EMBEDDED SYSTEM IMPLEMENTATION FOR REAL-TIME CONTROL OF ANKLE FOOT ORTHOSIS

by

Melih Bayraker

B.S., Metallurgical and Materials Engineering, Marmara University, 2014

B.S., Electrical and Electronics Engineering, Marmara University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2019

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Assist. Prof. Faik Başkaya for his continuous support, supervision, patience, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would like to express my sincere appreciation to Assoc. Prof. Evren Samur for his insightful suggestions and comments throughout this work. I would also like to thank Assoc. Prof. Rıfat Koray Çiftçi sincerely for his participation in my thesis jury.

I would especially like to thank Oğuzhan Kırtaş for his contributions and excellent work on the AFO prototype. I sincerely thank him for his suggestions and support.

My special thanks go to İnci Bayraker, my mother. She helped me to obtain experimental data by participating in the tests as a subject. Her endless support and belief always give me courage and strength.

Last but not least, I would like to thank my father, Hasan Tahsin Bayraker, and my brother, İsmail Hakan Bayraker, for their constant support and encouragement.

ABSTRACT

EMBEDDED SYSTEM IMPLEMENTATION FOR REAL-TIME CONTROL OF ANKLE FOOT ORTHOSIS

Exoskeletons are widely used to restore and augment human movements, and the global smart exoskeleton market has been rapidly growing because of the need for efficient rehabilitative and assistive purposes. One of the most demanded and used exoskeletons is ankle-foot orthosis (AFO), which is used to control and assist the position and motion of the ankle. In this thesis, an embedded system implementation for real-time control of an AFO is proposed. The compactness and portability of a device are greatly appreciated in the healthcare field. Therefore, the controller is designed to be a compact and portable system. The components of the embedded system are selected according to the design requirements, and the implementation is realized to achieve the maximum performance possible by using various hardware accelerators and peripherals. Furthermore, dual AFO support is developed as an innovative feature. Finally, experimental tests are conducted to validate and evaluate the proposed system. Results show that the proposed system operates accurately, and the performance, mobility, and capability of the overall system are improved.

ÖZET

AYAK-BİLEK ORTEZİNİN GERÇEK ZAMANLI KONTROLÜ İÇİN GÖMÜLÜ SİSTEM UYGULAMASI

Dış iskeletler, insan hareketlerini eski haline getirmek ve artırmak için yaygın olarak kullanılmaktadır ve küresel akıllı dış iskelet pazarı, etkili rehabilitasyon ve yardımcı amaçlara duyulan ihtiyaç nedeniyle hızla büyümektedir. En çok talep edilen ve kullanılan dış iskeletlerden biri, ayak bileğinin pozisyonunu ve hareketini kontrol etmek ve desteklemek için kullanılan ayak-bilek ortezidir (AFO). Bu tezde, bir AFO'nun gerçek zamanlı kontrolü için kullanılabilecek bir gömülü sistem uygulaması önerilmiştir. Bir cihazın kompaktlığı ve taşınabilirliği sağlık alanında büyük değer taşımaktadır. Bu nedenle, kontrol cihazı kompakt ve taşınabilir bir sistem olarak tasarlanmıştır. Gömülü sistemin bileşenleri tasarım gereksinimlerine göre seçilmiş ve çeşitli donanım hızlandırıcıları ve çevre birimleri kullanılarak mümkün olan maksimum performansı elde etmek için uygulama gerçekleştirilmiştir. Ayrıca, çift AFO desteği yenilikçi bir özellik olarak sisteme eklenmiştir. Son olarak, önerilen sistemi doğrulamak ve değerlendirmek için deneysel testler yapılmıştır. Sonuçlar, önerilen sistemin doğru çalıştığını ve tüm sistemin performans, mobilite ve kabiliyetini geliştirdiğini göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

c_t	Motor torque constant
F_a	Axial load
i	Moment current
L	Moment arm
M	Artificial muscle
M_a	Moment acting on the ankle joint
p	Ball screw pitch
P	Force generated by the artificial muscle
S	Spring
T	Motor driving torque
η	Ball screw efficiency

LIST OF ACRONYMS/ABBREVIATIONS

ADC	Analog-to-Digital Converter
AFO	Ankle Foot Orthosis
CLA	Control Law Accelerator
CLB	Configurable Logic Block
CPU	Central Processing Unit
cRIO	Compact Reconfigurable Input/Output
DMA	Direct Memory Access
DSP	Digital Signal Processing
ePWM	Enhanced Pulse Width Modulator
eQEP	Enhanced Quadrature Encoder Pulse
FF	Forefoot
FPGA	Field-Programmable Gate Array
FPU	Floating-Point Unit
FSR	Force Sensitive Resistor
GPIO	General Purpose Input Output
HF	Hindfoot
HX	Hallux
I/O	Input/Output
ISR	Interrupt Service Routine
KAFO	Knee Ankle Foot Orthosis
LiPo	Lithium-Ion Polymer
LUT	Look-Up Table
MCU	Microcontroller Unit
NI	National Instruments
OS	Operating System
PID	Proportional Integral Derivative
PIE	Peripheral Interrupt Controller
PLL	Phase-Locked Loop

PWM	Pulse Width Modulation
RIO	Reconfigurable Input/Output
RISC	Reduced Instruction Set Computing
RMSE	Root-Mean-Square Error
sbRIO	Single-Board Reconfigurable Input/Output
SEA	Series Elastic Actuator
SPI	Serial Peripheral Interface
TB	Tibia
TI	Texas Instruments
TMU	Trigonometric Math Unit
VCA	Voice Coil Actuator

1. INTRODUCTION

Exoskeletons are wearable devices that are utilized to improve the physical competency of the wearer. These wearable devices provide assistance to limb movements with better strength by overcoming physical limitations. They can be powered and integrated with various parts such as sensors, actuators, electric motors, pneumatics, hydraulics, etc.

Due to the growing demand for rehabilitation of people suffering from physical and neurological disorders, the use of exoskeletons has been rapidly increasing. More specifically, smart exoskeletons, which are utilizing actuators and sensors to provide improved support to particular body parts, are witnessing massive adoption for clinical purposes.

One of the most commonly used exoskeletons is ankle-foot orthosis (AFO), which surrounds the ankle, foot, and leg to control and assist the position and the motion of the ankle. AFOs also provide toe clearance and stabilization of the foot and ankle, which decreases the risk of falling. The goal of this study is to design and develop an enhanced embedded system based controller for the real-time control of an AFO. In the medical field, the mobility and flexibility of a system are highly regarded. Therefore, another goal of this study is to improve the mobility and flexibility of the AFO. Last but not least, improving the capability of the controller with dual AFO control is also targeted.

This thesis consists of six chapters. Chapter 1 gives the motivation and objective of the thesis, whereas Chapter 2 provides background information regarding gait fundamentals, AFO types, and AFO implementations. Chapter 3 presents the proposed embedded system implementation of the AFO, which is followed by Chapter 4 and Chapter 5 containing the experimental methods, experimental results, and discussion of the results. Lastly, the conclusion of the thesis is given in Chapter 6.

2. BACKGROUND

2.1. Gait

Gait is, quite simply, the manner or style of walking. It is a fundamental function for life and one of the keys to movement. Gait is performed by the movement of limbs and is accomplished through the action of the neuromusculoskeletal system [1].

Valuable answers can be found by gait analysis. For health-related issues, early detections and predictions are crucial. Gait analysis can give an idea about the overall health status of the subject [1]. One of the simplest ways to perform gait analysis is to observe the subject walking in a straight line.

2.1.1. Fundamentals of Gait

The gait cycle starts from the initial hill contact to the ground. This hill contact of one foot indicates the beginning of the gait cycle. It lasts until the next hill strike of the same foot.

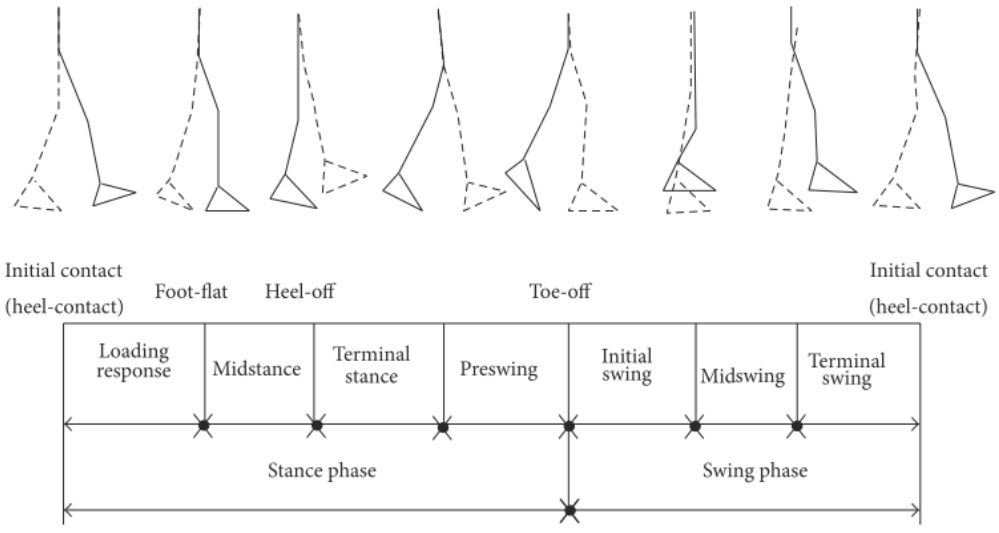


Figure 2.1. Gait cycle [2].

The gait cycle is divided into two phases, which can be seen from Figure 2.1. These phases are called the stance phase and the swing phase. The region where the foot is in contact with the ground is called the stance phase. It involves all of the contact related activities and accounts for 60% of the gait cycle. On the other hand, the region where the foot is not in contact with the ground is called the swing phase. It accounts for 40% of the gait cycle.

Table 2.1. Details of the gait cycle.

Phases	**Actions**	**Point of the Gait Cycle**
Stance Phase	Heel Contact	0%
	Foot Flat	8%
	Mid Stance	30%
	Heel Off	40%
	Toe Off	60%
Swing Phase	Initial Swing	60% - 75%
	Mid Swing	75% - 85%
	Terminal Swing	85% - 100%

2.1.2. Gait Disturbances

Medical professionals can identify gait disturbances by simply observing the way a person moves his or her body. This analysis can provide valuable information, and it is called observational gait analysis. It is an easy and straightforward technique.

There are various pathologies to cause gait disorders such as trauma, stroke, brain injury, multiple sclerosis, cerebral palsy, muscular dystrophy, and so on [3]. Because of the significant impact of gait on daily life, a compensatory action generally has to be taken before solving the root of the problem [4]. Exoskeletons are designed and can be utilized for diagnosis and rehabilitation of gait disorders.

2.2. Ankle Foot Orthosis

Exoskeletons are wearable devices with parts similar to those of the human body. They are designed to improve, reinforce or restore the muscular performance. An exoskeleton works in tandem with its wearer, unlike an autonomous system.

Exoskeletons can be generally divided into three groups: upper limb systems, lower limb systems, and integrated upper and lower limb systems. They can be categorized by their purposes as well [5].

An ankle-foot orthosis or AFO is a lower limb exoskeleton which encompasses the foot, ankle, and leg. AFOs are among the most commonly used orthoses. They are designed with a variety of goals including controlling alignment, increasing mobility, assisting rehabilitation, reducing pain and preventing deformities.

Figure 2.2. A commercial AFO example [6].

AFOs can be divided into two groups: passive and active. Passive AFOs do not contain any electronic elements or any power sources. It may have mechanical elements such as springs or dampers. Active AFOs contain an onboard or tethered power source, actuators, sensors, and control system [3].

2.2.1. Passive Ankle Foot Orthosis

Compactness is a critical factor for orthoses in daily usage. Therefore, the most commonly used AFOs on a daily basis are generally passive. No electronic components or power sources exist in passive AFOs. However, mechanical components can be used in passive AFO design.

Passive AFOs can be divided into two groups: articulated and nonarticulated devices. Moreover, they can be subdivided by material into four groups: metal and leather, composite, thermoplastic, and hybrid systems [3].

(a) Metal and leather AFO (b) Posterior leaf spring AFO

Figure 2.3. Passive AFO examples [6, 7].

Figure 2.3 shows some examples of passive AFO types. Figure 2.3(a) shows an example of a metal and leather AFO type. This AFO type is in the articulated device group. Movement of the ankle joint can be controlled or assisted by articulating joints. Figure 2.3(b) shows an example of a posterior leaf spring AFO type. This AFO type is in the nonarticulated device group. This type of single-piece AFOs is generally made of thermoplastic materials [3]. In conclusion, passive AFOs utilizes direct physical resistance to prevent undesired foot motion. However, what can be accomplished by them is limited.

2.2.2. Active Ankle Foot Orthosis

Despite the popularity of passive AFOs, they have limited robustness and adaptability. In this regard, active AFOs come into the game. Active AFOs contain an onboard or tethered power source, actuators, sensors, and control system. With these components, they overcome these limitations.

Active systems take advantage of sensor feedback to determine and adapt their tasks. For instance, they can use sensor feedback to control their actuators. Active AFOs can be divided by their usage area into two groups: AFOs intended for daily wear and AFOs intended for patient diagnostics and rehabilitation [3].

An AFO with a tethered power source is not practical for daily wear. Therefore, the development of better external powered AFOs is a significant topic. Researchers may initially use the tethered version of their AFO design to speed up the development of a prototype.

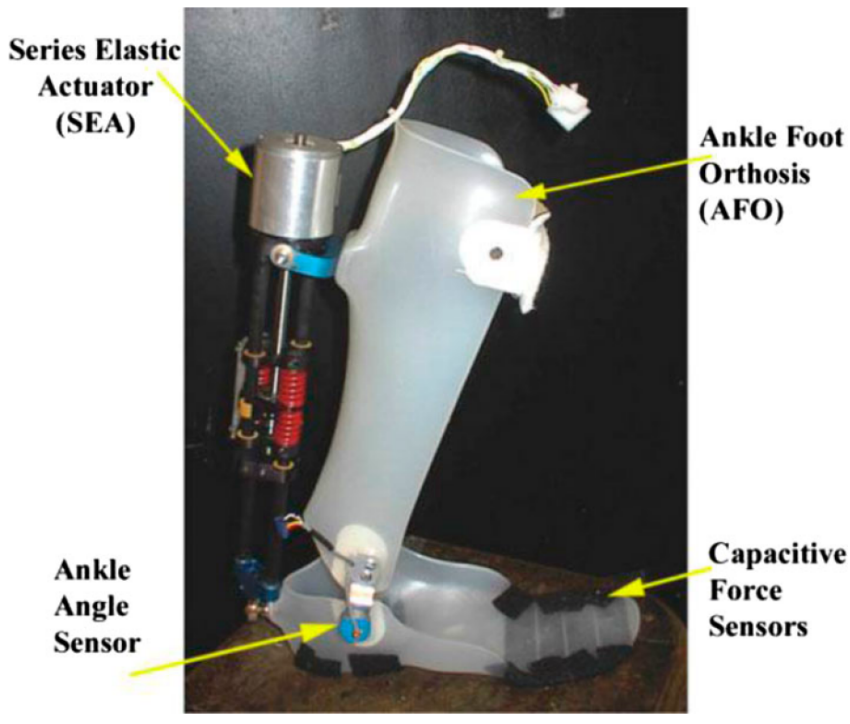


Figure 2.4. An active AFO example [8].

Figure 2.4 shows an example of an active ankle foot orthosis. This AFO utilized a series elastic actuator (SEA) for its actuator [8]. Different types of actuators can be used in active AFO designs.

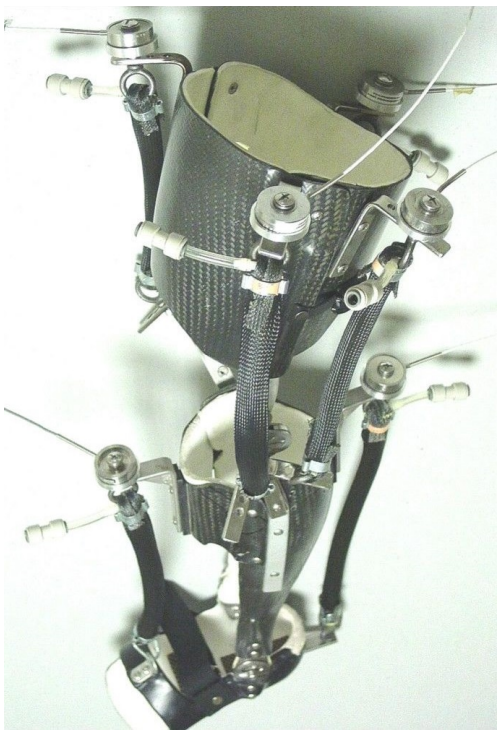


Figure 2.5. A knee ankle foot orthosis example [9].

Some AFO designs can extend to the knee joint, which can influence walking dynamics. This type of device is called knee ankle foot orthosis (KAFO). Figure 2.5 shows a knee ankle foot orthosis example from the University of Michigan [9]. It was designed for human locomotion study and gait rehabilitation.

There are many different implementations of actively powered AFOs. The differences can be in mechanical design, type of actuator, control system, control algorithm, and electronics (mainly controllers). These implementation design decisions generally depend on the usage area and the use case of the AFO. For instance, some use cases may require a control algorithm with a high computational cost. This requirement, as a result, directly affects the controller's specifications such as speed, power consumption, overall size, and so on.

Researchers have utilized different types of actuators in their AFO designs. An ankle-foot orthosis with a force controllable series elastic actuator was proposed [8]. Artificial pneumatic muscles were used as an actuator for an ankle-foot orthosis [10]. An AFO was developed which contains a voice coil actuator (VCA) with mechanical stops [11].

Another vital design decision is the controller choice for AFO. In recent years, the computational cost of the control algorithm for AFOs has been rising. More and more focus has been put on developing better control systems. Algorithms have started to process more feedbacks to increase precision. At the beginning of these studies, multi-purpose computers were the primary controller choice for these developed systems [12]. However, this made the control systems of AFOs huge and not portable.

In the present years, researchers have turned to better standalone systems to overcome the drawbacks of the multi-purpose computers. Nowadays, the most popular controller choice for AFOs is National Instruments's reconfigurable input/output devices, also known as NI RIO devices [12–16]. More specifically, NI CompactRIO [12,13], NI Single-Board RIO [14], and NI myRIO [15,16] devices are used as controllers. These devices share the same RIO technology, just in different form factors.

There has been ongoing research and development for untethered versions of the powered exoskeletons as well. A KAFO with a rechargeable battery of 24 V and 2 hours of operational time was developed [17]. A hydraulic AFO was designed with a 3300 mAh, 29.6 V lithium-ion polymer (LiPo) battery [18]. A unilateral hip exoskeleton with a 2500 mAh, 6 cells LiPo battery was developed [19]. Finally, a knee orthosis with a 2800 mAh battery and 2 hours of continuous operation was implemented as well [20].

2.3. NI CompactRIO Implementation of AFO

This section describes the details of the NI CompactRIO based AFO implementation from the M.S. thesis of Oğuzhan Kırtaş [13]. The mechanical prototype of this previous study is used in the embedded system implementation of AFO that is going to be presented in Chapter 3 as well. The section consists of three parts, which are mechanical design, controller design, and implementation.

2.3.1. Mechanical Design

Before forming a full mechanical design, system requirements must be calculated. In Figure 2.6, average ankle moment values during a normal gait cycle are shown. Dorsiflexion is the movement of raising the foot upwards. On the other hand, plantarflexion is the opposite of dorsiflexion. It is the action of moving the foot downwards. Based on Figure 2.6, the maximum moment applied during dorsiflexion is 1.1 Nm/kg while the maximum moment applied during plantarflexion is 0.3 Nm/kg.

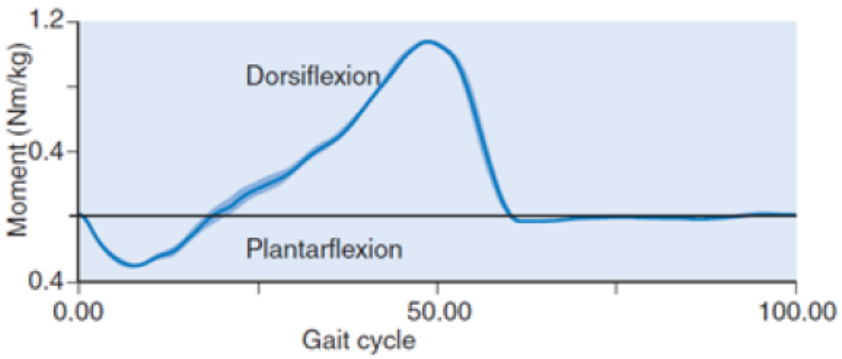


Figure 2.6. Average ankle moment during a normal gait cycle [21].

A free body diagram of the system was modeled to determine the requirements for the AFO prototype. The diagram is shown in Figure 2.7. The model was made of four parts [13]:

- Tibia (TB),
- Hindfoot (HF),

- Forefoot (FF),
- Hallux (HX).

Artificial muscle named M produces a force called P. The spring is denoted by S, and M_a represents the moment on the ankle joint. L is assumed as 0.08 m, which symbolizes the moment arm [13].

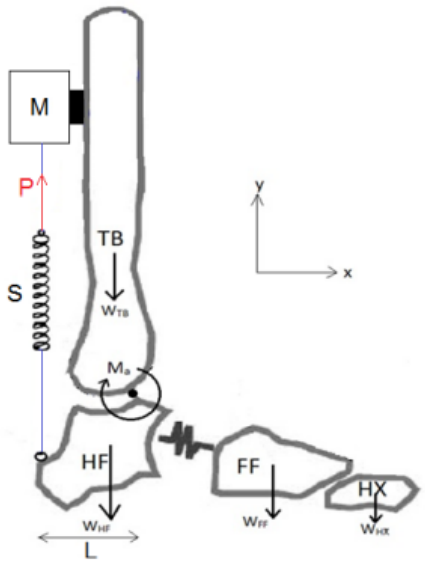


Figure 2.7. Free body diagram of the AFO prototype [13].

A subject of 80 kg was considered for the design. The maximum moment acting on the ankle joint was calculated as

$$(1.1\ Nm/kg)\,(80\ kg) = (88\ Nm)\,.$$

As a result, the maximum force required by the artificial muscle with the moment arm L was calculated as

$$(88\ Nm)\,/\,(0.08\ m) = (1100\ N)\,.$$

Based on these calculations, the actuation system of the AFO was designed. Because of the simplicity of its control, a DC motor was preferred as the main actuator of the

system. Moreover, a ball screw was required for the conversion of rotary motion into linear motion [13].

The ball screw driving torque equation was utilized to determine the specifications for the motor and the ball screw. The ball screw driving torque equation is defined as

$$T = \frac{F_a p}{2\pi\eta} \tag{2.1}$$

where T is the motor driving torque (Nm), η is the ball screw efficiency, F_a is the axial load (N), and p is the ball screw pitch (m). A ball screw with a 2 mm pitch and 90% efficiency was chosen in the design. The required axial load was previously calculated as 1100 N. By using these values in Equation 2.1, the motor driving torque needed for the design was calculated as 0.39 Nm. Based on the required driving torque, a Maxon 150 Watt DC brushed motor was selected [13]. The motor's specifications are as follows

- Nominal voltage of 24 V,
- Nominal speed of 6940 rpm,
- Nominal torque of 0.177 Nm,
- Nominal current of 6 A,
- Stall torque of 0.242 Nm,
- Weight of 0.48 kg.

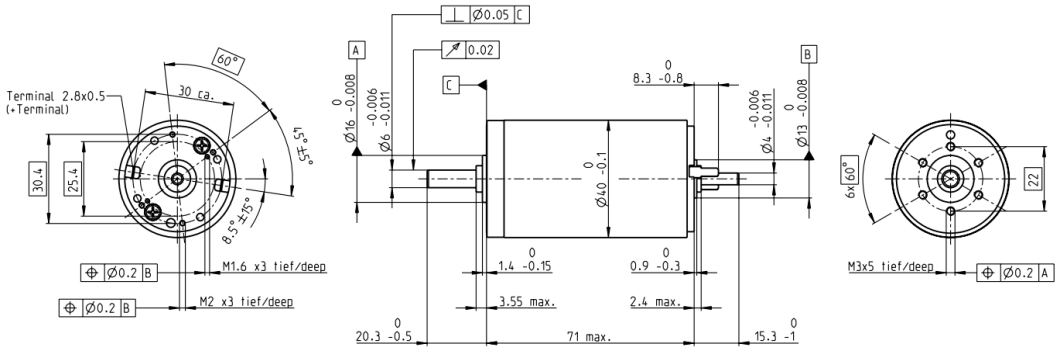


Figure 2.8. Maxon DC brushed motor [22].

A linear compression spring was also added to complete the actuator design. 0.084 m of free length and 50% of maximum spring deflection were aimed for the design. As a result, a linear compression spring with a 16 mm outer diameter and 3 mm wire diameter was chosen [13].

Upon completing the actuator system design, the AFO prototype was manufactured. The total weight of the AFO prototype was measured as 1.8 kg. Three sensors were introduced in the prototype. These sensors comprise a force sensitive resistor (FSR), a linear potentiometer, and an incremental encoder. AFO prototype can be seen in Figure 2.9 [13].

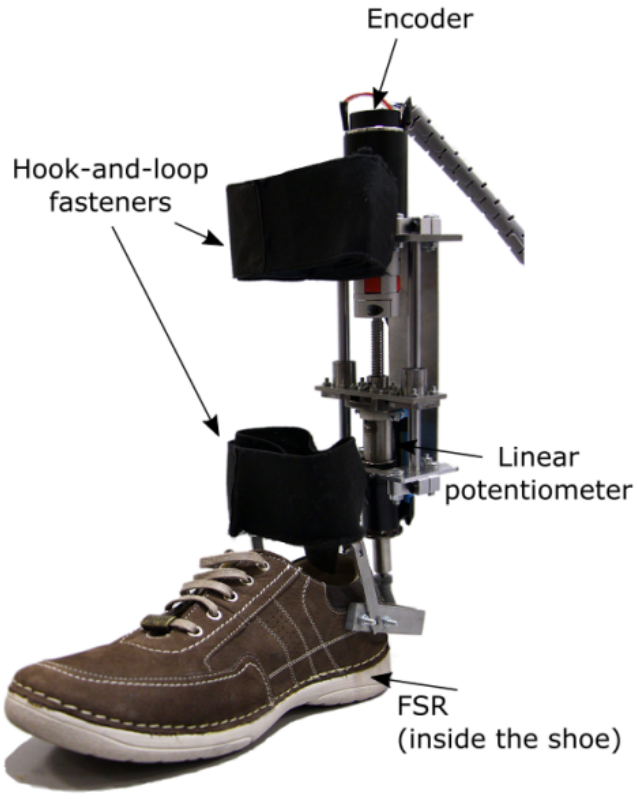


Figure 2.9. AFO prototype [13].

2.3.2. Controller Design

This section describes the controller part of the AFO that was designed with NI CompactRIO [13]. Controller device selection is a crucial design decision for an AFO, and it directly affects the size, power consumption, portability, and precision of the overall system. For these reasons, device details are described as well.

NI CompactRIO utilizes the reconfigurable I/O (RIO) technology from National Instruments [23]. RIO technology is the underlying technology behind RIO architecture, which is ideal for designing advanced control systems. RIO architecture comprises of four components [24]:

- Real-Time Processors,
- User-Programmable FPGAs,
- Modular I/O Interfaces,
- Complete Software Toolchain.

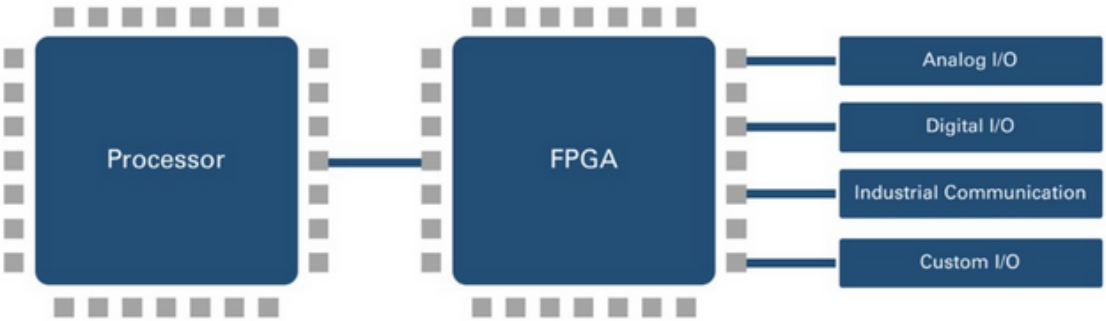


Figure 2.10. RIO architecture [24].

Processing targets of the NI CompactRIO consist of a real-time processor and a field-programmable gate array (FPGA). The processor can be an ARM or Intel processor ranging from ARM Cortex-A9 to Intel Atom. Specifically, the processor of the NI CompactRIO can be one of the three cores [24]:

- 667 MHz Dual-Core ARM Cortex-A9,
- 1.33 GHz Dual-Core Intel Atom,
- 1.91 GHz Quad-Core Intel Atom.

The processor runs NI Linux Real-Time OS which is based on a standard kernel. This operating system (OS) is designed for long-term deployments. It contains standard Linux features as well [24].

The other processing target of the NI CompactRIO is the FPGA. Time-critical operations are always necessary in real-time applications, and the user-programmable FPGA can offload these operations. FPGA is directly connected to I/O for low control loop latency and high-performance signal. NI CompactRIO uses FPGA technology from Xilinx [24].

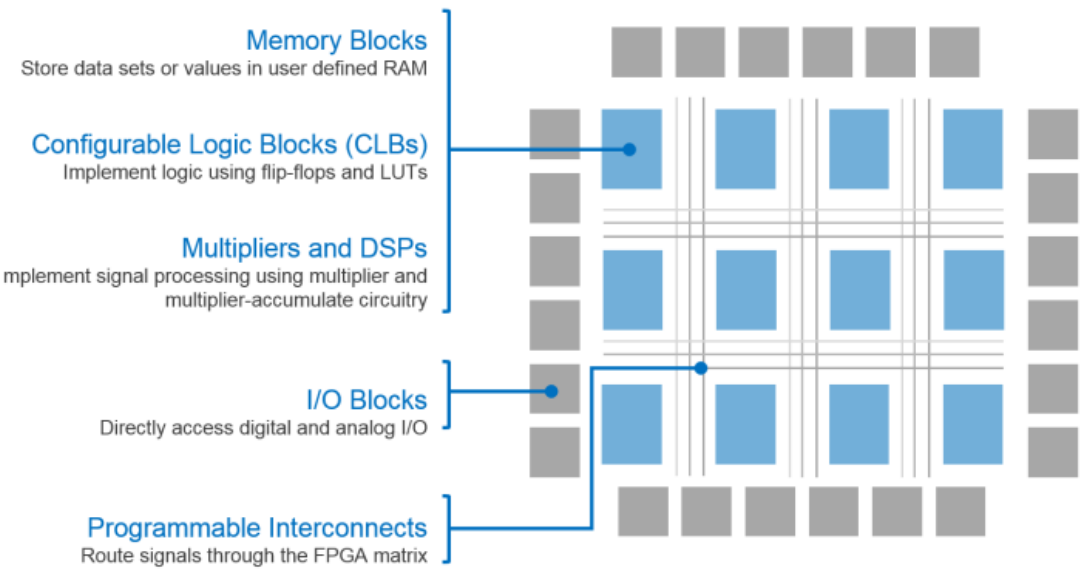


Figure 2.11. FPGA logic circuitry [24].

Complementary parts of the NI CompactRIO are modular I/O interfaces and complete software toolchain. NI I/O modules eliminate the necessity for separate subsystems. These modules can be plugged into NI CompactRIO. As a result, NI CompactRIO can be directly connected to components like motors, sensors, cameras, etc. NI offers over 100 different I/O modules [24].

Figure 2.12. NI I/O modules [24].

LabVIEW is a graphical environment to program the processor cores and the FPGA of the NI CompactRIO. It can be utilized without any knowledge of programming languages or hardware description languages [24]. The software environment can be extended with add-on software packages as well. For instance, the LabVIEW advanced signal processing toolkit adds a set of software tools for the analysis of time-frequency, time series, and wavelets.

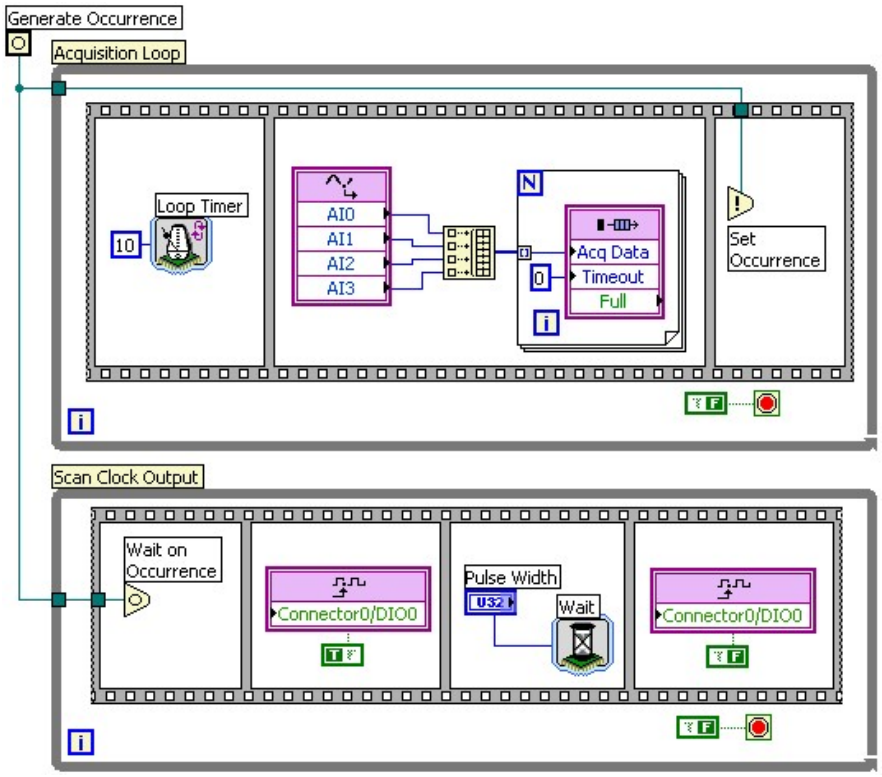


Figure 2.13. LabVIEW environment [24].

NI CompactRIO model used for the AFO is cRIO-9035 [13]. This model has 1.33 GHz dual-core Intel Atom as the central processing unit (CPU), Kintex-7 70T as the FPGA, 4 GB hard drive, 1 GB RAM, and 8 I/O slots.

Table 2.2. Detailed specifications of the NI CompactRIO.

PROCESSOR	
CPU	Intel Atom E3825
Number of Cores	2
CPU Frequency	1.33 GHz
Internal L2 Cache	1 MB (Shared)
FPGA	
FPGA Type	Xilinx Kintex-7 7K70T
Logic Cells	65600
Number of Slices	10250

Each FPGA slice contains four LUTs and eight flip-flops. Moreover, FPGA has 240 DSP slices that each of them contains a pre-adder, a 25x18 multiplier, an adder, and an accumulator.

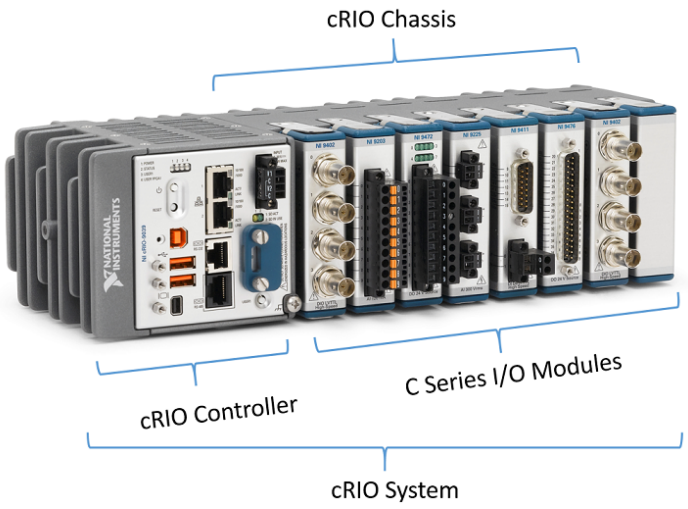


Figure 2.14. NI CompactRIO system [23].

2.3.3. Implementation

Mechanical and controller designs of the AFO were the major parts of the AFO implementation. These modules were joined together to complete the implementation. The name of the implementation was chosen after the controller device selection, which was NI CompactRIO.

The processor and FPGA modules of the NI CompactRIO were assigned to different tasks. As an overview, the main task of the FPGA was to read sensor values, generate pulse width modulation (PWM) signal, and pass it to the DC motor. On the other hand, the main task of the real-time processor was to calculate the required motor current according to the control algorithm and pass it to the FPGA [13]. Block diagram of the NI CompactRIO based AFO implementation is shown in Figure 2.15.

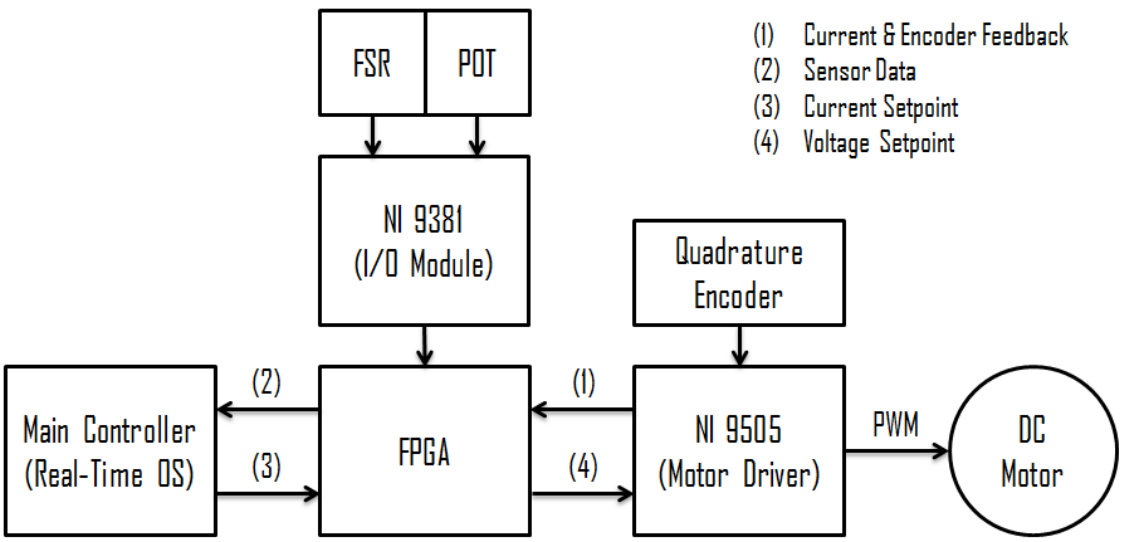


Figure 2.15. Block diagram of the NI CompactRIO implementation.

Two NI CompactRIO I/O modules were used. These were NI 9505, which was a brushed DC motor driver and NI 9381, which was a multifunction I/O module. The required motor current value was calculated and passed to the FPGA. Then, FPGA would use this current value to generate a PWM signal and send it to the NI 9505. Finally, NI 9505 would control the DC motor with the PWM signal. Also, NI 9505

would pass the quadrature encoder signals to the FPGA. Quadrature encoder signals were used to sense the position of the motor. On the other hand, NI 9381 was used to read sensor data values with its analog to digital converters. The linear potentiometer and FSR values were utilized in the control algorithm. 5 V electric potential was supplied to the I/O module from the system [13].

To power the whole system, two power supplies were used. The model of the power supplies was NI PS-15, which was a 5 A, 24 VDC power supply. One of them was used to supply power to the cRIO controller. The other one was used to provide power to the motor driver I/O module (NI 9505). The module would use this power to control the DC motor [13].

Figure 2.16. NI PS-15 power supply [25].

The real-time processor of the NI CompactRIO was in charge of the main control algorithm. The control algorithm would calculate the force input, which was converted to the required motor torque by utilizing the ball screw driving torque equation (Equation 2.1). The final output of the algorithm was the required motor current, which was obtained by using the equation

$$i = \frac{T}{c_t} \tag{2.2}$$

where i is the required motor current, and c_t is the motor torque constant.

3. EMBEDDED SYSTEM IMPLEMENTATION OF AFO CONTROL

3.1. Overview

Ankle-foot orthosis designs can have various implementations. Each part of the AFO can alter the overall implementation. One of the most critical parts of the AFO is the control system, which generally comprises a controller device performing a specific control algorithm. The control system directly affects the usage area and the use case of the AFO. As a result, this part of the AFO should be cautiously designed and implemented.

At the beginning of the AFO studies, the primary controller choice was the multi-purpose computers [12]. However, multi-purpose computers had many drawbacks for an AFO controller. Since these computers were designed for various purposes, many components and features would be unused, making the overall system unnecessarily huge and not portable.

In recent years, researchers have turned to better standalone systems to overcome the drawbacks of the multi-purpose computers. Also, more and more focus has been put on developing control algorithms with greater precision. These tendencies have caused a rise in the computational costs of the algorithms. To meet this kind of computational need, more powerful and application specific controllers were preferred. Nowadays, the most popular controller choice for AFOs is NI RIO devices [12–16]. However, this type of controllers also have several shortcomings. In this section, an embedded system implementation of AFO control is presented, which is designed to improve these shortcomings.

Firstly, design requirements were determined to develop an embedded system for the AFO controller. Multiple factors were considered while determining these requirements. The main goal was to design an improved controller alternative compared to the NI CompactRIO based AFO controllers. The main requirements of the design were to improve the performance, power efficiency, size, and mobility of the overall AFO system. Moreover, enhancing the capability of the system with new features was aimed all the while maintaining the same level of functionality required by the mechanical part of the AFO.

Simplifying the system and reducing the complexity are best practices in product design. The proposed embedded system controller followed these practices as well. A single controller block was designed to control the whole AFO. A motor driver and a specialized microcontroller unit with various peripherals form the AFO controller. The block diagram of the proposed system is shown in Figure 3.1.

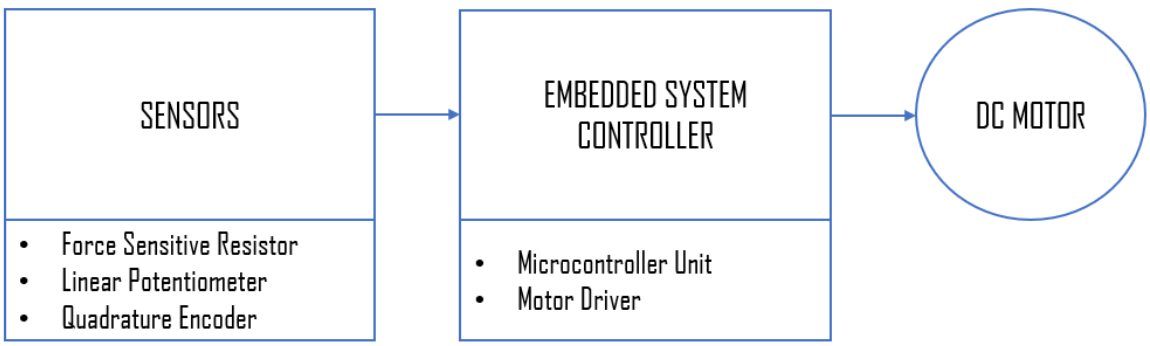


Figure 3.1. Embedded system block diagram.

In summary, the microcontroller unit handles sensor reading with its control peripherals and analog subsystems. It efficiently calculates the required motor current with the help of its accelerators and generates the PWM signal. Finally, the motor driver uses this PWM signal to control the motor. Detailed explanations related to these modules are given in the following sections.

3.2. Microcontroller Unit

In this section, the reasoning behind using a microcontroller, and the selection criteria for it are explained. The section also gives details about the chosen microcontroller unit, such as its specifications, features, and peripherals.

The goal was to use a central control unit that requires no or minimum external peripherals and modules. It must also be powerful enough to run complex control algorithms in real-time conditions and be compact enough to achieve mobility for the system. Based on these requirements, a microcontroller with integrated analog and control peripherals was preferred. More specifically, TMS320F28379D from Texas Instruments was chosen as the microcontroller unit.

Each part of the chip that contributes to the AFO control is explained individually in the following sections. In addition to the specifications related to that specific part of the chip, the role and the usage details in the AFO control is also described. In the end, all of the selection criteria and fundamental design decisions associated with the microcontroller unit would have been explained.

3.2.1. Device Overview

TMS320F28379D from Texas Instruments (TI) is a powerful 32-bit floating-point microcontroller unit (MCU) designed for advanced control applications, sensing, and signal processing. Designers can eloquently combine control structures by using the integrated analog and control peripherals of the chip. This practice eliminates the multiprocessor and external peripheral use in high-end systems [26].

MCU has 32-bit C28x floating-point CPUs, which are optimized for sensing, processing, and actuation to improve closed-loop performance in real-time applications. These CPUs are further enhanced by the Trigonometric Math Unit (TMU) accelerator, which enables fast execution of algorithms with trigonometric operations common in transforms and torque loop calculations [26].

MCU features additional real-time control coprocessors, which are called Control Law Accelerators (CLAs). The CLA, which runs at the same speed as the main CPU, is an independent 32-bit floating-point processor. These coprocessors bring parallel processing capability to the system. As a result, the computational performance of a real-time control system can be effectively doubled [26].

Integrated analog and control peripherals on the MCU further enable system consolidation. For instance, independent Analog-to-Digital Converters (ADCs) provide precise and efficient management of multiple analog signals, which ultimately boosts system throughput [26]. There are several control peripherals on the MCU to improve the performance of the system as well. Features and peripherals of the MCU utilized specifically in the AFO controller are as follows:

- Dual-Core Architecture
  - (i) Two TMS320C28x 32-Bit CPUs (200 MHz)
  - (ii) IEEE 754 Single-Precision Floating-Point Unit (FPU)
  - (iii) Trigonometric Math Unit
- Two Programmable Control Law Accelerators (200 MHz)
  - (i) IEEE 754 Single-Precision Floating-Point Instructions
- On-Chip Memory
  - (i) 1 MB of Flash and 204 KB of RAM
- System Peripherals
  - (i) Dual 6-Channel Direct Memory Access (DMA) Controllers
  - (ii) Expanded Peripheral Interrupt Controller (PIE)
- Communications Peripherals
  - (i) Three High-Speed Serial Peripheral Interface (SPI) Ports
- Analog Subsystem
  - (i) Four Analog-to-Digital Converters
- Enhanced Control Peripherals
  - (i) 24 Enhanced Pulse Width Modulator (ePWM) Channels
  - (ii) Three Enhanced Quadrature Encoder Pulse (eQEP) Modules

3.2.2. C28x Processor

The C28x CPU is an exclusive 32-bit core from Texas Instruments's C2000 series. These CPUs, combined with powerful integrated peripherals, are integral to the real-time, single-chip control solutions. Designers have the flexibility to improve system efficiency and system reliability with these C28x math-optimized CPUs.

The reason why the C28x CPU was selected for the AFO controller is that this CPU brings the code density and execution speed of a DSP with the ease of use and accessibility of an average microcontroller. Furthermore, these processors are integrated with on-chip hardware accelerators that dramatically increase the performance.

The CPU draws from the best features of digital signal processing, reduced instruction set computing (RISC), and microcontroller architectures, such as modified Harvard architecture, circular addressing, single-cycle instruction execution, and register-to-register operations. For instance, the modified Harvard architecture of the CPU enables instruction and data fetches to be performed in parallel [26].

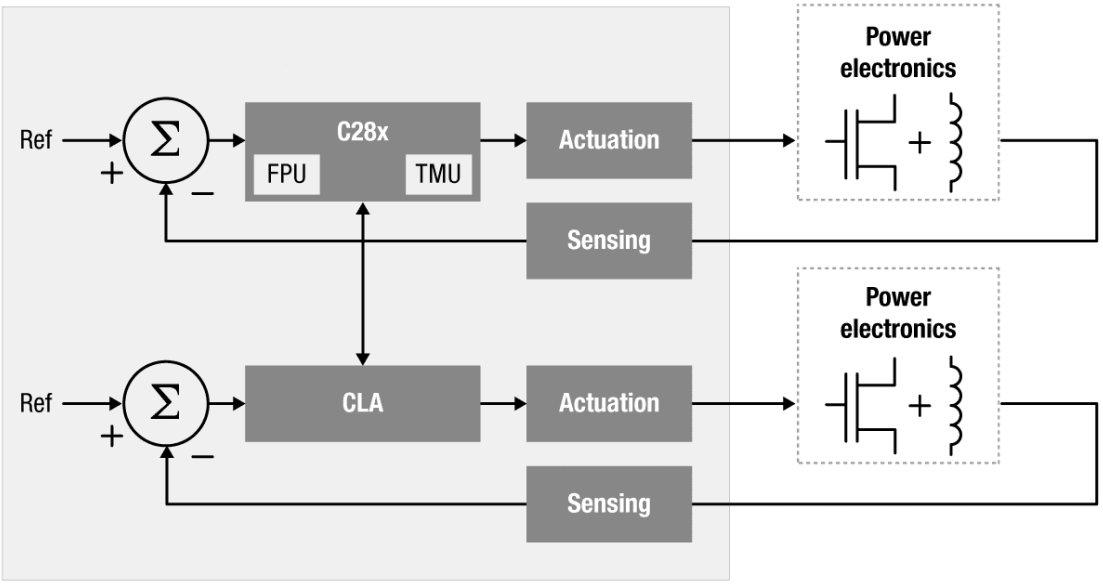


Figure 3.2. C28x processing capabilities.

3.2.3. Floating-Point Unit

Generally, advanced control system algorithms are developed with floating-point math, which provides a large dynamic range. With this range, the programmer typically no longer needs to worry about scaling and saturation. On an overflow or underflow, floating-point values do not wrap around the number line, unlike the fixed-point values. Robustness is improved due to this property [27]. The proposed AFO controller was designed with FPU to benefit from these characteristics.

The addition of the FPU extends the capabilities of the C28x CPU by adding registers and instructions to support IEEE single-precision floating point operations. These additional registers are the following:

- Eight Floating-Point Result Registers,
- Floating-Point Status Register,
- Repeat Block Register.

Zero overhead looping is added by the repeat block, which enables flexibility to the processor. Excluding the repeat block register, all of the floating-point registers are shadowed. Register shadowing is particularly useful for control system applications to reduce latency during context saving and restoring floating-point registers [27].

Most FPU instructions operate in one or two pipeline cycles because of these instructions being extensions of the standard C28x instruction set. Thanks to the hardware support of the FPU, single cycle instructions can be performed on multiply-accumulate, fast inverse and inverse square root operations, and data conversions (integer to float). Floating-point performance dramatically enhances the mathematical computation capability required in signal processing and control algorithms. On the average, greater than 2.5 times performance enhancement can be achieved with floating-point math compared to fixed-point math [27].

3.2.4. Trigonometric Math Unit

In control system applications, trigonometric and arithmetic operations are frequently utilized. The TMU extends the capabilities of the C28x CPU by adding instructions that are used for efficiently executing these operations. The TMU is an IEEE-754 single-precision floating-point hardware accelerator that is focused explicitly on trigonometric math functions. It enhances the C28x+FPU by accelerating several specific trigonometric math operations that would otherwise be very cycle intensive. These operations include square root, divide, cosine, sine, and arctangent [27]. The summary of TMU supported instructions is given in Table 3.1.

Table 3.1. The summary of TMU supported instructions [27].

Operation	**C Equivalent Operation**
Sin Per Unit	a=sin(b*2pi)
Cos Per Unit	a=cos(b*2pi)
Arctangent Per Unit	a=atan(b)/2pi
Square Root	a=sqrt(b)
Divide	a=b/c
Multiply by 2*pi	a=b*2pi
Divide by 2*pi	a=b/2pi

On many commonly used real-time control algorithms, the TMU can have a notable impact. For instance, an 85 percent performance increase can be achieved on a Park Transform with TMU. In a typical control application, about 1.4 times performance improvement can be gained using the TMU over just the FPU [27].

The proposed AFO controller performs a particular control algorithm with many trigonometric operations to calculate the required motor current. Accordingly, the controller was designed with TMU to benefit from the performance boost.

3.2.5. Control Law Accelerator

The CLA is a 32-bit floating-point hardware accelerator, which can be programmed completely. It is designed to accelerate math-intensive computations. The CLA and the C28x CPU can execute real-time control algorithms in parallel. This capability effectively doubles the performance of the system. The CLA has a couple of advantages over the CPU that boosts the performance even more. For instance, it accesses memory directly, which removes the data page pointer overhead. Moreover, no delay slots are needed for the CLA multiplier. These features result in about 1.3 times performance improvement when compared to the C28x CPU. When the CLA is used, the C28x CPU can service other tasks [27].

Latency reduction can be achieved because of the direct peripheral access capability of the CLA. For example, the CLA can read the registers of the ADC and PWM modules faster than the C28x CPU. Moreover, the peripherals can directly trigger the CLA without any CPU intervention. No context switching is made by the CLA when this happens. All of these improvements lead to a more power-efficient system [27].

3.2.6. Direct Memory Access

Generally, a significant amount of time is spent by the CPU to perform data transfer. Also, this data may not be in an optimized format. The DMA is designed to be utilized in these situations. In a nutshell, the DMA is used to transfer data between memory or peripherals without any CPU intervention. It can ping-pong data between buffers and rearrange data orthogonally during transfer [26].

A 6-channel DMA module is available to each C28x CPU. Each DMA channel can use a different interrupt trigger source. These interrupts are used to give information about the DMA transfer status. Status information includes the start and end of the DMA transfer. The priority of the channels can be configured as well [28].

3.2.7. Peripheral Interrupt Controller

An interrupt service routine (ISR) is a software routine that is invoked by an interrupt signal. Current execution is paused when that happens. Interrupt-based systems are superior to polling-based systems in terms of performance. An interrupt-based system must be carefully designed to avoid resource conflicts [28].

A total of fourteen interrupt lines can be used for the C28x CPU. Two of them are directly connected to the CPU timers. Remaining interrupts are used for peripheral interrupt signals. The PIE module extends the interrupt vector table, allowing it to have an ISR. Each C28x CPU core has its PIE module [27].

3.2.8. Serial Peripheral Interface

The serial peripheral interface is a high-speed synchronous serial I/O port that is generally used for communication between the CPU and external peripherals or controllers. By using SPI, a serial bit stream of programmed length can be transferred at a programmed bitrate. Master and slave operations are supported, which allows multi-device communications. Also, the port has its own receive and transmit FIFO, which reduces CPU overhead [28].

SPI utilizes four pins to communicate. These pins include:

- $SPISOMI$: SPI Slave Output Master Input Pin
- $SPISIMO$: SPI Slave Input Master Output Pin
- $\overline{SPISTE}$: SPI Slave Transmit Enable Pin
- $SPICLK$: SPI Serial Clock Pin

It has two operational modes, which are master and slave. 125 different baud rates can be programmed to be used for communication. The clocking scheme of the communication can be adjusted with clock polarity and clock phase bits. It also supports simultaneous transmit and receive operations [28].

3.2.9. Analog-to-Digital Converter

The type of ADC module used is a successive approximation style ADC with either 12 or 16 bits selectable resolution. The wrapper refers to the digital circuits of the converter. It comprises of logic for programmable conversions, interfaces to other modules, and result registers. The core refers to the analog circuits of the converter, such as the sample-and-hold circuit, the successive approximation circuits, and other analog circuits. The wrapper and the core blocks together form the ADC module [28].

Four independent high-performance ADC modules are available to the microcontroller. Start-of-conversion registers can be configured to handle the ADC triggering and conversion sequencing. Several trigger sources can start the ADC conversion, such as control peripherals, timers, external pins, and so on. After the conversion is completed, an ADC interrupt can be triggered [29].

3.2.10. Enhanced Pulse Width Modulator

PWM is commonly used to control power switching devices. By nature, it is a digital signal and performs a DAC function where the duty cycle is equal to the analog amplitude value. The ePWM module, which is a highly programmable hardware module, is designed to generate complex pulse width waveforms with minimal CPU overhead. Each ePWM module has two PWM outputs which are made available externally through the GPIO peripheral [29].

The ePWM module comprises eight submodules: counter-compare, time-base, PWM chopper, digital-compare, dead-band generator, action-qualifier, trip-zone, and event-trigger. The period and frequency control of the PWM can be configured with the dedicated 16-bit time-base counter. Two PWM outputs can be used in different configurations such as single-edge, dual-edge, symmetric, and asymmetric operations. All events of the module can trigger CPU interrupts [28].

3.2.11. Enhanced Quadrature Encoder Pulse Module

Slots on an incremental encoder disk generate an alternating pattern of light and dark lines. The disk count is calculated as the number of line pairs that occur per revolution. Generally, there is an extra track generating a signal that occurs once per revolution. The encoder manufacturers name this signal with various terms such as index signal, zero reference, and home position [28].

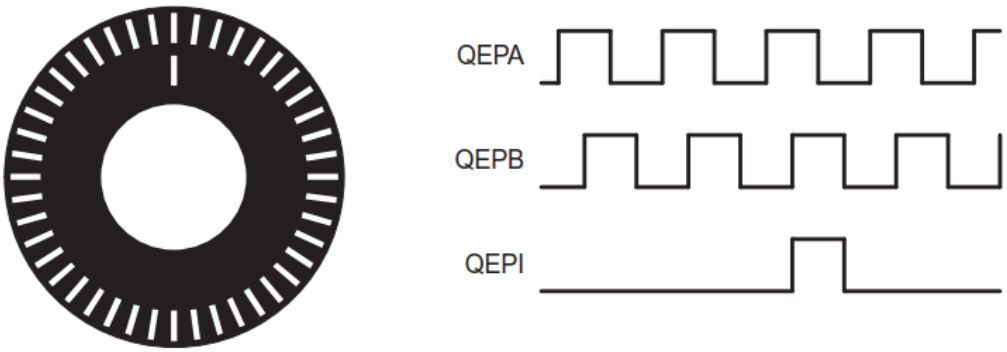


Figure 3.3. Optical encoder disk [28].

The eQEP module is a hardware module that is used to determine the position, direction, and speed information from a rotating machine by interfacing with a linear or rotary incremental encoder. There are four inputs to the module, which are QEPA, QEPB, QEPI, and QEPS. QEPA and QEPB, which are 90 electrical degrees out of phase, are used to calculate the position count and speed information. Also, the direction of rotation is determined from this phase difference. QEPI is used to indicate an absolute start position. QEPS is an optional strobe signal, which can be used to indicate that a defined position has been reached [28].

3.3. Motor Driver

The BOOST-DRV8711 module was selected as the motor driver of the proposed AFO controller. This module is capable of driving dual brushed DC motors. Moreover, it is a BoosterPack based module that is compatible with the MCU of the AFO controller. Both the MCU and the motor driver has a special connector named BoosterPack. The availability of this kind of connector can make the overall system significantly more compact. For this reason, the motor driver with a BoosterPack connector was chosen.

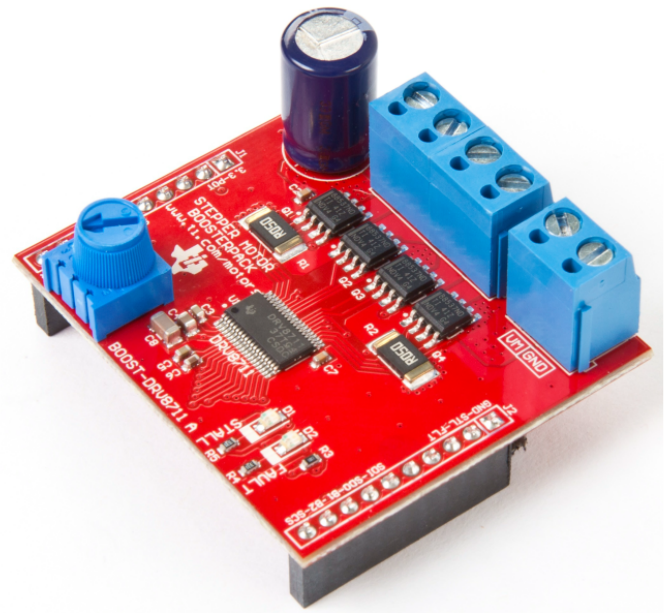


Figure 3.4. BOOST-DRV8711 module [30].

This motor driver is labeled as a stepper motor driver. However, it can be repurposed as a dual brushed DC motor driver. PWM mode configuration must be done to accomplish this. Furthermore, the motor driver has a fully protected drive stage, including overtemperature, overcurrent, undervoltage, and motor stall detect. When combined with the TMS320F28379D MCU, they create a complete, powerful control platform.

Pinout of the BOOST-DRV8711 is given in Figure 3.5, which shows the signal types and pin names.

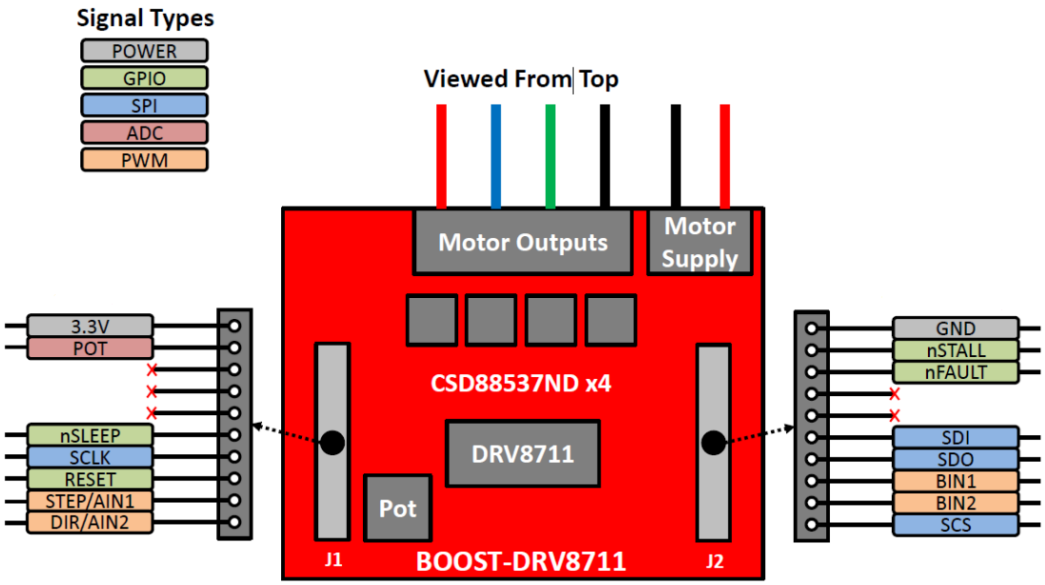


Figure 3.5. BOOST-DRV8711 pinout [30].

There are four pins for motor outputs, which can be used to drive two brushed DC motors if two pins are directed to each motor. Fault and motor status reporting are done through nFAULT and nSTALL signals. Communication to the motor driver can be performed over the SPI. For example, configuring operating parameters, setting device configuration, and reading diagnostic information can be done over this way.

The BOOST-DRV8711 comprises of two main parts:

- DRV8711 Motor Pre-Driver
- CSD88537ND Dual 60-V N-Channel NexFET^{TM} Power MOSFET

The DRV8711 motor pre-driver is a motor controller that uses external N-channel MOSFETs to drive a bipolar stepper motor or two brushed DC motors [31]. On the other hand, the CSD88537ND is a power MOSFET that can be used as a half bridge in motor control [32].

Programming of the motor driver is performed with a determined serial data format over SPI. The serial data comprises 16 bits in total. These are 12 data bits, 3 address bits, and a read/write bit. SPI chip select must be set to a logic 0 to complete the read or write transfer [31].

Internal registers of the motor driver control the operation of the motor. There are eight registers which are CTRL, TORQUE, OFF, BLANK, DECAY, STALL, DRIVE, and STATUS. Bit-fields of these registers must be configured and tuned according to the specifications of the motor. For instance, PWMMODE bit-field must be used to control a brushed DC motor instead of a stepper motor. The detailed register map is given in the Figure 3.6 [31].

DRV8711 REGISTER MAP

Name	11	10	9	8	7	6	5	4	3	2	1	0		Address Hex
CTRL	DTIME		ISGAIN		EXSTALL	MODE				RSTEP	RDIR	ENBL	RW	00
TORQUE	Reserved	SMPLTH			TORQUE								RW	01
OFF	Reserved			PWMMODE	TOFF								RW	02
BLANK	Reserved			ABT	TBLANK								RW	03
DECAY	Reserved	DECMOD			TDECAY								RW	04
STALL	VDIV		SDCNT		SDTHR								RW	05
DRIVE	IDRIVEP		IDRIVEN		TDRIVEP		TDRIVEN		OCPDEG		OCPTH		RW	06
STATUS	Reserved				STDLAT	STD	UVLO	BPDF	APDF	BOCP	AOCP	OTS	RW	07
Name	11	10	9	8	7	6	5	4	3	2	1	0		Address Hex

Figure 3.6. Motor driver register map [31].

3.4. Battery

The battery was chosen according to the power and design requirements of the proposed system. Maxon 150 Watt DC brushed motor was previously chosen based on the maximum required driving torque calculation. This motor has a nominal voltage of 24 V and a nominal current of 6 A. On the other hand, the proposed system was targeted to be untethered and relatively lightweight. As a result, the lithium-ion polymer (LiPo) battery shown in Figure 3.7 was selected.

Figure 3.7. LiPo battery.

The specifications of this high discharge LiPo battery are as follows:

- Capacity: 3300 mAh
- Voltage: 22.2 V (6S1P)
- Maximum Continuous Discharge: 35 C (115.5 A)
- Weight: 530 g

6S1P suggests six series and one parallel, which means there is only one set of 6 cells connected in series. The maximum continuous discharge current indicates the maximum current at which the battery can be discharged continuously.

3.5. Adaptation of AFO Control Unit to MCU Based System

In this section, details about the adaptation of AFO control unit to MCU based system are given. The goal is to adapt the NI CompactRIO based AFO control system to the MCU based AFO control system without losing any functionality while improving the previous system. The main areas to improve are the performance, power efficiency, size, and mobility of the system.

NI CompactRIO based AFO control system comprised an FPGA, a CPU, a motor driver I/O module, and an analog I/O module. The FPGA part was in charge of the hardware-related functions. The main tasks of the FPGA were to read sensor values and generate PWM signals. The FPGA utilized the analog and the motor driver external I/O modules to read the sensor values and run the DC motor. On the other hand, the CPU part, which ran with the NI Linux Real-Time OS, was in charge of executing a control algorithm to calculate the required motor current. The CPU would read the sensor values from the FPGA as inputs and would write back the required motor current to the FPGA as output.

The proposed MCU based AFO control system only comprises an MCU and a motor driver. No additional external modules are required. Moreover, the motor driver is selected as a specific plug-in module that can be placed on top of the MCU. Thanks to this compatibility, the entire control system is implemented as a single product. All controller tasks are realized by the MCU. At the output, the motor driver is used to run the DC motor.

Two points stand out when comparing these two systems on a broad level. First, the FPGA and the external I/O modules of the NI CompactRIO based system are discarded, which decreases the complexity and the total size of the new system. This, as a result, improves the system mobility. Second, the CPU of the NI CompactRIO based system is replaced with an application-specific MCU, which enhances the performance and the power efficiency of the system.

The proposed system is designed as a bare-metal system. No operating system is used. The reason for this design decision is to eliminate the overhead of an operating system and improve the performance of the system. The system is scheduled with hardware interrupts. Compared to the NI CompactRIO based system, the huge overhead caused by the NI Linux Real-Time OS is removed entirely in the new system. The control flow of the proposed system comprises two stages, which are initialization and execution stages.

3.5.1. Initialization Stage

The first step of the control flow is called the initialization stage. This stage includes the following operations.

(i) MCU System Control Initialization
(ii) Peripheral General Purpose Input Output (GPIO) Initialization
- ePWM
- eQEP
- SPI
- ADC

(iii) Motor Driver GPIO Initialization
(iv) PIE Initialization
(v) CPU Timer Initialization
(vi) Peripheral Configuration and Initialization
- ePWM
- eQEP
- SPI
- ADC

(vii) PIE Configuration
(viii) Motor Driver Configuration and Initialization
(ix) AFO Control Algorithm Initialization

The initialization of system resources is performed at the MCU system control initialization step. Operations, such as initializing system Phase-Locked Loop (PLL), turning on peripheral clocks, and initializing flash control registers, are completed at this step.

At the peripheral GPIO initialization step, GPIO configurations of the peripherals used for the AFO control system are done. These peripherals include ePWM, eQEP, SPI, and ADC. For the ePWM, two special-purpose pins for PWM signal outputs and two general-purpose pins for routing PWM signals are configured. Three special-purpose pins for quadrature encoder inputs are configured during the eQEP configuration. For the SPI, three special-purpose pins for SPI communication pins and one general-purpose pin for slave transmit-enable pin are configured. Lastly, two ADC input pins for FSR and potentiometer are configured.

During the motor driver GPIO initialization step, pins used for motor driver control and status reporting are configured. The functions of these pins include fault condition reporting, stall condition reporting, resetting driver, and enabling low power sleep mode.

PIE control registers are initialized at the PIE initialization step. After this step, CPU timer initialization is performed. One hardware timer is configured to be used for all time measurements. This timer is started at the startup, and time measurements are done via timestamps of this timer.

During the peripheral configuration and initialization step, peripheral settings needed for AFO control are realized. These peripherals, which have crucial roles in the overall AFO control, include ePWM, eQEP, SPI, and ADC. For ePWM, configurations such as time base period and compare settings are done. Then, the unit timer period and quadrature count mode of eQEP are configured. For SPI, the clock phase and baud rate are set. Finally, the resolution of ADC is configured.

Interrupts from peripherals such as ePWM and eQEP are enabled at the PIE configuration step. During the motor driver configuration and initialization step, motor driver registers are set via SPI serial interface. These motor driver registers are used to control settings such as torque, decay mode, stall detection, etc. The last step of the initialization stage is the AFO control algorithm initialization. At this step, state variables of the AFO control algorithm are initialized.

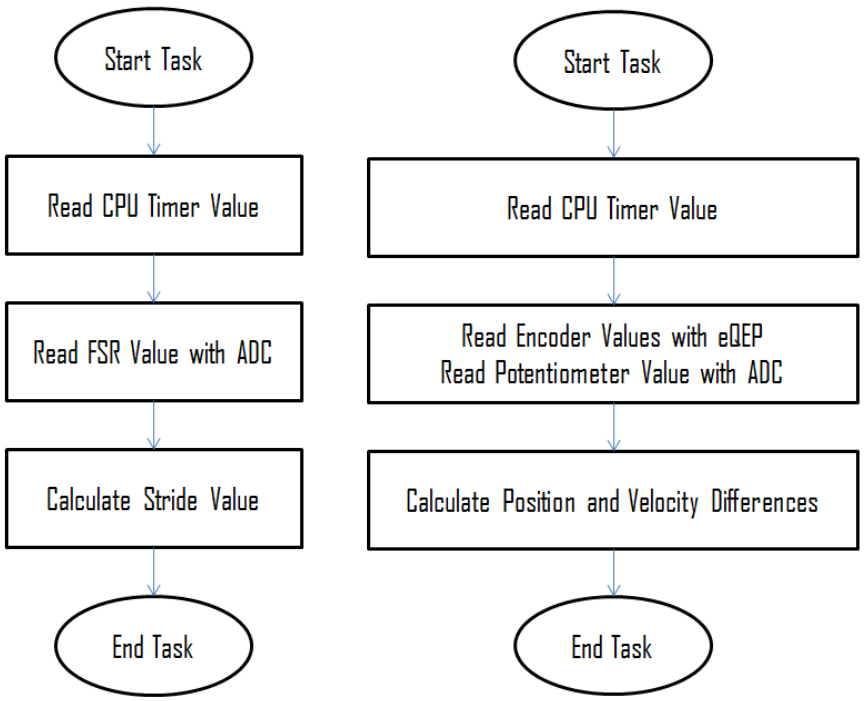
3.5.2. Execution Stage

The second step of the control flow is called the execution stage. This stage is divided into different tasks. The goal of it is to perform AFO control tasks. It comprises following four tasks.

(i) Stride Measurement Task
(ii) Displacement Measurement Task
(iii) Algorithm Execution Task
(iv) PWM Generation Task

Stride measurement operations are performed during the stride measurement task. This task is triggered by the eQEP peripheral interrupt, and uses FSR and elapsed time values as inputs. The FSR value is obtained with ADC. The elapsed time value is calculated with the CPU hardware timer. As an output, the task calculates the stride value.

Position and velocity differences of the AFO are measured during the displacement measurement task. It takes linear potentiometer value and quadrature encoder outputs, which consist of the position counter and the speed value of the motor, as the inputs of the task. The linear potentiometer value is read with ADC. The quadrature encoder outputs are obtained with the eQEP peripheral. Finally, the task calculates the position and velocity difference values. This task is also triggered by the eQEP peripheral interrupt. This type of interrupt-based mechanism reduces the CPU load.

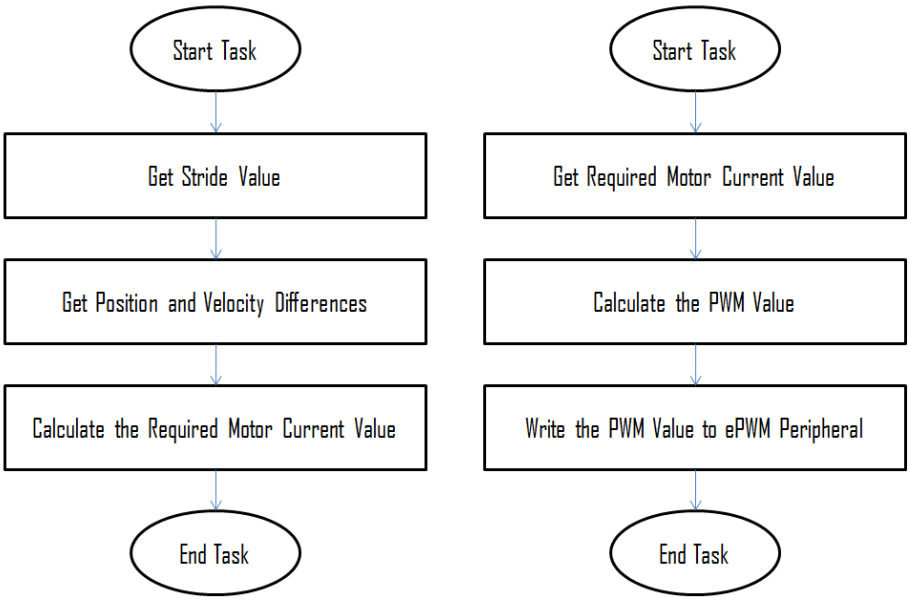


(a) Stride measurement task (b) Displacement measurement task

Figure 3.8. Measurement tasks.

The algorithm execution task calculates the required motor current as an output. It uses the outputs from the stride measurement and displacement measurement tasks. The task is designed to run after the completion of the displacement measurement task. As a result, this task is also part of the interrupt-based mechanism. Proportional integral derivative (PID) control and backstepping control designs are implemented as the main control algorithms.

The PWM generation task is in charge of calculating the required PWM value and transferring it to the ePWM peripheral. It takes the required current value from the algorithm execution task and converts it to the PWM value. For efficient operation, the algorithm execution task triggers this task after its completion. The ePWM peripheral generates the PWM signal with its counter compare registers. This task prepares the required PWM value in that format. Outside of these four tasks explained in this section, CPU waits in the idle stage.



(a) Algorithm execution task (b) PWM generation task

Figure 3.9. Algorithm and PWM tasks.

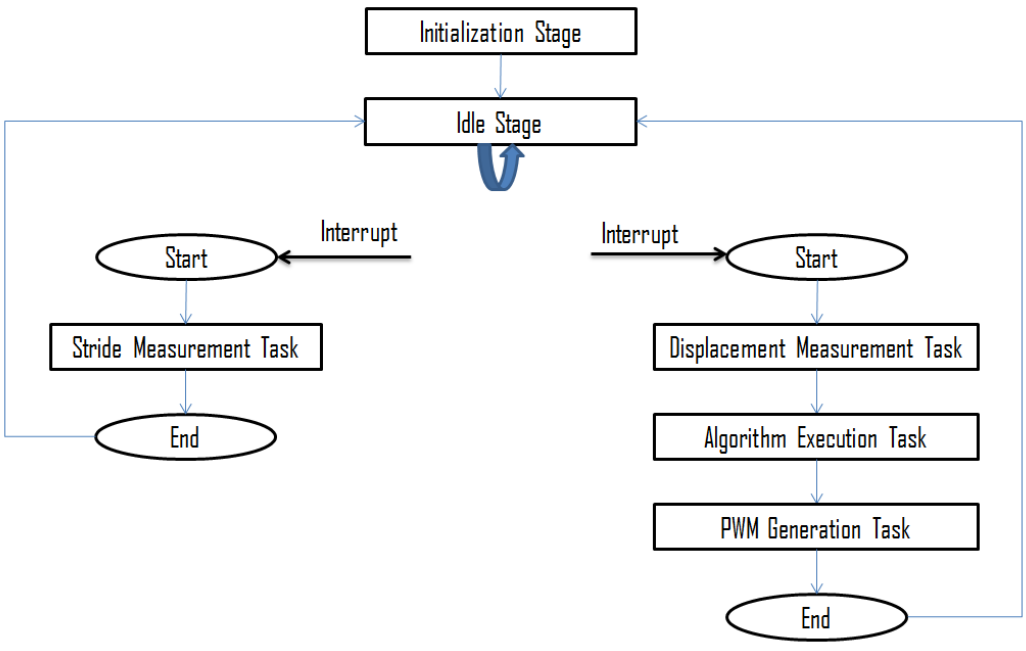


Figure 3.10. Execution flow diagram.

3.5.3. Performance Optimization

Various performance optimizations were applied to improve the execution performance of the proposed controller. These optimizations are as follows:

(i) Floating-Point Unit (FPU) was utilized.
(ii) Trigonometric Math Unit (TMU) was utilized.
(iii) Compiler optimization level was increased.
(iv) Redundant division operations were removed.
(v) Interrupt based execution was performed.
(vi) Algorithm precision was changed from double precision to single precision.

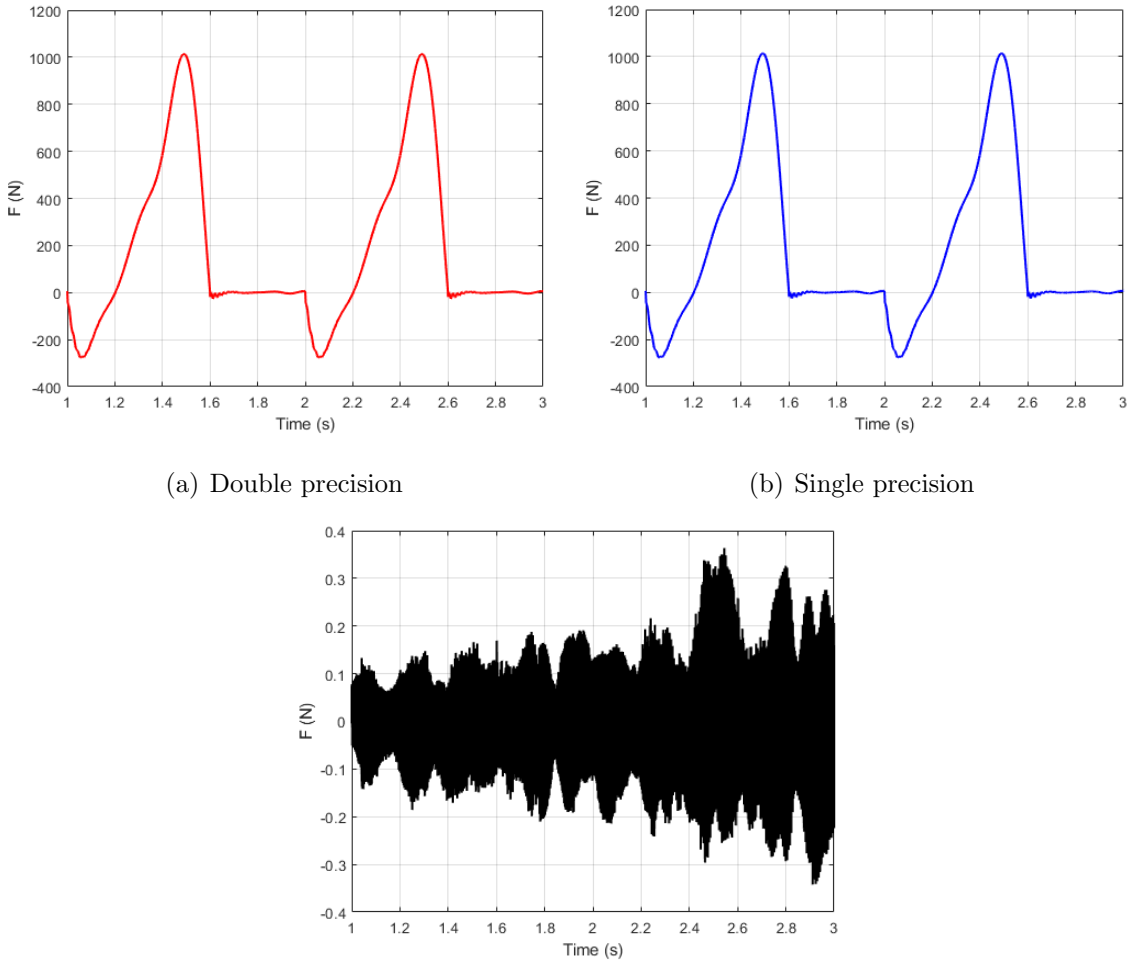
The first three items were implemented via the TMS320C28x compiler, which is an extremely powerful compiler to optimize the code. The compiler generates the FPU and TMU hardware instructions by configuring the necessary compiler settings, which were done for the proposed controller. Moreover, the compiler optimization level was increased to level four to achieve maximum optimization.

The fourth item involves the removal of redundant division operations. The CPU can generally execute multiplication operations faster than division operations. Thus, the control algorithm was searched to find operations that involved division by a constant value. Then, they were changed to multiplication operations to enhance the execution time.

The fifth item includes the interrupt based execution. Since the proposed system was designed as a bare-metal system, no operating system was used. As a result, the overhead caused by the NI Linux Real-Time OS was eliminated. Moreover, interrupt-driven scheduling was used instead of polling, which improved the efficiency of the overall system.

Lastly, the sixth item involves the algorithm precision change. The CPU of the proposed system was designed to execute single precision operations much faster.

As a result, changing from double precision to single precision caused a substantial performance improvement. Accuracy was tested by comparing the double and single precision results. Figure 3.11 shows the comparison results.



(a) Double precision

(b) Single precision

(c) Difference

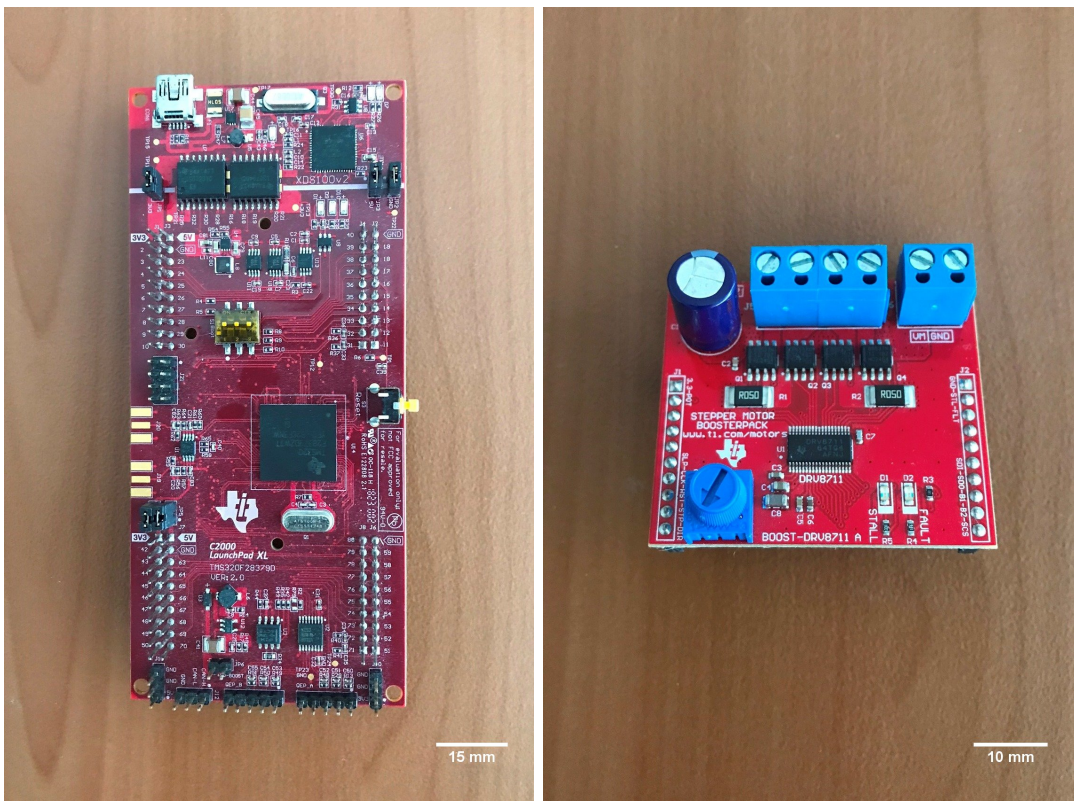
Figure 3.11. Precision comparison results.

When the results are compared, no significant accuracy loss was detected. The root-mean-square error, which gives a relatively high weight to large errors, of the results was calculated as 0.0639. As a result, single precision was chosen as the data format.

3.6. Implementation of Embedded AFO Control Unit

Implementation details for the embedded AFO control unit are given in this section. The section includes peripheral and motor driver configurations, and details of physical connection, communication protocol, and programming environment.

The embedded AFO control unit consists of three parts, which are the MCU, motor driver, and battery. The MCU and the motor driver of the system are TI's TMS320F28379D and BOOST-DRV8711, respectively. Figure 3.12 shows these two parts with a scale bar.

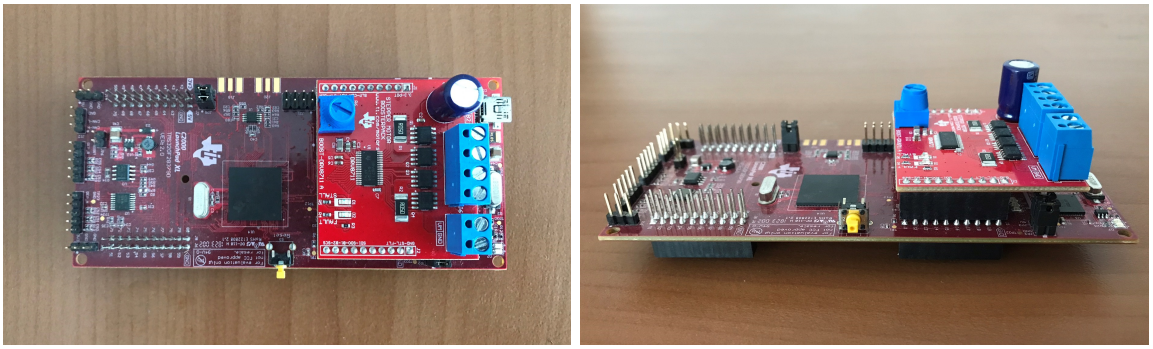


(a) TMS320F28379D (b) BOOST-DRV8711

Figure 3.12. MCU and motor driver.

The motor driver module can be placed on top of the MCU through the BoosterPack connections, which makes the total unit more compact. Thanks to this connection, the overall size of the embedded controller becomes smaller than the battery unit of the system, which is a LiPo battery.

The top and side views of the embedded controller with the motor driver connected to the MCU is shown in Figure 3.13.



(a) Top view (b) Side view

Figure 3.13. Embedded AFO control unit.

The top view of the embedded controller and the battery unit together is shown in Figure 3.14 with a scale bar. The battery unit used in the design is 3300 mAh 6 cells high discharge LiPo battery.

Figure 3.14. Top view of the controller and the battery.

The compactness of the proposed controller can be observed when the size of the battery and the controller are compared with each other. Figure 3.15 displays the battery and the controller from the side view.

Figure 3.15. Side view of the controller and the battery.

3.6.1. Peripheral Configuration

Configuration details of the peripherals that were used in the proposed system are given in this section. These peripherals consist of ePWM, eQEP, SPI, and ADC. Critical parameters and GPIO configurations of them are described as well.

The ePWM module has a time base period register, which determines the PWM frequency. This register was set as 20 kHz during the initialization stage. Furthermore, four GPIO pins were configured for this module, which are shown in Table 3.2.

Table 3.2. GPIO configuration of ePWM.

GPIO Pin	**Pin Function**
GPIO0	EPWM1A
GPIO1	EPWM1B
GPIO124	GPIO
GPIO125	GPIO

GPIO configurations of the ePWM module were done according to the pin functions from Table 3.2. GPIO0 and GPIO1 were configured for PWM output function, whereas GPIO124 and GPIO125 were configured for normal GPIO function. For physical connections, GPIO0 and GPIO1 were connected to GPIO124 and GPIO125, which were further connected to the motor driver's PWM input pins.

The eQEP module requires three GPIO pins for its operation. These pins are shown in Table 3.3. All of the three pins were configured for eQEP input functions.

Table 3.3. GPIO configuration of eQEP.

GPIO Pin	**Pin Function**
GPIO20	EQEP1A
GPIO21	EQEP1B
GPIO23	EQEP1I

The SPI module was configured with 500 kHz baud rate during the initialization stage. The module uses four GPIO pins, which are shown in Table 3.4. SPI chip select pin was emulated with a normal GPIO pin in order to satisfy the motor driver communication protocol. GPIO29 was configured for this purpose.

Table 3.4. GPIO configuration of SPI.

GPIO Pin	**Pin Function**
GPIO58	SPISIMOA
GPIO59	SPISOMIA
GPIO60	SPICLKA
GPIO29	GPIO

Lastly, ADC pins were configured. These pins were physically connected to the FSR and the potentiometer to obtain the sensor values.

3.6.2. Motor Driver Configuration

This section comprises implementation details of the motor driver, including motor control parameters and GPIO configurations. Motor control parameters can be configured through the motor driver's registers, which are accessed over SPI communication. These registers were configured according to the motor specifications of the proposed system. Some of the values were tuned based on the test results. Final motor driver configurations are given in Table 3.5.

Table 3.5. Motor driver configuration.

Register	**Field**	**Value**	**Register**	**Field**	**Value**
CTRL	ENBL	0x01	STALL	SDCNT	0x03
CTRL	RDIR	0x00	STALL	VDIV	0x03
CTRL	RSTEP	0x00	DRIVE	OCPTH	0x01
CTRL	MODE	0x00	DRIVE	OCPDEG	0x01
CTRL	EXSTALL	0x00	DRIVE	TDRIVEN	0x01
CTRL	ISGAIN	0x00	DRIVE	TDRIVEP	0x01
CTRL	DTIME	0x03	DRIVE	IDRIVEN	0x00
TORQUE	TORQUE	0x90	DRIVE	IDRIVEP	0x00
TORQUE	SIMPLTH	0x00	STATUS	OTS	0x00
OFF	TOFF	0x18	STATUS	AOCP	0x00
OFF	PWMMODE	0x01	STATUS	BOCP	0x00
BLANK	TBLANK	0x00	STATUS	APDF	0x00
BLANK	ABT	0x00	STATUS	BPDF	0x00
DECAY	TDECAY	0x00	STATUS	UVLO	0x00
DECAY	DECMOD	0x05	STATUS	STD	0x00
STALL	SDTHR	0x40	STATUS	STDLAT	0x00

GPIO configurations of the motor driver are shown in Table 3.6. The four pins from the table were configured for normal GPIO function. The corresponding motor driver functions of these pins were also presented in the table. The CPU can perform

basic functions of the driver such as the sleep and the reset operations via these pins. Moreover, status and fault reporting can be realized by using these pins as well.

Table 3.6. GPIO configuration of motor driver.

GPIO Pin	**Pin Function**
GPIO111	GPIO (nSLEEP)
GPIO22	GPIO (RESET)
GPIO61	GPIO (nSTALL)
GPIO123	GPIO (nFAULT)

3.6.3. Power Configuration

A single LiPo battery was used to power both the MCU and the motor. 3.3 V supply of the MCU was obtained from 22.2 V by using a tiny step-down regulator shown in Figure 3.16.

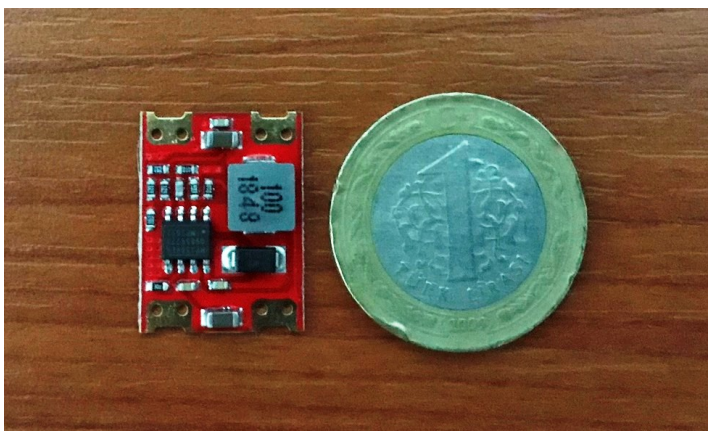


Figure 3.16. Step-down regulator.

3.7. Dual AFO Feature

Gait analysis with right and left leg is shown in Figure 3.17. Since the AFOs are generally used for providing support and assistance, they may be required to be worn on both legs. However, almost all AFO controllers are designed for single AFO control, including the NI CompactRIO based AFO controller. For dual AFO usage, two controllers and two batteries of single AFO control system need to be carried by the wearer in addition to the mechanical parts. The dual AFO feature of the proposed controller solves these shortcomings.

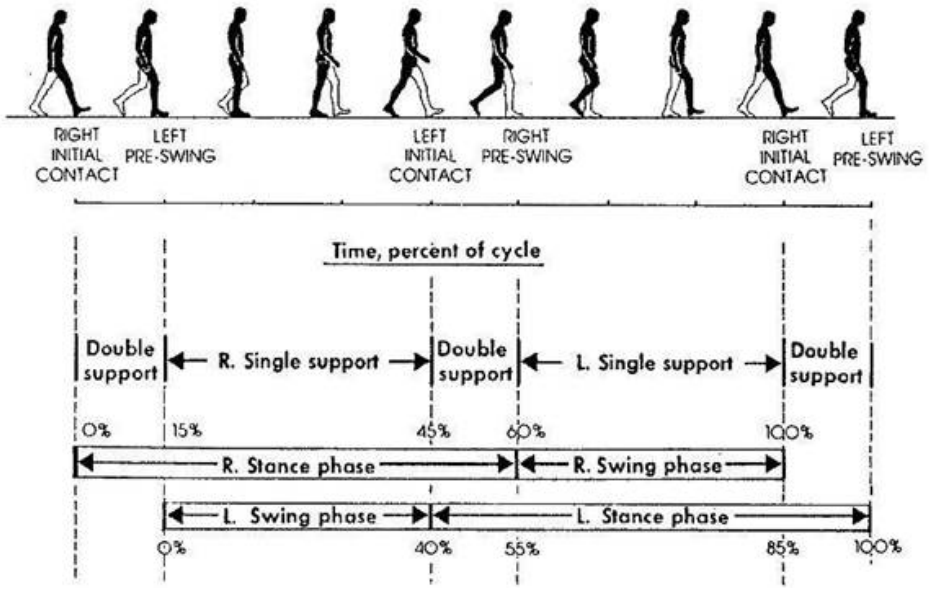
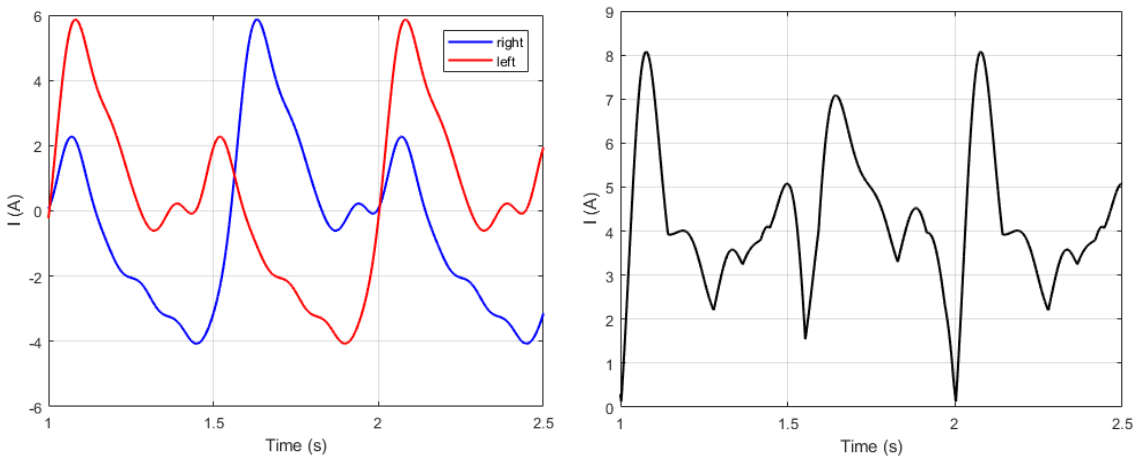


Figure 3.17. Dual gait analysis [33].

Components of the proposed AFO controller were selected to support dual AFO control. A microcontroller with a dual-core architecture was chosen for this purpose. TMS320F28379D has two 32-bit CPUs to execute control algorithms in parallel. Dual-core can allow computationally heavy control algorithms to be executed for AFO control. Furthermore, another resource required to support dual AFO control is the peripherals. This requirement was also considered during the MCU selection. TMS320F28379D has 4 ADCs, 3 eQEP modules, and 24 ePWM channels. It also has a shared RAM between the CPUs.

Another critical component to be considered for dual AFO support is the motor driver. Two motors need to be controlled by the driver. Therefore, the motor driver was selected with the capability of driving two motors. BOOST-DRV8711 can drive two brushed DC motors simultaneously. Because of this selection, no additional motor driver was required.

The final resource of the system to be evaluated for dual AFO support is the power source. The proposed system is designed based on the power requirement of a dual AFO usage. This power requirement is determined by simulating the motion shown in Figure 3.17 and obtaining the maximum required current. Figure 3.18 shows the dual AFO simulation results.



(a) Current consumption (b) Absolute sum

Figure 3.18. Dual AFO simulation with motion reference signals.

Figure 3.18(a) shows right and left reference current signals whereas Figure 3.18(b) shows the absolute sum of these signals. From the simulation results, the maximum required current is calculated as 8.07 A. According to this power requirement, the power source of the system is selected as 3300 mAh, 6 cells, 35 C LiPo battery. Battery's maximum continuous discharge rate of 35 C (115.5 A) easily satisfies the maximum required current.

Dual AFO support was added by extending the single AFO design. Since the system components are selected to achieve parallelism, each CPU and each required peripheral are dedicated to a different leg. For instance, the CPU1 was responsible for

the right AFO, whereas the CPU2 was responsible for the left AFO. Each CPU utilized different ADCs, eQEP modules, and ePWM channels. The block diagram of the dual AFO system is shown in Figure 3.19. The right AFO runs MOTOR1, whereas the left AFO runs MOTOR2.

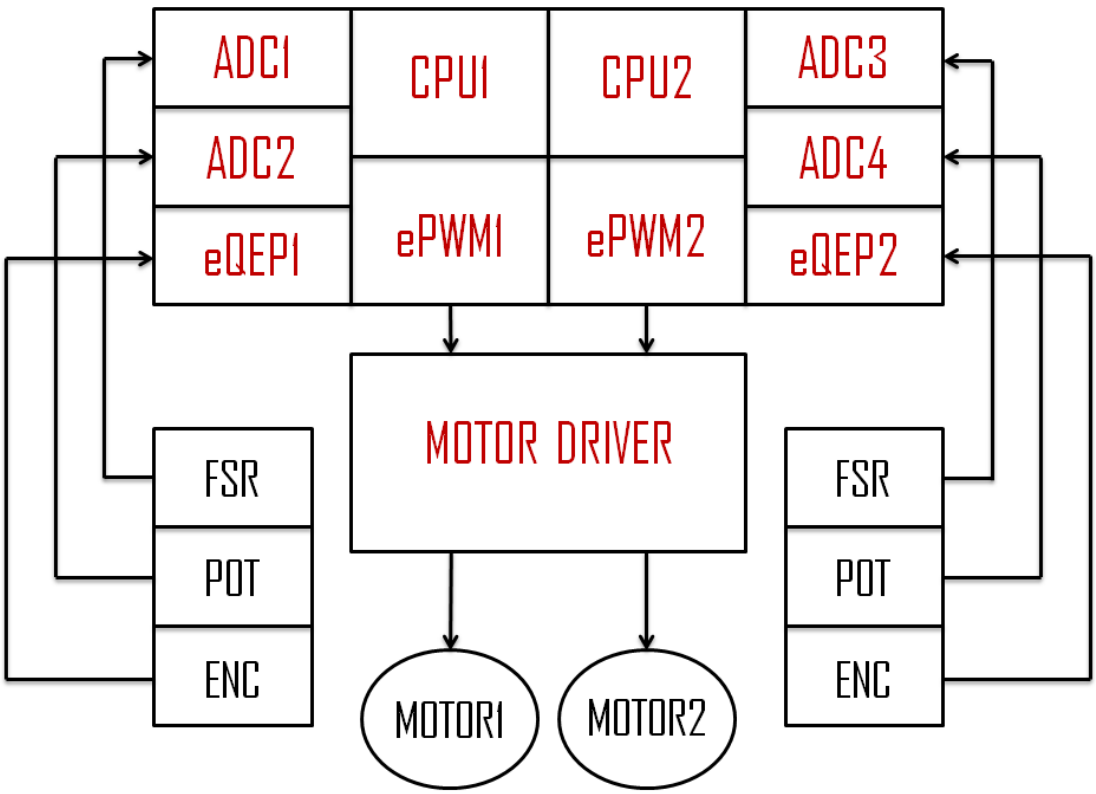


Figure 3.19. Dual AFO block diagram.

4. EXPERIMENTAL METHODS

In order to validate and evaluate the embedded system based controller, various tests were prepared and conducted. In this chapter, experimental methods of these tests are presented.

Experimental tests were divided into three categories, which are validation tests, performance tests, and power consumption tests. Validation tests were conducted to obtain the heel position values during a gait cycle, whereas performance tests were conducted to obtain the maximum duration of a single execution cycle that starts with the displacement measurement task and ends with the PWM generation task. On the other hand, power consumption tests were conducted to obtain the current consumption values during a gait cycle. Finally, the dual AFO counterparts of these tests were performed.

All of these tests were performed with the same test procedure. A subject of 70 kg wore the AFO prototype on the right foot and walked for eight steps on a wooden floor. In each test, data acquisition starts after the second heel strike, and two consecutive gait cycles of data are obtained.

For each test category, two different stride times and two different control algorithms were tested. Stride times were selected as 1 s and 0.75 s. On the other hand, control algorithms were selected as PID control and adaptive backstepping control. The details of the adaptive backstepping control algorithm are explained in [13, 34]. The stride time and the control algorithm of the controller were set before starting the test, and the user walked according to the stride time.

Data acquisition is made by the MCU, and the related data of the test are saved to a global buffer not to disrupt the timing of the execution. After the test is completed, the data from the buffer is extracted via the debugger from Code Composer Studio. Since the size of the MCU's RAM is limited, the size of the buffer is adjusted to hold

10000 samples of data. Depending on the execution rate of the algorithm, multiple data acquisition is required to complete the two gait cycles of data. In each run, one portion of the gait cycle is saved, and later the extracted data are joined together in MATLAB. The synchronization of data is done according to the heel strike. Furthermore, if the signals such as PWM output and SPI communication signals are required to be probed, a USB logic analyzer of 24 MHz and 8 channels, shown in Figure 4.2, is utilized.

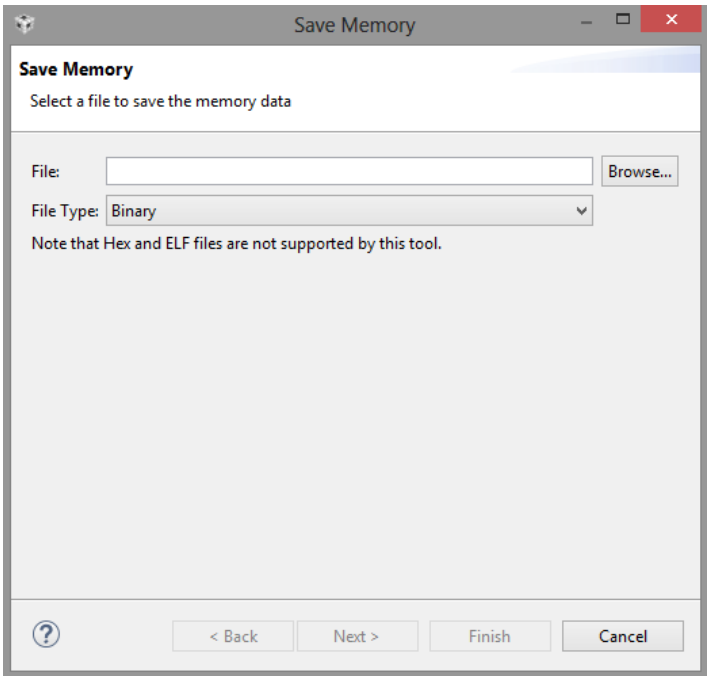


Figure 4.1. Save memory from Code Composer Studio.

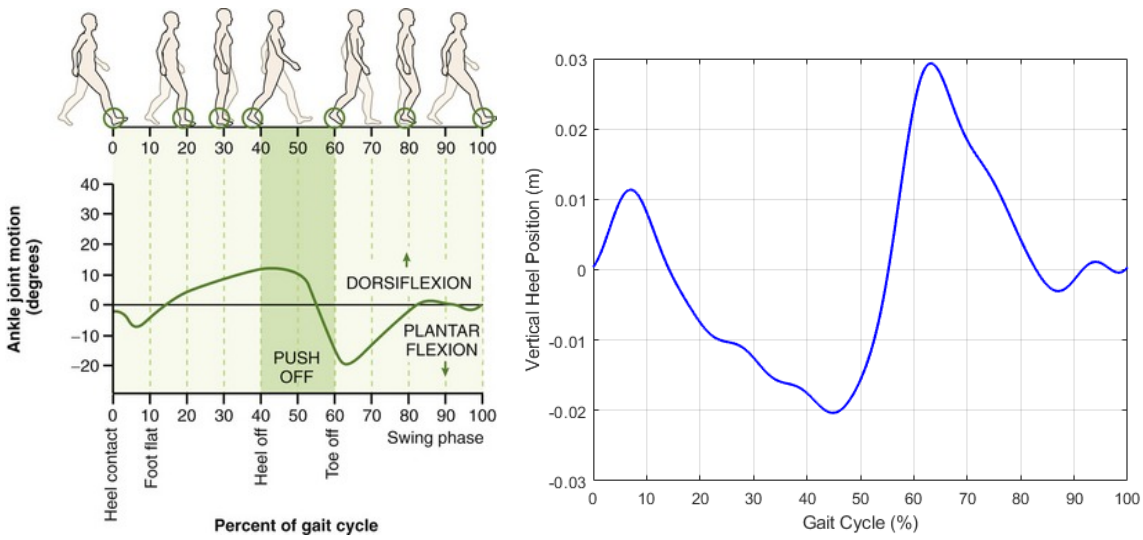
Figure 4.2. Logic analyzer.

5. EXPERIMENTAL RESULTS AND DISCUSSION

In this chapter, experimental results and discussion of them are presented. The experimental results are divided into four parts, which are validation results, performance results, power consumption results, and dual AFO results. At the end of the chapter, discussion and comparison of the results for both single and dual AFO are presented.

5.1. Validation Results

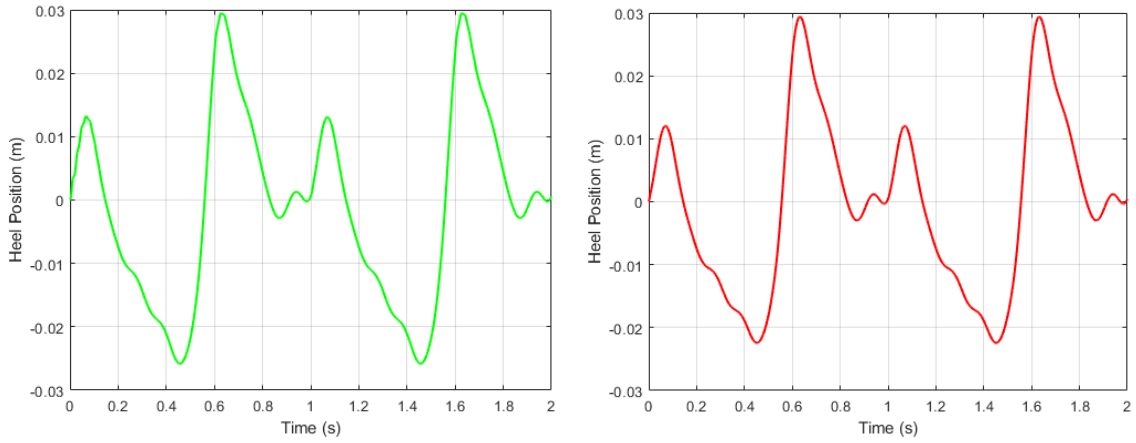
In order to validate the functionality of the embedded controller, the reference position signal from [13] is used. This reference signal was generated from the ankle joint motion of healthy individuals during a gait cycle. Figure 5.1(a) shows this ankle joint motion and Figure 5.1(b) shows the reference position signal from [13]. Experimental position data were compared with this reference signal to validate the functionality of the system.



(a) Ankle joint motion (b) Reference position signal

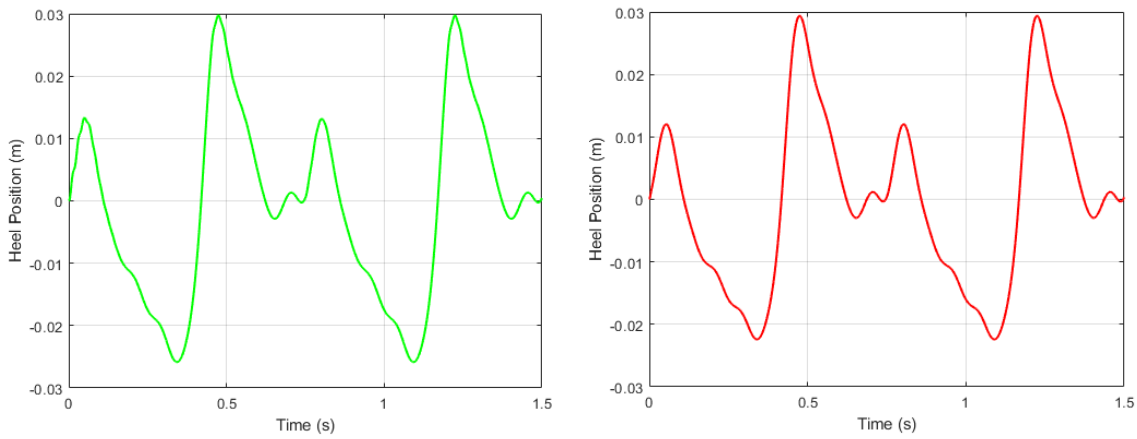
Figure 5.1. Ankle joint motion [21] and reference position signal [13].

Validation tests were conducted for two different stride times and two different control algorithms, as explained in the experimental methods. Figure 5.2 shows the results for the 1 s stride time, whereas Figure 5.3 shows the results for the 0.75 s stride time.



(a) PID (b) Adaptive

Figure 5.2. Heel position results (1 s stride time).



(a) PID (b) Adaptive

Figure 5.3. Heel position results (0.75 s stride time).

Results from Figure 5.2 and Figure 5.3 were compared with the reference signal from Figure 5.1(b) to validate the proposed embedded controller. For each test, root-mean-square error (RMSE) of the tracking during one gait cycle was calculated and shown in Table 5.1.

Table 5.1. RMSE measurements for validation.

Stride Time	**Algorithm**	**RMSE (mm)**
1 s	PID	2.36
1 s	Adaptive	0.92
0.75 s	PID	2.41
0.75 s	Adaptive	1.00

5.2. Performance Results

The performance results of the proposed controller were obtained by measuring the execution time. CPU hardware timer was utilized to get precise measurements. CPU timer was started during the initialization stage, and it runs off of the 200 MHz system clock. It uses a 32-bit counter register and decrements the counter once every cycle.

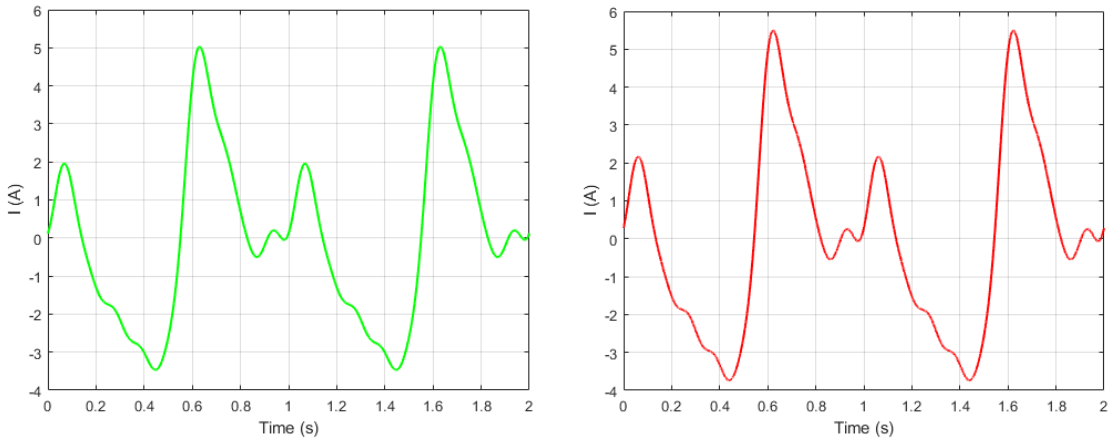
The maximum and minimum durations of one complete execution were measured in terms of clock cycles during the tests. Duration values in terms of clock cycles can be converted to time values in seconds by dividing them to the system clock frequency value. For max duration, the PID algorithm took 22.91 μs to finish, whereas the adaptive algorithm took 55.07 μs. Furthermore, the stride time did not affect the performance results. Performance results are given in Table 5.2.

Table 5.2. Execution performance results.

Stride Time	**Algorithm**	**Min Duration**	**Max Duration**
1 s	PID	20.96 μs	22.89 μs
1 s	Adaptive	55.03 μs	55.06 μs
0.75 s	PID	20.98 μs	22.91 μs
0.75 s	Adaptive	55.04 μs	55.07 μs

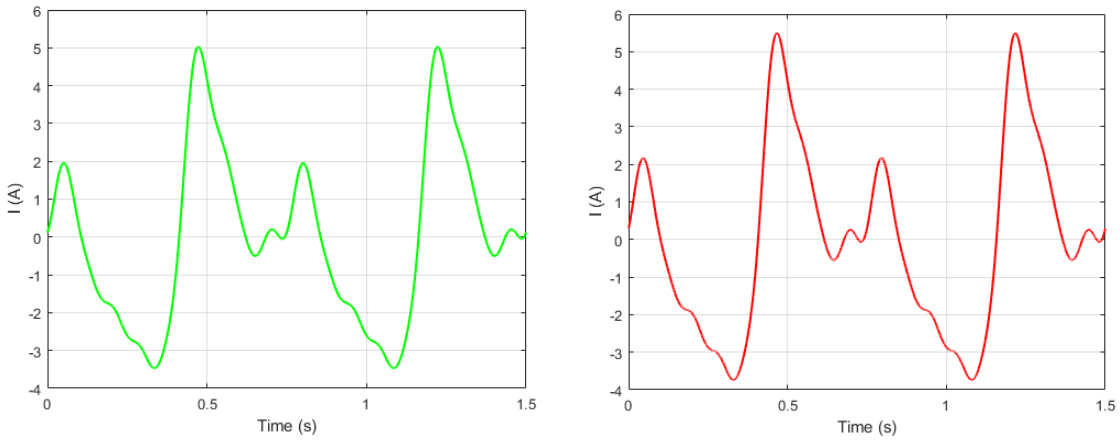
5.3. Power Consumption Results

The algorithm execution task of the MCU calculates the required force value, which is converted to the required torque for the motor by using Equation 2.1. Finally, the required current value is obtained by dividing this torque value to the torque constant of the motor. During tests, current values were saved to observe the power consumption. Figure 5.4 and Figure 5.5 show the actuator current consumption plots for each algorithm and stride time.



(a) PID (b) Adaptive

Figure 5.4. Current results (1 s stride time).



(a) PID (b) Adaptive

Figure 5.5. Current results (0.75 s stride time).

Since the voltage of the battery is a fixed value, average power consumption during a gait cycle can be calculated from the current consumption plots. Table 5.3 shows the actuator power consumption results per a gait cycle.

Table 5.3. Actuator power consumption results.

Stride Time	**Algorithm**	**Average Current**	**Average Power**
1 s	PID	1.826 A	40.537 W
1 s	Adaptive	1.975 A	43.845 W
0.75 s	PID	1.826 A	40.537 W
0.75 s	Adaptive	1.975 A	43.845 W

High discharge LiPo battery of 3300 mAh 6 cells was used as the battery of the system. For 1 s stride time, the AFO system with adaptive control can actively operate for 1.67 hours, whereas the AFO system with PID control can actively operate for 1.80 hours. For 0.75 s stride time, these values are calculated as 1.25 hours and 1.36 hours for adaptive and PID control, respectively.

The power consumption results presented before are for the DC motor only. In addition to that, the embedded controller consumes power as well. There are two modes of the controller, which are idle and operational. The controller consumes approximately 105 mA of current (0.189 W of power) in idle mode, whereas it consumes about 325 mA of current (0.585 W of power) in operational mode. There is more than 70 times difference between the power consumption of the controller and the DC motor.

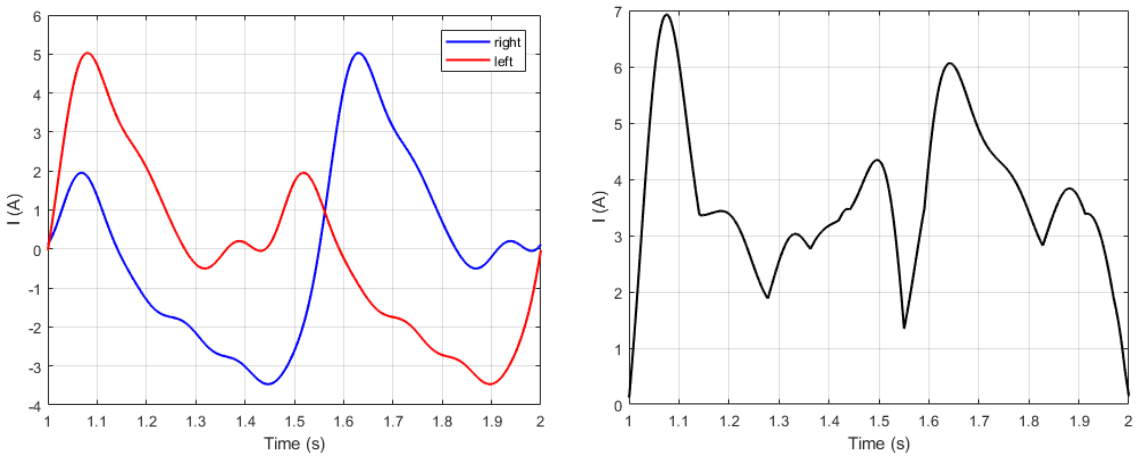
5.4. Dual AFO Results

In this section, the results for the dual AFO feature are presented. Since there was one mechanical unit of AFO prototype available at the time of the experiments, the results were obtained by testing consecutively with the single unit.

The same three test categories from the single AFO tests are performed, which are the validation, performance, and power consumption. Dual AFO operation is performed completely in parallel with the additional CPU and peripherals. Therefore, validation and performance results from the single AFO tests are valid for the dual AFO tests as well. The tests and results are identical in these two categories.

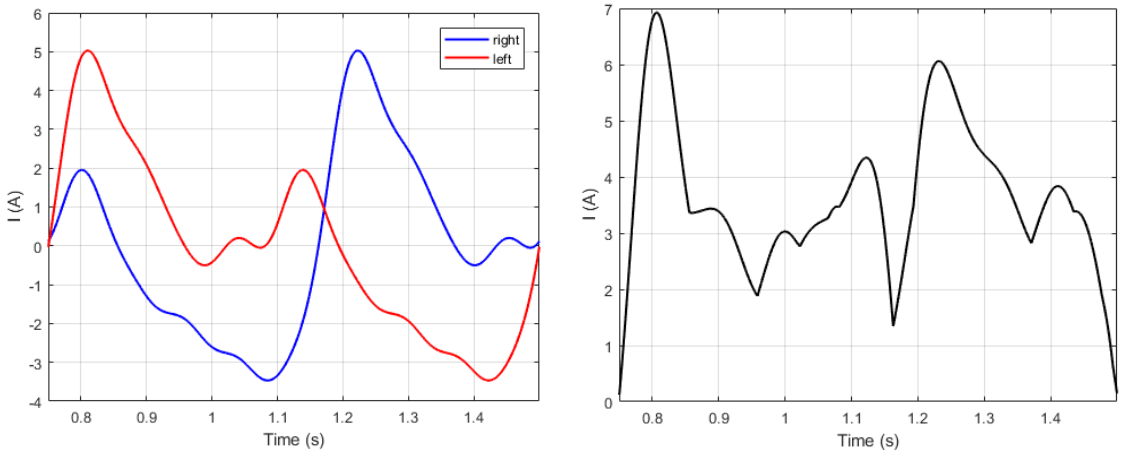
For power consumption, the dual AFO results are different from the single AFO results as expected, since two motors consume power simultaneously during the motion. The current consumption values of the system were saved for each AFO consecutively. Later, they are added together with a motion delay in between to calculate the total current consumption of the dual AFO operation. Finally, power consumption results are calculated from the current consumption results.

Figure 5.6 and Figure 5.7 shows the dual AFO results of the PID control algorithm in 1 s and 0.75 s stride time values, respectively. On the other hand, Figure 5.8 and Figure 5.9 shows the dual AFO results of the adaptive control algorithm in 1 s and 0.75 s stride time values, respectively. In each figure, dual AFO operation and absolute sum of the current signals are displayed.



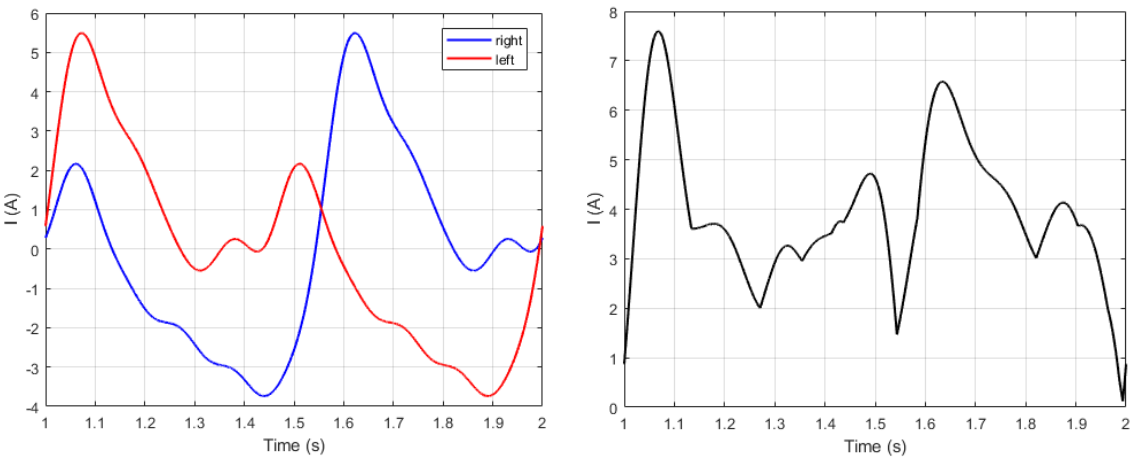
(a) Current consumption (b) Absolute sum

Figure 5.6. Dual AFO current results (PID & 1 s stride time).



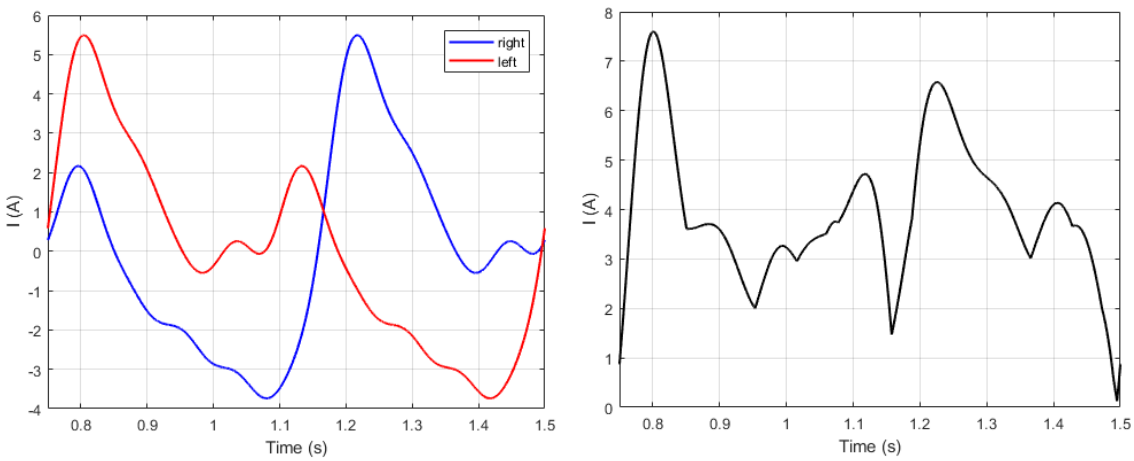
(a) Current consumption (b) Absolute sum

Figure 5.7. Dual AFO current results (PID & 0.75 s stride time).



(a) Current consumption (b) Absolute sum

Figure 5.8. Dual AFO current results (Adaptive & 1 s stride time).



(a) Current consumption (b) Absolute sum

Figure 5.9. Dual AFO current results (Adaptive & 0.75 s stride time).

Table 5.4 shows the dual AFO power consumption results. These results are calculated from current consumption plots since the battery of the system has a fixed voltage. For 1 s stride time, the dual AFO system with adaptive control can actively operate for 0.84 hours, whereas the dual AFO system with PID control can actively operate for 0.90 hours. For 0.75 s stride time, these values are calculated as 0.63 hours and 0.68 hours for adaptive and PID control, respectively.

Table 5.4. Dual AFO power consumption results.

Stride Time	**Algorithm**	**Average Current**	**Average Power**
1 s	PID	3.652 A	81.074 W
1 s	Adaptive	3.949 A	87.668 W
0.75 s	PID	3.652 A	81.074 W
0.75 s	Adaptive	3.949 A	87.668 W

If the battery of the system is changed from 3300 mAh LiPo to 14000 mAh LiPo, the overall system can run with 4.24 times more operation time. For instance, the dual AFO system with PID control powered by the new battery can actively operate for 3.82 hours for 1 s stride time. However, the weight of the battery will increase from 530 g to 1820 g, while the dimension of the battery will increase from 137 x 45 x 45 mm to 183 x 77 x 66 mm. Also, the price of the battery will increase by about 3 times.

5.5. Discussion

In this section, the discussion about the experimental results of the proposed system is given. Five main topics, including validation results, performance results, power consumption results, dual AFO results and physical characteristics, are discussed.

The functionality of the proposed system is tested by comparing the experimental results to a reference signal generated from the ankle motion of healthy individuals. The reference position signal is shown in Figure 5.1(b), whereas the position results of the proposed system are shown in Figure 5.2 and Figure 5.3. Two different stride

time values and control algorithms are tested. Table 5.1 shows the comparison of these results with the reference signal in terms of RMSE. From the results, no significant error is measured. Moreover, when these measurements are compared with the measurements of the NI CompactRIO based AFO system [13], the results are within approximately 2.5% of each other. As a result, validation tests show that the proposed system functions accurately.

The execution performance of the proposed system is measured with a hardware timer of the MCU. Two stride time values and control algorithms are tested, and the performance results are given in Table 5.2. The results show that stride time does not affect the execution performance. Also, the proposed system can achieve an execution rate of 42 kHz and 17 kHz with PID and adaptive control algorithms, respectively. More than 3.4 times better execution rate, depending on the algorithm, is achieved when compared to the NI CompactRIO based controllers. Moreover, this is accomplished with a 200 MHz MCU compared to the 1.33 GHz CPU of the NI CompactRIO device.

The current consumption plots for the two different stride time values and control algorithms are given in Figure 5.4 and Figure 5.5. Power consumption results that are calculated from these plots are given in Table 5.3. Since the main power consumption is done by the DC motor, the same consumption results are achieved with the NI CompactRIO based system. That being said, the critical improvement is realized for the power supply of the controller device itself. The MCU of the proposed system uses a 3.3 V supply while the NI CompactRIO uses a 24 V supply. This difference makes the controller part of the proposed AFO system significantly more power-efficient.

The dual AFO support is a new feature developed to extend and enhance the AFO use cases. Since the dual AFO operation is designed to be executed entirely in parallel, the discussions about the single AFO validation and performance results apply to the dual AFO as well. On the other hand, power consumption results of the dual AFO system are given in Table 5.4. The results show that the power consumption of the system is doubled as expected. Conversely, the peak current requirement was not

doubled because of the delay between the motion of the legs. This situation helps to maintain the temperature of the power MOSFETs, which extends the lifetime of the overall device. The LiPo battery satisfies the current requirements of the dual AFO operation.

The proposed system uses a single LiPo battery as the power source, whereas the NI CompactRIO based system uses two 24 V tethered power sources, one for the controller and one for the DC motor. This design brings both mobility and flexibility to the system. Furthermore, the weight and size of the controller are significantly reduced as well. NI CompactRIO device itself without the I/O modules weighs about 2250 g whereas the embedded controller with the motor driver weighs about 60 g.

The proposed embedded AFO controller weighs about 590 g, which consists of a 530 g battery and 60 g embedded controller. The proposed controller is lighter than the hip exoskeleton controller from [19], which weighs 853 g including a 300 g battery. It is also lighter than the KAFO controller from [17], which weighs 700 g with a battery. The proposed controller and the mechanical AFO prototype weigh about 2.4 kg altogether, which is lighter than the hydraulic AFO from [18] that weighs 3.5 kg.

6. CONCLUSION

In this thesis, an embedded system implementation for real-time control of AFO is proposed. AFOs, which are utilized to control and assist the position and the motion of the ankle, are widely used for patient rehabilitation and assistance. In healthcare, the mobility and compactness of the device directly affect the patient. Considering these trends, the design requirements for a compact and mobile system are determined.

The component selection of the embedded system based AFO controller is made considering the requirements of the system such as speed, power, memory, and peripherals. An embedded system consisting of an MCU and a motor driver is chosen as the controller of the AFO. Both components are selected to be capable of supporting dual AFO operation. The selected MCU is a world-class microcontroller for leading floating-point performance and analog integration to control applications. For the final component, a LiPo battery is selected as the power source to bring mobility to the overall system.

The implementation of control is realized on a bare-metal system to achieve maximum performance. The control flow of the execution is optimized to benefit from hardware accelerators and peripherals. Finally, dual AFO support is added to the design as an innovative feature.

Experimental results are obtained for validating the functionality and measuring the performance and power consumption of the system. Validation results suggest that the proposed system functions correctly, including the dual AFO operation. Performance results show a noteworthy improvement in execution rates. The last but not the least, system mobility is achieved by using a powerful and lightweight battery.

REFERENCES

1. Mirelman, A., S. Shema, I. Maidan and J. M. Hausdorff, "Gait", *Handbook of Clinical Neurology*, Vol. 159, chap. 7, Elsevier Science Inc., 2018.

2. Alam, M., I. A. Choudhury and A. B. Mamat, "Mechanism and Design Analysis of Articulated Ankle Foot Orthoses for Drop-Foot", *The Scientific World Journal*, Vol. 2014, 2014.

3. Shorter, K. A., J. Xia, E. T. Hsiao-Wecksler, W. K. Durfee and G. F. Kogler, "Technologies for Powered Ankle-Foot Orthotic Systems: Possibilities and Challenges", *IEEE/ASME Transactions on Mechatronics*, Vol. 18, No. 1, pp. 337–347, Feb 2013.

4. Giladi, N., B. R. Bloem and J. M.Hausdorff, "Gait Disturbances and Falls", *Neurology and Clinical Neuroscience*, chap. 36, Japanese Society of Neurology and John Wiley & Sons Australia, Ltd, 2007.

5. Lee, H., W. Kim, J. Han and C. Han, "The technical trend of the exoskeleton robot system for human power assistance", *International Journal of Precision Engineering and Manufacturing*, Vol. 13, No. 8, pp. 1491–1497, Aug 2012.

6. Becker Orthopedic, *Online Catalog: Custom Fabrication AFOs [Digital Image]*, `https://www.beckerorthopedic.com`, accessed in June 2019.

7. Star Medical, *Online Catalog: Posterior Leaf Spring AFO [Digital Image]*, `http://www.starmedicalllc.com`, accessed in June 2019.

8. Blaya, J. A. and H. Herr, "Adaptive control of a variable-impedance ankle-foot orthosis to assist drop-foot gait", *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 12, No. 1, pp. 24–31, March 2004.

9. Sawicki, G. S. and D. P. Ferris, "A pneumatically powered knee-ankle-foot orthosis (KAFO) with myoelectric activation and inhibition", *Journal of NeuroEngineering and Rehabilitation*, Vol. 6, No. 1, June 2009.

10. Ferris, D., J. M Czerniecki and B. Hannaford, "An Ankle-Foot Orthosis Powered by Artificial Pneumatic Muscles", *Journal of Applied Biomechanics*, Vol. 21, pp. 189–197, June 2005.

11. Veneva, I., "Intelligent Device for Control of Active Ankle-Foot Orthosis", *Proceedings of the 7th IASTED International Conference on Biomedical Engineering, BioMED 2010*, Vol. 2, pp. 100–105, Jan 2010.

12. Kanthi, M., I. S. V. Karteek, H. S. Mruthyunjaya and V. I. George, "Real-time control of active ankle foot orthosis using LabVIEW and Compact-RIO", *2012 International Conference on Biomedical Engineering (ICoBE)*, pp. 296–299, Feb 2012.

13. Kırtaş, O., *Design and Control of an Active Ankle-Foot Orthosis*, Master's Thesis, Boğaziçi University, 2018.

14. Sanz-Morère, C. B., M. Fantozzi, A. Parri, F. Giovacchini, A. Baldoni, S. Crea and N. Vitiello, "A Bioinspired Control Strategy for the CYBERLEGs Knee-Ankle-Foot Orthosis: Feasibility Study with Lower-Limb Amputees", *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, pp. 503–508, Aug 2018.

15. Lv, G., Hanqi Zhu, T. Elery, Luwei Li and R. D. Gregg, "Experimental implementation of underactuated potential energy shaping on a powered ankle-foot orthosis", *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3493–3500, May 2016.

16. Zhang, Y., R. J. Kleinmann, K. J. Nolan and D. Zanotto, "Design and Evaluation of an Active/Semiactive Ankle-Foot Orthosis for Gait Training", *2018 7th IEEE*

International Conference on Biomedical Robotics and Biomechatronics (Biorob), pp. 544–549, Aug 2018.

17. Arazpour, M., F. Ahmadi, M. Ahmadi Bani, S. W. Hutchins, M. Bahramizadeh, F. Tabatabai Ghomshe and R. V. Kashani, "Gait evaluation of new powered knee–ankle–foot orthosis in able-bodied persons: a pilot study", *Prosthetics and orthotics international*, Vol. 38, No. 1, pp. 39–45, 2014.

18. Neubauer, B. C., J. Nath and W. K. Durfee, "Design of a portable hydraulic ankle-foot orthosis", *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1182–1185, Aug 2014.

19. Ishmael, M. K., M. Tran and T. Lenzi, "ExoProsthetics: Assisting Above-Knee Amputees with a Lightweight Powered Hip Exoskeleton", *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, pp. 925–930, June 2019.

20. Zhu, H., C. Nesler, N. Divekar, M. T. Ahmad and R. D. Gregg, "Design and Validation of a Partial-Assist Knee Orthosis with Compact, Backdrivable Actuation", *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, pp. 917–924, June 2019.

21. Richards, J., *Biomechanics in Clinic and Research*, Elsevier Health Sciences, Jan 2008.

22. Maxon Motor AG, *Online Catalog: Brushed DC Motors [Digital Image]*, `https://www.maxonmotor.com`, accessed in June 2019.

23. National Instruments Tutorials, *Learn RIO*, Tech. rep., National Instruments, 2019.

24. National Instruments White Papers, *The LabVIEW RIO Architecture: A Foundation for Innovation*, Tech. rep., National Instruments, March 2019.

25. National Instruments, *Online Catalog: PS-15 Power Supply [Digital Image]*, `https://www.ni.com`, accessed in June 2019.

26. Texas Instruments Technical Documents, *TMS320F2837xD Dual-Core Microcontrollers Datasheet*, Tech. rep., Texas Instruments, November 2018.

27. Texas Instruments Technical Documents, *Accelerators: Enhancing the Capabilities of the C2000 MCU Family Technical Brief*, Tech. rep., Texas Instruments, November 2016.

28. Texas Instruments Technical Documents, *TMS320F2837xD Dual-Core Microcontrollers Technical Reference Manual*, Tech. rep., Texas Instruments, September 2017.

29. Texas Instruments Technical Documents, *The TMS320F2837xD Architecture: Achieving a New Level of High Performance*, Tech. rep., Texas Instruments, February 2016.

30. Texas Instruments Technical Documents, *BOOST-DRV8711 User's Guide*, Tech. rep., Texas Instruments, May 2014.

31. Texas Instruments Technical Documents, *DRV8711 Motor Controller Circuit Datasheet*, Tech. rep., Texas Instruments, May 2017.

32. Texas Instruments Technical Documents, *CSD88537ND Power MOSFET Circuit Datasheet*, Tech. rep., Texas Instruments, August 2014.

33. O'Connor, R. and R. Pankajam, *Clinical Gait Analysis in Rehabilitation Medicine [University of Leeds]*, `https://minerva.leeds.ac.uk`, accessed in July 2019.

34. Savas, Y., O. Kirtas, H. Bastürk and E. Samur, "A backstepping control design for an active ankle-foot orthosis", *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 262–267, Dec 2017.

APPENDIX A: DATASHEETS

TEXAS INSTRUMENTS
TMS320F28379D, TMS320F28377D
TMS320F28376D, TMS320F28375D, TMS320F28374D
SPRS880J – DECEMBER 2013 – REVISED MAY 2018

TMS320F2837xD Dual-Core Delfino™ Microcontrollers

1 Device Overview

1.1 Features

- Dual-Core Architecture
  - Two TMS320C28x 32-Bit CPUs
  - 200 MHz
  - IEEE 754 Single-Precision Floating-Point Unit (FPU)
  - Trigonometric Math Unit (TMU)
  - Viterbi/Complex Math Unit (VCU-II)
- Two Programmable Control Law Accelerators (CLAs)
  - 200 MHz
  - IEEE 754 Single-Precision Floating-Point Instructions
  - Executes Code Independently of Main CPU
- On-Chip Memory
  - 512KB (256KW) or 1MB (512KW) of Flash (ECC-Protected)
  - 172KB (86KW) or 204KB (102KW) of RAM (ECC-Protected or Parity-Protected)
  - Dual-Zone Security Supporting Third-Party Development
  - Unique Identification Number
- Clock and System Control
  - Two Internal Zero-Pin 10-MHz Oscillators
  - On-Chip Crystal Oscillator
  - Windowed Watchdog Timer Module
  - Missing Clock Detection Circuitry
- 1.2-V Core, 3.3-V I/O Design
- System Peripherals
  - Two External Memory Interfaces (EMIFs) With ASRAM and SDRAM Support
  - Dual 6-Channel Direct Memory Access (DMA) Controllers
  - Up to 169 Individually Programmable, Multiplexed General-Purpose Input/Output (GPIO) Pins With Input Filtering
  - Expanded Peripheral Interrupt Controller (ePIE)
  - Multiple Low-Power Mode (LPM) Support With External Wakeup
- Communications Peripherals
  - USB 2.0 (MAC + PHY)
  - Support for 12-Pin 3.3 V-Compatible Universal Parallel Port (uPP) Interface
  - Two Controller Area Network (CAN) Modules (Pin-Bootable)
  - Three High-Speed (up to 50-MHz) SPI Ports (Pin-Bootable)
  - Two Multichannel Buffered Serial Ports (McBSPs)
  - Four Serial Communications Interfaces (SCI/UART) (Pin-Bootable)
  - Two I2C Interfaces (Pin-Bootable)
- Analog Subsystem
  - Up to Four Analog-to-Digital Converters (ADCs)
    - 16-Bit Mode
      - 1.1 MSPS Each (up to 4.4-MSPS System Throughput)
      - Differential Inputs
      - Up to 12 External Channels
    - 12-Bit Mode
      - 3.5 MSPS Each (up to 14-MSPS System Throughput)
      - Single-Ended Inputs
      - Up to 24 External Channels
    - Single Sample-and-Hold (S/H) on Each ADC
    - Hardware-Integrated Post-Processing of ADC Conversions
      - Saturating Offset Calibration
      - Error From Setpoint Calculation
      - High, Low, and Zero-Crossing Compare, With Interrupt Capability
      - Trigger-to-Sample Delay Capture
  - Eight Windowed Comparators With 12-Bit Digital-to-Analog Converter (DAC) References
  - Three 12-Bit Buffered DAC Outputs
- Enhanced Control Peripherals
  - 24 Pulse Width Modulator (PWM) Channels With Enhanced Features
  - 16 High-Resolution Pulse Width Modulator (HRPWM) Channels
    - High Resolution on Both A and B Channels of 8 PWM Modules
    - Dead-Band Support (on Both Standard and High Resolution)
  - Six Enhanced Capture (eCAP) Modules
  - Three Enhanced Quadrature Encoder Pulse (eQEP) Modules
  - Eight Sigma-Delta Filter Module (SDFM) Input Channels, 2 Parallel Filters per Channel
    - Standard SDFM Data Filtering
    - Comparator Filter for Fast Action for Out of Range

Figure A.1. TMS320F2837xD device overview.

7.2 Functional Block Diagram

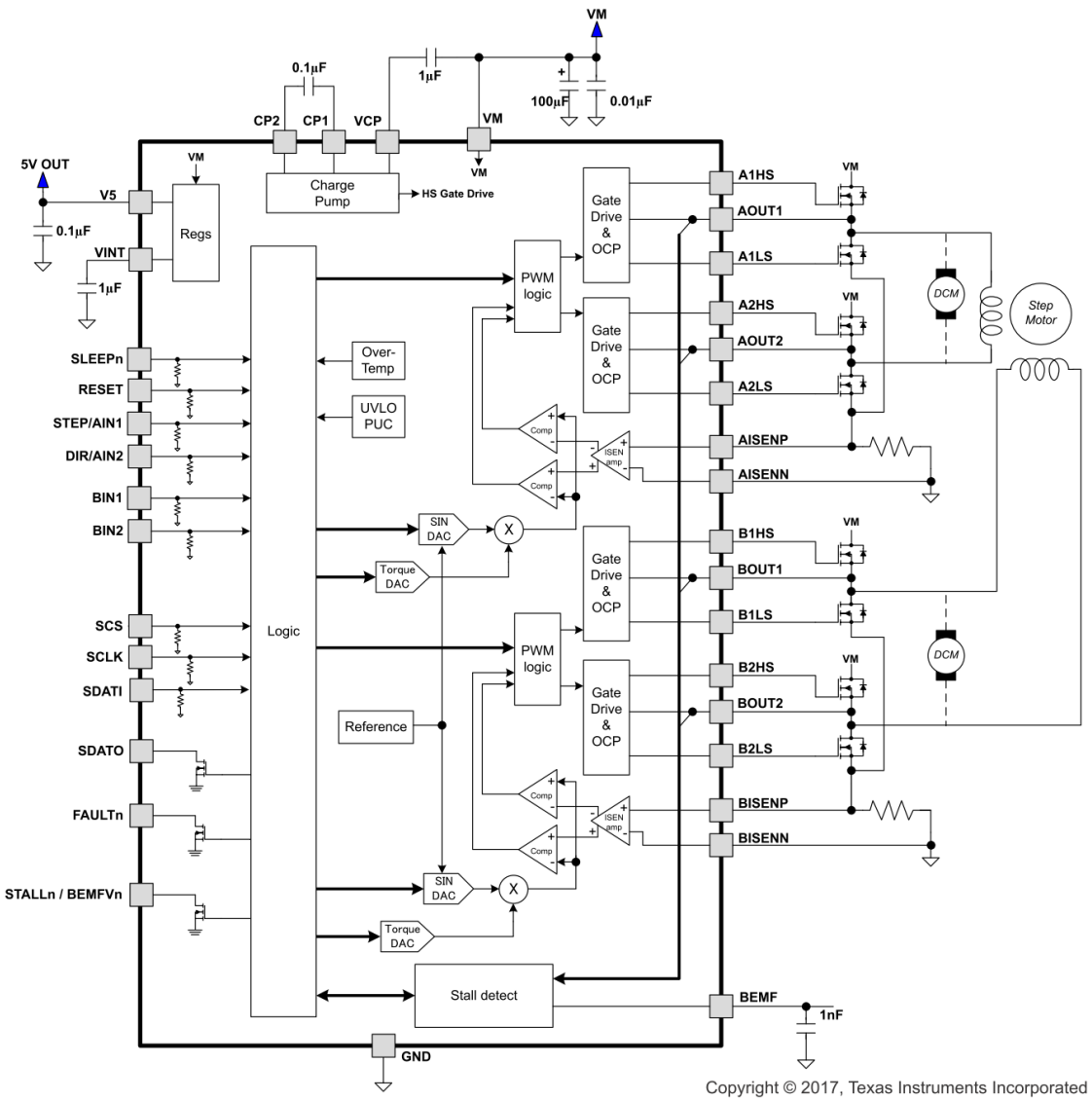


Figure A.2. DRV8711 functional block diagram.