A microcontroller-based software framework for controlling a mechatronic system

Nikolaos Nikolaidis[1], *Nikolaos Evgenidis*[1,*], *Dimitrios* Bechtsis[1], *Fotis* Stergiopoulos[1], *Apostolos* Tsagaris[1], *Dimitrios* Triantafyllidis[1], *Asterios* Papaoikonomou[2] and *Anastasios* Filelis[2]

[1]International Hellenic University (IHU), Department of Industrial Engineering & Management, PO Box 141, Sindos, Thessaloniki, 57400, Greece
[2] Evresis S.A DA 12a Block:39b Industrial Area, 57400, Sindos, Thessaloniki

Abstract. The proposed software framework is presented and an Application Programming Interface (API) is developed based on the Arduino Mega 2560. The API processes external commands that follow the operational logic of a gel electrophoresis device. The API acts as an intermediary layer between the gel electrophoresis mechatronic system's microcontroller and the motors' controllers. The microcontroller enables the basic functionalities of the gel electrophoresis system while the use of 2 axis (X, Z) motor controllers is necessary for controlling the moving parts of the mechatronic system. We control the movement's direction, position, speed, and acceleration. The developed API controls the stepper motors drive axles and the DC motors for opening and closing the drawers and other moving parts of the mechatronic system.

1 Introduction

An Application Programming Interface (API) has been developed in order to control a gel electrophoresis device that is currently under development. Electrophoresis has been known for about a century and involves the movement of proteins when an electric current is applied in a compartment filled with gel [1]. A agarose enhance solution is used as a substrate to facilitate the proteins movement, as a common practice in many Gel Electrophoresis (GE) devices The device (Fig. 1) consists of the following basic mechanical components [2]: (i) tube drawer system, (ii) bar code scanner, (iii) wash tank, (iv) robotic arm/gripper, (v) sample carrier, (vi) electrophoresis chamber, (vii) frame/film manipulator, (viii) staining-unstaining chamber, (ix) drying chamber, (x) camera analysis chamber. In this context, the typical steps of the gel electrophoresis process are: (i) take the sample and place it at a gel substrate, (ii) Apply an electric current, (iii) Dry the sample, iv) Stain and un-stain the sample, (v) Dry the sample and (vi) Analyze the sample and print the results. In order to automate the typical steps of the gel electrophoresis process, a mechatronic system was used, and the proposed API has been developed for controlling the mechanical arm of the mechatronic system. The use of an API for controlling a microcontroller is widely used when a lot of external sources must be coordinated [3].

*Corresponding author: evgenidisnikos@gmail.com

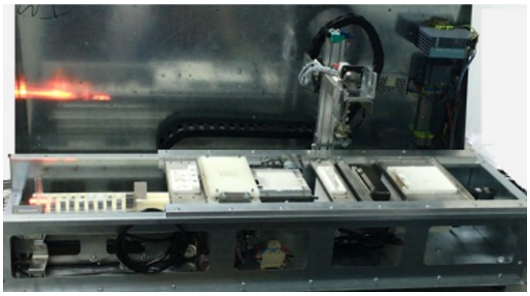

Fig. 1. The overall mechatronic system

2 The microcontroller's connections

Microcontrollers can provide accurate control for the moving parts of a mechatronic system [4]. For controlling the mechatronic system of the electrophoresis device an Arduino microcontroller has been used and the schematic diagram of the controller is presented at Fig. 2.

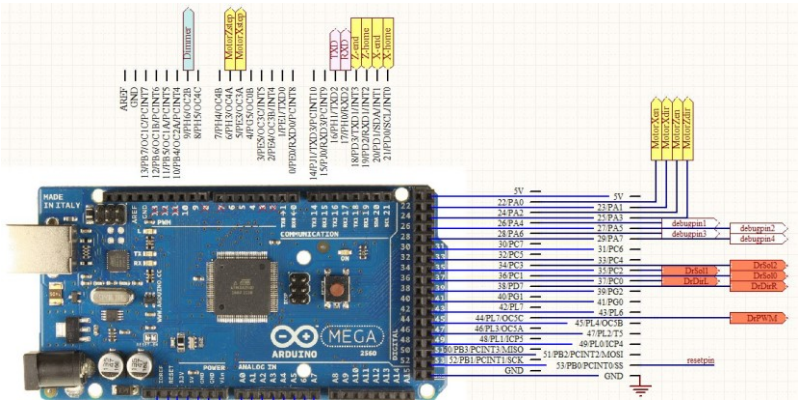

Fig. 2. The connections of the microcontroller

The yellow colored ports stand for the stepper motor connections, the blue for the power connections and the orange ports control the drawer movement. Moreover, the diagram indicates the debugging signals and an additional serial port for future use. Ports 18 to 21 are selected in order to provide an independent external stop signal INT0 to INT3, ports A12 to A14 are used as the drawers limit switches in a Pin Change INTerrupt (PCINT[2:0]) business logic. The drawers' velocity is controlled by port 44 PWM OC5C signal and finally the actuation of the motors MotorXen (X axis) and MotorZen (Z axis) uses ports 22 and 24 respectively. On overall microcontroller operated control circuits that drive motors and gear-bearing arrangements are considered of major importance in custom devices [5].

3 The messaging scheme of the developed API

For every input message the API provides an answering message using the USB port as an interface. Warning messages are also sent at an error handling situation when the software identifies an error. In general, 4 types of answering messages are provided: (i) Information Messages, (ii) Warning Messages, (iii) Error Messages and (iv) Setup Messages.

Table 1. The 4 types of Answering Messages: Information, Warning, Error and SetUp Messages

I1 Info: Motor X initialized I2 Info: Motor Z initialized I3 Info: motor X started I4 Info: motor Z started I5 Info: motor X finished I6 Info: motor Z finished I7 Info: X enable output is high active I8 Info: X enable output is low active I9 Info: Z enable output is high active I10 Info: Z enable output is low active I11 Info: X drive is always active I12 Info: X drive is in auto mode I13 Info: Z drive is always active I14 Info: Z drive is in auto mode I15 Info: Motor X drive set to manual mode and enabled (use S,XE,M for permanent change). I16 Info: X drive set to manual mode and disabled. X,L and X,R commands will be ignored until re-enabled. I18 Info: Motor Z drive set to manual mode and enabled (use S,ZE,M for permanent change) I19 Info: Z drive set to manual mode and disabled. Z,L and Z,R commands will be ignored until re-enabled **I30** Info: Drawer 0 is opening I31 Info: Drawer 0 stopped I32 Info: Drawer 0 is closing I33 Info: Drawer 0 closed I34 Info: Drawer 1 is opening I35 Info: Drawer 1 stopped I36 Info: Drawer 1 is closing I37 Info: Drawer 1 closed I38 Info: Drawer 2 is opening I39 Info: Drawer 2 stopped I40 Info: Drawer 2 is closing I41 Info: Drawer 2 closed I42 Drawer 0=status, 1=status, 2=status	W0 Warning: you must connect pin 53 to RESET to activate reset command W1 Warning: motor X already stopped W2 Warning: Motor Z already stopped W3 Warning: speed of X exceeds max and replaced with 800 W4 Warning: speed of Z exceeds max and replaced with 800 W5 Warning: speed of X cannot be 0 - replaced with 1 W6 Warning: speed of Z cannot be 0 - replaced with 1 W7 Warning: acceleration of X lower than min - replaced with 10 W8 Warning: acceleration of Z lower than min - replaced with 10 W30 Warning: Drawer # is already closed W31 Warning: Drawer # is already closing W32 Warning: Drawer # is already opening W33 Warning: drawer # is already stopped W34 Warning: drawer # is closing now and cannot be opened W35 Warning: drawer # is neither opening nor in unknown status - stop command is not allowed E0 Error: unknown command received E1 Error: motor X already running E2 Error: motor Z already running E3 Error: direction X must be L (left), R (right), 0 (stop) or E (enable) E4 Error: direction Z must be L (left), R (right), 0 (stop) or E (enable) E5 Error: minimum number of steps in X is 1 - received 0 E6 Error: minimum number of steps in Z is 1 - received 0 E7 Error: Valid S,ZE parameters are S,ZE,H S,ZE,L S,ZE,A and S,ZE,M E8 Error: Valid setup commands are S,aE,H S,aE,L S,aE,A and S,aE,M where a = X or Z E9 Error: Valid S,XE parameters are S,XE,H S,XE,L S,XE,A and S,XE,M EA Internal error: OCR and prescaler arrays are not of the same size E10 Error: Motor X enable must be ON or OFF E11 Error: Motor Z enable must be ON or OFF E30 Error: Wrong drawer number. It must be 0, 1 or 2 E31 Error: Wrong drawer command. Available: H=Home, O=Open, S=Stop or U=statUs E32 Error: you must wait for drawer # to finish moving E33 Error: Max time of 3000 ms exceeded in move of drawer # and move cancelled	S1 Setup: Enter the active level of the enable input for drive X (H for high, L for low) and press S2 Setup: X enable output set to high active S3 Setup: X enable output set to low active S4 Setup: enter the active level of the enable input for drive Z (H for high, L for low) and press S5 Setup: Z enable output set to high active S6 Setup: Z enable output set to low active S7 Setup: enter the enable mode for drive X (M for manual, A for Automatic) and press <Enter> S8 Setup: X drive set to manual mode S9 Setup: X drive set to auto mode S10 Setup: enter the enable mode for drive Z (M for manual, A for Automatic) and press <Enter> S11 Setup: Z drive set to manual mode S12 Setup: Z drive set to auto mode S13 Setup: X enable set to high active S14 Setup: X enable set to low active S15 Setup: X drive set to auto mode S16 Setup: X drive set to manual mode S17 Setup: Z enable set to high active S18 Setup: Z enable set to low active S19 Setup: Z drive set to auto mode S20 Setup: Z drive set to manual mode

4 Software flowcharts for controlling the motors

Arduino supports the development of the software program in two distinct phases. The first phase is the program setup which is executed once either at the initialization step or after every reset signal. The second phase includes a continuously running loop, the main program, that continuously monitors all the ports and reacts at every signal. Our main goal was to use a serial programming architecture and as a result the routines are controlled with interrupt routines. The main program monitors the ports for an input message string (a comma separated character string) from the USB port and makes a call to the parser in order to select and execute the proper interrupt routines. The main program also controls the motors (Stepper and DC motors) for the mechatronic system (arm, drawer's and other moving parts). The flag stringcomplete is updated from Arduino's serialEvent() that is called after the end of a single loop and indicates the routines that must be executed. The main loop checks if stringcomplete is activated and calls the serialEvent() routine (Fig. 3a) in order to get the characters from the input message string. For every distinct character at the string, the system safely stores the character and returns stringcomplete =1 for indicating that a input was successfully inserted. The next step is to execute the parser routine (Fig. 3b) for indicating the procedure and the commands that should be followed. The procedure includes a maximum of 5 substrings that are separated with a comma (the message format is: scmd, s1, s2, s3 and s4). Depending on the first substring (scmd values are R,X,Z,D) the main loop executes the corresponding routines.

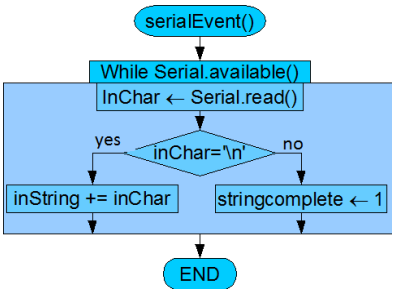

Fig. 3a. Flowchart of the serialEvent routine

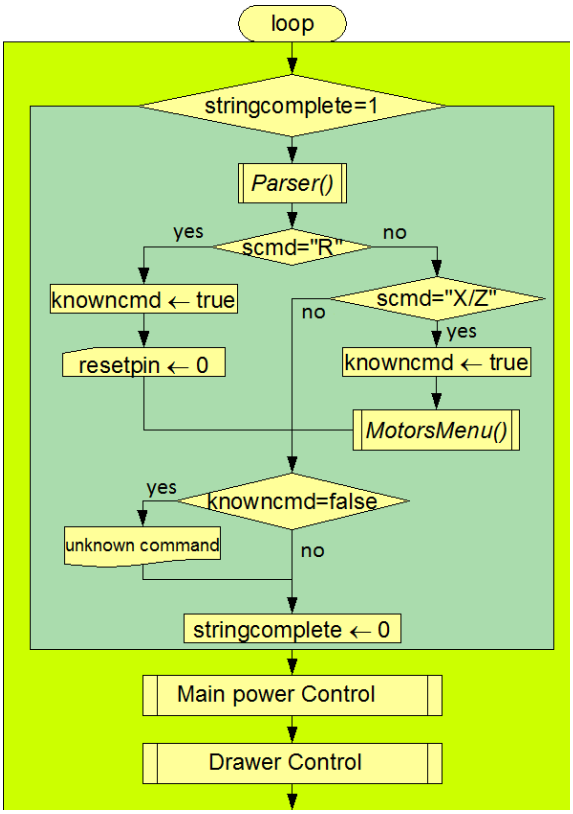

Fig. 3b. Flowchart of the main loop

The controller is based on Arduino Mega 2560 and uses three (3) digital outputs for each stepper motor: (i) enable, (ii) direction and (iii) moving step. The outputs are used both for

the X and the Z axis. The serial port of the controller is used for the communication with the mechatronic system in order to control the motors. The control message setup includes information for (i) the movement at the X or Z axis (ii) the left (L), right (R) or (0) stop signals (iii) the maximum or min speed (vi) the acceleration at a certain level (10 to 65.535) and (v) the number of steps from 1 to 65.535 (Fig. 4a). For example, a message that states (X,R,100,200,800) would inform the system that the motor that controls the X axis should turn right at a speed value of 100 and an acceleration value 200 for 800 steps. Furthermore, we could handle the motors by using a setup command in order to enable either the manual or the automatic mode. For example, a message setup message that states (S,XE,M) would inform the system for controlling the axis X in manual mode. At the automatic mode the output signal becomes active just before the start movement and inactive just after the end of the movement. This reduces the energy concussion of the system, the motor's temperature and the restraint torque at our prototype. At the manual mode, the output signal is activated until the user sends a deactivation command.

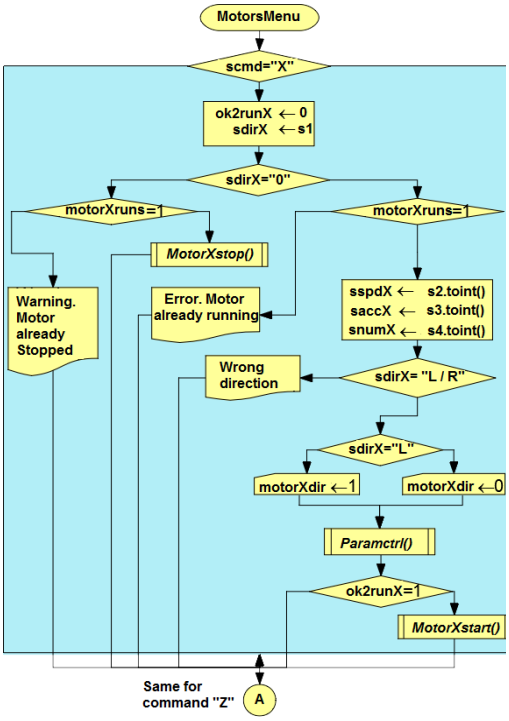

Fig. 4a. The motors basic menu

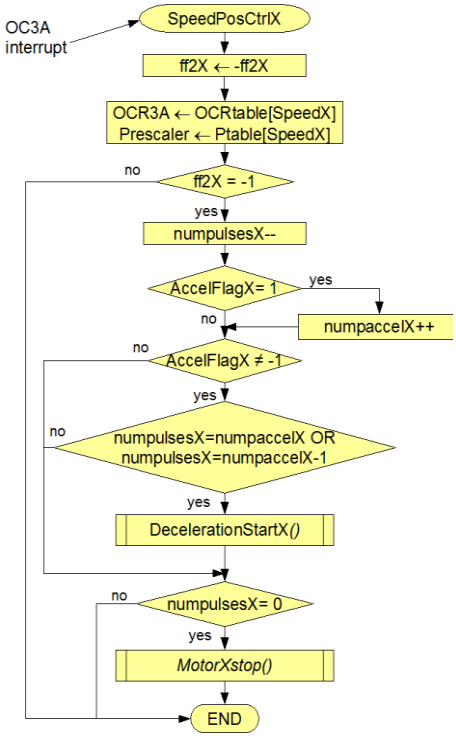

Fig. 4b. The motors acceleration control

At Fig. 4b we present the control flowchart for the movement of the motor. To set the motor movement in a specific position we need timers to control the speed, the position and the acceleration. For speed control and position identification at the X axis we use a timer3 in clear time (CTC) mode and for acceleration we use timer1 in compare register (OCR) mode. For moving at a specific position, we first identify the total number of pulses needed and the exact steps for the motor. In order to reach the total number of pulses the motor gradually accelerates and waits for the timer 3 who controls axis X to stop the motor using the routine MotorXStop. While AccelFlagX equals to one, the motor's acceleration gradually increases the pulses at numpaccel. As a next step in order to smoothly stop the motor the DecelerationStartX routine is used. If AccelFlagX is not equal to minus one (-1) and the number of pulses X (numpulsesX) equals the number of acceleration pulses (numpaccelX)

the DecelerationStartX routine is activated in order to gradually stop the motor. For the Z axis the routine uses timer 4 and repeats the logical steps.

5 Conclusion

In the previous sections, a software framework based on the Arduino microcontroller for controlling the mechatronic system of an innovative fully automatic electrophoresis system has been presented. The proposed API takes into consideration all the functionalities of the mechatronic system and responds well in real world conditions. The total size of the software is 11.162 bytes that is about 4% of the total available storage (253.952 bytes). The global variables are using 1208 bytes that is about 14% of the total RAM memory (8.000 bytes). The results showed high accuracy in operation and efficiency in the control of the mechatronic system's moving parts, according to the requirements.

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code: T1EDK-02403)

References

1. B. Aslam, M. Basit, M. A. Nisar, M. Khurshid, & M.H. Rasool, J. Chromatogr. Sci., **55(2)**, 182-196 (2017)
2. K. Theodoridis, F. Stergiopoulos, D. Bechtsis, N. Nikolaidis, D. Triantafillides, A. Tsagaris, A. Filelis, A. Papaikonomou, An innovative and fully automated system for gel electrophoresis, *ESCAPE 2020 Conference, Italy (*Accepted*)* (2020)
3. L. Sungchul, J. Juyeon , K. Yoohwan , St. Haroon, A Framework for Environmental Monitoring with Arduino-Based Sensors Using Restful Web Service, *Proceedings of the 2014 IEEE International Conference on Services Computing* (2014)
4. R. Md. Kamruzzaman, H.B. Muhibul, Microcontroller Based DC Motor Speed Control Using PWM Technique, *International Conference on Electrical, Computer and Telecommunication Engineering* (2012)
5. V. K. Singh, A. Sahu, A. Beg, B. Khan and S. Kumar, "Speed & Direction Control of DC Motor through Bluetooth HC-05 Using Arduino", *International Conference on Advanced Computation and Telecommunication (ICACAT)*. (2018)