# Arduino CommShell, An Interactive Tool for Mechatronic Classroom Teaching

Li Xiaoming and Xiao Yao
Department of Mechatronics
School of Mechanical Engineering and Automation
Zhejiang Sci-Tech University
Hangzhou, China 310018
email: lxmzist@zstu.edu.cn, 597435983@qq.com

**Abstract—In this paper a CommShell for Arduino has been proposed and designed. CommShell is a shell program running on the Arduino platform, accepting interactive commands from its simulated serial communication port. With the help of Arduino CommShell, users can communicate with Arduino using pre-defined commands and implement mechatronic instrumentation and control tasks without writing and compiling codes. Arduino CommShell is initiated with the idea of easier in-class demonstration on mechatronic system control. However it is also very useful for beginning engineering students to learn the concepts of Mechatronic system or Arduino beginners to get familiar with Arduino. Arduino CommShell is designed as an open architecture which is composed by a command interpreter, a function library and a base set of Arduino API. The function library can be extended to make the shell accept more commands, or be customized to do jobs on designated areas. Experiments and examples have been conducted to verify its usefulness, easiness and extendibility.**

***Keywords-Arduino; Classroom Teaching; Mechatronics; CommShell; Serial Communication***

## I. INTRODUCTION

The classroom teaching of mechatronic courses is sometimes tedious because the concepts of them are usually abstract and hard to understand without practical experiences on mechatronic system. Therefore the demonstration experiments in classroom are considered very helpful form of teaching in the lecture part of teaching activities.

However the traditional mechatronic experiment rigs are complicated, non-portable and non-personalized. Some of them are designed for specific series of experiments on a specific topic or a macro fields, such as robotic arm or mobile robots or product line; others are designed for specific courses but cannot fit to the classroom teaching, such as the MCU experiment kit. From the requirements of classroom teaching, the demonstration experiment kit should be simple, light-weight, and easily personalized for one or more lectures.

Arduino is an open-source project, including its hardware and software[1]. It has many features that make it a very good candidate for developing platform of classroom demonstrating experiment kit. Arduino is based on AVR MCU, which makes it powerful and industrial; The Arduino board is connected with PC or laptop through standard USB interface makes it easy to use and portable; The integrated developing environment for Arduino is simple, and the programming language is C and Java-like, which makes it has a flat learning curve; The API Arduino provides in its programming language is simple and conceals the complexity of MCU such as registers, timers and interrupts. Above all, Arduino has a very low unit price.

Arduino has a great success in market due to features listed above. It even drew attentions from engineering educators. Arduino has been used to develop low cost robotic system with 3D printed parts for research and education[2]. In another application, Arduino has been adopted as a competition platform by a undergraduate class in mechatronics at Santa Clara University in US[3]. In Europe, there is an E-learning project called CLEM (Cloud E-learning Program for Mechatronics), using Arduino as their remote hardware in laboratory[4]. In China, Arduino is also introduced to the competition[5], college students' practical teaching[6] and robot DIY course in High School[7].

However the adoption of Arduino in mechatronic classroom teaching is still rare. The reason lies in the balance between the effectiveness and trouble it will bring. When you give a presentation in classroom, you want a handy tool to show the concepts you are explaining. Arduino is compact, easy to setup and pack, easy to program. However it still needs the cycle of coding, compiling and uploading. This process is sometimes tedious and error-prone, may interrupt the fluency of teaching activity. Therefore one may think it causes more trouble than the benefit it brings, one will not use it in the classroom.

In this paper a powerful tool is introduced. This tool would be extremely easy to use, always available at hand, extendable and interactive. We name it the Arduino CommShell, which is a program running on the Arduino platform and accept user commands through the serial communication interface. With the help of Arduino CommShell, teachers can store the demonstration as a series of commands into the Arduino platform, and just hit the command when need to show them.

In next section a brief review of current Arduino based shell software will be given, and the unique advantages of Arduino CommShell will be argued, followed by the detail explanation on the design and implementation of it. Experiments are conducted and will be described in section V. And section VI will conclude the paper.

## II. REVIEW OF EXISTING ARDUINO SHELL SOFTWARE

Since the birth of Arduino, some researchers and developers have designed and implemented some kinds of

 

shell programs for Arduino. These include the Bitlash, AVRSh, arsh and microBox.

### A. *Bitlash*[8]

Bitlash is an open source interpreted language shell and embedded programming environment for Arduino. The Bitlash interpreter runs entirely on the Arduino and interprets commands that typed in a terminal window or send programmatically to the serial port. Technically speaking the Bitlash is implemented as a library for Arduino and can be programmed using Arduino IDE. It provides many build-in functions to visit the registers, and I/O and Analog pins. The shell support variable definition, which makes it more useful to programmers than teachers.

### B. *AVRSh*[9]

AVRSh is a Unix-like shell with log-in function and file system, and can run on ATMega328P and Arduino devices. It's a firmware which will replace the origin firmware of Arduino. It is powerful shell just like a real system running on the Arduino, however it has replaced the Arduino software so the user need to learn a new language and way of interaction. Also it requires skills to install the shell on Arduino.

### C. *Arsh*[10]

Arsh is a basic Arduino Shell, with a basic interactive environment to explore the Arduino. So far it has only 6 commands, which can read/write pins or set the mode of pin. It is just in a very early stage of development.

### D. *MicroBox*[11]

MicroBox is a Arduino library that provides a interface with Linux-Shell-like look-and-feel for Arduino applications. This shell allow users to define their own commands after they get the source code. However the user must be able to write C/C++ code to make new commands.

### E. *Requirements Analysis in Reality*

From above review of current shell softwares for Arduino, ones may find that nearly all softwares are made for developers, not for teachers in mechatronics. They use C/C++ to develop the shells, making them powerful but losing the advantages of the easiness of the Arduino which itself was designed for.

As a teacher as well as a developer, one could easily define the requirements for a product which is suitable for classroom demonstration on simple mechatronic concepts or application. They should have features listed as follows.

- Portable. Potability is the very important feature of classroom teaching aid because it need to be carried on from classroom to classroom.
- Ease of use. It should be easy to pack, easy to setup, and always be ready during the whole lecture. There is no need to configure it every time it is used.
- Customizable. The users should be able to choose the function they need for their specific topics on mechatronics. For example, the motor control functions are needed in transmission and control course but it is not needed in interfacing technology course. The users also may need to develop their own functions for their own courses or experiments.

Arduino is portable, ease of use, has been widely used in teaching activities. However it lacks the software to support it to be a convenient tool in classroom. That why Arduino CommShell is developed.

## III. DESIGN OF ARDUINO COMMSHELL

### A. *Architecture Design*

Arduino CommShell is a program running on the Arduino board. This program accepts commands through the Serial Communication port. Therefore the whole system is composed by two parts: a serial port terminal and an Arduino based system. The typical structure of the system is as shown in Fig .1.

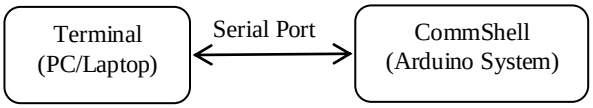


Figure 1. Structure of the system

Arduino CommShell is part of the Arduino System shown in Fig. 1. Its main function is to accept the inputs from the serial port, analyze them, extract the commands and the parameters, and call the corresponding functions that defined earlier in the system. To make the system extendable and easy to maintain, the architecture is designed as shown in Fig .2.

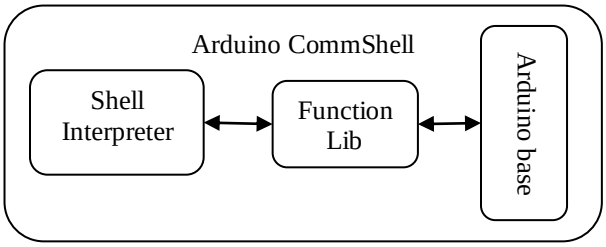


Figure 2. Architecture of Arduino CommShell

Shell Interpreter is used to process the input characters from the serial port and turn them into formative strings containing the name of the command and the parameters. With the name of the command the shell interpreter can search the function lib to get the corresponding function and call it with the parameters.

Function lib is a library that stores the predefined functions. These functions are the main part of the system, and are designed to implement all kinds of functions that are expected by the users. By modifying the function lib (add or remove functions from it) the system can be customized for specific jobs. This feature is useful because the memory of Arduino is limited, and it may be impossible to store the whole function lib inside the chip when the lib grows bigger and bigger.

Arduino base is part of the Arduino system, consisting of APIs of Arduino software.

### B. *Working Principle*

The working principle of the architecture is straight forward. User send characters using any kind of terminal program that support the serial communication protocol to the Arduino CommShell. The character stream will be

processed by shell interpreter consequently. If the shell interpreter inspects correct sequences it will try to convert them into commands and parameters. There is a mapping table inside the function lib. The shell interpreter uses this table to look for the function it need to call. If a function is found, the shell interpreter will then call it and return the output to the terminal. This process is described in Fig .3.

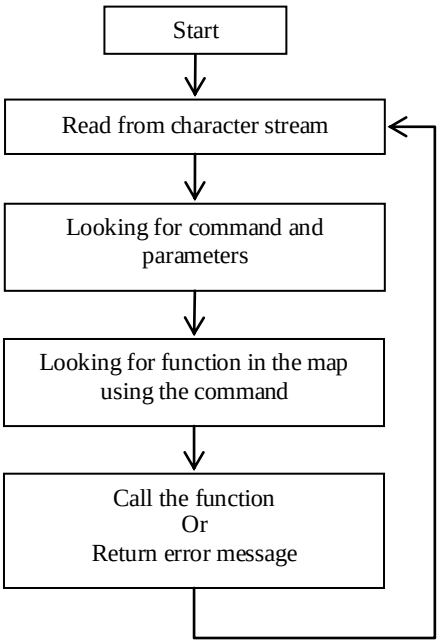


Figure 3. Working process of Arduino CommShell

### C. Design of the Shell Interpreter

Shell interpreter is the core of the Arduino CommShell. It interacts with the user through the interface of serial port, and execute the functionality provided by function lib. Therefore the shell interpreter contains three parts: the interface with the terminal, the interface with the function lib, and working body of itself.

*1) Protocol of the serial communication*

The interface with the terminal is mainly about the communication protocol. It includes the hardware configuration and software protocol.

The shell interpreter makes the Arduino serial port working under the configuration of 9600, N, 8, 1, which are baud rate, parity, data bits and stop bit respectively. This is the default working configuration and supported by most terminal programs.

The application protocol was designed as:

*PC→Arduino:*

*(command)(␣)[parameter](␣)[parameter](\n)*

*Arduino→PC:*

*(Status Code)(␣)(Status Abbr.)(␣)[Message](\n)*

In the protocol the *command* is a string like the variable name in C or Java. The *parameters* are separated strings by blank character. The *status code* is a number indicating the status of the execution of the command. The *status abbreviation* is a short string presenting the result of the command. *Message* is the text message send by the function called by shell interpreter. *Status code* and *status abbreviation* are one-to-one matches. Examples of *status code* and *status abbr* are (0, OK) and (1, ERR).

The protocol is very import for the *shell interpreter* need it to analysis the character stream and find the correct command and parameters, and how to respond to the terminal as well.

The existence of *status code* is designed for softwares who want to communicate with Arduino CommShell in an automatic way.

*2) How to analyze the character stream*

The Arduino programs follow a software framework called the "super loop", which Arduino divided into two parts: the setup function and the loop function. The setup function is first be called when system is powered on, then the loop function will be called repeatedly until the board is shutdown.

In order to keep the shell interpreter working interactively a state machine model has been adopted for implementation, as shown in Fig .4.

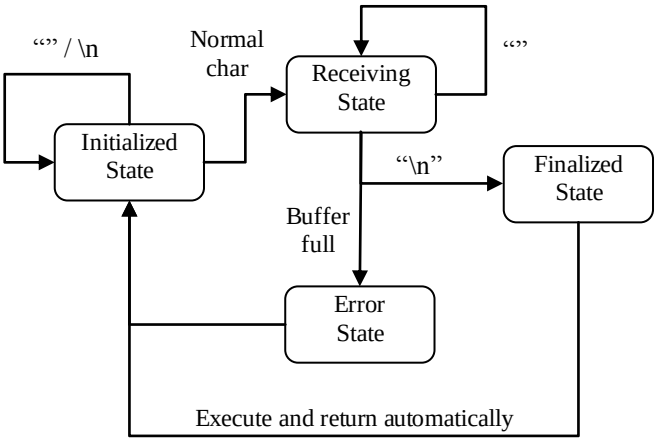


Figure 4. State machine model of the shell interpreter

When shell interpreter receives a character, it will determine how to process it according to the current state and the content of the character. When shell interpreter enters the ending state, it will automatically stores the whole string and completing the remaining jobs of shell interpreter.

*3) How to locate the function in function library*

Function lib is defined as a collection of functions in code level, and all these functions are stored in a single source code file for convenience. The interface for each function in the library is defined uniformly as:

*int function_name*
*(char[] params, char[] return_msg)*

In which *params* is a string containing all parameters separated by blank character, and *return_msg* is a string containing the text based output of this function.

A data structure has been defined to store the information of each function lib item, including the command string and the pointer to specific function, and an array of the items has been created to act like an dictionary. With these data structures the functions in the library have been indexed and can be searched and located using simple methods.

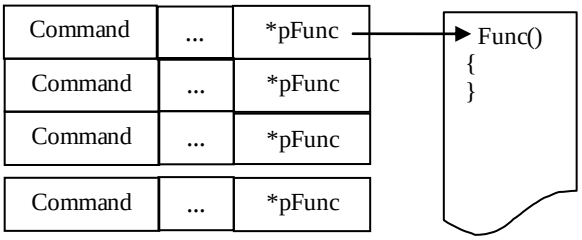


Figure 5. Data structure of function dictionary in function lib

## IV. IMPLEMENTATION

### A. Choose the programming language

The Arduino CommShell is implemented using the Arduino programming language, which is Java-like, and C/C++ compatible. Arduino programming language is actually not an independent language because the source codes are mapped into a C++ source code framework and then compiled and linked as normal C/C++ source codes. However the Arduino use these tricks to simplify the programming process and hide the hardware details. People with little programming skills can use this language to develop simple Arduino software applications.

In the architecture discussed above the basis of the platform (shell interpreter and the interface with the function lib) can be developed using Arduino language plus a little bit of C language. Fortunately the basis is relatively concrete and the final users need not to worry about it. The only thing the final users need to do is design their own functions in the function lib, which can be realized simply in Arduino language.

### B. Implementation of function library

The steps to create custom function lib includes: (1) Write the function following the interface defined in section 3. In the function the program can access all the APIs provided by Arduino and must return the status code and message to the caller. (2) Add the function name into the dictionary defined by the function lib manually. (3) Compile and upload the whole program into the Arduino board using the IDE provided by Arduino. (4) Test and run by connecting to the Arduino board using arbitrary terminal program that supports the serial communication port. Arduino IDE also provides a serial terminal program for test and debugging. Be sure to adjust its settings before use it to communicate with Arduino CommShell by allow attaching a newline character after the string being send.

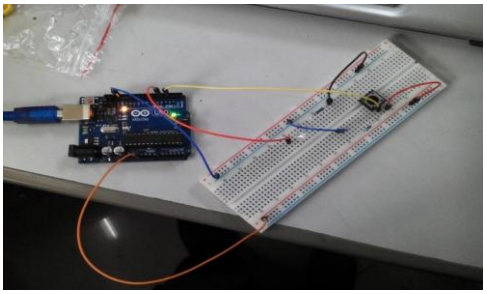

Figure 6. Simple experiment of digital I/O

There are some build-in functions in the library. They are simply the encapsulations of existing APIs of Arduino, such as *digitalWrite*, *digitalRead*, *analogRead*, *pwmWrite*, *makePulse*, etc. Some experiments can be demonstrated using these build-in functions without writing their own functions. For example, use "*digitalWrite 4 1*" will turn on the light in the circuit shown in Fig. 6. This experiment is used to show the digital I/O and its function in interfacing technology class.

## V. EXPERIMENTS

In order to prove the usefulness and extendibility of Arduino CommShell, and make it applicable in read classroom, we designed a command set for step motor control. In this parts we will explain how to extend the Arduino CommShell to add the step motor library and how it works in classroom. We take these activities as experiments to verify the Arduino CommShell.

### A. Step motor control example

Step motor is a kind of actuator in industry and very popular in the application of mechatronics. The principle and application of step motors are a very important part in teaching the mechatronics. The typical configuration of step motor control system is shown in Fig. 7.

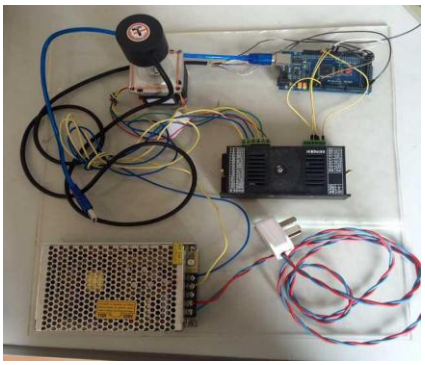

Figure 7. Typical configuration of step motor demo experiment

To control the step motor, the micro-controller need to send control signals to the driver of step motor (black box in Fig. 7), which in turn drive the step motor to rotate in desired direction and speed. The signals send by controller are two kinds: the direction signal and the speed signal. Direction signal is a LOW or HIGH voltage and the speed signal is electric pulse with specified frequency.

To control the step motor through the Arduino CommShell, we designed the following commands:

**Step_Setup**: Initialize the values shared by these functions.

**Step_ON**: Start the internal pulse generator. The internal pulse generator will use the current values of configuration.

**Step_OFF**: Stop the internal pulse generator.

**Step_DIR**: Set the running direction of step motor.

**Step_Speed**: Set the running speed of step motor.

**Step_Move**: Move the step motor with specified steps.

**Step_GetPos**: Return the current position of step motor.

Pulse generator is a nested function in the Arduino CommShell, can be switch on or off by a control variable. These functions are all used to adjust a group of variables which are used to control the way the output signals be generated. These variables include: direction, frequency, acceleration and count of steps.

With these commands, teacher can give demos in classroom to explain the control process of the step motors.

### B. *Classroom demonstration example*

The experiment system shown in Fig. 7 has been used in classroom to show how to control the step motor. Here is the process.

First, introduce the hardware connection among Arduino, step motor driver and step motor. Tell the students the working principles of this configuration.

Second, use Step_Setup to initialize the control system.

*Step_Setup 3,5,7*

Where 3,5,7 are the pin numbers of Arduino connected with the driver.

Third, use Step_ON to turn internal pulse generator on.

*Step_ON*

At this time, the pulse generator is working. Since the control variables are all set to 0 so there is actually no signal out of the Arduino.

Fourth, use Step_DIR to control the direction of rotation.

*Step_DIR 0*

In which 0 means positive direction and 1 means negative direction.

Fifth, use Step_Speed to let the motor run.

*Step_Speed 100*

The number in this command is the pulse frequency in PPS (Pulse Per-Second). After sending out this command, the step motor should be rotating forward at a steady speed. The teacher can change the speed and direction using commands like above to explain how to control the speed and direction of step motor in real applications.

For convenience we also designed a brief version of command to control the step motor:

*STmotor PinDir 0|1 PinPulse Rate*

The command and interface is shown in Fig .8.

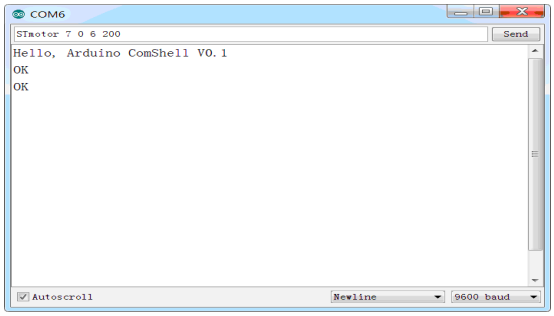


Figure 8. User interface for step motor control

With the help of hardware and Arduino CommShell, students can easily understand the usage of step motors in real applications. It is also possible to enrich the commands with more functions added to the function lib, such as the reading of encoders, open/close loop control, and motion planning. Alternatively one can add commands to demonstrate the phenomenons such as loss of steps or positioning errors of step motors.

The feedback from the students in this classroom were very positive, and some talented students even use the Arduino CommShell to developed another demo system of step motors, shown in Fig .9.

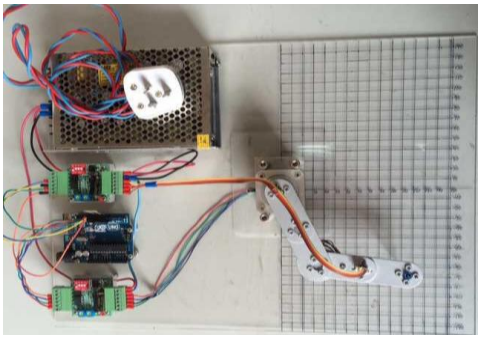

Figure 9. An robot arm using 2 step motors

## VI. CONCLUSIONS

In this paper a teaching tool Arduino CommShell has been designed and implemented. This tool provides a way to interactively using the Arduino as a controller in mechatronic system. This is very helpful when explain new mechatronic concepts to students. The extendable architecture of Arduino CommShell allow teachers to add their own commands to the system and make their own portable teaching kits. Arduino CommShell is also very easily accepted by students who can learn Arduino or mechatronics through it by themselves.

## ACKNOWLEDGMENT

The work in paper is partially supported by the Major Construction Project of ZSTU (Zhejiang Sci-Tech University) with granted number “zyjh201201”. It is also part of the work of Excellent Engineer Education and Training Program sponsored by both ZSTU and Chinese Ministry of Education.

## REFERENCES

[1] “Arduino-Introduction,” https://www.arduino.cc/en/Guide/Introduction, Oct. 27, 2015.

[2] C. García-Saura, and J. González-Gómez, “Low Cost Educational Platform for Robotics, Using Open-Sourced 3D Printers and Open-Sourced Hardware”, Proceedings of ICERI2012, Nov. 2012, Madrid, Spain , pp. 2699-2706.

[3] R. Grover, S. Krishnan, T. Shoup, and M. Khanbaghi1, “A Competition-based Approach for Undergraduate Mechatronics Education Using the Arduino Platform.” Interdisciplinary Engineering Design Education Conference (IEDEC), 2014 4th. IEEE, Mar. 2014, pp. 78-83.

[4] A. E. James, K. M. Chao, W. Li, et al. “An Ecosystem for E-learning in Mechatronics: The CLEM project.” e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on. IEEE, 2013, pp. 62-69.

[5] Y. Yue, R. Li, N. Che and Y. Wang, “The Cultivation of the Creativity of Students Based on the Arduino Competition Pattern.” Northern Economy and Trade, 2013 2. Feb. 2013, pp. 129,131.

[6] Y. Li, X. Pei, and H. Li, “Introduction of Arduino in Practice Teaching for Mechanical Electronics Specialty”, China Modern Educational Equipment, vol 2015, Jan. 2015, pp. 61-63.

[7] J. Zhou, “Implementation of the Course of Making Robots Using Arduino in High School” , Educational Information Technology, 2012 8, Apr. 2012, pp 15-16.

[8] “Bitlash”, http://bitlash.net/, Dec. 27, 2015.

[9] “AVRSH: A Command Interpreter Shell for Arduino/AVR”, http://www.instructables.com/id/AVRSH-A-Command-Interpreter-Shell-for-ArduinoAVR/, Dec. 27, 2015.

[10] “Arsh - The Arduino Shell ”, http://biot.com/arsh/, Dec. 27, 2015.

[11] “Overview - microBox”, http://www.sebastian-duell.de/en/microbox/index.html, Dec. 27, 2015.