Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Coordenação de Eletrotécnica

Microcontrolador Arduino: Aplicação em controle PI

Autor: Prof. Alessandro N. Vargas

Objetivo

Conhecer o funcionamento do Microcontrolador Arduino e realizar a sua programação para que ele atue como um Controlador PI adaptado ao Kit Motor DC.

1 Arduino

O microcontrolador Arduino é um dispositivo programável versátil, fácil de programar, e tem encontrado muitas aplicações recentes em robótica, eletrônica, e inclusive em processos industriais. O Arduino é *open-source*, o que significa que seu software de desenvolvimento é grátis e seu hardware foi desenvolvido para que tenha um preço mais acessível. O link do projeto Arduino é `http://www.arduino.cc` e o preço do Arduino Uno, o modelo mais tradicional, encontra-se na faixa de R$ 70,00 (lojas no Brasil).

Usando a placa Arduino, pode-se escrever programas e criar rotinas para ler sinais, por exemplo sinais gerados por sensores, e pode ser usado também para gerar sinais, por exemplo para controle de motores, luzes, relés, transistores. Pode-se inclusive gerar sinais PWM de maneira muito simplificada.

A linguagem de programação do Arduino é uma versão simplificada da linguagem C/C++.

O Laboratório de Controle e Automação da UTFPR têm disponível o modelo Arduino Uno Rev3, que possui as seguintes características:

Figura 1: Foto de um Arduino modelo Uno.

- Microcontrolador ATmega328 operando sob 5 V com 2 Kb de RAM;
- Tensão de alimentação da placa entre 7-12V;
- 14 pinos de Entrada/Saída digitais (6 saídas PWM);
- 6 Entradas Analógicas;
- 32k Flash Memory para guardar os programas e 1 Kb de EEPROM para guardar os parametros;
- 16Mhz velocidade de clock, que significa uma execução de aproximadamente 300.000 linhas de código em linguagem C por segundo;
- Conector USB para que o Arduino converse com um PC que hospedará o ambiente de programação;

1.1 Estrutura do programa

Todos os programas Arduino devem conter duas estruturas básicas: `setup()` e `loop()`. As instruções colocadas dentro do `setup()` são executadas uma única vez; essas instruções normalmente são usadas para inicializar outros procedimentos. As instruções colocadas dentro do `loop()` são executadas repedidamente, permanentemente, e forma a principal tarefa do programa. Então cada programa deve conter a seguinte estrutura:

```
void setup()
{
// commands to initialize go here
}
void loop()
{
// commands to run your machine go here
}
```

1.2 Comandos usuais

1.2.1 `pinMode`

Esse comando `pinMode`, que deve ir dentro da função `setup()`, é usado para gravar a direção de um pino I/O digital. Grave o pino como `OUTPUT` se o pino está gerando um sinal de saída, por exemplo acendendo um LED, controlando um motor, etc. Grave o pino como `INPUT` se o pino está lendo um sinal de sensor ou chave ou outro sensor. O exemplo ao lado grava o pino 2 como saída e pino 3 como entrada.

```
void setup()
{
pinMode(2,OUTPUT);
pinMode(3,INPUT);
}
void loop() {}
```

1.2.2 `Serial.print`

Esse comando `Serial.print` permite visualizarmos o que ocorre dentro do Arduino através do PC acoplado ao Arduino via cabo USB. Para o correto funcionamento, o comando `Serial.begin(9600)` deve ser inserido dentro do `setup()`. Após o comando ser programado, é necessário abrir no PC a janela do ambiente "Serial Monitor". Há duas maneiras de visualizar a informação: `Serial.print()` imprime na mesma linha e `Serial.println()` começa a imprimir numa nova linha.

Programe o código a seguir, e use um fio jumper para conectar o pino 2 na tensão +5V e no Gnd.

```
void setup()
{
Serial.begin(9600);
}
void loop()
{
Serial.println(digitalRead(2));
delay(100);                              // realiza atraso de 100 ms
}
```

1.2.3 digitalWrite

Esse comando grava um pino I/O em "high" (+5V) ou "low" (0V) e é um comando extremamente importante para interfacear o Arduino com o mundo externo. Não se esqueça de usar o comando `pinMode()` dentro de `setup()` para **gravar o pino como saída**.

```
digitalWrite(2,HIGH); // sets pin 2 to +5 volts
digitalWrite(2,LOW); // sets pin 2 to zero volts
```

1.2.4 delay

O comando `delay` congela a execução pela quantidade de tempo especificada em millisegundos.

```
digitalWrite(2,HIGH); // pin 2 high (LED on)
delay(500); // wait 500 ms
digitalWrite(2,LOW); // pin 2 low (LED off)
```

1.2.5 analogRead(pinNumber)

Para receber um sinal analógico, o Arduino usa os numeros de pinos 0 à 5 correspondendo aos pinos físicos `A0,A1,A2,A3,A4,A5`. Portanto há seis entradas analógicas. Para ler uma informação analógica basta programar o comando `analogRead(pinNumber)` no qual `pinNumber` é o numero do pino de entrada no qual deve-se realizar a leitura. Esse comando `analogRead` retornará um número inteiro entre 0 e 1023, que deve ser entendido como uma leitura proporcional das tensões entre 0V e +5V.

1.2.6 analogWrite(pinNumber, value)

Arduino não possui saída analógica, então essa função com nome contraditório, `analogWrite`, **não realiza saída analógica**. O Arduino só possui saída digital, que por sua vez só pode variar entre os níveis "low" e "high". O que ocorre é que o Arduino é capaz de gerar sinal na forma de Pulse Width Modulation (PWM) com níveis variando em frequencia em "low" e "high". Os pinos digitais 3, 5, 6, 9, 10 e 11 possuem a saída PWM. Então para usarmos a saída PWM chamamos o comando:`analogWrite(pinNumber, value);` no qual `pinNumber` representa o pino digital capacitado para PWM e `value` designa um número entre 0 e 255 (variação de 0% à 100% do *duty cycle* do PWM).

Uma solução simples para converter sinal PWM em sinal analógico é utilizar um simples circuito RC em série, mas os valores de R e C devem ser projetados com cuidado, conforme veremos no experimento a seguir.

Experiência 8A – Leitura e escrita de dados no Arduino

Monte no protoboard o circuito mostrado na Fig. 2. Use $R = 1K\Omega$, $POT = 10K\Omega$, $C = 10\mu F$. Conecte o Arduino no PC usando o cabo USB. Implemente o código abaixo no Arduino.

- Abra a janela "Serial Monitor" dentro do Software IDE Arduino. Varie o cursor do potenciômetro e veja no "Serial Monitor" os numeros inteiro que correspondem as tensões de Entrada e Saída. **Preencha a tabela a seguir**.

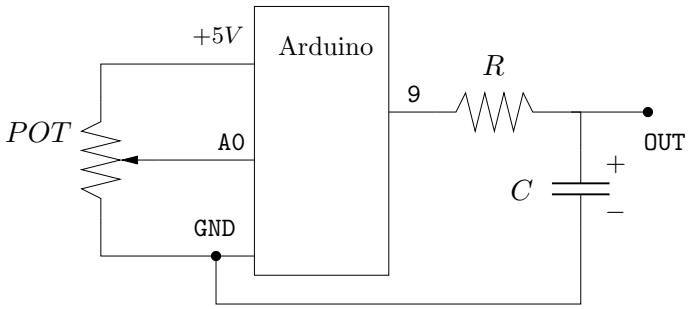

Figura 2: Esquemático do circuito do Arduino da Experiencia 8A.

```
// These constants won't change.  They're used to give names
// to the pins used:
const int analogInPin = A0;  // Analog input pin
const int analogOutPin = 9; // Analog output pin

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 10 milliseconds before the next loop
  delay(10);
}
```

tensão em A0	0V								5V
valor inteiro no Arduino correspondendo A0									
tensão em OUT									
valor inteiro no Arduino correspondendo OUT									

PROCEDIMENTOS DE SEGURANÇA

1. Desligue o módulo de alimentação.
2. Monte o circuito indicado e certifique-se de que todos os elementos seguem exatamente o diagrama indicado no experimento.
3. Após autorização do monitor ou professor, ligue o módulo de alimentação.
4. **Mudanças no circuito devem ser feitas sempre com o módulo DESLIGADO**.

Experiência 8B – Implementação de leitura e escrita de dados em malha aberta no servomecanismo

Monte o circuito mostrado na figura abaixo. Use $R_1 = 15K\Omega$, $R_2 = 1K\Omega$, $POT = 10K\Omega$, $C = 10\mu F$. A tensão de referencia é aquela aplicada no Pino `A0`.

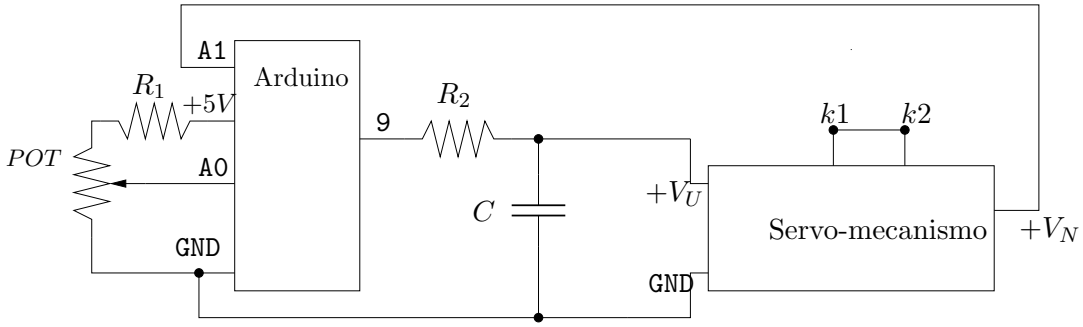

Diagrama esquemático do experimento de leitura-escrita em malha aberta usando o kit Servomecanismo

Experiência 8B - Tabela de tensão de entrada-saída $+V_u \times V_N$

$+V_U$										
$+V_N$										

Use o código-fonte da proxima pagina para preencher a tabela.

Código Arduino

Implemente no Arduino o código que realiza a **leitura entrada-saida**.

```
/*
  Codigo leitura entrada-saida
 */

const int analogInPinA0 = A0;
const int analogInPinA1 = A1;
const int analogOutPin = 9;

int sensorValue = 0;
float I=0; float Vu=0; float Vn=0; float Ref=0;

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPinA0);
  Ref = sensorValue * (5.0 / 1023.0);

  sensorValue = analogRead(analogInPinA1);
  Vn = sensorValue * (5.0 / 1023.0);

  Vu = Ref;

  analogWrite(analogOutPin, Vu * (255.0 / 5.0));

  Serial.print("Vu = " );
  Serial.print(Vu);
  Serial.print("\t Vn = ");
  Serial.println(Vn);
}
```

Experiência 8C – Implementação de estratégia de controle PI no Arduino

Mantenha o mesmo circuito da Experiencia 8B. Ajustando o codigo-fonte do Arduino podemos implementar um Controlador PI Digital.

Código Arduino

Implemente no Arduino o código que realiza o **Controlador PI Digital no Arduino**.

```
/*
  Codigo PI Arduino
 */

const int analogInPinA0 = A0;
const int analogInPinA1 = A1;
const int analogOutPin = 9;

int sensorValue = 0;
float I=0; float E=0; float Vu=0; float Vn=0; float Ref=0;
float Kp=1; float Ki=0.02;

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPinA0);
  Ref = sensorValue * (5.0 / 1023.0);

  sensorValue = analogRead(analogInPinA1);
  Vn = sensorValue * (5.0 / 1023.0);

  E=Ref - Vn;
  I=I + Ki * E;
  Vu = Kp * E + I;

  analogWrite(analogOutPin, Vu * (255.0 / 5.0));

  Serial.print("Ref = " );
  Serial.print(Ref);
  Serial.print("\t Vn = ");
  Serial.println(Vn);
}
```

Procedimento complementar

1. Abra a janela “Serial Monitor” no Software IDE Arduino.
2. A entrada de Referencia é dada pela tensão no ponto central do potenciômetro. Mova o potenciômetro e veja no “Serial Monitor” as tensões de Referencia e da velocidade angular V_N.
3. Faça o seguinte experimento: ajuste o potenciômetro para Ref=1 e mova-o abruptamente de modo que obtenha Ref=0.4.

(a) Determine o tempo de assentamento para $K_p = 1$ e $K_i = 0.02$.

(b) Determine o tempo de assentamento para $K_p = 1$ e $K_i = 0.05$.

(c) Determine o tempo de assentamento para $K_p = 1$ e $K_i = 0.10$.

(d) Altere com cuidado os valores de K_p e K_i de modo que o tempo de assentamento se aproxime de zero.

Preencha a tabela com os valores obtidos nos itens (a), (b), (c) e (d).

Item	Resposta
(a)	
(b)	
(c)	
(d)	

Relatório:

1. Apresente as tabelas das experiencias.

2. Descreva quais as vantagens e desvantagens observadas nas implementações da Estratégia PI na (i) placa analógica, (ii) PC com Matlab e (iii) Arduino. Faça um comparativo técnico entre essas três tecnologias.