Servo and DC Motor Control using Desktop Application

Arindam Banerjee, Sayanta Banerjee, Srinjon Sadhukhan

JIS College of Engineering, Kalyani, Nadia, West Bengal, India

Abstract: In this paper, our main aim is to automate servo and dc motors using desktop application for industrial automation. Here we have designed a desktop application using python and interfaced the application with the microcontroller. As microcontroller, Arduino Uno ATMEGA328P/ Mega 2560 has been used to automate servo and dc motors. These motors have wide application in industry where robotic arm or conveyer belts are used in the production unit. This application reduces human efforts and the accuracy in production can be enhanced.

Keyword: Desktop application, Servo Motor, DC Motor, Python, Tkinter, Arduino Uno/ Mega microcontroller

1 INTRODUCTION

Introduction:

In earlier ages, manual control of the machines in the industry required large man power and tremendous human efforts were needed to operate them. In the era of automation, these efforts have been reduced to a large extent by introducing robots and automotive machines in the industry. In earlier days, the accuracy in production was a major challenge due to manual operation which has been greatly improved with the help of automation. With the technological advancement, many industries now use desktop applications for automation and monitoring [1-4]. How industrial automation can be accelerated has been discussed in [5].

Like industrial technologies described in [1-4], we have also focused on the application based hardware control and monitoring. In most of the industries, there is mechanical operation for controlling dc motors and servo system. Here our aim is to control the motors electronically by using a desktop application. There is computing system which sends the control data to the microcontroller. Microcontroller processes the data and sends command to the motors accordingly.

Servo Motor Operating Principle:

Servo motor is an electromechanical system which changes the arm connected to the motor by changing the angle. Servo motor is operated by magnetic flux which is controlled by the electrical pulse applied to it. If the pulse width is changed and the amount of magnetic flux generated is also changed. This method is called **pulse width modulation (PWM)**. If the pulse width is large then the amount of driving current to the motor circuit is quite large which increases the magnetic flux in the circuit. Thus the angle of the servo motor is changed proportionately. Similarly, small pulse affects the flow of current to the circuit and the magnetic flux is reduced to change of angle.

If T is the period of a pulse and T_1 is the time interval for the pulse to be high then T_1 is called the pulse width of the signal. $\left(\frac{T_1}{T} \times 100\right)$ is called the **duty cycle** of a pulse. If the duty cycle is less than 50% then the current to the magnetic circuit is small because the magnetic flux is directly proportional to the triggering angle which is basically the pulse width discussed above. Similarly if the duty cycle is greater than 50% then the current is large which results in high magnetic flux and the current is large accordingly. The following equation shows the relation between magnetic flux (Φ) and magnetic current (I).

$$\Phi = nBA = n\mu HA = n\mu A\,(I/L) \quad (1)$$

where, n=no. of turns in the motor, B=magnetic field, H=magnetic intensity, μ=permeability of the medium, A=cross sectional area of the closed loop wire, L=length of the wire. Thus from eq nagnetic flux is i nagnetic current n of the servo m ion with the chan t the angle of ro of pulse width.

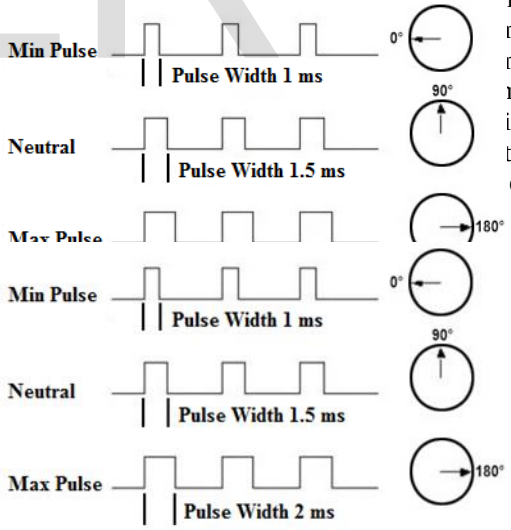


Fig. 1 Change of angle of rotation with the change of pulse width for servo motor

The period of the repetition pulse sent to servo motor is 20ms as set by the manufacturers. It has a tick event which triggers the motor to rotate. The change in pulse width per degree rotation is $\left(\frac{1}{180} \times 1000 = 5.5\right)$ microseconds. Thus for **Neutral** position of the motor, the default pulse width is $(90 \times 5.5 = 495)$ microsec-

onds.

DC Motor Operating Principle:

DC motor is also an electromechanical system which is operated by electrical pulses. Like servo motor, speed of dc motor can also be controlled by changing the amount of magnetic flux applied to it. The commutator and the brush help the flow of current in alternative direction when an electrical pulse is applied to the motor. Its speed is also controlled by PWM technique discussed earlier. The applied voltage and the magnetic flux are related as follows.

$$\Phi = VT \qquad (2)$$

where, T is the applied pulse width. Now, we know that Φ=nBA=nμHA=nμA(I/L). Thus we get the relationship between the applied voltage and the magnetic current, which flows through the armature of the motor, as shown below.

$$I=VTL/(n\mu A) \qquad (3)$$

DC motor is connected in H-bridge configuration for bidirectional movement. In H-bridge, there are four switches connected at the two arms and the motor is connected in the middle of it. Fig. 2 shows the es are on to make the mo .

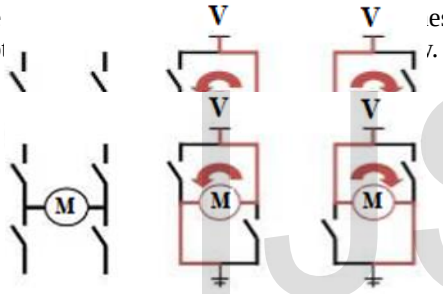


Fig. 2 H-Bridge connection for dc motor

Proposed Technique (application development):

The proposed technique is based on the creation of application to set the values of the angle of servo motor and DC motor. The application is desktop based and it has been designed using Python language. Next, the application has been interfaced with the microcontroller and the motors have been controlled by the microcontroller by taking the values from the application. We have created a user interface where two Sliders (Scale) have been used to control the servo and the dc motors.

Fig. 3 The desktop application to control the servo motor and the DC motor

An application has been designed using python to control the servo and the DC motor. The first slider is used to set the angle of the servo motor. The second slider is used to control the speed of the DC motor. The first slider has the angle range from 0^0 to 180^0. The speed of the motor can be mapped into 8 bit representation whose numeric range is from 0 to 255. The maximum speed is achieved when the speed value is set to 255. The application in Python language has been developed using the **Tkinter** library which is useful to design graphical user interface. For communication with the microcontroller another library has been used which is called **Pyserial**. This library establishes serial communication between between the software IDE and the hardware. In this library, the port number and the baud rate for the data transmission is set by which the device, that is the microcontroller, is identified. The design algorithm is given below.

Algorithm:

Step 1: Create a label for the servo motor using the Label() method in-built in Tkinter.

Step 2: Create a slider for the servo motor control using the Scale() method.

Step 3: Create another label for the dc motor.

Step 4: Create another slider for the dc motor speed control.

Step 5: Create a button using the Button() method and call a function which takes the values from the sliders, convert into string and send it to the microcontroller when the button is clicked.

```
m=x.get()
n=y.get()
p=str(m)+','+str(n)
board.write(p.encode())
print(p)
```

```
tk=Tk()
tk.winfo_toplevel().title('Automation App')
tk.geometry('370x370')
tk.resizable(True,True)
lbl1=Label(tk,text='Set servo angle:',fg='red')
lbl1.grid(row=0,column=0)
```

File Edit Format Run Options Window Help

```
from tkinter import *
import serial
board=serial.Serial('COM18',9600)


def setvalue():
    m=x.get()
    n=y.get()
    p=str(m)+','+str(n)
    board.write(p.encode())
    print(p)



tk=Tk()
tk.winfo_toplevel().title('Automation App')
tk.geometry('370x370')
tk.resizable(True,True)
lbl1=Label(tk,text='Set servo angle:',fg='red')
lbl1.grid(row=0,column=0)
x=IntVar()
spb1=Scale(tk,from_=0,to=180,variable=x,orient=HORIZONTAL)
spb1.grid(row=0,column=1)
lbl2=Label(tk,text='Set dc motor speed:',fg='red')
lbl2.grid(row=1,column=0)
y=IntVar()
sld1=Scale(tk,from_=0,to=255,variable=y,orient=HORIZONTAL)
sld1.grid(row=1,column=1)
btn1=Button(tk,text='Set Value',fg='green',command=setvalue)
btn1.grid(row=2,column=1)
tk.mainloop()
```

Fig. 4 The python program for generation of graphical user interface using Tkinter

Fig. 4 shows the complete program to generate the user interface for the servo and dc motor control. When the **mainloop()** method for the Tkinter object is executed then the window shown in Fig. 3 is displayed. For serial communication, the data is converted to string and then the string is converted to byte using the **encode()** method which is based on the Unicode Transformation Format (UTF-8). This format converts alphanumeric characters to byte or byte to alphanumeric characters.
After the execution of the program, when the application window is open, then set different values of the sliders. If we want to print the values of the sliders then it looks like the figure shown below.

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\4th_year_project_2019_2020\servo_control.py ==========
148,101
```

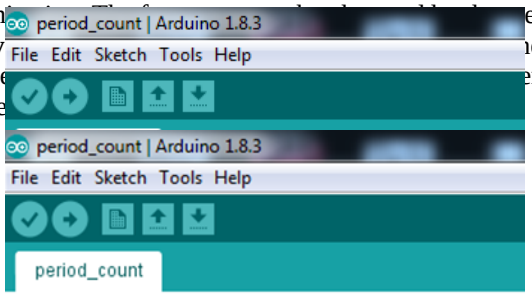
```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\4th_year_project_2019_2020\servo_control.py ==========
148,101
>>>
========== RESTART: D:\4th_year_project_2019_2020\servo_control.py ==========
90,225
>>>
```

Fig. 5 Slider values are shown in the Python Idle console when the button is pressed

Fig. 5 shows the values of the sliders at the Python Idle console after the button is pressed.

Development of hardware program:

In this project, we have used Arduino Mega 2560 board for programming. Arduino Uno ATMega328P board can also be used. Arduino is run by 16MHz frequency which is given by a crystal oscillator. When the board is connected to the computer where the Python program runs then in the Arduino IDE, the port number is shown which is given in the python program for serial communi… …ethod **clockCy**… …e clock cycles pe… …erve the clock fre…




```
int x;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
x=clockCyclesPerMicrosecond();
Serial.println(x*1000000);
delay(100);
}
```

Fig. 5 Program to observe the frequency of the microcontroller

If the output of the above mentioned function is multiplied by 10^6 then the actual clock frequency is obtained which is shown in Fig. 6. Here "**Serial.begin()**" function initializes the serial communication between the microcontroller and the device connected to it. The IDE has its own connection with the serial monitor via serial communication. The argument, inside the function i.e. 9600, is the baud rate which is defined as the rate of change of symbols per second.

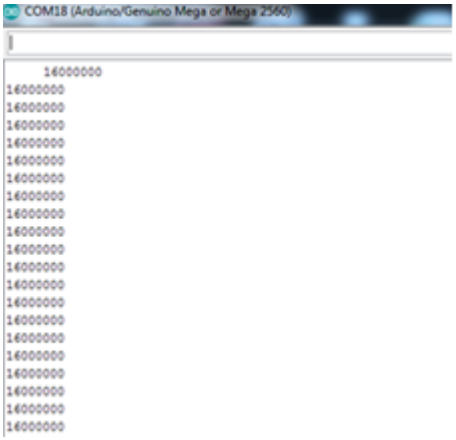


Fig. 5 Observation of frequency

Servo motor is attached with any of the PWM pins which is indicated on the board. The pin generates PWM output which sets the angle of the motor. The pulse width of the signal is fixed by the integer input sent by the python application. Similarly, to control the speed of the dc motor, the motor is connected to PWM pin and the value of the speed is set by the python application. The Embedded C program for Arduino has two main functions: (1) **setup()** and (2) **loop()**. "**setup()**" function is executed once whereas "**loop()**" function is executed for indefinite time. In the "**setup()**" function, the input and the output ports are declared. In the "**loop()**" function, the main program is executed. Fig. 6 shows the program for servo motor operation.

servo_test_arduino | Arduino 1.8.3

File Edit Sketch Tools Help

servo_test_arduino

```
#include<Servo.h>
Servo servo;
int j;
String b;
void setup() {
  // put your setup code here, to run once:
servo.attach(3);
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
if(Serial.available())
{
  b=Serial.readString();
}
j=b.toInt();
servo.write(j);
delay(100);
}
```

Fig. 6 Embedded C program for servo motor operation

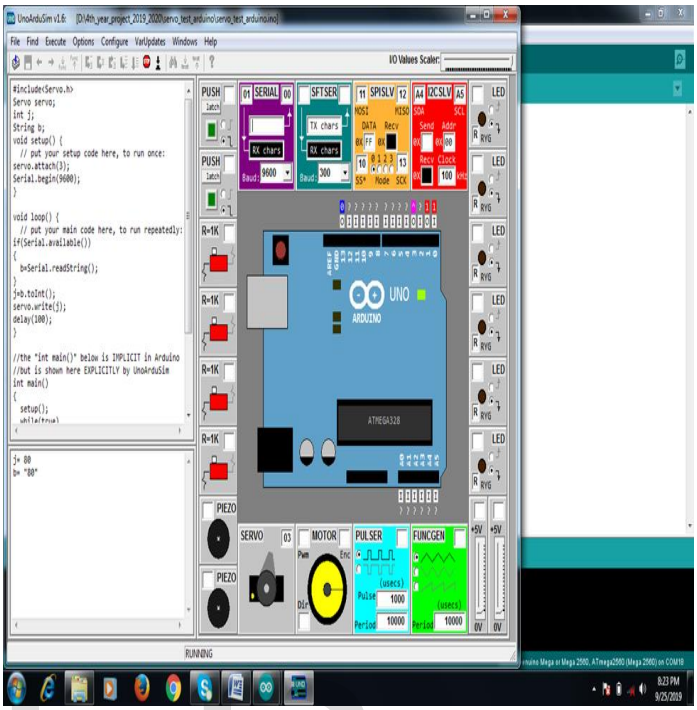


Fig. 7 Verification of the program by simulation

Fig. 7 shows the simulation program window to verify the program of servo motor operation.

dc_motor_test | Arduino 1.8.3

File Edit Sketch Tools Help

dc_motor_test

```
const int dcm=5;
int j;
String b;
void setup() {
  // put your setup code here, to run once:
pinMode(dcm,OUTPUT);
Serial.begin(9600);
analogWrite(dcm,0);
}

void loop() {
  // put your main code here, to run repeatedly:
if(Serial.available())
{
  b=Serial.readString();
}
j=b.toInt();
analogWrite(dcm,j);
delay(100);
}
```

Fig. 8 Embedded C program for dc motor test

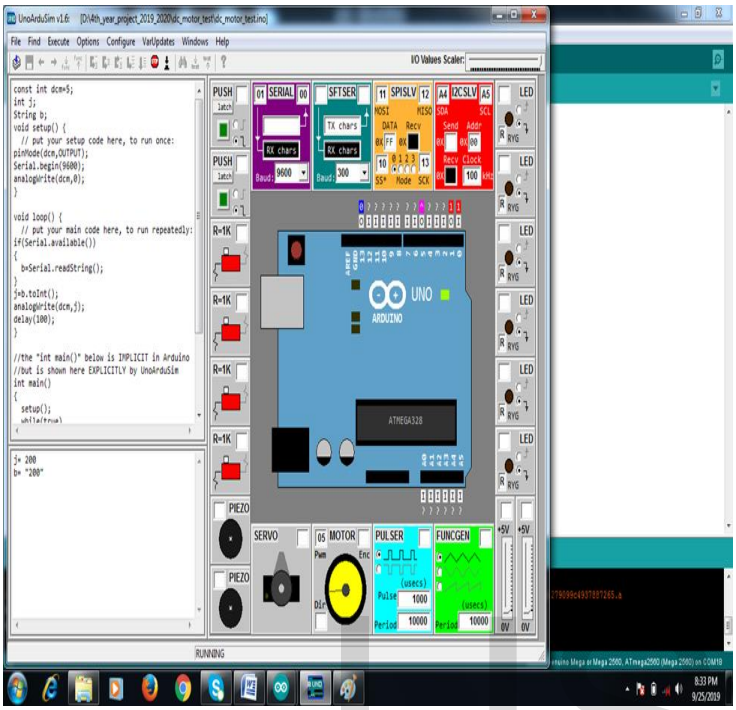


Fig. 8 Verification of the program for dc motor test

Figs. 7 and 8 show the Embedded C program for dc motor test and the verification of the program by simulation. Here all the programs have been verified using **UnoArduSim** simulator

Complete program for servo motor and dc motor operation in Embedded C:

```
#include<Servo.h>
Servo servo;
int a=2;
int j,k;
String b="120,170",c="",d="";
const int dcm=5;
void setup() {
pinMode(dcm,OUTPUT);
Serial.begin(9600);
servo.attach(3);
}
void loop() {
if(Serial.available())
{
  b=Serial.readString();
}
k=b.length();
for(int i=0;i<k;i++)
{
  if(b[i]==',')
  {
    j=i;
  }
}
for(int i=0;i<j;i++)
{
  c+=b[i];
}
for(int i=j+1;i<k;i++)
{
  d+=b[i];
}
Serial.println(c+' '+d);
servo.write(c.toInt());
analogWrite(dcm,d.toInt());
c="";
d="";
delay(1000);
}
```

Experimental Result:

Fig. 9 Servo motor angle movement to 0^0

Fig. 10 Servo motor angle movement to 90^0

Fig. 11 Servo motor angle movement to 180^0

Figs. 9 to 11 show the angle movement of servo motor using the desktop application which has been created in Python language and shown in Fig. 3. Figs. 9 to 11 show the movement of the servo motor for the slider positions 0^0, 90^0 and 180^0 respectively.

Conclusion: In this paper, desktop application based industrial automation technique was shown. Basically we operated the different motors like servo and dc motors using the Python based desktop application. As per the literature survey, this kind of desktop based hardware control operation was new to best of our knowledge. Our next target would be the monitoring the function of different sensors using the desktop based application.

References:

[1] https://www.lanner-america.com/blog/industrial-pc-applications-industrial-automation/
[2] https://www.automation.com/automation-news/article/applying-apps-in-industrial-automation
[3] https://www.britannica.com/technology/automation/Manufacturing-applications-of-automation-and-robotics
[4] http://www.ti.com/applications/industrial/factory-automation/overview.html
[5] Tushar Jain and Meenu, "Automation and Integration of Industries through Computer Vision Systems", International Journal of Information and Computation Technology, Vol. 3, no. 9, 2013, pp. 963-970