A DATA REQUISITION TREATMENT INSTRUMENT FOR CLINICAL QUANTIFIABLE SOFT TISSUE MANIPULATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Abhinaba Bhattacharjee

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2019

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. Stanley Y.P. Chien, Chair

Department of Electrical and Computer Engineering.

Dr. Brian King

Department of Electrical and Computer Engineering.

Dr. Sohel Anwar

Department of Mechanical and Energy Engineering.

Dr. Terry Loghmani

Department of Physical Therapy.

Approved by:

Dr. Brian King

Head of Graduate Program

This thesis is dedicated to my beloved Parents for their unparallel support in all my past and present endeavors.

ACKNOWLEDGMENTS

I feel privileged to be a part of the technical team and honored to contribute to the growth and potential prospects of the development of this technology. As a graduate research assistant, I would like to thank the graduate chair Dr. Brian King and the department of Electrical and Computer Engineering to provide me this opportunity to work for this research. I would like to share my gratitude to Dr. Stanley Y.P. Chien for his special support, encouragement and parental guidance throughout the overall research. Huge appreciations to Dr. Sohel Anwar for motivating me and supporting my ideologies related to this research. A special thanks to Dr. Terry Loghmani, for her significant feedbacks in the research and promoting me as an engineer of this novel project in different health presentations and conferences.

I would also like to express my appreciation to Mr. Barry Davignon (Mechanical Engineer at Rose Hulman Ventures) for his co-ordination with the research team to yield impressive inputs in designing the medical device prototypes. Additionally, many thanks to Josh Roy, Alanna Fennimore, Carly Francis and all other undergraduate research assistants from the department of Physical Therapy for their efforts, feedback and contribution in elevating this research from pre-clinical testing phase to a potential commercialization phase. Lastly, thank you Mom and Dad for being there with me and supporting me at all my hard times.

TABLE OF CONTENTS

Page

Page

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

SYMBOLS

Ω	Resistance in Ohms
μs	Micro Seconds
g	Acceleration due to gravity ($9.8\, m/s^2$)
F_X	Latitudinal Translational Force along Treatment Plane
F_Y	Longitudinal Translational Force along Treatment Plane
F_Z	Compressive Force perpendicular to Treatment Plane
$F(\theta_Z)$	Angle between Compressive Force component and Treatment Plane
Hz	Hertz
KHz	Kilo Hertz
m	Meter
MHz	Mega Hertz
N	Newtons
ns	Nano Seconds
Pa	Pascals
s	Seconds
V	Voltage

ABBREVIATIONS

1D	One Dimensional
3D	Three Dimensional
A/D	Analog to Digital
ADC	Analog to Digital Converter
CSV	Comma Separated Values
EMR	Electronic Medical Record
FFT	Fast Fourier Transform
IASTM	Instrument Assisted Soft Tissue Mobilization
I^2C	Inter-Integrated Circuit (Serial Communication Protocol)
DC	Direct Current
GPIO	General Purpose Input Output Pins
IMU	Inertial Measurement units
LPF	Low Pass Filter
MCU	Micro Controller Unit
USB	Universal Serial Bus
QSTM	Quantifiable Soft Tissue Manipulation
STM	Soft Tissue Mobilization

ABSTRACT

Bhattacharjee, Abhinaba. M.S.E.C.E., Purdue University, May 2019. A Data Requisition Treatment Instrument For Clinical Quantifiable Soft Tissue Manipulation. Major Professor: Stanley Y.P. Chien.

Soft tissue manipulation is a widely used practice by manual therapists from a variety of healthcare disciplines to evaluate and treat neuromusculoskeletal impairments using mechanical stimulation either by hand massage or specially-designed tools. The practice of a specific approach of targeted pressure application using distinguished rigid mechanical tools to breakdown adhesions, scar tissues and improve range of motion for affected joints is called Instrument-Assisted Soft Tissue Manipulation (IASTM). The efficacy of IASTM has been demonstrated as a means to improve mobility of joints, reduce pain, enhance flexibility and restore function. However, unlike the techniques of ultrasound, traction, electrical stimulation, etc. the practice of IASTM doesn't involve any standard to objectively characterize massage with physical parameters. Thus, most IASTM treatments are subjective to practitioner or patient subjective feedback, which essentially addresses a need to quantify therapeutic massage or IASTM treatment with adequate treatment parameters to document, better analyze, compare and validate STM treatment as an established, state-of-the-art practice.

This thesis focuses on the development and implementation of Quantifiable Soft Tissue Manipulation (QSTM™) Technology by designing an ergonomic, portable and miniaturized wired localized pressure applicator medical device (Q1), for characterizing soft tissue manipulation. Dose-load response in terms of forces in Newtons; pitch angle of the device; stroke frequency of massage measured within stipulated time of treatment; all in real-time has been captured to characterize a QSTM session.

A QSTM PC software (Q-WARE©) featuring a Treatment Record System subjective to individual patients to save and retrieve treatment diagnostics and a real-time graphical visual monitoring system has been developed from scratch on WINDOWS platform to successfully implement the technology. This quantitative analysis of STM treatment without visual monitoring has demonstrated inter-reliability and intra-reliability inconsistencies by clinicians in STM force application. While improved consistency of treatment application has been found when using visual monitoring from the QSTM feedback system. This system has also discriminated variabilities in application of high, medium and low dose-loads and stroke frequency analysis during targeted treatment sessions.

1. INTRODUCTION

Neuromusculoskeletal pain disorders are common world-wide. Manual therapy is a diagnosis and treatment process frequently used by clinicians to stimulate mobility, remediate impairments and restore normal body functionality using manually-applied soft tissue manipulation (STM) or instrument-assisted biomechanical therapy. In this study, our prime focus is confined to instrument-assisted mechanotherapy. For instance, STM can be used to mobilize adhesions in soft tissue injury or musculoskeletal disorders. These restrictions can result from the body's healing attempt, which may involve a lengthy inflammation process of scar tissue generation or fascial adhesions, eventually leading to aberrant tissue tension and stresses resulting in pain and immobilization [1]. Instrument-assisted cross fiber massage is an example of an IASTM technique adopted to remediate impairments stemming from fascial restrictions. IASTM practice applies forces targeted to scar tissue or regions of inflammation to reorganize affected tissues, inhibit inflammation and reduce pain along with restoration of range of motion and function.

The GRASTON Technique®(GT) is an established instrument assisted massage therapy which deals with treatment of soft tissue injuries using six different mechanical tools as shown in Fig. 1.1 [2] on page 2. GT has been used clinically to reduce adhesions, lengthen muscles and tendons, resolve inflammation, improve range of motion and restore functionality. These tools complement one another by applying forces with sustained pressure in various patterns of motion. Thus, the fundamental IASTM parameters to be recorded in real-time are: quantification of delivered force using the tool, its angle of inclination with respect to both skin and earth, the frequency of strokes applied during the treatment, duration of active treatment excluding time of treatment inactivity, and lastly, the targeted resultant force achieved by the clinician during the treatment. These parameters are of high clinical significance for charac-

terization of a massage treatment in a respective treatment session. Hence, in order to assess and analyze these parameters, it is necessary to implement a miniaturized, portable Quantifiable Soft Tissue Manipulation (QSTM™) mechatronic device, intellectual property owned by Indiana University, which involves sensor fusion and user-friendly data acquisition software to generate and display real-time treatment data for quantitative analysis and evaluation of STM treatment.

Fig. 1.1.: GRASTON Tools (Six Set)

Preliminary research for implementing massage force quantification dates back to 2016 when Alotaibi et. al. [2] proposed electronic architecture for force quantification using 1-D as well as 3-D load cells. The research contributed devising ergonomic mechatronic designs to house load cells and electronics for sensor fusion and force transformations serving clinical needs. The mechatronic system simulated the data acquisition and measurement models in LabVIEW workbench, for an electronic ver-

sion with a localizing treatment tip similar to the GT-3 tool widely used in IASTM based practices. This simulation established the proof-of-concept and feasibility of development of a clinically accepted version of the same.

Based on the foundations laid by the above simulation and testing, a study of the technical constraints, cost effectiveness and efficiency of the previous model was made to perfect the design and implementation of a clinically accepted version of the system, referred to a QSTM Q1 system. This research addresses and resolves several issues of the previous initial model with extra additions to develop QSTM to an extent ready for clinical use. Some of these issues, their solutional approaches and the additionally included contributions of this research are listed below:

1. Bulky Size− A significant amount of changes were made in the design of the mechatronic system by completely modifying the electrical architecture. This modification excluded all external electrical blocks and confined all computations into the firmware (embedded software) of the Q1 device, thus producing a more ergonomic and miniaturized QSTM Q1 treatment device.

2. Computation Speed− Several computation models were implemented and tested initially on the firmware of a beta prototype involving an AVR 8-bit MCU. After studying the issues of this implementation, the whole architecture was switched and customized to a 32-bit ARM Cortex M4 based MCU to implement the same, right from scratch.

3. Orientation Angles Calculated With Respect To Skin− The study of rigid body rotations through Euler transformations based on cartesian coordinated system led to an approximation of measuring the device inclination from the plane of skin in real-time.

4. Noise Reduction of sensor data− Different signal processing and noise filtering approaches based on time series approximation of noisy data were tested and studied. This resulted in an optimal solution to reduce noise and calculate essential QSTM parameters.

5. Stroke Counting− This research introduces a novel algorithm to count number of strokes and stroke frequencies for massage characterization by eliminating false positive stroke aberrations, minimizing missed strokes, and distinguishing the active and dead time of treatment sessions.

6. QSTM PC Software− This research also introduces a QSTM PC software built for WINDOWS right from scratch. The software includes an adaptive user interface for real-time graphical monitoring of dose-load in terms of forces in newtons, orientation angles calculated with respect to skin and earth along with analyzed treatment data. This software also features a treatment recording system subjected to individual patients, for research studies, treatment comparison and referenced clinical diagnosis. All copyrights of this QSTM PC Software (Q-Ware) are reserved to Indiana University-Purdue University at Indianapolis (IUPUI).

7. QSTM clinical standards− Based on the implemented system, some of the QSTM practice standards are illustrated and reasoned logically, to improve consistencies of force application within and between practitioners.

The structure of this thesis is organized in such a way that it illustrates and explains all the necessary aspects in a methodical way. Chapter 2 describes the literature review of IASTM its efficacies and the need to develop this QSTM technology. Chapter 3 illustrates the system overview with design specifications indicating technical and clinical requirements. Chapter 4 ponders on the mechatronic design options pinpointing the electrical component selection and differences of certain electrical and mechanical design options with some design modifications to make a portable system. Chapter 5 focuses mostly on the hardware assembly and the analysis of the electrical circuits. Chapter 6 discusses the architecture of the whole system defining the operating modes of the treatment device system. This chapter also emphasizes on the communication protocol and interfacing between the treatment device and the visualization on QSTM PC software. Chapter 7 elaborates the implemental processes

of the device firmware where the operations of microcontroller embedded logic has been elucidated based on the operation modes of the system. Chapter 8 is completely dedicated to the QSTM PC software which illustrates the PC software structure and elaborately describes the methodologies of the software applications that execute during the operating modes of the QSTM system. This chapter also discusses the statistical data analytics performed to obtain QSTM parameters and the final treatment report. Chapter 9 states necessities of hardware and software testing involved during the implementation phase along with the accuracy of different algorithms achieved. It also includes the need to set clinical standards for practicing QSTM by elaborating the analysis of some clinical data. Finally, Chapter 10 concludes the thesis with discussions of the expansion of QSTM technology and its future steps to make this clinical technology widely available with a long line of future work and clinical applications.

2. LITERATURE REVIEW

The intervention of instrument assisted soft tissue mobilization (IASTM) has by far improved assessment and efficacies [3] [4] in clinical treatment of musculoskeletal pathology. A survey in 2016 [3] shows that different IASTM methods uses rigid mechanical tools of distinguished structures to assess soft tissue adhesions and break down scar tissues. IASTM proves to be effective in performance with respect to hands on massage because the tools used have specially designed tips for deeper penetration and vibrational sensations, using targeted pressure application. Other such therapeutic forms of IASTM are Gua Sha [5] [6] and Augmented Soft Tissue Mobilization (ASTYM®) [7], the latter of which is a form of myofascial treatment using a combination of stretching and strengthening along with instrument intervention. Some of the well-known enterprises which manufactures tools for IASTM practice are GRASTON Technique®, Adhesion Breakers AB®, Fascial Abrasion Technique FAT™, Hawk Grips®, etc. Out of these the instruments designed for GRASTON Technique® (GT) employs a large variety of tooltips which provide instrument palpation and treatment [8] for targeted pathology. There are six stainless-steel GT instruments designed with specific treatment tips of different shapes which are able to penetrate the tissue and enable sensibility of their underlying textures for restrictions. These steel tools magnify palpation sensations to a greater extent as they behave analogous to a tuning fork during treatment by producing resonance-based reverberation as the instruments are moved over the contact site. These vibrations are transmitted to the clinicians hands to better enable them to detect soft tissue abnormalities and adapt treatment strokes during a session.

Various clinical studies involving IASTM are been mentioned in the following section. These studies also indicate how the different critical IASTM parameters, like application of pressure and force over stipulated time period, have different biomechanical effects on the soft tissue with a varied range of clinical implications which leads to the motivation of this research.

2.1 Preliminary Clinical Research

As mentioned by Cheatham, Scott W.et al. [3] [9] GT is a protocol of soft tissue manipulation constituted by a set of components which are: examination, cardiovascular warm-up, IASTM treatment (timed session 30-60 seconds per lesion), post treatment stretching, strengthening, and cryotherapy [9] for concerns of subacute inflammation and soreness. The GT IASTM practice is appraised with improved healing effects and tissue restructuring on both animal studies and respective therapeutic modalities on human studies.

2.1.1 Animal Study

Effects of instrument assisted cross fiber massage (IACFM) were studied by Loghmani and Warden [10] to investigate tissue level healing on knee ligament injuries by inducing bilateral knee medial collateral ligament injuries on 51 rodents. After morphological assessment rapid improvements were observed on the treated ligaments showing increased strength and stiffness with elevated energy due to more collagen fiber formation on the treated scar tissue as compared to the collateral non treated side.

A follow-up study on the same rodent model was performed [11], which demonstration enhanced regional tissue perfusion. A micro vascular morphology revealed increase in blood vessel resulting in better blood flow on the healing medial collateral ligament. These studies on rodents show that application of targeted pressure applied

during timed treatment sessions, alters bio-mechanical properties of soft tissues and can be harnessed to address healing efficiencies of soft tissue injuries and improve physical performances.

2.1.2 Human Study

Reports from a case study by Loghmani et. al [12], where a 55-year old male guitarist suffering from chronic pain and sprain due to an injured proximal interphalangeal joint of his left hands index finger experienced significant reduction in pain and improved finger grip with increased mobility in flexion range of motion just after receiving 4 successive sessions of GT-IASTM treatment.

Looney et. al [13] conducted a survey with 10 patients with plantar fasciitis a form of heel pain and found significant self-reported clinical improvements after a maximum of eight GT-IASTM sessions within a period of 3-8 weeks followed by dynamic stretching and strengthening.

Similarly, GT-IASTM proved to be significant in improving the efficiency of range of motion for shoulder injuries in overhead athletes. In a pilot study by Launder et. al [14] the GT-IASTM proved its treatment effectiveness on posterior shoulder tightness, where thirty-five baseball players were randomly grouped to study improvements in limited glenohumeral range of motion (GH-ROM) on the posterior shoulder. 35 of them were exposed to GT treatment of 40 seconds per treatment session, while the remaining were non treated control groups. Improvements were quicker for players treated with IASTM as compared to controlled groups.

Likewise, in clinical trials for overhead athletes (volleyball, softball or baseball) were performed by Heinecke M.L. et. al. [15] to assess the effects of GT-IASTM over dynamic stretching and strengthening. A group of 14 athletes were randomly grouped for treatment and non-treatment control group, where the treated group

demonstrated improved range of motion from goniometer tests and Apleys scratch test results on internal rotations, external rotations and horizontal adductions as compared to the control group.

GT-IASTM has been addressed as an effective rehabilitation measure against post-surgical complications like arthrofibrosis of knee. A GT-IASTM intervened rehabilitation treatment performed by Black D.W et. al. [16] , reports modified gait performance with reduced pain and increased mobility in quadriceps activity. The treatment was performed at the patellar tendon of surgical incision site post-surgery, with treatment bouts of 30-60 seconds each over 15 to 20 minutes, accompanied with dynamic stretching and strengthening over 30 minutes for almost 5 to 6 visits. Significant improvements in range of motion of the knee was observed after the sixth visit.

All of these studies address the effectivities of GT-IASTM protocol [9] where IASTM practice is dependent on the tool material (stainless steel), the shape of the treatment tip and the adaptations of the clinician practicing it. There are no generic standards of the IASTM practice to evaluate a treatment stroke in terms of: amount of mechanical load delivered, stipulated time for each delivery, the treatment angle and directionality of the application of these strokes. These critical treatment components form the bottleneck of our research to better measure clinical therapeutic treatment outcomes. Our main goal is to standardize therapeutic STM practice with a validated scale of physical parameters as treatment variables [17]. Treatment variables which define a Instrument Assisted Soft Tissue Manipulation treatment as subjected to our research include:

1. Force over Time− The quantification of force in terms of mechanical load delivered by the treatment tool over regular or irregular intervals of time, as guided by the underlying pathology.

2. Stroke Direction−The direction of resultant force applied by the tool during treatment following the direction of motion of the treatment tool.

3. Stroke Amplitude−The range of tool movement or average positional displacement of the tool during treatment application.

4. Stroke Rate/Rhythm−the rate defines the frequency of massage strokes applied during application. The rhythm defines the regularity of repetition of the massage strokes applied by the treatment tool.

5. Treatment Area−the amount of surface area of the skin contact covered by the tooltip during treatment. However, the area of the underlying tissue coverage should also be taken into consideration. The treatment area along with the quantified force over the area would essentially provide the measure of pressure applied during treatment.

6. Stroke Angle−the inclination of the treatment device from the skin validated with inclination reference of earth's horizon.

These treatment variables considerably can characterize a treatment in terms of STM dose to better assess post-treatment response of soft tissue in terms of elasticity, mobility, blood circulation and tissue temperature.

The background of this research discusses research on different mechatronic approaches adopted to evaluate treatment strokes in terms of quantified force and pressure, to come up with an optimum STM dose for the existing STM practice on both human subjects and animal models.

2.2 Mechatronic STM Background

This section discusses the various mechatronic instrumentation approaches developed to assess and measure STM doses in terms of mechanical loading.

Wang et al. [18] in their research developed a mechatronic experimental setup for massage loading and mechanical property characterization. The setup involved a solid framework with two movable axis mechanical loading mechanism supported with piezoelectric force sensors along a solid treatment base. The movable loading axes

were positioned to measure the compressive load i.e. vertical (z-axis) to the treatment plane, and transverse load i.e. horizontal (x-axis) to the treatment plane. These axes were controlled with stepper motor motions for loading actions. A computerized data acquisition feedback system was interfaced to monitor mechanical loading. Treated subjects for the experiments involved rats and rabbits, who were exposed to treatment tips of various shapes and sizes for dose analysis and treatment response assessment. Interesting results of various reproducible mechanical dose loading were observed on the hind limbs of treated subjects during post-treatment analysis. However, this setup is highly impractical for clinical application on human study due to its bulky size, lack of treatment maneuverability and setup complexity.

In a similar animal model with an automated mechatronic setup developed by Zeng et al. [19], the compressive loading with lengthwise strokes were studied on hind limbs of anesthetized New Zealand white female rabbits. The setup constituted a computer controlled vertical motion force transducer (Pasco, Inc., Philadelphia, PA) for compressive loading and a horizontal motion pneumatic actuator to produce lengthwise strokes. Sensor feedback was measured by the NI DAQ system. Test subjects were firmly attached to the setup without movement where the kneading wheel applying fitted to the compressive loading axis was rolled by the pneumatic actuator on the horizontal axis. However, this system can only register one dimensional compressive load magnitudes discarding the effect of forces acting horizontally along the plane of treatment. Thus, this set-up fails to adequately quantify the resultant force acting on the treatment site. Moreover, the setup inconvenience and the system complexity reject the mechanism for considerations of clinical acceptance although the study illustrates viscoelastic properties of treated tissue as a function of the delivered load.

A human study by Lee et. al. [20] performed experimentation to estimate the effects of pressure vs. time analysis as a treatment variable for Transverse Friction Massage (TFM) on Flexor Carpi Radialis (FCR) motoneuron pool excitability. This is, however, a novel approach to setup an electronic experimental arrangement to

estimate momentary pressure, stroke rate and total cumulative pressure. The setup is, promising for clinical applications as it uses separate amplified EMG recording electrodes fixed on the flexor carpi radialis to measure the H-reflex. The treatment of transverse friction massage was performed on a marked area of 15 cm2, on the FCR muscle for pressure analysis. An ultra-thin pressure sensor (ConTacts C500) mounted on a thumbpad secured by a finger glove was used to deliver the mechanical loads and record cumulative mechanical energy delivered in real-time during the application of the TFM treatment. This is performed on randomly selected 14 male subjects to measure Hmax to Mmax ratios. This treatment approach has several drawbacks for clinical acceptance. The pressure sensor detects unidirectional compressive forces only and is prone to slippage due to finger-glove contact. Additionally, it might result in inadequate measurements due to relative finger softness or sizes of therapists. Also, the setup is too sophisticated lacking durability needed for clinical applications, and frictional motions leading to slippage of the EMG electrodes might end up in inaccurate readings while exposure of sweat on the electrodes might end up in electric shock hazards. Even if this setup suffices to measure unidirectional pressures on a targeted smaller area, it would fail to account for dispersive pressures on a broader area.

Considering the instrumentation constraints needed to devise a mechatronic STM treatment tool to adequately quantify mechanical load delivered during STM in terms of forces both compressive and transverse, our research dates back to 2016 when Ahmed Alotaibi et al. [2] proposed mechatronic IASTM force sensing device designs. The proposed designs were prototyped and simulated with LabVIEW workbench to measure momentary treatment forces and treatment angles. i.e. device inclination from the earths horizontal reference frame during STM application. The treatment tool-tip followed the shape of clinically accepted GRASTON Technique tool, GT-3, for localized pressure applications of STM. The proposed initial design involved a four, unidirectional force sensor model to quantify 3 dimensional forces, including:

compressive forces acting orthogonally to the treatment plane and transverse forces acting horizontally along the treatment plane. The transverse forces include X and Y directions, while the compressive force include the Z direction.

An additional simulated finite element analysis of the same mechatronic model was performed [21] to study the Stress/Strain distribution on a human arm model simulated in ANSYS Workbench. This experimentation was exclusively executed to determine the necessity to place a 5th force sensor along the negative Y direction that would involve measurements of transverse forces opposite to the longitudinal movement of the device. Hence this analysis concluded a non-requirement of the same which can eventually reduce device cost.

However, concurrent studies with the initial prototype revealed the necessity of force quantification along the negative longitudinal direction. Hence a mechatronic design of 3D load cell measuring transverse forces both latitudinally (+X and -X) and longitudinally (+Y and -Y) along with compressive force measurement (Z direction) is prototyped in this research. This prototype has been devised to study and evaluate Quantifiable Soft Tissue Manipulation (QSTM) parameters with respect to STM treatment variables [17] except, for now, Treatment Area and Stroke Amplitude.

This research develops a QSTM system which can adequately quantify STM treatment in terms of resultant force peaks attained in every arbitrary irregular treatment stroke, and estimate device inclination from arbitrary skin plane during treatment. The research also includes a custom, computerized QSTM PC software, developed from scratch, to maintain a treatment record system for individual subjects or patients.

The next chapter describes the system overview focusing on the medical need to address different technical and clinical specifications.

3. SYSTEM OVERVIEW

3.1 Medical Need

Instrument-assisted soft tissue manipulation (IASTM) is a widely practiced mechanotherapy used in the treatment of common neuromusculoskeletal pain disorders. It offers a non-invasive, non-pharmacological and cost-effective treatment option − a national priority to address painful conditions and injuries. IASTM compliments hand massage with the use of rigid devices [9] for enhanced soft tissue assessment and treatment. It delivers a mechanical force through the intact surface of the body to affect biological and functional change through the processes of mechanotransduction. IASTM tools are mechanical tools that can be made of steel, wood, plastic, etc., which are used in conjunction with the hand massages to assess and treat soft tissue conditions. Currently, neither the manual hand massage nor the IASTM practice can adequately characterize the amount of dose pressure delivered in a treatment session. Thus, assessments are clinician dependent and have a high degree of variability in dose pressure responses. Hence, there is a significant need to measure the delivered doses accurately during soft tissue evaluation and treatment sessions. The term dose-pressure can be characterized by the amount of forces delivered on a particular area of tissue over a stipulated amount of time. Since the physical definition of pressure is applied force per unit area, the treated tissue area plays an important role in determination of dose pressures. Henceforth, recording force measurements per unit area, with determination of motion trajectory by tracking angular orientation of STM tool and quantitative measurement of stroke frequency over the complete treatment duration, can establish a Quantified Soft Tissue Manipulation (QSTM™) practice.

In order to address this medical need, we designed and developed the QSTM Q1 to measure localized pressure applications. This rehabilitation technology marks an essential step in reproducibility, standardization, comparison and optimization of STM practice to treat musculoskeletal conditions and better analyze soft tissue examinations using objective metrics. QSTM will be used on a daily basis by healthcare clinicians that employ manual therapy as part of their practice, including physical therapists, chiropractors, osteopathic physicians, athletic trainers, and veterinarians in a variety of settings, including clinics, sports and universities. This medical device system will have value in patient care, research and education and training. QSTM will revolutionize manual therapy practice with technology that can be used to apply and monitor precise, targeted forces to soft tissue, evaluate treatment parameters and analyze dose pressure responses for musculoskeletal disorders.

3.2 QSTM Parameters

QSTM parameters include stroke force (magnitude), rate (frequency and pattern), length (amplitude), direction and motion trajectory, angle of application, and treatment duration (time). All of the parameters can be adjusted according the soft tissue injury or disease, body region and location, stage of tissue healing and repair, and patient tolerance, leading to individualized rehabilitation. These parameters and related concepts will be defined and discussed here briefly, and elaborated upon as needed throughout the thesis.

3.2.1 Amount Or Magnitude Of Force

In current practice, the amount and type of forces delivered during soft tissue mechanotherapy are determined subjectively and dependent on the practitioner. Thus, it is necessary to quantify STM dose pressures in terms of measured forces and direction of motion trajectories of the IASTM tools during treatment to establish and compare treatment methods and analyze results. Massage applied to tissues during

therapy follows linear and non-linear oscillating patterns in restricted soft tissue areas for a localized treatment application. This quantification of force as related to its outcome can be described for this modality as STM dose-load response, or STM dose-pressure response based on the amount of area covered by the treatment-tip during treatment. Pertinent terminology includes:

1. **Force** (Newtons or Pounds) − An energy exerted on an object to cause motion or change. Force is a vector quantity, comprised of both magnitude and direction. Forces applied to the tissue can be measured in the form of:

   (a) *Translational Forces* − are the force vectors being applied along the plane of the tissue.

   (b) *Compressive Forces* − force vector acting normal to the plane of the tissue

   To accomplish the quantification of all force components, we define the force measurements as follows:

   (a) *±X Direction and ±Y Direction* − are the translational force components, i.e. latitudinal and longitudinal force vectors along the plane of the tissue.

   (b) *Positive Z Direction* − the force vector being applied orthogonally to the plane of the tissue.

   ±X direction and ±Y direction The resultant force of the translational and compressive force components generally describes the root mean square (RMS) force applied to the tissue. The pattern of both compressive and translational forces with respect to time provides a better description of directionality of massage.

2. **Pressure** (Newtons/Area, Pounds/Area)− Pressure is the amount of force applied per unit area i.e. (F/area).

3. **Load** − Instantaneous, resultant force applied at the point of contact of tool-tip to skin.

4. **Dose-Load**– Amount of resultant force applied to the point of contact of tool-tip to skin over one stroke cycle. Since stroke cycle is a measure of time hence the dose-load is measured resultant force as a function of time.

5. **Dose-Pressure** – As mentioned above, pressure, by definition, is force per unit area (F/area). Here, the area considerations are taken for the skin, i.e. the amount of skin area covered by tool-tip during treatment. Therefore dose-pressure can be termed as amount of dose-load delivered to the skin area covered by tool-tip during one stroke cycle i.e. force per unit area as a function of time.

The tip of the QSTM Q1 tool for our research is similar to the GT-3 device of the GRASTON Technique®, used to treat small, localized regions or areas of the body. The Q1 tip has a curved treatment end with diameter of 24mm, and the therapist determines the part of the treatment end of the tip that is actually in contact with the skin. Nonetheless, the area of skin contact changes as skin deformation occurs during application of higher and deeper pressures. Treatment observation with GT-3 tool has recorded skin reddening on particular spots of treatment, which may be more prominent in some subjects depending on race, condition and other qualities. This area of reddened skin is usually confined to a relatively small skin area ($<= 1cm^2$ approximately) as covered by the treatment tip for localized pressure applications.

3.2.2 Co-ordinate Systems For Treatment Angles

The treatment site on the patient body may not be perfectly facing upwards orthogonal to the horizon and can be in any orientation. Therefore, to enable the easy understanding of the treatment parameters, the force and motion trajectory of treatment needs to be described with respect to the orientation of the treatment site and that of tool during treatment. Hence, we consider force measurements with respect to a co-ordinate system derived with respect to the gravitational reference frame. This derived co-ordinate system is termed as the skin co-ordinate system. However, the gravitational reference frame (XYZ) is the orthogonal three axis cartesian co-ordinate

space where the X-Y plane aligns parallel to the horizon and the Z axis aligns along the direction of acceleration due to gravity. All orientation measurements are calculated with respect to this gravitational reference frame. We define the skin coordinate system as a derived approximation of rotated cartesian co-ordinate system, when the treatment plane is not aligned parallel to plane of horizon. Thus, to describe the force parameters in the skin coordinates, we adopted the coordinate transformation mechanics used in aviation technology to detect three-dimensional spatial rotation and rigid body dynamics. Rotation of medical device over time are quite similar to rigid body rotations, which are calculated with inertial measurements units like acceleration, angular momentum, and magnetic field directionality.

3.2.3 Motion Trajectory

When tissue is treated by applying dose pressures, its oscillating pattern needs to be described by the motion trajectory (path, speed, and maybe acceleration). Moreover, measuring the instantaneous angular orientation of the device with both magnitude and directions are necessary to yield information about the oscillating patterns generated during treatment.

3.2.4 Stroke Frequency

Using the geometric co-ordinate transformations from the rectangular Cartesian coordinate system to the spherical polar co-ordinate system, three-dimensional rotational data of the device can be measured along with its instantaneous angular position vector during treatment session. This rotational data along with the peaks of forces measured enables to determine the rate of change of massage strokes and the frequency at which the device is being used during a treatment session.

3.2.5 Treatment Time

Therapeutic massage is a process where the therapists apply forces by hand or with tools to the body continuously or discreetly in back and forth motion. Accounting for the treatment time is highly important, as the quantitative amount of dose-loads applied over a prolonged time with larger rest times has a different biological effect on tissue as compared to similar dose-loads applied for shorter time periods. Therefore, it is important to document the actual time of contact of the treatment tip of treatment device with the skin contact point i.e. site of treatment, and to build a mathematical model for calculating measured forces as a function of time. Consequently, the time parameters for QSTM needs to be segregated into:

1. ***Session*** − the time duration of a complete treatment session, which may include sub-sessions that involve position changes, changes of tissue area being treated, etc.

2. ***Sub-Session*** − within a session, there may be two or more sub-sessions during which significant treatment inactivity time occurs.

3. ***Total Treatment Time*** − is the total treatment duration (seconds or minutes) Total Time = Active time + Dead time:

   (a) *Active Time* − the amount of time the device is in actual contact with the tissue while measurable force is being applied.

   (b) *Dead Time* − the amount of time the device may or may not be in actual contact with the tissue but during which miniscule (treatment inactivity) no force is being applied. This includes the rest time (e.g. non-active force application time) either within a sub-session or treatment interval time between sub-sessions.

3.3 Major Design Requirements

The primary goal of this research is to design and develop a miniaturized handheld portable electronic QSTM prototype. The device should provide QSTM treatment parameter information in real-time, be user-friendly and applicable clinically as a standard of patient care. IASTM practice includes tools of different shapes and sizes for both localized and dispersive pressure applications. This research intends to focus only on devising QSTM Q1 medical device to be mostly used for the quantification of localized soft tissue manipulation treatments.

3.3.1 Technical Requirements

1. A low power device, with the ability to measure forces in 3 axes of the skin coordinate system, with translational force measurements up to 50 Newtons and the compressive force along positive Z direction up to 100 Newtons.
2. The accuracy of the all forces to be measured should be below 1 Newton.
3. Devices should be able to track and record the direction of oscillating motion patterns by measuring the instantaneous orientation of the device during treatment sessions.
4. A real-time data acquisition system with response time in milliseconds.
5. A user-friendly graphical visualization window to monitor measured forces.
6. An easy to use electronic health data record interface to enlist enrolled patients in a database, to enable the analysis of the treatment data at the end of treatment sessions and generate a report file in the form of excel sheets.

3.3.2 Clinical Requirements

1. An ergonomic, portable, compact, lightweight mechatronic design of the Q1 device, which when handheld, must cover the lateral part within the palm. The tip of the massage device should be made of steel to provide desirable sensitivity for the user and comfort to the patient.

2. The medical device should have narrow neck separating the force and electrical cavity such that, it is easy to grip the device like a pencil and apply adequate pressure to the steel tip.

3. A 10 feet long wire to connect the device to the PC. Wire should never be disconnected and sealed to the case. Provision should be made for the wire to swivel freely along its edge. Ultimately, QSTM will be wireless devices.

4. All electrical connections should be covered within the casing having no external sharp edges.

5. The medical device should be reusable, waterproofed, and able to clean and sterilize.

3.3.3 Treatment Data Analytics

The needs and requirements mentioned in the above sections of this chapter establishes necessities of formulating different mathematical models adopted from interdisciplinary engineering fields to measure, calculate and display the following data on screen as final output of every QSTM session, which are mentioned below:

1. Instantaneous Forces in X, Y and Z directions.

2. Average Compressive force (Avg Z component of Force).

3. Average Resultant Force in X, Y and Z directions (RMS of forces in X,Y,Z directions).

4. Total resultant force (RMS for force X,Y,Z component).
5. Maximum Resultant Peak Force (Maximum of RMS of force X,Y,Z components).
6. Instantaneous rotation angles in X(Pitch), Y(Roll) and Z(Yaw) axes with respect to skin.
7. Average Peak Force (Average of every stroke peak).
8. Stroke frequency per second.
9. Total Number of Strokes.
10. Total Treatment Time.
11. Treatment Active Time.

The final output of every QSTM session is not restricted to these data and are subjected to change with different upcoming versions of the PC software based on alterations in clinical requirements.

3.4 System Design

The complete QSTM system comprises of separate hardware units with their corresponding software counterparts. The Hardware of QSTM system can be separated into two parts QSTM Q1 device as the localized dose-load applicator and its Display Unit. However, the display unit can essentially be any device with a digital screen either a smart phone or a PC/MAC. Our research has confined the display unit to be a Windows PC as the visual monitoring software for display has been developed on a Windows PC.

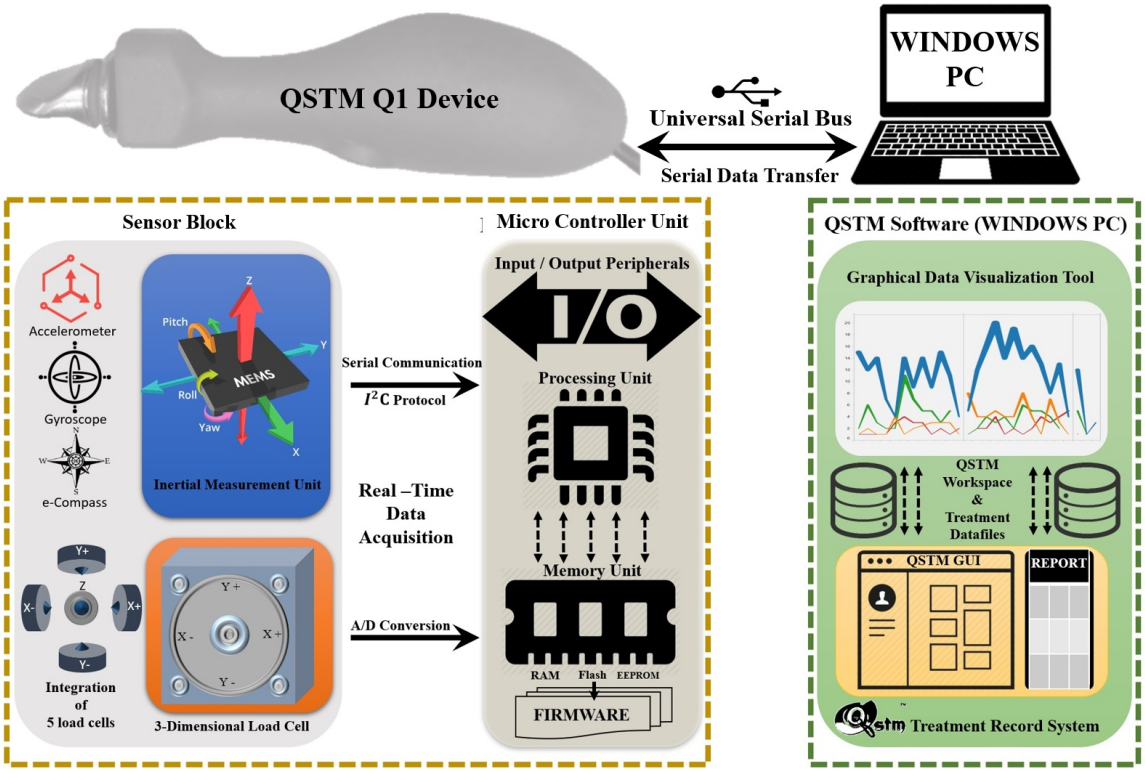


Fig. 3.1.: System Diagram of QSTM Q1 Medical Device (Wired Version)

3.4.1 Hardware Specifications

The hardware of the QSTM system has several significant blocks which includes the following units:

1. **Force Measurement Unit** – The Load cells with strain gauges to measure changes in pressure by varying electrical voltages.

2. **Inertial Measurement Unit** – These are respectively sensors which measure motion of a rigid body by measuring changes in acceleration due to gravity, angular momentum or the magnetic field intensity. An IMU sensor essentially contains a combination of accelerometer, gyroscope and magnetometers.

3. **Processing Unit** – This unit is the main decision-making unit to read sensor values and compute real-time QSTM parameters to send to the display unit. Our research uses microcontroller units as the core processing unit. The main device working methodology term as Firmware is written in this unit.

4. **Display Unit** – A windows PC has been chosen to be a display unit to visualize QSTM parameters graphically in real-time while performing a treatment.

However, the force measurement unit and the IMU comprises the sensor fusion block as shown in Figure 2. Moreover, the sensor fusion block and the MCU are consolidated into the QSTM Q1 device as an approach to miniature the architecture and retain portability. The display unit is the only external unit that is used to monitor real-time treatment data.

3.4.2 Software Specifications

As the QSTM system requires the Q1 medical device connected to a PC for its successful operation to serve its purpose during a treatment session, hence it is important to specify the software parts.

1. **Embedded Software (Firmware)** – All microcontroller units contain a significant amount of memory to store data and program. The algorithms written in the program memory (reprogrammable flash memory) is called the firmware of the device. Hence, the application firmware of the device is written in the program memory of the micro controller unit of the Q1 device. The firmware constantly communicates with the display unit transmitting real-time data to it for visual monitoring.

2. **QSTM PC Software (Q-Ware)** – This software is a windows application that can run only on PCs having Windows operating system. This software, being a part of the display unit, has been built right from scratch, which includes a health record system to enroll and enlist patient and store QSTM

treatment information. Apart from this, it also features a real-time graphical visual monitoring system which communicates with the device firmware reading and transforming sensor data in to useful information during treatment.

Fig. 3.1 on page 23 represents the complete system diagram of the wired version of the QSTM Q1 medical device and its display unit. Both the software tools of QSTM PC software work in coordination with one another and transmit data using the USB protocol. The configuration of the PC where the QSTM PC software is developed has a minimum resolution of 8GB RAM, 1 GB hard drive, 3rd gen core-i5 2.6 Ghz clock with turbo boost to 3.2GHz and 2GB AMD Radeon external graphic card. The details about the working principle and the processing involved in the QSTM software are further described in Chapter 8.

The next chapter illustrates more about the hardware and its configuration for different mechatronic designs and electrical architectures.

4. HARDWARE DESIGN OPTIONS

This chapter describes the design options for developing the QSTM treatment instrument with different electrical component selections to satisfy the QSTM requirements.

4.1 Electrical Component Selection

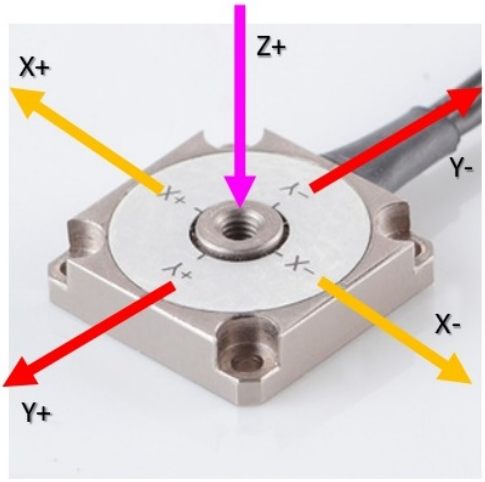
The electrical components necessary in designing the treatment instrument and its peripheral units depends on the technical requirements to satisfy clinical expectations of physical therapists already mentioned in the last chapter. To accomplish this, the following options have been chosen for the implementation of both technical and clinical aspects of the overall QSTM treatment instrument and its supporting system.

4.1.1 Force Measurement Unit

Integration of multiple standalone one-dimensional force sensors or a single embedded three-dimensional force sensor, are the options taken into considerations. Both approaches can detect and measure both compressive and translational type force components applied to the soft tissue during dose-load application. A detailed description of these sensors are explained below.

1. FC-08 (1D Load Cell) − This is the smallest compressive load cell manufactured by FORSENTEK CO.LTD. [22] It has small cylindrical shape, with a diameter of 8mm and a height of 5 mm, which can measure loads up to 200 Newtons. It has a small sensitive nob on its head which measures external vertical compressive force only. It operates in the input voltage range of 3-10V DC provides a linear output of 1mV/V and thus requires an external electrical signal amplification circuit to expand the scale to its measurement range.

(a) Forsentek FC-08 [22] (1D Load Cell)



(b) Tech Gihan USL-H5-AP [23] (3D Load Cell)

Fig. 4.1.: Proposed force measurement units

2. USL-H5-AP (3D Load Cell) – A three-dimensional high precision load cell manufactured by TECH-GIHAN CO LTD. [23] It is a typical square shaped box like structure with a facility to allow a central load shaft to be screwed-in perpendicularly. The box contains internal miniaturized strain gauges along the opening of shaft bearing screw threads which measures compressive and translational forces along Z and X-Y axes respectively. The compressive force is orthonormal to the resting plane of the sensor unit i.e. the force vector acting along the vertical direction to the sensor. The translational forces are the force vectors acting horizontally along the resting plane of the sensor as shown in Fig. 4.1(b). These forces comprise the longitudinal and latitudinal force vectors in the X-Y plane. The USL-H5-AP, being a high precision 3D load cell has the following classification that were considered in our design:

   - Non-Amplified Version – This type requires a large external amplifier device to boost the signals for interfacing with the data acquisition system.

- Pre-Amplified Version − This version contains a miniaturized signal amplification circuit embedded inside the load cell which boosts the signals acquired from the sensor in all the three axes. Hence, for this version of the load cell it is required not to screw the central load shaft too deep (less than 3.5mm) into its opening which in turn might damage the internal signal amplification circuit.

Both of these versions operate in 3.3V to 5V DC voltage range. Based on the maximum force measurement range, the sensor is classified into four types which are 50, 100, 250 and 500 Newtons of force.

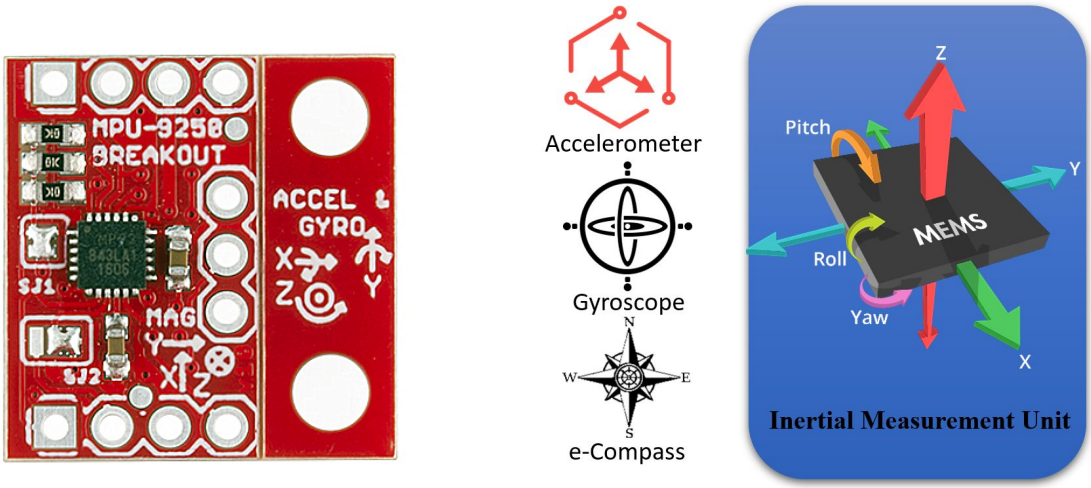
4.1.2 Inertial Measurement Unit

This is the main tilt and orientation sensing unit with an embedded nine degrees of freedom sensor comprising an accelerometer, a gyroscope and a magnetometer. It is also known as the inertial measurement unit. It makes use of specific sensor fusion algorithms for geometric transformations to yield three-dimensional spatial information to track spatial orientation of the treatment instrument with respect to the gravitational reference frame. The Z axis of both accelerometer and gyroscope aligns with the direction of acceleration due to gravity when the device is placed at rest. The nine degrees of freedom for these types of sensors are described below:

- Accelerometer − measures the rate of change of linear velocity and amount of acceleration due to gravity acting on all three axes (X, Y and Z) of the sensor respectively. Its unit is g, where 1g = 9.8 m/s2 (acceleration due to gravity).

- Gyroscope − measures the rotational motion about all the three axes (X, Y and Z) in terms of angular velocity i.e. in (degrees/sec).

- Magnetometer − measures magnetic field intensity due to earths gravity in microtesla units.



(a) 9-DOF IMU sensor MPU-9250 [24] [25]

(b) Inertial Measurement Unit sensor fusion block

Fig. 4.2.: Overview on Inertial Measurement Unit (IMU) sensors

The inertial measurement units that has been chosen as options for our proposed designs are:

1. MPU-6050 – This is a 6 Degrees of Freedom motion processing MEMs chip manufactured by INVENSENSE [26], which contains a 3-axis accelerometer and a 3-axis gyroscope module, which can measure acceleration with varying full scale ranges from $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$ and angular velocities in the range from ± 250, ± 500, ± 1000, and ± 2000 degrees/second.

2. HMC5883L Magnetometer – This is a 3-Axis Digital Compass chip manufactured by Honeywell which is capable of measuring change in magnetic fields ranging from ± 8 milli-Gauss with a minimum noise level of ± 2 milli-Gauss and field intensities in microteslas. Also, the 12-bit analog to digital channel helps in getting a good resolution data.

3. MPU-9250 – This is an upgraded version of MPU-6050 measuring 9 Degrees of Freedom MEMS chip manufactured by INVENSENSE [24], which contains a 3-axis accelerometer, a 3-axis gyroscope module forming MPU-6050 and a 3-axis magnetometer module AK8963, that can measure acceleration varying full scale ranges from $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$; angular velocities in the range from ± 250, ± 500, ± 1000, and ± 2000 degrees/sec, with measurements of magnetic field in micro Tesla. MPU-9250 is supported with a total of nine 16-bit analog to digital conversion channels making it a stable, high resolution, low noise level and low power optimized 9 DOF sensor which can be used for any motion sensing application.

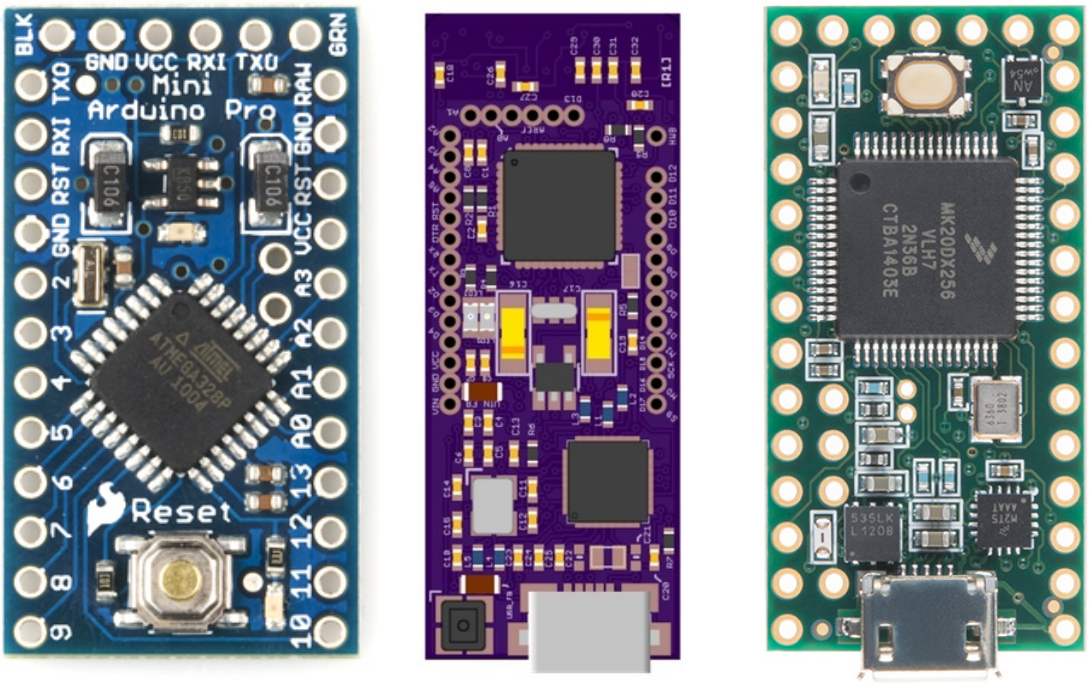
All the chips are low power activated sensors which operates in range starting from 2.6 to 3.6 Volts DC.

4.1.3 Processing Unit

This is the main decision making and control unit which reads, computes and converts raw sensor data from analog to digital domain and processes into logical meaningful information. The clinical requirements of the mechatronic design restrict the form factor of this unit to be within $40mm$ by $20mm$ in area. Henceforth, after an extensive research, the initial processing units chosen were mostly confined to commonly used 8-bit AVR microcontroller based DIY (do it yourself) development boards which are popular among hobbyist electronic engineers for prototyping and research applications. However, test results and several other performance factors led the research to adopt 32-bit ARM Cortex-M4 based microcontroller units to be standardized for the development of QSTM medical devices. The development boards opted for implementation of QSTM Q1 medical device are illustrated below:

1. AVR Family – Development boards of this family mostly comprise of 8-bit ATtiny/ ATmega/ to 32-bit ATXmega processors of the ARDUINO family.

(a) Arduino Pro Mini – This development board [27] comes with a form factor of 33mm by 18mm, which satisfies the clinical specifications and hence was chosen to implement the firmware logic of the treatment instrument - Q1 device. The on board micro-controller unit Atmels ATmega 328p supports a 16Mhz maximum clock speed with 32KB programmable flash memory, 2KB volatile SRAM, 20 dedicated GPIO with 14 digital I/O pins out of which 6 pins can facilitate pulse width modulation and the rest 8 supports Analog to digital conversion of 10-bit ADC resolution. It also supports serial communications protocols of I^2C and SPI and UART for external peripheral or sensor interfacing and can operate in both 3.3V DC to 5V



(a) Arduino Pro Mini [27] (b) IMUduino [28] (c) Teensy 3.2 [29]

Fig. 4.3.: Processing Units used to implement the QSTM Q1 prototype

DC power supply. It doesnt provide any onboard USB to serial connector, so an FTDI breakout board is recommended to be attached to it for boot-loading and flashing the program memory.

(b) IMUduino − This is an advanced low power 4 layered PCB [28] prototyping board operating at 3.3V DC which integrates several interesting technical features comprising processing unit an associated memory unit, on board embedded sensors, communications channels and add-on peripheral units listed below:

i. Atmel ATmega32u4 is an 8bit microcontroller unit with 32K flash, 2.5K RAM and 16Mhz maximum clock frequency.

ii. On board MPU6050 Six-Axis (Gyroscope/Accelerometer)

iii. Measurement Specialties MS561101BA03-50 Barometer/Altimeter Sensor (High resolution mode, 10cm)

iv. Honeywell HMC5883L 3-Axis Digital Compass IC.

v. Nordic nRF8001 low energy Bluetooth module.

vi. It consists of 33 I/O pins supporting I^2C and SPI protocols for internal serial communication and 10-bit A/D converter channels.

Apart from this it has as very small form factor of (39.8x15.72 mm) which assists in building the assembly and integration of our proposed implementation.

2. ARM Family − The development boards of ARM series mostly includes 32-bit to 64-bit multi-core processing unit ranging from cortex M0 to M7 dedicated for high computation and multithreading operations commonly used in commercialized applications.

(a) Teensy 3.2 − This development board [29] is equipped with NXPs MK20D family ARM cortex M4 32-bit processing unit [30] supporting M4 based Digital Signal Processing capabilities with 256 KB programmable flash,

64KB SRAM and an elevated clock speed of 72 Mhz overclocked to 96Mhz. It contains 16 bits hardware ADC supporting 21 analog input pins and 34 digital GPIO pins out of which 12 supports pulse width modulation. It also facilitates commonly used master-slave serial communication protocols namely I^2C, SPI, CAN, UART, I^2S buses for specific peripheral interfacing and supports Nested Vector Interrupt Control mechanism for hardware interrupts. Operated at a voltage of 3.3V DC with a form factor of 35mm by 18mm, this board serves as one of the best prototyping boards in terms of features and performance.

Finally Teensy 3.2 which outweighs the other two micro-controllers in computation power is used in prototyping the final version of the treatment instrument - QSTM Q1 Device.

4.1.4 Peripherals Unit

1. Power Source − Most of the peripheral boards to be assembled are low power consumption units which operate from 3.3V DC to 5V DC. Hence, a constant 5 Volt DC power supply unit is essential to provide uninterrupted power to the device. If the device is powered with a USB cable then, the power source is the system connected to the other terminal of the USB cable, which can be a PC, a Mobile Phone or the main Supplies to which the USB cable can be plugged in to draw power. If the device is powered wirelessly then a separate power supply unit needs to be provided to power up the circuit of the device along with a power button. It can be a rated battery unit of 6 Volt DC or 3.6 Volt DC based on the processing requirements.

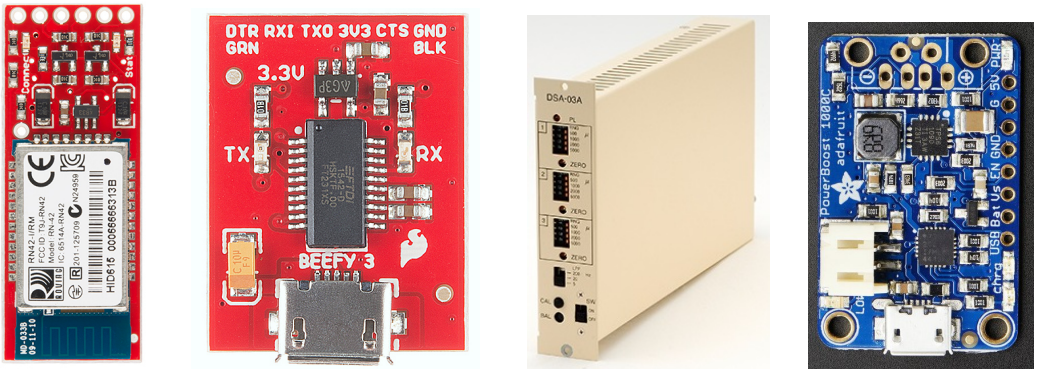
2. Wireless Bluetooth Module − This unit is important to develop a wireless version of the device. So, a fast communication Bluetooth module is required to communicate between the processing unit and the display unit using BLE or Bluetooth protocol using 2.4Ghz RF communication, which can be facilitated by the UART GPIOs of the microcontroller unit.

   (a) Nordic-nRF8001 − is a low energy Bluetooth module with BLEv4.0 specification, which is embedded in IMUduino. The module can only act in Slave mode which is a limitation to application development to some extent.

   (b) Bluesmirf-Silver − is a low range Bluetooth module manufactured by Sparkfun [31], uses the RN-42 Bluetooth Chip to establish wireless connectivity between processing and display units. It requires a RS232 to TTL converter circuit to connect to a PC without any processing unit. This unit has facilities to be configured as master, slave or auto modes and hence it is much flexible as compared to other modules.

3. FTDI Breakout Board − This board [32] supports USB to UART serial communication which uses the USB mini cable to facilitate this functionality. The microcontroller uses UART protocol to transmit data while the PC receives data using the USB protocol. Thus, to establish a successful communication the baud rates on both ends should be same, which ensures compatibility in data transmission such that the encoded serial data payload is decoded and received using the same format specified by the data rate and communication protocol.

4. Signal or Power Amplification − Two kinds of amplification circuits have been used in development of QSTM Q1 prototypes. A signal amplification circuit is mandatory, in case of the 1-D compression load cells or a non-amplified 3-D load cell. However, power amplification might be necessary for a wireless prototype to provide amplified voltage from low power source unit.



(a) BlueSmirf Silver Bluetooth module [31]

(b) FTDI Breakout Board [32]

(c) DSA03A Amplifier [33]

(d) Adafruit Powerboost [34]

Fig. 4.4.: Peripheral Units as options for QSTM Q1 medical device.

(a) DSA-03A – This is a signal amplification and a force calibration unit recommended by the 3-D load cell manufacturer Tech Gihan Co [33]. which amplifies the signal to an optimum readable scale.

(b) Adafruit Powerboost 1000C – This is a 3.6 V DC compatible power amplification come battery charging circuit [34] which can provide and output voltage of 5.2V DC such that it provides uninterrupted power to its electrical units.

The prototyped version did not use compressive load cells and hence did not use any of the signal amplification modules discussed. An earlier prototype designed for the proof of concept of QSTM used power amplification modules.

5. External reset circuit – A digital push button is used to enable the device reset function such that the device returns to its default state after completion of every session. This circuit involves a pull down resistor and a short circuit connection to the microcontroller GPIO pin, which reads the button state, whether it is high or low.

4.1.5 Display Unit

This unit is the visual unit where the clinician can observe the magnitude of the quantified QSTM parameters both digitally and graphically. It includes a screen which can display the response of the device on a graphical user interface whose layout needs to be determined by the clinicians. Approaches adopted to integrate this can either be a display fixed into the medical device by assembling a LCD or LED screen into the device or it can be kept as a separate standalone unit running on a general purpose display device like a PC, smart phone or tablet, where the later approach involves a bidirectional master slave communication between the medical device and the Display unit.

4.2 Mechatronic Design Options

The study of various commercially available IASTM tools, the QSTM parameters and the technical requirements; led our research to distribute the design specifications into different concentrations. However, our research restricts its exposure to treatment tip design for localized does-load applications. Hence the typical mechatronic design segments are:

1. Treatment tip for localized dose-load applicator as used in GT-3 tool.
2. Force Cavity to house force sensors to measure forces applicable on the treatment tip.
3. Design of the handle which includes the electronic cavity to place other electrical nodes.
4. Ergonomic outer casing of the handheld which needs to cater the clinical requirements.
5. Supporting facilities for future design modification of the design to improve precision and accuracy.

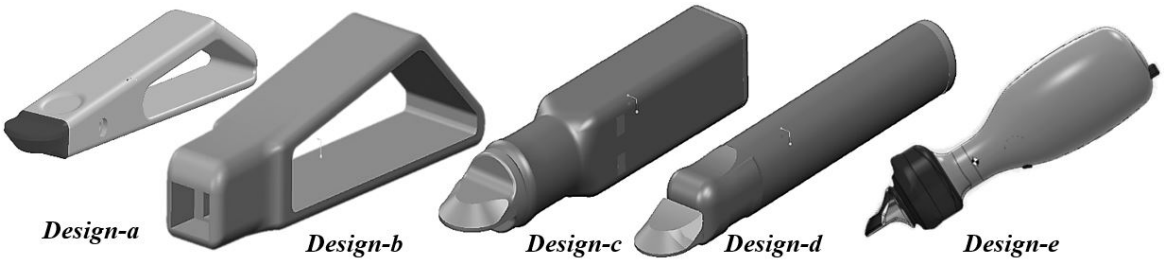


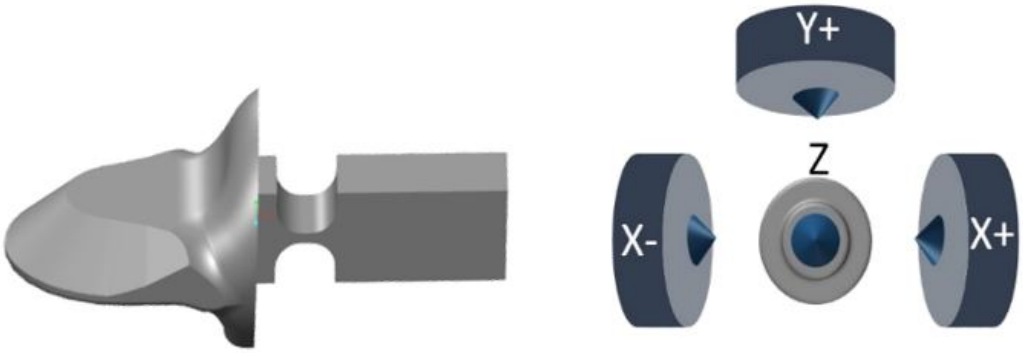
Fig. 4.5.: QSTM Q1 device design evolution based on technical requirements and clinical specifications

The initial designs were based on concepts and computational models simulated in MATLAB and LabVIEW tools, based on which our research team made two designs of two different versions of the device. All designs were initially formulated using Computer aided design tools especially like CREO Parametric and SOLIDWORKS. However, the final models were prepared in IRONCAD.

4.2.1 Prototype-A

This prototype, analogous to design-d in Fig. 4.5, was initially designed to measure pressure applications using a combination of four 1-D compression load cells, which decomposes the force vectors into three axes and computes a resultant force vector acting on the treatment tip as shown in Fig. 4.6(a) and Fig. 4.6(b) on page 38.

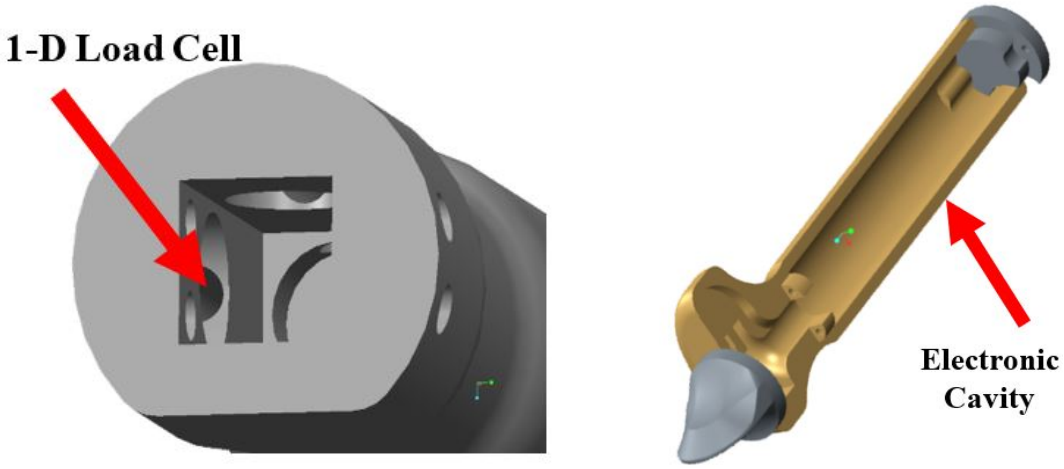
1. Treatment Tip − The treatment tip has a cuboidal extension at its back as shown in Fig. 4.6(a) on page 38 to fit into the force cavity such that the corresponding force sensors are triggered when a resultant force is acting on the tip. The steel tip, which has direct association with the load cells, is recommended to be made up of stainless steel as in GT-3, such that it registers precise forces to its corresponding load cell.



(a) Treatment tip with cuboidal extension (b) 1D Load Cells registering 3D force

Fig. 4.6.: The load cell combination and Treatment tip of design-d of Fig. 4.5

2. Force Cavity – The force cavity was especially designed to house the 1-D load cells FC-08 manufactured by Forsentek Co. Ltd, at their corresponding compartments along with their nobs facing the cavity as shown in Fig. 4.7(a), where the tip is fitted to activate the corresponding load cell based on the resultant



(a) Force cavity for Prototype-A (b) Electronic cavity for Prototype-A

Fig. 4.7.: Internal housing of prototype-A analogous to design-d of Fig. 4.5

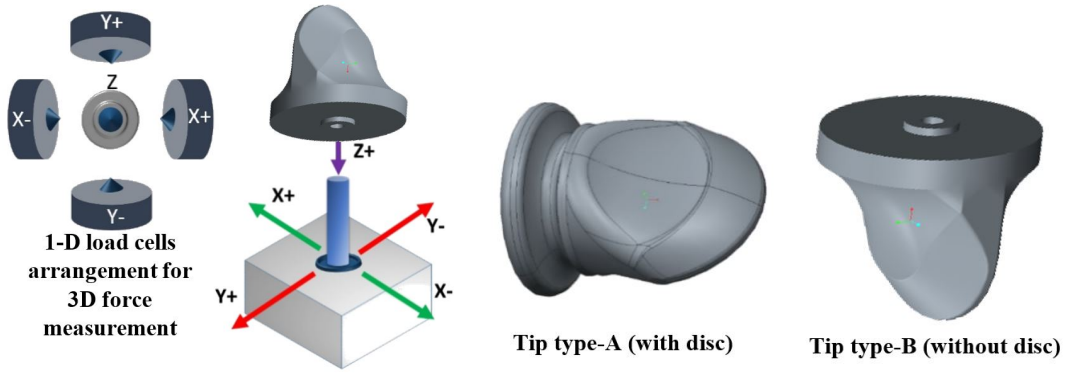
force acting on the tip. Small wiring canals were made in these load cell compartments with channels leading to electronic cavity to connect these load cells to the processing unit.

3. Electronic Cavity − This is the cavity that lies inside the handle of the device. It houses all the electronic nodes, especially the power unit, IMU sensor, processing and amplification circuit along with an external reset circuit. Hence a long cylindrical cavity has been left to decide and best fit the electronic circuit for this version of the prototype.

4. External Casing − The external chassis of the instrument defines the aesthetics and ergonomics of the device and based on the clinical requirements, there should be a connected narrow neck between the force compartment and the handle so that it is convenient to exert force on the tip and the resultant reaction force acting on the treatment tip is measured by the load cells. Additionally, it should have a sealed, completely waterproofed casing. If it has a wired communication, then provision should be made for the communication wire coming out of the device to swivel properly at its end so that the internal circuit wiring does not exert any inadvertent stretching.

4.2.2 Prototype-B

A modified prototype was proposed to measure applied forces using a composite 3-D load cell (USL-06-AP) manufactured by Tech-Gihan Co. Ltd, which precisely measures the three force components with respect to a Cartesian co-ordinate system. The force measurement of this load cell is quite alike the measurement structure designed for prototype A but with five 1-D measuring model. The only difference in this measurement structure is that, it measures translational force components acting along the plane of the skin i.e. ($\pm$X & $\pm$Y) axes along with compressive forces acting orthogonal to the plane of the skin i.e. (Z axis) of the load cell respectively.



(a) Load cell combinations for 3D force measurements

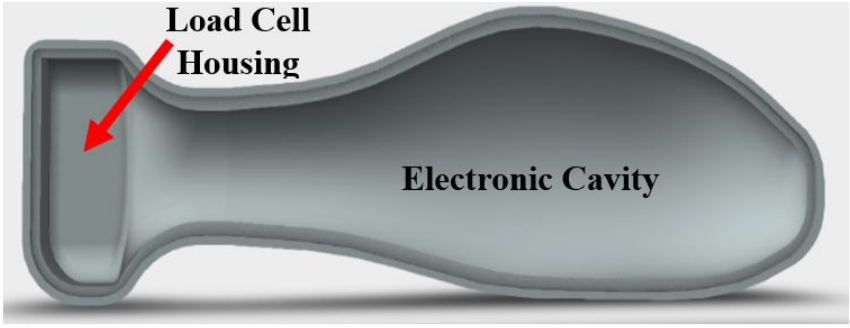
(b) Treatment Tip to support 3D load cell

Fig. 4.8.: Force measurement structure of Prototype-B

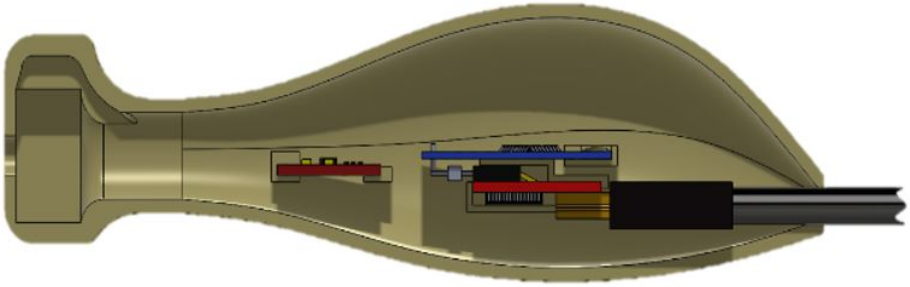
1. Treatment Tip − The treatment tip designed for this prototype is slightly different from its earlier counterpart. The treatment end of the tip is tapered according to the GT-3 tool tip while the other end is disc shaped as shown in Fig. 4.8(b), with a small opening for a load shaft to be screwed in. The other end of this load shaft fixes into the screw threads of the 3-D load cell. The load shaft, which can conveniently be connecting the tip with the force sensor, needs a nut locking mechanism to restrict it from screwing inside the load cell tunnel over 3.5 mm length.

2. Load Cell Compartment − This compartment is supposed to be the most significant compartment of the design as it houses the 3D load cell in a rectangular cavity carved out of its dimensions along with a crescent shaped extension as shown in Fig. 4.9(a) and Fig. 4.9(b) on page 43. This extension has two purposes of which, one is for ergonomic finger placement and the other is for load cell wire channeling to the processing unit in the electronic cavity.

3. Electronic Cavity − This cavity, which is the hollow space inside the handle of the device, is supposed to house all the different electrical and electronic units once the electrical architecture is decided as shown in Fig. 4.9(b) on page 41.



(a) Cad Model of Q1 Right half shell



(b) Cad Model showing component placement in electronic cavity

Fig. 4.9.: CAD model of Prototype-B showing force and electronic cavity

The electrical architecture can be either a device with wireless communication or a device with wired communication to monitor the QSTM parameters in the Display Unit. Based on the classification of the electrical architecture of the device, the electronic cavity can be populated with walls and excavations to hold the electrical units in position.

4.3 Electrical Design Options

Prototype-B serves as the working mechatronic design for the implemented versions of the medical device. This is further classified into two electrical architecture based on the communication protocol with the display unit.

4.3.1 Proposed Wireless Architecture

This architecture uses, Bluetooth communication protocol to communicate between the processing unit in the device and the external display unit. Hence, the proposal recommends using following electronic components.

1. Processing unit − IMUduino was chosen as processing unit for its on board Bluetooth module (Nordic nRF8001) and an embedded IMU sensor (MPU-6050 & HMC5883L Magnetometer) facilitated on its circuit. Due to its low cost and minimum space it proved to be a promising processing unit for a wireless version.

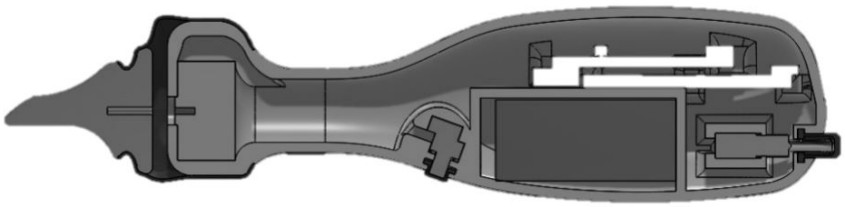
2. Power Unit − a power unit basically includes a 3.7 volts Li-ion battery to supply uninterrupted power to all electrical units in the device. It also involves a power booster circuit to amplify the voltage to 5V so that sensors of 5V range can be activated when necessary. Additionally, a charging circuit is also required to charge the li-ion battery when it is discharged.

3. Force measurement unit − This architecture should include a pre-amplified 3D load cell, which does not require an externally wired signal amplification unit. Otherwise it is better to stick to the wired architecture.

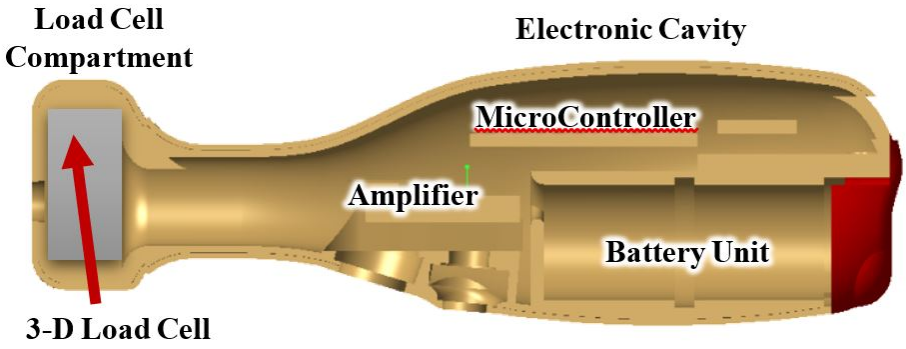
4. Power Button It is essential for a wireless prototype to have a power button because this button powers up the device for clinical use at the start of a treatment session and can be used to switch off the device once the treatment is over.

Hence, the electrical compartment for wireless version comprises of a battery cavity to hold the li-ion battery, wall trenches to hold the processing unit IMUduino and the power boost circuit along with the charging circuit in position. Moreover, an external reset circuit also needs to be included to facilitate hard reset of the device for recalibration of orientation angles during treatment sessions. The inertial measurement unit for motion sensing should be aligned with the central axis of the load shaft to make computation easier. Fig. 4.10(a) shows an embodiment to house all the

mentioned components in a compact ergonomic casing. But the major problem lies in the fact that the motion sensing IMU unit is not aligned to the central axis of the load shaft, which affects calculations required for initial calibration of the device. To resolve this issue, the components can be arranged in a cascaded structure along the electronic cavity as shown in Fig. 4.11 on page 44. But this solution does not make optimal use of space and increases the device handle, which is discarded from clinical point of view. To conclude, the proposed wireless architecture was too premature to be implemented. The following section illustrates various limitations of the wireless architecture and the need to replicate the same using a simplified wired Prototype-B along with some design modifications.



(a) Cad Design of Prototype-B (Wireless Version)



(b) Cross-Section view of Wireless prototype-B with electronic cavity

Fig. 4.10.: CAD model of a wireless Prototype-B showing force and electronic cavity

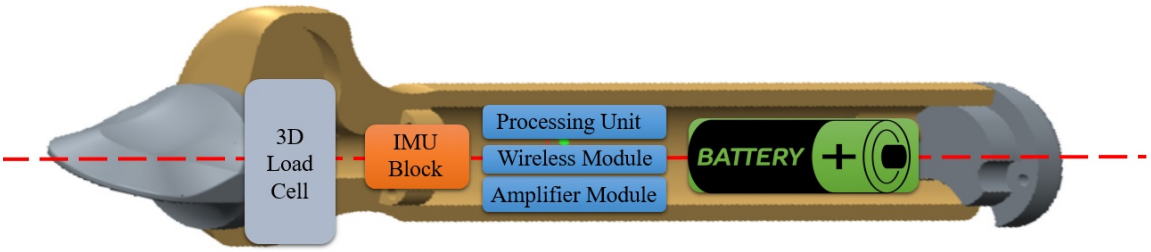


Fig. 4.11.: Cascaded electronic blocks in the electronic cavity of Prototype-B (Wireless version)

4.3.2 Limitations Of Proposed Wireless Architecture

The design in the wireless version used a complex electrical architecture which is difficult to arrange in a small electronic cavity. The significant limitations of the wireless prototype design are listed below:

1. IMUduino, although a highly compact processing unit supporting a lot of features, has its accelerometer, gyroscope and magnetometer in different positions of the board. Therefore, it is difficult to align them with the central axis of the load shaft entering the load cell.

2. The battery unit consumes a lot of space in the electronic cavity and hence it is difficult to make space for extra peripherals. Also, a power amplification circuit was required to activate the load cell in its highest resolution. Eventually, a voltage divider circuit was necessary to step down the input voltage to activate the low power electronic units.

3. The complexity of the electrical circuit was high as it included a power button, a reset button, a voltage amplification circuit, a filter circuit and a battery charging circuit along with a li-ion battery. Assembling all these components in

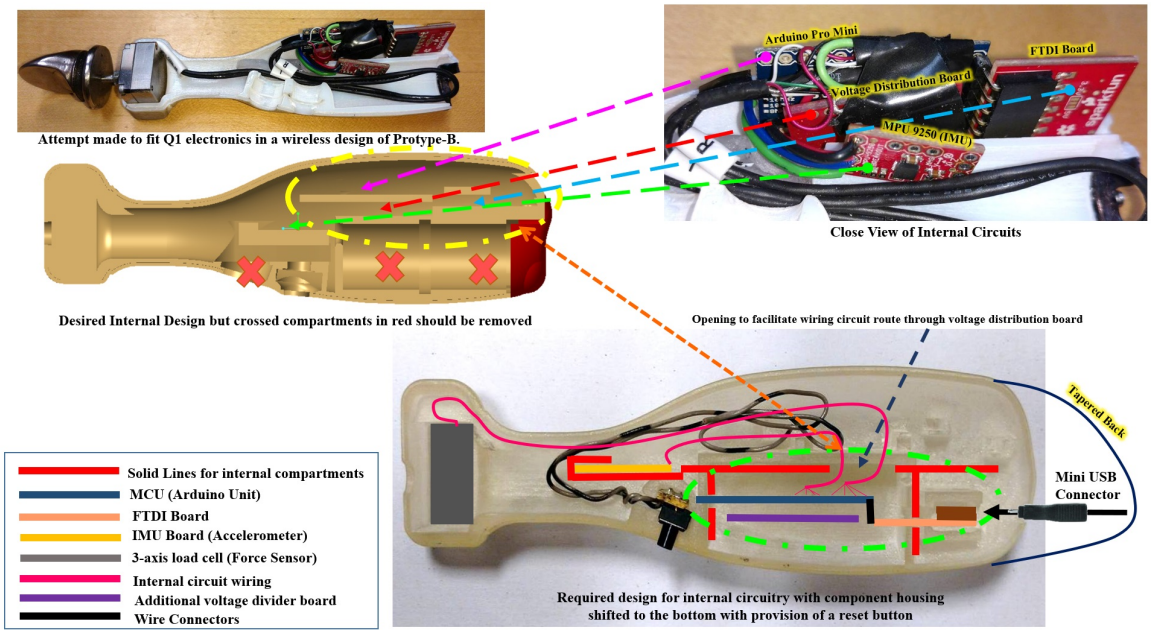


Fig. 4.12.: Modifications necessary to design the electrical component's housing for a wired prototype-B

such a space restricted electronic cavity is a bigger challenge, unless a customized printed circuit board is developed solely to integrate all electrical modules on a single chip.

4. The design had its battery unit at its bottom hemisphere with the processing unit positioned in the upper half. A USB cable connected to the processing unit determines the holding of the device from clinical perspective. Hence it needs to be moved to the lower half with the device having a smoothed tapered end with USB cable coming out of tail end.

Henceforth, electrical architecture of the proposed wireless version of Prototype-B due to various design constraints and technical limitations could not be successfully implemented. Thus, our research focused on proposing a design for a wired version, which is more simplified in terms of electrical architecture, computation, optimal space usage and device response, satisfying all the clinical requirements.

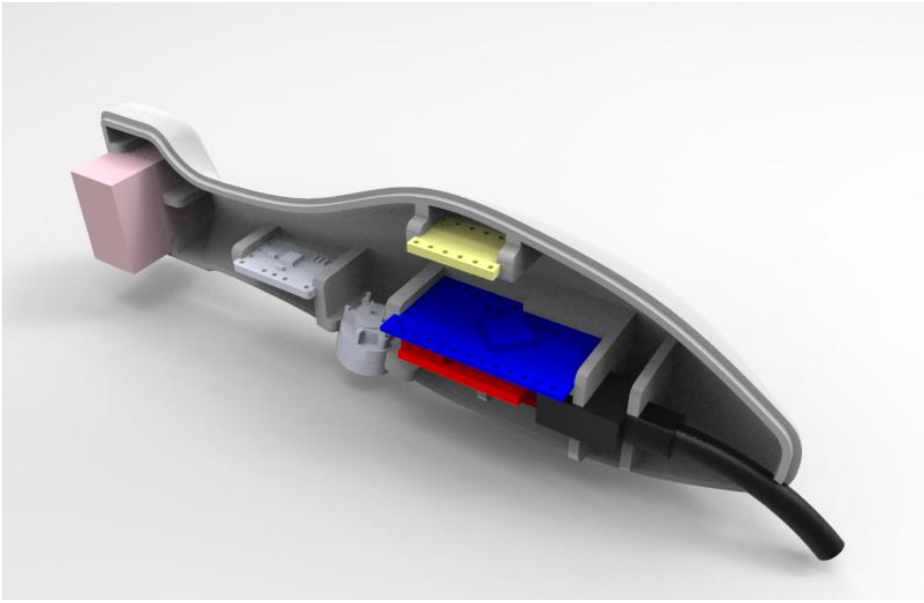


Fig. 4.13.: Cross-sectional view of wired version of Prototype-B

4.3.3 Proposed Wired Architecture

The wired electrical architecture is quite simplified as compared to the former wireless version. This version does not include specific space consuming compartment to house batteries for power supply. Henceforth there is no need of power amplification or a battery charging circuit to be spaced in the electrical cavity. Here the IMU sensor is separated from the processing unit to resolve the alignment issue unsolved in the former version. The wire connected to the display unit (WINDOWS PC) provides enough power to activate the electrical circuit. However, an external reset circuit is still necessary, as it facilitates hard reset of the device for system re-calibration. The electrical components adopted for this electrical architecture is further illustrated below:

1. Force Measuring Unit − Since this is a subset of Prototype B, this unit is confined to a pre-amplified 3-D load cell for force sensing.

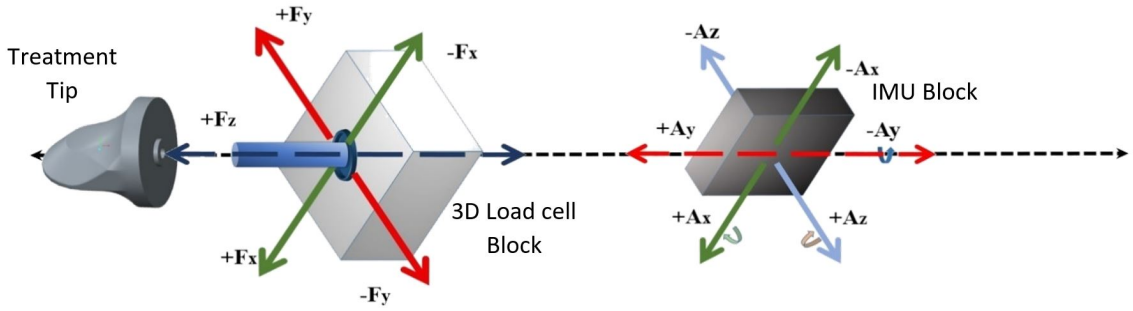


Fig. 4.14.: Alignment of the IMU sensor with the central axis of 3D load cell

2. Inertial Measurement Unit – The discontinuity of IMUduino enabled our research to test more advanced, high resolution 9 degrees of freedom commercially available motion sensing units. MPU-9250, a 16-bit low noise level inertial measurement unit, changed the internal spacing of the electrical components in the electrical cavity. Another important factor which has been modified in this architecture is the placement of the IMU. The walls holding the IMU is placed in such a way that it should be aligned to the central axis of the load shaft holding bearing the steel tip as shown in Fig. 4.14.

3. Processing Unit – Taking the form factor into account, AVR family's 8-bit Arduino Pro mini served as a processing unit of a beta version for this architecture. Its low cost and minimum space proved to be an apt processing unit for the wired version. Arduino Pro Mini does not come with an on board USB to Serial connector. Thus an external USB to serial FTDI connector is required to attach to this board to establish serial communication between PC and device. However, the final version of this architecture uses a 32-bit ARM CORTEX M4 processor for faster signal processing and better memory usage.

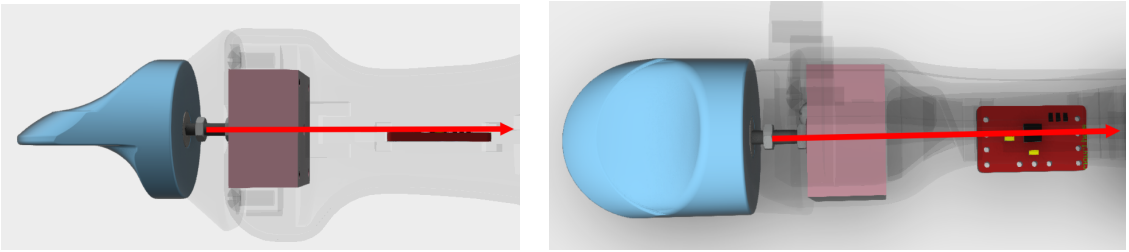
4. Peripheral Unit – An additional power distribution board is needed to provide adequate power to the force sensor- operating at 5V dc and the IMU sensor operating at 3.3V DC. Results showed that the serial data and the serial clock channel of the I^2C bus in Arduino Pro Mini couldnt support 5V DC. Thus the all over circuit was powered by a 3.3V compatible FTDI board.

4.4 Final Hardware Modifications

The design modifications mostly involve re-designing the internal casing for a wired version of prototype-B by populating the electronic cavity with different electrical nodes as per convenience of computation and assembling the components. The necessary modifications incorporates the following changes in the design.

4.4.1 Electronic Modifications

1. The IMU sensor should be positioned anywhere along the central axis of the load shaft connecting the treatment tip, such that the roll axis (Y axis) of the sensor should correspond to the Z axis of the load cell as shown in Fig. 4.14 on page 47. The red arrow in Fig. 4.15 shows the modified cad models establishing the fact that the IMU is aligned properly.



(a) Side view of aligned IMU with central axis (b) Top view of aligned IMU with central axis

Fig. 4.15.: Red arrow showing the central alignment of the IMU with central axis of load cell shaft in the Cad model

2. The processing unit should be placed such that the wire connected to it extends from the bottom of the tail end.

3. Provision of a reset button to facilitate button click reset during transition of sessions in between treatments.

4. Switching from a low resolution 8-bit processing unit to a high resolution 32-bit processing unit Teensy 3.2. This makes computation almost 6 times faster and increases memory by 8 times, with higher performance, better efficiency and faster response.

5. Switching the processing unit also eliminates the FTDI board from the electronic cavity. The Teensy 3.2 includes a USB to serial connector for flashing and serial communication.

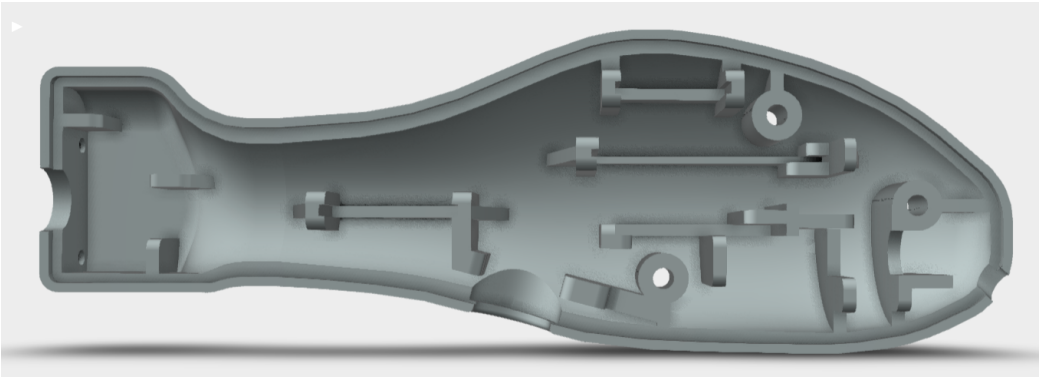
6. Since Teensy operates at 3.3V as the base voltage, a voltage divider circuit is required to scale 5V logic to 3.3V logic, such that the Force sensor can operate at its full resolution when activated a 5V DC.

4.4.2 Design Modifications

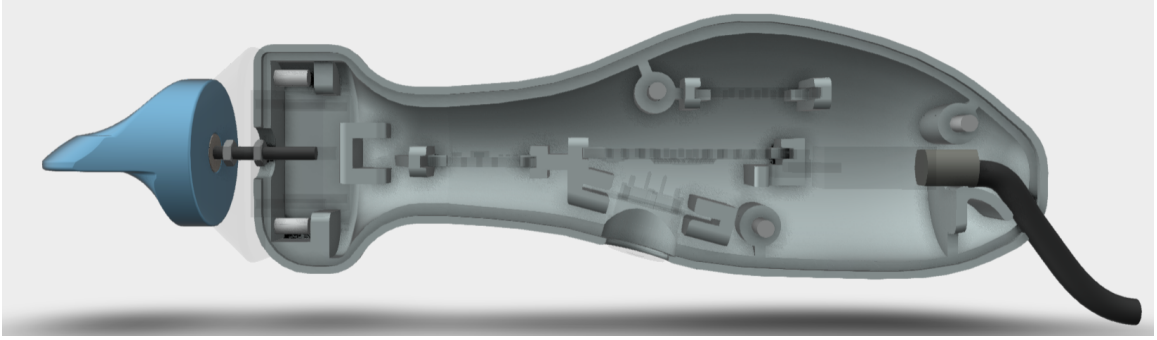
1. The tail end should have a smoothed tapered carving so that it is convenient to grip the device with the palm and exert force from the tail end if necessary.

2. The two halves of the device are made such that one fit onto another to enclose gaps in between the halves.

3. Provision of supporting screws are made to enclose the gaps within the front and back half of the prototype after assembling the electrical components. However, locking nuts need to be provided to fix the tip and screw the force sensor in its place.



(a) Force cavity-base not aligned with the electronic cavity-base



(b) Force cavity with rounded external surface and resting flat with the base.

Fig. 4.16.: CAD model modifications to align base of force cavity with base of electronic cavity to enable flat resting of device

4. Observations of the beta version showed that the base of force cavity is slightly higher than that of the electrical cavity, due to which the central axis was always tilted when the device rested flat as in Fig. 4.16(a) Hence the force cavity is enlarged towards the bottom to rest the device flat on a surface as in Fig. 4.16(b). This is required for the initial calibration of the device.

5. The external surface at the bottom of force cavity needed to be rounded as much as possible so that there is minimal interference and grazing of the edges with the skin during treatment.

6. The wire coming out of the device needs to swivel around its opening such that external strains do not harm the internal circuit or break the communication. Thus, a lock wall has been included in the internal casing such that external strains does not move the position of the USB port.

These modifications were made by working with collaboration with Rose-Hulman Ventures, Inc. Thus, the above-mentioned modifications are necessary for implementing a working prototype with wired communication between the device and the visual monitoring system i.e. the display unit.

The next chapter demonstrates the implementation of the electrical circuits and the hardware assembly of the treatment device.

5. HARDWARE IMPLEMENTATION

Our research, aims at implementing prototype-B of QSTM Q1 medical device which follows the wired electrical architecture, as described in the last chapter. So, this chapter focuses on illustrating the electrical architecture of a wired version with some initial circuit testing approaches and its limitations, followed by the final hardware assembly and circuit analysis of the same.

5.1 Beta Version Prototyping

The initial circuit implementation was tested on a breadboard based prototyping platform as shown in Fig 5.1 on page 53. The processing unit used for testing was AVR family's Arduino UNO, an 8-bit micro controller unit supporting ATmega 328p controller. A 3-D force sensor of the range 0-500 Newton's was tested at the same platform to layout the electrical circuitry for the device. Since, the microcontroller development board supports 5 Volts DC power source, hence the force sensor operated at its rated voltage resolution. However, the operating voltage rating for Inertial Measurement Unit was 3.3 to 3.6 Volts DC. Fortunately, the microcontroller development board additionally supported a stepped down 3.3V power source and hence the power distribution issue was solved. But the main challenge laid in the fact that the overall surface area of Arduino UNO did not satisfy the clinical form factor requirements. Thus, this board was discarded from further usage. Nevertheless, a similar miniaturized version of the same micro-controller development board Arduino Pro Mini replaced the discarded board. This board contained the same 8-bit controller Atmega 328p, whose form factor satisfied the clinical specifications too.

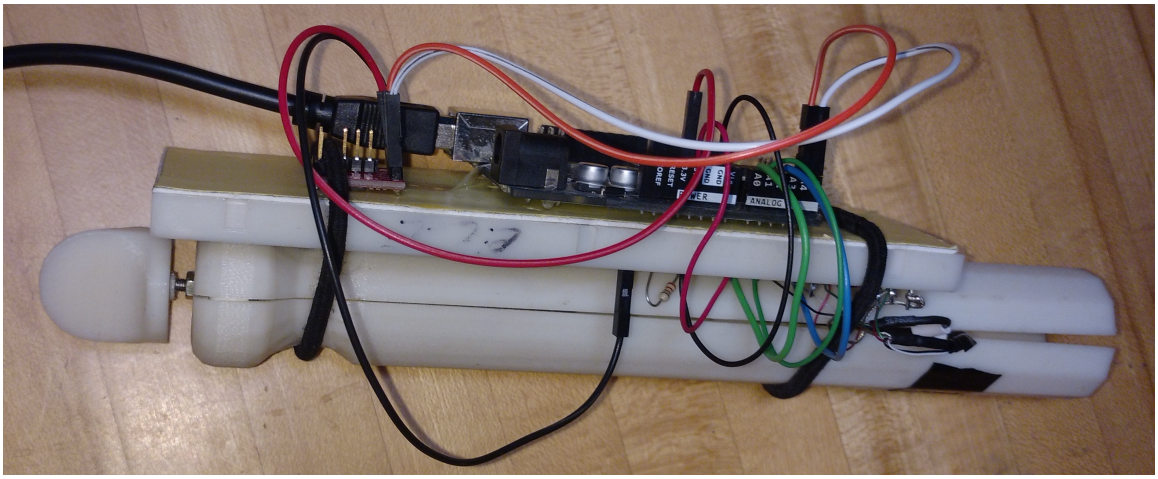


Fig. 5.1.: Initial testing prototype on a breadboard platform

An implementation of this version resulted in various hardware limitations, which yielded several computational constraints and performance issues and ended into an unsuccessful implementation.

5.1.1 Restrictions Of Beta Version

This section elaborates the bottleneck of both the hardware and computational limitations of the beta version of QSTM Q1 prototype.

Force Sensing Arduino Pro Mini supports 10-bit approximation analog to digital converters. The 3D load cell voltage rating is 5V DC. However, the load cells no-load voltage is approximately 2.5 volts i.e. this is the activation voltage of the load cell. Rise in every quantization level would approximately produce a resolution of 1 Newton for a 500N 3D load cell. This limits precision of the device and affects noise normalization below 1Newton. Thus, a lower range 3D load cell measuring 0-100N was preffered for implementation after clinical agreement.

Power Distribution - Arduino Pro Mini did not support an internal voltage regulator. Consequently, it operated on the power delivered by the external USB to serial FTDI converter. That is why the power output pin of the MCU board

generated the same voltage transmitted by the FTDI board. An attempt to power up the MCU board with 5V compatible FTDI converter resulted in running I^2C bus of the MCU at 5V logic level. Eventually the IMU sensor which operates at 3.3V could not read data and became dysfunctional. Therefore, the circuit was confined to 3.3V compatible FTDI converter, which compromised the full hardware resolution of the 3D load cell. Subsequently voltage manipulations by force scaling was required to validate sensor reads of the load cell.

Memory Utility The AVR family's Atmega 328p MCU has a RAM of 2KB and a program memory of 32KB. The algorithm developed to calibrate 3D load cell and reduce sensor noise using low pass filters and smoothing filters requires extensive floating-point arithmetic. Every variable which stores a single precision floating point element uses up 4bytes of RAM. Eventually, the MCU freezes to process extensive floating-point calculations and the RAM tends to max out.

5.2 Electrical Implementation

The restrictions listed above proved to compromise a successful and precise implementation of the whole system. To resolve these severe issues, our research team agreed on discarding 8-bit AVR based MCU and adopting higher resolution 32-bit MCU from the ARM family. The ARM COXTEX M4 is a 32-bit architecture which supports distinct hardware extensions for Digital Signal Processing (DSP) and Floating-Point Units (FPU) for float type arithmetic. Keeping in mind the form factor requirements from Clinical standpoint to fit the processing unit inside the device, our research came up with accepting Teensy 3.2 as the processing unit.

5.2.1 Resolving Hardware Restrictions

This development board features the MK20DX256VLHS controller chip manufactured by NXP, resolves all the restrictions mentioned in the last section. This chip supports 16-bit approximation analog to digital conversion (ADC) out of which 13-bits are made usable by the Teensy manufacturers.

1. Even on using 12-bit resolution of ADC, rise in every quantization level could yield a resolution of 0.25 Newton for a 0-500N 3D load cell, assuming the activation to be 2.5V DC. Henceforth, readings from a 3D load cell ranging from 0-100N would be highly precise.

2. Teensy 3.2 operates at a 3.3V logic level. The board also features 5V compatible digital GPIO pins, but the analog pins do not support 5V input and thus they are restricted to 3.3V logic. Thus, to operate the 3D load cell at its rated resolution, the load cell needs to be powered up by a 5V DC power source with the sensor output channels stepped down to a 3.3V logic. This is possible by a corresponding 5V to 3.3V voltage divider circuit. Moreover, Teensy 3.2 facilitates a power pin shorted to the micro USB connectors power pin. USB devices always outputs 5V DC, which can be harnessed to supply 5V DC input voltage to the 3D load cell.

3. Teensy 3.2 features 64KB RAM, almost 32 times more than that of Arduino Pro Mini. Its clock frequency of 72 Mhz dedicates it for faster higher complexity arithmetic and its distinct DSP unit for running advanced filtering logic. It can accommodate more than ten thousand float type variables at run time without freezing. Therefore, computational complexity specific to our application would not freeze the system.

5.2.2 Electrical Circuit Analysis

The digital circuit for the ARM based processing unit was initially tested on a breadboard platform. After powering up the circuit with a USB device, current and voltages at different GPIO pins were measured using a digital multimeter. The load cell output was tested separately, connecting it to a 3.3V DC power line and a 5V DC power line. Output voltages at different forces were measured using a cathode ray oscilloscope. Observations proved that analog inputs pins of the processing unit are not 5V DC compatible. However, as mentioned in the manufacturer data sheet, the 3D load cell operates in its highest resolution when power to 5V DC. Thus, there was a high necessity to introduce a voltage divider circuit for each of the load cells X, Y and Z channels such that the sensors output measurements at 5V logic is stepped down to 3.3V logic. Based on this knowledge we developed the voltage divider circuit with appropriate resistances, which after integrating completed the overall working electrical connections. The schematic diagram in Fig 5.2 shows the

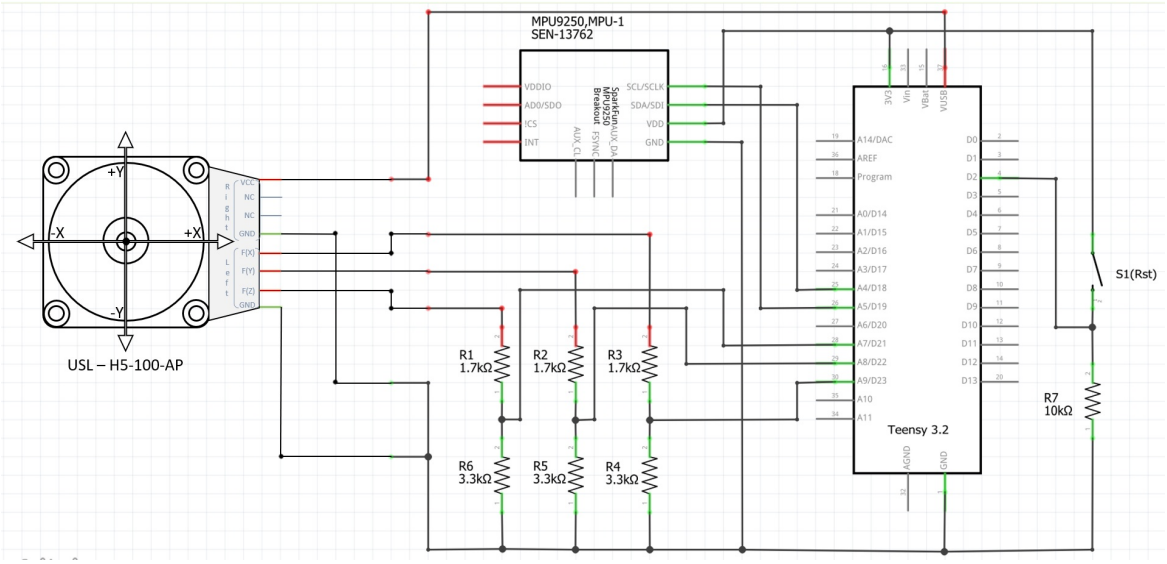


Fig. 5.2.: Schematic diagram of electrical circuit of QSTM Q1 Medical device

overall connectivity of sensors and other peripheral units with the microcontroller Teensy 3.2. All the connected red markings resemble load cell connections, while

the green connections resemble other connections. However, it is evident from the diagram that VCC pin of the 3D load cell is connected to the V_USB pin of Teensy 3.2. This confirms that the supply voltage of the load cell is directly fed from the USB output i.e. 5V DC. Subsequently, connections from load cell X, Y and Z channels are grounded after passing them through a series resistance summed up to 5KΩ. The analog channel of Teensy 3.2 reads the voltage drop across the 3.3KΩ resistance, thus confirming the voltage division. The connections for the motion sensor (MPU-9250) are quite straight forward. The power pins are connected to the 3.3V and GND pins of Teensy 3.2, while the data pin SDI and clock pin SCL are connected to the I^2C bus of Teensy 3.2. The push button which is supposed to be the reset button, is grounded with a pull-down resistance of 10KΩ to avoid short circuit issues at closed switch.

5.2.3 Electrical Circuit Wiring

The restrictions listed above proved to compromise a successful and precise implementation of the whole system. To resolve these severe issues, our research team agreed on discarding 8-bit AVR based MCU and adopting higher resolution 32-bit MCU from the ARM family. The ARM COXTEX M4 is a 32-bit architecture which supports distinct hardware extensions for Digital Signal Processing (DSP) and Floating-Point Units (FPU) for float type arithmetic. Keeping in mind the form factor requirements from Clinical standpoint to fit the processing unit inside the device, our research came up with accepting Teensy 3.2 as the processing unit.

3D Load Cell The load cell contains two wire bundles − namely left and right each of which consists of Red, Green, Black and White wires. Table 5.1 shows the Load Cell Connection. The red wire, white wire and the black wire in the left bundle are connected to analog input pins A7, A8, A9 respectively, routing through the voltage divider circuit. These pins are internally connected to analog to digital converter - ADC_0 of Teensy 3.2. To restore wiring color conventions, the green ground wires of

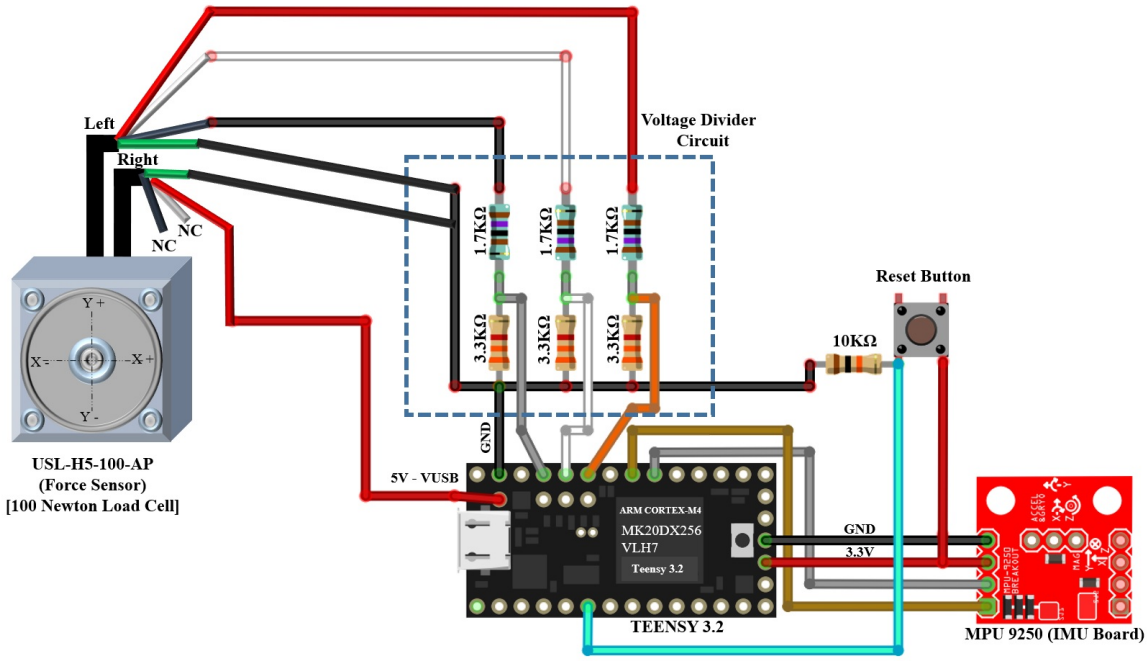


Fig. 5.3.: Wiring diagram of overall electrical circuit of QSTM Q1 Medical device

the load cell are connected to the black ground wires of the processing unit as shown in Fig. 5.3 on page 58.

Table 5.1.: USL-H5-AP Load Cell Connections

Left Wire Bundle		**Right Wire Bundle**	
Color	Description	Color	Description
Red	Force X Value (F_X)	Red	Supply Voltage VCC 5V
White	Force Y Value (F_Y)	White	No Connection
Black	Force Z Value (F_Z)	Black	No Connection
Green	Ground	Green	Ground

IMU Sensor (MPU-9250) The IMU sensor is a nano-electro-mechanical sensor fusion module. It uses master and slave inter integrated circuit (I^2C) serial communication protocol to send and receive data from the microcontroller unit. This protocol is widely used to reduce GPIO pin usage for parallel communication and save space. I^2C uses two channels for communication SDI i.e. serial data input and SCL i.e. serial clock. The serial clock generates the pulse to send data, while the SDI channel sends data in bits per frame format whenever a clock pulse is triggered. Ochre and dark grey wires are connected to GPIO A5 and A4 of Teensy 3.2 respectively. These two GPIO pins supports I^2C communication with a maximum speed of more than 400kbps. The Red and the Black pins are connected to power and ground pins which provides 3.3V from the processing unit.

Reset Button The reset button is a digital push button or toggle switch which activates on pressing it. One leg is connected to ground through the pull-down resistor and black wire, while the other is connected to power pin 3.3V DC through red wire. So, when the button is un-pressed it is pulled down to ground by the 10KΩ pull down resistor. The cyan colored wire makes a parallel connection from the push button site to the digital GPIO D4. When the button is pressed, pin D4 is shorted to 3.3V and reads HIGH, while released reads LOW for open circuit. This fulfills activation of reset condition for the QSTM Q1 device.

Table 5.2.: MPU 9250 9DOF IMU Sensor Connections

Color	**Pin Name**	**Description**
Ochre	SCL	Serial Clock pin for I^2C Communication
Dark Grey	SDA	Serial Data pin for I^2C Communication
Red	VCC(3.3V)	Power Supply Pin − Activates only at 3.3V DC
Black	Ground	Ground Pin

5.3 Hardware Assembly

Fig. 5.4 on page 60 shows the hardware block diagram of QSTM Q1 medical device, fulfilling all technical as well as the clinical requirements. The circuit used to implement a working prototype of this device uses minimal electrical components for the wired architecture described above. The mechatronic assembly involves screwing the tip with an M3 screw and locking the tip edge with M3 anti-vibrational nut. The other end of the M3 screw goes into the 3D load cell. Once the electrical connections are made and tested successfully, the components can be placed in their respective slots enclosing the left half. Next part elaborates the ease of electrical assembly for manufacturing aspect and the clinical convenience for practicing massage sessions.

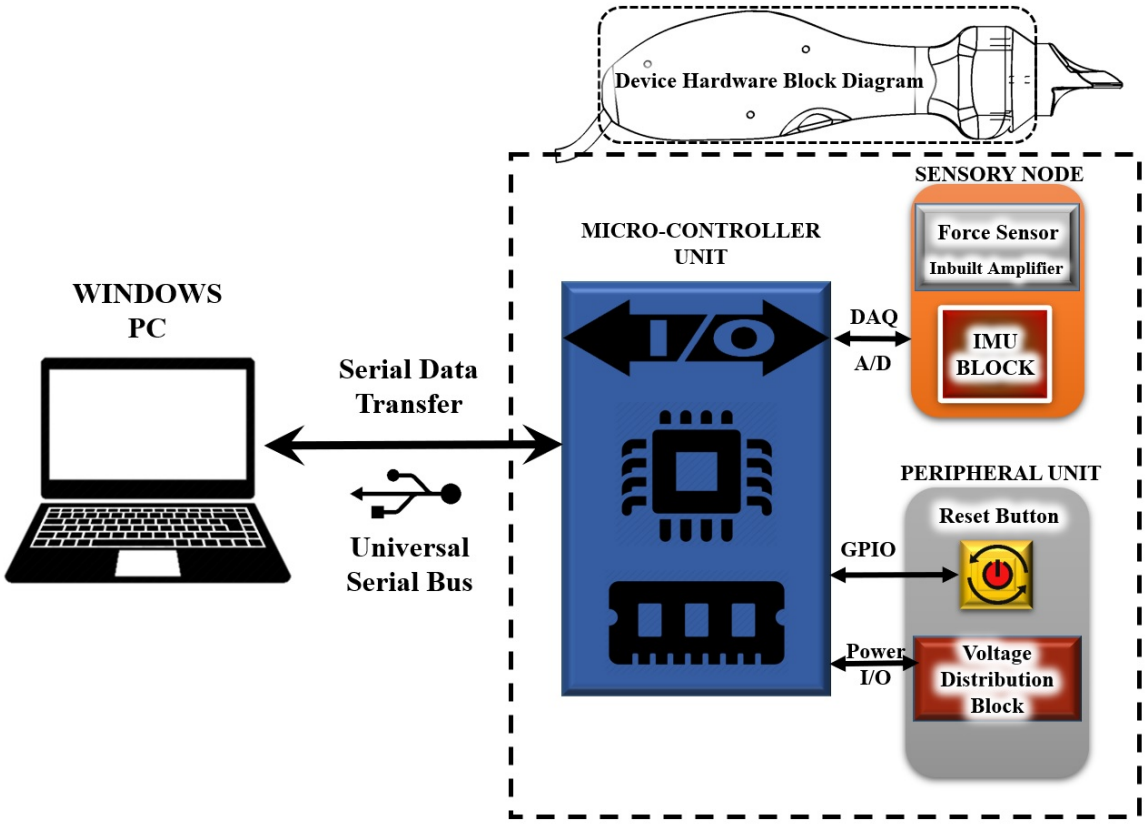


Fig. 5.4.: Hardware block diagram of QSTM Q1 device

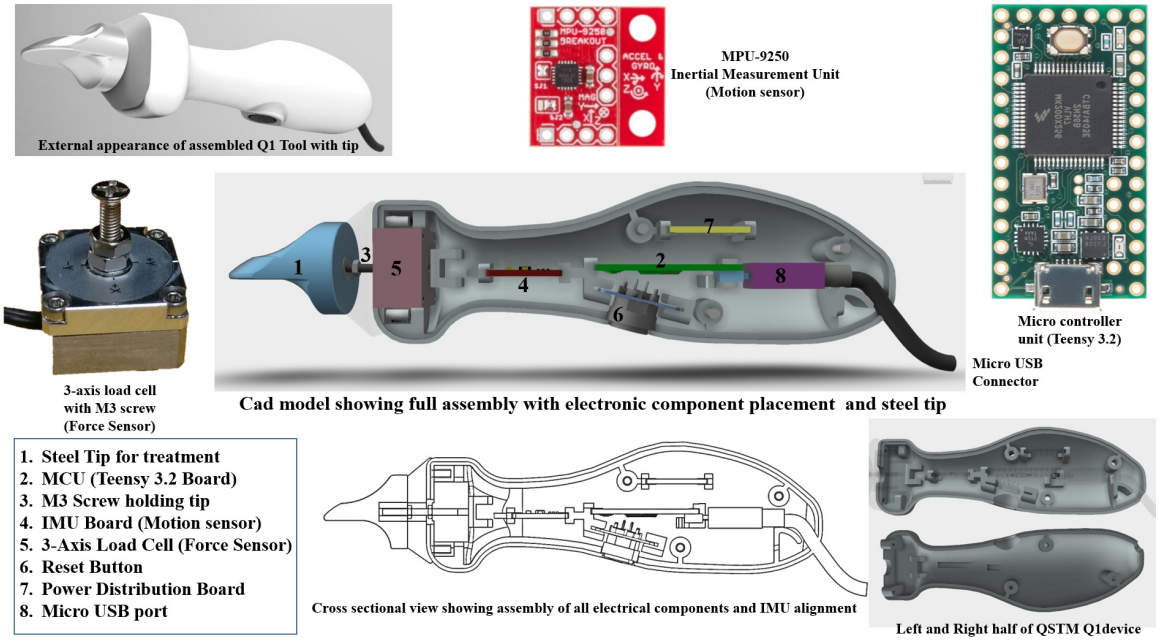


Fig. 5.5.: Layout diagram illustrating electronic and mechatronic assembly of QSTM Q1 medical device

5.3.1 Clinical Convenience

1. The ergonomic external appearance of the device with a narrow neck makes it convenient for clinicians to exert desired amount of force on the treatment tip.

2. The hump shaped middle portion and tapered tail end of the device allows proper grip of the palm to hold it during ongoing treatment.

3. The internal walls at the tail of right half of the device provides rigid support to the fixture of the USB micro cable at block 8 in Fig. 5.5 on page 61, such that it can easily swivel around the orifice from which the wire comes out. Moreover, it also assures that external shear stresses on the device due to abrupt pull would not harm the connectivity of the power cord as the walls provide intact support to fix it in its position.

4. The wire orifice from the device is purposely made at the bottom so that the external surface of the tapered tail can exclusively be used for gripping the device.

5. Provisions have been made to include foam structures in between the 3D load cell block and the treatment tip to protect the supporting M3 screw and preserve play of the treatment tip.

6. The bottom of the force cavity is externally rounded to avoid any interference or grazing of the device casing through the skin during treatment.

5.3.2 Technical Convenience

1. The hump shaped middle portion allows enough room for housing all the electronics in the medical device in a distributed orientation as shown in Fig. 5.5 on page 61. This structure also enables leaving some extra wires to fix future wear and tear.

2. The IMU placement at block 4 in Fig. 5.5 on page 61 ensures it aligns with the central axis of the load shaft connecting the 3D load cell and the steel tip.

3. An additional slot for a voltage distribution and division circuit is provided at block 7 in Fig. 5.5 on page 61, to distribute parallel connections and step-down input logic level for 3D load cell.

4. The button at block 6 is supported by a button plate as shown in Fig. 5.5 on page 61. This holds the button in place, restricts any lateral movement of button and prevents any false button click.

5. Block 5 in Fig. 5.5 on page 61 is the Force cavity where enough room has been left to leave some extra wire of the 3D load cell for future use and repair purpose.

6. The Steel tip at block 1 in Fig. 5.5 on page 61 is connected to the Load cell by an M3 screw. A locking arrangement has been enable using anti-vibrational m3 nuts to support over-screwing the load cell which might lead to its permanent damage.

7. The walls have been thickened to prevent external shear stress of the material. The edges of both the halves are contoured such that one fits onto another and blocks all the gaps. Certain portions of the left half internal walls are chiseled to support accurate fit for the electronics.

8. As mentioned earlier, provision of foam structures in between load cell and the treatment tip has been arranged to support the M3 screw which forms the load shaft. This arrangement will protect the load shaft from any external shear force which might skew the resultant force magnitudes during treatment. Also, it prevents the wobble effect of the tip and allows qualitative compressive force application during treatment.

External screws and screw walls have been provided to intact and lock both the halves of the device, such that the electrical cavity is protected from water, dust or lotions during treatment. Decisions are being made to provide a rubber coating at the external surface of the device to make it water repellent and waterproof.

The next chapter defines the systems architecture and states the operating modes of the system. It distinguishes the computation processes defined on the device hardware and the PC during different operating modes f the QSTM system. The communication protocol stating the interface between the Q1 treatment device and PC is also elaborated in this chapter.

6. SYSTEM DEVELOPMENT

This chapter defines the operation modes of the QSTM system. The complete system architecture with interaction between the major hardware and software components are elaborated based on the operation modes of the system. This chapter also demonstrates the communication between the major hardware components and the handshaking in between applications of the hardware and software components.

6.1 QSTM Operations

The QSTM system has two major hardware components: A treatment device and a Windows PC. The treatment device does the data acquisition, real-time data processing and sends the processed data to the PC. PC displays the real-time data, does the treatment-level data processing, and displays and records user interested data, treatment parameters and treatment results on the graphical user interface and treatment visualization screens of the PC software. The computation is distributed between both device and PC to sustain real-time system performance and efficiency. The block diagram for the system operation is shown in Fig. 6.1 on page 65, which points out the working modes, the underlying working states, and the activation messages responsible for triggering transition in between working modes and states. The workflow is characterized into two modes, namely – ***Idle Mode*** and ***Treatment Mode***. The idle mode confirms that the system is powered up, but no therapeutic treatment is performed. The treatment mode is where the actual therapeutic treatment takes place. Following subsections provide a functional overview of these two operating modes.

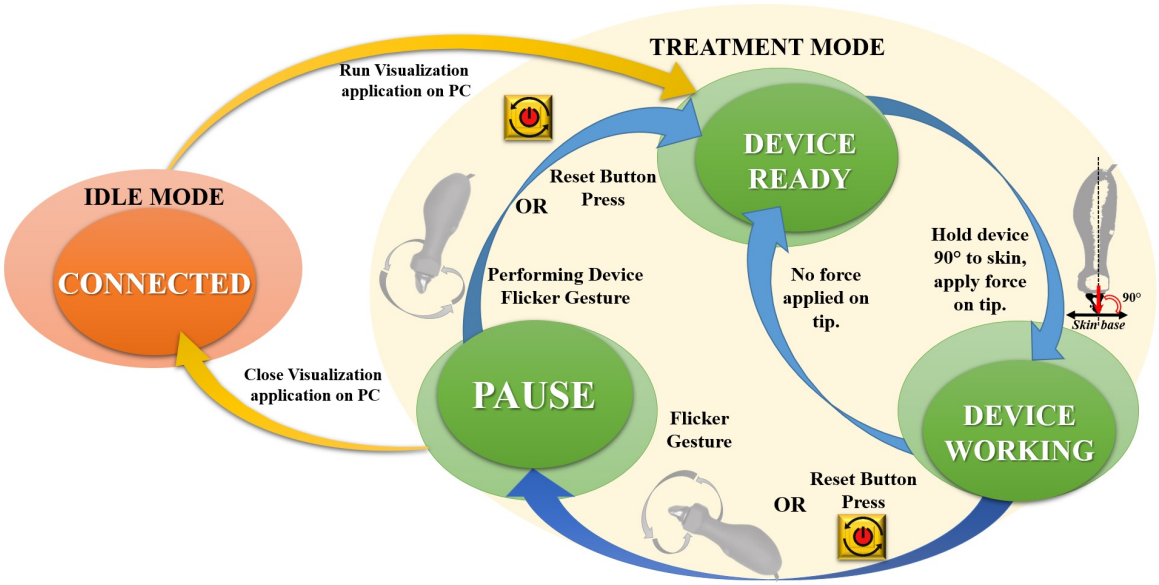


Fig. 6.1.: Operation Modes of QSTM Q1 Device

6.1.1 Idle Mode

The idle mode is the non-functional mode of the system. During this mode the device waits for certain PC messages to perform a series of initial tasks before treatment, while the PC performs certain final operations to record treatment results after treatment. During the idle mode, the treatment device blinks an LED every second to indicate that the system is in the idle mode.

If the device is plugged in to USB port on PC and the PC software starts running, the PC sends a message to the device to initiate the serial communication. If the serial communication is successfully established, the PC software requests the device to provide its device-Id and register the device on the PC with a registration ID. This step allows the PC to distinguish between several QSTM devices if multiple devices are plugged into the same PC.

Once the user is ready to perform therapeutic treatment, the user triggers the start of treatment on the PC software. The PC performs a Device ID Check processes to confirm device registration on PC software. If the device ID check is successful, then the PC acknowledges the device to initiate real-time device-PC communication for treatment mode.

Once the treatment is over the user shuts down the visualization application on PC software. This terminates the treatment mode and the system reverts back to idle mode once again. During this period the PC performs operations for treatment parameters and treatment report storage, computed during treatment mode.

As the idle mode repeats itself both before and after treatment with specific operations on the PC software and treatment device, this mode can be separated into Pre-treatment Idle Mode and Post treatment Idle Mode of the system respectively.

6.1.2 Treatment Mode

In the treatment mode, the device read the sensors, process the sensory data and transmits the processed sensor data to the PC. The transmitted data is extracted and displayed graphically on the monitor of PC. Data transmitted in real-time from the device to the PC in this mode are quantified forces in Newton's and angular device orientation in degrees both based on skin co-ordinates and gravitation co-ordinates systems, respectively. In a therapeutic treatment, the system is characterized by three operation states – *Device Ready*, *Device Working*, and *Device Pause*.

6.1.2.1 Device Ready

This state is where the treatment device is laid in rest or when external resultant forces less than 1 Newton's is acting on the treatment tip of the device. Forces exerted by the device due to the weight of the treatment tip of device is considered negligible

because it is less than 1 Newton's. The resultant force magnitude on the force graph in Q1 Treatment App of PC, reads less than 1 Newton's during this state. The time elapsed during this state accounts to the dead time of the treatment.

6.1.2.2 Device Working

This state can be triggered by exerting any resultant external force above one Newton's on the treatment tip of device. The device is required to be positioned orthogonally to the skin and exert force above 1 Newton's on its treatment tip first, before starting the treatment. This action assists the device to calculate approximate inclination angles from skin during treatment. A real-time visualization window is opened on the monitor to display the force and angle measurements in real time. Additionally, peaks and valleys on force graphs are based on the exerted forces on the treatment tip during the device working state. The device working state defines the active time of the treatment.In both device ready and device working states, the treatment device shows a constant LED glow.

At the end of treatment, the user needs to close the real time data display window of the running visualization application to end this mode. This action closes the serial communication and terminates all real-time operations of treatment mode. Then the system goes back to the Idle mode.

6.1.2.3 Device Pause

This state can be activated any time by a user input, which is either a button press on the treatment device or a flicker gesture of the device. This state has been featured into the system for three important activities which are:

1. Switching positions from one treatment site to another in between sub-sessions.

2. Intermediate analysis in between sub-sessions to maintain clinical reliability and consistency throughout the session. A table widget featured on the software visualization window is updated with treatment analysis data during this interval for self-assessment of clinicians performing the treatment.

3. It also re-initializes treatment parameters to zero to automatically adjust the device for the next treatment sub-session. In the process, the device maintains its performance efficiency and resets memories in the hardware.

In device pause state, the PC neither process incoming data and nor generate graphical output on the visualization screen on PC software. This state is visually marked by the device LED turned off. Time elapsed during this state accounted as reset time which is counted as part of the dead time of the total treatment. Moreover, the system waits in this state until the user resumes it back to the Device Ready state either by performing a button press on the device or a flicker gesture of the device. Specifically, after the reset action is performed the device goes back to the Device Ready state, as it cannot go back to the Device Working state directly.

6.2 QSTM System Architecture

This subsection provides an architectural view in technical terms. The two major hardware components of the system are the QSTM Q1 device and the PC. Communication between the two components uses USB serial communication protocol. Each of these hardware components has complex algorithms to perform tasks of the operating modes while interacting with the user. Fig.6.2 on page 69 shows a simplified architecture diagram of the hardware and the software layers, outlining the operating modes of the system and the applications that communicate between the two hardware components.

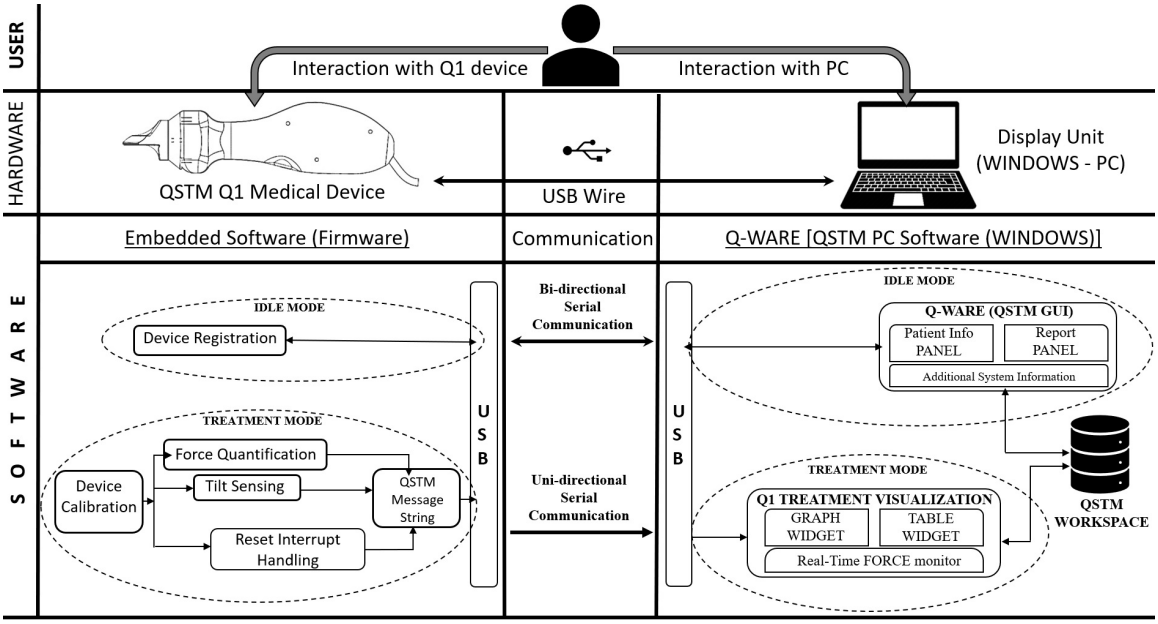


Fig. 6.2.: QSTM system architecture based on operation modes

This section is organized as follows. Section 6.2.1 describes the working of the device in both Idle and treatment mode, with an emphasis on sensor data acquisition, data processing and communication to PC during the treatment mode. Section 6.2.2 explains the operations of the system operations modes, performed on the PC side for data display, data storage and data analysis of treatment sessions.

6.2.1 Device Description

The operating modes of the system is reflected in the operation of the Q1 device. In the idle mode, the device accomplishes device registration step by sending out its device identity to the PC. During the treatment mode, the device performs dedicated process of device calibration, force quantification and tilt sensing from sensor data, and generate QSTM message string to transmit to PC.

The ***Device calibration*** step involves sensor noise range detection and noise offset determination to perform corresponding calculations. This is performed for all sensors 3D Load cell, accelerometer and gyroscope used in this device. This calibration step makes future calculations more accurate for force quantification and tilt sensing.

For ***Force quantification***, the sensor data in electrical voltages are first transformed into actual forces with manufacturer provided calibration matrix. These force data are then filtered and scaled to Newton's unit to generate 3D translational and compressive force and their resultant.

For ***Tilt sensing***, the IMU sensor which includes accelerometer and gyroscope is used to determine orientation angles of the device. Euler transformations are used to generate angles based on accelerometer data, while timestamp-based integration is used to determine the same from gyroscope data. Later both are combined to form complementary filter for integration drift compensation to approximate orientation angles based on gravitational reference frame. The final output generated from tilt sensing yields Yaw, Pitch and Roll angles of the device with respect to gravity. These orientation angles are further manipulated to generate an approximation of device inclination angles based on the skin plane (skin co-ordinate system). This step marks a primary novelty of the research.

The device also checks for user reset input such as reset button press or flicker gesture action.

The force data, the angular orientation data with respect to both gravity coordinates and skin co-ordinates, with linear resultant 3D acceleration of the device, along with the reset data bit collectively forms the QSTM message string. This message string is transmitted to the PC in every iteration (10 ms) of the embedded software.

6.2.2 PC Description

The QSTM™software for PC is named Q-Ware©. This software includes a QSTM GUI for patient enrollment and treatment report storage which can be adapted to QSTM EMR in the future for all QSTM users. During treatment mode the Q-Ware starts device specific visualization application based on the type of device used for treatment. This research will be extended to other types of QSTM device (especially Q2, Q3, etc) development in future. In this research, Q1 device is used for treatment, hence the name of the visualization application for PC is called Q1 treatment App.

During Idle mode of the system, when the device is plugged in and Q-ware is running, Q-ware opens a QSTM GUI, which first initiates serial communication with device, registers device, and then waits for user command to start the treatment mode.

After user input, once the treatment mode is started, the Q-ware opens up Q1 treatment app on PC. This app sends acknowledgment message to start treatment mode in the Q1 device. Then the Q1 device starts transmitting QSTM message string to the Q1 Treatment App on PC. The Q1 Treatment application reads the incoming message string and extract 3D forces and device orientation angles for both gravity co-ordinates and skin co-ordinates. These data are displayed graphically on the Q1 Visualization GUI of the app in real-time and stored as CSV charts for post processing.

After the completion of treatment as a part of post processing, a data analysis QSTM library is used to analyze and evaluate average treatment session parameters. This QSTM library include signal filters and calculation functions to analyze graph chart data and calculate output in desired format accepted by clinicians.

The CSV files stored by the Q1 treatment app uses filter methods and calculation functions to detect force peaks and determine number of strokes applied during massage. This stroke determination algorithm, specific to the QSTM application, marks another novelty of the research.

The upcoming section elaborates on the communication between the Q1 device and PC based on the user interactions. A communication sequence diagram has been introduced to better understand the procedures of user interactions to trigger device operations.

6.3 Interfacing Q1 Device With PC

The USB cable connected to processing unit (ARMv4 Microcontroller-Teensy 3.2) on Q1 device acts as the main hardware communication channel between the device and PC. It is also the primary power source for the QSTM Q1 device. This section describes the methods for generating unique device-id to mitigate incompatibility issues of several similar devices on the PC software (Q-ware). These explanations lead to the methodology to establish serial communication and data exchange between the device and the PC. This is followed by sections describing device registration of the QSTM Idle mode and device-id check method responsible to switch the system from idle mode to treatment mode.

The complete communication sequence diagram [35] with user interaction is shown in Fig. 6.3 on page 73, which clearly distinguishes QSTM operation modes and describes triggering process of all tasks on the PC side and on the device side.

The sequence diagram of Fig. 6.3 on page 73 shows interaction between Device-PC, User-PC and User-Device. The bold arrows in Fig. 6.3 indicates message requests by an application or by the user, while the dashed arrows indicate acknowledgement for the requested service. The icons in the Fig. 6.3 represents embedded software of the device, Q1 Treatment application on PC, QSTM datafiles for patients, QSTM PC Software (Q-Ware) application and the User, respectively from left to right. The dotted lines hanging from the icons are the lifelines of the components represented in the diagram. Finally, the curved arrows represent methods or functions specific to the software of the mentioned components. The enclosed rectangles in red dashed lines represent a specific operation.

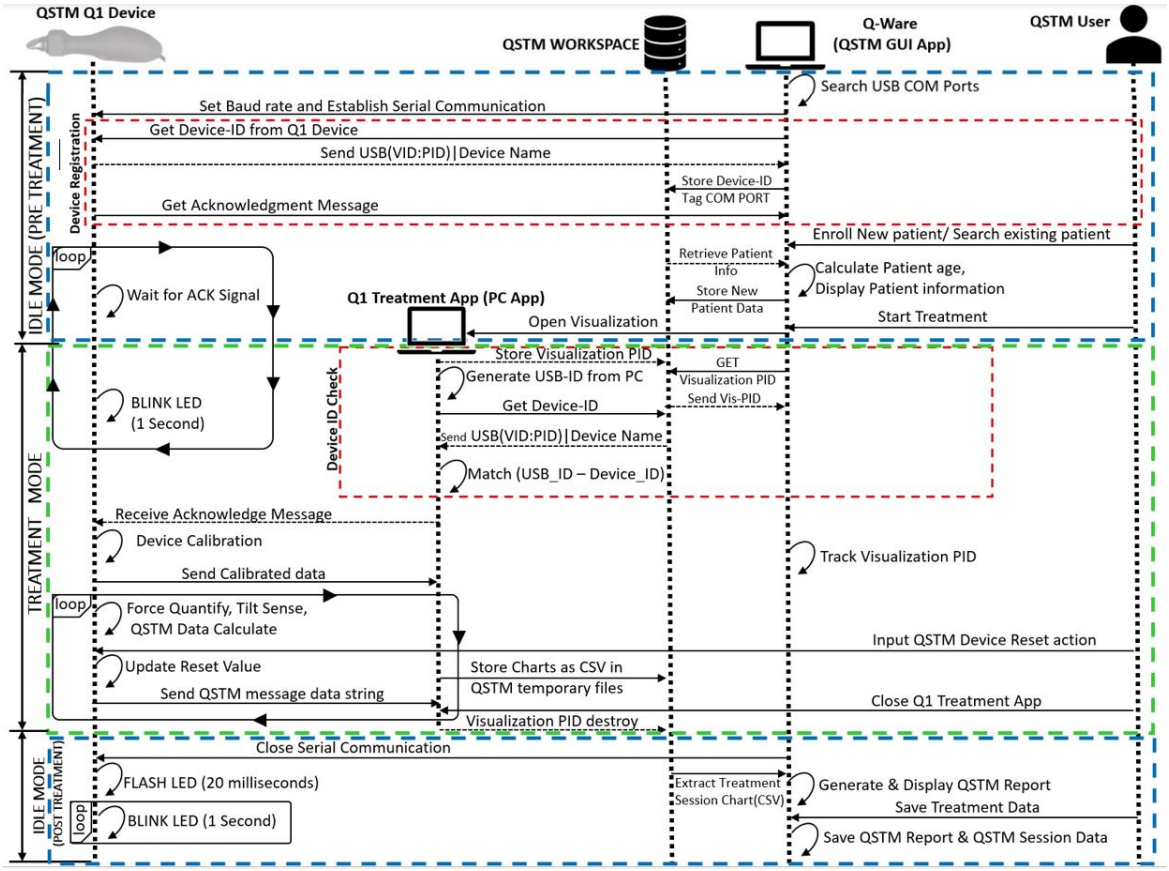


Fig. 6.3.: Communication sequence diagram [35] of Q1 Device − PC interface with user interaction

There are three sequences in the diagram, the sequence in the idle mode (pre-treatment), the sequence in treatment mode and the sequence for idle mode (post-treatment). Following subsections explains all processes mentioned in the sequence diagram with an emphasis on establishment of serial communication, device registration to transmit messages from PC to Q1 device or vice versa.

6.3.1 Serial Communication

This section is divided into two parts, where the first part describes the communication protocol used between the device and the PC, whereas the second part discusses the communication set-up process.

6.3.1.1 Communication Protocol

Communication port (COM-#) is a common hardware port on PC, dedicated for establishing serial communication from host to client device or vice versa. A COM port is assigned by USB standards supported by operating system (WINDOWS) of PC when the Q1 device is plugged to a USB port. For operating system WINDOWS version 8 or above, COM ports are automatically assigned followed by the USB driver installations, as soon as a USB device is plugged in. The USB standard requires a complete software protocol stack to define different layers from physical layer to the application layer analogous to the OSI model. It facilitates a four-wire connectivity in physical layer with two power pins and two differential data transmission pins. The link layer specifies the manufacturer hardware details such as USB Vendor ID and USB Product ID. These hardware identities are unique for a specific USB device. These IDs are harnessed to generate a unique device ID for device registration. Furthermore, the network layer assigns the communication port to establish communication. The transport layer determines the speed of communication over the data bus. A serial data stream is sent in the form of symbols with specified baud rate. Our system specifies a baud rate of 115200 bits per second for synchronized transmission and receive of data bidirectionally. The application layers perform OS specific device driver installation for the type of device being connected to ensure device compatibility.

The processing unit on the device supports UART (universal asynchronous receiver and transmitter) for of data transmission using RS-232 communication standard. But most PCs of today support serial communication through USB-Universal

Serial Bus. Most modern operating systems, like Windows, MacOS, Linux supports USB, uses custom made universal device drivers to read connected devices and assign a hardware communication port to enable serial communication. Hence most AVR or ARM microcontroller development boards, like Teensy 3.2, provide RS-232 to USB converters to facilitate standard bidirectional serial communication through USB. Each USB cable uses a standard 4-wire connection where two are power pins and the other two are responsible for data transfer.

6.3.1.2 Communication Setup Process

Once the QSTM Q1 device is connected to the PC through a USB wire, The USB uses its VCC and GND pins to power up the QSTM Q1 device, while D+ and D- pins are the differential data transmission channels, polled by the PC to initiate communication. PCs Q-Ware application keeps track of the available USB COM ports and establishes a serial connection at the specified baud rate (115,200 bps). This is how the automatic serial connection is made between device and PC.

6.3.2 Idle Mode(Pre−Treatment) Communications

The pre-treatment idle mode of the system performs two exclusive tasks which are establishment of serial communication between PC and treatment device; and device registration of the treatment device on the PC software. The communication sequence for these two tasks are visually represented in Fig. 6.4 on page 76 under the blue colored idle mode (pre-treatment) bounding box.

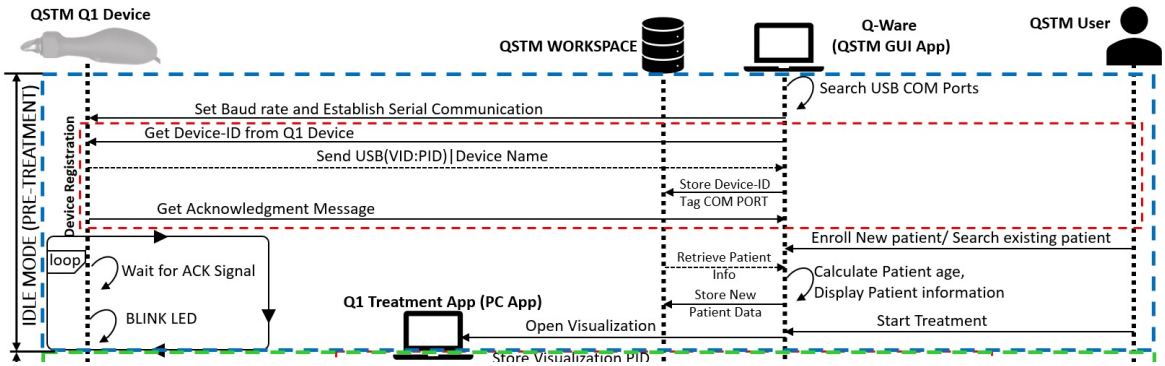


Fig. 6.4.: Communication sequence diagram of PC − Device Interface during Pre-Treatment Idle Mode

6.3.2.1 Establishment Of Serial Communication

Once the PC software searches of connected USB ports, it starts establishing a serial communication with the enlisted connected COM Port. Since the baud rate of the communication sequence is set at 115200 bits per seconds hence, the corresponding connected COM port of the treatment device responds with a successful communication.

6.3.2.2 Device Registration

If the serial connection is successfully established, then it enters the phase of device registration; otherwise, it raises a method for exception handling. The QSTM GUI of Q-Ware on PC sends a request message to get device-id from the Q1 device. At this point, the device retrieves its unique device id, provided by the manufacturer, from its EEPROM. The device-id is a combination of the device name, its USB Vendor id (VID) and USB Product id (PID). These USB hardware IDs are permanent 16-bit numbers universally unique to the device.

If the COM port on the PC side receives an acknowledgement message from the device with the encoded device-id, it decodes it, combine with the COM port number and generates a unique Registration ID of the Q1 device. This registration ID is then

stored into the QSTM Temporary workspace as process information. Unless this step is done successfully, the Q-ware raises an error message for unsuccessful device registration. This device registration step is essential to enable auto-connect facilities of serial communication between the PC and device, for future message transfer. This step completes the process of device registration on the PC side.

On the device side, after sending the device ID to the PC, the Q1 device sends a request to get an acknowledgment message from the PC, for receive of device-id and waits for Q-Ware on PC to acknowledge it. The device enters an infinite loop and blinks its default LED every second until it receives the acknowledgement from the PC. Hence the process of device registration and waiting for acknowledgement message from PC with LED blink operation as shown in Fig. 6.3 on page 73, marks the end of idle mode on the device side.

However, the PC waits for the user to provide patient credentials either by enrolling new patient or by searching for an existing patient from patient list for treatment records. Once the patient information is provided, the PC waits for the user to start the treatment mode. These are the two user dependent tasks of the pre-treatment idle mode for which the PC waits to enter the treatment mode.

The next section describes the communication sequence during the treatment mode of the system.

6.3.3 Treatment Mode Communications

The treatment mode is triggered by the user on Q-Ware of PC once all the patient information subjected to treatment is ready. This mode performs an initial Device ID Check process followed by real-time Serial Communication of transmitting device calibration messages and then QSTM message strings from device to PC.

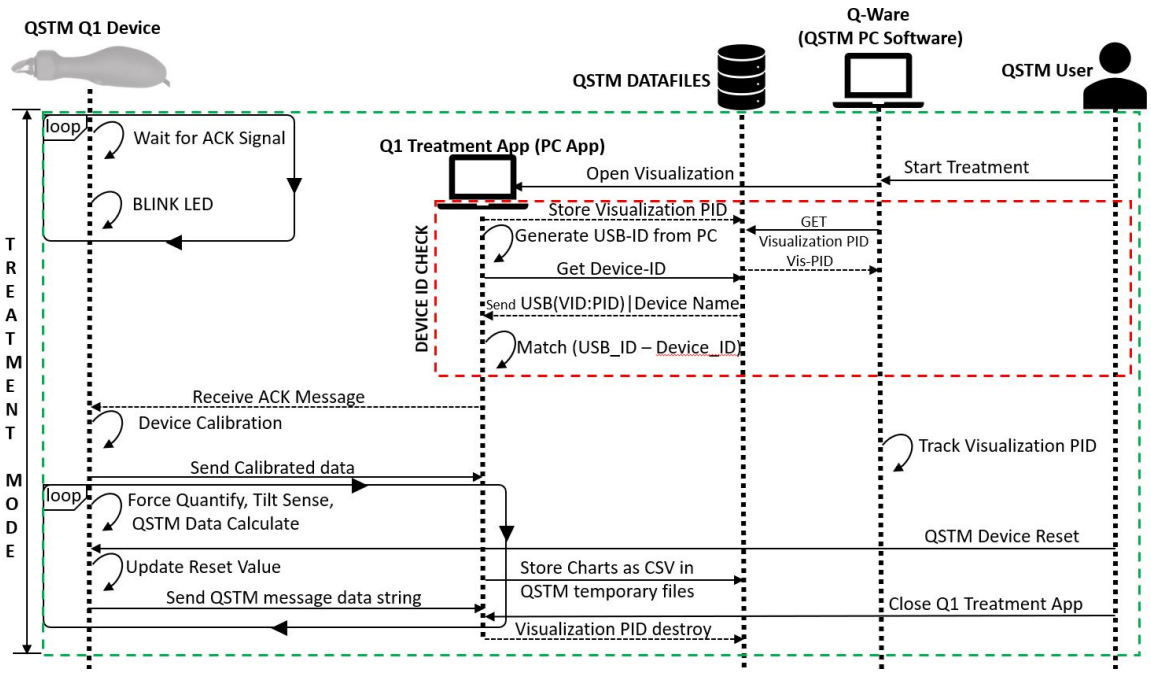


Fig. 6.5.: Communication sequence diagram of PC − Device Interface during Treatment Mode

6.3.3.1 Device ID Check

This communication sequence starts from users request to start treatment mode by clicking a GUI-button (Start Treatment Mode) on QSTM GUI of Q-Ware on PC. Then the Q-Ware starts Q1 Treatment App on PC, which is responsible for real-time data visualization and graphical monitoring. At the start of Q1 treatment application, Q1 treatment application sends its OS generated process ID to the QSTM work space. This assists Q-Ware to track whether Q1 Treatment App is running or not. Then the treatment application grabs the hardware properties of the tagged USB COM port. The hardware properties comprise of Port-type, Port-name, Port product-ID, Port vendor-ID, port serial number, respectively. The Q1 Treatment App separates the 16-bit USB Product-ID and Vendor-ID from the hardware properties and combines it with the device name from the registration-ID. This is how the PC recreates the device-id and matches to the stored registration-id in temporary files in

QSTM workspace. Once the recreated device-id matches the stored registration-id, the QSTM visualization application sends the acknowledgement message to the Q1 device, which the device has been waiting for. This step completes the device-id check process. As the device receives the acknowledgement from PC, it enters the real-time serial communication process with device calibration as the first execution process.

6.3.3.2 Real-Time Serial Communication

As soon as the Device receives an acknowledgement from the PC to start treatment mode on the device, it enters the device calibration phase. During device calibration it reads sensor data to perform sensor noise level detection and various offset calculation to generate QSTM message string. These calibration values are sent to the Q1 treatment application before the start of QSTM message string transmission.

Once the calibration step is complete, the Q1Treatment App constantly receives the QSTM message string from the device, tokenizes and typecasts the quantified force and device tilt data from the string. Then it stores each of these data into distinct fixed sized FIFO buffers(queues). Each element of these buffers is graphically plotted against their corresponding timestamps for real-time graph display on the visualization screen of Q1 treatment App on PC. As the queue overloads over time due to incoming data, the application moves the displayed data points from the queue and stores them into temporary charts in .csv format, as shown in Fig. 6.5 on page 78. The Q-Ware tracks the status of the Q1 treatment app during this period with the help of the visualization process-id. These processes continue indefinitely until the serial communication is interrupted by a user request to close Q1 Treatment App on PC.

If the user chooses to close the treatment app after treatment, then the real-time serial communication is interrupted. This terminates all the running processes on the treatment app and the visualization process-Id of the treatment app is destroyed. This marks the end of treatment mode and the system goes back to the Idle mode (post treatment).

6.3.4 Idle Mode (Post-Treatment) Communications

As the Q1 treatment app shuts down, the treatment mode is terminated. The system goes back to the post treatment idle mode where the PC closes the serial communication with the device. During this mode, the closure of serial communication is acknowledged by the treatment device by flashing the device LED flashes faster for 10 seconds. Then the device resumes its LED blink every second until another treatment mode starts up.

The PC extracts the QSTM data stored in CSV charts for data analysis and calculates the QSTM treatment parameters to yield a QSTM treatment report at the end of treatment. This report is displayed on the QSTM GUI of Q-Ware. If the user chooses to save this report for future treatment reference by clicking the save button on the QSTM GUI, then the Q-Ware saves the treatment report to the destination directories of the treated patient in the QSTM workspace.

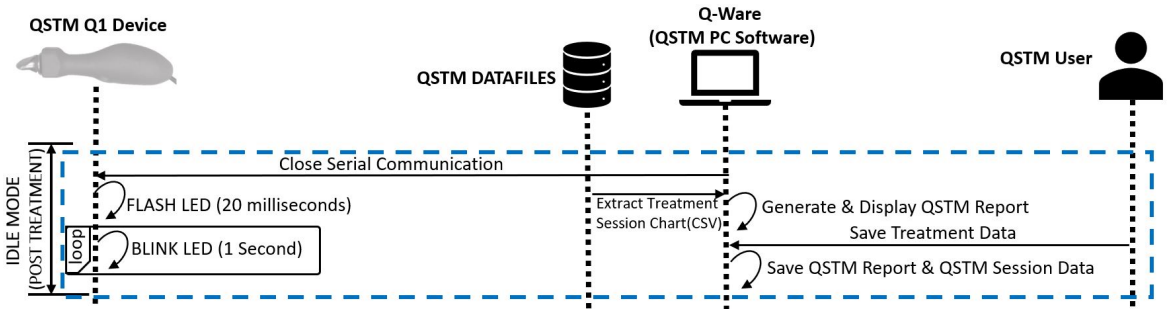


Fig. 6.6.: Communication sequence diagram of PC − Device Interface during Post-Treatment Idle Mode

This marks the end of QSTM operations for one complete treatment. The next chapter discusses the development framework and operations of Q1 device software (firmware) emphasizing the device operations during treatment mode.

7. EMBEDDED SOFTWARE (FIRMWARE)

This chapter describes the workflow of computational logic written in the embedded software on the processing unit of the treatment device for both Idle and Treatment operating modes of the system. The workflow of Idle mode detailing from the initialization processes to start the treatment mode. The treatment mode part explains sensors actuation, sensor calibration, data acquisition and transformations required for force quantification and orientation angle calculations with respect to skin in real-time. The description for the treatment mode also includes user interaction with the device for reset operations and its working principle. All results are placed in the QSTM message payload and transmitted to the PC for display and further analysis. An overview is provided in section 7.1. Furthermore, section 7.2 gives a brief description has been provided about the embedded software development environment. Section 7.3 gives the details about algorithm development in Idle mode and section 7.4 describes the details about algorithm development in the treatment mode on micro-controller Teensy 3.2.

7.1 Device Working Overview

Fig. 7.1 on page 83 represents an algorithmic overview of the embedded program logic distinguished by the operating modes. This figure highlights the prime operations of both modes and their transitions. Serial communication setup and device registration to PC are the main tasks of the idle mode. If the serial connection to the PC is detected, the device will Establish Serial Communication with the PC; otherwise, the device keeps flashing the LED every second. Once the serial communication is established, the device waits for incoming serial messages from the PC. If this message is a request for registration, then it performs the Device registration

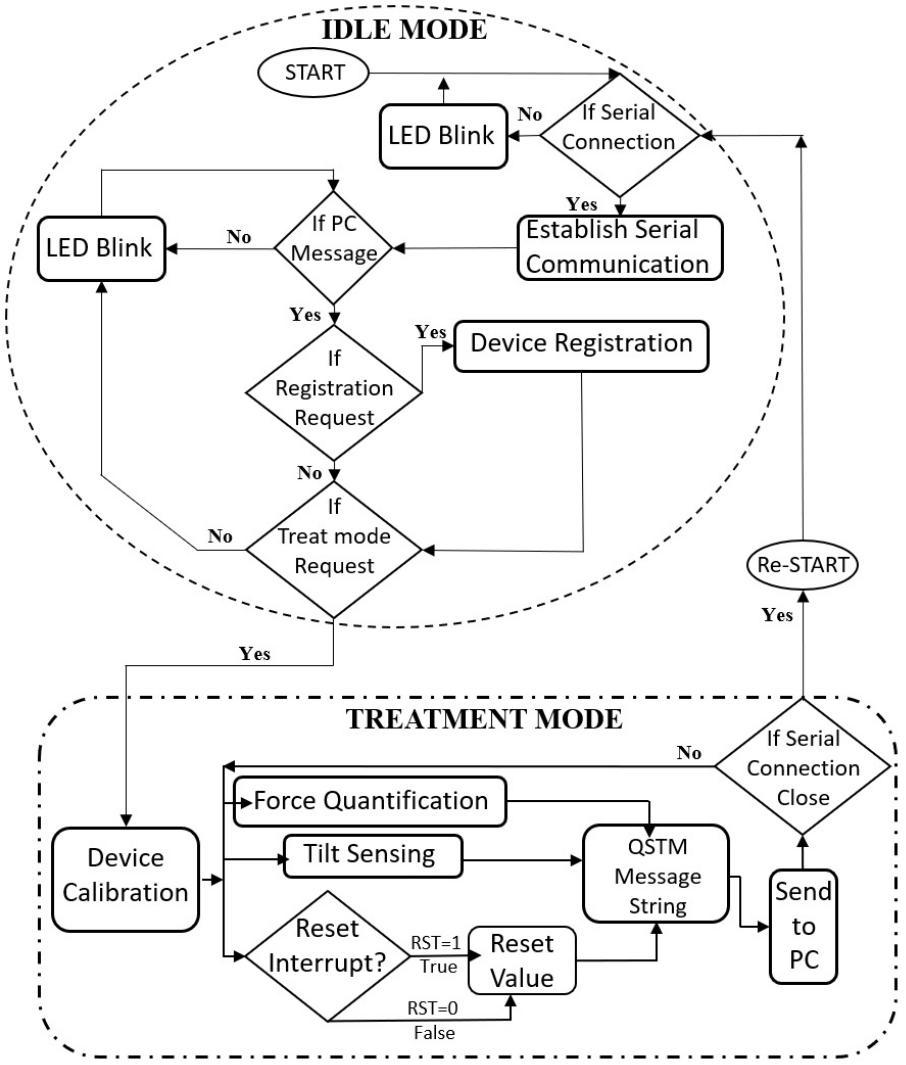


Fig. 7.1.: Flowchart for the complete Embedded Software Overview

step. This step is further described in the IDLE mode section. If the incoming message is a request to start treatment mode then it leaves the idle mode and transits to the treatment mode. All the executions in idle mode with device registration and transition to treatment mode will be explained in detail in the IDLE mode section.

After entering the treatment mode, the device reads the Force and IMU sensors in different pre-determined timing intervals. Then it performs sensor calibrations for all individual sensors 3D load cell and IMU sensor. These calibration data are further processed for force quantification and orientation determination. This calibration process takes approximately 4 seconds to complete.

After the calibration process, the device starts three parallel processes scheduled.

1. Reading Force sensor to perform calculations of force quantification in Newton's.

2. Reading Accelerometer and Gyroscope of IMU to calculate device orientation angles with respect to gravity and then Skin co-ordinates. This process is also called tilt sensing to yield Yaw, pitch and roll measurements of the device with respect to skin plane being treated.

3. Finally, all these calculated parameters are combined into a QSTM message string. This string is sent to the PC for further processing and visual monitoring in real time. The serial communication timer has the lowest sampling rate of 10 milliseconds.

The next section briefly describes the development environment of the embedded software written in the flash memory of the MCU.

7.2 Embedded Development Environment

The microcontroller unit Teensy 3.2, which runs the embedded software, features a 32-bit single core processing unit (ARM Cortex M4) [30] to perform complex computation in real-time with a 64KB primary data memory and 256KB of reprogrammable flash memory. All the instruction sets lies in the flash memory, while runtime variable values are contained in 64KB volatile data memory. The EEPROM of the MCU is also used to store the unique QSTM Device-ID, which essentially contains USB hardware information (Vendor-ID & Product-ID) with device type. Teensy 3.2 also features various hardware peripherals out of which two hardware timers have been

used to characterize and schedule data acquisition and data processing of 3D load cell and IMU for force quantification and device orientation calculations respectively. Another hardware timer has been used for synchronized serial communication with PC visualization using UART to USB converters through a micro USB port. 3 GPIO pins with analog input feature has been used to connect the 3 channels of the load cells. However, GPIO pins supporting two-wire communication with SDA(Serial Data) and SCL(Serial Clock) has been accessed to connect the IMU hardware as shown in Fig. 5.2 on 56 mentioned in Chapter 5. Normal 12 bit analog to digital conversion is adopted through analog pins to acquire data from 3D load cell while a two-wire serial communication I^2C protocol enables registration and data acquisition of the Inertial Measurement Unit. Hardware interrupts from nested vector interrupt controller has been operated to control reset operation and activate serial communication. This completes usage of different hardware peripherals of Teensy 3.2 to develop the embedded application of the system.

The Integrated Development Environment (IDE) of the microcontroller is − ARDUINO IDE version 1.8.1 [36]. The manufacturer of Teensy development boards provides an add-on tool to ARDUINO IDE called TEENSYDUINO [37], which is extensively used to write program in embedded C, compile and upload codes into Teensy development boards. This tool supports all available AVR libraries and compiles them for ARM based micro-controllers. An additional boot-loading tool called TEENSY-LOADER must be used in correspondence with TEENSYDUINO to compile embedded C programs and convert to hex codes forming binary files using a halfkay bootloader.

7.3 Idle Mode Operations

This section explains all the operations performed in embedded software during IDLE mode of the system. Fig. 7.2 on page 86 represents the flowchart for complete algorithm of operations performed in the Idle mode. Once the device is plugged in and

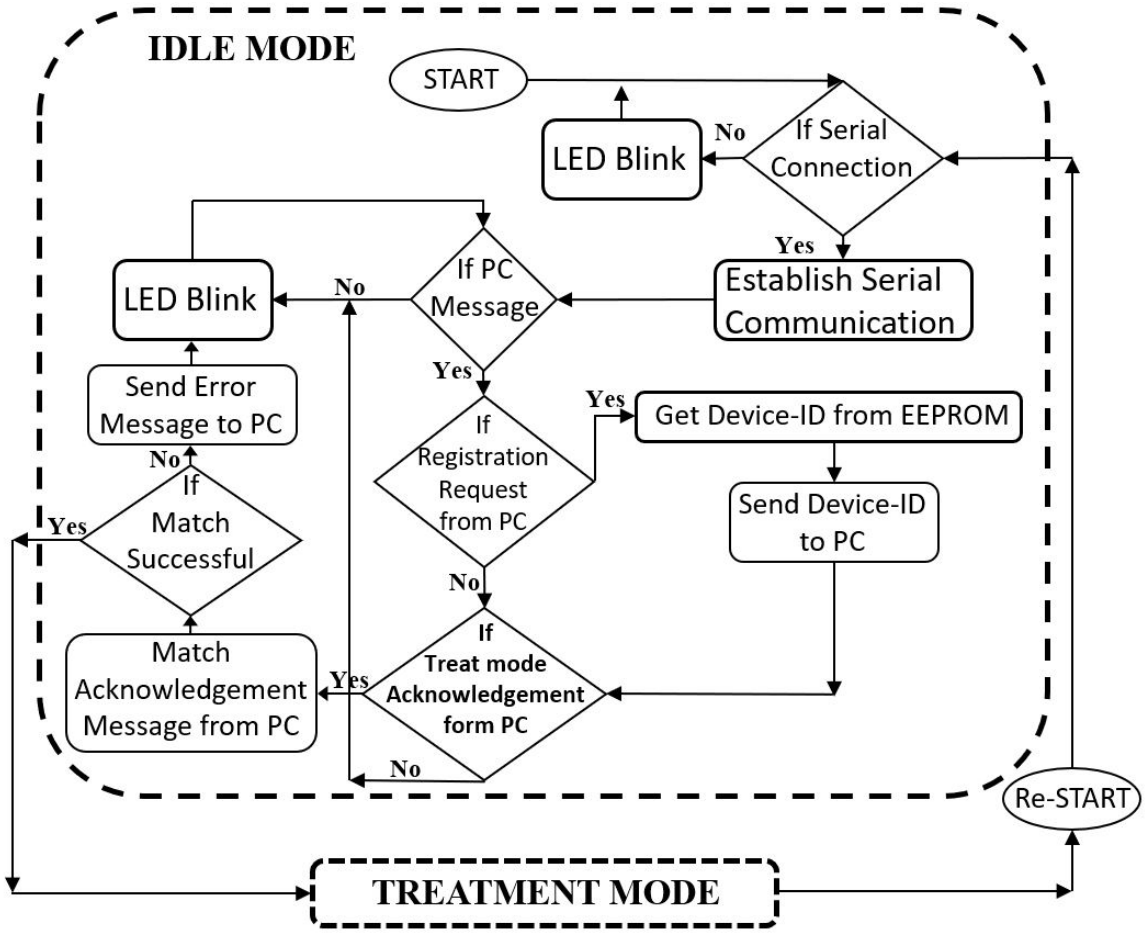


Fig. 7.2.: Flowchart for MCU operations in Idle Mode

powered up, it waits for the PC software (Q-Ware) to start running. As soon as the Q-Ware starts running, the device initiates serial connection at 115,200 baud rate. Once the serial port is connected, it begins serial communication with PC. As mentioned in earlier chapter, this Idle mode is visually marked by a repetitive LED blink function. After serial connection, the device waits for a device registration request from the PC, where the PC software requests the device to provide the device-ID stored in the device memory. The Device-ID is a combination of 16-bit USB Vendor-ID, 16-bit USB Product-ID, and a 16-bit Device type (QSTM Q1). Hence a typical Device-ID is of the format VID:PID:type, for example (16A3:22CF:5131), here 5131-is hex code for ASCII characters Q and 1, where Q1 is the device type . The VID and PID are

written in hexadecimal codes. This unique Device-ID is stored in the EEPROM of the micro-controller Teensy 3.2 by us for this project. Hence, the device calls a method to read the Device-ID and then send it to the PC through UART TX/RX serial port and UART to USB conversion.

7.3.1 Transition To Treatment Mode

After providing the Device ID to the PC, the device waits for the PC to acknowledge the device back with the device-ID. If the acknowledgement message from PC successfully matches with the device-ID then the system transits to the treatment mode; otherwise, the device sends an error message to the PC and keeps waiting PCs message.

The device can only be transited to the treatment mode by request from the PC side. The device terminates the treatment mode whenever the serial communication is terminated. Once the termination of treatment mode is executed, the serial connection is disconnected by the device and the device flashes its LED faster for 10 seconds, closes the serial port and reboots the system. This forces the device to go back to the start of the Idle mode.

7.4 Treatment Mode Operations

The primary task of the device in the treatment mode is to read data from sensors and process them into suitable formats that can depict force and angle information in skin coordinates. Treatment with this treatment device is critically characterized by the amount of mechanical force applied on the tissue, with the instantaneous angular inclination of the device from skin coordinates during the treatment.

Force quantification is the foremost task of the device to calculate the resultant of translational and compressive forces applied on the treatment tip. To ensure the measurement accuracy, the treatment mode calibrates its 3D load cell by reading changes in electrical resistances due to actual application of load on tissue. The re-

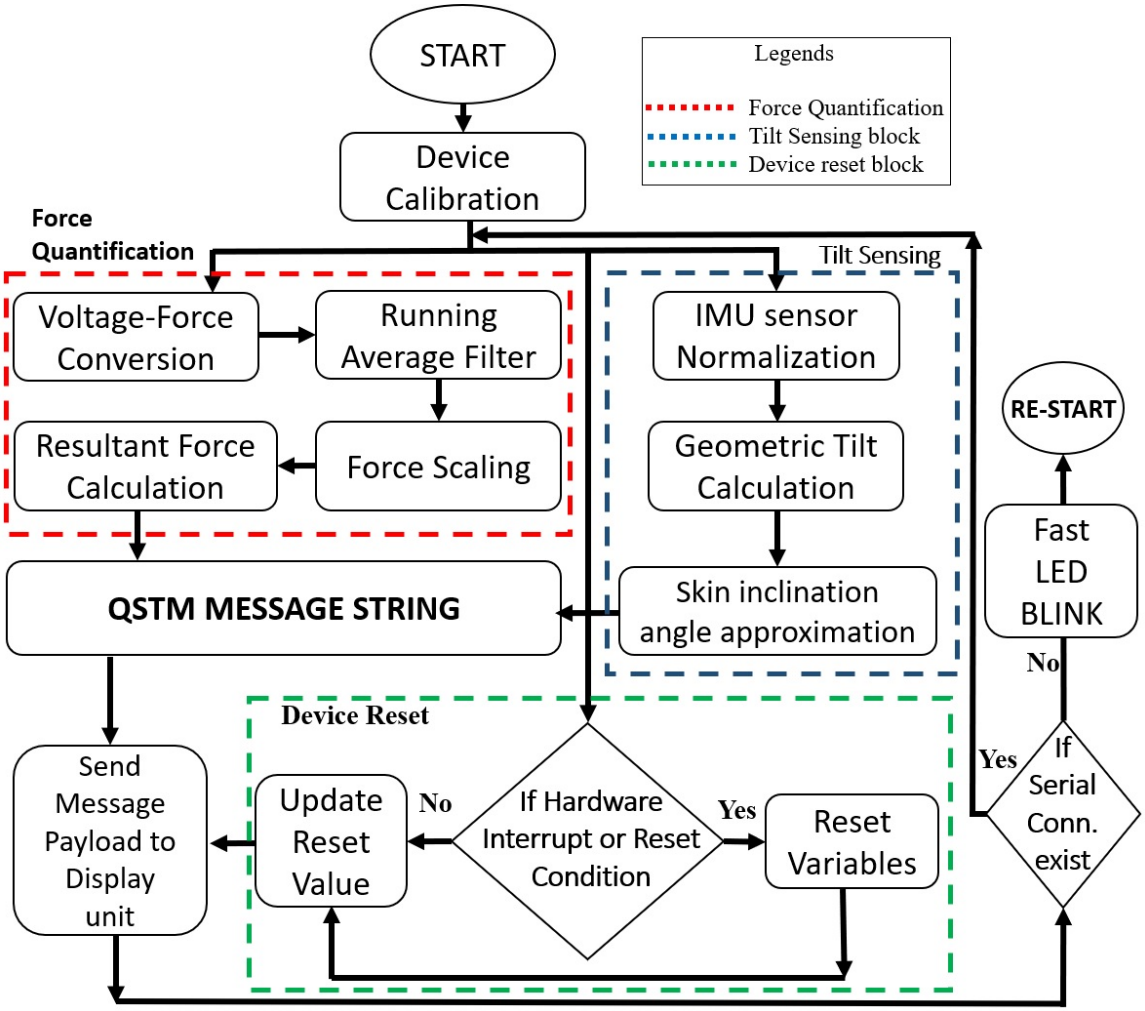


Fig. 7.3.: Treatment mode flowchart for Q1 Device (Firmware)

sistance change is converted into actual forces in Newton's. This mode also adjusts accelerometer and gyroscope readings based on gravitational reference frame to measure angular orientation and inclination of the treatment device from skin during treatment. It uses Eulers rotation transformations from the concepts of rigid body rotations to calculate device angles based on the accelerometer. The gyro values are integrated and then combined with the accelerometer values to get yaw, pitch and roll angles of the device in degrees. These data are then sent to the PC.

The basic operations involved in force quantification and tilt sensing are depicted in Fig. 7.3 on page 88. The user interacts with the device for readjustments in between treatment sessions. These interactions are done by resetting the device by actions like button press or flicker gesture on the device. On completion of the treatment mode, the PC software aborts the serial communication and disconnects the serial connection (terminating Q1 Treatment App on PC). The device loses its serial connections, which marks the end of treatment mode. Once the serial communication is interrupted by the serial port, the device flashes its LED every 50ms for 2 seconds, closes the serial port and runs an interrupt subroutine to reboot the system. The device restarts and resumes IDLE mode all over again.

The rest of this chapter focuses on describing the detailed methodology of different operations performed by the device application during the treatment mode starting with force quantification by 3D load cell and tilt sensing by IMU sensor. Before that, an elaborate illustration about the sensor adjustments to sustain device accuracy and measurement precision is made by the device calibration step, which is equally important.

7.4.1 Device Calibration

This section describes the most significant and vital step in the firmware logic, without which proper functioning of the device would be difficult. This calibration step includes the I^2C communication, sensor registrations, setting up timers for sensor reads, sampling time determination, noise range detection, voltage to force conversion, and offset calculation for every input channel of all sensors used in the device. The IMU uses I^2C communication for sensor registration and data stream transmission to the microcontroller in real-time. The 3D load cell is connected to the analog pins, which uses 12-bit analog to digital converters to quantize a range of 0-3.3V to a digital range of 0-$(2^{12}-1)$. A thorough description of the calibration step of each sensor is elaborated below.

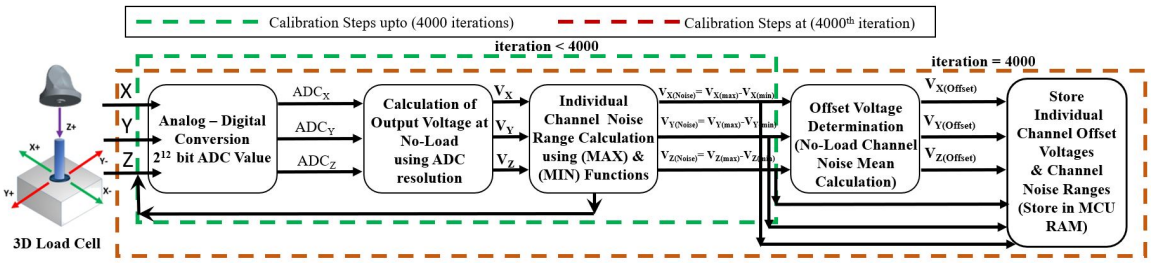


Fig. 7.4.: Block diagram of Load Cell Calibration steps showing offset voltage calculation

7.4.1.1 Calibration Of 3D load Cell

This process involves 5 steps which include timer set up for ADC reads, conversion of ADC values to measured voltages, offset voltage determination for every channel at no load condition and noise range detection of every force channel at no load.

No-load Condition specifies the state when the device is laying at rest on a flat surface and no external force is being applied on the tip (except its own weight due to gravity). This section also includes methods to calculate scaled force noises in Newton's based on voltage to force transformation of detected channel noise ranges in volts.

The green block in Fig. 7.4 on page 90 describes the first 3999 successive iterations in the calibration phase of the load cell for noise monitoring. The brown block marks the 4000^{th} iteration of force quantification which is the last iteration of load cell calibration and the monitored data is further processed to achieve load cell calibration parameters like individual channel voltage noise range, channel offset voltages, converted force noise levels in Newton's and load cell resultant force noise levels. Fig. 7.4 shows the voltage noise range detection and offset voltage calculation.

I. **ADC Readings**

During the device calibration, it is required to keep the device at rest state and laying on a flat surface with no force being exerted on the tip. The operating voltage of ARM MCU chip MK20D is 3.3V, but from manufacturer's data sheet

the I/O pins of Teensy 3.2 board are both 3.3V and 5V tolerant. The board is incapable of reading 5V tolerant input voltage at its analog pins, so a maximum input voltage of 3.3V was used for analog input. Therefore, it was needed to scale down the force sensor output from 0-5V to 0-3.3V range using a voltage divider circuit. In this case, the 12bit ADC reads will vary from 0 4095 for input of 0-3.3V. This reference voltage can be altered to any reference up to 5V DC by specific instruction sets. However, the load cell is powered up by a 5V DC power source from the USB micro connector as shown in Fig. 5.2 on 56 mentioned in Chapter 5.

Standalone electrical testing using digital multimeters and oscilloscopes proves that the measured electrical voltages at no load condition varies approximately 40-50 percent of the set analog input GPIO reference voltage (which is 3.3V). These measurements almost aligns with the manufacturer's data sheet of the 3D Load cell. The intricate details about this will be mentioned later in Chapter 9 (Results Chapter) describing electrical test results. These measured multimeter readings mark the no-load voltage of every output channel of the 3D load cell. The force quantification reads ADC values of 3D Load cell input channels at every 100 microseconds.

II. **Calculation Of Output Voltage At No-Load Condition**

With sampling period being 100 ms, the MCU reads 10 thousand samples per second. These ADC reads are converted to its measured electrical voltage by the formula given below.

$$Analog\,Voltage_{Calculated} = \frac{Analog\,GPIO\,Reference\,Voltage}{ADC\,Resolution} \times ADC\,Reading \tag{7.1}$$

Here, the maximum allowed MCU analog input is 3.3V, while ADC resolution is 12-bit i.e. 4096 levels. The converted analog voltage readings from ADC should match the measured voltage value using the multimeter or the oscilloscope. The

calculated analog voltages of individual channels of the load cell are recorded and compared to actually measured voltage readings for validation before hardware assembly.

III. **Individual Channel Noise Range Calculation**

The calculated voltages at no-load condition retain high frequency noise from the preamplifier circuit embedded in the load cell. In the first 4000 sensor read iterations, this noise is monitored and the upper and lower limits of the noise are detected to determine the noise range. However, the fluctuations observed in calculated voltages for successive iterations are considered to be noise in no load condition. These voltage fluctuations are calculated using the *max()* and *min()* functions.

$$VoltageNoiseRange_{Channel} = AnalogVoltage_{Maximum} - AnalogVoltage_{Minimum} \tag{7.2}$$

The difference between the upper and lower limit of the calculated voltage fluctuations for the first 4000 iterations in the calibration phase marks the detected noise range at the no-load condition of the 3D load cell. The maximum analog voltage calculated forms the noise upper limit, while the minimum analog voltage marks the noise lower limit.

IV. **Offset Voltage Determination**

Once the noise range is calculated, the offset voltage at no load condition is measured using the formula:

$$Offset\,Voltage_{Channel} = \frac{Analog\,Voltage_{Maximum} + Analog\,Voltage_{Minimum}}{2} \tag{7.3}$$

This offset voltage calculation takes place at the 4000^{th} iteration marked by a brown colored box shown in Fig. 7.4 on page 90. The calculated offset voltage is considered as the reference zero at no-load condition. Subtracting this offset voltage from the calculated voltage readings during treatment would result either in positive or negative values. Calculated voltage readings above

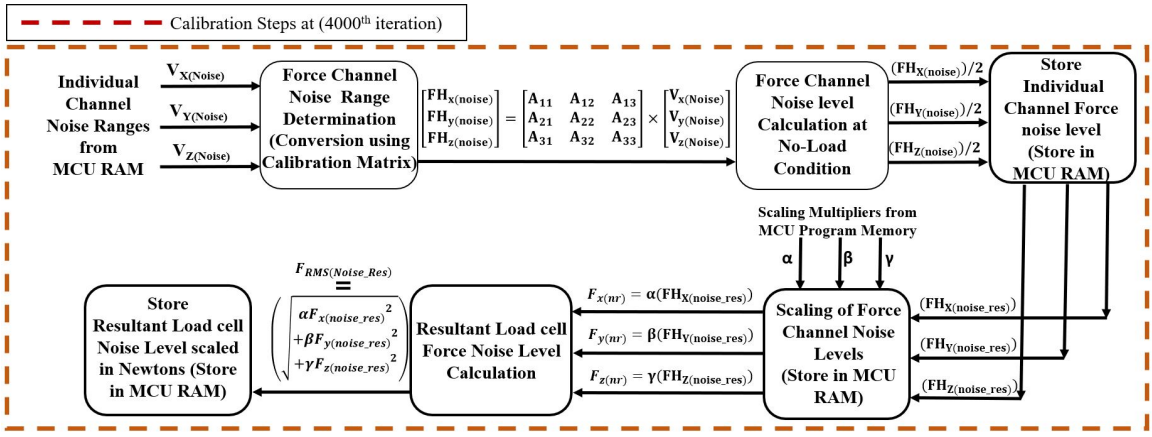


Fig. 7.5.: Block diagram of Load Cell Calibration steps showing Load cell noise level calculation

or below the offset voltage is considered to be actual voltages of measured forces during load conditions. The term ***Load Conditions*** specify the state when actual external forces are being applied on the tip of the device.

V. **Store Calibration Data**

In the end, these offset voltages along with the detected voltage noise ranges of each channel are stored into the primary memory of the MCU for further calculation. Fig. 7.5 on page 93 describes all steps in calculation of the overall resultant force noise level of the 3D load cell. All these calculations are performed at the 4000^{th} iteration of the force quantification timer. Hence, these calculations are denoted inside a brown colored box in Fig. 7.5.

VI. **Force Channel Noise Range Determination**

The voltage difference between the calculated offset voltage and instantaneous voltage ($V_i - V_{i(offset)}$), where i indicates individual channels (X or Y or Z) of load cell, during no-Load condition of the load cell calibration phase, adds up to the channel voltage noise readings. Individual channel voltage noise ranges are first converted into their actual force noise range by applying linear transfor-

mations of individual channel voltage noise ranges with respect to a calibration matrix provided by the Load Cell Manufacturer.

$$\begin{bmatrix} FH_{X(noise)Upper-limit} \\ FH_{Y(noise)Upper-limit} \\ FH_{Z(noise)Upper-limit} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \times \begin{bmatrix} V_X(max) \\ V_Y(max) \\ V_Z(max) \end{bmatrix} \tag{7.4}$$

$$\begin{bmatrix} FH_{X(noise)Lower-limit} \\ FH_{Y(noise)Lower-limit} \\ FH_{Z(noise)Lower-limit} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \times \begin{bmatrix} V_X(min) \\ V_Y(min) \\ V_Z(min) \end{bmatrix} \tag{7.5}$$

Based on these two equations, 7.4 and 7.5, where A_{ij} represents the elements of calibration matrix provided by the Load Cell Manufacturer to calculate force channel noise upper and lower limit, the equation for the individual Force channel noise ranges has been derived.

$$\begin{bmatrix} FH_{X(noise)} \\ FH_{Y(noise)} \\ FH_{Z(noise)} \end{bmatrix} = \begin{bmatrix} FH_{X(noise)Upper-limit} \\ FH_{Y(noise)Upper-limit} \\ FH_{Z(noise)Upper-limit} \end{bmatrix} - \begin{bmatrix} FH_{X(noise)Lower-limit} \\ FH_{Y(noise)Lower-limit} \\ FH_{Z(noise)Lower-limit} \end{bmatrix} \tag{7.6}$$

$$\begin{bmatrix} FH_X(noise) \\ FH_Y(noise) \\ FH_Z(noise) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \times \begin{bmatrix} V_X(max) - V_X(min) \\ V_Y(max) - V_Y(min) \\ V_Z(max) - V_Z(min) \end{bmatrix} \tag{7.7}$$

VII. **Force Channel Noise Level Calculation**

These force noise ranges are used to calculate individual force channel noise levels, which is defined as half of the calculated force channel noise ranges, i.e. ($F_{i(noise_{res})} = \frac{FH_{i(noise)}}{2}$), where i is the individual force channel, as shown in Fig. 7.5 on page 93. The force channel noise ranges has force channel noise upper limit and lower limit. The upper limit would essentially be in the positive force values while the lower limit would be in the negative force values for X and Y force channels. Hence, half of the noise range for every channel forms the noise level for both positive and negative directions of forces in X and Y. Since the

Z channel has no negative direction hence the negative force noise values are discarded. All these corresponding steps are performed at the 4000^{th} iteration of the hardware timer designated for load cell calibration and force quantification.

VIII. **Store Individual Channel Force Noise Levels**

The calculated Force channel noise levels for both positive and negative directions of all the channels are stored in the primary memory of the processing unit Teensy 3.2 for further calculation steps.

IX. **Scaling Of Force Channel Noise Levels**

The force measurements during load-condition in device working state of treatment mode needs a force validation step, which is performed by the embedded software developer of this research during electrical testing and assembly of the device. This validation step will be further discussed in detail with different test results later in Chapter 9 (Results Chapter). The main idea to perform this validation step is to match the calculated force readings during treatment to measured readings in Newton's on a standard force plate. This is performed by manually pressing the steel tip against the Newton's force plate sustaining a particular amount of force to match the Newton's scale. The proportional gains are determined experimentally by manually positioning the treatment tip on the force plate in orientations specific to which the individual force components dominate the resultant force.

Future scaling methods sees the necessity of specific test rigs to orient and point steel tip for validating Newton's for different force magnitudes against a standard Newton's force plate. This testing helps to experimentally yield constant proportional gain or scaling multipliers $\alpha, \beta and \gamma$ for respective X, Y and Z force channels of load cell to further scale calculated forces into original Newton's. These scaling multipliers are hardcoded on the program EEPROM memory of MCU by the developer after testing, for further use in the treatment mode during device calibration and force quantification phases.

The scaling multipliers determined during electrical testing phase are fetched from the EEPROM and multiplied with the calculated channel force noise levels. This is performed such that these noise levels are adjusted or tuned according to a standard Newton's scale used to validate quantified forces of the load cell. This step is significant for calculation of the resultant load cell force noise levels.

X. **Resultant Load Cell Force Noise Level Calculation**

The root mean square of force channel noise levels, scaled in last step, constitutes Load Cell resultant force noise level in Newton's.

$$F_{RMS(Noise)} = \sqrt{\alpha FH_{X(noise)} + \beta FH_{Y(noise)} + \gamma FH_{Z(noise)}} \tag{7.8}$$

The calculation of Load Cell resultant force noise level is only executed at the 4000^{th} iteration of the force quantification. This completes the load cell calibration phase and marks the Device ready state of the treatment mode where the device performs force quantification at no-load condition and waits for the user to start treatment.

As soon as the device registers forces greater than 1N it changes its state from Device ready state to Device Working state. The treatment mode switches between these two states back and forth during actual treatment based on application and release of force on the device tool tip during actual treatment.

XI. **Store Resultant Force Noise Level Of Load Cell**

The resultant force noise level once calculated is stored in the MCU primary memory RAM for further use during the treatment mode.

7.4.1.2 Calibration Of IMU Sensor

The complexity of IMU calibration is higher as compared to that of the 3D load cell. The major steps involved in IMU sensor calibration are:

1. Establishment of I^2C communication at 400 KHz to read sensor output.

2. IMU Channel self-tests for both accelerometers and gyroscopes to confirm acceleration and gyration trims from factory settings.

3. Calculation of acceleration and gyration bias values to adjust measuring resolutions. Magnetometer offset determination to adjust sensitivity.

4. Periodic sampling to read accelerometer and gyroscope, to manage discrete time sampling for avoiding aliasing or under-sampling.

5. Complementary filter initialization to measure instantaneous orientation of the device in the form of Yaw (rotation about Z axis), Pitch (Rotation about X axis) and Roll (Rotation about Y axis).

Our system uses an open-source software library (9 DOF IMU sensor) from Sparkfun for the respective IMU sensor MPU-9250, which does most of the calibration steps of offset and bias calculations.

This library is structured in object-oriented programming principle, which includes a header file with *.h* extension containing namespaces, register hex-code values from the register map of the datasheet, class-names, public and private variables used inside the library. Additionally, it also comprises of a *.cpp* which includes methods of different class attributes, along with calling communication libraries for I^2C and SPI protocols.

I. **Establishment Of I^2C Communication**

   At first the controller calls the WIRE library to initiate a master slave I^2C communication. The method which calls this communication function is $readByte()$. This method initializes the serial communication between the sensor and the processing unit at 400KHz data rate, and returns the data stored in the corresponding register of the IMU being read.

   For both the accelerometer and the gyroscope, it specifies a "WHO AM I" register. If the register value is for accelerometer, the raw data stored from the

accelerometer buffer is read and returned to the processing unit. Whereas, if it is for the gyroscope it reads gyroscope raw data and returns the value to the processing unit in I^2C message bits.

II. **IMU Channel Self-Test**

Once the I^2C communication is established, then "WHO AM I" registers are confirmed for reading. The processing unit requests for an overall self-test of both the accelerometer and the gyroscope to determine deviation of raw data of every channel from its actual factory settings. Hence it calls a function with the name $ImuSelfTest()$ to perform the same. These deviation from the factory settings marks the sensitivity or adjustment factor of the sensor at the rest-state.

III. **Calculation of Acceleration and Gyration Biases**

The function named $CalibrateIMU()$ calculates the channel offsets, initially set to zero, for both the accelerometer and the gyroscope as input, and updates them by calculating the average of at-rest readings of raw data for around 1 second. Then loads the resulting offsets into accelerometer and gyro bias registers. These offsets are further used to adjust sensitivity based on sensitivity resolutions of factory settings during actual tilt sensing phase to read and convert raw data into acceleration measurements with respect to milli-g (g=9.8 m/s^2) and that of gyration data into degrees per second. This step completes the calibration of the IMU sensor. The next steps are the initialization steps to start the active tilt sensing.

IV. **Periodic Sampling**

The timer interrupt for the IMU sensor is assigned after the calibration offsets of the IMU has been determined. The method with the name $IMUinit()$ has the essential instructions to read raw channel data based on desired scales from respective sensor registers of the IMU and write them into specific buffers for conversion. However, $readAccelData()$, $readGyroData()$ are functions which reads data while $getARES()$ for accelerometer and $getGRES()$ for gyroscope

data are used for unit conversion. The timer interrupt to read the channel data is attached to the interrupt subroutine $IMUinit()$. The IMU timer interrupt is set to read both accelerometer data and the gyroscope data at a sampling frequency of 1 MHz i.e. 1 microseconds per sample. The IMU serial communication is 400Khz. However, the sampling frequency is purposely kept high such that the processing unit does not incur any data loss from incoming sensor data due to task execution of machine cycles and result in under sampling.

These sensor reads are further fed into three soft timers. The first is for tilt sensing due to accelerometer tilt, while the second is for position estimation by time integration of gyro degrees and the last is initializing a P-I controller for complimentary filtering to compensate drift due to gyro integration.

The first software timer for accelerometer-based tilt sensing updates after every 500 microseconds. The second software timer corresponds to the hardware timing interrupt of sampling time 1 microseconds. Thus, the gyro readings in degrees are integrated for 500 microseconds at a sampling rate of 1 microseconds for 500 samples to estimate the angular position during treatment. These two angular readings are further fed to a P-I controller based complimentary filter which operates at a sampling rate of 1 milli second. Since the complimentary filter depends on the calculated inputs of both the tilt angles determined by accelerometer and gyroscope, hence its sampling time is kept higher. This completes the IMU sensor fusion scheduling.

V. **Complimentary Filter Initialization** The IMU sensor library also includes methods for yaw, pitch and roll calculation, which did not prove to be much suitable for our application. Different models to calculate rotation and orientation angles has been tested. The methods of complimentary filtering of Euler angles from Accelerometer data and time integrated gyro positions proved to be much effective to yield yaw, pitch and roll angles.

The complementary filter is basically a proportional-integral controller with constants Kp and Ki values, complementary to one another. The fundamental idea to implement a complimentary filter is to combine slow moving signals from the accelerometer-based tilt sensing with high moving signals of integration of gyro-based tilt sensing to compensate drift due to gyroscope integration. Hence the complimentary filter introduces a low pass filter for accelerometer-based readings while it feeds the gyro readings to a high pass filter. Here, Kp which is supposed to be the proportional constant for accelerometer-based tilt sensing is determined experimentally to be a lower value. While, Ki is the integral constant for gyro-based readings which is the complimentary value of proportional constant Kp. The values of both Ki and Kp experimentally determined during device testing and are stored in the MCU program, which is used later during actual treatment in treatment mode. The complimentary filter updates its values every 1 millisecond.

7.4.2 Force Quantification

Dose-pressure quantification is the clinical term for estimation of a massage dose delivered to a treatment spot over the session duration. Since QSTM Q1 is the medical device which treats a tissue site covering a smaller skin area approximately in centimeters square, hence it is called a localized pressure applicator. In this study, we dont have a method to estimate the amount of skin area covered during treatment, hence we confine our treatment in terms of force quantification over time on the basis of mechanical load applied on the skin through the treatment tip.

All the initialization steps are explained in the device calibration section. So, this section would include all the calculation steps for force quantification which involves real-time processing to determine final resultant force and distinguish *Device Ready State* and *Device Working State* of Treatment mode. Device Ready state is defined by the no-load condition, while Device Working state indicates when pressure is exerted

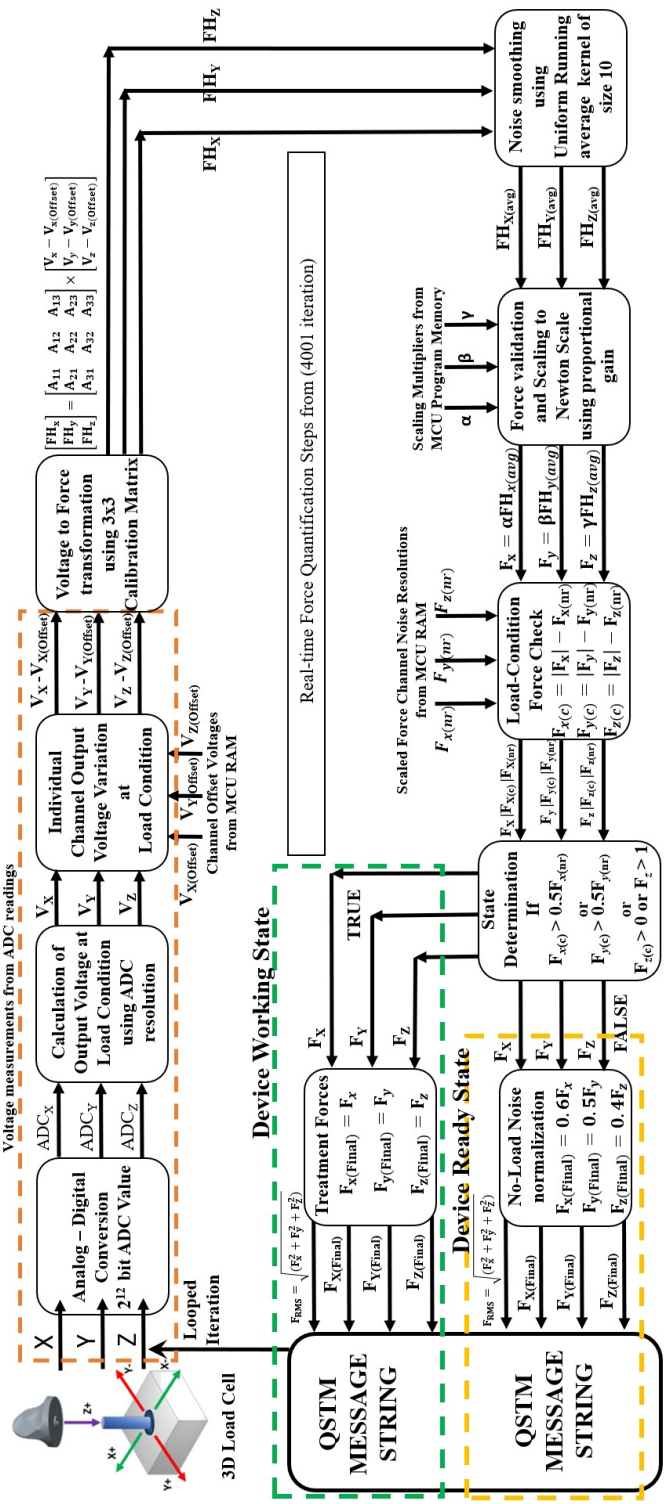


Fig. 7.6.: Complete block diagram for load cell force quantification during the treatment mode

on the treatment tip during an ongoing treatment. These states switch between one another during an actual treatment. During the device ready state the quantified force of every channel should remain within the channel noise level, with the resultant force below the Load cell resultant noise level.

Fig. 7.6 on page 101 describes the block diagram for force quantification in the treatment mode after the calibration of the load cell. This quantification of force continues for successive iterations from 4001^{th} iteration of the force quantification timer.

7.4.2.1 Voltage Measurements From ADC Readings

The first 3 blocks in Fig. 7.6 in page 101 determines ADC reads from sensor, conversion to corresponding voltages and then calculation of voltage variation during load conditions.

I. **ADC Readings**

The 12-bit analog to digital converter in the MCU quantizes the analog reference voltage into 4096 levels and based on that the ADC reads the channel output voltages of the load cell. This is shown in the first block.

II. **Calculation Of Output Voltage**

The second block shows these ADC reads are converted to their respective voltage readings using 7.1 mentioned in Load Cell calibration section.

III. **Individual Channel Output Voltage Variation**

The calculated voltage reads are subtracted from their corresponding channel offset voltages, as shown in third block. These differences between calculated voltages from their respective Channel offset voltages determine the linear increase or decrease of applied load during an actual treatment. Since the load cell uses strain gauge-based load measurements along its central load shaft, the X and Y channels of the load cell measures magnitudes of force vectors along

positive and negative X and Y axes (see Fig. 7.7 in page 103). These forces account to the translational forces acting along the plane of the skin. Hence, positive values add to force application in the positive X and Y directions while negative ones add to forces in negative X and Y directions during load condition. However, the Z vector measures compressive push forces acting orthogonal into the plane of the skin. Since, there is no measurement in the negative Z direction, i.e. a pull force on the tip, we discard all the negative noise voltage values below Z-channel offset voltage and zero it out. However, the positive values form the compressive forces acting into the plane of the skin during load conditions. The root mean square force of all three directions yield the resultant force acting on the tissue during treatment.

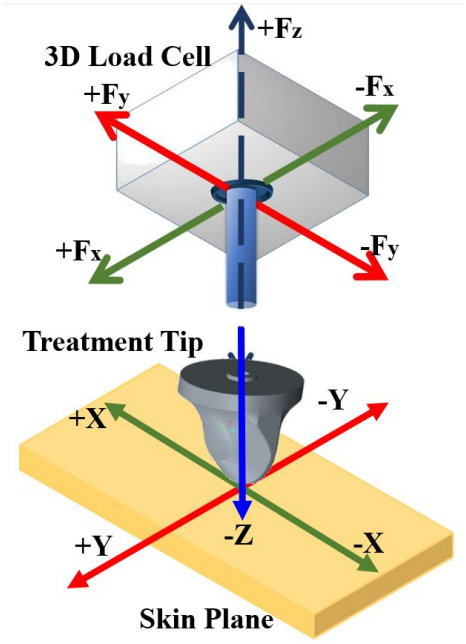


Fig. 7.7.: Figure showing the X-Y(Translational) and Z (Compressive) Forces applied on the skin plane during treatment

7.4.2.2 Voltage To Force Transformation

This section describes the necessary methods and computation models required to transform measured voltage into actual forces in Newton's. Since the offset voltage at zero balance no load condition is calculated at the calibration step, it is used in successive iterations to calculate the corresponding force components FHx, FHy, FHz in Newton's. This conversion is performed by multiplying the differences of offset voltages from their corresponding with-load voltages by a 3×3 coefficient matrix A_{ij} provided by the manufacturer in the calibration sheet of the force sensor as mentioned below. This 3×3 coefficient matrix is determined by the sensor manufacturer for each individual sensor. Following is an example.

$$\begin{bmatrix} FH_X \\ FH_Y \\ FH_Z \end{bmatrix} = \begin{bmatrix} 25.5468 & -1.3139 & -0.1773 \\ -0.0929 & 25.3307 & 0.6875 \\ 0.1754 & 0.7355 & 49.8882 \end{bmatrix} \times \begin{bmatrix} V_X(load) - V_X(offset) \\ V_Y(load) - V_Y(offset) \\ V_Z(load) - V_Z(offset) \end{bmatrix} \tag{7.9}$$

Naked eye observations show that calibration matrix provided by the manufacturer contains significant multipliers along the diagonal elements of the matrix. This concludes that the difference of offset voltages from its with-load voltages are mostly amplified by multipliers, calibrated to a scale used by the manufacturers. This multiplication also amplifies the channel noise levels and hence the noise range at no-load condition increases. The next step constitutes the smoothing of these noise values.

7.4.2.3 Noise Smoothing

Forces above the resultant force noise level of the load cell, measured in calibration phase, rises linearly with the linear application of force at the treatment tip. Fluctuations of the resultant force reads, due to noise, can be observed by monitoring the graph of particular sustained force magnitudes (5N) over a certain amount of time. These fluctuations are due to high frequency noise that could be eliminated by using

a low pass filter with a cut off frequency. Commonly used low pass filters like higher order Butterworth filter or Bessel filter could have been used as options to perform this. However, the main challenge lies in determination of the cut off frequency.

Although this device is a targeted force applicator for localized area, still observations from clinical trials might result in uneven or rippled force peaks due to irregularities on tissue or friction on skin. These irregularities need to be accounted for treatment. Cutting off noise due to high frequency might remove these high frequency ripples due to underlying tissue irregularities and result in data loss.

Our research uses time series approximation-based smoothing over frequency domain filters. Time series approximation-based filters uses statistical distribution function as smoothing kernels to diminish noise uniformly mimicking the curve of the distribution function. Hence, uniformity in smoothing is maintained retaining the patterns of irregularities during massage. The smoothing filter used in the embedded software is a running average filter of size n, where $(n-1)$ past data values are summed with the present data value to calculate the arithmetic mean of n inputs.

$$FH_{X(Avg)} = \frac{\left(\sum_{i=1}^{n-1} \left(FH_X \right)_i \right) + FH_{X(present)}}{n} \tag{7.10}$$

$$FH_{Y(Avg)} = \frac{\left(\sum_{i=1}^{n-1} \left(FH_Y \right)_i \right) + FH_{Y(present)}}{n} \tag{7.11}$$

$$FH_{Z(Avg)} = \frac{\left(\sum_{i=1}^{n-1} \left(FH_Z \right)_i \right) + FH_{Z(present)}}{n} \tag{7.12}$$

It uses a queue or filter window, as known in signal processing, of size $n = 10$. The present value is inserted at the end of the queue while the value contained in the first index is popped out in every iteration to achieve a running average output. The filter averages ten samples of individual channel force values in every iteration as

output shown in Equations 7.10, 7.11, 7.12. Here uniform and equal weights of "one" are used, as the window function follows a uniform distribution. Therefore the more the window size the more is the smoothing.

These smoothed values are then subjected to a scaling step where the smoothed force magnitudes are multiplied by scaling multipliers or proportional gains to fit to the actual Newton's range.

7.4.2.4 Force Validation And Scaling

The force components smoothed by the running average filter is further multiplied with scaling multipliers or proportional gains (α, β and γ) determined during electrical testing and calibration of force sensor. This step is essentially done to scale the smoothed forces to the actual Newton's range of the calibration force plate. The following equation is used to yield scaled forces after smoothing.

$$F_{Channel} = Proportional\,Gain_{Channel} \times FH_{Channel(avg)} \tag{7.13}$$

7.4.2.5 Treatment State Determination

The scaled forces need to go through a conditional decision step to determine whether the device is in ready state or the device is in working state. Hence to perform this, there are two successive steps that has been included: **Load Condition Force Check** and **State Determination**.

1. **Load Condition Force Check**

   Here the difference between the smoothed forces from their individual channel force noise levels are generated. The X and Y channels have negative forces magnitudes due to forces exerted on the negative direction during load condition. Hence the channel force noise levels of X, Y and Z channels are subtracted from their absolute scaled values to yield a ***Force Check Value***. If this force check value of each and every channel is negative or equal to zero, then it con-

firms that the quantified forces are within the channel force noise level and no external force is applied on the tip. Whereas if this force check value is greater than zero then it proves that external forces are being registered by the force sensor.

2. **State Determination**

   A conditional checks are implied to avoid unusual spikes in noise and malfunctioning of the device. These conditions are stated as:

   (a) if the force check value exceeds 50% of the noise level of the X and Y channels.

   (b) if the Z channel reads more than 1Newton's, or if the force check value of the Z channel exceeds zero.

   If the device satisfies any of these two conditions then the device is in working state. Otherwise, all quantified forces are within their channel force noise levels, no load is being applied to the treatment tip and the Device is ready to use, which confirms the Device Ready State.

Treatment Forces – In the Treatment mode, the scaled forces constitute the Final quantified forces in the QSTM message string, which is transmitted to PC. However, the quantified forces in Device ready state are further diminished to lower values by noise diminishing. This is done to avoid random spikes due to weight of the treatment tip suspended on the load cell load shaft. This step prevents malfunctioning of the device and provides convenience for further processing for graphical analysis on the PC.

7.4.2.6 Noise Diminishing In Device Ready State

The noise diminishing step is the last calculation step for the device ready state of the treatment mode, before which the final force values in Newton's of each channel are combined into the QSTM message string and send to the PC.

This step reduces channel noise of the sensor during the Device Ready State at no load condition. Most observations show that the force noise levels of each channel determined in the calibration stage is below 1 Newton's. The no load quantified force magnitudes during Device Ready state are reduced to an amount less than 0.5 Newton's by respective scalar factors. These scalars used to diminish real-time force channel noise, reduce the quantified force noise readings by 40% for X channel, 50% for Y Channel and 60% for Z channel. Hence the resultant force noise is almost reduced by 50% at the device ready state. These noise elements are not trimmed to zero purposely to observe nose patterns on the PC side Q1 treatment app in real-time during device ready state. There is a need to modify this step in a future version with normalization equations to stabilize or eliminate noise completely during electrical testing before device assembly at the manufacturer end. The other significance of this step lies in the fact that these noise elements help in graphical analysis of stored Force Charts during post processing by the PC described in Chapter 8.

However, the steel tip, suspended by the load shaft, itself weighs around 0.11lbs which accounts to almost 0.5N. Moreover, the overall weight of the device is 0.22 lbs which makes around 1N force just at no load. Thus, our research needs precise test rigs or force calibration setup to hold and sustain forces less than 1N to be measured and modelled accurately. Therefore, it is quite difficult to measure forces ranging between 0 to 1N. Henceforth, it is always important to mention that the lowest measurement resolution of the device is restricted from 0.8N to 1N.

The next section contains the explanation for calculating the orientation angles of the device first with respect to gravity and then with respect to skin reference frames. These orientation angles are the orientation angles of Yaw, pitch and roll of the device calculated for rotations based on the cartesian co-ordinate system.

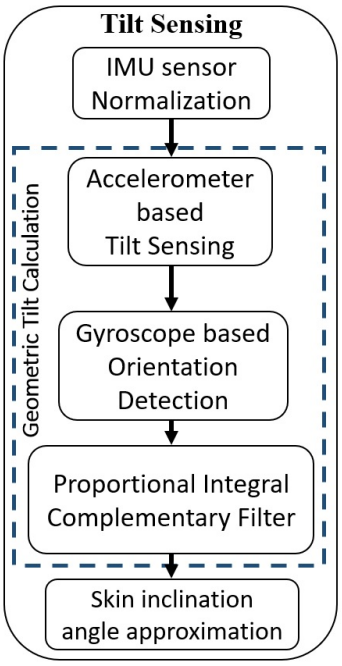


Fig. 7.8.: IMU Tilt Sensing Block Diagram

7.4.3 Tilt Sensing

This section describes the methodology to calculate the orientation angles from 9 degrees of freedom, inertial measurement unit (IMU). The IMU contains a 3D accelerometer to capture linear acceleration in 3 axes, a 3D gyroscope to capture rotations about the Cartesian coordinate and a 3D magnetometer to capture heading directions with respect to earth's magnetic field. The axes X, Y and Z of both the accelerometer and gyroscope are aligned along the width, length and height of the prototype, as the sensor sits exactly at the horizontal cross section of the prototype. During Treatment mode, the accelerometer is set to a maximum swing of $2g$ (where $1g$ is $9.8m/s^2$). The sensor outputs in the form of milli-g and reads 1000 milli-g at its Z axis, when kept at rest on a flat surface aligned with the horizon.

7.4.3.1 IMU Sensor Normalization

After the calibration step of the IMU sensor, the accelerometer reads tri-axial linear acceleration with a function $getARES()$. The acceleration due to gravity vector acting on the plane which is normal to the gravity is registered as $1g$ or $1000mg$, where g is 9.8 m/s^2. Hence on tilting the prototype the acceleration due to gravity acting on the plane increases or decreases. This variation in acceleration due to gravity enables to determine the tilt angles using accelerometer reads for the three axes accelerometers.

The unit vector for each axes of acceleration is calculated during the normalization step as shown in Fig. 7.8 on page 109. Hence $Accel_X = \frac{A_X}{\sqrt{A_X^2+A_Y^2+A_Z^2}}$, $Accel_Y = \frac{A_Y}{\sqrt{A_X^2+A_Y^2+A_Z^2}}$, $Accel_Z = \frac{A_Z}{\sqrt{A_X^2+A_Y^2+A_Z^2}}$ are calculated as a normalization measure before actual calculations of the tilt angles. These acceleration values are further used for calculations of the accelerometer based geometric tilt sensing.

7.4.3.2 Accelerometer Based Tilt Sensing

Tilt sensing for each of the individual axis of the accelerometer can be measured by calculating tilt from a reference position. This reference position is considered to be the position of the prototype at rest, where the X and Y axes of the sensor lies along the horizontal plane [38] and the gravity vector acts along the Z axis. This position is also the initial calibration position of the sensor when the device is powered up. At the calibration position, both the X and Y axes reads $0g$ while the Z axes reads approximately $1g$ of acceleration due to gravity. Rotations about the X,Y and Z axes are calculated from three dimensional rotation matrices using the Euler transformation theory [39] on xyz reference frame using the basic trigonometric formula [38] as mentioned in Eqn. 9.2, Eqn. 7.15 , Eqn. 7.16 below.

$$Rot_X(\theta_A) = \tan^{-1}\left[\frac{Accel_X}{\sqrt{Accel_Y^2 + Accel_Z^2}}\right] \times \pi \tag{7.14}$$

$$Rot_X(\psi_A) = \tan^{-1}\left[\frac{Accel_Y}{\sqrt{Accel_X^2 + Accel_Z^2}}\right] \times \pi \tag{7.15}$$

$$Rot_X(\phi_A) = \tan^{-1}\left[\frac{\sqrt{Accel_X^2 + Accel_Y^2}}{Accel_Z}\right] \times \pi \tag{7.16}$$

These trigonometric equations enable the calculation of the geometric tilt angles with respect to gravity. The rotation along the X and Y axes produces angles in the range $-90°$ to $90°$ degrees while that in the Z axis is from $0°$ to $180°$. Hence the rotation scale of the Z axis is mapped with a scale of $-90°$ to $90°$ with 0 as $-90°$ and 180 as $90°$. The rotation along the X axis determines the Pitch angle i.e. (θ), Rotation along the Y axis determines the Roll angle i.e. (ψ), while Rotation along the Z axis determines the Yaw angle i.e. (ϕ).

It is observed that if the sensor rotates along an axis that is aligned with the gravity vector acting on it, the accelerometer fails to detect any change in rotation. It is difficult to detect tilt changes in the X-Y plane of the accelerometer when the device is rotated along the Z axis at its reference orientation mentioned above. Moreover, the angular variations cease to produce full swing deflection along an axis in the accelerometer as the axis tends to align with the axis of the gravity. This marks a singularity [39] in the Euler transformation approach which is more confined to reference frame.

Thus, it is necessary to use angular position measured by the gyroscope as described in next section.

7.4.3.3 Gyroscope Based Orientation Detection

The gyroscope measures the angular rotation of the sensor with respect to time in all three axes X, Y and Z. The 3-axis gyroscope used in the research has a maximum resolution to measure up to $\pm 2000°/s$. In actual implementation, the gyro registers are activated to read the angular velocity not more than $\pm 1000°/s$. So, the easiest approach to determine instantaneous angular position of the device is to integrate gyro readings with respect to time.

$$Rot_X(\theta_G)(t) = Rot_X(\theta_G)(t-1) + (Gyro_X \times \Delta t) \tag{7.17}$$

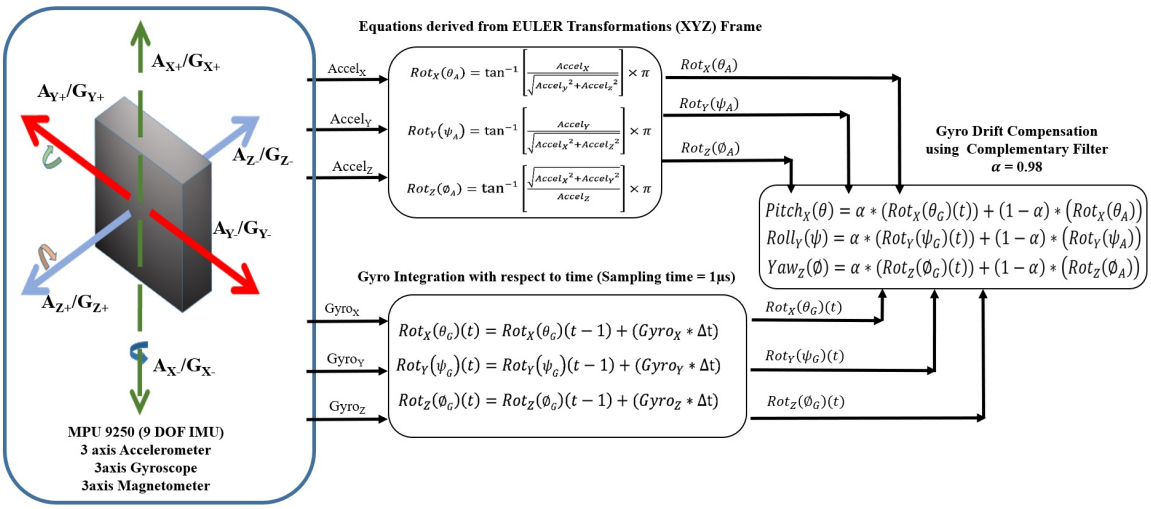


Fig. 7.9.: Real-time device orientation angle calculation model from IMU sensor

$$Rot_Y(\psi_G)(t) = Rot_Y(\psi_G)(t-1) + (Gyro_Y \times \Delta t) \tag{7.18}$$

$$Rot_Z(\phi_G)(t) = Rot_Z(\phi_G)(t-1) + (Gyro_Z \times \Delta t) \tag{7.19}$$

Here $Rot_x(\theta_G)(t)$ which is a function of time, uses its previous value to integrate the present normalized gyro reading $Gyro_x$ over a sample time Δt. From processing point of view, the gyro readings are actually sampled at 1 microseconds, to generate and capture instantaneous angular position. But longer the time elapsed, the larger the integrated error [40] component of the gyro which skews the data from its original position. Thus, there is an observed tendency of positional drift due to gyro measurement noise over time. The position does not return to zero when the device returns to its initial position. Therefore, it is recommended to implement an extended KALMAN filter or a non-linear complementary filter [41] to avoid such errors in the long run and improve device precision.

7.4.3.4 Proportional Integral Complimentary Filter

An attempt to implement sensor fusion of tilt data from accelerometer and positional data from gyro with a constant proportional gain α is analogous to implementation of linear complimentary filter, similar to a P-I Controller using the following formulae:

$$Pitch_x(\theta) = \Big(\alpha \times \big(Rot_X(\theta_G)(t)\big)\Big) + \Big((1-\alpha) \times Rot_X(\theta_A)\Big) \tag{7.20}$$

$$Roll_y(\psi) = \Big(\alpha \times \big(Rot_X(\psi_G)(t)\big)\Big) + \Big((1-\alpha) \times Rot_X(\psi_A)\Big) \tag{7.21}$$

$$Yaw_z(\phi) = \Big(\alpha \times \big(Rot_X(\phi_G)(t)\big)\Big) + \Big((1-\phi) \times Rot_X(\phi_A)\Big) \tag{7.22}$$

The proportional gain α is taken to be 0.98 to amplify the effect of the gyro and minimize the effect of tilt altered due to accelerometer. However, the linearity of the complimentary filter for the gyro drift compensation significantly reduces the effect of integration of gyro noise but does not completely eliminate the skewness of the of the orientation angles. Observations of complimentary filters shows that a slow alteration in angular position over time suffers significant drift while a fast change is unable to be captured. So, a more accurate model with a combination of scalar and vector- quaternion representation [42] system should be used to estimate orientation angles. Since quaternion representations are more convenient in terms of computation and are not affected by reference frame singularities, this approach generates better results. An additional method for drift compensation can be use of anti-windup filters. Moreover, angular tilt measured due to acceleration can be harnessed to make orientation corrections by measuring the drift error and the acceleration tilt. The quaternion and anti-windup filters methods are some of the future observations to be made to correct measurements of orientation angles.

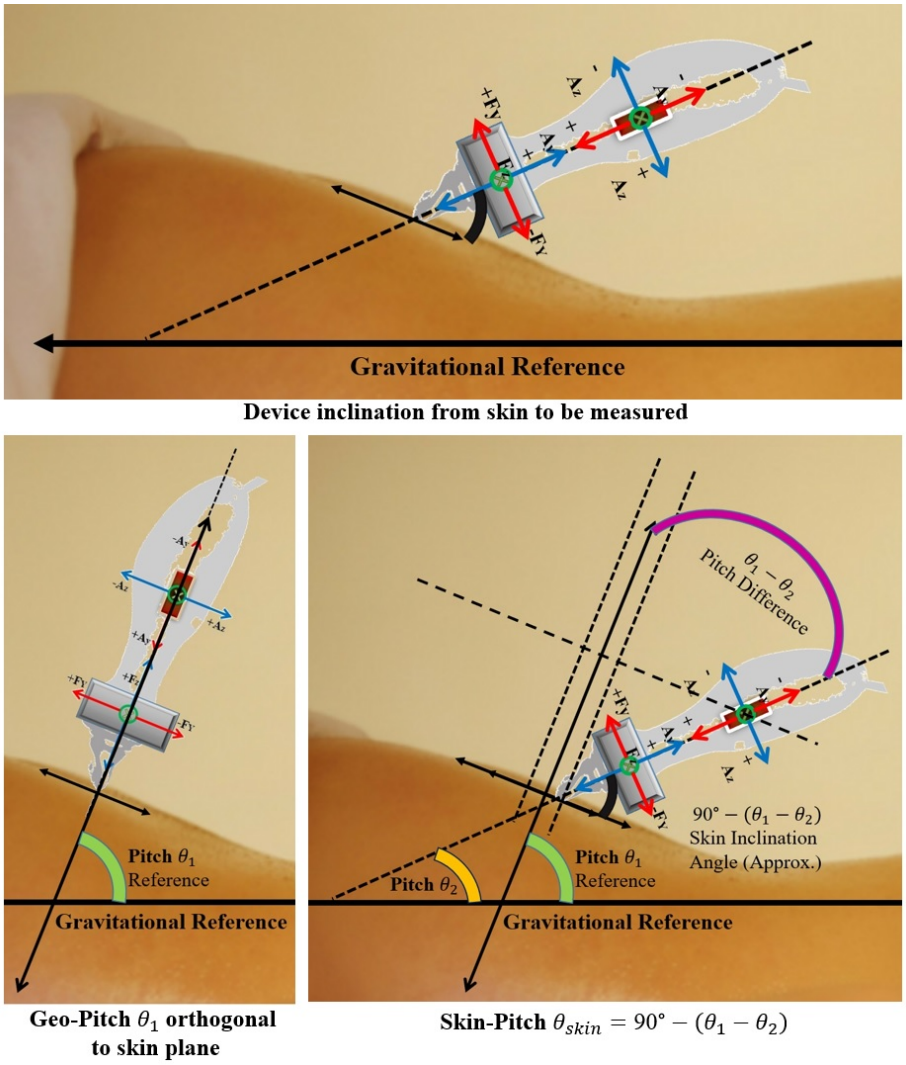


Fig. 7.10.: Approximation to measure device inclination angle from skin

7.4.3.5 Skin Angle Approximation

The QSTM Q1 device consists of a 9 DOF IMU sensor which calculates the instantaneous geometric Yaw, Pitch and Roll angles using the computation model explained in Fig. 7.9 on page 112, as mentioned above. This model measures angles restricted to the gravitational reference frame, as shown in Fig. 7.10 on page 114. This section explains an extended approach to approximate device inclination angle from skin at any arbitrary skin plane alignment.

To perform device inclination angle approximation from skin at any arbitrary orientation, the clinician is recommended to hold the device at the treatment site approximately orthogonal to the skin surface and start the active time by registering minimum force above 1Newton's. The geometric orientation at this instance marks the reference angle (pitch). When the status changes to Device Working, all geometric deviations (change in pitch) are calculated with respect to the marked reference orientation angle (reference pitch). So, the complementary angles of the angular deviation (pitch difference) from the reference angles (reference pitch) forms the approximation for inclination angle with respect to skin. This approach can be used to do the same for the Yaw angles. However, the Roll angles are due to rotation of the device along the central axis of the device aligned to that of load shaft hence, it is independent of the inclination from the skin plane.

7.4.4 Device Reset

During the treatment mode in between a treatment session when the clinician aims at changing positions of the device to start treating a new soft tissue area, the device needs to reset and the skin angles needs to be initialized to zero. Our research incorporates two different approaches to execute this device reset implementation.

7.4.4.1 Gesture Approach

In this approach the Device needs to be shaken abruptly along the vertical axis which aligns with the X axis of the IMU sensor. By shaking abruptly (i.e. Flicker Motion) there is a rapid change in the Gyro readings along the X axis. The operating condition to implement this functionality is to set an offset value in the difference between the present and past values of the gyro X reads. If the difference meets the threshold then the reset is done and the angles with respect to skin are set to zero. All readings since this time stamp are not considered in the session. This time difference from the end of one session to the start of another session is called as the dead time.

The reset gesture to be performed is a whipping motion which is facilitated in such a way that two successive motion sequences need to be performed for terminating a session and recalibrating the device respectively. On the first motion it starts the dead time and shows the results of all recorded QSTM parameters in the GUI, while on the second motion it recalibrates the devices and sets all the parameters to its initial state or default values.

7.4.4.2 Button Approach

A reset button which is also provided to execute the same functionality. A digital two pin toggle switch has been chosen to facilitate the reset function in the device, which is used to differentiate between an active time of session and the dead time. The hardware circuit to implement this is button is very simple, as it requires a pull-down resistor and a short circuit connection to the GPIO pin to read its ON or OFF state in the micro controller. As mentioned above, two successive gestures are needed for successful reset of the device. Similarly, in the button case − two successive button clicks are necessary to execute the same functionality, the first terminating the active session while the second setting the parameters to its default state.

7.4.5 QSTM Message String

The raw QSTM parameters which needs to be transmitted from the treatment device to the PC for visualization constitutes the QSTM Message String. These parameters which constitute the message string are:

1. ***3D Forces***− The final Force magnitudes in Newton's calculated as a result of the force quantification process acting in all the 3 Dimensions i.e. X, Y directions aligned with the direction of the skin plane and the Z direction, which acts orthogonal to the skin plane, during the treatment mode.

2. ***Resultant Force***– The root mean square of all the 3-dimensional force components acting on the skin during treatment mode. This resultant force in Newton's provides the amount of resultant mechanical load applied to the tissue at an instant of time.

3. ***Geo-Orientation Angles Of The Device***– These are the orientation angles i.e. Yaw, Pitch and Roll angles with respect to the gravitational reference frame calculated as a result of tilt sensing process. The unit of these orientation angles are in degrees.

4. ***Skin-Orientation Angles Of The Device***– The orientation angles of the device i.e. Yaw, Pitch and Roll angles calculated with respect to its inclination from the skin plane. The angles are derived from the Geo-Orientation angles.

5. ***RMS Acceleration***– The Root mean square of the linear acceleration in m/s^2 of the 3-dimensional acceleration of the device in instantaneous time.

6. ***Reset Bit***– This is quite significant with respect to the treatment mode because, the reset bit determines, whether the system is in device pause state or Device Ready/Working state. If the reset bit is high, i.e. 1, then the system represents Device Pause state, or conversely Device Ready/Working state.

Thus, the QSTM message string combines all the above-mentioned parameters into a string length of approximately 100 bytes and sends it to the PC in every 10 milliseconds. The sequential order of the QSTM Message String is Force X value, Force Y value, Force Z value, Force Resultant Value; followed by Geo-Pitch, Geo-Roll, Geo-Yaw; followed by Skin-Pitch, Skin-Roll, Skin-Yaw; followed by RMS Linear Acceleration and the Device Reset bit.

This QSTM message String is finally transmitted to the PC software using USB communication protocol at the rate of 115200 baud rate. The rest of the processing for further calculation of the final QSTM treatment parameters are done by the PC software.

The next chapter demonstrates the PC software structure, its design features and finally the implementation of the PC software to visualize raw QSTM data and yield QSTM report at the end of treatment mode.

8. QSTM PC SOFTWARE (Q-WARE)

This Chapter explains the QSTM™ PC Software, which is also called Q-Ware©. The features required to design Q-Ware primarily depends on the system working modes, i.e., *Idle Mode* and *Treatment Mode*. An overview of the software design requirements is stated below which assisted in formulating the layout of the working structure of the software. This proposed software layout forms the basis of software implementation, which includes the User interface features, its computational logic, and the step by step workflow of all the processes involved to serve all clinical requirements.

8.1 Overview Of PC Software Requirements

The QSTM PC software should act as a display unit to graphically visualize quantified treatment parameters transmitted from the Q1 treatment device and record better replicate and analyze STM treatment response.

QSTM Treatment Session – A Treatment session includes targeted massages on soft tissue sites by the application and release of force on skin with the tip of Q1 treatment device. The complete treatment covering targeted massages on all soft tissue sites from the start of treatment mode of the system until its end is called a QSTM Treatment Session. The Treatment session time is arbitrary and is completely therapist dependent.

It should be mentioned that, the treatment sequence of QSTM initially starts with the Idle mode. This is when the treatment device is prepared for treatment, followed by the treatment mode where the actual treatment takes place. Then after the treatment session, the Idle mode follows again where the Q-Ware is supposed to save the recorded treatment data into a respective patient data file.

During a QSTM treatment session, the therapist is subjected to change the treatment site from one soft tissue location to another and switch hands. It is therefore necessary to take into account the treatment intervals (dead-times) of a treatment session. During this treatment interval (dead-time), the treatment device should perform a device reset to flash its memory of its previous treatment values. Therefore, the actual contact time of the treatment device with skin, account for the active treatment time of a treatment session.

QSTM Treatment Sub-Session − This includes a sustained targeted massage on a particular tissue site during treatment mode. So, there can be several treatment sub-sessions during the complete treatment session of the treatment mode. The treatment intervals (dead-times), that occur during treatment mode differentiates between the QSTM treatment sub-sessions. Moreover, the active time of a treatment sub-session is completely therapist dependent and depends on the contact time of the device with skin during treatment.

These observations of a typical QSTM treatment session lays out some major requirements to be satisfied by the PC software:

1. A Real-Time visual monitoring system with automatic device connectivity to PC, to visualize quantified QSTM data, which can be referred to during ongoing treatment to study inconsistencies in practice.

2. A QSTM workspace to enroll patients and auto-save respective treatment data to patient directories for treatment diagnostics, characterize QSTM practice and better analyze soft tissue manipulation.

3. Further statistical data analysis of visualized data to generate overall QSTM treatment report after each session. These data should contain statistical parameters like applied stroke frequency, peaks and averages of quantified data during the whole treatment session.

4. Our research is confined to a localized treatment applicator of soft tissue manipulation. But the future research would extend to several other treatment devices. So, the software structure should be made such that it can adapt future devices and their concurrent working with real-time visualizations simultaneously.

These requirements see the necessity to structure the QSTM PC software in such a way that it meets all the present needs with provisions to expand for future adaptations. Hence our research proposes a software structure for Q-Ware in the following section.

8.2 Q-Ware Design And Working Structure

The working structure of QSTM PC software (Q-Ware) should be constructed such that it performs all tasks of both the operating modes Idle Mode and Treatment Mode of the QSTM system sequentially with least user interference. Therefore, to ensure this, the tasks were first categorized in a way that services for user requests were separated from real-time operations and data analysis operations. Most of the tasks during the Idle mode before treatment are preparatory user interactions for the treatment mode, while that after treatment involves treatment data preservation and manipulation.

For this the Idle mode is segregated into two parts before starting treatment and after completion of treatment. Idle mode before treatment is called Pre-Treatment Idle mode, while that after treatment is called Post Treatment Idle mode. The next section demonstrates a proposed software working architecture that needs to be implemented in order to satisfy all the required needs keeping in mind the future adaptions of the system.

8.2.1 Proposed Q-Ware Working Structure

The Quantifiable Soft Tissue Manipulation is essentially a practice which needs to be adopted in clinics for better soft tissue analysis. In future, this practice would include different types of QSTM devices with treatment tips of different shapes used concurrently during a treatment session. Since our research only uses a localized treatment applicator Q1 device, hence the treatment mode uses only one treatment device.

Hence the Q-Ware architecture should be structured with standalone PC applications adopting the following approaches:

1. **Child Application(s)**:

   Treatment device specific treatment visualization applications for each device. Our research uses only Q1 treatment device, hence the visualization application for the Q1 device is called Q1 Treatment App. This application should be activated as a child application only during the Treatment mode of the system and terminated once the treatment mode ends.

2. **Parent Application**:

   QSTM GUI application would serve as a parent application of Q-Ware throughout all modes of the system. This parent application should perform all initial tasks to prepare the system for treatment mode before treatment; activate treatment application of treatment device and track time during treatment mode; finally preserve stored treatment data from treatment devices after treatment based on user requests. The preparatory tasks might involve user interactions to select or enroll patients to be treated during treatment mode and initiating the child application Q1 treatment app to start treatment.

3. **QSTM Workspace**:

   QSTM Workspace is a disk folder, which contains shared system and process information between the parent and child applications and record treatment

data. The QSTM workspace should also include a folder to store patient treatment data. A Folder Structure of the proposed QSTM workspace is shown in section 8.4.1.

4. **QSTM Data Analysis Library**:

   Object Oriented Programming (OOP) approach should be adopted to design the user interfaces of the parent and child applications such that methods for

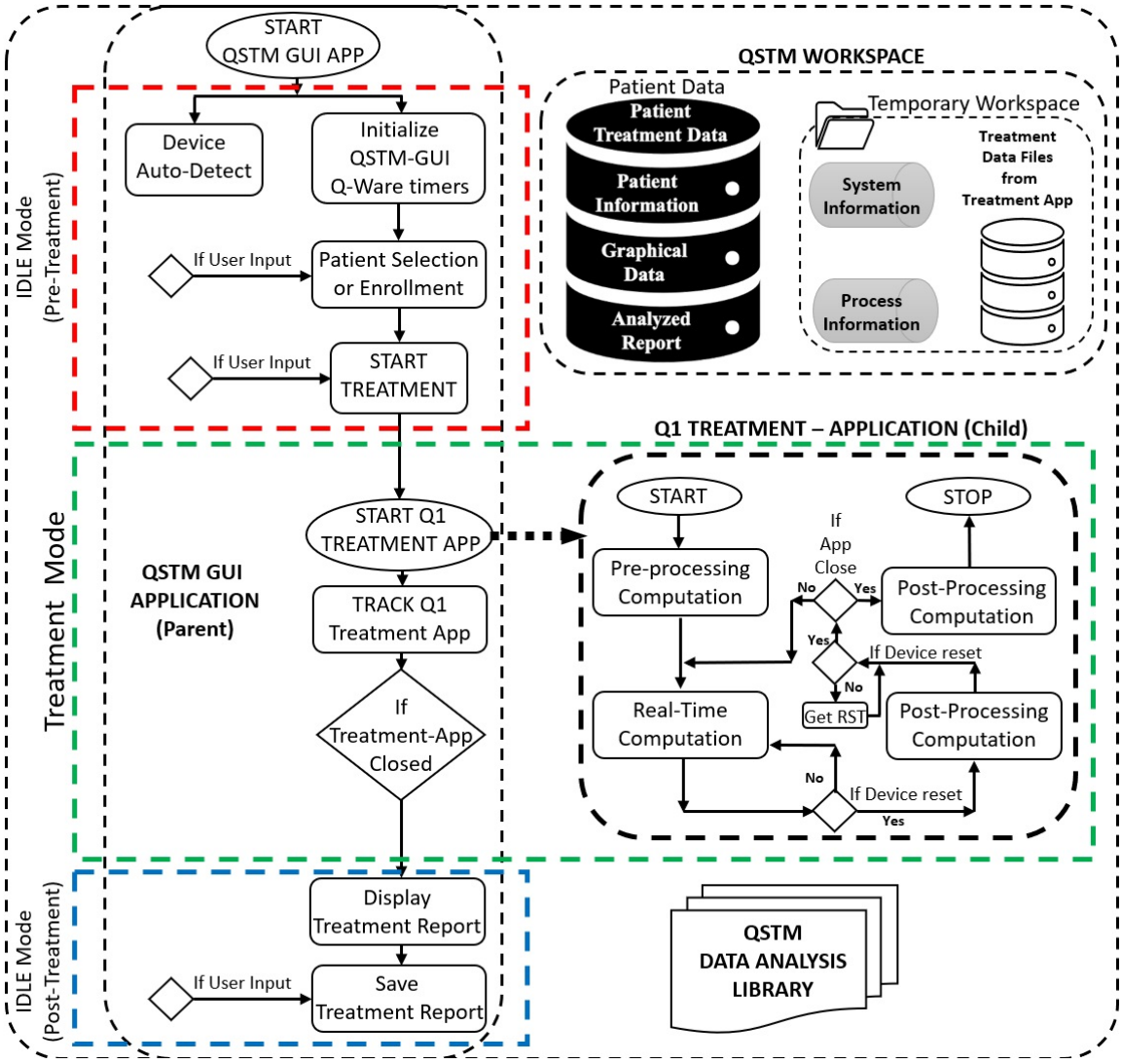


Fig. 8.1.: Flowchart representation of proposed Q-Ware structure with Parent and Child applications

a particular task can be reused in both. Using this approach, a QSTM data analysis library can be developed to analyze QSTM raw data to yield QSTM treatment report parameters and mark treatment data graphically for future treatment reference.

Fig. 8.1 in page 123 demonstrates the proposed working structure of Q-Ware with all above mentioned concepts. It represents the Parent and the Child application and its working structure on different modes of the system namely Idle mode (Pre-treatment), Treatment mode and finally Idle mode (Post treatment).

Pre-Treatment Idle Mode – to perform ideal QSTM practice, the Q-Ware will initially open up QSTM GUI application (parent application) during the Pre-treatment Idle mode of the System. This QSTM GUI will prepare the device for treatment by auto detecting QSTM device. Then the user would select or enroll patient for treatment and enable treatment mode to start treatment. This marks the end of pre-treatment idle mode of the system.

Treatment Mode – Once the treatment mode is activated by the user, the parent application i.e. QSTM GUI, would spawn the child application i.e. Q1 Treatment App, for data visualization and graphical data analysis. Meanwhile, the QSTM GUI (Parent application) would track the running status of the child application by multiprocessing.

The Q1 treatment App performs three kinds of computation during the treatment mode. Pre-processing computation involves all the preparation steps to initialize visualization and make the device ready. The Real-time Computation is when the actual treatment is undergoing, and graphs are updated. The Post-processing is when the device took a pause for switching in between sub-sessions and it uses methods from the QSTM data analysis library to generate treatment report. The Treatment device needs to be reset when the QSTM GUI transfers from real-time processing to post processing and vice versa. Once the treatment mode is terminated, the child application is closed, and a treatment report is saved in the QSTM temporary workspace.

Post-Treatment Idle Mode – The termination of the treatment application marks the initiation of the Post-treatment Idle mode, where the treatment report is displayed on the QSTM GUI (parent application). After report display, the QSTM GUI waits for the user consent to save the report permanently in Patient folder of QSTM workspace in PC. This is how a treatment session should be done by the proposed software structure.

This concept of developing separate parent and child applications is adopted to help augment Q-Ware in future, for several treatment devices to work together simultaneously. In that scenario the QSTM GUI of the Q-Ware will be talking to multiple treatment applications of several treatment devices in action particularly during the treatment mode.

The next subsections explain the details of the GUI design features of the parent application (QSTM GUI) along with the visualization interface design features of the child application (Q1 treatment app).

8.2.2 QSTM GUI Design Features

Following the above-mentioned software structure, the design of the Q-Ware assigns the QSTM GUI application as the main parent application. This QSTM GUI would be accountable for new patient enrollment or searching existing enrolled patients for treatment, along with automatic treatment device detection during Idle mode of the system. The QSTM GUI should also keep track of the running status of the device specific treatment application Q1 Treatment App during the treatment mode. Finally, this GUI is responsible for displaying the treatment data with all treatment parameters, write diagnostic remarks and automatically save them to respective patient folders. These final tasks are during the post treatment mode i.e. when the idle mode follows after the treatment mode ends.

Thus, to facilitate and execute all the tasks properly the QSTM GUI is required to have all the following features and attributes:

1. User input dialog boxes to enter patient information for new enrollments or searching existing ones.

2. Panels to display selected patient information, additional system information, treatment reports in formats accepted by clinicians.

3. A temporary QSTM workspace to store runtime QSTM data and a default folder path to store patient datafiles and organize treatment data into respective patient folders.

4. Input button to start treatment mode for real-time data acquisition after user input for patient enrollment or patient search is completed.

5. Additional input methods like checkboxes or buttons to choose patient names from patient lists or to visualize previously save treatment data.

6. Method to track whether the Q1 Treatment Application is running or not.

7. Display treatment analytics in tabular format, include diagnosis remarks or respective comments and options to permanently save them at respective patient folder.

8. Options to retrieve saved treatment data for later use or future analysis.

9. Consoles to print running values of different processes for exception handling and debugging. This feature is only for developer version which should be removed in the commercial version.

Implementing the above-mentioned features makes the QSTM GUI user friendly with necessary user interactions to switch between the idle and treatment mode of the system. The next section includes the attributes required to design the Q1 treatment app, which is a child application spawned by the QSTM GUI to visualize QSTM treatment data during the treatment mode.

8.2.3 Treatment App Design Features

This Q1 Treatment application is supposed to be the child application spawned by the QSTM GUI of the Q-Ware only during the treatment mode of the system. So, this application should perform all computational tasks for initializing treatment mode, real-time graphical data monitoring and storing, data analysis and displaying treatment parameters during intervals between treatment sub-sessions. The data analysis should be performed by implementing signal processing functions for noise smoothing of raw data and calculating treatment parameters. Hence, a QSTM Data Analysis Library, for calculating significant treatment parameters from raw QSTM data, should essentially be implemented and tested. Based on these specified requirements and medical need of the system, the Q1 Treatment App should have the following design features and GUI characteristics:

1. Graph widgets required to monitor real-time forces applied on tissue. So, a suitable tool with multi-plotting graph widget needs to be chosen and tested.

2. As the graphs are not confined to force graphs only, a series of graph widgets plotting the orientation angles with respect to gravity called Geo-Angles as well as angular device inclination from skin based on skin co-ordinates called Skin-Angles need to be played in real-time. Hence a Dock Area to house the multiple graph widgets is needed to structure the GUI of the Q1 treatment app.

3. There should be minimized communication latency less than 100th of a second to maintain real-time performance of the system.

4. A data analysis table needs to be featured on the GUI, which is responsible to upload QSTM parameters for sub-session treatment data in between reset intervals, for self-assessment of practice and reliability of treatment of the clinicians.

5. A real-time force monitor is another extended feature which was demanded to measure error rates of clinical reliabilities of constrained treatments with and without visual monitoring.

6. The visual monitor needs to be as concise as possible with limited or no user interference. So, auto connection of device with the treatment app to establish serial communication along with auto saving of treatment data in temporary QSTM Workspace needs to be incorporated.

The design features mentioned above are liable to every or any change as defined by clinicians, after running extensive customer feedback. As of now, the features mentioned above are the significant and mandatory ones needed to develop an initial software prototype to physically assess QSTM treatments.

8.3 Software Development Environment

The software implementation focuses on the development of two separate run-time applications both with user interfaces featuring the attributes mentioned in the last section. An intensive research on tools and open source libraries to develop PC applications were performed keeping in mind the software compatibility to incorporate features and maintain software performance and integrity with highest possible efficiency.

Several software development platforms were tested to account for the complexity of development, feasibility of integration, estimate compatibility issues and run tests for performance analysis. However, Python was selected as the programming language over other programming languages. Python's built-in high-level data structures, dynamic data types, powerful polymorphism feature to structure object-oriented programming and compatibility to support a variety of cross-platform scientific libraries written in C++ or Java with faster computation makes it suitable to develop real-time applications.

The programming development environment for both the applications constituting Q-Ware is Python IDE of PYTHON-2.7.14 [43]. Python scripts has been used to develop the application code while the default python shell is used to test and debug the codes. The GUI development platform for implementing QSTM GUI is a python toolkit "WX-Python" version (v3.0.0 msw) [44], where object-oriented programming has been used to develop the GUI code. The platform chosen to develop the Q1 treatment application is "PyQT4" [45] which is a cross-platform library originally written in C++ to support application program interface for GUI development. An extension of PyQT4 called "PyQtGraph" [46] to plot multiple real-time scientific graphs for the QSTM raw data during treatment mode is facilitated in this Q1 treatment app. These are the main GUI development platforms used to design and implement Q-Ware. Apart from these, various other process specific and system specific python libraries has been used to implement the software. However, a special QSTM Graphical Analysis library has been developed which includes methods and functions for signal processing and noise smoothing, graph peak filtering and standalone graphical visualization of recorded treatment data in *.csv* format.

The following section describes the implementation of QSTM GUI application which serves as the parent application of Q-Ware. The description below explains the GUI widgets, their utility and significance used in the application. However, the description of sequential processing of the QSTM GUI application is completely based on the operation modes of the system. All tasks are explained in step by step order.

8.4 QSTM GUI (Q-Ware Parent Application)

This application is developed on a python-based GUI toolkit named WX-Python, as mentioned earlier, especially designed for python-based GUI developers. The primary segments comprising the QSTM GUI application includes:

1. *QSTM Temporary Workspace (folder)* − to store system specific data for parallel processing, treatment data processing as well as treatment data storage.

2. *QSTM Patient Datafiles* − comprising an organized patient lists maintaining each patient's treatment history.

3. *Graphical User Interface (GUI)* − for user interaction to control operating modes of the system and display or record treatment data.

The patient datafiles forms the basis of treatment records, where patient information and their corresponding treatment data are stored for present and future use. The Patient data files along with the QSTM temporary workspace together completes the QSTM workspace as described in the system. The following section discusses the implemented folder structure of the QSTM workspace indicating its default location.

8.4.1 QSTM Workspace Directory Structure

The QSTM workspace is the default directory generated by the Q-Ware to store recorded Patient Treatment data. A QSTM temporary workspace is allotted in this directory to store system and process information used for multiprocessing and treatment data analysis. Whenever the Q-Ware is installed into a WINDOWS PC and run for the first time, the QSTM GUI is executed as the parent application. This QSTM GUI (parent application) by default selects the directory path "C:/Users/PC-Name/Documents" to create QSTM folder as default QSTM Workspace. Once the workspace is generated, the following folder structure is created to store patient treatment data in Patients Treatment Folder, while save temporary system and process data in "*QSTM_Temp*" folder, which is the QSTM temporary workspace, as shown in Fig. 8.2 on page 131.

The three important sub directories that constitute the QSTM Workspace are explained as follows:

1. **QSTM Patients** − This sub directory usually contains a list of subfolders of all enrolled QSTM patients. Each of these subfolders in turn contains a folder named with the respective Enrollment_ID of the patient. Inside this

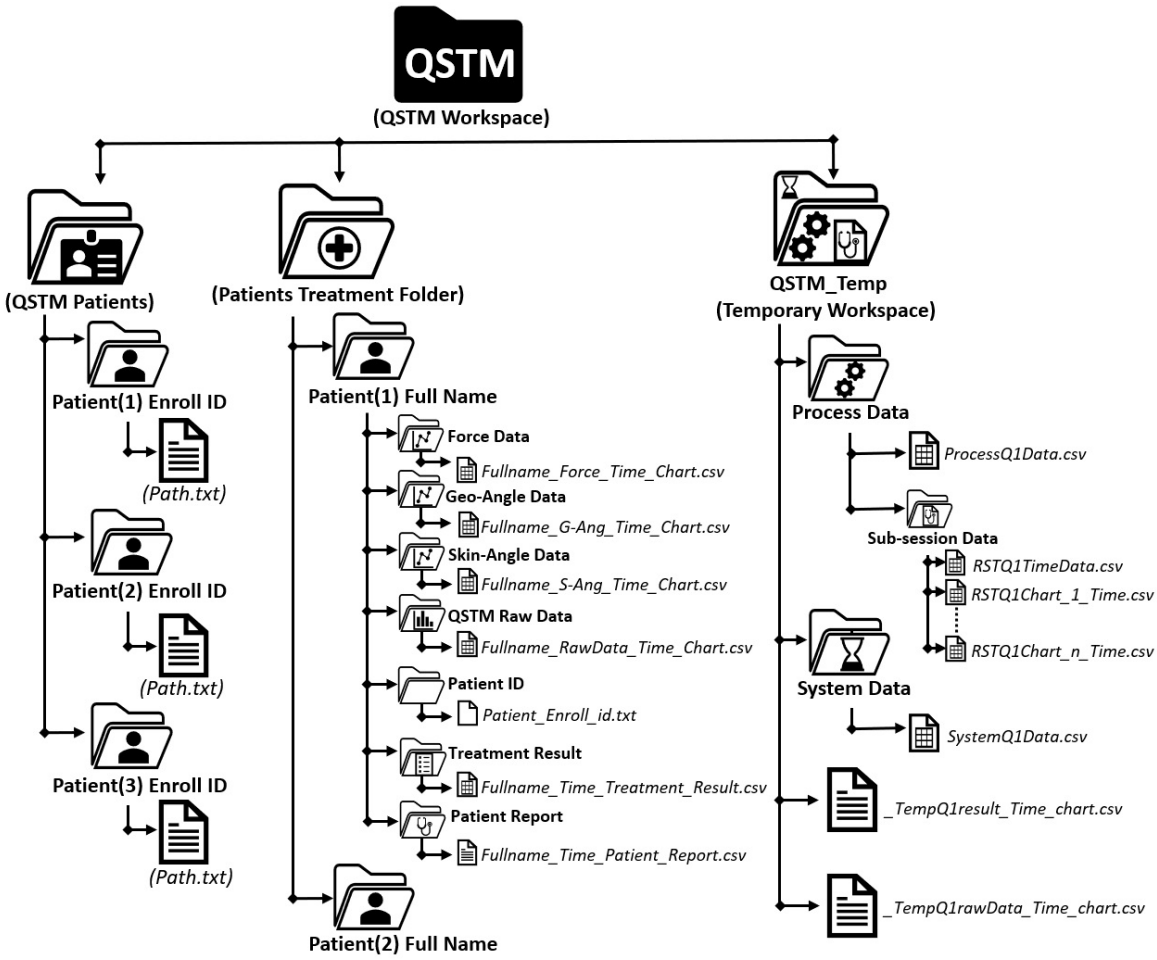


Fig. 8.2.: Folder tree structure of the QSTM Workspace

Enrollment_ID folder, a text file is maintained with all the path locations of the corresponding sub directories of the patients treatment data. The patient list during the existing patient search option, explained later, is extracted from the list of subfolders of this directory.

2. **Patients Treatment Folder** − This subdirectory serves as the main QSTM treatment record folder, which typically contains subfolders of all enrolled QSTM patients with their permanently recorded QSTM treatment data and treatment reports. Each of the patient subfolder holds a collection of six sub-subfolders of categorized QSTM treatment data which are:

(a) *QSTM Raw Data* − The QSTM message string which is transmitted from the treatment device during treatment is actually stored in a chart in *.csv* format. This chart is named with naming format *"Fullname_RawData _Time_Chart.csv"* and held in this QSTM Raw Data folder. This filename corresponds to the filename *"_TempQ1_rawData_Time_chart.csv"* in the QSTM temporary folder, during a treatment session. This file is saved to the patient folder from *"QSTM_temp"* folder by the parent application. The QSTM raw data is further segregated into Force charts, Geo-angle charts and Skin-angle charts in *.csv* format respectively, which are further stored into the next subfolders described below. This segregation is done by the Parent application to extend the software feature of post treatment data retrieval in future.

(b) *Force Data* − The force data from the QSTM raw data chart is extracted and replicated into a *.csv* chart with filename format *"Fullname_Force _Time_Chart.csv"* in this folder. This subfolder comprises of all force data files of treatment.

(c) *Geo-Angle Data* − Similarly the orientation angles of the device measured with respect to gravitational reference frames is also extracted and stored in the form of geo-angle chart with naming format *"Fullname_G-Ang_Time_Chart.csv"*.

(d) *Skin-Angle Data* − Likewise the derived orientation angles of the device with respect to the skin plane is also stored in the similar way in a .csv chart with filename *"Fullname_Skin-Ang_Time_Chart.csv"*. This subfolder holds all such files.

(e) *Patient ID* − This subfolder contains text file holding the QSTM patient Enrollment_ID of the respective patient folder.

(f) *Treatment Result* − All the treatment parameters produced as a result of the complete treatment session are stored in a *.csv* chart of filename format *"Fullname_Time_Treatment_Result.csv"* in this folder. This corresponds to the filename *"_TempQ1_result_Time_chart.csv"* in the temporary folder *"QSTM_Temp"*, which is copied by the parent application.

(g) *Patient Report* − The treatment report constitutes the QSTM treatment prescription which lists the QSTM report parameters and diagnostic remarks of the therapist who treated the respective patient. The naming format of such a file is *"Fullname_Time_Patient_Report.csv"*.

The QSTM Patients Folder and the Patients Treatment Folder together forms the QSTM patient data files in the QSTM workspace.

3. **QSTM-Temp (Temporary Workspace)** − This subdirectory houses all the temporary treatment data files and subfolders to hold live system information and process information of the running parent and child application. The structure of this subdirectory is categorized as follows:

   (a) *Process Data* − This subfolder holds the *.csv* file storing the real-time process information of the parent and child application so as to ensure multiprocessing using file sharing. All the paths of files to be shared between parent and child applications are stored in a filename *"ProcessQ1Data.csv"*. This subfolder also houses an additional sub-subfolder to record real-time sub-session data for calculation of treatment parameters in between Treatment sub-sessions. This sub-session data sub-subfolder, consists of QSTM data charts of each sub-session from first to last.

   (b) *System Data* − This subfolder comprises of a *.csv* file maintaining the system data, involving the number of devices connected, their COM port numbers and corresponding "Registration-IDs". These will be further described later in this chapter. The filename of this chart has the format *"SystemQ1Data.csv"*.

(c) *Treatment Raw Data Chart* − This chart mainly holds the QSTM Data transmitted by the treatment device during treatment mode. All QSTM data captured during the treatment session are stored in this *.csv* chart with format *"_TempQ1rawData_Time_Chart.csv"*. This chart in turn is copied and saved into the QSTM Raw Data subfolder in the Patient Treatment Directory of QSTM workspace.

(d) *Treatment Result Chart* − The Treatment result chart is created at the end of treatment session after the post processing computation. The data analysis performed on the QSTM Raw Data Chart to generate QSTM Report and QSTM treatment parameters are stored in this chart with filename format *"_TempQ1Result_Time_Chart.csv"*. This chart is further copied and saved into the Treatment Result subfolder by the parent application in the Patient Treatment Directory of QSTM workspace.

The files and folder contained in the QSTM Temporary Workspace are flushed out of the allotted memory space before the start of a corresponding treatment session. Therefore, the files stored in this memory space needs to be saved into the QSTM Patient Treatment folder by manually clicking a Save Report button facilitated on the QSTM GUI. The background processes and methodology involved to execute these tasks are explained later in this chapter. However, the patients treatment folder holds multiple treatment sessions with the same filenames, where *_Time_* parameter of the filename, i.e. local date & time, is only updated in the naming convention of the files every time a new file is saved to the same folder. Thus, the treatment data can be retrieved later by accessing the *_Time_* parameter of the filename, which takes the format of *"hh-mm-ss_DD-MM-YYYY"* i.e."hour-minute-second_Date-Month-Year".

The User Interface of the QSTM GUI (parent application) forms the main human PC interface to control the operating modes of the QSTM system. This research only deals with the implementation of Q1 device, hence the QSTM GUI operates the idle and treatment mode of this particular treatment device by user interactions.

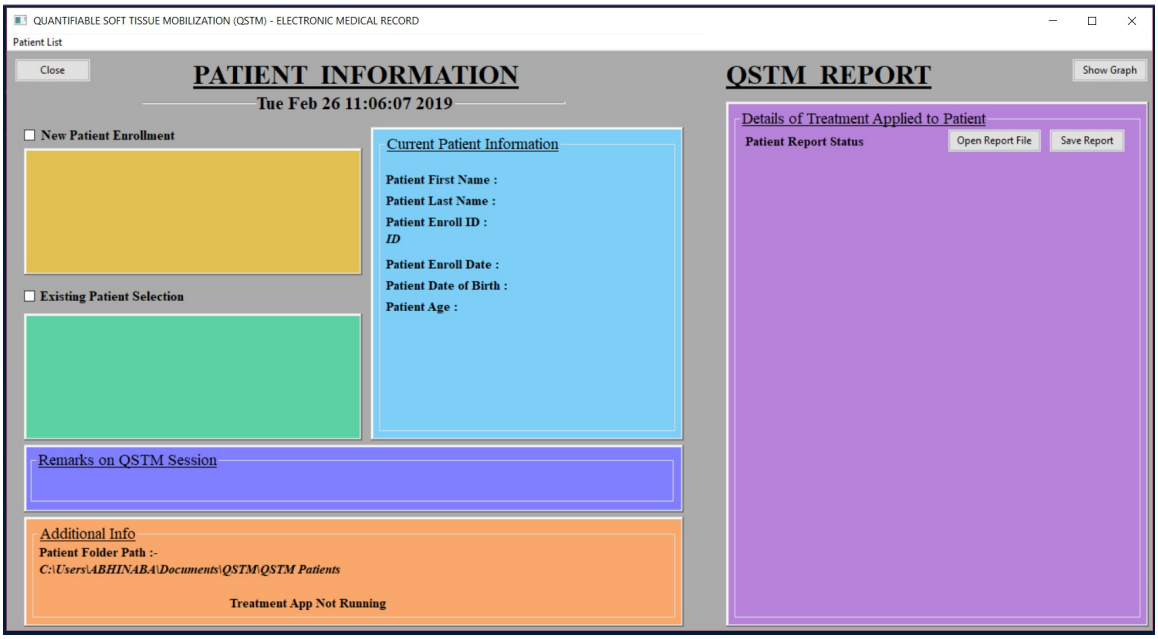


Fig. 8.3.: Screenshot of QSTM GUI – Parent application of Q-Ware

An Object-oriented programming approach was followed to develop this parent application, where standalone functions leveraging data handling, system operation and communication features were separated from functions describing GUI widgets.

The next section describes all the GUI features that has been implemented into parent application (QSTM GUI) along with their significances.

8.4.2 Description Of QSTM GUI Widgets

The GUI of the parent application shows two divisions, which involves – Patient information and QSTM treatment report. The patient information section includes separate dialog boxes with tagged checkboxes for new patient entry or existing patient search as shown in Fig. 8.3 on page 135. This part also holds panels for additional system information and editing treatment remarks. On the other hand, the Report panel includes buttons to save and display generated QSTM treatment report. Also, it displays the QSTM report in a tabular format explained later in this section.

8.4.2.1 Patient Information

This part of the QSTM GUI forms the most significant part of Q-Ware where, patient details can either be entered into the QSTM datafiles or retrieved from the same. It also includes panels to display labels about operating mode status of the Q-Ware and edit treatment remarks post treatment mode.

New Patient Enrollment – The yellow dialog box is provided in the GUI to facilitate data entry of the patient into the QSTM workspace by the user during the idle mode of the system. Checking the checkbox activates a new dialog box and a window to enter patient details appears on the screen (see Fig. 8.4 on page 136).

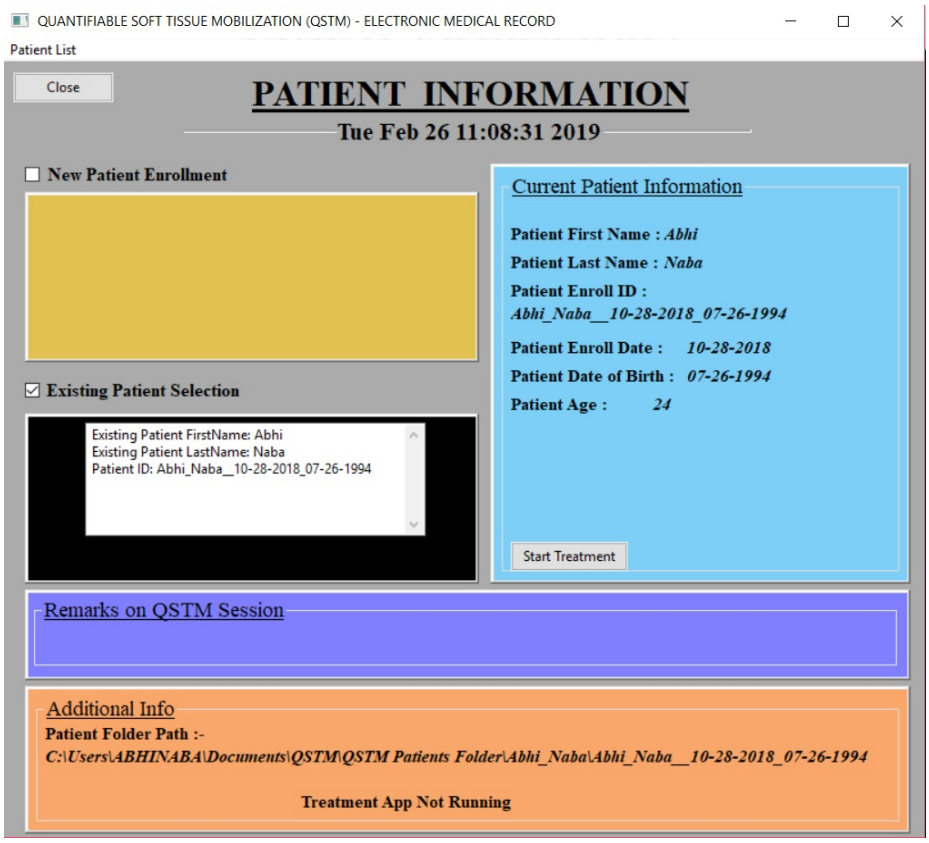


Fig. 8.4.: Patient retrieval by Existing Patient Selection from patient list

Once the data is entered by the user, the process information appears on the console of the dialog box. This console feature is incorporated for testing the GUI for failure modes and error analysis.

Existing Patient Selection This green dialog box acts similarly like the new patient selection dialog box (Fig. 8.4 on page 136). When activated checking the checkbox, it opens the input methods to find an existing patient either by manually locating patient from a patient list or by entering patient credentials for existing patient search. Once the patient credentials are matched with a particular entry in the list then the system executes a function to retrieve path of patient data folder and extracts its patient id to display stored personal information.

Current Patient Information Panel – This panel is responsible for displaying the personal details required for identifying an existing patient in the database or enrolling a new patient who is currently subjected to QSTM treatment. The details which are displayed by this panel are:

i. Patients first and last name.

ii. Patients enrollment date.

iii. Patients date of birth.

iv. Patients current age.

v. Patients enrollment-id

First four elements are the very basic requirements needed to identify a patient based on which the QSTM GUI generates a unique enrollment id for every patient. The enrollment id comprises of the first and last name of patient including a combination of enrollment date and date of birth. The operations of new patient enrollment and the existing patient search are explained later in this section.

The new patient enrollment process performs a series of tasks which includes opening a user dialog for credential input, assignment of patient id, calculation of present patient age, creating data folder and subfolders with patient name and storing necessary patient information to respective subfolders. Once these tasks are performed, the panel displays the data on screen.

Similarly, the existing patient search matches entered data to that stored in the workspace. If the selection is direct from the patient list, then it retrieves the patient information from the patient path and displays it on screen.

This panel also introduces a Start Treatment button to start treatment mode of the QSTM system, after the user credentials are uploaded on the patient information panel. Activation of this button starts the treatment mode of the system and opens up the Q1 Treatment App. This button acts as a bridge between the parent and the child application of the Q-Ware i.e. QSTM GUI and the Q1 visualization application.

Remarks on QSTM Session Panel – This panel is purposely facilitated to make critical comments on the type of treatment performed by the clinician and to indicate the diagnosis features as remarks for the overall treatment session post treatment mode of the system. However, this panel in future can be extended to edit or manipulate previous treatment remarks as required by a therapist.

Additional Information Panel – This panel displays the running status of the child application (Q1 treatment App). However, if this software were extended to monitoring multiple QSTM devices in the future, then this panel would be responsible for displaying the statuses of operating modes of all specific QSTM devices being concurrently used for a particular treatment session. It also displays in which mode the system is operating now. It also labels the content for the running information of the system and displays patient path of particular patient being treated.

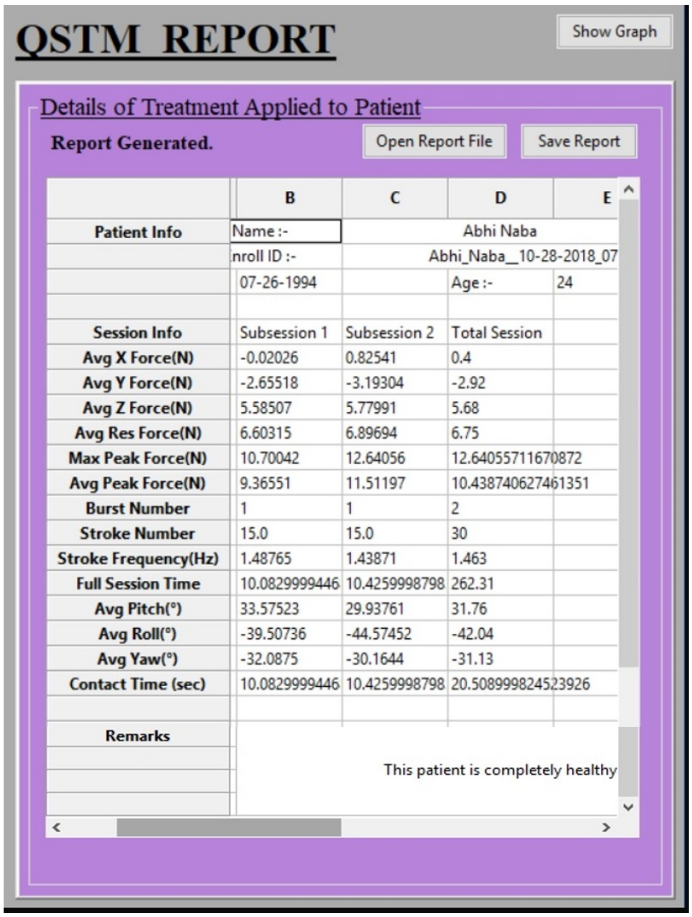


Fig. 8.5.: Report Panel Diagram

8.4.2.2 QSTM Report

The QSTM Report panel acts as a space holder for generating and displaying reports of corresponding treatment sessions performed on a patient or a subject. It displays the outcome of the complete session in a tabular format, detailing and separating critical QSTM parameters necessary for clinical use. However, the QSTM parameters are distinguished in a row-column format, where the columns reflect the sub-sessions done between successive resets. The bottom of the table includes space for remarks, where a clinician can comment on the diagnosis of the treatment. Moreover, this panel includes a couple of buttons to save the generated report and view

saved report for future reference. A button has been featured at the top right corner of the report panel to show standalone graphs of the session with filtered peaks and distinguished stroke cycles.

8.4.3 QSTM GUI Working Methodology

Since QSTM GUI is the parent application of the Q-Ware, it tracks the program execution statuses of all tasks performed during the operating modes of the system. The two operating modes Idle mode and Treatment mode forms the basis of all the program executions of the parent application. The Idle mode of the system follows successively both before and after every treatment mode. Hence to explain the working algorithm of the parent application the Idle mode can be distinguished based on the succession of the treatment mode.

Idle mode of the system before treatment is called - *Idle Mode (Pre-Treatment)*, whereas that after treatment is called - *Idle Mode (Post-Treatment)*. Each of the Pre-Treatment and Post-Treatment Idle mode performs distinguished functions of the parent application in the following subsections. The sections that will follow are Idle mode (Pre-Treatment), Treatment mode, and finally Idle mode (Post-Treatment).

8.4.3.1 Idle Mode (Pre-Treatment)

This section describes the tasks performed by the parent application of the Q-Ware before starting treatment mode of the system. The fundamental tasks of the parent application during this part of the idle mode are:

i. GUI initialization and Q-Ware soft-timers setup.

ii. Auto-Detection of Q1 Device.

iii. Patient selection by user for treatment mode.

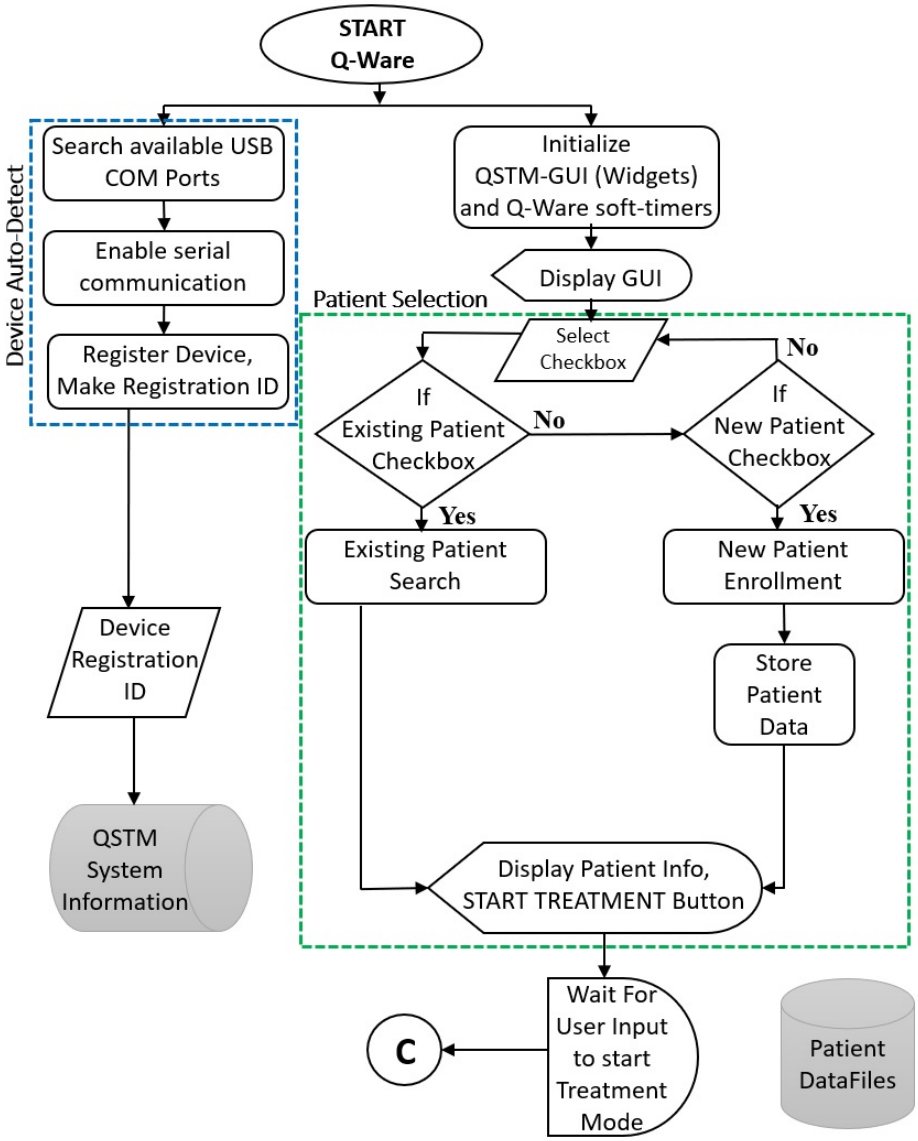


Fig. 8.6.: Flowchart representation of Idle mode (Pre-treatment) of the QSTM GUI (Parent application)

Fig. 8.6 on page 141 demonstrates the working flowchart of the parent application of Q-Ware during the pre-treatment Idle mode of the system. The device auto-detection blocks of the flowchart are marked by the blue colored dotted lines, while the patient selection process is marked by green colored dotted lines. The grey colored objects represent the datafiles for system information and patient treatment

information in the QSTM workspace. The terminal block of the algorithm is a continuation block which will be discussed later explaining the treatment mode operations of the Q-Ware parent application.

I. **GUI And Software Timer Initialization**

The Graphical User Interface of the parent application constitutes all widgets for user interactions to provide input for several process executions. The initialization instructions structure the graphics of the user interface window and positions the widgets and aligns their corresponding display tools in allotted slots on the screen. The initialization instructions also automatically determine font sizes, colors of labels, and the scaling factor of the widgets in order to resize the application window uniformly when the resolution is changed to different pixel formats.

A major part of the of the initialization involves timer set-up for timestamp-based scheduling of several runtime applications. The key software timers of this parent application includes:

(a) *Wall Clock Timer* − This timer accesses the system clock of the windows operating system to display the date and local time with a timing interval of 1 second. This timer updates the clock on the QSTM GUI and displays the system time.

(b) *Treatment Timer* − This timer is the elapsed timer, which activates only on the user input to start treatment mode of the system. This timer triggers the live heart beat checking of the Q1 treatment application to track whether the application is alive or not.

Once the initialization process is successfully completed, the GUI window appears on the PC screen and displayed until the application is terminated.

II. **Device Auto Detection**

This task has three fundamental processing blocks which include checking for available communication ports connected to USB serial devices, initiating serial communication and lastly requesting device ID for device registration.

(a) *Search available USB Serial Communication Ports* − This is the initial task for QSTM device auto-detection, where the parent application requests the Windows operating system to enlist USB serial devices connected to PC and their respective communication port number. This task is performed iteratively every minute to update changes in the list. Whenever this list updates with a change, the successive tasks are followed, else the program raises an exception and waits until the next update.

(b) *Enable Serial Communication* − The parent application tries to set-up serial communication with all the available USB communication ports enlisted. The serial communication baud rate is set at 115200 bits per second, which outweighs communication initiation with other USB devices that doesn't comply to the set baud rate. This raises communication exception for USB devices with other baud rates, the app discards them as non-QSTM devices and moves to the next available USB device. If serial communication is established, the parent application requests the USB device to perform Device Registration, failing which the USB device is discarded as a non-QSTM device.

(c) *Register Device* − If the serial communication is successful, the parent application requests the device to send its device-ID for device registration and waits for an acknowledgement message. Once the acknowledgement from the QSTM device is received, the parent application decodes the 48-bit device ID, which is the input for device registration. This device ID contains a16 bits USB product-ID, a 16-bit USB vendor-ID, and a 16-bit QSTM Device Type. Then the communication port number is tagged to

the QSTM Device ID to form the registration-ID of the QSTM device. This registration-Id has the format of *(USB product-ID : USB vendor-ID : QSTM device Type : Connected COM Port number).* This Registration-ID is then stored into the QSTM system files for later use by the child application for Device Id Check.

This completes the process of auto detection of QSTM treatment device by the Q-Ware. The other processes of the pre-treatment Idle mode involves including Patient information by the user, either by choosing from the existing patient list or by enrolling new patient into the QSTM Patient datafiles.

III. **Patient Information**

The process of saving and retrieve patient information is necessary for managing treatment parameters for future reference or study. The probable methods in which this can be done are:

(a) If the patient is being treated using QSTM for the first time, the patient record needs to be established in the QSTM system by the New Patient Enrollment Process. The checkbox for new patient enrollment needs to be checked to start this process.

(b) If the patient is already enrolled in the system, then patient name should be searched either by providing patient credentials or by selecting the patient name from the existing patient list. This method called Existing Patient Search process can be activated by checking the corresponding checkbox.

A. **New Patient Enrollment Process**

When the checkbox for the new patient enrollment is checked, a new dialog box for patient entry appears on the screen, where the user is supposed to enter Patients first Name, Surname and Date of Birth. These data are further used to calculate patients age and generate a Patient Enrollment-ID. Patients Enrollment-Id essentially contains a combination of Patients

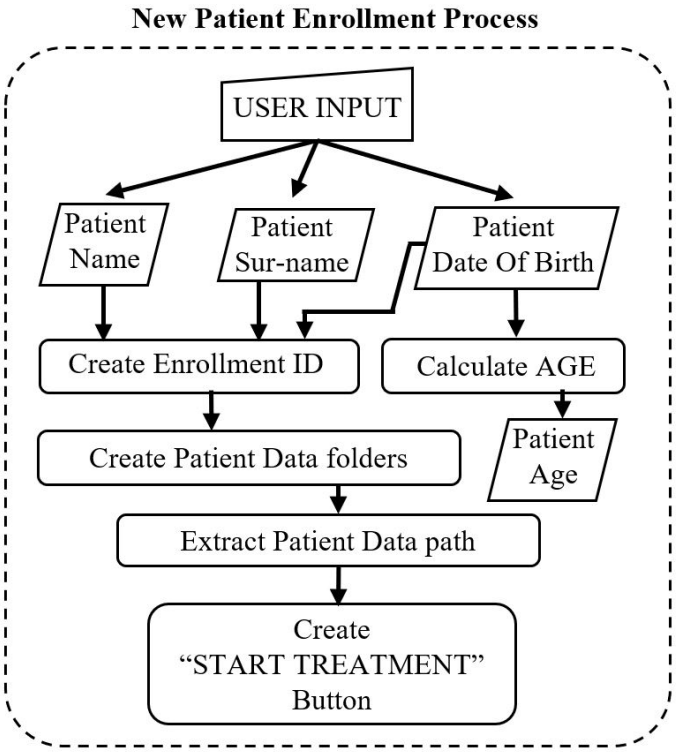


Fig. 8.7.: Flowchart representation for New Patient Enrollment Process

first and last name with the enrollment date and the date of birth. Once all Patient information are available, the parent application creates a Patient data folder in "Patients Treatment Folder" subdirectory of QSTM workspace and names it with the patients first and last name. The corresponding QSTM Patients folder holds a subfolder named with the patient's Enroll_ID comprising a text file *(Path.txt)*. This text file stores all path directories of files with respective patient information and correspondingly generates necessary subfolders, mentioned in section 8.4.1, to save raw and processed treatment data for QSTM treatment diagnosis in "Patients Treatment Folder" subdirectory of QSTM workspace. The created patient directory location of *"Path.txt"* is extracted to be displayed on the

Additional Information Panel of the QSTM GUI. In the end, the process creates the *"Start Treatment"* button and displays it along with the patient information on the Current Patient Information panel.

B. **Existing Patient Selection Process**

When the checkbox for the existing patient selection is checked, a "Search" button appears on its dialog box. This check mark activates patient search by patient credential input as well as patient selection from the existing patient list on the menu bar.

For patient search using the search button, the user needs to click the search button. Then a new dialog box appears on screen where the user needs to manually type the patient name and surname. These credentials are searched in the patient directory using a binary search algorithm. If the credentials match a name in the directory then it extracts stored patient information for the respective patient name, else it raises an error message "Name Error".

The user can also select a patient name from the existing patient list available in the menu bar at the top right corner of the GUI screen. This directly locates the directory path of the patient name and extracts the patient information needed to be displayed on screen. Lastly, this process also generates the "Start Treatment" button to activate treatment mode of the system. All the extracted patient information and the "Start treatment" button are displayed on the Current Patient Information panel of the QSTM GUI.

This completes the patient selection processes and now the system waits for the user to start treatment mode by clicking the "Start Treatment" button. However, the connectivity of the device needs to be ensured at the QSTM GUI before activating the treatment mode; otherwise, on clicking the "Start

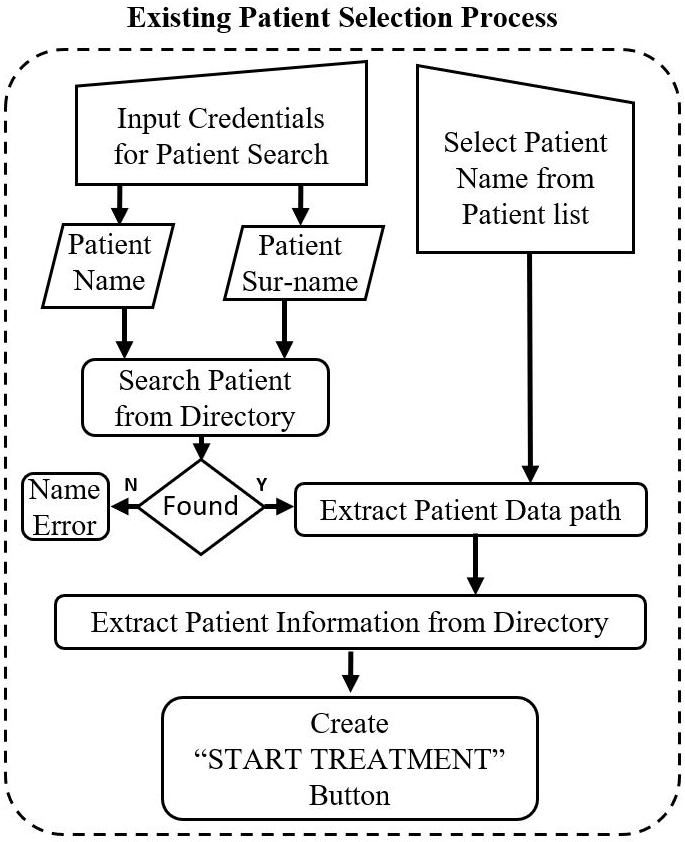


Fig. 8.8.: Flowchart representation for Existing Patient Selection Process

Treatment" button, the Q1 treatment app would throw an error message "No Device Connected". The next section describes the operations of the QSTM GUI parent application during the treatment mode of the system.

Fig. 8.9 on page 148 shows the flowchart representation of operations of QSTM GUI (Q-Ware parent application) during the Treatment mode and the Idle mode (Post treatment). The flowchart shown in Fig. 8.9, is a continuation of Fig. 8.6 on page 141.

The flowchart blocks inside the green dotted boundary demarcates the treatment mode operations of the parent application. While the flowchart block inside the red dotted boundary marks operations of the parent application during post-treatment Idle mode of the system. However, Fig. 8.9 on page 148 also represents the QSTM

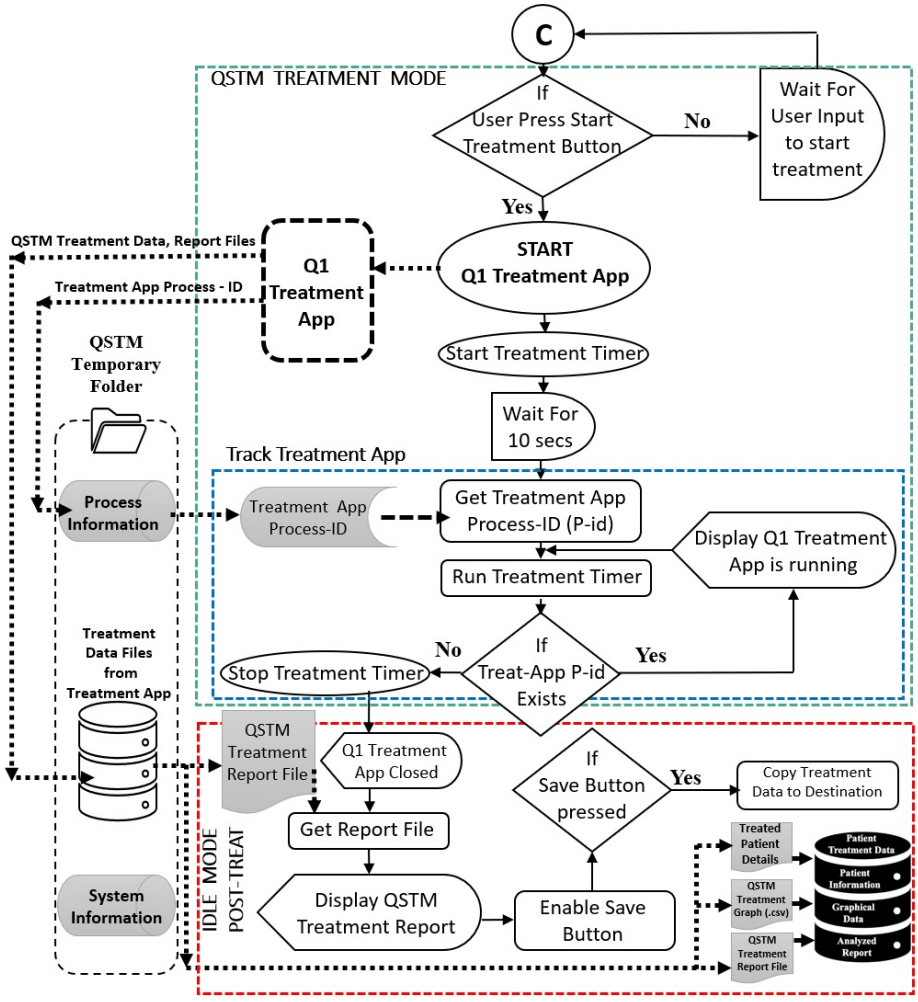


Fig. 8.9.: Flowchart and diagrammatic representation of Treatment mode and Post-treatment Idle mode operations of Parent application

temporary folder of QSTM workspace where several system information, process information and treatment information are stored for concurrent process executions and operations. The thick black dotted block inside the treatment mode represents the spawned child application Q1 treatment app, with black dotted arrows featuring the file sharing techniques to facilitate multi-processing of both the running applications (Parent and Child) simultaneously.

8.4.3.2 Treatment Mode

Once the user clicks the "Start Treatment" button, the parent application spawns the child application Q1 Treatment App, which performs the real time operations of QSTM raw data processing and create a new window to display data graphically and calculate results as QSTM parameters. The parent application of Q-Ware concurrently performs heart beat checking every 10ms to find out whether the child application is running or not, throughout the treatment mode. To perform this, a software timer is enabled by the parent application to track the child application. This timer is called treatment timer which updates every 10 milliseconds. The parent application waits for approximately 10 seconds after the start of the Q1 treatment app (child application) to start the heart beat tracking. The termination of the child application marks the end of the treatment mode which is followed by Idle mode (post treatment). The blocks inside the green dotted box in the flowchart represents the Treatment Mode operations of the parent application.

I. **Tracking Treatment Application**

The Q1 treatment app, i.e., the child application, being a separate windows application, generates its application process-ID and saves it to *"processQ1Data.csv"* of *Process_Data* folder in the QSTM temporary workspace. This process-ID is accessed by the parent application during the treatment mode to track if the Child application is running or not.

(a) *Get Treatment App Process-ID* – After starting the child process, the parent application waits for 10 seconds for the treatment app to start successfully, and then completes the initialization processes. 10 seconds wait time is needed to avoid malfunctioning due to software inconsistencies. The parent application calls a function *"pid_exists()"*, which checks for this Treatment App Process-ID at the Operating Systems processes list. If the process-id exists, then the function returns true else false.

(b) *Run Treatment Timer* – As long as the Q1 treatment app is running this process-id exists and parent application updates the treatment timer every 10 milliseconds and displays "Treatment App is running" at the additional information panel.

When the treatment app terminates, the process-id is destroyed, and the parent application stops the treatment timer and gets ready to perform tasks of the post-treatment Idle mode. The parent application then displays "Treatment Application closed" on the additional information panel. This marks the end of the treatment mode of the system.

8.4.3.3 Idle Mode (Post-Treatment)

The post-treatment Idle mode performs most of the treatment result display operations on the Report panel of the QSTM GUI (parent application). The red dotted boundary marked in Fig. 8.9 on page 148 shows the operating flowchart for the post treatment idle mode of the parent application.

When the Q1 treatment app is closed, it asks the user to save the treatment data and the treatment report is generated as a result of the treatment session. If the user consents to store the data before termination of the Q1 treatment app, it saves the treatment result file in *.csv* format at the *QSTM_Temp* folder, which comprise the QSTM temporary workspace. The treatment app also writes the saved path location of the report file with the file naming convention of *"PatientName_TempQ1Result_Time_Chart.csv"* and corresponding file path in the Process Data directory of the temporary workspace.

I. **Get Report File**

The QSTM GUI (parent application) checks the Process Data subfolder in "QSTM_Temp" folder for the saved report path, which if found, is accessed

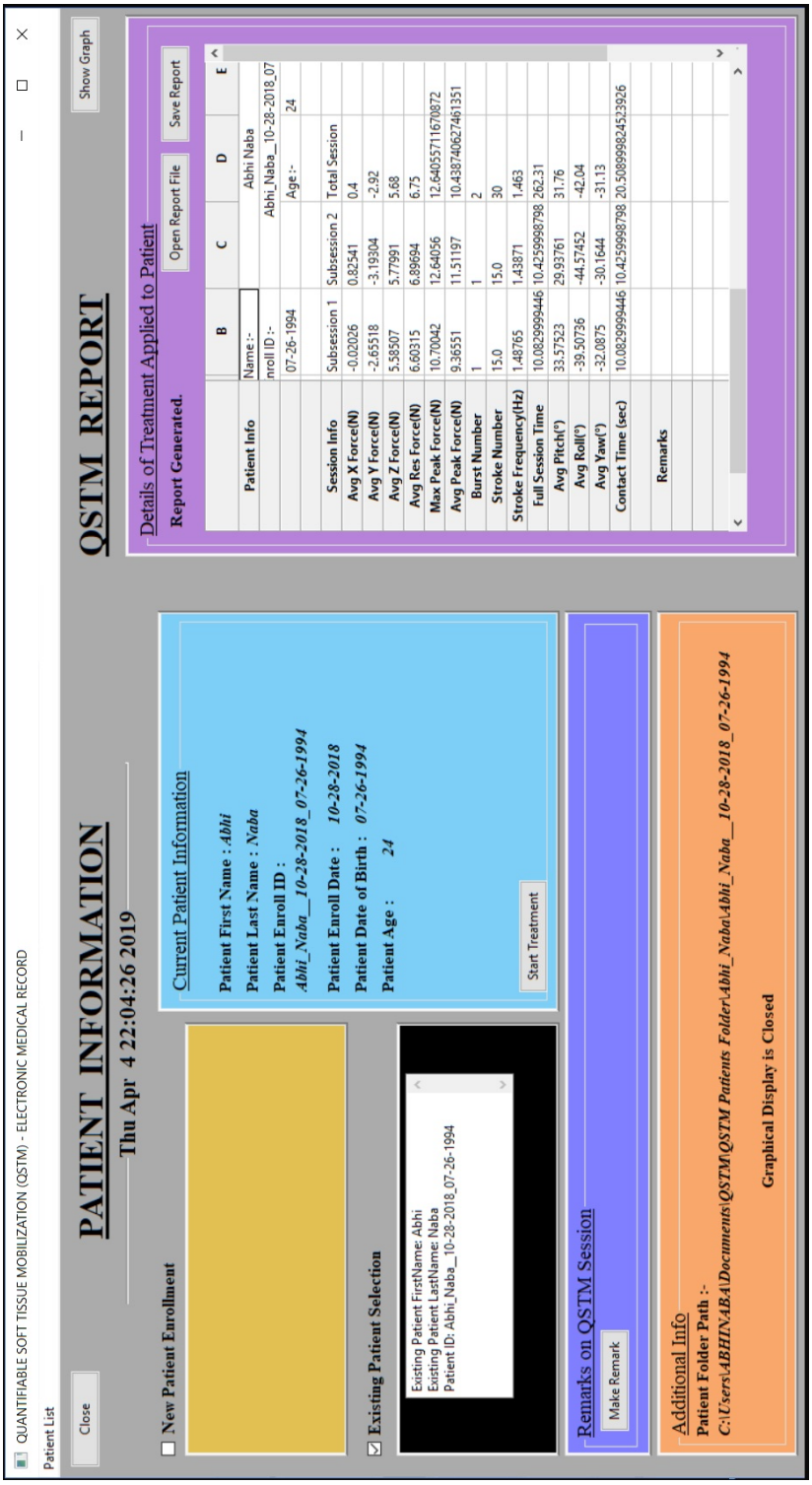


Fig. 8.10.: Screenshot of the QSTM GUI after one complete treatment session

from the mentioned location. This report file in *.csv* format is then parsed by the QSTM GUI to display the results in an organized tabular format on the report panel of the QSTM GUI.

II. **Enable Save Button**

The Save report button is enabled to permanently save the report file in the current patients directory. Before saving the User is allowed to make remarks on the treatment session for any future references. The remarks can be directly made at the remarks cell of the Report table displayed on the report panel. However, the remarks can be further edited by using the *"Edit Remarks"* Button on the Remarks Panel allotted in the QSTM GUI.

III. **Copy Treatment Data To Destination**

After making remarks if the user chooses to save the report file permanently, then the *"Save Report"* button needs to be clicked. This action calls a function to grab the QSTM raw datafile for graphical analysis and the saved report file, both in *.csv* format, and save them to the corresponding subfolders of the patient directory of the patient name being treated in Patients Treatment Folder. The raw QSTM data is segregated into separate charts as Force chart, Geo-angles chart and Skin angles chart which are stored in respective locations in the corresponding patient subfolder, whereas the report file is stored in the treatment results location of the patient directory. This completes all the operations of the QSTM GUI (parent application) during one complete treatment session.

Now if the User feels the necessity to treat another patient, then the checkbox adjacent to the patient selection dialog box needs to be unchecked. This uncheck destroys all the previous data, and clears the QSTM GUI and the QSTM temporary workspace and resets the software to start a new treatment.

The next section describes the operations of the Q1 treatment application, which is the child application of the Q-Ware. The activation of this treatment application marks the beginning of the treatment mode while its termination marks the end of the treatment mode. Hence all operations of the Q1 treatment app are specific to the treatment mode of the system, when actual treatment is carried out by the therapist.

8.5 Q1 Treatment App (Child Application)

This application is responsible for real-time graphical display of raw QSTM data transmitted from the Q1 treatment device during treatment mode. This application processes the raw incoming QSTM data further to calculate QSTM treatment parameters at the end of each treatment sub-session and consequently yields the QSTM treatment report files at the end of the treatment mode.

As mentioned earlier in Chapter 6, the Treatment mode has three states Device Ready, Device Working and Device Pause.

i. **The Device Ready State** – is when the device is prepared to start being applied on a soft tissue site for actual treatment. During this state the device reads forces less than 1 Newton i.e. within the load cell noise level. Time elapsed during this state is considered to be treatment inactivity time.

ii. **The Device Working State** – is when the Device is actually in contact with the skin and apply forces. During this state the variation of forces can be observed graphically on the GUI screen of the treatment app.

iii. **The Device Pause State** – is marked by the interval in between two sub-sessions, when the user re-positions the device or checks the treatment parameters generated for the last sub-session. Time elapsed during this state mainly constitutes the dead time of the whole treatment session.

These states depict the operation of the treatment device during the treatment mode as described in last two chapters. The QSTM message sent to PC is received by this treatment app, decoded and displayed graphically as well as textually on screen to assist visual monitoring of QSTM treatment. This application does not involve any user interaction with its graphical user interface during treatment but demarcates changes in states of treatment mode to distinguish treatment mode operations.

To filter the noise of the sensory signals, a QSTM Data Analysis library is built to graphically analyze incoming data of QSTM message stream, filter them as needed and calculate QSTM parameters for every treatment sub-session.

The next section describes how these states are visualized sequentially on the visualization screen of the Q1 treatment app. The working of different Widgets involved to facilitate these visualizations are described below.

8.5.1 Description Of Treatment App Visualization

Graphical visualization of treatment device's sensor data in terms of 3D forces in Newtons, and the Yaw, Pitch & Roll angles in degrees with respect to both gravitational co-ordinates and arbitrary skin co-ordinates are the prime focus of this treatment application. Additionally, this treatment app also includes − analysis of these data to calculate critical treatment parameters (to be described later) and store them for treatment records. Hence, the structure of the visualization GUI is organized in such a way that majority of the visualization screen would comprise of Widgets playing graphs in real-time to represent real-time forces and angles in which the device is acting on skin. The rest of the screen is covered with − a real-time force monitor where resultant forces are displayed textually and a parameter table to show QSTM treatment parameters in between treatment sub-sessions.

The whole Application has been scripted in "Python 2.7.14". The GUI development platform to develop this treatment app is based on "PyQT4", a cross-platform visualization programming interface, while the Graph widgets to plot real-time graphs

has been adopted from a QT platform derived open source python library called "PyQtGraph". The next section describes the GUI structure of the treatment application followed by description of the visualization widgets.

8.5.1.1 Description of GUI Structure

The GUI screen of the treatment app forms the main graphics window for the visualization application. A Dock Area type widget supported by the "PyQT4" platform is mounted on the main graphics window to dock cross-platform widgets from several other GUI platforms. The graphics window is first divided into grid layout with default sizes of different widgets to be docked. Then an additional feature of customizing these dock areas are added to it at run-time, such that the user can maximize the area by dragging any widget window as per convenience.

Our visualization is divided into six dock areas (Fig. 8.11 on page 156), to support three multi-plotting graph widgets for real-time force data (middle) , geo-angle data (lower left) and skin-angle data (upper left), which covers the majority of the screen and three dock. A fourth dock is reserved for a runtime console to print messages (upper right top). This dock is necessary for debugging the application. The fifth dock is where the Treatment parameters table resides with rows and columns as per clinical requirements (middle right) below runtime console. The sixth dock supports a real-time force monitor (lower right) below the treatment table, where the resultant of 3D forces being applied to the tissue during treatment appears numerically in real-time for visual assistance to the therapist. Each of these dock areas can be customized in different sizes. The runtime Console, the Parameters Table and the Force Monitor are positioned in a cascade one below the other at the right-hand side of the window as shown in Fig. 8.11 on page 156.

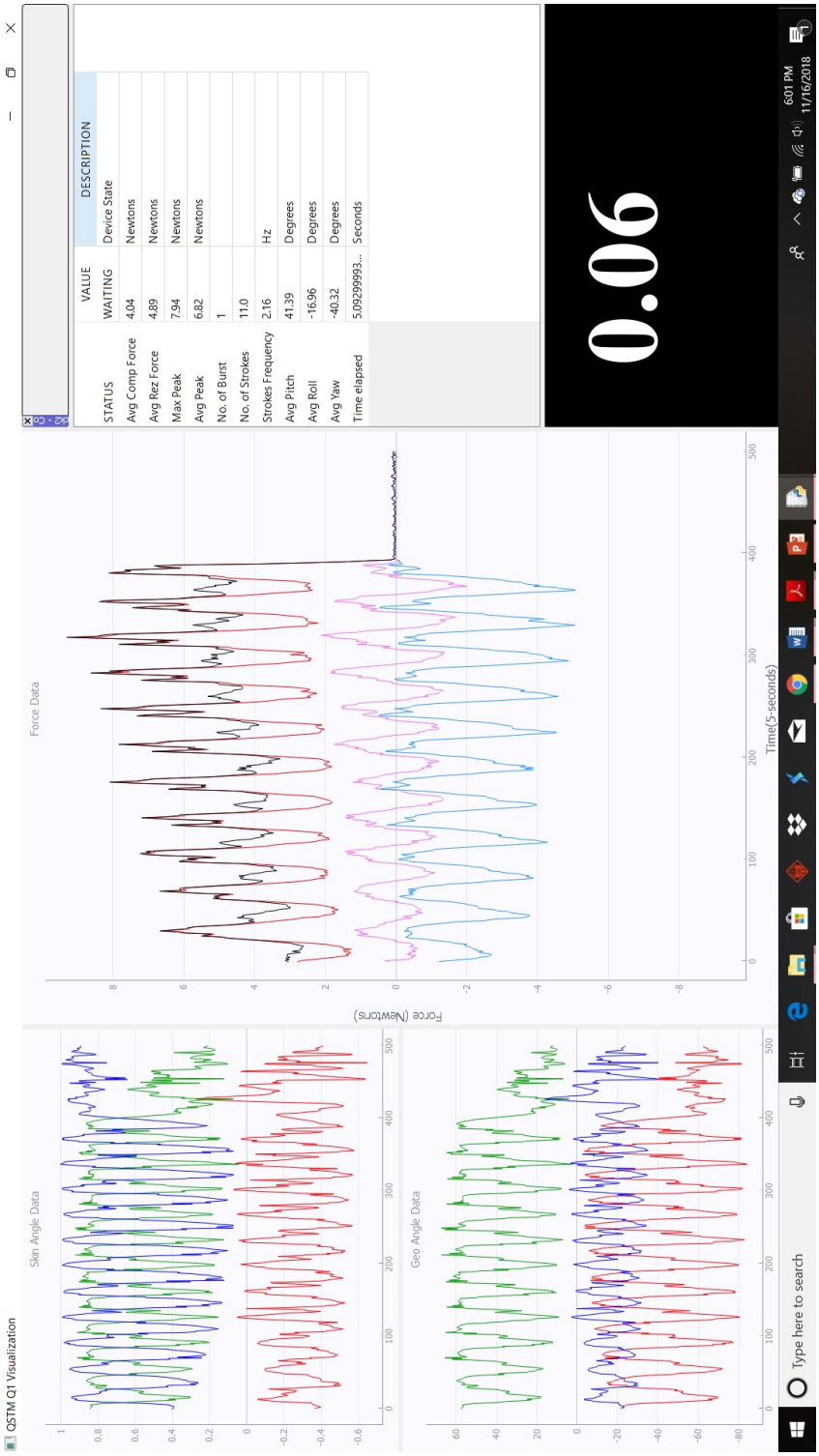


Fig. 8.11.: Screenshot of the Q1 Treatment Application during treatment mode

8.5.1.2 Graph Widgets

The graph widget of PyQtGraph serves the provision to plot multiple time series plots in real-time. This widget has the capacity to plot more than 100,000 points in a single plot within a second. This widget also features image processing facilities of 2D and 3D objects in real-time. This feature of the graph widget is harnessed to plot real-time plots of all transmitted QSTM data − force data, geo-angle data and skin-angle data.

Each update for all time-series plot takes place every 5 milliseconds. However, a method of time division multiplexing is used to update the plots to produce synchronized real-time visualization of the graphs. The details about the time division multiplexing algorithm for visualization will be explained later in this chapter.

The Force Graph Widget plots four time series plots i.e. Forces acting in X, Y and Z direction of the tissue plane as shown in Fig. 7.7 on page 103 of Chapter 7, along the resultant of these three forces. This widget plots up-to 500 data points of each of the 4 time-series in every update. So, a total of 2000 samples are plotted in every update.

The Geo-Angles Graph plots three time-series Yaw, Pitch and Roll angles of the treatment device with respect to gravitational reference frame. Similarly, the Skin-Angles Graph plots another three time-series plots of yaw, pitch and roll angles with respect to the device inclination form the skin plane. 500 samples of each of these time series are plotted in every update. So, both the Geo-angles plot and the Skin-Angles plot contains around 1500 points on every update.

For reference, the persistence of human eye is assumed to be 10^{th} of a second i.e. every 100 milliseconds. Normal motion pictures has a maximum frame rate [47] of 60 frames per second to create visual illusion of smooth motion. Our research uses a frame rate of more than 65 frames per second, since the serial port receives a new QSTM message string every 10 milliseconds. These messages are processed

and displayed immediately within the next 5 milliseconds. Hence, the visualization is synchronized with minimum latency. This is one of the significant contributions of the research.

8.5.1.3 Table Widget

This Table Widget, i.e. the Treatment Parameters Table, is featured on top of Force monitor at the middle right position of the visualization screen to display calculated treatment parameters on-to it for every treatment sub-session. The table widget updates itself on every button press of the treatment device which activates during Device Pause State of the Treatment mode. There are three columns of the table which represents treatment parameters particularly - Parameter name, Parameter value and Parameter description in physical units. The rows contain the treatment parameters which will be explained later in this chapter.

8.5.1.4 Real-Time Force Monitor

This is a graphics view widget mounted on the lower right position of the visual window, below the treatment parameters table to display real-time resultant forces delivered to the soft tissue during treatment. This widget displays the real-time resultant force applied on tissue for visual monitoring during treatment to maintain self-consistency of practice by the therapist. This Force monitor is built by mounting a view-box widget on top of a graphics view widget. This view-box is layered and aligned to the center of Graphics view widget and supports a text label of font size 20 to display Resultant forces numerically. The resultant force value attached to this label updates every 10 milliseconds.

8.5.2 Working Methodology Of Q1 Treatment App

The working methodology of the child application, i.e. the Q1 treatment app, is highly dependent on the requirement for Real-Time Computation to display QSTM visualization data on the screen and concurrently store treatment data. The whole working methodology is distinguished into three computation steps namely − pre-processing computation, real-time computation and post-processing computation, respectively.

Before the start of the real-time computation, the treatment app performs several tasks of visualization initialization and preparing the treatment device for treatment mode. This part of the computation is called Pre-Processing Computation.

Real-time computation of the Treatment app during the treatment mode includes the Device Ready state, the Device Working state and Device Pause state. This real-time computation can be momentarily halted by user interruption with a button press on the treatment device. This halt marks the Device Pause state of the treatment mode. Next button press resumes the real-time computation again. However, this real-time computation can be terminated only when the user shuts down the Q1 treatment app by closing the visualization window running on the PC screen, which marks the end of treatment mode.

Additionally, as mentioned earlier, a QSTM data analysis library has been written to process the collected input data and calculate QSTM treatment parameters for every treatment sub-session and for every complete treatment session. The QSTM data analysis library is called only during Device Pause State, the treatment app calls methods from the data analysis library to perform calculations on treatment sub-session data to yield and display treatment parameters. This computation is not real-time in nature and carried out only during the Device Pause State. Hence, this kind of computation is call Post-Processing. Post-Processing computation is carried out only during the Device Pause State as well as when the treatment mode is terminated.

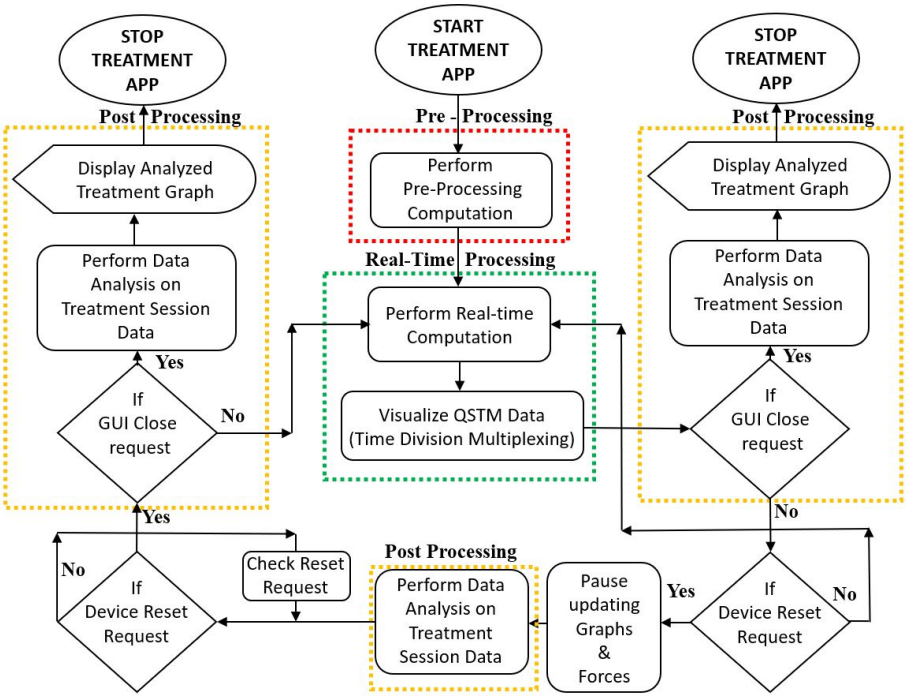


Fig. 8.12.: Flowchart representation of types of computation of Treatment App

Fig. 8.12 on page 160 shows the overall processing flowchart of the Q1 treatment app, which specifies the sequences of Pre-processing, Real-time processing and Post-Processing computations carried out by the treatment app during treatment mode. The red color boundary represents the pre-processing computation, for application initialization and device preparation for treatment mode. The green color boundary represents processes performed sequentially during real-time computation and data visualization. Both the Device Ready state and the Device Working state are embedded in this block, as depicted in Fig. 8.13 on page 161. A separated division marked by Blue colored boundary in Fig. 8.13 demarcates the Device Pause state. The Post-Processing computation is marked by yellow colored boundary, shown in Fig. 8.12, which appears either during Device Pause State or at the termination of the treatment app. The upcoming sections demonstrate all processes sequentially performed during each type of computation from Pre-Processing to Post-Processing.

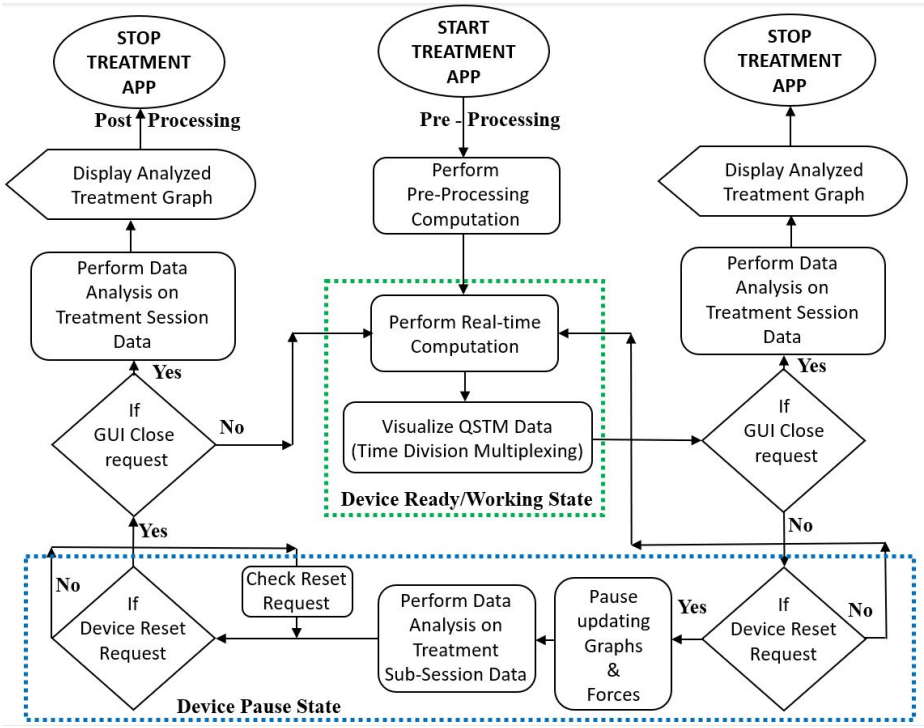


Fig. 8.13.: Flowchart representation indicating states of Treatment mode

8.5.2.1 Tasks of Pre-Processing Computation

This section mostly includes the initial tasks to prepare the treatment device for real-time computation of the treatment mode. Additionally, it also prepares the Q1 treatment app for visualization and yields process parameters to accomplish multi-processing between the parent QSTM GUI application and the child Q1 Treatment app. The tasks, which are performed sequentially to cater the preparation of the treatment device for real-time computation, are listed below:

i. Clear QSTM temporary workspace.

ii. Configure Treatment Visualization.

iii. Store Treatment application Process-ID.

iv. Check Q1 Device ID.

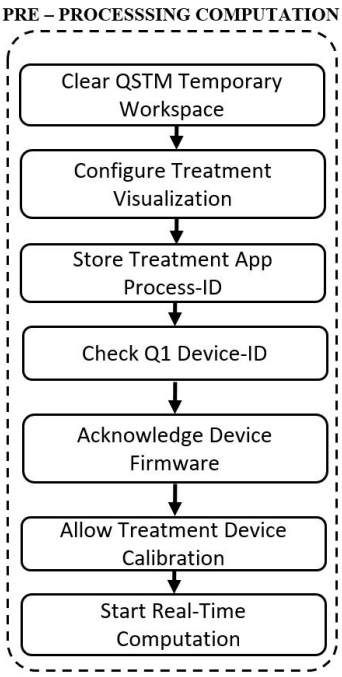


Fig. 8.14.: Flowchart representation of the sequential logic of pre-processing computation

v. Acknowledge Device Firmware.

vi. Allow Treatment Device Calibration.

vii. Start Real-time Computation.

After the Q1 treatment app accomplishes all the above listed tasks, it starts the real-time computation. All above listed tasks are as mentioned in Fig. 8.14 on page 162 are described below in the following sections.

I. **Clear QSTM Temporary Workspace**

The QSTM Temporary workspace is a folder *"QSTM_TEMP"* in QSTM workspace of the PC that stores all temporary QSTM data, that includes process information, system information and treatment data of each sub-session and the complete treatment session as mentioned in section 8.4.1. Hence, it is essential to clear all previous data from this file to write new incoming data of

the on-going treatment during the real-time computation of the treatment app. Therefore, this task has the highest priority and is performed right at the beginning of the pre-processing computation.

II. **Configure Treatment Visualization**

The structure of the visualization screen has already been discussed in section 8.5.1.1. The visualization screen is initialized according to the GUI structure mentioned above. The graph widgets hold almost 75% of the visualization screen from the left while the table widget and the Force monitor takes up the rest 25% of the visualization screen.

The Force Graph widget is initialized with four fixed length data queues to visualize real-time force applied to the skin along X, Y and Z directions and along with the resultant of the 3D forces.

The Geo-Angle graph widget is initialized with data queues for the Yaw, Pitch and Roll angular data measured with respect to gravity. The Skin-Angle graph widget is also initialized similarly with data queues for Yaw, Pitch and Roll data measured with respect to skin.

The Table widget is initialized with 3 columns with treatment parameter specifications. They are parameter name, parameter value and parameter representations units. The parameter details will be further explained later in this chapter.

The characteristic functions of the graph widgets for auto-scaling and grid view are also initialized during this step of pre-processing. The default visualization timer to display the graphical data is also set-up here in this step. This timer updates every 1 millisecond. Also, the paths and filenames to store the QSTM sub-session data for post-processing is also determined here in this step.

III. **Store Treatment Application Process-ID**

The Process-Id is a process information, which is generated by the Windows operating system, when any standalone PC application starts running. The

operating system generates a unique process-id for every application running on it to keep track of the hardware resources used by the corresponding application. Hence, the Q1 Treatment App requests the operating system to provide its own process-Id. Once the treatment app knows own process-Id, it writes a copy of process-Id to the process information file, *"processQ1Data.csv"*, of the QSTM temporary workspace.

IV. **Check Q1 Device ID**

This process is primarily performed to ensure that the serial communication is established between the correct treatment device and the treatment app on PC, especially when multiple devices are connected to the same PC. This process acts as a secondary communication check between source and destination. The intermediate tasks involved to perform this check includes the following:

(a) *Retrieve Device Registration ID* − The Treatment app locates the COM port of the connected Q1 device and retrieves the registration Id generated by the parent application from the system information files stored in QSTM temporary workspace. This registration ID is further used to match the USB properties of the COM port with its device registration Id.

(b) *Parse Device Registration ID* − The device registration Id is originally generated at the Device Auto Detection Process at the QSTM GUI (parent application) during the pre-treatment Idle mode of the system. This device registration-Id is a combined QSTM device-Id and the COM port number.

(c) *Match COM Port Attributes* − Finally the Device registration Id is further parsed into each of its 16bit USB vendor ID, USB product ID and Device name. The treatment app requests the operating system of PC to enlist the USB hardware attributes of the respective connected COM port. The enlisted USB hardware-ID is matched with the 16bit USB vendor-ID and

16bit USB product-ID for verification. Once this verification is successful then the treatment app ensures to send an acknowledgment message to the treatment device through the connected COM port.

V. **Send Acknowledgement To Device Firmware**

After the successful verification of the USB COM port attributes with the device Registration-Id, the treatment app sends an acknowledgement to the treatment device. This acknowledgement message consists of the verified QSTM Device-ID of the treatment device, which was originally sent to QSTM GUI (parent application) during the device auto detection process of pre-treatment Idle mode of the system. If the acknowledgement message is approved at the device firmware, then the treatment device further re-acknowledges the Treatment app of PC with an "ACK OK" message. If the "ACK OK" is not received, it the treatment app raises a "Device Connection Error" message. Once the "ACK OK" is successfully received by the treatment app, Treatment app allows the QSTM Q1 treatment device to perform the Device Calibration Process.

VI. **Allow Treatment Device Calibration**

On successful receival of "ACK OK" message, the treatment app sets itself to slave receiver mode, i.e., it accepts all forms of serial communication messages from the treatment device. At this instance, the treatment device performs the device calibration procedures and constantly transmit Calibration messages to the treatment app. These calibration data are further stored at the treatment data files of the QSTM temporary workspace for later use during treatment data analysis.

VII. **Start Real-Time Computation**

The device calibration step takes around 4 seconds to be successfully completed. The treatment app looks for a particular message "Treatment Mode Active" to be transmitted from the treatment device at the end of device calibration step.

This message ensures the Treatment App that all calibrations are completed. The next subsequent messages received from the device would be the QSTM message string for real-time computation and visualization.

8.5.2.2 Tasks Of Real-Time Computation

During this phase of the treatment mode, the PC behaves as the Slave while the Q1 treatment device acts as the Master in a Master-Slave communication model. The Q1 Treatment App (Child Application) constantly receives QSTM Message Strings from the Q1 treatment device. Each of these messages follow the following execution sequence for real-time visualization and post processing.

i. First the Software Timers are initialized before the start of the real time communication.

ii. Each of the incoming QSTM messages is a serial string and processed for data classification and formatting

iii. Then the formatted data are held in FIFO buffers for display.

iv. When data elements are visualized, they are automatically stored in charts of *.csv* format.

v. If the message is the device reset request triggered by the user by device button press, then it performs processes of the Device Pause State and post processing on stored treatment sub-session chart data. An alternate device reset condition terminates the Device Pause State.

vi. When the user terminates the Q1 treatment App, by closing the visualization screen of Q1 Treatment App, the real time computation process ends, followed by post processing of the treatment session chart data.

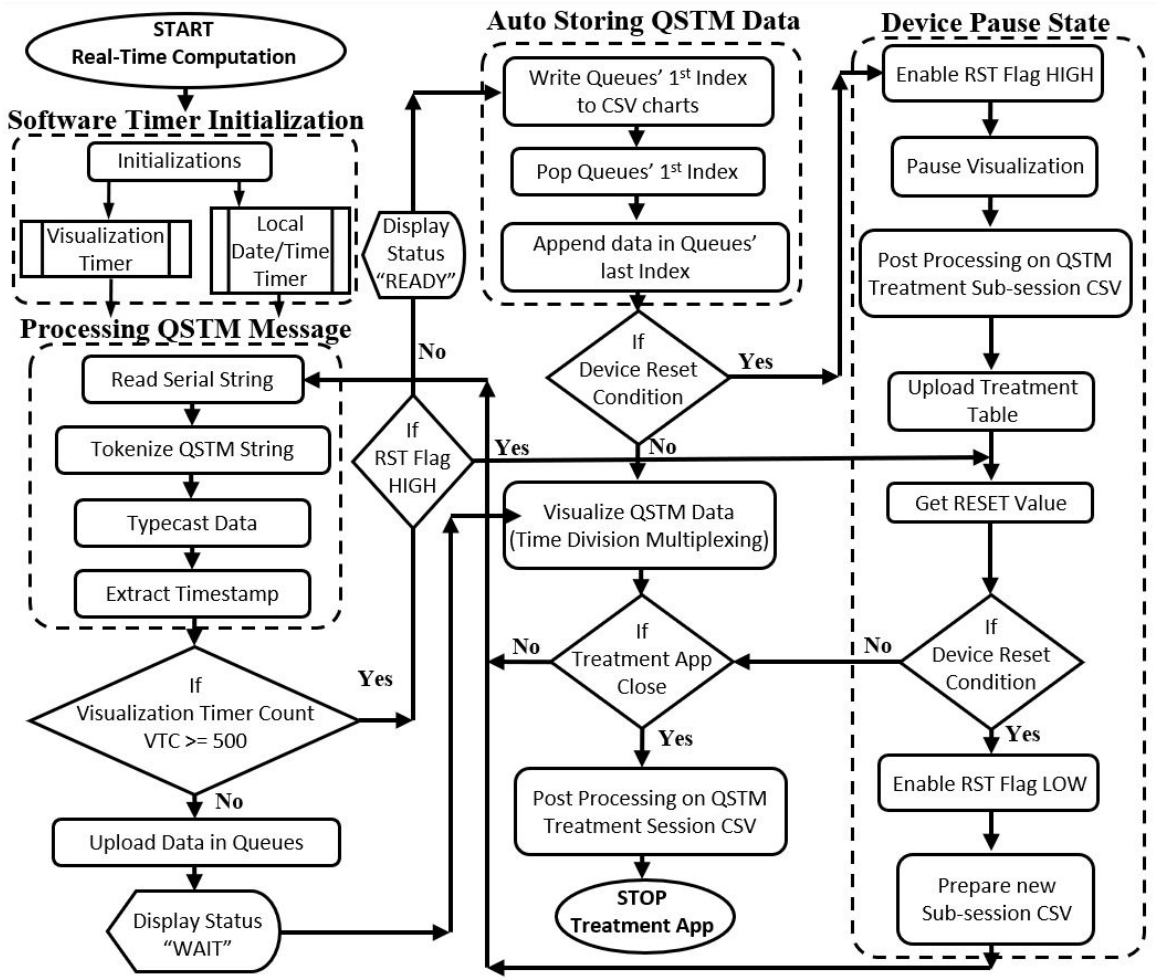


Fig. 8.15.: Flowchart representation of the computation sequence for Real-Time Computation

All these processes are represented in a flowchart in Fig. 8.15 on page 167, explaining the processes of the real-time computation of the Q1 Treatment App during the treatment mode of the system. A detailed description of all the processes and their corresponding tasks in the above algorithm are given below.

I. **Software Timers Initialization**

The two main software timers used in this Q1 Treatment App (Child application) are Visualization (clock) Timer and Local Date Timer. The Visualization Timer is responsible for displaying the real-time data on the screen, while the

Local Date/Time Timer is used to generate the timestamp of the local time of the PC set to a resolution of 10^{-9} seconds. Both the software timers are illustrated below:

(a) *Visualization Timer (Wall Clock)* As from Section 8.3, the Q1 treatment app is written using PyQT4 software development visualization framework, hence the timing module of PyQT4 is harnessed to configure the visualization timer. The inbuilt QT-Clock, which is a wall clock timer, represents the Visualization Timer of the treatment app and is set to update every millisecond. However, several process counter variables have been introduced to track the tick and tock of this visualization timer based on the tasks of real-time computation. This is the primary timer or program counter which updates all processes of the Q1 treatment App (child application).

(b) *Local Date/Time Timer (Elapsed Time)* The software development environment i.e. Python 2.7 supports a Time Library which extracts the UNIX Epoch time [48] [49] and provides several objects which returns time in different formats. For dates and local-time it refers to the deviation of the time zone from the Greenwich Mean Time. This is vehemently used to represent filenames of the treatment sub-session data files. But for processing, it returns the CPU time i.e. time elapsed by the process from start of its function call till the end of the execution in nanoseconds. This elapsed CPU time is captured as the timestamp for reading every incoming QSTM message string.

Both of these timers are initialized right at the beginning of the real-time computation. The starting epoch time for initiating the real-time computation is generated here. This is later used to get time stamps in seconds. Moreover, the visualization timer triggers the retrieval of incoming QSTM message string,

processing them and uploading them to data buffers for real-time visualization. The sequence of these processes continues unless it is interrupted by a user request to close the visualization screen of the Treatment App.

The upcoming sections focusses on discussing the various task involved in real-time computation. The steps in the following subsections from B to E are executed for every QSTM message.

II. **Processing QSTM Message String**

The QSTM Message String contains a total of twelve elements of QSTM raw data characters in series separated by a blank-space delimiter. These data are first read as a String by the serial port receiver, tokenized into its respective categories and then type casted into readable data formats for future processing and finally stored into their respective data buffers (queues) for data display on visualization screen. All these processes are illustrated below:

(a) *Read Serial String* − The incoming QSTM messages are received by the USB serial port decoded and read as a String message varying from 80 to 100 bytes in length depending on the data payload.

(b) *Tokenize String* − The twelve elements of the QSTM message string in series from the first till the last index are : QSTM Forces in X, Y and Z direction and their resultant force; followed by QSTM Geo-Angles Yaw, Pitch and Roll; concurrently followed by the Yaw, Pitch and Roll of Skin-Angles; the RMS acceleration of the accelerometer; and finally the Reset Value of the Q1 device. As all these data are separated uniformly by a single character blank-space, hence it is easier to tokenize with the blank-space delimiter.

(c) *Type Cast Data* − All delimited data are first converted into their respective readable formats. Most delimited data are floating point numbers and hence these data are converted from String to unsigned Float data for-

mats. An exception to this includes the delimited data on the last index of the QSTM message string i.e. the device reset value. This reset value is converted to signed 8-bit Integer data format.

(d) *Extract Timestamp* − The local date/time wall clock timer is responsible for generating the time stamps in UNIX epoch time as mentioned earlier. A function named *"time.time()"* is used to extract the present time (in seconds) which returns a floating point number. This function is called iteratively to generate successive UNIX epoch timestamps on every update of the visualization timer. However, the subtraction of the last epoch timestamp, generated in every iteration, from the initial epoch timestamp, generated at the start of real-time computation, yields the elapsed timestamps in seconds up to 9 decimal places.

These type casted QSTM data are stored in respective data buffers to plot the corresponding data in graph widgets of the visualization screen. These data buffers expand in length with incoming data for the first 500 elements. Once the expansion of buffers exceed 500 elements, then *"pop()"* and *"append()"* methods are used to transform these data buffers into fixed length FIFO buffers i.e. data queues. This is performed to maintain a constant flow of data for a running visual display in real-time as explained below in the next section.

III. **Auto-Storing QSTM Data In Temporary Space**

The Data queues are the fixed length FIFO buffers to hold corresponding formatted data elements of QSTM message string for real-time graphical visualization and data storing for post processing. There are13 such queues, each maintaining queue length of 500 elements. These queues categorize QSTM data into a cascade of charts in the order of − Time series chart (1 queue), Force charts (4 queues), Geo-angle Charts (3 queues), Skin-angle charts (3 queues), RMS acceleration Chart (1 queue) and finally QSTM device reset chart (1 queue).

The default visualization timer triggers filling of all the 13 data queues for the first 500 iterations. A conditional check statement is used to verify the Visualization Time Counter (VTC) exceeding 500 counts. The corresponding processes of the conditional check statement are explained as:

(a) *Uploading Data In Queues* − The formatted data elements are inserted in their respective data queues unidirectionally for the first 499 iterations of the visualization time counter (VTC). On every count up of the visualization timer, the lengths of data queues expand by one and the data is displayed on the visualization screen using the time division multiplexing algorithm (will be described in next section). From visual observation, every plot on each graph widget expands during this time window, followed by a "WAIT" Display Status.

(b) *Copy The* 1^{st} *(oldest) Index Of Each Queue To CSV Charts* − Once the visualization time counter reach 500 counts, the queue lengths stop expanding. The formatted elements are inserted into and dropped from the queues following the first in-first out FIFO logic from count 500 onwards. The Display Status changes to *"READY"* which indicates now the device can be used by the user. This step extracts the data values stored in the first index of each of the 13 queues in a comma separated string and write them in two separate *.csv* charts. These two *.csv* charts include the subsession chart and the total treatment session chart, which are eventually used for data analysis in post processing computation.

(c) *Pop Queues (De-queue)'* 1^{st} *Index* − The *"pop()"* method is used to drop the element stored in the first index of all 13 QSTM data queues once the comma separated string is written to respective *.csv* files. This reduces the length of the queue by 1 element and the start index shifts to the 2^{nd} element. This makes room for one incoming data element to be inserted into the queue at its last index for consecutive iterations.

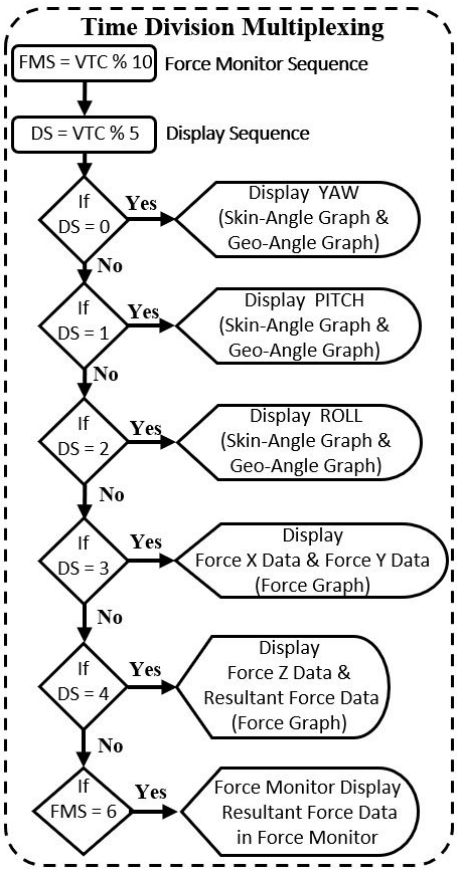


Fig. 8.16.: Flowchart representation of Time Division Multiplexing for data visualization.

(d) *Append Data In Queues' Last Index* – When the first element of the queue is dropped the successive formatted data element is filled into the last index of the queue. This process helps in retaining the original length of the data queues.

The *"pop()"* and *"append()"* methods are exclusively performed to avoid buffer overloading, computational redundancy and sustain a synchronized data flow in real-time throughout the real-time computation of the treatment mode.

Every QSTM data queue, except the reset queue, is plotted against the time series queue on their respective graph widgets on the visualization screen using a special time division multiplexing algorithm explained below.

IV. **Visualize QSTM Data (Time Division Multiplexing)**

A time division multiplexing algorithm is introduced to enable synchronized graphical display of more than 5000 data points on every iteration i.e. every frame and reduce visual latency due to extended computation cycles. This algorithm replicates a form of parallel processing for visualization and distributed computation overload.

The visualization timer, which updates at a refresh rate of every millisecond is harnessed to generate the clock sequences. The visualization time counter (VTC) is calculated at every update (one millisecond) of the visualization timer. For graphical display a total of 10 data queues, each with 500 data points, are plotted against the time series data. The multiplexing algorithm is structured such that, 2 data queues i.e. 1000 datapoints are updated on every millisecond, with the rest 8 data queues i.e. 4000 data points remaining constant. Hence, updating all the 5000 datapoints is executed at 5 successive milliseconds using the time division multiplexing algorithm.

Fig. 8.16 on page 172 shows how the display sequence (DS) is generated by calculating the remainder of the visualization time counter when divided by 5.

Another Force Monitor Sequence (FMS) is yielded by calculating the remainder of the visualization time counter when divided by 10.

The Display Sequence (DS) yields integer outputs in the range of 0 to 4. Henceforth, the algorithm follows a multiple conditional check statements for every output of the display sequence.

(a) If display sequence value is 0 then it updates the YAW queues of both skin-angle graph and geo-angle graph.

(b) If display sequence value is 1 then it updates the PITCH queues of both skin-angle graph and geo-angle graph.

(c) If display sequence value is 2 then it updates the ROLL queues of both skin-angle graph and geo-angle graph.

(d) If display sequence value is 3 then it updates the Force X and Force Y queues of the Force Graph.

(e) If display sequence value is 4 then it updates the Force Z values and Resultant of all 3 force queues on the Force Graph.

When the Force monitor sequence (FMS) yields 6 i.e. on the 6^{th} iteration every 10 milliseconds the numeric value of the resultant force is updated on the force monitor.

This multiplexing algorithm optimizes the real-time latency and sustains the frame rate of real-time visualization on screen. The next section describes how the device reset condition affects the real-time computation and how the Device Pause State is enabled or disabled on the PC software.

V. **Device Pause State**

This section explains what device-reset condition on the Q1 treatment app is, what action triggers it and what are the sequential processes that follow up as a result of this this device reset condition to execute the device pause state of the treatment mode.

Device Reset Condition – A device reset is triggered by the user either the reset button press on Q1 device or a flicker gesture of the device as explained in Chapter 7. The change in the device reset value either from 0 to 1 or from 1 to 0 is captured as the device reset condition. So, the treatment app in its real-time computation method checks whether the reset value in the previous iteration is equal to the present reset value on every update of its visualization timer.

However, a reset flag (pause flag) is introduced into the real-time computation method to retain the knowledge of the device pause state. Whenever the device reset condition occurs, the system enters the device pause state and waits for next device-reset to exit the pause state. The sequence of different process execution of the device pause state as shown in Fig. 8.15 on page 167 is explained below:

(a) *Make Reset Flag "HIGH"* − As soon as the device reset is detected in the real-time computation method, the reset flag is assigned with a HIGH value. This marks the end of one treatment sub-session.

(b) *Pause Visualization* − The real-time upload of data are ignored. The graph widgets displays the set of data points last updated before the reset flag was set to HIGH value. All incoming data from the treatment device, except the reset value, are dropped without processing. Also, writing data to both sub-sessions data chart and total sessions data chart is halted. The writer function, that writes data to the sub-sessions data chart, is terminated. This means that that the corresponding sub-sessions chart data for last sub-session is ready for data analysis.

(c) *Post Processing On QSTM Sub-Session Chart* − The post processing usually involves analyzing QSTM sub-session chart data with the QSTM data analysis library to calculate all the QSTM treatment parameters to generate a treatment report. Since the Q-Ware uses object-oriented programming, it invokes corresponding methods from the QSTM data analysis library to yield QSTM treatment parameters for the last performed sub-session. The QSTM treatment parameters and the sequence of data analysis using the QSTM data analysis library will be discussed later in this chapter.

(d) *Upload Treatment Table* – Once the QSTM parameters are calculated for the last sub-sessions data, the treatment table is uploaded with corresponding QSTM treatment parameters, which includes:

i. Average compressive force i.e. (force perpendicular to skin Z direction).

ii. Average Resultant force i.e. (resultant of all 3D forces).

iii. Maximum Peak of resultant force attained.

iv. Average of all peak forces of all strokes.

v. Total Number of Strokes.

vi. Total Number of Bursts (Sustained continuous train of treatment strokes)

vii. Total Stroke Frequency.

viii. Average Skin Pitch angle.

ix. Average Skin Yaw angle.

x. Average Skin Roll angle.

xi. Total elapsed Time of sub-session.

(e) *Get Reset Value* – After Updating the Treatment Table on the visualization screen, the device actually enters the device pause state and waits for an alternate device reset condition to occur. During this process it receives the device reset value from the incoming QSTM message string and checks if it differs from the last device reset value to satisfy a device reset condition on every successive iteration.

(f) *Make Reset Flag "LOW"* – Once an alternate device reset condition is satisfied then it changes the reset flag status from HIGH value to LOW value. This marks the end of device pause state and a new treatment sub-session is about to start.

(g) *Prepare New Sub-Session CSV* – The reset flags status-change confirms that the system gets back to the Device ready state of the treatment mode. Hence at the start of the new sub-session, certain initializations need to be made which include:

i. Preparing a filename for new sub-session chart to write next sub-session data. This filename is prepared by accessing the local date and time format from the local date/time timer.

ii. Setting the treatment parameters to default values and clearing treatment memory for next sub-session data.

At the end of this the Q1 treatment app resumes processes of the device ready or device working states of the treatment mode and continues real-time computation.

VI. **Treatment App Close**

The Treatment app is terminated only when the user manually close the Treatment app visualization window. When this event occurs, a function call terminates all methods of the Treatment app visualization window, all software timers and closes the serial communication to the USB COM port. The final task is to analyze the prepared treatment session chart and calculate overall treatment report to display on the report panel of QSTM GUI (Parent application). Writing QSTM data to the Treatment session chart also terminates.

Post Processing On QSTM Treatment Session Chart – Once the treatment app visualization is closed, the overall treatment session chart is then analyzed by methods of the QSTM Data Analysis Library to yield treatment report parameters. A separate graphical window pops up, right after. An analyzed Force graph is displayed on this separate graph window before wrapping up the treatment mode. This displayed Force graph, displayed in this separate graphics window, is marked with its valleys and peaks of every stroke applied during the treatment.

Once the user closes this separate graph window, the Q1 Treatment app (child application) destroys its process-id and flushes all variables from the memory and shuts down. The treatment mode comes to an end and the system goes back to the post treatment idle mode for treatment report observation and report saving at the QSTM GUI (Parent application).

8.5.2.3 Tasks Of Post-Processing Computation

Post-processing computation signifies data analysis or processing on a collected dataset. The post-processing is performed, on every device pause state of the treatment mode, to display calculated treatment parameters for every treatment sub-session on the parameters table of the Q1 Treatment App. These processes also compute the treatment report at the end of the treatment session. The main goal is to filter the QSTM raw data saved in csv charts, calculate treatment parameters from the treatment sub-session charts, and finally yield a treatment report from the overall treatment session chart.

The important tasks involved in calculating QSTM treatment parameters are:

1. **Reading Data From Stored CSV Charts** – CSV chart data stored by the Q1 treatment app, in *QSTM_Temp* folder in *.csv* format files, are read by the Data Analysis Library for further processing.

2. **Data Extraction And Data Slicing** – The Data read in the last step are stored in a multidimensional data array. The force data is extracted from the multidimensional array, stored in data-lists or queues and subjected to threshold based slicing to locate treatment information.

3. **Data Smoothing** – The sliced force data from last step is smoothed to remove high frequency noise but retain frequencies of abrupt tissue irregularities captured during treatment.

4. **Local Peak And Valley Estimation** – The smoothed force data is analyzed to mark local peaks and valleys, which further assists to determine number of Bursts and Strokes of a treatment sub-session and eventually the complete treatment session.

5. **Number Of Treatment Stroke Determination** – After several graphical observations of the Force patterns a novel peak filtering algorithm is developed to determine the number of treatment strokes delivered during the treatment. The local peaks and valleys are passed through this peak filtering algorithm to mark the peaks corresponding to treatment strokes and eventually calculates Stroke frequency during treatment.

6. **Calculating Treatment Parameters For Treatment Report** – So far processed Force data, is used to calculate average data parameters as treatment variables like – Average Compressive Force, Average Resultant force, Average Peak Force; applied during treatment.

   It also calculates the average Pitch, Roll and Yaw angles with respect to gravity frame. The number of strokes are already calculated in the last step.

   The data slicing helps to measure the active time of load delivery to the tissue during treatment. It also keeps track of the time of treatment inactivity and treatment interval during a session i.e. the dead time. The summation of both active and dead times provides the total elapsed time of overall treatment.

These tasks of post-processing computation are all written in functions of the QSTM Data Analysis Library. The next section describes all of the above-mentioned data analysis processes responsible to calculate QSTM Treatment parameters.

8.6 QSTM Data Analysis Processes

The QSTM Data Analysis Library is primarily built to calculate the QSTM treatment parameters during the post processing computation of the Q1 treatment app. As mentioned in the last section, the QSTM data charts stored in *.csv* format is analyzed by functions of this library to calculate the QSTM treatment parameters. The sequential processes involved in executing corresponding tasks by corresponding functions of the library to accomplish calculation of treatment parameters are:

1. ***Reading*** data from CSV charts stored in *"QSTM_Temp"* Folder by Q1 treatment app is read by the function *"csvReadFile(pathFile)"*.

2. The next function helps to ***analyze*** each row element of the chart to calculate treatment parameters as described in tasks of post-processing computation. The name of this function is *"GraphAnalyse(csvRows)"* and it takes a 2D array *"csvRows"* of QSTM Raw data read in the previous step. This function follows a series of task mentioned below:

   (a) ***Extracting*** 4-Dimensional force data and time series data for further processing.

   (b) ***Slicing*** force data based on treatment inactivity marked by states of the treatment mode.

   (c) ***Smoothing*** force data, where The sliced force data is subjected to noise smoothing by kernel based smoothing algorithms. The function that is iteratively accessed to perform force data smoothing is *"getAvgGauss(forceRow)"*. Here, the argument *"forceRow"* is the one-dimensional array of the resultant force data extracted and sliced during the last steps.

   (d) ***Local Peak and Valley Determination*** for force data as a preliminary step to determine Force peaks and treatment strokes.

This function returns analyzed graph data in the form of appended data lists to other corresponding functions of the library to calculate QSTM treatment parameter.

3. ***Filtering Redundant Peaks*** to estimate treatment strokes this is performed on the created arrays of force peaks and force valleys to eliminated redundant peak values. The remaining filtered peaks corresponds to the number of treatment strokes performed treatment. the function which performs the redundant peak filtration is *"humpReduction()"*. This function takes local peak array, local peak array timestamps, local valley array, local valley array timestamps to eliminate redundant peaks.

4. ***QSTM Treatment Parameter Calculation*** where finally, the QSTM Treatment parameters for each treatment sub-session and eventually the overall treatment session are calculated. The function which calculates treatment parameters of treatment sub-session data is named as *"calculateVisTable()"*, while the function calculating treatment report parameters of the treatment session is *"calculateReportTable()"*.

This is how the data analysis of the QSTM raw data chart is performed methodologically in a sequential step by step manner. The systematic explanation of all these processes are covered in the upcoming sections.

8.6.1 Read CSV Chart

The CSV library of Python is harnessed to read CSV files of QSTM treatment sub-session data and QSTM treatment session data respectively. At the start of data analysis, the stored CSV file in *"QSTM_Temp"* Folder stored by Q1 treatment app is subjected to a *"csvReader()"* function. This function belongs to the CSV library which reads the data and parses the comma separated data and stores them into data lists of the length of the file. This length of file corresponds to the number of rows

contained in the file. Each row contains data in the sequence of time data, force data, geo-angle orientation data, skin-angle orientation data, RMS acceleration, device reset data. Out of these, the force data and the time data are further manipulated and analyzed to calculate parameters like treatment strokes and bursts.

8.6.2 Extract Time And Force Data

The CSV chart read in the last step is read as a multidimensional array of $L \times 13$ dimension, where L is the length of the file (number of rows) and 13 is the number of columns. The Time data column which contains the timestamps, the force data columns which contains the 3-Dimensional force data and the Resultant force data are separated as lists from the other CSV file data columns of the multidimensional array. The memory spaces for these data lists are made globally available to other functions of the QSTM data analysis library for successive processing. This data extraction is performed in the function *"GraphAnalyse(csvRows)"*, where *"csvRows"* forms the multidimensional array passed as an argument to the function.

8.6.3 Slice Force Data

This operation exclusively focuses on the resultant force data list and the time series data list. The data slicing operation is classified into two categories:

(a) Data slicing on treatment session chart based on number of device resets.

(b) Data slicing on treatment sub-session chart based on time of treatment inactivity.

This slicing performed by the data anlysis library, helps in determining the number of Treatment Bursts (train of sustained continuous treatment strokes). Additionally, this further helps in finding out the local maxima (peaks) and minima (valleys) of the resultant force data from which the treatment strokes are calculated.

A. **Data Slicing On Session Chart)**

This operation is performed by extracting the first and last timestamps of the start and stop event of all device pause states, hence finding all sub-sessions. From previous discussions, it is evident that the device pause state is enabled and disabled by alternate device reset activity. The present and previous reset values are monitored iteratively, to locate row-indexes of corresponding timestamps for change in reset value. Henceforth, the start and stop timestamp of device pause separates the dead-time of the treatment session chart. The forces of the resultant force-data list, corresponding to the dead-time, are excluded from any further processing, thus retaining only the treatment sub-session data into the processing pipeline.

B. **Data Slicing On Sub-Session Chart**

To remove the sensor noise, the sliced sub-session data are further processed. Since the load cell noise level of the treatment device for force quantification is set at 1 Newton's, as described in Chapter 7, the resultant force values less than the cut-off and their corresponding timestamps, are added to the time of treatment inactivity included in the treatment dead-time of the session and thus are discarded from further analysis. The remaining force values corresponds to the active time of treatment in a treatment sub-session. Hence the active time and dead time of treatment are determined as a result of data slicing operation.

The next section includes the detailed discussion on Force data smoothing to remove high frequency force data noise and retain the force oscillations of treatment to determine force patterns of treatment strokes.

8.6.4 Force Data Smoothing

The force data sliced in the last step are subjected to spurious data due to skin and device friction or tissue irregularities or sensor signal distortion which creates data noise. Our research adopted Gaussian smoothing techniques on force-time charts to

reduce noise by the function getAvgGauss(forceRow), where forceRow is the sliced resultant force data list . The justification to choose time series approximation over frequency domain smoothing is discussed in this section. Additionally, it discusses about transformation of continuous Gaussian kernel to discrete Gaussian using binomially distributed rows of Pascal triangle. Finally, the main implementation for smoothing along with some observations are mentioned below.

The term smoothing in signal processing implies suppression of noise or redundant data points to extract useful information from an input signal to derive to the desired output. Smoothing can either be done by a time series analysis or a frequency series analysis. In our research the signals that need smoothing involves especially the resultant of all the three forces measured by the load cell, including the individual X, Y and Z forces of the same.

The output signals from sensors suffer signal attenuation, distortion or unwanted spikes due to the system, transmission channel, processing error, electromagnetic effect or any other factor. Since the load cell used in our research is a pre-amplified 3D load cell, it carries some amount of signal amplification noise added with some A/D conversion noise. Hence when the RMS (Root Mean Square) of the force channels are calculated it includes the noise elements in every channel, which need to be removed from the signal without attenuating the original signal. In a frequency domain analysis, it has been observed that noise mostly involves high frequency elements in a signal. So, a discrete time low pass filter or a band pass filter would definitely serve the purpose if the cut-off frequency for the noise band is known. Hence, coming up with a choice of filtering to smooth the given signals was a great challenge.

8.6.4.1 Frequency Domain Filtering Method

Several digital filter designs are based on frequency domain analysis in which a time series data is converted to its frequency components using Fourier Transforms, to visually detect repetitive patterns of the signal. A concise and comprehensive algo-

rithm to perform this is Fast Fourier Transform (FFT) which reduces the computation cost from $O(n^2)$ to $O(n \log n)$. This approach works best for a periodic time series data. More the samples, more accurate is the analysis. A time series observation at the no-load condition of the 3D load cell is necessary to detect the noise frequency components using Fast Fourier Transform (FFT). Once this is performed, then a Low Pass Filter (LPF) algorithm can be applied to suppress frequencies above the cut-off frequency while passing the ones lower than that. In real case a first order LPF is marked by the cutoff frequency along with its attenuation steepness from cutoff frequency at the log scale in a bode plot. Higher the order more is the gradient of attenuation steepness, better is the filter response. Butterworth Filter is a good example of a typical low pass filter, whose n^{th} order type yields better response, where $n > 5$.

This approach is a suitable approach for noise reduction and if performed at the pre-processing stage during device calibration can highly attenuate higher frequencies. However, the noise generated by the load cell is aperiodic and the processing unit of the device doesnt involve an additional storage unit to save an extracted time series data for frequency domain analysis. In conclusion, FFT is an additional iterative step on a stored time series, followed by n^{th} order Butterworth filtering which needs (n) successive iterations. Hence this approach needs a thorough simulation to analyze the memory power and computation complexity involved for implementation in the real-time scenario, without affecting the corresponding computations.

8.6.4.2 Time Series Approximation Method

The foremost approach used for low pass filtering and smoothing a signal is a *running average filter* which is also known as a rolling mean filter. Since we are dealing with one dimensional data, this approach is a time series approximation of signal values acquired with respect to time, where the arithmetic means of $(n-1)$ past data values are calculated with the present data value.

However, an extension to this method represents a statistical approach which involves weighted average filters with an approximation function determining the weights of the filter of size (n). This type of smoothing can also be extended to form kernel based smoothing estimators. The weights of such estimators take over outputs from a real valued function that is convoluted with time series data of neighboring points to yield a weighted mean. This smoothing kernel is run throughout the time series data stream, suppressing the unwanted spikes and smoothing the signal with respect to the approximation function.

Running Average Kernel − This approach has a fixed n number of inputs, where n is the size of a queue or window, as known in signal processing. The present value is inserted at the end of the queue while the value contained in the first index is dropped out in every iteration to achieve a running average output. An analysis with respect to kernel estimator shows that this approach basically uses a kernel with uniform weights equivalent to unity, whose sum is equal to the window size. Statistically it uses a uniform distribution and hence running average filter is also called uniform weighted filter or box filter.

Other kernels used for smoothing [50] are exponential, sine, cosine, triangular, gaussian, local regression etc. In our research gaussian estimators with different approximations has been applied and tested to smooth high frequency data while retaining the original pattern of the signal. The gaussian kernel estimator has been briefly described below.

Smoothing Gaussian Kernel − In statistics, a continuous distribution is said to be gaussian if its probability density function $f(x|\mu, \sigma)$ takes up the following equation where x is the domain, μ is said to be the mean, σ is the standard deviation and σ^2 is the variance of the distribution.

$$f(x|\ \mu,\ \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{8.1}$$

A special version of this gaussian distribution is called a standard normal distribution, if its mean $\mu = 0$ and standard deviation $\sigma = 1$, then the equation for $f(x)$ reduces to $X \sim \mathcal{N}(\mu,\, \sigma^2) = X \sim \mathcal{N}(0,\, 1)$, which is:

$$f(x|\, \mu,\, \sigma) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{8.2}$$

However, the gaussian kernel iterates this function $f(x|\mu, \sigma)$ from its initial index to its final index i.e. x varies from 0 to $n-1$, to obtain a discrete gaussian curve of size n weights. Here, n is the size of the kernel to generate gaussian weights, which in turn are convoluted with neighboring noisy data points to yield smoothed output. The size of the kernel is intentionally taken to be an odd integer so that the highest weight is achieved at $(n/2)^{th}$ index of the kernel. Thus, the normal distribution takes the shape of $\mathcal{N}(\mu,\, \sigma^2) = \mathcal{N}((n/2),\, 1)$, where n is the kernel size, and the present data value along with adjacent $(n-1)$ past data values are convolved with the weights. The equation given below forms the basis of the gaussian kernel using the normal distribution formula.

$$Gaussian\ Filter_{Output} = \frac{\sum_{x=0}^{n-1} \left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\frac{n}{2})^2}{2\sigma^2}} \times Data_x \right)}{\sum_{x=0}^{n-1} \left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\frac{n}{2})^2}{2\sigma^2}} \right)} \tag{8.3}$$

However, implemented versions of the Gaussian kernel looks similar to a discrete Gaussian distribution with n samples, the more the size of the kernel the more bell shaped is the curve. the implementation of this filter is described in the following section.

8.6.4.3 Smoothing Implementation And Analysis

The initial concept of a smoothing kernel with discrete weights, analogous to a bell-shaped curve, was adopted from closely studying the properties of a Pascals triangle. Its row elements closely imitates the coefficients of a binomial expansion $\binom{n}{k}$ whose formula is given by:

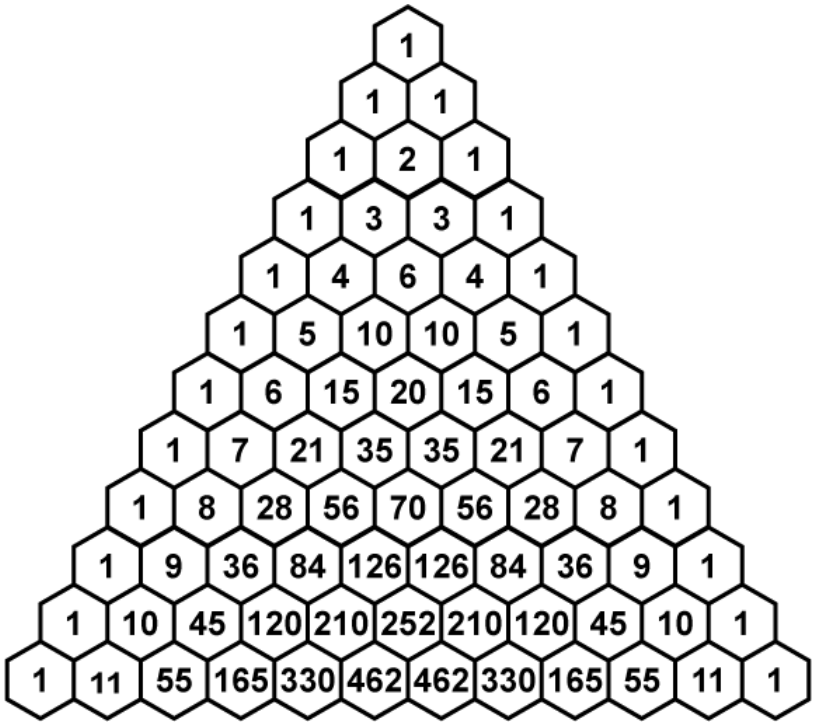


Fig. 8.17.: Pascals Triangle with $n = 11$

$$(p+q)^n = \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} \tag{8.4}$$

The Fig. 8.17 on page 188 shows a Pascal's triangle, whose row elements from 0 to n takes up values of combinations $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, where n is the row size and k is the element index of the row. By the characteristic property of the probability mass function, a binomial distribution $B(n, p)$ with varying k from 0 to $n-1$ reaches its maximum value at $k = n/2$, where n is the number of Bernoulli's trials, p is the probability of success, q is probability of failure and an important condition $p = q$ is satisfied, where $p + q = 1$.

Due to this analysis an observed variation of the Pascal's triangle is marked by the exponential increase in the magnitude of central elements of every row as the row length tends to infinity. Each row of the triangle sums up to $2n$, thus maintaining the area under the curve equivalent to 1. The mean of the distribution $\mu = np$, while

the standard deviation of the distribution $\sigma = \sqrt{npq}$. Since p and q are equal or equivalent to $\frac{1}{2}$, therefore, the standard deviation σ varies half times the square root of n, i.e. $\sigma \propto \frac{\sqrt{n}}{2}$, with increase in row length. This variation is harnessed to scale the exponential increase of the central values by a normalizing factor (β), to limit to a convergence point [51] for any value of n, retaining the area under the curve as shown in Fig. 8.18 on page 189.

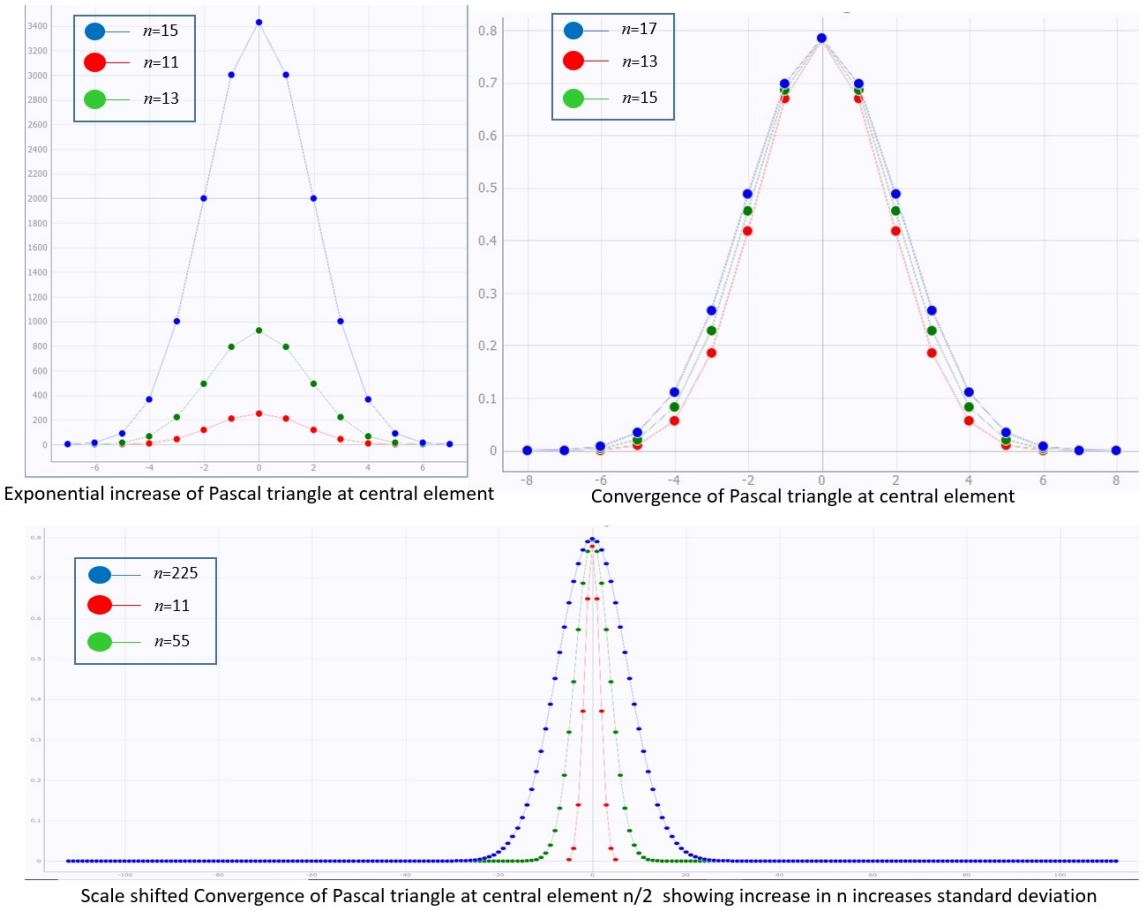


Fig. 8.18.: Graphs from Pascals Triangle Analysis

The normalizing factor $\beta = \frac{\sqrt{n}}{2^n}$, has been derived from Stirlings formula of convergence and characterizes the varying factor of standard deviation i.e. $\sqrt{n}$ and the total sample space of the distribution i.e. 2^n. The derived formula for smoothing using discrete Gaussian window generated from binomial distribution or rather Pascals triangle is:

$$Binomial\ Filter_{Output} = \frac{\sum_{k=0}^{n}\left(\binom{n}{k} \times \frac{\sqrt{n}}{2^n} \times Data_k\right)}{\sum_{k=0}^{n}\left(\binom{n}{k} \times \frac{\sqrt{n}}{2^n}\right)} \tag{8.5}$$

Eqn. 8.5 forms the basis of the discrete Gaussian kernel generated from n^{th} row of the normalized pascals triangle. The main advantage of this method is that it involves two parameters n and k for implementation which is much more memory efficient for real-time computing. Moreover, a Gaussian kernel consists of 99.7% of its values within three standard deviations from its mean. Hence, the window size should be wisely chosen according to the amount of smoothing needed.

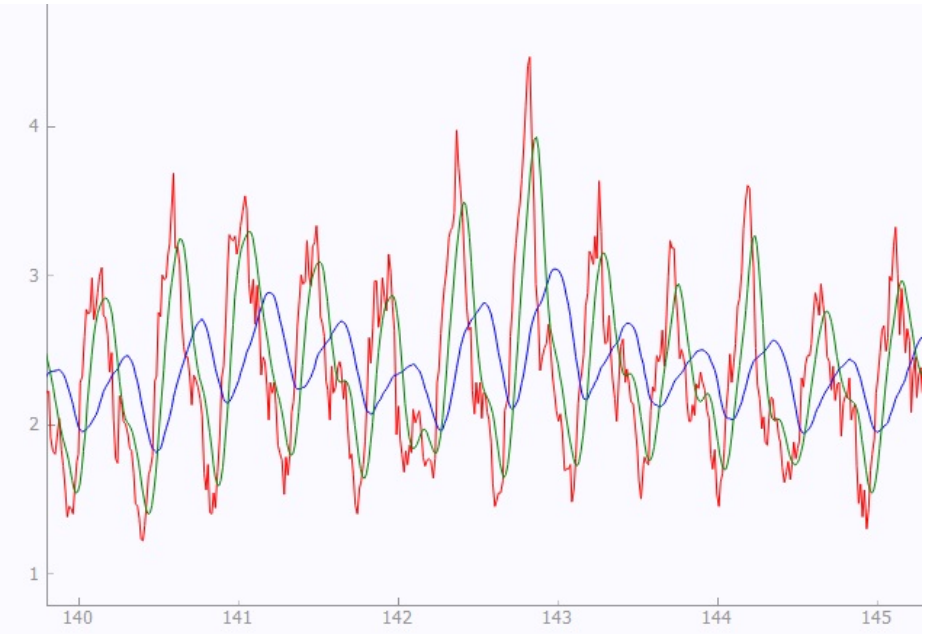


Fig. 8.19.: Kernel smoothing using both Uniform and Gaussian kernals

Fig. 8.19 on page 190 shows a comparative study of smoothing resultant force with respect to a uniform distribution in blue and Gaussian distribution in green, while the red chart is the original resultant force chart. A visual observation of this chart shows that the kernel with uniform distribution fails to retain original data at higher frequencies, while Gaussian kernel replicates a smoothed variation of the original signal for all frequencies. However, the Gaussian kernel fails to smooth constant sustained forces over time where, the kernel with uniform distribution acts best for this scenario. This observation proves that there is a necessity to alter the bandwidth [52] of the Gaussian kernel to obtain low frequencies during sustained forces and high frequencies during heavy device movement. Hence, larger values of standard deviation of a Gaussian distribution would help to obtain smoother estimates [53] clipping fluctuations while moderate values would retain the peaks. This analogy leads us to generate a sensor fusion model where the bandwidth of the filter would vary based on the fluctuations of linear acceleration captured by the accelerometer. So, the bandwidth would be inversely proportional to the acceleration, as more the fluctuation in accelerometer during force application, moderate is the bandwidth, the lower the fluctuation, the more is the bandwidth.

Hence our research finally adopted the technique of time series smoothing of noise due to force channels using smoothing filter with Discrete Gaussian i.e. binomial kernel shown in Eqn. 8.5.

$$RMS\ Force_{Smoothed} = \frac{\sum_{k=0}^{25}\left(\binom{25}{k} \times \frac{\sqrt{25}}{2^{25}} \times Force(RMS)_k \right)}{\sum_{k=0}^{25}\left(\binom{25}{k} \times \frac{\sqrt{25}}{2^{25}} \right)} \tag{8.6}$$

The kernel used is a discrete binomial kernel of window size $n = 25$. The weights of the window are highly dependent on the standard deviation of the distribution function, which is directly proportional to smoothing the curve. Here the standard deviation for binomial distribution is dependent on the window size. The higher the standard deviation the smoother the curve.

This smoothed resultant force values are further analyzed to determine the local force peaks and local force valleys discussed in the next section. Additionally, implemental approaches for running average box filters and continuous Gaussian kernel with normalized weights has also been made to check smoothing accuracy of filters which will be discussed in the results chapter.

8.6.5 Determine Local Force Peaks And Valleys

To calculate treatment strokes in a treatment, local force peaks (maxima) and valleys (minima) of the resultant force data in the smoothed resultant force chart need to be determined. This is performed by the function *"humpReduction()"*. Initially, the resultant force takes up values within the load cell noise level which is less than 1 newton. So, the treatment chart always starts with a rising force curve. The Resultant force at every index of the resultant force list is fed as the input into this algorithm to determine whether the force at that index is a local maxima or minima. So, the algorithm operates iteratively for k iterations where k starts from 0 to the length of the list. Initial conditions of all checking variables that update iteratively are set to zero. There are two conditional check statements which compares the resultant force magnitudes of the present and the last iteration to determine the slope of the curve as shown in Fig. 8.20 on page 193. Following the conditional checking, the change in the slope of the curve determines the local peaks or valleys. The corresponding timestamps of the respective indexes in time series data list are also marked and recorded in a peak-timestamp array and valley timestamp array. The details of the algorithm is explained as follows:

i. The smoothed resultant force at the k^{th} index of the resultant force data list is read and fed to two check statements. Initially the value of force at $k - 1^{th}$ index is set to zero, while the initial value of slope is also set to negative.

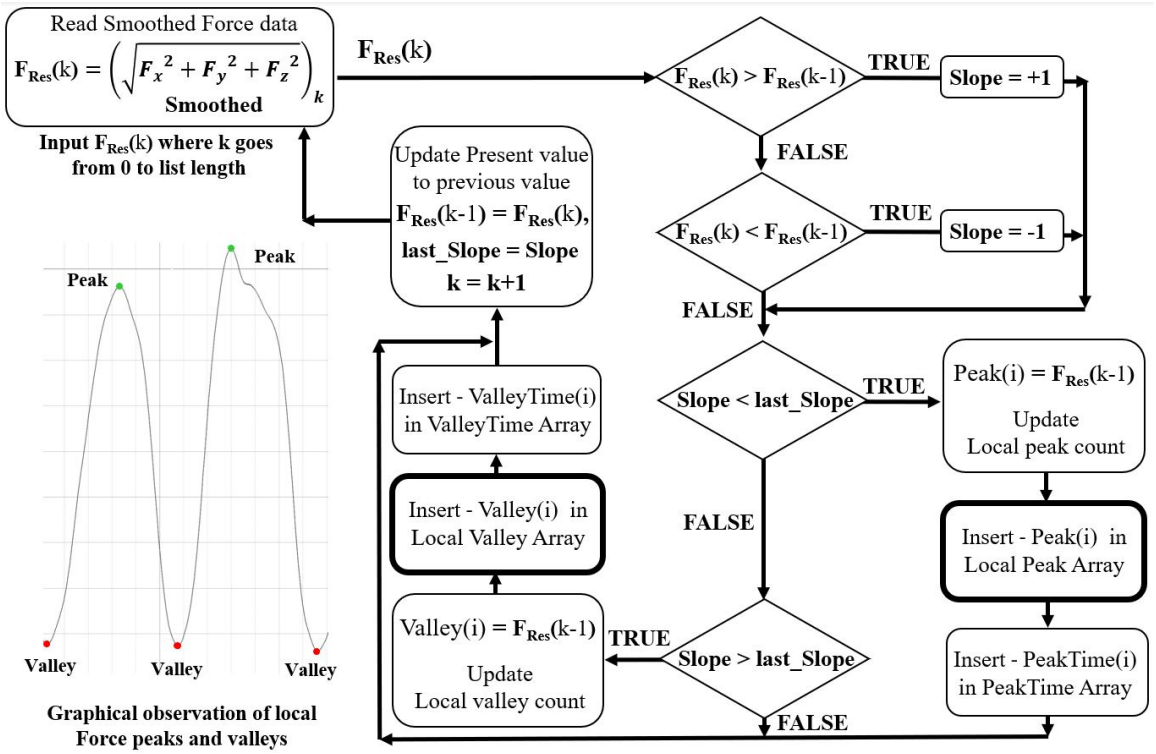


Fig. 8.20.: The flowchart representation of the Force Peak and Valley Determination algorithm

ii. If the value of resultant force in the k^{th} index, i.e. the present iteration, is greater than that of $k-1^{th}$ index, i.e. the last iteration, then the slope of the curve is positive. This denotes that resultant force in the k^{th} index is on a rising curve.

iii. Similarly, if the value of resultant force in the present iteration, is less than that of the last iteration, then the slope of the curve is negative. This denotes the resultant force in the k^{th} index is on a falling curve.

iv. Likewise, the present slope value of the curve in the k^{th} iteration, determined in the last two steps, is compared with the slope value of the curve in the previous $k-1^{th}$ iteration.

v. If the present slope value (k^{th} iteration) is less than the last slope value ($k-1^{th}$ iteration), then the resultant force of the last iteration, which is $k-1^{th}$ index, is marked as the local maxima or data peak. Following this, the peak counter is increased by 1 and this resultant peak force is stored in the local peak array.

vi. The timestamp on the $k-1^{th}$ index in the time series data list is marked as the data peak timestamp and stored in the Peak-Time Array.

vii. Similarly, if the present slope value (k^{th} iteration) is greater than the last slope value ($k-1^{th}$ iteration), then the resultant force value of the last iteration, which is $k-1^{th}$ index, is marked as the local minima or data valley. The valley counter is increased by 1 and this resultant valley force is stored in the local valley array along with its timestamp in the valley-time array.

viii. At the completion of every iteration, the present resultant force value and the present slope value, are updated as their respective previous values for processing the following successive iteration, along with increment of the loop counter k.

These processing steps continue iteratively till all the data peaks (local maxima) and data valleys (local minima) are extracted in their corresponding peak and valley arrays along with their respective peak and valley timestamps. These force peak values may contain peaks due irregularities of tissues, device friction during treatment and some miscellaneous aberrations. Hence it is necessary to filter the unwanted force peaks to calculate treatment strokes obtained during active time of treatment. These strokes are further summed up to obtained total number of strokes in a treatment sub-session and finally for the total session. The difference between the force peak timestamps helps in calculating the stroke frequency of the treatment sub-session.

8.6.6 Filter Redundant Force Peaks

The goal is to find the strokes. According to graphical observations of test data, it has been inferred that the smoothed resultant force curve retains the repetitive stroke patterns performed during a treatment. However, these patterns include multiple force peaks due to bumpy stroke motion of the device caused either by irregularities on the tissue or by the friction of device tip with the skin. Our extended aim is to remove peak and valleys caused by irregularities of the tissue. Hence it is highly essential to eliminate the false stroke peaks from the smoothed force data and retain the actual treatment peaks corresponding to every treatment stroke.

At first, the number of peaks in the local peak array is counted. If the number of valleys in the local valley array doesnt equal the number of peaks, then an additional valley with value 0.5N is added to the array at the last index. Thus, the peak and valley in the rising curve, i.e. $(P1, V1)$ in Fig. 8.21 on page 196, forms a peak-valley pair. As force curve starts from magnitude zero hence, a valley always comes first in the chart.

From implementation perspective every peak is studied closely with two distinct parameters i.e. magnitudes of amplitude differences of the rise curve and fall curve of a peak in between two adjacent valleys. Studying these amplitude differences for force peaks corresponding to a treatment stroke versus that of redundant peaks, we came up with a technique of calculating a confidence ratio feature. This confidence ratio is derived by dividing the rise amplitude difference by the fall amplitude difference of two adjacent valley forces (minima) of a peak force (maxima). This is further used to threshold and eliminate redundant peak forces as discussed by the algorithm.

From experimental values it is evident, if the redundant peak lies on the falling edge, magnitude of the confidence ratio would be less than 0.5, while if it lies in the rising edge then the ratio would be larger than 2. These thresholds are essentially used to characterize a stroke peak. Successive and repetitive analysis of this algorithm enhances the output and indicates all the redundant peaks. The array of these

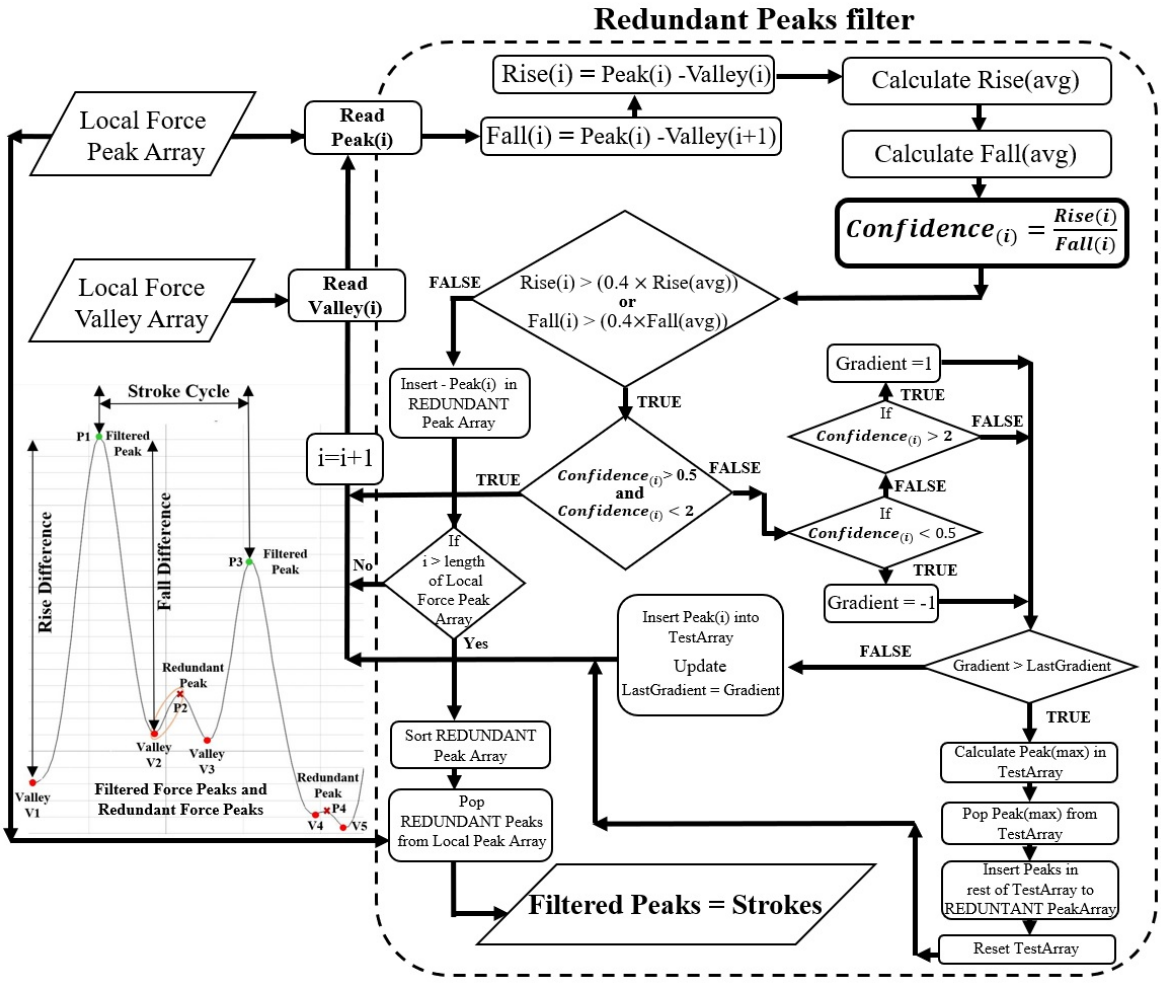


Fig. 8.21.: The flowchart representation of the Redundant peak filter algorithm to determine Treatment Strokes

redundant peaks is sorted and then eliminated from the local peak buffer to obtain only the stroke peaks. Hence the number of these filtered peaks forms the number of strokes in subsequent treatment sub-sessions. The array of filtered peaks can be used to generate the average peak force applied during treatment. This average peak marks the target force attained during the whole massage session. The complete algorithm of redundant peak elimination is experimentally formulated and contributes to another novelty of the research, as discussed in Fig. 8.21 on page 196.

i. The resultant force local peak array and the resultant force local valley arrays are used as input in this algorithm.

ii. Every maxima (peak) is surrounded by two minima (valley). This concept is used to calculate the magnitude of amplitude rise and magnitude of amplitude fall of a force peak from its neighboring valleys. Hence, amplitude rise is calculated by $(P_i - V_i)$ i.e. subtracting force magnitudes of a peak-valley pair, where i indicates the index on the local peak array.

iii. Similarly, the amplitude fall is calculated by $(P_i - V_{i+1})$ i.e. subtracting the magnitude of adjacent force valley from present force peak.

iv. These rise values and fall values are iteratively subjected to calculating their self-averages as the loop count increases.

v. A Confidence-Ratio of the peak curve is calculated for the corresponding rise and fall values of each peak to determine whether it is located on the rising edge or on the falling edge. The confidence ratio is calculated by dividing the amplitude rise value by its corresponding amplitude fall value of a respective peak.

vi. If either of the rise or fall values of the peak exceeds 40% of their present average values, then it enters the thresholding check else the peak is considered to be redundant peak. This step confirms that peak due to a small hill in between two actual strokes is discarded as redundant peak.

vii. If the peak force value satisfies the condition in step 6 then it is checked with its confidence ratio value. If the confidence ratio is within 0.5 to 2 then the peak is considered to be of a corresponding treatment stroke and hence excluded from the redundant peak array else, it is considered for further processing.

viii. If peak force value satisfies condition in step 6 but doesn't satisfy condition in step 7, then the peak is checked for lying on the rising or on the falling edge. If the confidence ratio value is less than 0.5, then it is lying on the falling edge and gradient is negative.

ix. If the confidence ratio value is more than 2, then the peak is lying on the rising edge and the gradient value is positive.

x. The corresponding peaks with confidence ratio values mentioned in step 8 and 9 are accumulated in a test array till the gradient of the curve changes from negative to positive. This is because these peaks are all supposed to be aberrated peaks of a single treatment stroke.

xi. If the gradient of the curve changes from negative to positive, then the test array is processed. The maximum value in the test array is selected and this value is dropped from the array. This maximum value is supposed to be the maximum of all aberrated noise peaks of one corresponding treatment stroke. Hence this peak is considered as the stroke peak.

xii. Once step 11 is done the rest of the peaks in the test array is added to the redundant peak array and the test array is cleared and reset.

xiii. All the steps from 1 to 12 continues till all the peaks in the local peak array are covered checking. Once this is done, the redundant peak array is sorted in descending order.

xiv. The values of these redundant peaks are searched with their respective indexes on the local peak array and these values are dropped. This step filters the local peak array with only force peaks corresponding to every treatment stroke.

xv. Thus the number of filtered peaks provide the number of treatment strokes of the treatment sub-session.

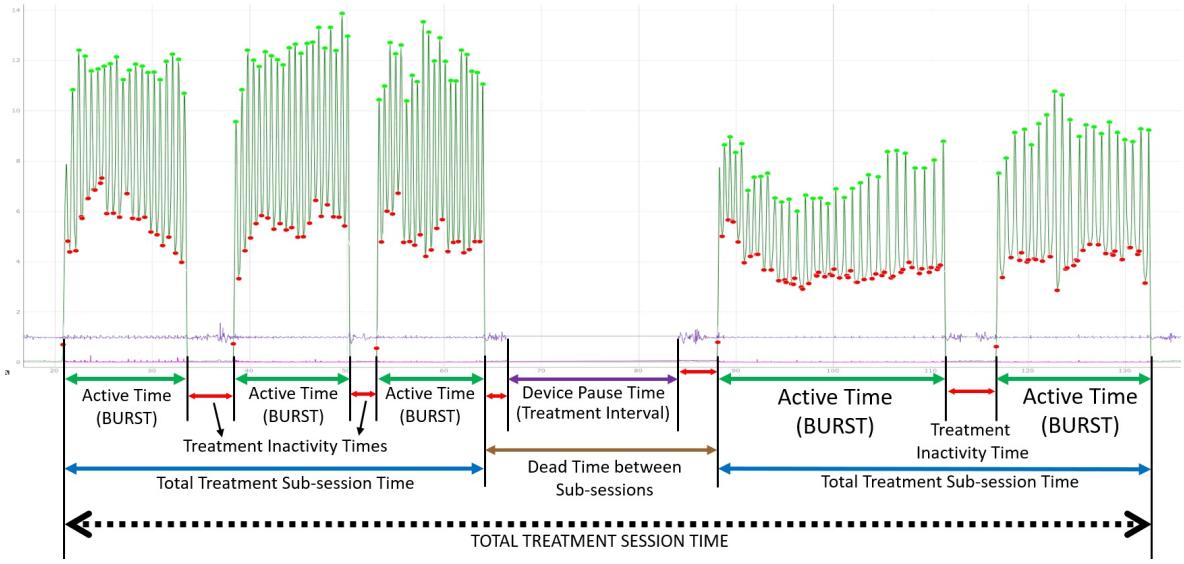


Fig. 8.22.: Graphical representation of the session chart with Active and Dead times of a treatment session

Thus, the filtering algorithm which produces the filtered force peaks approximates the number of treatment strokes performed during the treatment, because each filtered peak corresponds to each treatment stroke. This algorithm has been validated with experimental results discussed in the next chapter.

8.6.7 Calculate Treatment Parameters

The QSTM treatment parameters which essentially characterize a therapeutical treatment sub-session are yielded as a result of the above mentioned data analysis processes during the post processing computation of the treatment mode of the system. This process is also performed at the end of the treatment mode when the complete treatment session gets over.

All these QSTM treatment parameters have been elaborately illustrated in this section along with their mathematical calculations from the recorded chart data.

The time elapsed during the complete treatment mode refers to the treatment session time. While the time in between two sub-sessions are separated by the interval time of the pause state which adds up to the dead time of the treatment. Fig. 8.22 on page 199 shows a detailed diagram of how data is sliced based on the time and which time correspond to active and dead times. The times denoted in red or brown in Fig. 8.22 on page 199 indicate dead times of treatment while the time in green indicate the active treatment time. The device reset value separates the pause state from the device ready and device working state in treatment mode. Therefore, the first and foremost parameter which needs to be recorded is the active and dead time of treatment. These two parameters are figured during the data slicing step of data analysis.

I. **Active Time Of Treatment**

The active time of a sub-session constitutes the device working state of treatment mode, which states the amount of time the device is held with force with actual skin contact treating the soft tissue site. From the sub-session CSV chart, the active time taken into account is considered to be the successive timestamps of resultant force above 1 Newton. In a treatment sub-session, i.e. when device reset value is set to 0, when resultant forces are less than 1 Newton, the corresponding time stamps are accounted into time of treatment inactivity and discarded from the active treatment time.

Treatment Bursts − From visual observations, in a typical treatment sub-session a train of continuous treatment strokes with sustained oscillating forces are followed by a treatment inactivity time. This uninterrupted train of continuous treatment strokes with sustained oscillating forces is termed as a treatment burst. The active time of treatment addresses to each of the treatment burst times followed by treatment inactivity time i.e. dead time of treatment during a sub-session.

Hence these active times of treatment in between treatment inactivity times are summed up to calculate the total active time of a treatment sub-session. Similarly, the active times of treatment sub-sessions are summed to yield the total active time of the treatment session.

II. **Dead Time Of Treatment**

Dead Time Of Treatment Sub-Session − is the device ready state which accounts to the treatment inactivity time, when the resultant force is recorded below 1 newton, within the load cell noise level. This treatment inactivity times in between every treatment burst adds up to form the total dead time of the sub-session.

Dead Time Of Treatment Session − dead times of sub-sessions are added up with the device pause state interval times to form the overall dead time of the treatment session.

III. **Average Resultant Force**

From the order of processed QSTM data written in the sub-session CSV chart or the session CSV chart, the resultant force takes the fifth column in the chart. The data from this column is read during force data smoothing process and smoothed with Eqn. 8.6 on page 191, discussed above. The list of smoothed resultant force data is summed up for N iterations, where N is the number of timestamps recorded for a burst in between two treatment in-activities (dead times). An, elaborate equation for calculating the average resultant force has been given below.

$$(F_{Rez})_{Avg} = \frac{\sum_{i=1}^{N} \left(\sqrt{F_x^2 + F_y^2 + F_z^2} \right)_{i_{smoothed}}}{N} \tag{8.7}$$

This produces the average resultant force for each treatment burst. However, the average of these average resultant forces for all bursts in a sub-session yields average sub-session resultant force, while that of a session yields average resultant force for the total session.

IV. **Average Compressive Force**

The force acting perpendicular to the plane of the application i.e. force along the Z direction of the load cell is known as the compressive force. The average is calculated in a similar way like the average resultant force for every treatment burst where F_z is summed up for N iterations from initial to final timestamps of one treatment burst. The equation is given as follows:

$$(F_Z)_{Avg} = \frac{\sum_{i=1}^{N} \left(F_Z \right)_i}{N} \tag{8.8}$$

The total average compressive force is calculated by calculating the average of average compressive forces for total number of bursts delivered during a treatment sub-session or a session.

V. **Average Orientation Angles**

This marks the average inclination of the device from the skin. $Rot_X(\theta)$, which is rotation around the X axis of the IMU marks the inclination angles from the gravity called Geo-pitch, while $Rot_Z(\phi)$ marks the swing angles of the device rotating around the Z axis called Geo-Yaw as described in section 7.4.3.4 last chapter. The rotation of the device during treatment along the y axis determines the Geo-Roll angles $Rot_Y(\psi)$. The average of all of these are calculated using the similar formula as in Eqn. 8.8. $Rot_X(\theta)$, $Rot_Z(\phi)$ and $Rot_Y(\psi)$ are summed up for N iterations till the terminal timestamp of the activation time of every treatment burst and thus the average is calculated using the following formula.

$$Pitch_X(\theta)_{Avg} = \frac{\sum_{i=1}^{N} \big(Pitch_X(\theta) \big)_i}{N} \tag{8.9}$$

$$Roll_Y(\psi)_{Avg} = \frac{\sum_{i=1}^{N} \big(Roll_Y(\psi) \big)_i}{N} \tag{8.10}$$

$$Yaw_Z(\phi)_{Avg} = \frac{\sum_{i=1}^{N} \big(Yaw_Z(\phi) \big)_i}{N} \tag{8.11}$$

The average pitch and yaw angles provide the approximate inclination of the device during every treatment burst. These averages are summed up and averaged out for the total number of treatment bursts to yield the average yaw and average pitch of the treatment sub-session. Similar calculations are followed to yield the same for overall treatment session.

VI. **Number Of Strokes**

From visual observations every stroke corresponds to a peak force during massage treatment. These peak forces are filtered from noisy redundant forces peaks due to massage friction or tissue irregularity. Hence the total number of filtered peaks filtered during redundant peak filter step equals the total number of treatment strokes.

VII. **Number Of Bursts**

The continuous train of treatment strokes during the active time of treatment constitute a treatment burst (see Fig. 8.22 on page 199). The number of strokes per burst is dependent on the therapist. The timestamps corresponding to change in resultant force amplitude from less than 1 newton to more than 1 newton is recorded as a burst state. These bursts states are added up to yield the total burst count in a sub-session.

A statistical data analysis is performed on specific lists of stored QSTM data from session or sub-session chart to get the average of quantified force and orientation angles of the device used during treatment. All the average parameters are calculated for the corresponding data recorded during the active time of the treatment. Hence corresponding timestamps of every burst (uninterrupted continuous train of treatment strokes with sustained oscillating forces), which addresses the active treatment time, are recorded to extract the sliced data from stored CSV charts for statistical data analysis.

VIII. **Maximum Peak Force**

This refers to the maximum peak of resultant force attained during the treatment sub-session. The local resultant force peak array generated at the end of the determination of local peaks and valley step is used to calculate the maximum force peak. The local force peak arrays contain all the peaks of resultant force. The maximum value of this local force peak array yields the maximum force peak attained during treatment. This maximum peak corresponds to a particular treatment stroke delivered during treatment.

IX. **Average Peak Force**

This average peak force refers to the average of all stroke peaks attained during a treatment sub-session. These peaks are filtered form noisy redundant peaks during the redundant peak filter step. The average of all the filtered stroke peaks forms the average peak force. This is calculated by summing up the force peak values of all filtered peaks and averaging with the number of strokes i.e. filtered peaks. Here P is the number of filtered peaks or treatment strokes.

$$(Peak_{Force_{RMS}})_{Avg} = \frac{\sum_{i=1}^{N} \left(Filtered\ Peak\ Force_{RMS} \right)_i}{N} \tag{8.12}$$

The average peak force attained during treatment helps the clinicians to assess the how much targeted load was applied during the soft tissue treatment.

X. **Stroke Frequency**

The total number of strokes applied during the active time of the treatment produces the stroke frequency in hertz. The difference between the recorded timestamps of two filtered force peaks yields the stroke period.

Alternatively, if the all the stroke period for the total number of strokes can be averaged to yield average stroke cycle of each burst, this average stroke cycle can be used to get the stroke frequency of each burst by dividing total burst time.

This completes description of all the QSTM Parameters that characterize quantifiable soft tissue manipulation (QSTM) treatment. The next chapter discusses about the results and experimental observations performed to implement the system.

9. TEST RESULTS AND OBSERVATIONS

The data requisition QSTM treatment instrument has been developed to augment instrument assisted soft tissue manipulation, in order to study and characterize STM quantitatively with QSTM treatment parameters explained in the last chapter. This chapter emphasizes on different results achieved during several levels of testing which can vary from hardware to software level to determine the precision or accuracy of the treatment device achieved during implementation. It also focuses on the clinical aspects of certain testing based on which an accuracy analysis has been discussed.

9.1 Hardware Testing

This part of the chapter mostly elaborates, the test scenarios involved to develop the hardware of the medical device. It also informs about the challenges faced during implementation with the solutions to solve the issues.

9.1.1 Electrical Tests For Circuit Implementation

Since the 3D load cell uses the analog pins of the processing unit Teensy 3.2 [29] to measure the changes in voltage due to change in resistance, by exertion of force at the load shaft of the load cell; certain electrical tests are essential to determine the electrical circuit needed to be implemented. The load cell can operate both at 3.3V DC and 5V DC. The 3.3V DC will provide lower force resolution than that of 5V DC powered to the sensor. The load cell has been tested in both the scenarios, and the voltage readings of the zero-state no-load condition of the load cell is measured by digital multimeters to enable proper voltage value to force conversion. The diagram shown in Fig. 9.1 on page 207, demonstrates the experimental setup to test the load

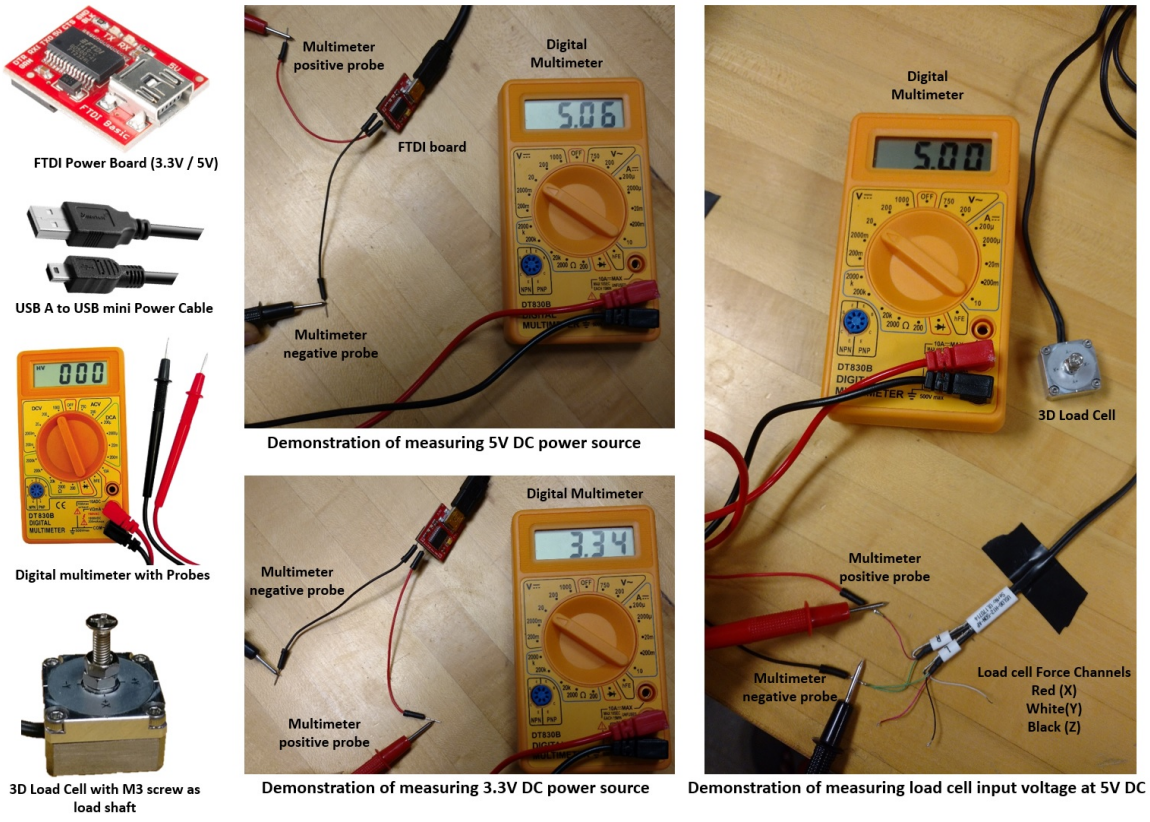


Fig. 9.1.: Demonstration of the equipment set-up to measure experimental voltage readings of the load cell

cell with voltages (5V DC and 3.3V DC), where FTDI boards [32] of both driving voltages has been used to test the same. Table 9.1 on page 208 shows the no load voltage output of the load cell at different driving voltage.

These voltage measurements vary from sensor to sensor or load cell to load cell. Therefore, it is recommended to do these measurements to validate the activation voltages of manufacturer datasheets. Thus, this preliminary test analysis is essential to determine the working efficiencies of individual channels of the 3D load cell. The observed results produce a 15% additional gain in the Z channel at 5V which infers that the Z channel will read up-to 85% of its maximum level at its voltage upper limit. This shows that there is a 15% data cut off and a 100N 3D load cell [23] will read only up-to 85N at its highest voltage. These measurements establish the usefulness

Table 9.1.: Determination of Error Rates in efficiency measures of Offset Voltage at Zero State No-Load Condition

Measure	**No-Load Readings (3.3V)**	**No-Load Readings (5V)**	**No-Load Measurements of Data-Sheet by Manufacturer**		**Error Rates in Efficiency Measures of Offset Voltage**	
Power Source	3.3V DC	5V DC	3.3V Dc	5V DC (recommended)	3.3V DC	5V DC
Force X Channel	2.18V	2.67 V	1.65	2.5V	12%(gain)	7%(gain)
Force Y Channel	1.11V	2.31V	1.62	2.45V	31%	6%
Force Z Channel	2.14V	3.0V	1.71	2.6V	23%(gain)	15%(gain)

of a voltage divider circuit in this case and based on these data the resistances values for the voltage divider circuit needs to be determined. But for an ideal case, the resistances needed for 5V to 3.3V conversion are mentioned in the schematic diagram. For measured no-load force values as mentioned in Table 9.1 on page 208, trial and hit method is used to test and come up with adequate resistances that maintain the operating range of the load-cell intact. The following section shows the force validation measured on a Newton's scale. This is used to determine the scaling factors as mentioned in section 7.4.2.4 in Chapter 7.

9.1.2 Force Validation In Newton's Scale

The voltages measured at the calibration stage to calculate the offset for individual channel activation is further used in the voltage to force transformation using the manufacturer calibration matrix provided in the manufacturer datasheet. Since the implementation uses a 100N load cell, the calculation involved in Eqn. 7.9 on page 104 from section 7.4.2.2 is linear combination of calibration matrix from manufacturer

with the measured difference of load voltages from the calculated offset. Since the input power voltage of 3D load cell used in this research is 5V DC, the zero force at no-load condition in X, Y, and Z directions are supposed to be 2.5V DC, 2.45V DC and 2.6V DC, respectively, as mentioned in datasheet. The magnitudes of the calibration matrix for a 100N load cell with the voltage to force transformation equation is given below as implied to our implementation.

$$\begin{bmatrix} FH_X \\ FH_Y \\ FH_Z \end{bmatrix} = \begin{bmatrix} 25.5468 & -1.3139 & -0.1773 \\ -0.0929 & 25.3307 & 0.6875 \\ 0.1754 & 0.7355 & 49.8882 \end{bmatrix} \times \begin{bmatrix} V_X(load) - 2.5V \\ V_Y(load) - 2.45V \\ V_Z(load) - 2.6V \end{bmatrix} \tag{9.1}$$

This equation (Eqn. 9.1) or the values provided at the calibration matrix by the manufacturer holds true when the measured offsets corresponds to the no-load activation voltage mentioned in the manufacturer datasheet. As per the implementation, voltage to force transformation follows Eqn. 7.9 from section 7.4.2.2. mentioned in Chapter 7. A Force validation table is provided below with forces of different force

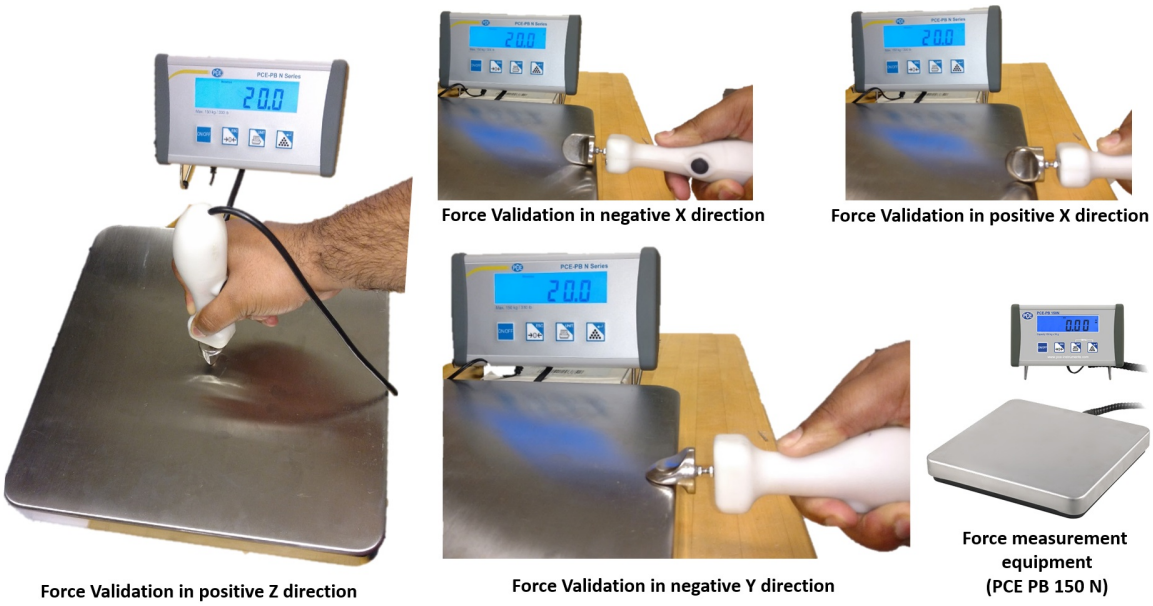


Fig. 9.2.: Demonstration of the equipment set-up to validate calculated Compressive and Translational force components in Newton's scale

Table 9.2.: Force Validation test om a Newton's scale for Force Scaling

Measured Force Readings (Newton's)		***Calculated Force Readings of 3D Load cell (Newton's)***				***Error Rate***
Newton's Scale		**Force X Channel**	**Force Y Channel**	**Force Z Channel**	**Resultant Force**	**% Error**
30 N	-X	**-30.31**	-2.01	4.84	30.76	2.53(gain)
	+X	**27.81**	0.68	1.60	27.86	7.13
	-Y	-0.49	**-27.21**	3.96	27.50	8.33
	+Z	-0.23	-0.75	**28.61**	28.62	4.6
20 N	-X	**-19.85**	-0.84	2.42	20.02	0.1(gain)
	+X	**20.82**	-0.07	2.08	20.91	4.55(gain)
	-Y	-1.72	**-17.64**	1.24	17.76	11.2
	+Z	0.34	-0.53	**19.08**	19.09	4.55
15 N	-X	**-15.08**	-0.06	3.37	15.45	3.00(gain)
	+X	**16.29**	-1.29	3.73	16.76	11.7(gain)
	-Y	2.53	**-13.35**	4.45	14.29	4.73
	+Z	0.12	0.21	**14.43**	14.44	3.73
10 N	-X	**-9.29**	-0.64	1.12	9.38	6.2
	+X	**9.32**	-0.25	1.93	9.52	4.8
	-Y	-1.66	**-7.83**	1.23	8.1	19
	+Z	1.22	-0.73	**8.83**	8.95	10.5

channels measured for 30N, 20N, 15N and 10N, respectively. This table also provides an error rate between measured forces and calculated force readings. The measured force readings are taken against an electronic force scale (PCE-PB 150 N with a resolution of 0.5N) as shown in Fig. 9.2 on page 209. This states the necessity to perform a manual force validation testing of the transformed forces of individual channels of the load cells on a Newton's scale.

For validation of lower force magnitudes, it is observed that the percentage error increases. This percentage error can be compensated by scaling the transformed smoothed force magnitudes yielded as a result in section 7.4.2.3 in Chapter 7. The smoothed forces are scaled with experimental proportional gains to manipulate calculated forces and approximate them to measured forces in Newton's scale as mentioned in section 7.4.2.4 in Chapter 7.

The experimental data observed in Table 9.2 on page 210 corresponds to the contribution of calculated forces of individual channels of the load cell towards the resultant force generation. The error rates of every channel are averaged to yield the average % error due to each channel which are - 5.84% for the load cell Z Channel, 7.26% for the load cell negative Y Channel and 1.08% for the load cell X channel (RMS % error of both negative X and positive X channels), respectively. This scaling step helps in reduction of error rate below 2% by introducing scalar proportional gains (α, β, and γ) for X, Y and Z channels' forces respectively, to scale the calculated Newton's. The values of these proportional gains are completely determined by performing several trials of the observations mentioned in Table 4. In our implementation the value of $\alpha = 0.98$, $\beta = 0.92$, and $\gamma = 0.94$, respectively. These values are experimental values and might change due to prolonged usage of the device and hence re-validation is necessary after every 6 months of usage.

The following section discusses the orientation angle validations of the device particularly directing towards the Geo-Pitch and Skin-Pitch angle validation of the device with respect to the gravitational reference frame as well as the skin (treatment) plane.

9.1.3 Tilt Angle Validation

This section focusses on the validation of the tilt angles (in particular Pitch) of the treatment device measured with respect to both the earth's horizon as well as an arbitrary skin (treatment) plane. A set of goniometers have been used to perform the validation of the Skin-Pitch from an inclined skin (treatment) plane, while the validation of Geo-Pitch used a square set instrument.

9.1.3.1 Geo-Pitch Angle Validation

The Geo-Pitch angle of the Treatment device is validated using a developed square set used by Ahmed Alotaibi et. al. [2]. The Pitch angle of the treatment device has been validated with respect to oriented angles measured by the square set by fixing the device on the set square ruler as shown in the Fig. 9.3 on page 213.

The Table. 9.3 below shows the error rate of the calculated tilt angles with respect to the measured tilt angles. The tilt angles, i.e. Geo-Pitch, with respect to earth's horizon has been measured for particular orientation angles especially 0°, 30°, 45°, 60°, 90°, respectively as shown in Fig. 9.3 on page 213.

Table 9.3.: Geo-Pitch angle validation test using a square set

Measured Pitch Angles in Degrees	**Calculated Pitch Angles in Degrees (X-Axis Rotation)**	**Pitch Error Rate %**
0	0.06	6
30	30.5	1.66
45	44.67	0.73
60	60.15	0.25
90	88.20	2

Fig. 9.3.: Demonstration of the equipment set-up to validate calculated Geo-Pitch angles of the treatment device

The calculated pitch angles from Table. 9.3 shows the efficiency of the tilt angle measurement system of the treatment device with an average error rate ±2.12%. The implementation of other orientation angles i.e. Yaw and Roll angles, has been performed using the proportional integral complementary filtering approach as mentioned in section 7.4.3.4 of Chapter 7. However, these Yaw and Roll angles are not validated with any measuring scale but are observed on the basis of anecdotal evaluations. This states a need to design a test rig to calibrate all angular measurements in any spatial orientation based on the gravitational reference frame.

Moreover, the approximation of the device orientation measurements from an arbitrary skin plane discussed are solely dependent on − holding the device orthogonally to the treatment site of skin at the start of treatment. This provides a measure for the Angular orientation of the device (Yaw, Pitch and Roll) with respect to gravity called Geo-Angles (reference). These Geo-Angles (reference), measured at that particular orthogonally held orientation, are recorded to derive subsequent orientation angles with respect to skin as mentioned in section 7.4.3.5 in Chapter 7. The approximation for skin angles is specifically done for the Skin-Pitch and Skin-Yaw angles respectively. The roll angles measured with respect to gravity equals that measured with respect to skin. Therefore, there is not much effect of the roll angles with respect to skin even if the skin is aligned in an arbitrary plane.

9.1.3.2 Skin-Pitch Angle Validation

A set of dual experiments have been performed to study the goniometer readings corresponding to tilt orientation of the treatment device, when the treatment plane is aligned with the earth's horizon as well as when the treatment plane is inclined to the earth's horizon. The skin-reference angle measurement plays an important role to estimate the measured skin-pitch angles with respect to the calculated geo-angles.

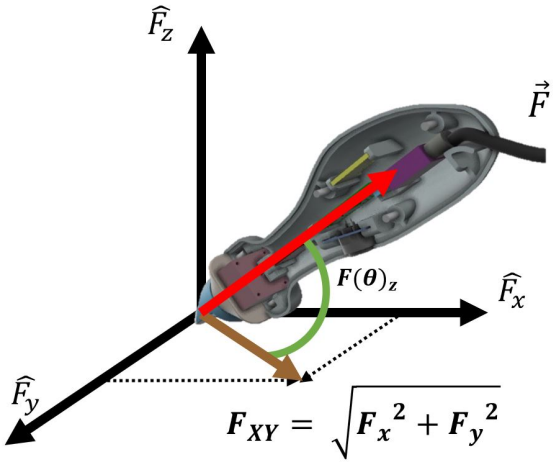


Fig. 9.4.: Angular orientation of Compressive Force's unit vector component from Treatment Plane

Skin Reference Angle Measurement – An additional compressive force reference angle parameter has been introduced to measure the skin-pitch reference angle, from where successive calculations are made to yield skin-pitch angles. This force reference angle, i.e. $F(\theta_Z)_{(ref)}$, constitutes the angle made by the unit vector of the compressive force component (F_z) with respect to the force (X-Y) plane as shown in Fig. 9.4 on page 215. When the treatment device is held perpendicularly applying a compressive force of around 1-3 Newton's to the plane of treatment, to measure the skin reference angle, this force reference angle and the corresponding Geo-Pitch angle are recorded as reference angles for successive Skin-Pitch calculations.

$$F(\theta_Z) = \tan^{-1}\left[\frac{\hat{F}_Z}{\sqrt{(\hat{F}_X)^2 + (\hat{F}_Y)^2}}\right] \times \pi \tag{9.2}$$

Table 9.4 on page 216 demonstrates the relative errors of the Compressive Force Angle $F(\theta_Z)$, made by the compressive force component, from the calculated skin-pitch, when the treatment device is held perpendicular to the treatment plane aligned

Table 9.4.: Skin reference angle validation at 90 degree when skin plane is aligned to earth's horizon

Angle	**Calculated Force Components (Newton's)**				**Calculated Pitch Angle**			**Error Rate**	
Measured Pitch (Degrees)	$\mathbf{F_X}$	$\mathbf{F_Y}$	$\mathbf{F_Z}$	$\mathbf{F_{RMS}}$	**Force Angle** $\mathbf{F(\theta_Z)}$	**Geo Pitch (Degrees)**	**Skin Pitch (Degrees)**	**Geo-Pitch Relative Error %**	**Skin-Pitch to Force Angle error %**
90	0.23	-0.35	4.94	4.96	89.59	86.84	88.77	3.51	-0.92
	0.26	-0.36	4.99	5.01	89.55	87.08	89.01	3.24	-0.6
	0.26	-0.41	5.11	5.14	89.44	87.52	89.45	2.76	0.01
	0.26	-0.47	5.22	5.25	89.40	87.90	89.82	2.33	0.46
	0.27	-0.53	5.26	5.30	89.27	87.84	89.77	2.4	0.55

to the earth's horizon, applying uniform force of 5 Newton's. As skin pitch to force angle error % are within (±1)%, hence this compressive force angle parameter is used to determine the skin-pitch (reference) angle.

Therefore, the instantaneous skin-pitch angle is the calculated acute angle of, difference between instantaneous Geo-Pitch angle and Geo-Pitch (reference) angle, with respect to the compressive force reference angle, $F(\theta_Z)_{(ref)}$. Henceforth, when the treatment device is held perpendicular to a skin-plane of any arbitrary inclination to the earth's horizon, by applying 1N to 3N of compressive force; if the compressive force angle measure exceeds 85 degrees then it is considered to be the compressive force reference $F(\theta_Z)_{(ref)}$ angle to calculate corresponding skin-pitch. Then the corresponding Geo-Pitch angle is recorded for successive skin-pitch calculations. The following Eqn. 9.3 is used to calculate the corresponding skin pitch angles.

$$Skin_{Pitch} = F(\theta_Z)_{(ref)} - \left(Geo_{Pitch(Ref)} - Geo_{Pitch} \right) \tag{9.3}$$

The next section illustrates the experiment performed to validate the Skin-Pitch calculated with respect to treatment plane aligned to the earth's horizon,which is almost equivalent to the calculated Geo-Pitch angle.

Skin-Pitch validation when skin-plane is earth's horizon – This experiment has been performed to validate the fact that Skin-Pitch angles almost align with the Geo-Pitch angles when the treatment plane is the earth's horizon. An important factor for skin-pitch measurement is holding the treatment device perpendicular to the treatment plane, which if tampered might produce erroneous results. In this experiment, the calculated compressive force angle reference is 86.04 degrees, for skin

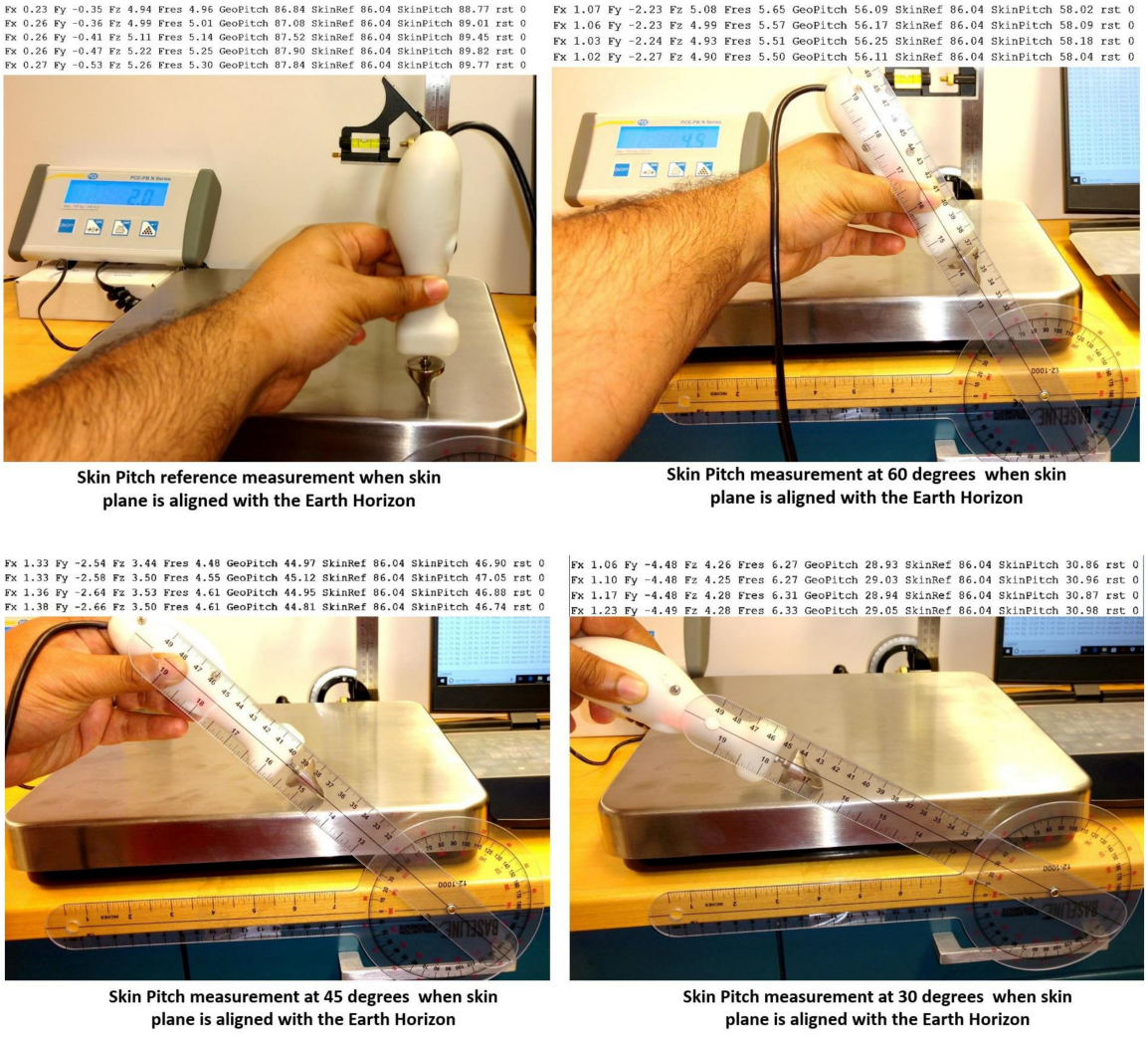


Fig. 9.5.: Skin Pitch angle validation when Skin (treatment) plane is aligned with the Earth's Horizon

Table 9.5.: Skin-Pitch angles validation when skin plane is earth's horizon

Angle	Calculated Force Components				Calculated Pitch Angles		Error Rate	
Measured Pitch (Degrees)	F_X	F_Y	F_Z	F_{RMS}	Geo-Pitch (Degrees)	Skin-Pitch (Degrees)	Geo and Skin Pitch Difference	Skin-Pitch error %
30	1.23	-4.49	4.28	6.33	29.05	30.98	1.93	3.26
45	1.33	-2.58	3.50	4.55	45.12	47.05		4.5
60	1.03	-2.24	4.93	5.51	56.25	58.18		3.03
90	0.26	-0.47	5.22	5.25	87.90	89.82		0.2

angle measurements. However, angle validation of 30°, 45°, 60° and 90° are performed in this experiment as shown in Table 9.5 on page 218, corresponding to Fig. 9.5 on page 217.

Table 9.6.: Skin Pitch angle validation when skin plane is inclined to earth's horizon by 20 degrees

Angle	Calculated Force Components				Calculated Pitch Angles		Error Rate	
Measured Pitch (Degrees)	F_X	F_Y	F_Z	F_{RMS}	Geo Pitch (Degrees)	Skin Pitch (Degrees)	Geo and Skin Pitch Difference	Skin-Pitch error %
30	0.02	-3.59	3.44	4.97	5.79	29.52	23.73	1.6
45	0.15	-2.87	4.52	5.36	19.53	43.26		3.86
60	0.40	-1.10	2.86	3.09	37.07	60.79		1.31
90	0.62	-0.63	6.81	6.87	66.86	90.59		0.65

Performing increased number of trials for every angle should provide more specific error rates, that might vary depending upon the skin-pitch reference measured when device is held perpendicular to the treatment plane. This statement is true for any arbitrarily inclined treatment plane within $0^\circ - 85^\circ$ inclination from earth's horizon.

Skin-Pitch validation when skin-plane is inclined to earth's horizon – A similar experiment has been performed with a set of goniometers to measure skin-pitch angles when the skin (treatment) plane is inclined at an angle of 20 degrees (see Fig. 9.6 on page 220). The measured skin-pitch reference i.e. compressive force reference angle measured is 87.83°, when held perpendicular to the 20° inclined skin (treatment) plane from the earth's horizon.

Table 9.6 on page 218 illustrates the error readings of the calculated skin-pitch angles with respect to the measured skin-pitch angles where the average error rates lies within ±2% tolerance. Observations proves that the measured skin-plane inclination from the horizon as compared to the calculated inclination varies by more than 15%. These measures provide an approximate estimation of the skin-pitch angle validation, which specifies the need to design specific test rigs to improve the precision of skin-pitch calculation. Although this is a major contribution of the research, it needs to be fine-tuned with better precision in future versions. Precision measurements of Geo-Pitch for increased number of trials would give an estimate of the errors for the present implementation while optimization measures needs to be adopted to reduce relative errors.

The next section discusses different aspects of software testing with respect to the algorithms developed to implement Q-Ware. It describes the effects of smoothing with different time series kernel filters used in the implementation, along with the efficiency of the Stroke count algorithm.

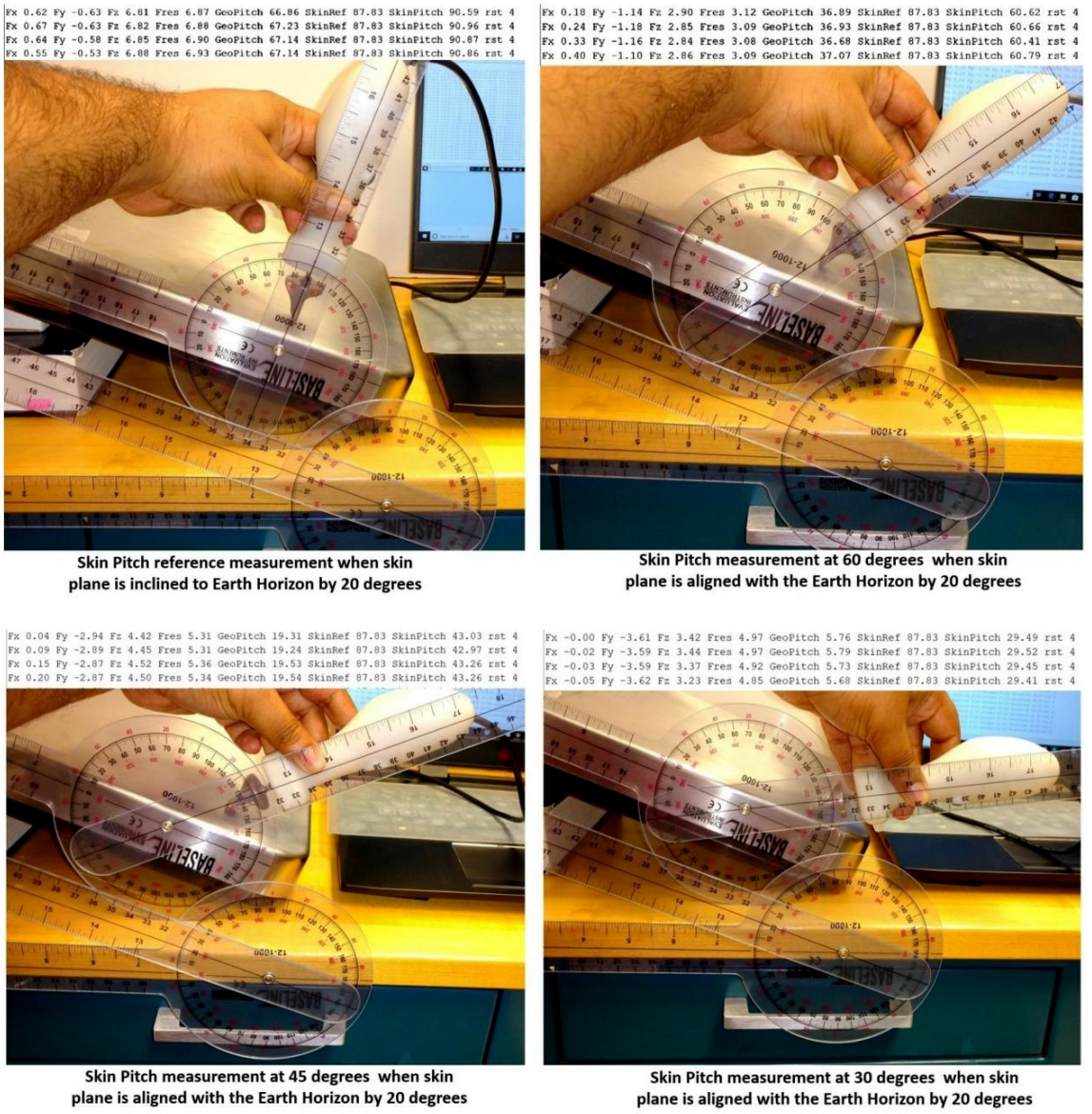


Fig. 9.6.: Skin Angle validation when Skin (treatment) plane is inclined to Earth's Horizon by 20 degrees

9.2 Software Testing

The QSTM software is developed to calculate and measure all the QSTM parameters for characterization and generalization of Treatment Session. Moreover, an analysis of the synthesis models for smoothing with Uniform distribution, Binomial

distribution and Gaussian distribution-based kernels are performed to specify a recommended method. Additionally, some observations are needed to be discussed to come up with a sensor fusion based validation algorithm for strokes from accelerometer and gyroscope data. Last but not the least, this section shows the accuracy of novel stroke count algorithm developed solely for QSTM data analysis with respect to manually counted strokes in several clinical trials performed on human subjects.

9.2.1 Analysis Of Smoothing Synthesis Models

This section emphasizes on the effects of Uniform distribution kernel and Binomial distribution kernel derived as discrete Gaussian smoothing kernel with different sizes of the filtering window ranging from $N = 15$ to $N = 55$.

Fig. 9.7 on page 222 illustrates the results of the smoothing filters with Uniform kernel and Binomial kernel on a handheld sustained force of 6N. The Red Curve represents the noisy resultant force data, which is uniformly smoothed by the Blue Curve, while the Black curve smooths it with Binomial kernel. It is evident from the diagram shown in Fig. 9.7 that the data smoothing by both the kernels are similar at $N = 15$, whereas significant changes are observed in the smoothed data curves at $N = 55$. As observed from Fig. 9.7, the Binomial distribution tends to retain the peaks to some extent while the Uniform distribution diminishes the peaks as much as possible at $N = 55$. Hence, uniform kernel at $N = 55$ seems to achieve the goal of smoothing "sustained forces held − over a period of time".

Fig. 9.8 on page 223 demonstrates the results of smoothing resultant forces from actual treatment data. Likewise, the Red curve indicates the noisy forces while the Black and the Blue curve represents the smoothed forces due to Binomial and Uniform smoothing kernels in all the three sub-figures of Fig. 9.8. The Binomial smoothing supersedes the Uniform smoothing for treatment data as the window size (N) of the kernel increases from $N = 15$ to $N = 55$. During window size $N = 15$, it is observed that the Binomial curve almost takes the shape of the noisy data curve retaining all

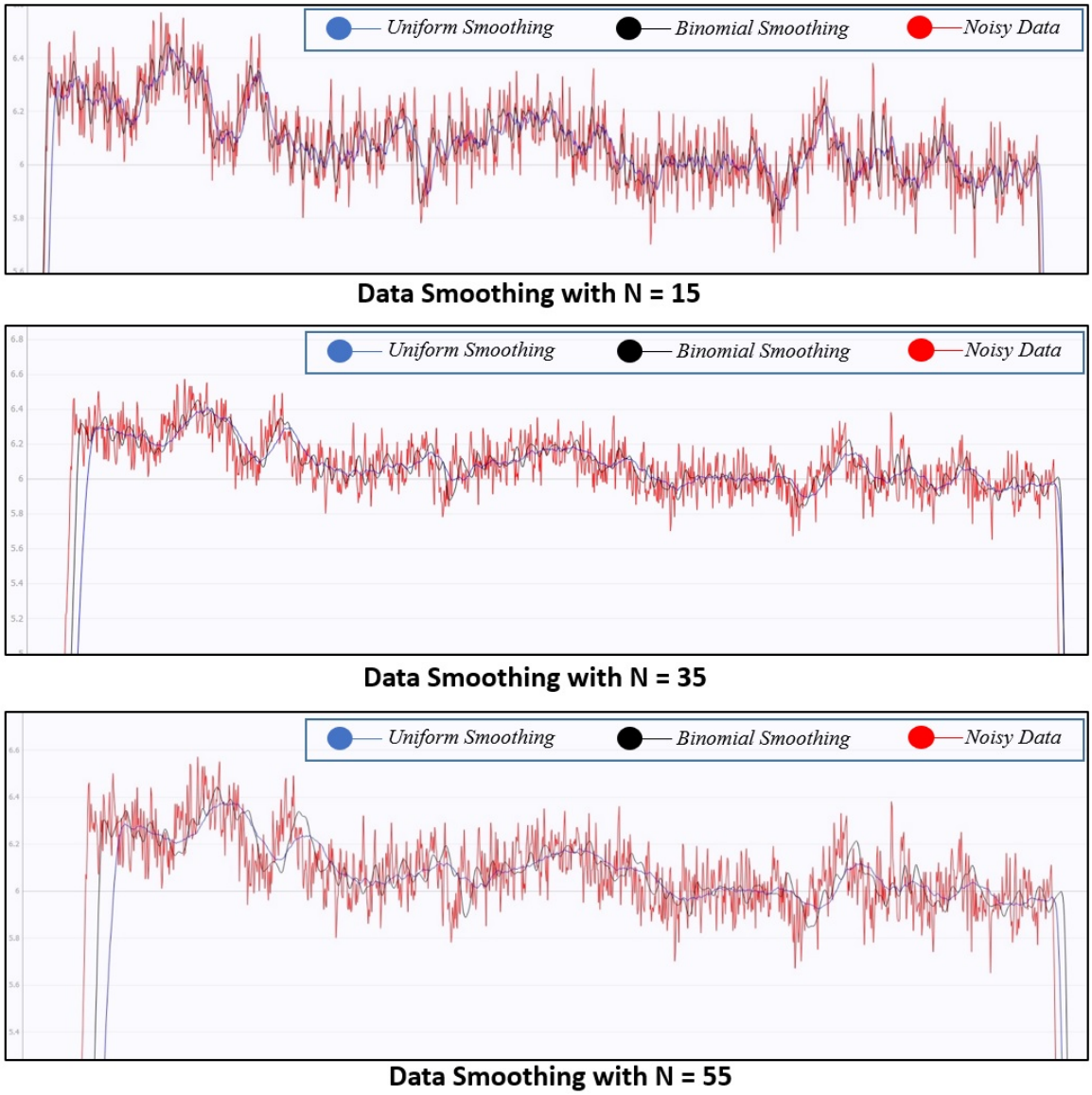


Fig. 9.7.: Comparison of Uniform and Binomial kernel smoothers with varying window sizes(N) on manually sustained force data

the peaks. At $N = 35$ it obtains better results smoothing the aberrant peaks while retaining the peak magnitudes. However, at $N = 55$ the peaks of Binomial kernel show much deviation from the actual force peaks with a much better smoothing, but

at the cost of data loss. Thus, it would be better to confine the window size of the kernels in between $N = 20$ to $N = 40$ for smoothing treatment data, without much data loss.

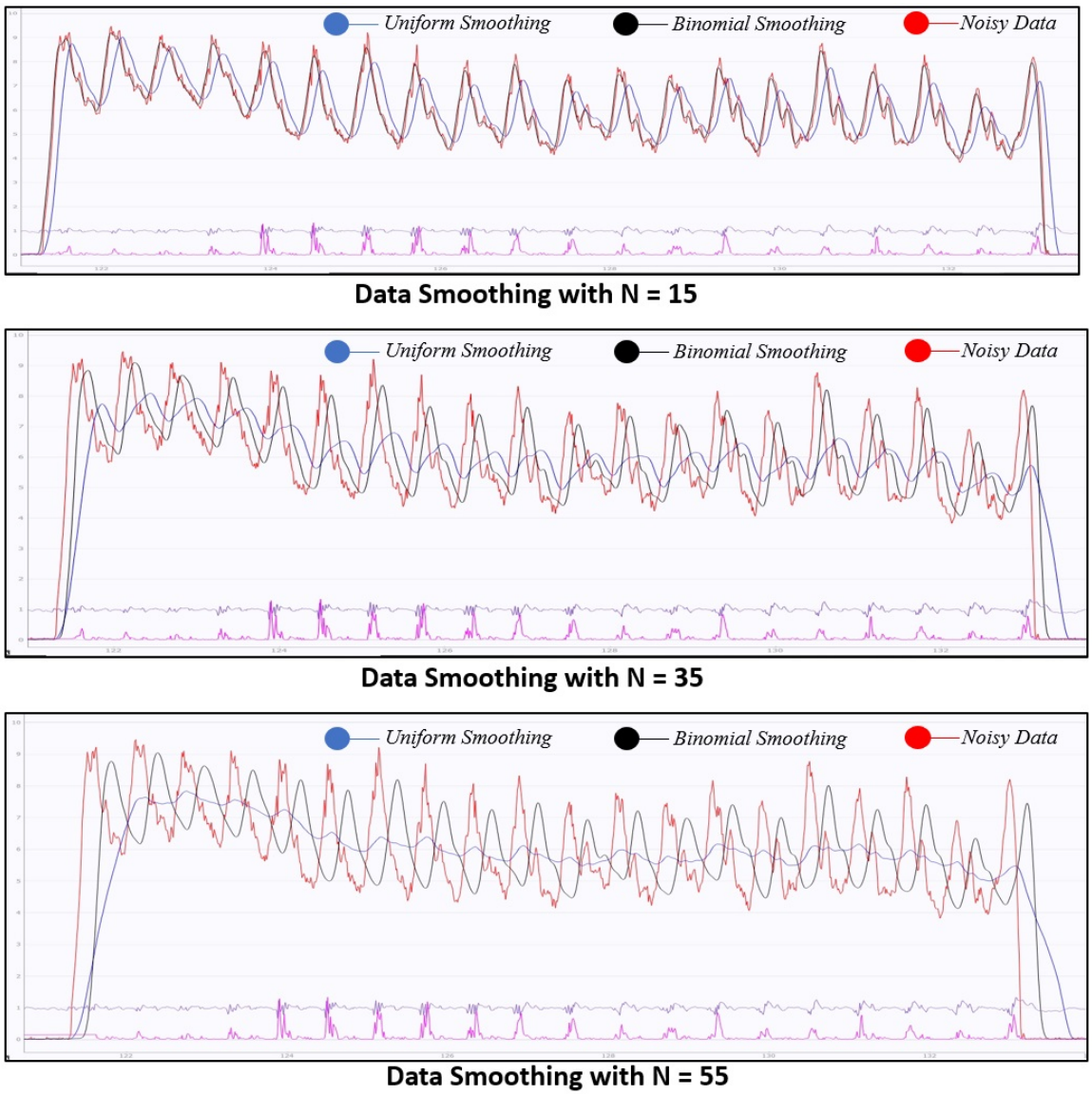


Fig. 9.8.: Comparison of Uniform and Binomial kernel smoothing filters with varying window sizes (N) on Treatment data

After several experimental trials, our implementation came up with an optimum kernel window size of $N = 25$, where better results are obtained from both the above mentioned smoothing kernel filters.

From mathematical observations and proofs, a discrete Gaussian distribution takes the shape of Uniform Distribution if its Standard Deviation equals its window size i.e. $\sigma = N$, whereas it takes the shape of Binomial distribution if the Standard Deviation is $\sigma = \frac{\sqrt{N}}{2}$, where N is the window size of the kernel. Thus, the Standard Deviation of a Gaussian distribution is a key factor to manipulate data smoothing of time varying noisy data.

A special observation from both Fig. 9.7 on page 222 and Fig. 9.8 on page 223 infers that − a steadily sustained force, without any oscillation, applied over time, implies no stroke frequency; while treatment with oscillating force patterns due to treatment strokes implies stroke frequency. Thus, low frequency data requires more smoothing while high frequency data require low smoothing to retain treatment information and stroke patterns.

The Pink colored curve at the bottom of each sub-figure in Fig. 9.8 on page 223 indicates − the second derivative of linear RMS 3-axis accelerometer data, with respect to time. The disturbance in measured RMS acceleration, as captured by the Pink curve, corresponds to each of the actual treatment strokes made, as shown in Fig. 9.8 on page 223 and in Fig. 9.9 on page 225. This disturbance in acceleration can be used as a feedback to pre-determine the treatment frequency which in turn can be used to generate an Adaptive Gaussian Smoothing Filter with adjusting Standard Deviation for varying treatment frequencies. For such a scenario, the lower the treatment frequency, the higher will be the standard deviation of the Adaptive Gaussian Smoothing Filter and vice versa.

9.2.2 Analysis of Stroke Count Algorithm

The Local Peak and Valley determination algorithm is developed to measure the local peaks for the treatment sub-session data captured by the treatment app (Child application) of Q-Ware in between two subsequent device resets. These local peaks are mixed with false stroke peaks. The false peaks can be due to massage patterns, irregularities in tissue layer or noise; hence, it is necessary to eliminate these false peaks to find the max peaks for each stroke, as performed by the peak filter algorithm, which in turn sums up to produce average peak force or the target force of the treatment. Thus the local peak and valley determination algorithm and the redundant peak filter algorithm combines to form the Stroke Count Algorithm.

The novel peak filter algorithm is used to calculate the number of strokes in each sub-session. Fig. 9.9 on page 225 shows the filtered stroke peaks of resultant force in green dots, while the valleys of forces are marked as red dots. It is evident from the Fig. 9.9 that all the aberrant peaks are discarded by the peak filter algorithm with minimum false positives. Therefore, to validate the Stroke count calculated by the algorithm, a clinical human study was performed where calculated strokes were

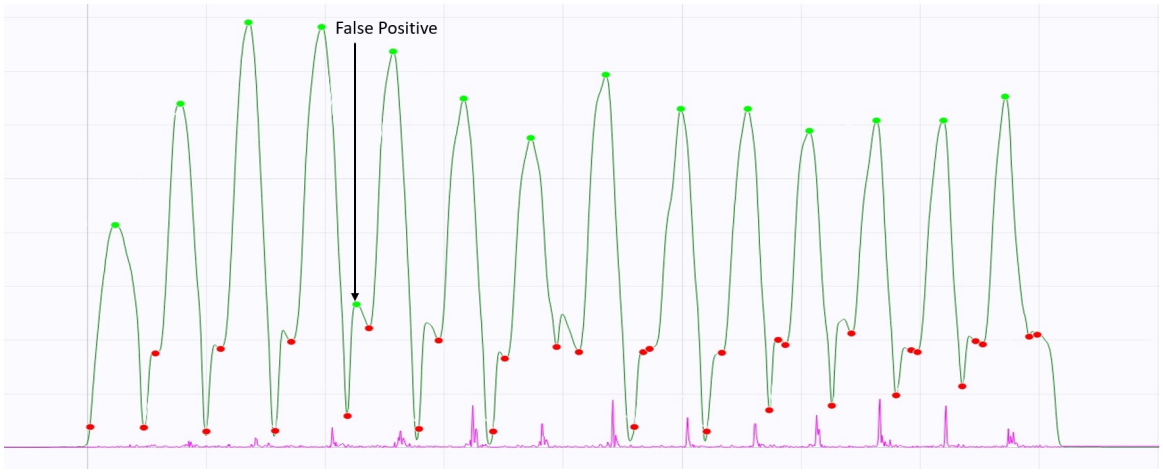


Fig. 9.9.: Treatment Strokes with corresponding peaks yielded from the redundant peak filtering algorithm

Table 9.7.: Dose-Load Type and Stroke validation clinical testing on Human Back by Examiner-A

SESSIONS ANALYSIS		EXAMINER - A (Experienced Physical Therapist)						
SESSIONS	**Sub-Sessions**	**Avg F(rez) Newtons**	**Avg Peak Newtons**	**Max Peak Newtons**	**Sessions Time Seconds**	**Stroke Manual Counted**	**Stroke Detected**	**Stroke Frequency (Hz)**
LOW Pressure Treatment for 15 Seconds	**Slow (freq.) Sub-Session**	2.50	3.88	4.47	16.89	14	14	0.835
	Medium (freq) Sub-Session	2.74	3.98	4.85	16.61	20	20	1.243
	High (freq) Sub-Session	2.18	2.91	4.39	16.55	35	36	2.174
MEDIUM Pressure Treatment for 15 Seconds	**Slow (freq.) Sub-Session**	6.55	10.03	12.21	15.92	13	13	0.816
	Medium (freq) Sub-Session	7.95	12.77	14.53	15.36	16	16	1.041
	High (freq) Sub-Session	9.03	13.29	17.52	15.76	32	32	2.029
HIGH Pressure Treatment for 15 Seconds	**Slow (freq.) Sub-Session**	11.67	19.34	22.49	21.02	11	11	0.528
	Medium (freq.) Sub-Session	10.41	17.74	20.87	17.44	15	15	0.862
	High (freq.) Sub-Session	12.05	21.33	15.14	16.59	40	40	2.411

Table 9.8.: Dose-Load Type and Stroke validation clinical testing on Human Back by Examiner-B

SESSIONS ANALYSIS		EXAMINER - B (Physical Therapy Student)						
SESSIONS	**Sub-Sessions**	**Avg F(rez) Newtons**	**Avg Peak Newtons**	**Max Peak Newtons**	**Sessions Time Seconds**	**Stroke Manual Counted**	**Stroke Detected**	**Stroke Frequency (Hz)**
LOW Pressure Treatment for 15 Seconds	**Slow (freq.) Sub-Session**	2.30	3.97	5.42	16.06	20	19	1.183
	Medium (freq) Sub-Session	1.47	2.54	3.42	16.24	19	20	1.231
	High (freq) Sub-Session	0.36	1.81	3.68	15.89	32	31	1.950
MEDIUM Pressure Treatment for 15 Seconds	**Slow (freq.) Sub-Session**	1.91	4.11	4.96	17.39	27	27	1.555
	Medium (freq) Sub-Session	3.16	5.45	7.17	16.52	18	18	1.08
	High (freq) Sub-Session	3.62	8.38	11.25	16.53	39	39	2.359
HIGH Pressure Treatment for 15 Seconds	**Slow (freq.) Sub-Session**	6.95	12.51	14.46	17.09	11	11	0.643
	Medium (freq.) Sub-Session	6.99	12.34	15.80	16.25	14	15	0.923
	High (freq.) Sub-Session	8.91	17.37	23.57	16.43	44	44	2.678

measured by manual counting to validate the error rate of the algorithm. In this clinical study, analysis of three consecutive sessions were performed by two different examiners of different experience levels on a same human subject. The examination was based on applying High, Medium and Low dose-loads with an approximation of high frequency, medium frequency and slow frequency of each. The data recorded by Examiner − A (Experienced Physical Therapist) are given in the Table 9.7 on page 226 that provides an error rate for the stroke count algorithm and its efficiency performed on smooth textured tissues on human back. However, another corresponding table with similar clinical data, as shown in Table 9.8 on page 227 with manual stroke counts are provided for Examiner-B (Physical Therapy Student) to perform the efficiency analysis of the stroke count algorithm, developed in the research, for smooth textured tissues. Data comparison from both the tables, Table 9.7 and Table 9.8, shows the variance of practice in between two different examiners and indicates that visual monitoring with this QSTM data requisition instrument would definitely improve the standard of STM practice reducing practice inconsistencies.

However, it is evident from the acquired clinical data that the stroke count algorithm is 99.6% accurate for the data collected in case of Examiner-A, while the it is 97.3% accurate in case of Examiner-B. Thus, the overall accuracy of the algorithm can be approximated as 98.5%. which is a great achievement for treatment on smooth tissue surface. This algorithm needs to be tested largely on irregular and rough tissue surfaces to approximate its precision and error rate on various clinical data and treatment scenarios.

Results of clinical data collected from Examiner-A and Examiner-B, respectively shows that the applied LOW forces are below 5 Newton's, while applied MEDIUM forces range from $9 - 15$ Newton's, whereas the HIGH forces applied start from the of 18 Newton's and above. These observations are based on the basis of the average peak force recorded by Examiner-A, who is over 20 years experienced in this field. On the other hand, average peaks recorded by Examiner-B, who is a novice practitioner, shows a wide range of variance where dose-load ranges overlap with

Fig. 9.10.: Treatment on Human Subject and Effects of High, Medium and Low Dose-Loads

LOW and MEDIUM ranges. This indicates that this novel QSTM data requisition instrument may assist in educating novice practitioner to achieve greater precision in practice. Moreover, the biological effects of High, Medium and Low dose-loads can also be studied as shown in Fig. 9.10 on page 9.10, where the tissue response for effects of dose-loads can be observed from Low to Medium to High forces applied from superior to inferior over the left thoracic paraspinals.

However, additional studies need to be performed to generate more data to estimate the efficiency of the data analysis performed by the QSTM PC software in the future. User manuals also need to be generated based on the different updates of the software in future to adapt to and reflect the changes in clinical practice or analytics.

10. DISCUSSION AND CONCLUSIONS

The existing STM approaches practice either by hand or with instrument-assisted mechanotherapy devices; both having been beneficial to remediate a wide range of musculoskeletal disorders. These practices have been used with immediate and long-term benefits for orthopedic impairments or post-surgical fitness. But it is evident from the literature review of clinical studies that the efficacy of these practices is hard to evaluate and compare due to lack of adequate quantification of forces applied during these rehabilitation techniques.

Instrument-assisted soft tissue manipulation (IASTM) helps to better distinguish tissue texture, irregularities, or aberrations in structure during palpation. Assessing the soft tissue quality informs the diagnostic process required to treat impairments but is mostly based on subjective or anecdotal practitioner experiences. Appropriate forces need to be applied during STM evaluation and interventions to avoid pain, bruising or excessive skin reddening due to overdosing of treatment, and to optimize the soft tissue treatment leading to better pain reduction and functional outcomes.

These limitations and challenges in current STM practice clearly evidence the need to characterize STM treatment dosing and clinically standardize it with respect to physical parameters of Quantifiable Soft Tissue Manipulation (QSTM™) namely − magnitude of resultant force applied per treatment strokes; direction and angle of force application from the skin (treatment) plane; and total active and dead time of treatment accounted during the overall treatment session.

The next section discusses how this characterization of STM treatment is addressed by developing a treatment instrument in this research, indicating the implementation approaches adopted to address all technical and clinical requirements of the developed clinical treatment device and its corresponding data requisition system.

10.1 Discussions

Our research addresses the characterization of STM treatment with physical parameters quantifying treatment variables mentioned in this thesis. Prime focus has been given on characterizing a STM dose-load with respect to mechatronic instrument assisted mechanical force delivered to the tissue over time. The data requisition treatment instrument developed in this research is defined as a custom designed ergonomic mechatronic handheld localized force applicator (QSTM Q1 Medical Device).

This treatment instrument clearly quantifies the dose-load magnitude in terms of Forces measured in Newtons as a function of time. These forces applied are quantified in terms of compressive and translational forces acting orthogonal and along the treatment plane, respectively, with the help of a precise 3D load cell (USL-H5-AP). All the force components acting during treatment are validated with the help of a pre-calibrated electronic force scale (PCE-PB-150N). The resultant of the compressive and translational force components is measured in real-time with the help of an ARM CORTEX M4 microcontroller unit (Teensy 3.2) of 72 MHz clock speed. A specific force calibration and scaling algorithm has been introduced in the device firmware to reduce sensor noise and scale forces to the electronic force scale with less than 2% error rate.

The treatment instrument addresses measuring the instantaneous angle of inclination of the treatment instrument from the skin (treatment) plane by measuring the instruments orientation angles (Yaw, Pitch and Roll) with respect to the earths horizon as well as an arbitrarily inclined skin plane. These measurements are performed by using an Inertial Measurement Unit (MPU-9250), which measures acceleration due to gravity and gyration of the device with respect to the cartesian co-ordinate systems. The concept of Euler transformations to determine orientation of rigid body rotations have been used to introduce a sensor fusion based proportional integral complementary filtering algorithm to measure angular orientation of the device with respect to gravity reference frame. A derived skin co-ordinate system has been con-

ceptualized and implemented by referencing the treatment device orthogonal to the skin (treatment) plane at the start of treatment to measure the inclination of the device orientation (Pitch angle) with respect to an inclined skin plane within 0-85 degrees inclination from the earth's horizon. The Geo-Pitch (Pitch angle with respect to earths horizon) and the Skin-Pitch (Pitch angle with respect to inclined skin-plane from earth's horizon) has been validated using − angle measurement experimental setups with an error rate less than 3% for Geo-Pitch and error rate less than 5% for Skin-Pitch, as described in Chapter 9. This concept is also valid for measuring the Skin-Yaw angles, but specific robotic angular orientation calibration test rigs are necessary to be designed to validate Yaw and Roll angles with respect to gravity reference and arbitrary skin plane reference. However, increased number of experimental trials of angular validation with respect to gravity reference and skin plane reference would substantiate the results within an optimum error rate.

Moreover, the direction of treatment is mostly subjective to treatment practice and presumed to be arbitrary, but the movement of the treatment device on skin is mostly along the longitudinal axis (Y axis of 3D Load Cell) of the device.

The instrument also features dual device resetting − either by reset button press on the device or by performing a flicker motion gesture to flash and re-initialize its memory to default factory state for processing new treatment data of next treatment sub-sessions.

A QSTM PC software Q-WARE© has been developed from scratch to implement a dual mode QSTM system indicating the operations of Idle mode and Treatment mode of the system. The treatment mode incorporates an instrument specific real-time QSTM data visualization, clinical data record and data analysis PC application. This application is a child application spawned by the Q-Ware(parent application) during treatment mode to calculate number of treatment strokes, stroke frequency, average of treatment variables calculated by the instrument and record treatment active and dead times as a part of post-processing computation. This data analysis helps to generate a treatment report with all calculated QSTM treatment parameters as an

STM prescription for the complete treatment session. A time series discrete binomial kernel filter-based data smoothing analysis of resultant force data has been introduced for further desired noise smoothing to detect local force peaks. Moreover, a novel algorithm has been designed to count particular resultant force peaks corresponding to each treatment stroke by filtering aberrant force peaks due to tissue irregularities or skin to tool-tip friction during treatment. This algorithm has been validated by performing 18 clinical trials on human subject testing, where the stroke count algorithm has proved to be effective with 98.5% accuracy when applied to smooth tissue textures as mentioned in Chapter 9. However, significant amount of clinical testing needs to be performed on human subjects at specific musculoskeletal treatment locations to determine the efficacy of the stroke count algorithm on rough tissue textures.

An increased amount of clinical testing would also address the inconsistencies of dose-load applications by clinicians and indicate the inter and intra rater reliabilities of STM treatment. This in turn would definitely open up the scopes for further improving this QSTM treatment characterization techniques.

10.2 Research Contributions

1. Architecture design of hardware and software of wired QSTM system, defining its operation modes (IDLE mode & TREATMENT mode) to perform, visualize and record STM treatment.

2. Mechanical design modifications of the treatment device to adapt the hardware specifications of clinical and technical requirements.

3. An Embedded Micro-controller program for Teensy 3.2's ARM CORTEX M4 Processor which involves:

   (a) Sensor Calibration algorithm using voltage to force transformation to smooth and scale measured forces in Newton's.

(b) Sensor-fusion algorithm using combination of Euler's transformation based Rigid body rotations and Proportional Integral Complementary filter to approximate Pitch angle of device with respect to arbitrary skin plane.

(c) Incorporated dual device reset condition using Device Button Press or Device Flicker Gesture.

4. A QSTM PC software (Q-WARE©) for WINDOWs systems, built from scratch, with two multiprocessing applications and some automatic features, that are:

(a) ***QSTM GUI*** − a treatment record system to analyze treatment reports and maintain treatment & patient record.

(b) ***Q1 Treatment App*** − to visualize treatment data transmitted from Q1 treatment device with more than 60Hz frame rate in real-time with a time division multiplexing algorithm, record incoming QSTM data to calculate treatment parameters as a part of post processing data analysis.

(c) ***Automatic Features*** − incorporated automatic features like − auto detection of treatment device and auto saving of treatment data.

5. QSTM Data Analysis library to calculate QSTM treatment parameters with following algorithms:

(a) Tested signal smoothing algorithm for resultant force data with discrete time Gaussian Kernel filter and derived discrete Binomial Kernel filter to minimize computation cost.

(b) Designed and tested a novel redundant force peak filter algorithm to calculate number of Strokes delivered during active time of treatment.

With these research contributions stated, the next section concludes the research with state of art existing QSTM treatment, indicating the augmentation and additional advantages that this technology could bring to the clinical standards of STM practice.

10.3 Conclusion

The Quantifiable Soft Tissue Manipulation (QSTM™) treatment instrument, discussed in this thesis, introduces a rehabilitation technology for therapeutic massage treatment assessment and evaluation that has been developed and demonstrated in this research. This pre-clinical research was performed to study the effects of quantitative, targeted mechanical loading using the QSTM Q1 Treatment Device on the consistency of localized force application, both with and without visual monitoring. These preliminary studies help evaluate the intra-rater reliability or the inter-rater reliability of practitioners and how to better adapt their practice to minimize inconsistencies. The experimental results shown in Chapter 9 is evident of the fact that recorded High, Medium and Low treatment dose loading vary subjective to the practitioner, and points to the need to characterize STM treatment using standardized STM terminologies and metrics, as described in this work. Thus, this technology can prove to be an effective tool to validate, standardize and revolutionize targeted force application of STM practice respective to musculoskeletal impairments.

Thus, characterization of STM treatment dose in terms of the QSTM parameters described in this research would address inconsistencies in treatment practice and distinguish variabilities in practice methods adopted by the practitioner. Also, experimentation and recorded treatment observations from multiple clinical studies with different practitioners of varying experiences would help to clinically standardize STM treatment protocols. Therefore, QSTM has the potential to better optimize STM practice standards by:

1. Performing clinical studies with this data requisition treatment instrument to study variation in performance of therapists.

2. Training and educating novice practitioners to the state of art practice using this treatment instrument as a self-assessment tool.

3. Establishing QSTM practice protocols.

4. Finally, standardizing quantified STM dose clinically with physical SI units.

These steps will eventually help in evaluation and advanced treatment determination of respective musculoskeletal disorders, and eliminate the barriers to optimal outcomes resulting from variable treatment dosing based mostly on subjective assessment and anecdotal clinician experience.

10.4 Future Work

This is the last section of this thesis which elucidates the unaddressed treatment variables of STM practice and their probable implemental procedures. It also throws some light to the improvement of present implemented algorithms to test unaddressed treatment scenarios. Finally, it also briefs about adaptation of additional treatment tools for specific targeted treatment applications or distributed force applications. The future goals of this ongoing line of research aims at:

1. Dose Pressure Quantification− This can be possibly done by standardizing the point of contact of treatment tool-tip surface area during treatment. This would give an estimate of amount of skin area covered during treatment. But a thorough study of Finite Element Analysis needs to be performed to simulate and validate the results.

2. Treatment Stroke Amplitude Calculation−This parameter can be measured by estimating the amount of linear displacement traversed by the tool treatment tip during a stroke cycle. This can be performed by positional mapping of the path of the treatment device during treatment performance. Also, the mapped path needs to be scaled to the actual linear meter scale.

3. Additional Treatment Tool Adaptations− Our future goals aim at development of additional mechatronic tool tip shapes and devices to facilitate treatment characterization and dose quantification as a significant evolution in IASTM practice throughout all regions of the body, in large and small areas alike.

Other adaptations include water proofing the devices, development of calibration cradles, and fabrication of stands to hold and store the devices for ease of use.

4. Modification Of Tilt Estimation Algorithm− The tilt estimation algorithm needs to be tested with anti-wind up filters to optimize drift due to integration of Gyro-readings, that would possibly help in eliminating singularities of Yaw and Roll measurements, that has neither been addressed nor these angles have been validated. However, the gimbal lock situations, due to axis locking for Euler transformation-based calculations, has not been addressed in this research. This restricts the derived skin angle calculations to the upper hemisphere of the earths horizon only. Therefore, quaternion transformation-based orientation angle calculation approaches need to be tested with non-linear complimentary filters to adequately calculate orientation angles (Geo-Yaw, Geo-Pitch, Geo-Roll) in all spatial orientations. Also, studies of experimental observations with magnetometer readings of the IMU sensor needs to be performed to visualize path trajectories of the device in real-time.

5. Stroke Validation Algorithm− The stroke count algorithm has been validated with musculoskeletal treatment on human subjects over smooth textured tissues. However clinical trials on rough textured tissues need to be performed to check efficacies of the same algorithm. But to make an optimized stroke count algorithm, the developed smoothing and redundant peak filtering algorithm need to be performed on compressive and translational force readings as well as on second derivative of linear acceleration readings or gyration readings with respect to time. A correlation metric on the filtered peaks of all these peak data would help generate the actual number of strokes on any rough textured tissue.

6. Prototyping Wireless Devices– The treatment instrument developed in this research is a wired device communicating with USB protocol. There is a vital need to develop wireless prototypes using Bluetooth communication protocol.

7. Clinical Acceptance of QSTM™ in STM practice – There is a high need to distributing and promoting this technology first among researchers to better understand and provide feedback analysis, and then amongst educators for training future clinicians in this manual therapy skill. Integration of QSTM into patient care of pets, sports animals and humans will help to evolve this cutting edge technology towards the evaluation and treatment of existing or future clinical needs.

REFERENCES

REFERENCES

[1] Physical Therapy Central, "Soft tissue mobilization," [accessed 23-April-2019]. [Online]. Available: https://ptcentral.org/treatment/soft-tissue-mobilization/

[2] A. M. Alotaibi, S. Anwar, M. T. Loghmani, and S. Chien, "Force sensing for an instrument-assisted soft tissue manipulation device," *Journal of Medical Devices*, vol. 11, no. 3, p. 031012, 2017.

[3] S. W. Cheatham, M. Lee, M. Cain, and R. Baker, "The efficacy of instrument assisted soft tissue mobilization: a systematic review," *The Journal of the Canadian Chiropractic Association*, vol. 60, no. 3, p. 200, 2016.

[4] M. Loghmani and S. Bane, "Instrument-assisted soft tissue manipulation: evidence for its emerging efficacy," *J Nov Physiother S*, vol. 3, p. 2, 2016.

[5] M. Lee, T.-Y. Choi, J.-I. Kim, and S.-M. Choi, "Using guasha to treat musculoskeletal pain: a systematic review of controlled clinical trials," *Chinese medicine*, vol. 5, no. 1, p. 5, 2010.

[6] A. Nielsen, "Gua sha research and the language of integrative medicine," *Journal of Bodywork and Movement Therapies*, vol. 13, no. 1, pp. 63–72, 2009.

[7] B. R. Kivlan, C. R. Carcia, F. R. Clemente, A. L. Phelps, and R. L. Martin, "The effect of astym® therapy on muscle strength: a blinded, randomized, clinically controlled trial," *BMC musculoskeletal disorders*, vol. 16, no. 1, p. 325, 2015.

[8] W. Hammer, "Graston technique, a necessary piece of the puzzle," *Dynamic chiropractic*, vol. 19, no. 20, 2001.

[9] M. Carey-Loghmani, J. Schrader, and W. Hammer, "Graston technique: M1 instruction manual," *3rd ed2010*, pp. 6–127, 2010.

[10] M. T. Loghmani and S. J. Warden, "Instrument-assisted cross-fiber massage accelerates knee ligament healing," *journal of orthopaedic & sports physical therapy*, vol. 39, no. 7, pp. 506–514, 2009.

[11] S. J. Warden and M. T. Loghmani, "Instrument-assisted cross fiber massage increases tissue perfusion and alters microvascular morphology in the vicinity of healing knee ligaments," *BMC complementary and alternative medicine*, vol. 13, no. 1, p. 240, 2013.

[12] M. Terry Loghmani, A. J. Bayliss, G. Clayton, and E. Gundeck, "Successful treatment of a guitarist with a finger joint injury using instrument-assisted soft tissue mobilization: a case report," *Journal of Manual & Manipulative Therapy*, vol. 23, no. 5, pp. 246–253, 2015.

[13] B. Looney, T. Srokose, C. Fernández-de-las Peñas, and J. A. Cleland, "Graston instrument soft tissue mobilization and home stretching for the management of plantar heel pain: a case series," *Journal of manipulative and physiological therapeutics*, vol. 34, no. 2, pp. 138–142, 2011.

[14] K. Laudner, B. D. Compton, T. A. McLoda, and C. M. Walters, "Acute effects of instrument assisted soft tissue mobilization for improving posterior shoulder range of motion in collegiate baseball players," *International journal of sports physical therapy*, vol. 9, no. 1, p. 1, 2014.

[15] M. L. Heinecke, S. T. Thuesen, and R. C. Stow, "Graston technique on shoulder motion in overhead athletes," *J Undergrad Kinesiol Res*, vol. 10, no. 1, pp. 27–39, 2014.

[16] D. W. Black, "Treatment of knee arthrofibrosis and quadriceps insufficiency after patellar tendon repair: a case report including use of the graston technique," *International journal of therapeutic massage & bodywork*, vol. 3, no. 2, p. 14, 2010.

[17] D. Morgan, "Principles of soft tissue treatment," *Journal of Manual & Manipulative Therapy*, vol. 2, no. 2, pp. 63–65, 1994.

[18] Q. Wang, H. Zeng, T. M. Best, C. Haas, N. T. Heffner, S. Agarwal, and Y. Zhao, "A mechatronic system for quantitative application and assessment of massage-like actions in small animals," *Annals of biomedical engineering*, vol. 42, no. 1, pp. 36–49, 2014.

[19] H. Zeng, T. A. Butterfield, S. Agarwal, F. Haq, T. M. Best, and Y. Zhao, "An engineering approach for quantitative analysis of the lengthwise strokes in massage therapies," *Journal of Medical Devices*, vol. 2, no. 4, p. 041003, 2008.

[20] H.-M. Lee, S.-K. Wu, and J.-Y. You, "Quantitative application of transverse friction massage and its neurological effects on flexor carpi radialis," *Manual therapy*, vol. 14, no. 5, pp. 501–507, 2009.

[21] A. M. Alotaibi, S. Anwar, and M. T. Loghmani, "Skin modeling analysis of a force sensing instrument-assisted soft tissue manipulation device," *Journal of Engineering and Science in Medical Diagnostics and Therapy*, vol. 1, no. 3, p. 031002, 2018.

[22] Forsentek, "0-1000n micro load cell 0-100kgf," [accessed 23-April-2019]. [Online]. Available: http://www.forsentek.com/prodetail_7.html

[23] TEC-GIHAN Co.,Ltd., "3-axis force sensor (usl06-h5 series)," [accessed 23-April-2019]. [Online]. Available: http://www.tecgihan.co.jp/en/products/3axis-force-sensor/usl6-h5/

[24] Invensense MPU-9250, "Mpu-9250 nine-axis (gyro-accelerometer-compass) mems motiontracking™device," [accessed 23-April-2019]. [Online]. Available: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/

[25] Sparkfun Store, "Sparkfun imu breakout - mpu-9250," [accessed 23-April-2019]. [Online]. Available: https://www.sparkfun.com/products/13762

[26] Invensense MPU-6050, "Mpu-6050 six-axis (gyro-accelerometer-compass) mems motiontracking™device," [accessed 23-April-2019]. [Online]. Available: https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/

[27] Arduino Store, "Arduino pro mini retired development board," [accessed 23-April-2019]. [Online]. Available: https://store.arduino.cc/usa/arduino-pro-mini

[28] Femtoduino Website, 2016, "Imuduino, femto, san franciso, ca," [accessed 21-February-2016]. [Online]. Available: http://femto.io/products

[29] PJRC Website, "Teensy usb development board," [accessed 23-April-2019]. [Online]. Available: https://www.pjrc.com/store/teensy32.html

[30] NXP Product, "Mk20dx256vlh7 kinetis k20 series chip," [accessed 23-April-2019]. [Online]. Available: https://www.nxp.com/part/MK20DX256VLH7

[31] Sparkfun Products, "Sparkfun bluetooth modem bluesmirf silver," [accessed 23-April-2019]. [Online]. Available: https://www.sparkfun.com/products/12577

[32] Sparkfun Store, "Sparkfun ftdi basic breakout - 5v/3.3v," [accessed 23-April-2019]. [Online]. Available: https://www.sparkfun.com/products/9716

[33] TEC-GIHAN Co.Ltd., "Precautions and settings for usl06 series setup procedures with dsa-03a," [accessed 23-April-2019]. [Online]. Available: http://www.tecgihan.co.jp/en/library/library_category/precautions-settings/#menu1-2

[34] Adafruit Industries, "Powerboost 1000 charger," [accessed 23-April-2019]. [Online]. Available: https://www.adafruit.com/product/2465

[35] Agile Modelling, "Uml 2 sequence diagrams," [accessed 23-April-2019]. [Online]. Available: http://www.agilemodeling.com/artifacts/sequenceDiagram.htm

[36] Arduino IDE Software, "Arduino previous ide releases," [accessed 23-April-2019]. [Online]. Available: https://www.arduino.cc/en/Main/OldSoftwareReleases

[37] PJRC Teensyduino, "Software add-on for the arduino ide," [accessed 23-April-2019]. [Online]. Available: https://www.pjrc.com/teensy/teensyduino.html

[38] C. J. Fisher, "Using an accelerometer for inclination sensing," *AN-1057, Application note, Analog Devices*, 2010.

[39] M. Pedley, "Tilt sensing using a three-axis accelerometer," *Freescale semiconductor application note*, vol. 1, pp. 2012–2013, 2013.

[40] Wetzstein, Gordon, "Ee 267 virtual reality course notes: 3-dof orientation tracking with imus," [accessed 17-April-2019]. [Online]. Available: https://scholar.google.com/

[41] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on automatic control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[42] J. Favre, B. Jolles, O. Siegrist, and K. Aminian, "Quaternion-based fusion of gyroscopes and accelerometers to improve 3d angle measurement," *Electronics Letters*, vol. 42, no. 11, pp. 612–614, 2006.

[43] Python Software Foundation [US], "Python 2.7.14," [accessed 23-April-2019]. [Online]. Available: https://www.python.org/downloads/release/python-2714/

[44] WX-Python, "The gui toolkit for python," [accessed 23-April-2019]. [Online]. Available: https://wxpython.org/pages/downloads/index.html

[45] PyQT4, "Platform to built application gui," [accessed 23-April-2019]. [Online]. Available: https://www.riverbankcomputing.com/software/pyqt/download

[46] PyQTGraph, "Scientific graphics and gui library for python," [accessed 23-April-2019]. [Online]. Available: http://www.pyqtgraph.org/

[47] Web Resource, "Perception of vision and illusion of motion," [accessed 17-April-2019]. [Online]. Available: https://paulbakaus.com/tutorials/performance/the-illusion-of-motion/

[48] Website, "Unix epoch time," [accessed 17-April-2019]. [Online]. Available: https://www.epochconverter.com/

[49] Web Resource, "Python time library," [accessed 17-April-2019]. [Online]. Available: https://docs.python.org/3/library/time.html

[50] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[51] T. Phillips, "From pascal's triangle to bell shaped curve," [accessed 17-April-2019]. [Online]. Available: http://www.ams.org/publicoutreach/feature-column/fcarc-normal

[52] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda, "Uniqueness of the gaussian kernel for scale-space filtering," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 1, pp. 26–33, 1986.

[53] W. Zucchini, A. Berzel, and O. Nenadic, "Applied smoothing techniques," *Part I: Kernel Density Estimation*, vol. 15, 2003.