Arduino 离散 PID 控制 simulink 仿真程序

谢业平

（机械工程 3 班 201810501055）

摘要：在 PC 中编写软件，运行一仿真程序，模拟一系统（例如温度控制系统，电机控制系统等），要求包含该系统的模型，以及控制接口。该控制接口能够接收来自串口的控制指令。在 Arduino 中编写控制程序，实现离散 PID 控制。要求该程序包括 PID 控制算法以及控制接口实现，该控制接口能够控制 PC 里的模型程序。分别运行上述实验系统，在 PC 端记录控制系统的状态曲线，绘制该曲线并机进行说明。本文采用 matlab 软件进行仿真，同时通过串口连接 arduino，实现 arduino 中编程，进行离散 PID 控制。

关键字：仿真；arduino；离散；控制

1、总体方案设计

为了实现仿真程序，采用 MATLAB 软件中的 simulink 进行仿真设计，并且可以与 arduino 串口相连，实现系统的电路仿真，同时设计对应的 MATLAB 程序。之后设计对应的 arduino 程序，实现电路的离散 PID 控制。最后对设计的产品进行调试，解决存在的问题。

1.1 仿真程序的设计与实现

本文主要是对电机 jie 控制系统进行仿真，实现电机的转速调节。仿真的电路图如下图所示：

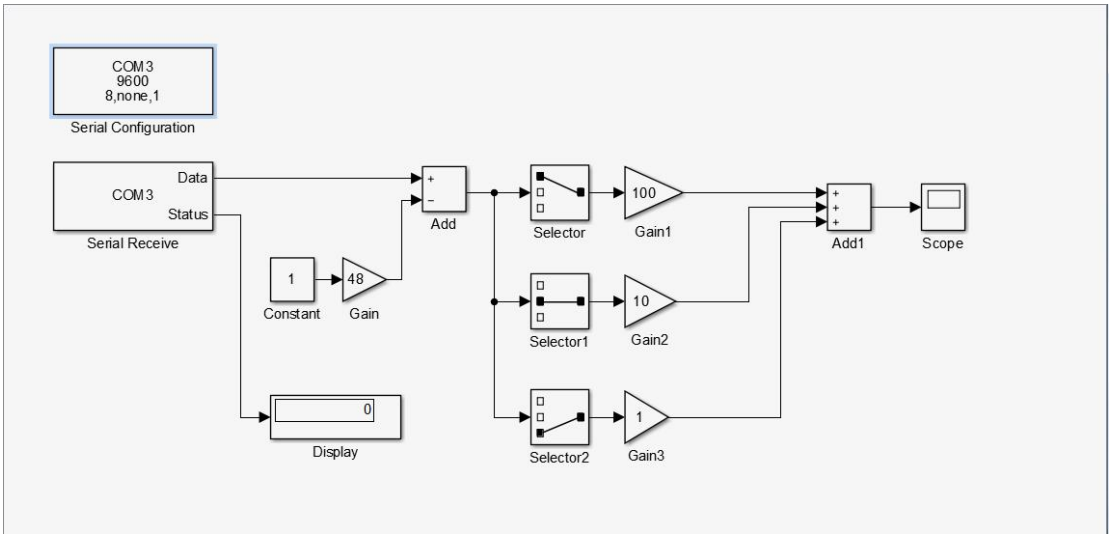

图 1 仿真电路示意图

1.2 离散 PID 控制设计与实现

在 arduino 中实现离散的 PID 控制过程，具体的结果如下图所示：

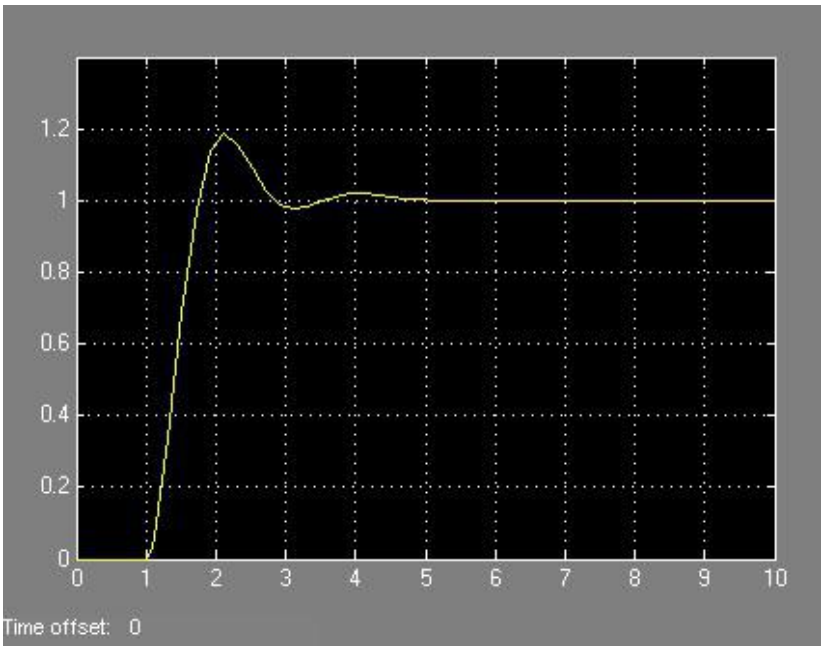

图 2 PID 整定后的阶跃响应曲线

2、方案的调试

在 arduino 中运行程序，显示出如下的结果：

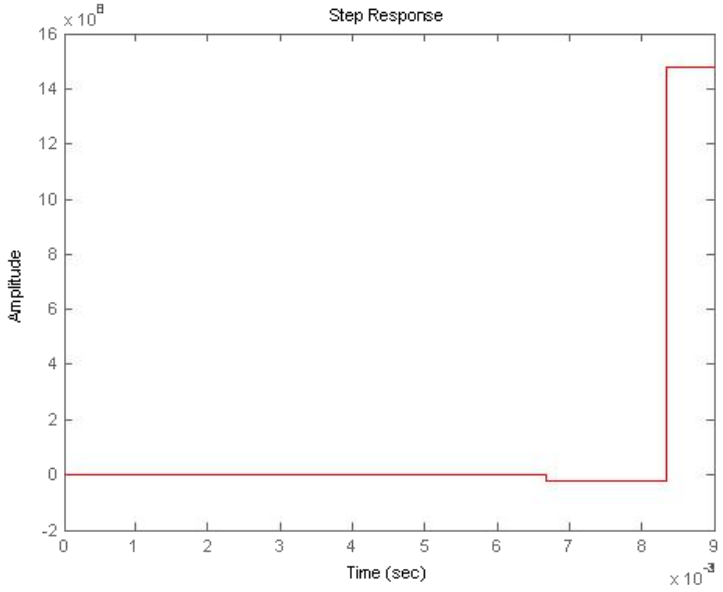

图 3　PID 校正后阶跃响应曲线如图

代码：

Matlab 仿真代码：

```
%建立系统的离散化模型
s=tf('s');
Gp=tf(55.85/((0.049*s+1)*(0.026*s+1)*(0.0167*s+1)));
Ts=0.00167;
Gpd=c2d(Gp,Ts); %连续系统离散化
%用根轨迹法找出临界值 Kcr 和 Wcr
figure(1)
clf %清除当前图形
rlocus(Gpd); %绘制根轨迹
[K,Poles]=rlocfind(Gpd); %从根轨迹确定临界点对应的增益和极点
Kcr=K;
Wcr=angle(Poles(1))/Ts;
Tcr=2*3.14/Wcr;
%设计 PID 控制器
%按表中公式确定参数 Kp，Ti，Td
Kp=0.388*Kcr;
Ti=0.5*Tcr;
Td=0.125*Tcr;
%按 PID 控制器模型确定 Ki 和 Kd
Ki=Kp*Ts/Ti;
Kd=Kp*Td/Ts;
disp('PID 参数 Kp,Ki,Kd 分别为:')
Kp
Ki
Kd
%建立 PID 控制器的离散化模型 Gcd(s)
z=tf('z',Ts)
Gcd=Kp+tf(Ki*z/(z-1))+tf(Kd*(z-1)/z);
%检验 PID 控制器的性能
Gd=Gpd*Gcd;
Gclose=feedback(Gd,1);
figure(1)
```

```
clf
step(Gclose,'r')
```

运行的结果如下:

```
selected_point =
-3.2749 - 0.0559i
```

PID 参数 Kp,Ki,Kd 分别为:

```
Kp =
1.7037e+004
Ki =
1.7045e+004
Kd =
4.2570e+003
Transfer function:
z
Sampling time: 0.00167
```

Arduino 程序代码:

```
#include <PID_v1.h>
#define PIN_INPUT 0
#define PIN_OUTPUT 3
//Define Variables we'll be connecting to
double Setpoint, Input, Output;
//Define the aggressive and conservative Tuning Parameters
double aggKp=4, aggKi=0.2, aggKd=1;
double consKp=1, consKi=0.05, consKd=0.25;
//Specify the links and initial tuning parameters
PID myPID(&Input, &Output, &Setpoint, consKp, consKi, consKd, DIRECT);
void setup()
{
  //initialize the variables we're linked to
  Input = analogRead(PIN_INPUT);
  Setpoint = 100;
  //turn the PID on
  myPID.SetMode(AUTOMATIC);
}
```

```
void loop()
{
  Input = analogRead(PIN_INPUT);
  double gap = abs(Setpoint-Input); //distance away from setpoint
  if (gap < 10)
  {  //we're close to setpoint, use conservative tuning parameters
    myPID.SetTunings(consKp, consKi, consKd);
  }
  else
  {
     //we're far from setpoint, use aggressive tuning parameters
     myPID.SetTunings(aggKp, aggKi, aggKd);
  }

  myPID.Compute();
  analogWrite(PIN_OUTPUT, Output);
}
```