FlexSEA: Flexible, Scalable Electronics Architecture for Wearable Robotic Applications

Jean-François Duval, Hugh M. Herr, *Member, IEEE*
Center for Extreme Bionics, MIT Media Lab, Cambridge, MA

Abstract—**Embedded systems for wearable robotics are ideally low-cost, lightweight, miniature, reliable and safe. They have high peak output and negligible standby power, are simple to use and program, can support high-performance real-time control loops and accept additional degrees of freedom, input and output devices. However, practical solutions cannot accommodate all of these criteria and compromises are made; while commercial systems favor cost, safety and reliability, research projects emphasize ease of development and rapid prototyping. To this day, no de facto embedded system exists, impeding the development of novel wearable robots. This work introduces FlexSEA, the FLEXible, Scalable Electronics Architecture designed for wearable robotic applications. This accessible and open-source architecture is useable across various wearable robotic research initiatives, eliminating the need to design a new embedded system for each and every research project.**

***Index Terms* – embedded system, control, wearable robotics**

I. Introduction

Electronic architectures for wearable robotics can be divided in two main categories: microcontroller-based and embedded computer-based. Commercial products tend to be microcontroller-based. Research prototypes use microcontrollers (or FPGA) [5] and embedded computers [1]. For example, in two research groups approaching the design of a prosthetic knee, Sup *et al.* used a microcontroller-based architecture with a single 80MHz microcontroller [6] while Rouse *et al.* designed an architecture based on an embedded computer, a 800MHz Raspberry Pi [1]. Table 1 presents a general comparison of the two design approaches.

Certain prostheses use off-board (i.e. external, bench-top) motors and electronics "to accelerate the development process" and "achieve high performance with a simple design" [8] (similar approach for exoskeletons in [9]) or plan on "implementing a self-contained version with on-board servo-amplifiers, batteries and computational capabilities" as further works [9]. Many designs rely on commercial motor drivers [1][4][6][8].

Embedded systems are an important building block in wearable robots, yet they are seldom the core focus of research initiatives and, as a result, documentation can be scarce. There is limited information about failed attempts and justifications for system redesigns. The main problems identified in academic papers, grant reports and through interviews with wearable robotic designers can be grouped into the following areas: limited reliability, compromised processing power and battery drain, slow communication peripherals, limited documentation or reliance on one key staff for system maintenance, low scalability, minimal support for custom actuators and control loops on commercial motor drivers, size, weight and mechanical integration issues.

These problems are faced by researchers in related fields such as humanoid robotics and wearable computers. Therefore, many designers and companies have attempted to design a unified embedded system that could be used in a broad range of projects. Commercially available modular hardware platforms include the Microsoft .NET Gadgeteer system, "an open-source toolkit for building small electronic devices using the .NET Micro Framework" [12], the popular Arduino and its Shields ("Shields are boards that can be plugged on top of the Arduino PCB extending its capabilities."[1])[13], the BeagleBone Black embedded computer with Capes and the Intel Edison with Blocks[2]. SparkFun popularized the use of "breakout boards", minimalist circuit boards that simplify prototyping. These products are now commonly integrated in academic research projects [1]. Custom embedded system designs have been published for wireless sensing [17], miniature mobile robots [14], and mechatronics education and teaching [15]. The common goals are to minimize the number of circuit redesigns and simplify prototyping [17]. In NASA and

TABLE 1 ARCHITECTURE COMPARISON

Microcontroller	Embedded Computer
Pros	
• Small form factor that can easily be adapted to different mechanical designs • Low standby power • Low unit cost, production • Processor-efficient bare-metal software (C and/or ASM)	• Quick design phase • Easy to use high-level software (C++, Python, Java, Matlab) • Minimize the number of specialized skills required to modify the system
Cons	
• Development (prototyping) cost can be higher • Longer design phase • Requires Electrical Engineering skills for the design, maintenance and modification • Bare-metal software (C and/or ASM): less portable, requires specialized skills	• Processor-inefficient high-level software (C++, Python, Java, Matlab) • Higher standby power • Relies on commercial parts (no control over the production and life cycle) • Harder to modify • Integration issues between different subsystems • Sub optimal wiring

The authors are with the Center for Extreme Bionics in the Media Lab at the Massachusetts Institute of Technology, Cambridge, MA, 02139. (email: jfduval@media.mit.edu, hherr@media.mit.edu)

[1] http://www.arduino.cc/en/Main/ArduinoShields
[2] Shields, Capes and Blocks are different name for the same product category: stackable expansion boards.

General Motor's Robonaut 2 "Modularity is prevalent throughout the hardware and software along with innovative and layered approaches for sensing and control." [18] However, to the knowledge of the authors, the specifics of the design are only available through licensing.

In many instances, the price to pay for modularity is an increased number of circuit boards, and supplementary inter-board connections. Wearable robotics projects have higher integration requirements than most pure robotics and wearable sensing projects: safety and reliability are primary functional requirements, especially in powered prostheses and medical devices. "Obtaining acceptable reliability and maintainability of connectors has become more acute due to the modular concept and miniaturization of electronic systems and efforts to conserve precious metals."[19] Compared to humanoid robotics, the number of degrees of freedom is relatively small (most cited work has one or two degrees of freedom), but the instantaneous power requirements are high [1][5][8]; a greater emphasis has to be placed on power electronics than on fast digital communication between the modules. The volume and the weight of the embedded system must be minimized because of their direct impact on the efficiency of devices attached to body extremities [5]. Simpler, more compact designs can be easier to weatherproof for real-world use.

Through a careful analysis of wearable robotic requirements across sensor, actuator and computational modalities, it will be demonstrated that an embedded system design can be achieved that is scalable across a plethora of wearable robotic research programs, and therefore will be used henceforth for more than one year in one project.

II. System Design

A. Core ideas and principles

An important trade-off between microcontroller and embedded computer based systems is ease of development versus optimal design. One important shared limitation is the presence of a single computing element to manage all the sensors and actuators. Developing high-level algorithms has the potential to introduce bugs in safety-critical routines, a problem than can be avoided by using hardware and software encapsulation. Leveraging the advantages of both microcontroller and computer based systems resulted in the following design requirements:

- Combine the power of efficient bare-metal code and the ease of development of operating systems and high-level programming languages
- Increase the prototyping efficiency of new wearable robots, both on the hardware and the software side
- Provide structure for quick modifications and additions of sensors and actuators
- Provide a scalable and flexible system architecture that can support one or many degrees of freedom (min. 4)
- Encapsulate safety and reliability watchdogs in low-level hardware and software
- Applicable to both research prototypes and early production units
- Minimize and simplify wiring

B. Subsystems

These specifications can be obtained by using one or many microcontrollers and a computer in the same system, with a clear boundary between their tasks and functions. The computer is used for only one task: high-level controls, such as finite state machines or continuous control laws. One powerful microcontroller per axis is used to interface with sensors and simple output devices. A separate circuit interfaces with the power electronics required for motion control. Following a business organization naming strategy, the three FlexSEA boards are named Plan, Manage and Execute.

1) FlexSEA-Plan

FlexSEA-Plan is an embedded computer, or a laptop/desktop, used for high-level computing. It boasts a powerful processor and can run an operating system such as Linux, thus making the software development process simple. High-level languages such as Python and Matlab can be used. Saving experimental data is as simple as writing to a text file and interacting with the system is simple. FlexSEA-Plan should be used when ease of development is important, such as for early prototypes, and when complex algorithms and control schemes require significant computing power.

2) FlexSEA-Manage

FlexSEA-Manage is used for mid-level computing tasks. It serves as an interface between FlexSEA-Plan and FlexSEA-Execute, providing communication protocols translation, data routing, and time-sharing. It has an Expansion connector that can interface with a wide variety of input and output devices. Data acquisition, processing, and aggregation can be done on this board, thus unloading FlexSEA-Plan from these simple tasks. For applications that do not require intensive computing, FlexSEA-Plan can be taken out of the system and FlexSEA-Manage can host the high-level state machines.

3) FlexSEA-Execute

FlexSEA-Execute is an advanced motor driver that supports brushed and brushless motors. Wearable robotics applications require different control loops than the typical position and current controllers found on commercial drives. FlexSEA-Execute has onboard sensors (6-axis IMU, temperature, voltage, current), interfaces (strain gauge amplifier), processing power and connectivity to make it possible to close

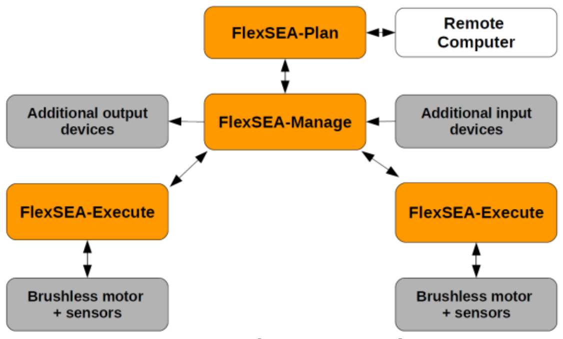


Figure 1 System Architecture Example: 2 DOF

most control loops onboard. It is well suited for the series elastic actuators (SEA) [20] commonly used in prostheses [1][3][8] (easy to close a control loop around different force sensors (including spring position), supports nested loops, high bandwidth, etc.) Refer to [25] for more details.

C. *System architecture*

Figure 1 shows an example architecture than can be used on a 2 DOF system, such as a bilateral exoskeleton or a transfemoral knee-ankle prosthesis. While developing and testing a new system it is expected to have most of the high-level algorithms running on FlexSEA-Plan. FlexSEA-Manage is mostly used as a bridge/translator between different communication busses, but it can also be used to add extra sensors to the system, to synchronize FlexSEA-Executes, or to extend the network. FlexSEA-Execute is in charge of all the motor control algorithms and critical safety features. As the software behavior stabilizes, the high-level control algorithms can be ported on FlexSEA-Manage and/or FlexSEA-Execute. The shared communication libraries make it seamless to use a different board as the network master. Over time, the role of FlexSEA-Plan should become less and less important; on a commercial or pre-commercial application that does not require major computing power it could be completely removed from the system, thus reducing the complexity, cost and volume of the product without designing a new embedded system.

III. Hardware Design

A. *FlexSEA-Execute*

At its core, the FlexSEA-Execute board is a brushless motor driver, customized and optimized for wearable robotics applications. The high level design goals were to maximize system integration (small physical dimensions, large number of integrated peripherals and interfaces, support for external input and output devices), allow fast communication and networkability via the use of a fast multi-drop communication interface, and have built-in safety features.

A Cypress Semiconductor Programmable System on Chip (PSoC) was selected as the main controller. Unlike most of the ICs used for this application, it is not a microcontroller optimized for motor control or a DSP [22] but a hybrid solution comprising of an ARM Cortex-M3 microcontroller, analog, and digital reconfigurable blocks, all integrated in one chip. It allows a tighter integration of analog peripherals, high performance control loops with minimal CPU intervention, and more flexibility for the expansion IOs.

To prevent user errors from creating dangerous situations, or to recover from a hardware failure, a second controller (a smaller PSoC), is used as a safety coprocessor. Figure 2 shows the PWM lines going through the second processor. It has total control over the power bridge and it can place the system in a fail-safe mode.

Switch-mode power supplies are used to downscale the battery voltage to logic level at high efficiency. The motor control bridges (MOSFET and gate driver) have been carefully optimized to minimize parasitic inductance and improve thermal dissipation via the bottom layer of the PCB.

Table 2 FlexSEA-Execute 0.1 Specifications

Elect.	Supply voltage	15-24V
	Motor current	8A Continuous, 25A pulsed (100ms every s)
	Intermediate	10V 500mA SMPS
	Logic supply	5V 500mA SMPS
Motor	Type	3-phase brushless (BLDC)
	Sensor(s)	Hall effect, optical encoder
	Commutation	Block (Sinusoidal & FOC HW sup.*)
	PWM	12 bits 20kHz, 9.65 bits 100kHz
CPU	Reference	PSoC 5LP - CY8C5888AXI-LP096
	Special features	Programmable analog and digital blocks
	CPU/RAM/IOs	80MHz ARM Cortex-M3, 256KB RAM, 62
	Software / IDE	PSoC Creator 3.1, C (GCC 4.7.3) and
	Co-processor(s)	PSoC 4 - CY8C4245LQI-483
Serial interface	Type	3x Half-Duplex RS-485
	Bandwidth	Up to 4Mbps with 1TP, 2Mbps tested
USB		Full-Speed (FS) 12 Mbps
Current sensing	Hardware	0.005Ω resistor
	Software /	20kHz Proportional-Integral controller
Safety features	Overvoltage	TVS clamps at 36V
	Overcurrent	Software protection
	Locked rotor	Hardware - lead shorting circuit
	Motor	Hardware measurement
	Board	CPU + bridge temperature reading
Clutch		Variable voltage, 8-bits PWM, 400mA
Strain		Dual stage, 500 < G < 10000, high CMRR
External periph.	Connector	Molex PicoClasp 40 positions, SMD 1mm
	IOs available	12
	Digital IOs	Up to 12
	Analog inputs	Up to 8 (12-bit SAR, 8-20-bits Sigma Delta)
	Serial	I²C, SPI, UART
	Other	1 optical encoder (A/B/I), 1 Hall effect
Physical	X (mm)	49
	Y (mm)	49
	Z (mm)	From 12 to 15mm depending on capacitors
	Weight	20.1g barebone, 34.8g with heatsink
PCB tech.	Layers	6
	Copper	1 Oz
	Trace/space/via	5/5 mils trace/space, 8/20 mils blind vias
	Assembly	Double-sided
Other		6-axis IMU, RGB LED

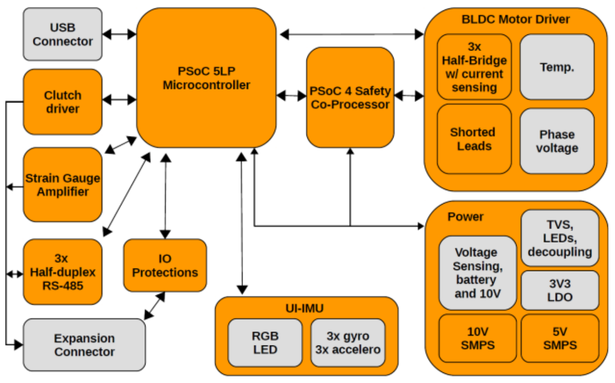


Figure 2 FlexSEA-Execute 0.1 Hardware

The 6-layer PCB uses planes to minimize trace resistance and carry heat away from the components. Blind vias are used to minimize the circuit's size.

A complete description FlexSEA-Execute v0.1's design is available in [25].

B. *FlexSEA-Manage*

FlexSEA-Manage is a polyvalent circuit that can have a wide range of usages depending on the system architecture. In the simplest system designs, it will act as a communication protocol translator, allowing FlexSEA-Plan and FlexSEA-Execute to communicate. When multiple FlexSEA-Execute are used, it routes packets, and can manage communication timings. It can be used to add extra sensors and output devices to the system. In systems that do not require the computing power of an embedded computer, FlexSEA-Manage can host the high-level state machines.

Its design is centered on a powerful 180MHz STM32F427 ARM Cortex-M4 microcontroller. Its floating point co-processor can be used, in the future, for complex real-time control algorithms.

TABLE 3 FLEXSEA-MANAGE 0.1 HARDWARE SPECIFICATIONS

Elect.	Supply voltage	5V in (from Plan or USB), on-board 3V3
	Current (mA)	105mA (standalone, latest code running)
Processor	Reference	STM32F427ZIT6
	Special features	Floating-point co-processor
	CPU/RAM/IOs	180MHz ARM Cortex-M4, 2MB FLASH,
	Software / IDE	Eclipse C/C++, ARM GCC, OpenOCD
Serial interfaces	Type	2x [3x Half-Duplex RS-485]
	Bandwidth	Up to 10Mbps
	Type	Full-duplex SPI
	Bandwidth	20+ Mbps (tested up to 12Mbps)
USB		Full-Speed (FS) / High-Speed (HS)
Periph. / features	FLASH memory	128Mbits
	IMU	6-axis (3x accelero, 3x gyro)
	Power output	2x 24V 1A high-side switches
	LEDs	2x green, 1x RGB
	Switches	1x user input switch
External periph.	Connector	Molex PicoClasp 40 positions, SMD 1mm
	IOs available	17, shared with functions below
	Digital IOs	Up to 9, protected
	Analog inputs	8x 12-bit SAR with special functions
	Serial	I²C, SPI, USART
Physical	X (mm)	40
	Y (mm)	40
	Z (mm)	10.7
PCB tech.	Layers	4
	Copper	1Oz
	Trace/space/via	5/5 mils trace/space, 8/20 mils vias
	Technology	Standard
	Assembly	Double-sided

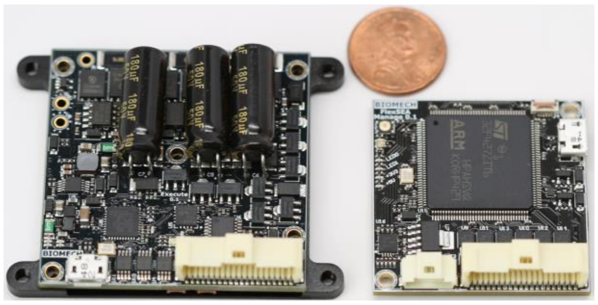


Figure 3 Execute (left) and Manage (right) prototypes (v0.1, 2015)

C. *FlexSEA-Plan*

FlexSEA-Plan can be a laptop/desktop computer, or an embedded computer. It can be linked to Manage or one Execute via USB or SPI (embedded computer only). Using a commercial embedded computer allows roboticists to select the device that works best for their application (form factor, OS, processing power). Processing power can easily be added to the FlexSEA system as new embedded computers are released. A BeagleBone Black was used for most of our experiments, and future work will be based on the Intel Edison. When using a laptop/desktop, Ubuntu was used with the C/C++ Plan project (including a Qt GUI).

D. *Communication*

1) *Interfacing Plan and Manage*

While embedded computers offer substantial processing power, communication isn’t always optimal. At one extreme, some of them have simple interfaces such as serial (UART) and I²C that could be used to directly interface with sensors and microcontrollers, but the data rates are limited. At the other extreme, the Ethernet port can be used but it requires substantial hardware and software overhead on the FlexSEA-Manage board (same for EtherCAT). SPI offers a compromise, with typical data rates above 20 Mbits/s, more than enough for our application. Due to data rate dropping quickly with distance, FlexSEA-Manage has to be physically close to Plan (recommended maximum: 15cm). USB can be used, allowing longer distances. For laptop/desktops it is the only communication option.

2) *Interfacing Manage and one or many Execute*

Common choices in robotics are CAN [21] and EtherCAT. While CAN is cheap, robust and safe, its 1Mbps bandwidth is an important bottleneck for application with multiple motor drivers. EtherCAT offers 100Mbps but that speed comes with a price; the bus requires a Master. When this project was started, no embedded computer was certified as an EtherCAT master, thus requiring the presence of a large computer in the network. The cables, connectors and special ASICs required for EtherCAT add to the cost, volume and complexity of the system. With the relatively small number of actuators on a typical wearable robotics project (less than 4) a simpler and slower interface can be used. RS-485 is often associated with old technology [18] but its simplicity, low cost, robustness and speed (theoretically up to 100Mbps; 20Mbps achievable in our application) made it an appealing option for FlexSEA, especially in multi-drop configuration. Three transceivers are used to allow different communication strategies; from one to 3 twisted pairs can be used to achieve asynchronous half-duplex (default), synchronous half-duplex, asynchronous full-duplex or synchronous full-duplex data exchanges.

IV. SOFTWARE DESIGN

While the focus of this work was system and hardware design, a large amount of embedded software had to be written to enable all the features of the FlexSEA system. This section takes a high-level approach to describe the software design of FlexSEA rather than diving into details. Only the communication stack shared by the three boards is described, as it is the key component of the system design. A detailed description of commands, control loops, organization and timings, trajectory generations, etc. is available in [23].

A. *Communication and networking: overview*

A custom communication protocol was developed for this project. It is used for the Plan-Manage interface, for the Manage-Execute interface(s) and with any other communication peripheral, such as the USB ports present on Manage and Execute. The hardware layer can be modified; as long as it can deliver bits from point A to point B, the system will be transparent to the changes. To avoid conflicts and to simplify programming, the current version of the communication stack is highly hierarchical: the Master always initiates the transfers. Slaves will only emit after a Read request was received.

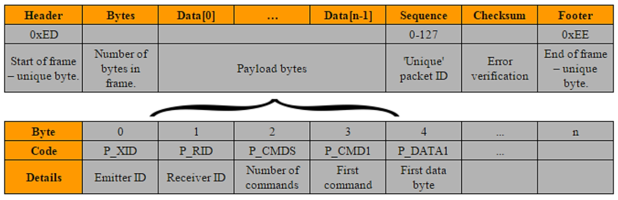
The FlexSEA software stack uses an approximation of the Open Systems Interconnection model (OSI model) by merging layers 4 through 7 into a single Application layer. With a limited need for packet routing, layer 3 (Network) is absorbed by layer 2 (Data link).

B. *Application Layer*

At the highest abstraction level, intelligent information is exchanged between different FlexSEA boards regardless of the physical media used. High-level commands like such as “set control mode” are exchanged. Command codes are 7-bits long, left justified; the LSB of the command byte is 1 for Read commands and 0 for Write commands (P_CMD1 in the lower part of Figure 4).

C. *Data-link Layer*

To preserve data integrity, raw bytes should not be sent on the physical layer. The FlexSEA communication software automatically adds a header, a footer, an indicator of the number of bytes in the frame, a packet ID, a checksum to detect invalid data, and escape characters. High values are used for the Header, Footer and Escape bytes to avoid confusion with the command codes. The Sequence byte is used to keep track of the commands exchanged between the boards and to detect missed packets[3]. At this point the command is ready to be transmitted on any physical bus.



Header	Bytes	Data[0]	…	Data[n-1]	Sequence	Checksum	Footer
0xED					0-127		0xEE
Start of frame – unique byte.	Number of bytes in frame.	Payload bytes			'Unique' packet ID	Error verification	End of frame – unique byte.

Byte	0	1	2	3	4	...	n
Code	P_XID	P_RID	P_CMDS	P_CMD1	P_DATA1	...	
Details	Emitter ID	Receiver ID	Number of commands	First command	First data byte		

Figure 4 Communication protocol fields

D. *Physical Layer*

The system is designed to be compatible with any physical interface. Currently, we use a full-duplex Serial Peripheral Interface (SPI) bus, or USB, from FlexSEA-Plan to one FlexSEA-Manage and multi-drop RS-485 busses from FlexSEA-Manage to FlexSEA-Execute boards. At this level, the data is in the forms of bits and bytes, represented by varying electrical levels. In software, sending a command is as simple as placing a packaged payload (generated by the Data Link layer) in the relevant transmission buffer.

E. *Receiving Commands*

In the layers description the emphasis was on the transmission of commands. When bytes are received by a physical communication interface, the inverse sequence is done. First, either after new data is received or on a fixed timing, the reception buffer is parsed by an “un-packaging” function (data-link level). It searches for a header, then for a footer in the right position (position calculated from the Bytes field that follows a valid header) and then for a valid checksum. Data with a valid Header and Footer is known as properly framed. When a properly framed command fails at the checksum test, it is eliminated (buffer erased). If the checksum is valid it is copied to a new buffer and the original version is erased to avoid double detection.

This new buffer containing the unpacked payload will be parsed at the application level. If the Receive ID belongs to another board the packet will be routed to the appropriate interface. If it belongs to the board that received it, it will be decoded and the appropriate application function will be called.

V. Discussion and Conclusion

A. *Results*

The core FlexSEA system comprises two custom circuit boards (FlexSEA-Manage and FlexSEA-Execute), a computer (laptop/desktop, or embedded) running a C/C++ program (with an optional C++ Qt GUI), a communication stack shared across all boards, and user code running on specific boards. The circuits present in a system, the network structure, and the software are dependent on the application. This variability prevents the authors from providing reliable, numerical, “system-wide” performance metrics. Unit tests, as well as inter-board communication tests, are documented in [23]. [23] contains detailed results specific to FlexSEA-Execute.

As of today, FlexSEA has been used to prototype a knee prosthesis, autonomous ankle exoskeletons and a bilateral ankle exoskeleton with neuro-inspired controls. Ongoing projects include a multi-DOF robotic ankle, a new prosthetic knee, and a 3-joint robotic manipulator.

B. *Future Work and Open Source*

At the time of this publication FlexSEA is an active project. The FlexSEA-Execute 0.2 hardware is currently being tested. Other than minor fixes listed in [23], the main changes are a wider input voltage range (up to 48V) and the presence of on-board memory for data logging during long experiments.

No further hardware modifications are planned, the actual circuits being stable, efficient and reliable. The ongoing efforts have to be concentrated on software. As a starting point, a C++ Qt GUI tool is under development. It allows the user to visualize variables and sensors, to send commands to different boards, and to tune control loops. Significant

[3] Not implemented in the current software release.

improvements in communication speeds can be achieved via software (better timing management, multiple transceiver support, etc.)

"Free software and open hardware have been successfully used to bridge access gap in areas where cost was a problem, democratizing the innovation process." [24] Closely related to the academic reality, the Personal Robot 2 (PR2) and Robot operating System (ROS) from Willow Garage are excellent examples of the value of open-source technologies. They allowed the exchange of software among researchers worldwide, promoting collaboration and quickening the advancement of science. All the FlexSEA design files and sources are open-source and can be used as-is, or modified, in your future project.[4]

C. Conclusion

In this paper we present a flexible and scalable embedded system optimized for wearable robotic applications. We humbly hope that FlexSEA will become a widely adopted platform in the field of wearable robotics, paving the way for revolutionary artificial limbs and human augmentation machines.

Acknowledgment

The authors thank Luke Mooney, Elliott Rouse and Dominic Létourneau for their invaluable expert advice and suggestions. Additionally, the authors thank Steven Keyes, Jake Isenhart and Erin Main for their hard work on the project as part of MIT's Undergraduate Research Opportunities Program (UROP).

References

[1] Rouse, E.J.; Mooney, L.M.; Martinez-Villalpando, E.C.; Herr, H.M., "Clutchable series-elastic actuator: Design of a robotic knee prosthesis for minimum energy consumption," in *Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on* , vol., no., pp.1-6, 24-26 June 2013

[2] E. J. Rouse, L. M. Mooney and H. M. Herr, "Clutchable series-elastic actuator: Implications for prosthetic knee design", *The International Journal of Robotics Research*, 9 October 2014

[3] Au, S.K.; Herr, H.; Weber, J.; Martinez-Villalpando, E.C., "Powered Ankle-Foot Prosthesis for the Improvement of Amputee Ambulation," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE* , vol., no., pp.3020-3026, 22-26 Aug. 2007

[4] Walsh, C.J.; Pasch, K.; Herr, H., "An autonomous, underactuated exoskeleton for load-carrying augmentation," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* , vol., no., pp.1410-1415, Oct. 2006

[5] Mooney et al. "Autonomous exoskeleton reduces metabolic cost of human walking." *Journal of NeuroEngineering and Rehabilitation* 2014 11:151.

[6] Sup, Frank; Varol, H.A.; Mitchell, J.; Withrow, T.J.; Goldfarb, M., "Preliminary Evaluations of a Self-Contained Anthropomorphic Transfemoral Prosthesis," in *Mechatronics, IEEE/ASME Transactions on* , vol.14, no.6, pp.667-676, Dec. 2009

[7] Grebenstein, M.; Albu-Schäffer, A.; Bahls, Thomas; Chalon, M.; Eiberger, O.; Friedl, W.; Gruber, R.; Haddadin, S.; Hagn, U.; Haslinger, R.; Hoppner, H.; Jorg, S.; Nickl, Mathias; Nothhelfer, Alexander; Petit, F.; Reill, J.; Seitz, N.; Wimbock, T.; Wolf, S.; Wusthoff, T.; Hirzinger, G., "The DLR hand arm system," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on* , vol., no., pp.3175-3182, 9-13 May 2011

[8] Caputo, J.M.; Collins, S.H., "An experimental robotic testbed for accelerated development of ankle prostheses," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on* , vol., no., pp.2645-2650, 6-10 May 2013

[9] Sup, Frank; Varol, H.A.; Mitchell, J.; Withrow, T.; Goldfarb, M., "Design and control of an active electrical knee and ankle prosthesis," in *Biomedical Robotics and Biomechatronics, 2008. BioRob 2008. 2nd IEEE RAS & EMBS International Conference on* , vol., no., pp.523-528, 19-22 Oct. 2008

[10] Witte, K.A.; Juanjuan Zhang; Jackson, R.W.; Collins, S.H., "Design of two lightweight, high-bandwidth torque-controlled ankle exoskeletons," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on* , vol., no., pp.1223-1228, 26-30 May 2015

[11] Fleischer, C., "Embedded Control System for a Powered Leg Exoskeleton," In Proc. 7th Intern. Workshop *Embedded Systems-Modeling, Technology and Applications*, pp. 177-185, 2006

[12] Hodges, Steve; Villar, N.; Scott, J.; Schmidt, A., "A New Era for Ubicomp Development," in *Pervasive Computing, IEEE* , vol.11, no.1, pp.5-9, January-March 2012

[13] Badamasi, Y.A., "The working principle of an Arduino," in *Electronics, Computer and Computation (ICECCO), 2014 11th International Conference on* , vol., no., pp.1-4, Sept. 29 2014-Oct. 1 2014

[14] Yan Meng; Johnson, K.; Simms, B.; Conforth, M., "A generic architecture of modular embedded system for miniature mobile robots," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* , vol., no., pp.3725-3730, 22-26 Sept. 2008

[15] Nursal, A.O., "Modular embedded system design for mechatronic education" in *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on* , vol., no., pp.109-112, 15-17 July 2010

[16] Mitchell, R.J.; Grimbleby, J.B.; Loader, R.J.; Kambhampati, C., "Modular embedded system for teaching real-time control," in *Control, 1994. Control '94. International Conference on* , vol.1, no., pp.471-475 vol.1, 21-24 March 1994 4

[17] Benbasat, A.Y.; Morris, S.J.; Paradiso, J.A., "A wireless modular sensor architecture and its application in on-shoe gait analysis," in *Sensors, 2003. Proceedings of IEEE* , vol.2, no., pp.1086-1091 Vol.2, 22-24 Oct. 2003

[18] Diftler, M.A.; Mehling, J.S.; Abdallah, M.E.; Radford, N.A.; Bridgwater, L.B.; Sanders, A.M.; Askew, R.S.; Linn, D.M.; Yamokoski, J.D.; Permenter, F.A.; Hargrave, B.K.; Piatt, R.; Savely, R.T.; Ambrose, R.O., "Robonaut 2 - The first humanoid robot in space," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on* , vol., no., pp.2178-2183, 9-13 May 2011

[19] Desmet, H., "Engineering and reliability aspects of connectors in electronic system applications," in *Microelectronics and Reliability*, Vol. 17, pp. 185-192, 1978

[20] G. A. Pratt, M. M. Williamson, "Series Elastic Actuators", MIT Artificial Intelligence Laboratory and Laboratory for Computer Science, 1995

[21] Harris, A.; Katyal, K.; Para, M.; Thomas, J., "Revolutionizing Prosthetics software technology," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* , vol., no., pp.2877-2884, 9-12 Oct. 2011

[22] Zongwu Xie; Jingdong Zhao; Jianbin Huang; Kui Sun; Genliang Xiong; Hong Liu, "DSP/FPGA-based highly integrated flexible joint robot," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on* , vol., no., pp.2397-2402, 10-15 Oct. 2009

[23] Duval, J-F., "FlexSEA: Flexible, Scalable Electronics Architecture for Wearable Robotic Applications," *Master's thesis*, Massachusetts Institute of Technology, Media Arts and Sciences 2015

[24] Jose, M.A.; Martinazzo, A.A.G.; Biazon, L.C.; Ficheman, I.K.; Lopes, R.D.; Zuffo, M.K., "Power wheelchair open platform," in *2014 5th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, vol., no., pp.455-460, 12-15 Aug. 2014

[25] Duval, J-F.; Herr, H. M., "FlexSEA-Execute: Advanced Motion Controller for Wearable Robotic Applications," *Biomedical Robotics and Biomechatronics, 2016. BioRob 2016. 6th IEEE RAS/EMBS International Conference on*, 26-29 Jun. 2016. In press.

[4]Documentation and files available at http://flexsea.media.mit.edu/