nano) contam apenas com dois pinos digitais para essa tarefa.

A conclusão mais ingênua nos leva a crer que para utilizamos mais do que dois sensores que operam via interrupção, temos que ter a disposição uma versão da placa que suporte essa quantidade de sensores.

Entretanto, existe uma biblioteca chamada *PinChangeInt.h* que nos permite fazer com que qualquer pino digital da placa opere com rotinas de interrupção corretamente.

Vale reiterar o ponto da "configuração de fábrica", pois uma questão importante é uma queda de desempenho na aquisição de dados por esses pinos "forçadamente" configurados. Isto é, podemos encontrar um atraso da ordem de milissegundos na transmissão dos dados.

Nessa circunstância, cabe ao usuário arcar com as consequências dessa adaptação. No caso estudado agora, podemos tranquilamente relevar o atraso causado por essa alteração, pois trata-se de um sensor de medição de vazão de água que leva um tempo significativa maior que o atraso gerado pela interrupção para sentir o estímulo de variação de vazão d'água.

O código começa a inclusão da biblioteca de mudança de configuração de pinos (*PinChangeInt)*), logo em seguida definimos o pino que será reconfigurado e após isso declaramos as variáveis. Foram declaradas três variáveis para a contagem de pulsos e três para receber o valor da vazão. Cada variável dessas é referente a um sensor.

Após a declaração vem a função void setup na qual fazemos as configurações necessárias para o projeto. Configuramos a taxa de comunicação serial entre o computador e o Arduino como (bits por segundo). Declaramos as portas digitais usadas como entrada e depois configuramos para serem usadas para chamar a rotina de interrupção a cada vez que o sinal muda de LOW para HIGH.

Na void loop começamos atribuindo o valor 0 às variáveis responsáveis por armazenar os pulsos. Para começar a leitura iremos habilitar a interrupção, pausar o programa por 1 segundo e depois desabilitar a interrupção. Durante esse intervalo de 1 segundo, a cada vez que a leitura

dos sinais de saída dos sensores captar uma transição de subida as ISR de entrada, intermediária e saída serão chamadas e nelas a variável de contagem de pulsos será acrescida de uma unidade.

Após a duração de 1 segundo o programa prosseguirá e fará a conversão do número de pulsos contados para a vazão usando a taxa de conversão. Após isso, as vazões serão mostradas na Serial Monitor.

Com isso, o programa fará a medição da litros por segundo e repetirá o processo.

5 Exemplo de código

```
//Desenvolvido por PET-Tele UFF
//Telemetria em tempo real de três sensores concomitantes.
//Arduino UNO (Pinos de interrupção adicionais)

#include "PinChangeInt.h"
#define PinItrpt11 11

float vazao_entrada;
float vazao_intermediaria;
float vazao_saida;

int contaPulso_entrada;
int contaPulso_intermediaria;
int contaPulso_saida;

int i=0;
void incpulso_entrada (){
  contaPulso_entrada++;
}

void incpulso_intermediaria (){
  contaPulso_intermediaria++;
}

void incpulso_saida (){
  contaPulso_saida++;
}

void setup(){

  Serial.begin(9600);

  pinMode(2, INPUT);
  attachInterrupt(0, incpulso_entrada, RISING);

  pinMode(3, INPUT);
  attachInterrupt(1, incpulso_saida, RISING);

  pinMode(PinItrpt11, INPUT);
```

```
  PCintPort::attachInterrupt(PinItrpt11, incpulso_intermediaria, RISING);

  Serial.println("\nINICIO\n"); //Imprime Inicio na serial
  Serial.println("\n ENTRADA -- INTERMEDIARIO -- SAIDA \n");

}

void loop (){

  contaPulso_entrada = 0;
  contaPulso_intermediaria = 0;
  contaPulso_saida = 0;

  sei();
  delay (1000);
  cli();

  vazao_entrada = contaPulso_entrada/7.5;
  vazao_intermediaria = contaPulso_intermediaria/7.5;
  vazao_saida = contaPulso_saida/7.5;
  i++;

  Serial.print("  ");
  Serial.print(vazao_entrada);
  Serial.print("    ");
  Serial.print(vazao_intermediaria);
  Serial.print("    ");
  Serial.print(vazao_saida);
  Serial.print("    ");
  Serial.print(" L/min");
  Serial.print("    ");
  Serial.print(i);
  Serial.println("s");

}
```

Referências

[1] *http://wiki.seeedstudio.com/wiki/G1/2_Water_Flow_sensor*

[2] *https://brainy-bits.com/blogs/tutorials/make-any-arduino-pin-an-interrupt-pin*

[3] ... REFERÊNCIA AO ARTIGO PUBLICADO ...

[4] *https://www.arduino.cc/en/Reference/AttachInterrupt*