PI controller for DC motor speed realized with Arduino and Simulink

Mario Gavran*, Mato Fruk** and Goran Vujisić**

* Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia
**University of Applied Sciences - Department of Electrical Engineering, Zagreb, Croatia
mario.gavran@student.um.si, mato.fruk@tvz.hr, gvujisic@tvz.hr

ABSTRACT – In this paper we describe a technical system for DC motor speed control. The speed of DC motor is controlled using Arduino programming platform and MATLAB's Simulink coder. This paper contains introduction to using an Arduino board and Simulink PI controller in closed loop system. It will be described how to program Arduino with Simulink coder and in the end we present the results of PI controller for DC motor speed will be given.

I. INTRODUCTION

Short settling time and minimized steady state error are desired in technical system of speed controlled DC motor. To accomplish these goals, closed control loop must contain a PI controller, DC-DC power converter and a negative feedback/speed sensor. Testing was done on laboratory model of small DC motor coupled with DC generator that is used as a load on the motor. Model is shown in Fig. 1.

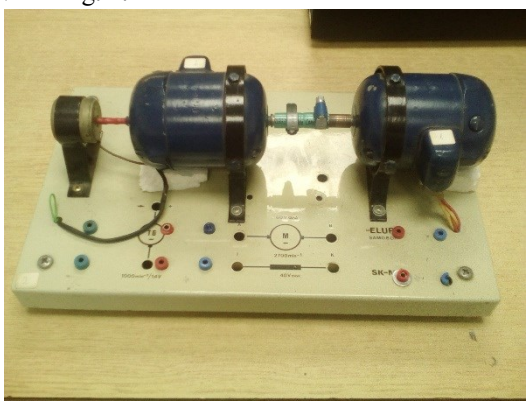

Figure 1 Laboratory model of DC motor

That model also includes the tachogenerator that was used as negative feedback/speed sensor. Arduino Uno board was used as controller in this closed loop. To accomplish short settling time and a small steady-state error regulator have to be adjusted to the rest of the closed loop.

An Arduino board is used due to low cost, simplicity and flexibility. MATLAB Simulink is used to create control algorithm and convert that algorithm in C++ code that can be uploaded in Arduino board. PC computer with Windows OS is required to make the model of controller in Simulink and to upload that model on Arduino board.

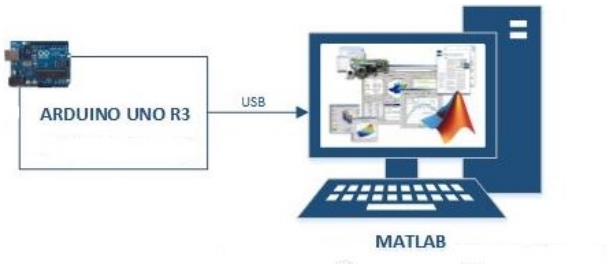

Figure 2 Aduino Uno and Simulink

II. ARDUINO UNO

Arduino Uno is microcontroller board with Atmega328p that has been used as a digital PI controller. This board is used to test the controller. It has 14 digital input/output pins, and 6 of them can be used as a PWM output pins. It also has 6 analog inputs, 16MHz clock crystal, a USB connection that is used to connect it to the PC and it operates at 5 Volts. Various Arduino boards can be used instead of Arduino Uno board.

Arduino Board	Shield Support	Interactive Tuning and monitoring	Comments
Arduino Due*	Y	Y	DAC and CAN channels not currently supported.
Arduino Uno*	Y	N	
Arduino Leonardo*	Y	Y	
Arduino Mega 2560*	Y	Y	
Arduino Mega ADK*	Y	Y	
Arduino Micro*	N	N	
Arduino LilyPad USB	N	N	
Arduino Esplora	N	N	Additional IO supported via analog multiplexer
Arduino Robot	N	N	Additional IO supported via analog multiplexer
Arduino Mini* (ATmega328)	N	N	Mini with ATmega168 not supported
Arduino Nano 3.X* (ATmega328)	N	N	Nano 2.X with ATmega168 not supported
Arduino Pro* (ATmega328)	N	N	Pro with ATmega168 not supported
Arduino Fio	N	N	
Arduino Ethernet Shield			See Shield Support column for compatibility
Arduino WiFi Shield			See Shield Support column for compatibility

Figure 3 Arduino boards compatible with Simulink

III. ARDUINO AND SIMULINK

Arduino boards are usually programed by writing C/C++ code in Arduino IDE window, but in this example it will be programmed using MATLAB Simulink package for Arduino. Simulink is a block diagram based environment for simulating mathematical models and it can be used to program some of the Arduino boards. To use Simulink with Arduino a support package is needed. It can be downloaded from Mathworks website with instructions how to install it and start using it. After installation is done correctly, a new library is now ready for use in Simulink library browser. In this paper analog input, digital input and PWM output blocks will be explained and shown.

IV. TESTING ARDUINO AND SIMULINK

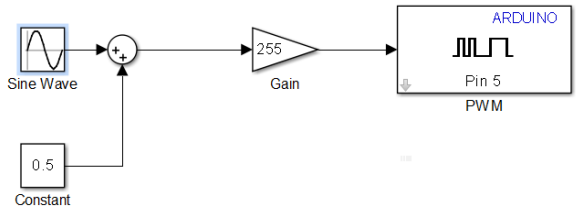

Figure 4 Block diagram of sine wave generator

This section explains how to use Simulink blocks to read analog signals from real world and how to use PWM output signals to control some kind of electric device. Figure 4 shows simple block diagram made in Simulink. Its purpose is to generate a sine wave signal from Arduino using only Simulink. First block is a sine wave block. This block outputs sine wave signal of amplitude 0.5 and frequency of 10Hz. In sum block that signal is added to the constant of 0.5. Sum block now outputs sine wave that has "DC" component of 0.5, and amplitude of 0.5. Because Arduino Uno does not have analog output, the PWM output is used with corresponding RC filter. That signal is connected to the gain block of 255 value of gain, and outputs sine signal that has amplitude of 255/2.

In the PWM block, pin number needs to be entered. The PWM block will generate a PWM signal on the specified pin. The duty cycle of that PWM signal depends on input signal. In this case pin 5 is used and generated sine wave signal with amplitude of 255 is on input. When that block diagram is compiled and uploaded in Arduino Uno board by clicking on "build model" icon (normal mode), on pin 5 of the Arduino board you can measure PWM signal and by using RC filter that PWM signal can be filtered and that is shown on Fig 5.

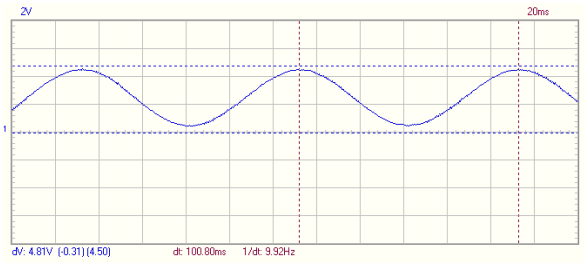

Figure 5 Sine wave generated with Arduino

For demonstration how to use analog input block and Arduino Uno and Simulink in real time, another block diagram is uploaded on Arduino board (Fig. 6) but this time in "external mode". "External mode" is used when you have to monitor any of the signals in block diagram.

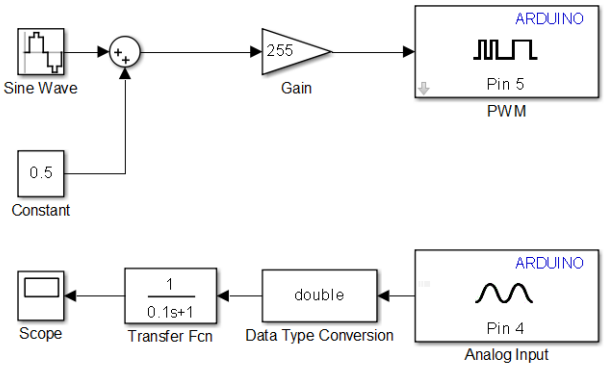

Figure 6 Block diagram of sine wave generator and analog read

Top part of this diagram is the same as on Figure 4. This time analog input block is defined by selecting analog pin 4 as an input pin. For demonstration PWM pin 5 is connected with analog input pin 4 via the RC filter. In this example Arduino generates sine wave signal and reads that same signal. Transfer Fcn block simulates the same RC filter but represented as first order filter with time constant of 0.1 s. When diagram is uploaded using external mode on scope block generated sine wave signal can be monitored as shown in Fig 7. Input block converts input voltage in uint16 type number in range from 0 to 1023.

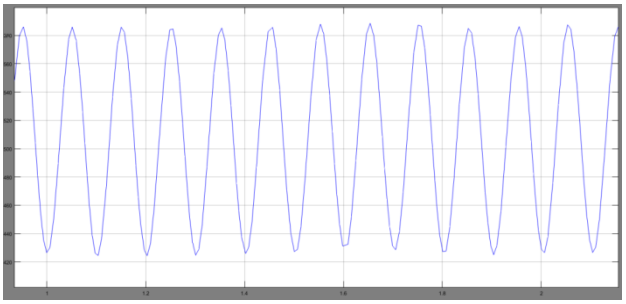

Figure 7 Sine wave read by Arduino

V. TESTING THE CONTROLLER WITH DC MOTOR

In order to apply the controller on DC motor model, the parameters of the controller need to be set. The parameters of the controller depend on the rest of closed loop components, so transfer functions must be known. Transfer functions of the motor, DC-DC power converter and speed sensor characteristics must be known to determine their transfer functions.

The motor is tested on load and data is collected. On Fig. 8 load characteristics is shown. Moment of inertia is determined with motor stopping experiment.

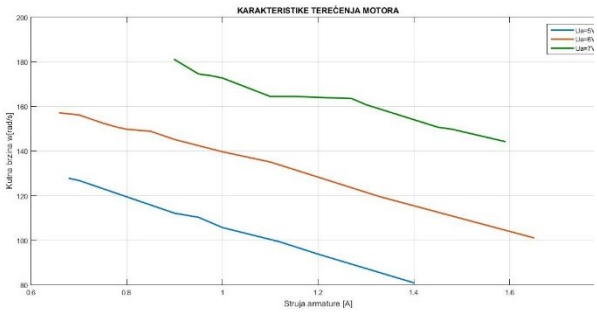

Figure 8 Load characteristics

Resistance of armature coil is determined as well as motor constant "k". In Fig. 9 time response of armature circuit is shown and inductance of armature coil is determined.

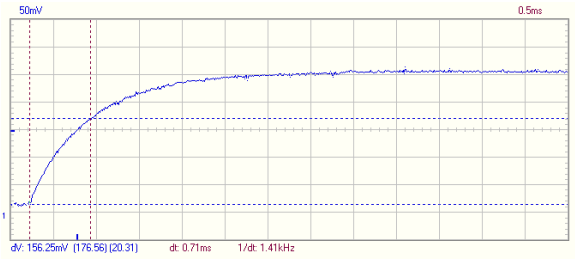

Figure 9 Time response of armature coil

Motor now can be simulated in Simulink using block diagram shown in Fig. 10.

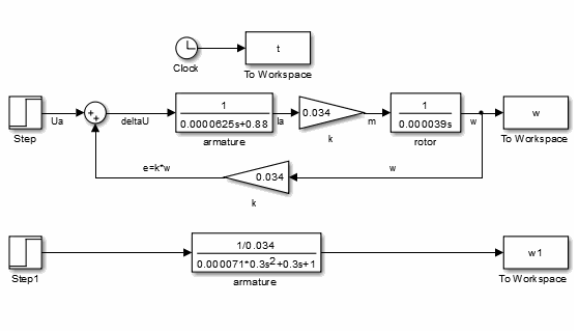

Figure 10 Simulation model of used motor

Because Arduino Uno does not have analog output, the PWM output is used and therefore chopper is used as DC-DC power converter. This is convenient because an RC filter is not needed to convert PWM to analog signal. DC-DC converter i.e. the chopper is shown in Fig. 11.

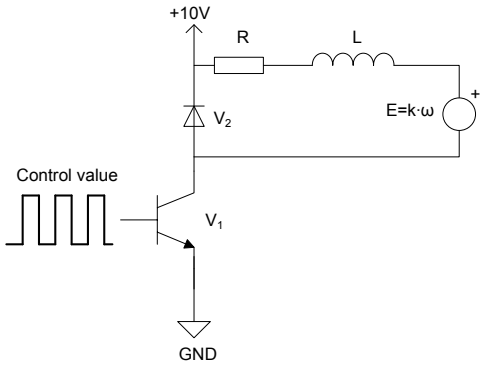

Figure 11 chopper used to power the motor

Transfer function of the speed sensor will be approximated by the first order filter.

Control block diagram is shown in Fig. 13. PI control algorithms will be tested and PID block is used. In PID block window various parameters can be changed. In "controller" drop menu, type of controller can be selected, and in the main window proportional and integral constants can be changed. In "form" drop menu window ideal or parallel algorithms can be selected. In Fig. 12 the function block parameters for PID block are shown.

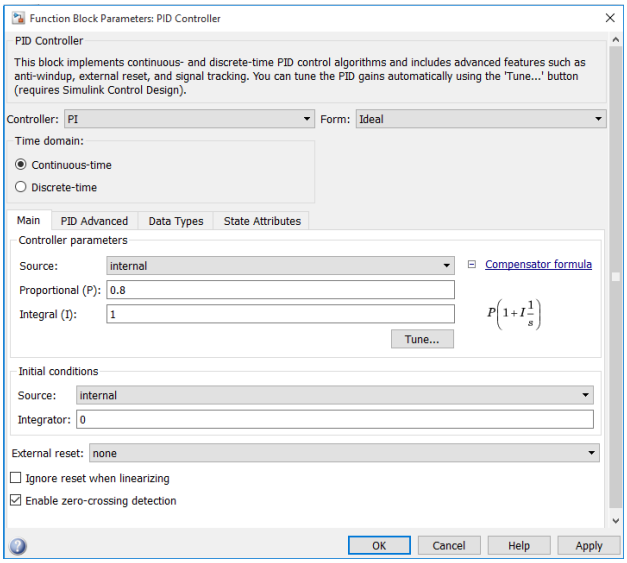

Figure 12 PID controller block parameters

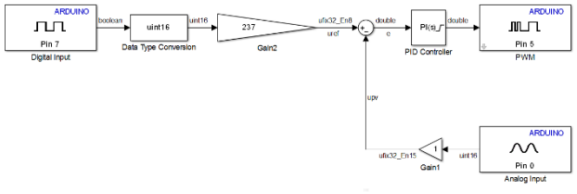

Figure 13 Block diagram of the controller

Controller diagram (Fig. 13) represents how reference and negative feedback are accomplished and used as input signals to the controller itself. Speed sensor is connected to the analog input pin 0. 5V pin on Arduino board is connected to the digital input pin 7 via the button and resistor. When button is pressed on pin 7, 5 volts appears. Speed sensor that is mounted on the motor shaft gives 1,14V for the speed of 1000 RPM. Analog input block convert voltage that is connected to the pin 0 in to number, so data conversion block is used to adapt number type. In this case it is 237. In order to initialize the motor speed of 1000 RPM the reference value has to be 237 so Gain2 block has value of 237.

When button is not pressed motor is not spinning. When button is pressed digital input block outputs 1 and multiplies it by 237. Still motor is not spinning so speed sensor connected to analog input gives 0V and value that analog input block gives is also 0. Now PID controller block has 237 on its input and starts to give signal to the PWM block. PWM output pin is connected to the DC-DC power converter and motor starts to spin. Depending on the controller parameters the motor will get to 1000 RPM speed and input in the PID block now is 0 and PI controller will maintain that speed of the motor. If motor gets more than enough energy from DC-DC converter it will overshoot wanted speed and input in the controller will be negative, so the controller will slow the motor down.

Fig. 14 shows the time response of the motor speed. Speed is measured with the speed sensor that is already mounted on the motor. For example, wanted speed will be 1000 RPM, which means the reference gain must be set to 237. Speed will be recorded using the speed sensor. Because of that speed will be represented with voltage. Speed sensor that is used for these tests has gain of 1,14 mV/RPM.

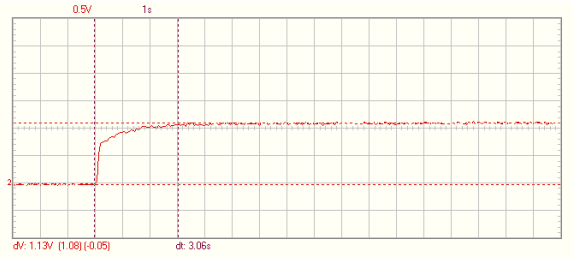

Figure 14 Motor speed time response

Steady state speed is around 1000 RPM as shown in Fig. 14. The time response graph is recorded with unloaded motor and using PI controller that is set to parallel form, proportional gain (P) set to 1 and integral gain (I) set to 1. In Fig. 15 parallel model of controller is shown.

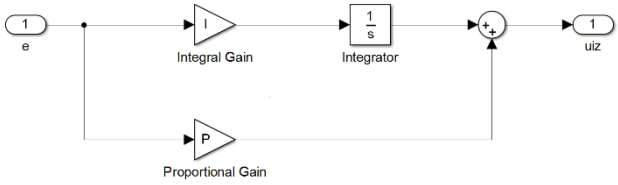

Figure 15 parallel model of controller

In this example, where P=1 and I=1, the motor will get to 1000 RPM in around 3 seconds. To test the speed response on load disturbance a generator coupled with motor is used. Motor will be spinning at 1000 RPM and the load will be switched on. When controller compensates that load disturbance and gets up to 1000 RPM again, the load will be switched off, as shown in Fig. 16.

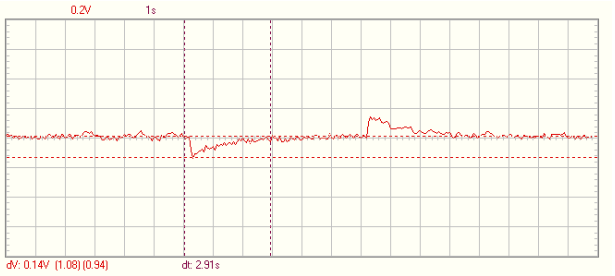

Figure 16 Motor speed response with load disturbance

It takes 3 seconds to get back to wanted motor speed. To get more dynamic response of the motor controller parameters must be changed. Speed and load disturbance time response is shown in Fig. 17. In that case the proportional gain is set to 1 and integral gain to 20.

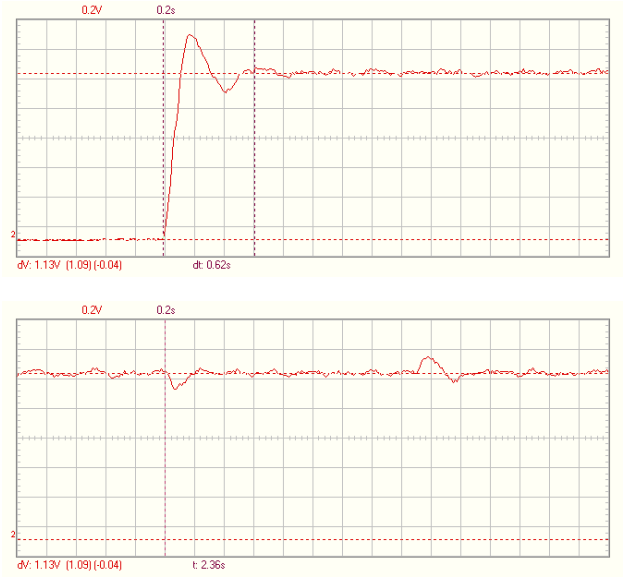

Figure 17 Speed and load disturbance response

With this set up of controller the speed response is underdamped but much faster. In this case motor spins up to 1000 RPM in 0,62 seconds and time needed for the controller to compensate the load is 0,36 seconds.

All tests of the controller were done in normal simulation mode and all sample times were the same. Sample time is a very important parameter of the model and must be set at 0.0001 s for examples shown in this paper. This model of speed controlled motor will be used in laboratory exercises, and so it is convenient to use external simulation mode. In this mode user can monitor any of the signals in the model. For example in Fig. 18 and Fig. 19 two signals are being monitored in real time.

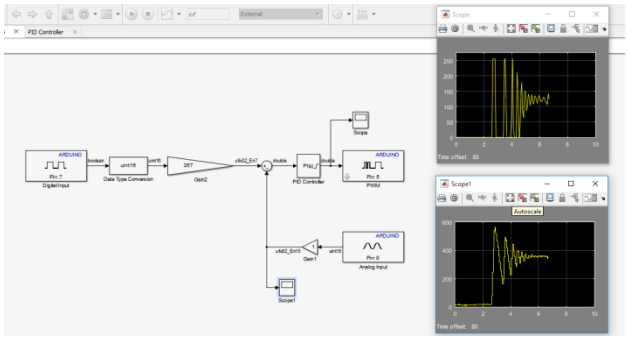

Figure 18 External mode with controller

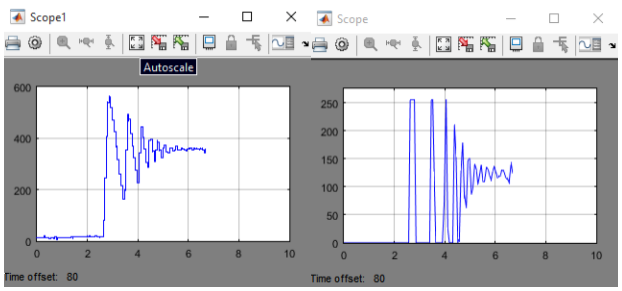

Figure 19 Scopes in real time working in external mode

VI. CONCLUSION

Using Simulink and Arduino is one of the simplest ways to make satisfying controller for many purposes in variety of technical systems. Tests that were done in this paper show that Arduino can be very flexible for programming a control algorithms using MATLAB's Simulink. One purpose of this kind of controller is using it in laboratory exercises where students can easily modify controller's parameters and see what is going on with output value of the system that they are studying. Sample time of this controller can be changed and adapted to the system requirements. Using external mode, parameters of the controller can be changed in real time and user can see what has changed in the output signal almost instantly. The model was already tested on laboratory exercises and it has yielded good results. The students were very interested how to program the controller with Simulink.

REFERENCES

1. Arduino website: https://www.arduino.cc/en/Main/ArduinoBoardUno
2. http://www.mathworks.com/, 29.7.2016.
3. Perić, Nedjeljko. Automatsko upravljanje-predavanja, FER, Zagreb, 2006.
4. https://www.fer.unizg.hr/_download/repository/SIMULINK_SKRIPTA.pdf, 10.1.2015.
5. Fruk, Mato; Vujisić, Goran; Tikvić, Ivan. CONTROL OF THERMAL PROCESS WITH SIMULINK AND NI USB-6211 IN REAL TIME // MIPRO 2013, Proceedings, Computers in Education – CE, Opatija, 2013, 697-702
6. Fruk, Mato; Vujisić, Goran; Špoljarić, Tomislav. Parameter Identification of Transfer Function Using MATLAB // MIPRO 2013, Proceedings, Computers in Education – CE, Opatija, 2013, 697-702