A DATA COLLECTION SYSTEM DESIGN FOR HAND GESTURES

ATILIM UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ERHAN AKAN

A MASTER'S THESIS

ELECTRICAL & ELECTRONICS ENGINEERING DEPARTMENT

JUNE 2019

This thesis was approved by Graduate School of Natural and Applied Sciences of Atılım University.

_____

Prof. Dr. Ali KARA
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science of Atılım University Electrical and Electronics field**.

_____

Assoc. Prof. Dr. Kemal Efe ESELLER
Head of Department

This is to certify that we have read the thesis "A Data Collection System Design for Hand Gestures" submitted by Erhan AKAN and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Asst. Prof. Dr. Erdem Akagündüz Co-Supervisor	Asst. Prof. Dr. İbrahim Baran Uslu Supervisor

Examining Committee Members:

Assoc. Prof. Dr. Fikret ARI
Electrical & Electronics Eng. Dpt, Ankara University _____

Asst. Prof. Dr. İbrahim Baran USLU
Electrical & Electronics Eng. Dpt, Atılım University _____

Assoc. Prof. Dr. Reşat Özgür DORUK
Electrical & Electronics Eng. Dpt, Atılım University _____

Assoc. Prof. Dr. Kemal Efe ESELLER
Electrical & Electronics Eng. Dpt, Atılım University _____

Asst. Prof. Dr. Hakan TORA
Electrical & Electronics Eng. Dpt., Atılım University _____

Date: *27.06.2019*

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Erhan Akan

ABSTRACT

A DATA COLLECTION SYSTEM DESIGN FOR HAND GESTURES

Akan, Erhan

M. Sc., Department of Electrical and Electronics Engineering

Supervisor : Assist Prof. Dr. İbrahim Baran Uslu

Co-Supervisor : Assist Prof. Dr. Erdem Akagündüz

June 2019, 62 pages

In this study, we aim at designing a smart glove, which consists of different inertial sensors and an EMG sensor and developing a human-machine interaction application by pre-processing and fusing these different sensory data. We also aim at providing solutions in cases where image processing-based approaches are inefficient. In the proposed smart glove, the quartenion-based orientation data to be produced by the magnetometer and gyroscope together, the acceleration data to be generated by the accelerometer, and the analog data generated by the EMG sensor are collected and then prepared for use by different applications.

Keywords: Sensor Fusion, Smart Glove Framework, Inertial Measurement Unit (IMU), Inertial Sensors, Electromyography (EMG), Principal Component Analysis (PCA).

ÖZ

EL HAREKETLERİ İÇİN BİR VERİ TOPLAMA SİSTEMİ TASARIMI

Akan, Erhan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Dr. Öğr. Üyesi İbrahim Baran Uslu
Ortak Tez Yöneticisi : Dr. Öğr. Üyesi Erdem Akagündüz

Haziran 2019, 62 sayfa

Bu çalışmada, bir akıllı eldiven tasarımının yapılması, eldiven üzerindeki farklı ataletsel sensörler ve EMG sensörden veri toplanması, bu verilerin ön işlemeye tabi tutulması ve bu farklı sensör verilerinin kaynaştırılması yoluyla bir insan-makine etkileşimi uygulamasının geliştirilmesi amaçlanmaktadır. Böylelikle görüntü işleme temelli yaklaşımların kusurlu olduğu noktalarda çözümler sunulması hedeflenmektedir. Akıllı eldivende, manyetometre ve jiroskop tarafından üretilecek olan dördey bazlı oryantasyon verileri ile ivmeölçer tarafından üretilecek olan ivme verilerinin ve EMG Sensor tarafından üretilen analog verilerin, toplanması ve daha sonradan farklı uygulamalarca kullanılmasına hazırlık konusunda bir çalışma yapılmıştır.

Anahtar Kelimeler: Sensör Kaynaştırma, Akıllı Eldiven İskeleti, Ateletsel Ölçüm Birimi (IMU), Ataletsel Sensörler, Elektromiyografi (EMG), Temel Bileşen Analizi (PCA).

For the advancement of humanity

ACKNOWLEDGEMENT

I’d like to thank to my supervisors Assist. Prof Dr. Baran Uslu and Assist. Prof Dr. Erdem Akagündüz for their great effects on my motivation and study. I am very grateful for their friendly guidance and their big patience. Before writing this thesis, also, I’d like to thank to Assit. Prof. Dr. Hakan Tora for his motivation on publishing my first conference paper. Also, I’d like to thank to my father Ertuğrul Akan and my mother Selma Akan who are two great people, who supported and motivated me with all of their efforts and life until now. Also, I’d like to thank to my dear girlfriend Mehtap Nur Bitmez, great sociologist and PhD candidate. Without her beautiful smile, motivation and sensibilities, it wouldn’t be that easy to complete my study.

LIST OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

SYMBOLS/ABBREVIATION LIST

CS	Chip Select
DMP	Digital Motion Processor
DOF	Degrees of Freedom
EMG	Electromyography
I2C	Inter Integrated Circuit
IMU	Inertial Measurement Unit
LED	Light Emitting Diode
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
PCA	Principal Component Analysis
PWM	Pulse Width Modulation
SCK	Serial Clock
SCL	Serial Clock
SDA	Serial Data
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
USART	Universal Asynchronous Synchronous Receiver Transmitter

CHAPTER 1

INTRODUCTION

Various methods have been presented in the literature on human machine interaction [1]–[5]. In this interaction, sensors are always used, and a quantity is converted as a signal into the format that the machine can interpret. Different types of sensors come in different devices. For example, photosensitive sensors are present in the camera or computer mouse, and sound-sensitive sensors appear in the phone or voice recorder. What we call a device is an integrated tool that allows us to use these sensors in a meaningful way. The device will be produced in the study will be a *smart glove* and the sensors that make up this device will be the gyroscope, accelerometer, magnetometer embedded in the IMU and EMG sensor that receives the electrical signal. In this study, a robust smart glove framework will collect quaternion, acceleration and EMG data in a satisfactory sampling rate. Then, it uniformly samples these signals to be used in various applications such as sign language recognition, robotics, gaming, medical treatments.

1.1 Objective and Motivation

By common usage of these sensors, it is expected that, analysis of these signals and figure out which sensors are necessary, sufficient and appropriate to the real-time applications which will lead the future sensor fusion studies and expected to contribute to the signal processing literature. In the studies conducted so far, there are no frameworks containing these sensors as one structure and the orientation data of these studies does not include the quaternion components. In this context, the novelty of this framework could be considered.

The main motivation of developing a smart glove using different kind of sensors is to create first sign language database composed of inertial and EMG sensors containing upper limb IMU. Thus, creating the first IMU and EMG based Turkish Sign Language database is possible using this framework.

1.2 Literature Search

In studies conducted with smart glove frameworks, there are sensors similarly used in this study however, these frameworks contain some of these sensors or includes some limbs. They generally used to identify sign language due to the mostly requirement of getting the data of the hinge joints. These studies are mostly mentioned as sign language recognizer frameworks in this part of the paper.

In studies conducted with smart gloves to identify signs, “word recognition” and “finger syllable or finger alphabet recognition” studies are often confused. Studies conducted so far are about generally in the area of the finger-syllable recognition. Finding the result from an existing look-up table by matching the final shape of the fingers with a predefined value with a template mapping or mean-squared-error approaches after pre-processing. Methods such as machine learning are not much preferred because of the calculation costs in this application. A local project of 2018, which recognizes a finger-syllable, can be accessed from the reference [6]. Similarly, in a project [7], glove system with a flex sensor was presented and finger-spelling recognition was carried out instead of sign language recognition. In the study, pre-processed quaternion, EMG and acceleration data placed in a time series will be used as feature vectors. This work can be used easily in sign language, finger spelling or gesture recognition applications.

Since the 1990s, studies on the recognition of sign language and the studies on the recognition of gestures show similarities [4], [8] in terms of classification algorithms and feature extraction. Moreover, when these studies are examined, it is seen that many of them recognize sign language or gesture based on image processing. However, this approach, as mentioned frequently in such studies, involves many strict requirements, such as the obstruction of the image, the necessity of having a camera in the environment.

In the literature, there are studies on many image processing-based gesture or sign language recognition. Although image processing may seem low-cost and practical at first glance, this method brings along many problems such as overlapping of the organs used to form the sign language, mixing of skin colors, poor quality of the camera and

insufficient lighting. In some studies, Kinect and Leap Motion as in [9], and customized cameras for such tasks, such as in [2], have been used. However, these methods also have some limitations that are specific to cameras, especially the cost.

In the study previously conducted in [10], a glove consisting of only flex sensors has been proposed which translates the ”finger spelling” in Hindi and English. The work simply sends the voltage readings from the flex sensors to an analog comparator, allowing the letter to be detected at that time. In [11], a study that can recognize 8 different hand gestures and uses magnetometer, accelerometer and gyroscope, classifying with PCA (Principal Component Analysis) and extracting with LDA (Linear Discriminant Analysis) and classification with SVM (Support Vector Machine) is presented. In [12], a study has been proposed in which all inertial sensors are used, such as compression, normalization and classification, which are extracted from time series data taken from a cell phone instead of glove.

When we look at the literature, with the development of technology and the decrease of costs in recent times, the studies focused on segmentation and recognition by EMG sensors. In this reference [13], a detailed presentation of segmentation and feature extraction with EMG sensors was made in [14]. Therefore, it is planned to include EMG sensors in this study to examine the effects of these sensors on performance.

When the intelligent human-machine interaction systems that have been presented so far, there are studies that include the inertial sensors and the EMG sensor that we will use at the same time [15]–[17], however in these studies the recognition process was performed by the means of EMG sensors are usually attached to certain areas of the arm and the recognition process is done by means of a 9-DOF inertial sensor located on the top of the hand or the arm.

CHAPTER 2

SYSTEM DEFINITION

In this study, one EMG sensor positioned on the forearm, the 6-DOF IMUs are positioned at the top of each one of five fingers, and 9-DOF IMUs are positioned at the top and the upper arm as seen in Figure 2.2. A powerful microcontroller, a SD card module and a 9-DOF IMU module is positioned on the back of hand. The system structure could be seen in Figure 2.1. *N* indicates the Nine DOF IMU, *S* indicates the Six DOF IMU modules. *i* is the indices of the IMU module.

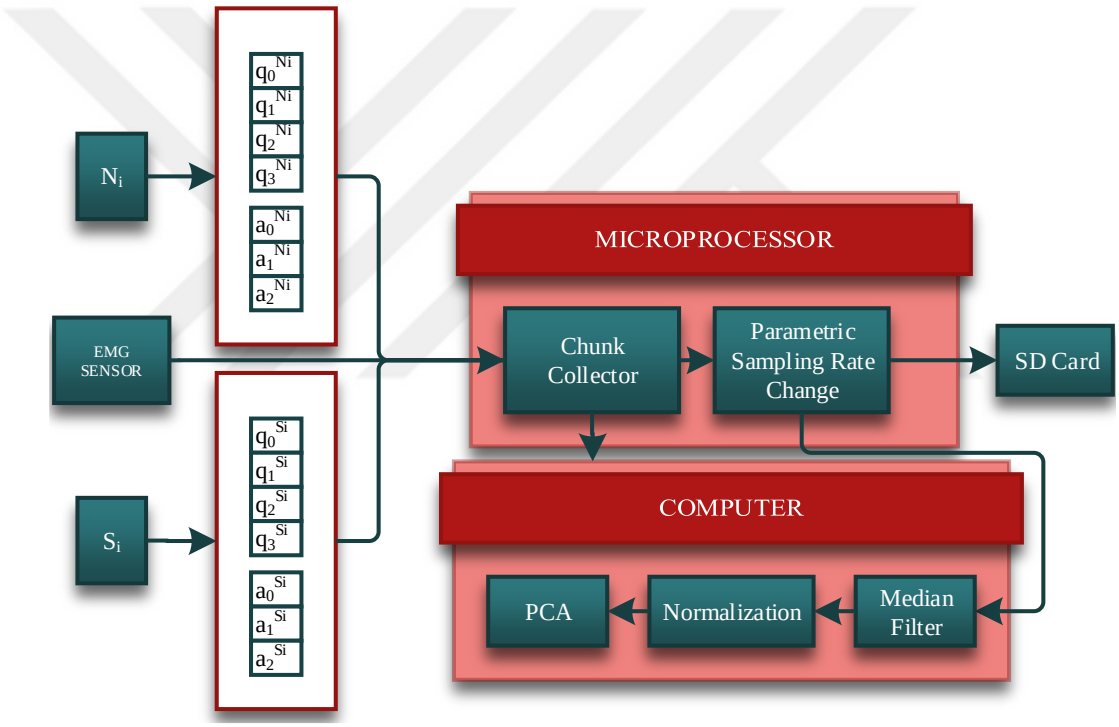


Figure 2.1 General system structure

2.1 Positioning of the Sensors

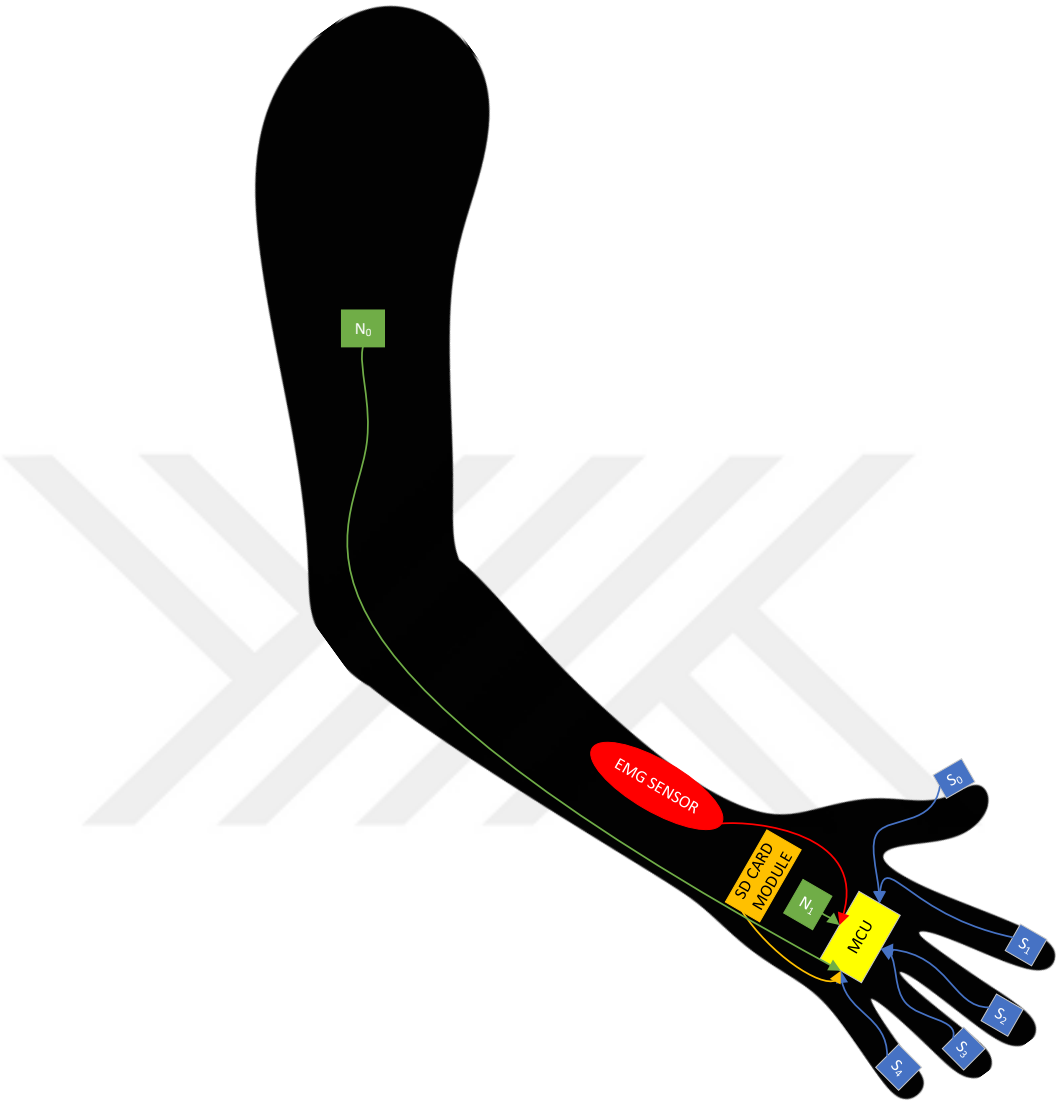


Figure 2.2 Schematic of sensor positioning on human limbs

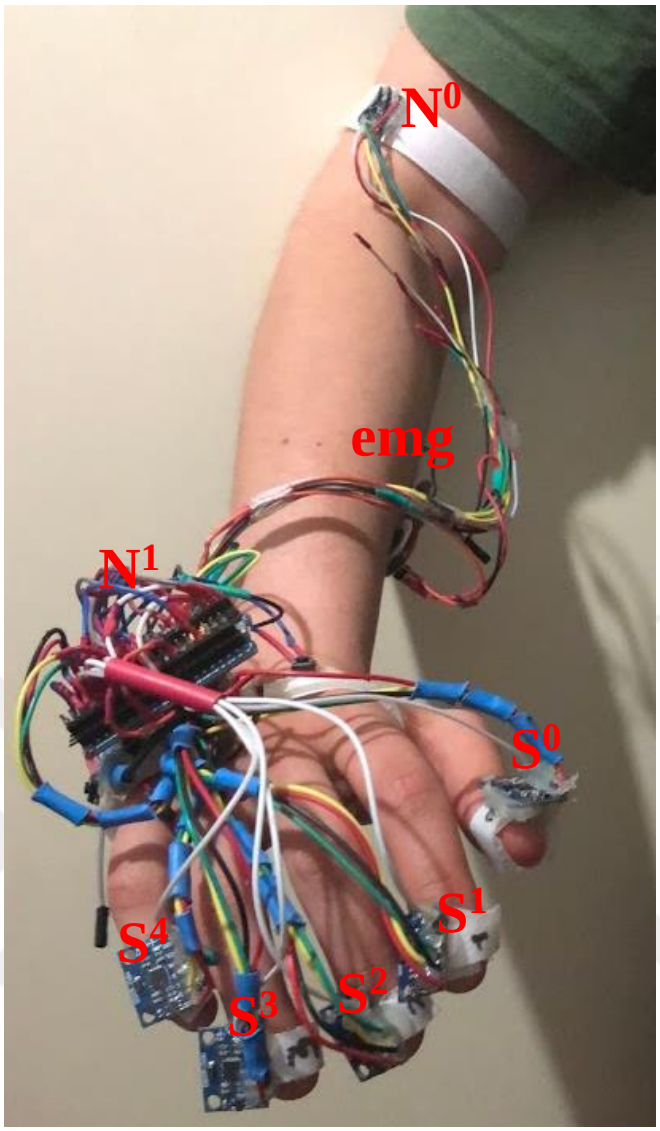


Figure 2.3 Photo of sensor positioning on human limbs

As shown in the Figure 2.2 and Figure 2.3, the 6-DOF IMUs are positioned above the fingertips because they are low in cost and many of them are used, while the 9-DOF inertial sensors IMUs are positioned above the hand in full center and above the rear arm. **Note that, the 6-DOF sensors should be positioned which the wired side should be like in Figure 2.3.** Sensors are deliberately placed on the fingertips because the fingertip of a person is the most moved limb in a gesture movement where the fingers are used and carries more characteristic of the current movement. Since 6-DOF sensor units do not have a magnetometer, they can provide absolute orientation data instead of relative orientation. 9-DOF sensor units, thanks to the magnetometer inside of them, considering the effect of the earth can provide absolute orientation data. Therefore, this 9-DOF measurement units, where we can get absolute orientation data,

are placed at strategic points to assist in calibrating the relative orientation, the fingers and the hand to the absolute orientation in the flat where the relative orientation data can be obtained. In addition, the EMG sensor can be positioned on *extensor carpi radialis longus extensor carpi ulnaris or extensor digitorum, flexor carpi radialis longus* muscle groups, some of the main muscle groups, which are highly efficient in enhancing performance as mentioned in [13].

It’s seen that placing this EMG sensor on the extensor carpi radialis longus muscle group is very efficient to obtain signal where the probability of the electrodes which are not able to acquire data easily because of staying on the hard skin and hair, and that the reaction during muscle movement is good.

CHAPTER 3

A SMALL INTRODUCTION TO ORIENTATION

Orientation of a vector defines the placement of that vector related to a reference axis. As seen in figure below, the orientation of a plane could be seen by referencing x-axis. The orientation of an object could be relative or absolute. **Relative orientation** needs the gyro and accelerometer data to obtain the orientation depends on a reference axis while the **absolute orientation** needs gyro, accelerometer and magnetometer data to obtain the orientation depends on Earth's surface.

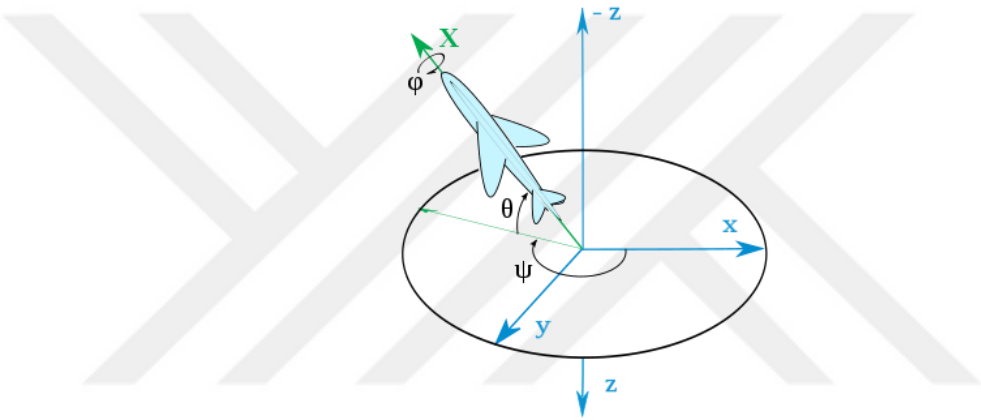


Figure 3.1 Orientation of a plane[18]

3.1 What are Quaternions?

Quaternions are a different form of the complex number system which has the notation in the form of equation below. Where w is scalar number and i, j, k components are imaginary numbers. It was described in 1843 by the famous mathematician William Rowan Hamilton. Due to the problem of Its main area of use is to calculate the rotation in 3D space.

$$w + xi + yj + zk \tag{1}$$

3.2 Why Quaternions?

Euler angles are a set of coordinate systems that are used to define the orientation of an object in 3D space, including the data of the yaw, pitch and roll. However, since each axis of rotation actually rotates another axis according to its own angle, there are situations where the 2 axes overlap. In this case, even if one of the two overlapped axes rotate, the other one emerges as if it is not rotating and waiting at the same angle, and a data loss occurs. This problem emerges in our daily life as the discrete movements occurring in 3D games or modeling programs. This unwanted state is called the gimbal lock (mentioned in 3.2.1). Because the Euler angles are inadequate to define the orientation in our 3D world. Therefore, when the orientation is examined in 3 dimensions, a fourth angle of the object is added without losing one of the rotation angles. The new system, which has a 4th dimension added, is called quaternions.

3.2.1 What is Gimbal Lock?

When we try to rotate a vector defined in Euclidian space, the rotation matrix will be as follows:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

When we try to rotate a vector through angle $\beta = \frac{\pi}{2}$, the equation above will be as follows:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Then, the equation will be reduced into:

$$R = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma)) & 0 \\ -\cos(\alpha+\gamma)) & \sin(\alpha+\gamma)) & 0 \end{bmatrix} \quad (4)$$

As seen in the last equation above, if we try to change the rotation angles α and γ, the first column and the last row will not be affected. This causes a problem named loss of one DOF and consequently the gimbal lock problem [19].

3.3 Quaternion to Euler Conversion

The quaternions produced by the IMUs are converted into the Euler angles by the relation in the equation below [20]. Where φ, θ, ψ denotes bank (rotation about new x-axis,), attitude (rotation about new y-axis), heading (rotation about z-axis) respectively [21]. They are known as Tait-Bryan angles which defines the orientation of a vector which are subsets of Euler Angles. Tait-Bryan angles are commonly used in quaternion to Euler conversion, thus the relation between the quaternion and Tait-Bryan angles was discussed for the equations below. In lots of papers, they both called as Euler angles and the differences are about the direction of the axes, reference points, sequence order or the angles. They both define the orientation of a vector in 3D space with similar approximations. Therefore, there is no problem if we name them as Euler Angles. In Figure 3.2, the orientation of vector N could be seen. In Figure 3.3, and the proper Euler Angles could be seen in Figure 3.3.

$$\begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \tan^{-1} \dfrac{2(q_0 q_1 + q_2 q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \sin^{-1}(2(q_0 q_2 - q_3 q_1)) \\ \tan^{-1} \dfrac{2(q_0 q_3 + q_1 q_2)}{1 - 2(q_2^2 + q_3^2)} \end{bmatrix} \tag{5}$$

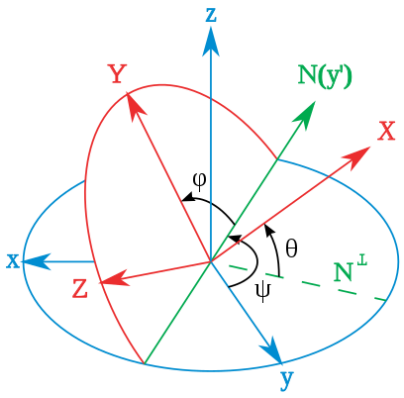


Figure 3.2 Tait-Bryan Angles [21]

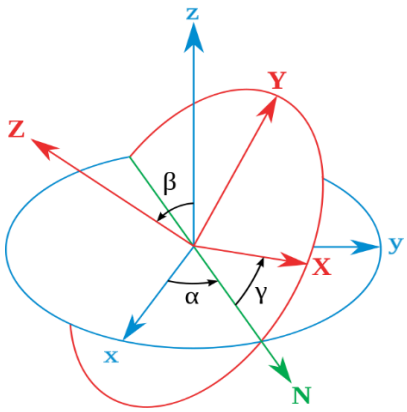


Figure 3.3 Proper Euler Angles[21]

CHAPTER 4

EQUIPMENT DEFINITION

The 6-DOF IMUs consist of Invensense MPU6050 modules with very low cost. 9-DOF measurement units consist of Bosch BNO055 modules with higher cost. The EMG sensor consists of the MyoWare ™ Muscle Sensor (AT-04-001) module, which can be preprocessed by the signal. The microcontroller used is an STM32F103C8T6 Blue Pill board operating at 72 MHz. All inertial based sensors using I2C digital communication methods do have registers to store data. Additionally, some of these registers are writable and thus, this property makes these devices easily software configurable.

4.1 Design Process

During the design process of this smart glove framework, various sensors, development boards and circuit structures were tested and evaluated. The important things about this framework were robustness and easy development. Therefore, a simplified C++ language adapted to Arduino was used in order to upgrade the board if needed. Also, the libraries of MPU6050, BNO055 and SD Card module are easily obtained for the Arduino platform. The performance/price ratio was also considered. For this reason, MPU6050 IMU boards were placed onto the fingers which were thought to have similar orientation with the BNO055 on the back hand in a position which all the fingers outstretched. Consequently, instead of using high cost IMU, BNO055 on each finger, a low cost IMU, MPU6050 was used.

About the microcontroller part, the first device in the first framework used was Arduino Pro Micro development board (Figure 4.1) which has 18 programmable input/output pins and 16MHz Microchip ATmega32U4 microcontroller. Due to small number of pins a decoder was used to increase the number of I2C address selection pins of MPU6050 IMU boards; however, it increases the power consumption and the consumed time among reaching the IMUs. A secondary problem comes with being a 16MHz microcontroller. While trying to read the FIFO buffer of an MPU6050, due to the low clock frequency, some of the samples could not be captured. Also, it has 32

kilobytes of flash memory to store the program codes which has caused an insufficient memory issue.

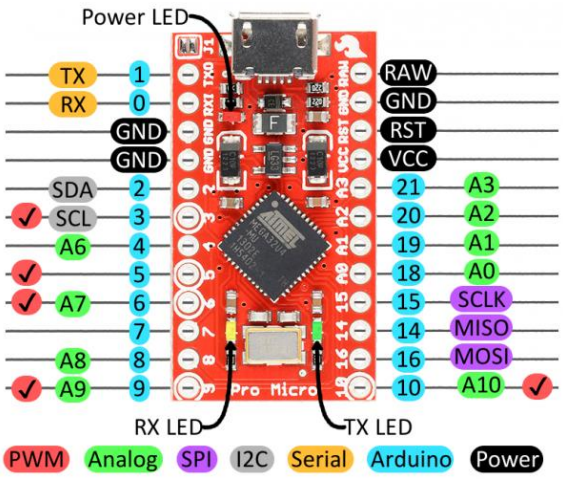


Figure 4.1 Arduino Pro Micro and pinout[22]

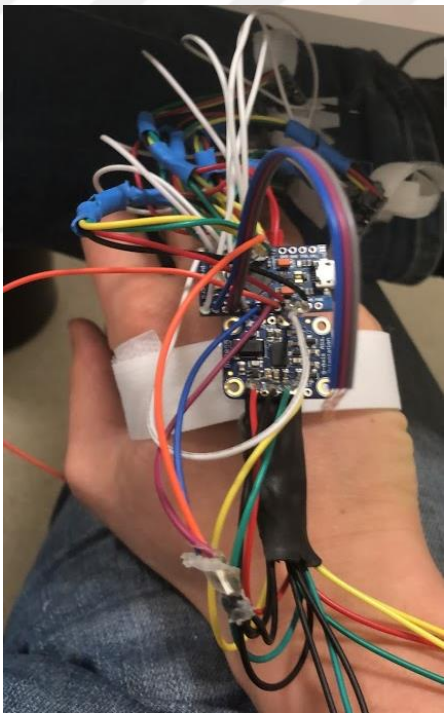


Figure 4.2 Old glove framework

For the complete framework, all the parameters have been evaluated and the final decision was to use STM32F103C8T6 Blue Pill development board.

4.2 MPU6050

MPU6050 (see Figure 4.3) is a low cost and low power IMU which supports the I2C digital communication protocol. The board has a 3.3V regulator to reduce 5V supply. MPU6050 has lots of programmable registers to set the resolution of the data to be read and lots of registers to show different kind of data such as Euler angles, quaternions, linear acceleration or temperature.

Figure 4.3 MPU6050 IMU Board

MPU6050 has a digital motion processor (DMP) in order to fuse the accelerometer and gyro data to generate the quaternion output which will provides the relative orientation data. Additionally, it has a low pass filter to deal with the noise.

As stated in the datasheet[23], maximum current draw on each MPU6050 IMU board is approximately 4mA and the sampling rate of the DMP is about 200Hz. In this project, MPU6050 was chosen because the power consumption, documentation and being a low cost IMU was the priority. In addition, the quaternion components are provided in a FIFO buffer structure.

4.2.1 About FIFO Buffer

The quaternion of relative orientation and the linear acceleration data read from FIFO buffer as 2 bytes from an 8 bits length of FIFO buffer (see Table 4.1). Each 2 bytes represent one quaternion axis. In addition to these axes, gyro and temperature data are always available in this buffer. However, they are discarded during the reading process.

Table 4.1 FIFO Buffer Structure

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

4.3 BNO055

BNO055 (see Figure 4.4) is a very talented and stable 9-DOF IMU manufactured by Bosch. As known for MPU6050, it has configuration and data registers to obtain quaternion, Euler angles and linear acceleration. Each of these values are stored in dedicated registers. Each BNO055 IMU board draws approximately 12.5mA due to having a magnetometer and register structure instead of a FIFO buffer structure. The sampling rate of this IMU is approximately 100Hz in absolute orientation mode. In addition, it has a low pass filter to deal with the noise.

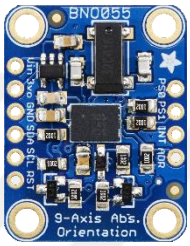


Figure 4.4 BNO055 IMU Board

4.3.1 Calibration of BNO055

BNO055 has an autocalibration property and calibration register to figure out the calibration status. As stated on the datasheet [24], for accelerometer calibration, the device should be lying in six stable positions and at least one perpendicular position to axes x, y and z. For gyroscope calibration, the device should be left stable for several seconds. For magnetometer calibration, drawing “8” with the device in the midair will be enough.

4.4 Myoware Muscle Sensor

MyoWare (see Figure 4.5) is an EMG sensor which generates two types of outputs. Which are raw EMG signal output and rectified and enveloped (preprocessed) signal output. For this work, the preprocessed signal output will be used.

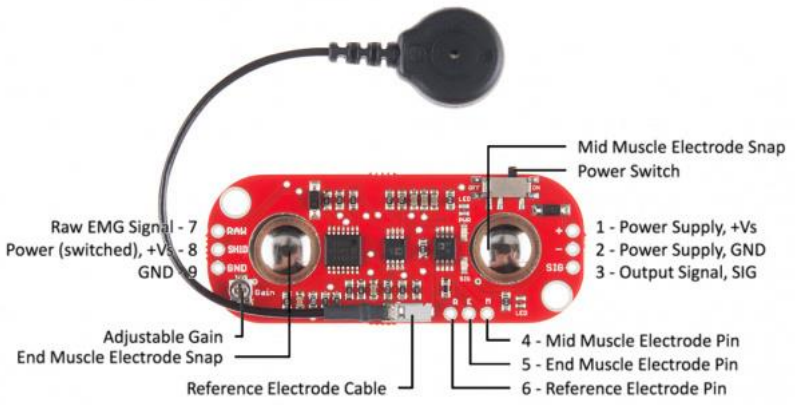


Figure 4.5 MyoWare Muscle Sensor[25]

This sensor uses dry electrode sticky pads in order to fix itself on a limb. It draws approximately 14mA as a maximum rating value. However, since the generated voltage from the muscles have a frequency approximately between 50Hz and 150Hz[26], unfortunately the 50Hz of network frequency will slightly interfere the measurement of the sensor. In order to deal with that noise, it is a better solution to power up this sensor using battery.

For the preprocessed output, the raw signal taken from the muscles will be amplified, rectified and integrated (envelope version). As seen in Figure 4.6.

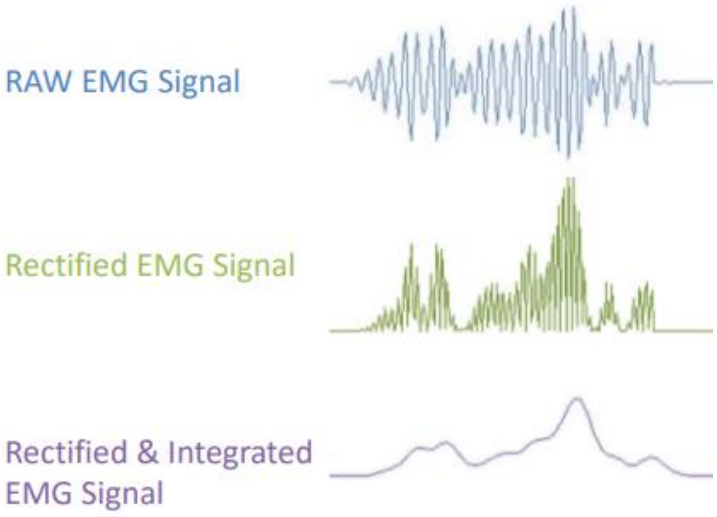


Figure 4.6 Raw versus preprocessed EMG signals[27]

4.5 SD Card module

SD Card module (Figure 4.7) uses the SPI protocol to communicate and the libraries written for this module support FAT32 file system.

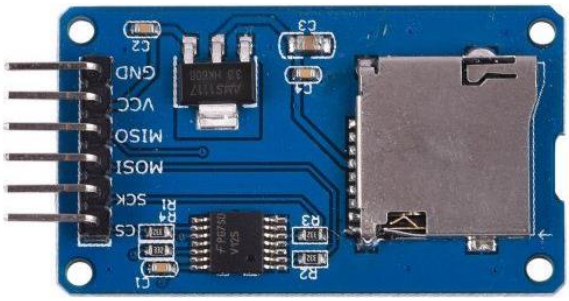


Figure 4.7 SD card module

4.6 STM32F103C8T6 Bluepill Development Board

As seen in Figure 4.8, the STM32F103C8T6 Blue Pill Development Board and the pinout diagram was presented. STM32F103C8T6 is ARM Cortex M3 Based, 72 MHz microcontroller manufactured by ST Microelectronics. The board contains 3.3V regulator if 5V supply to be used. Very powerful, low cost, and which is useful for real time applications. In datasheet of this microcontroller [28], it states that, the multiplication and division math will took 1 cycle which is very convenient for preprocessing of the signals. Normally, this board could be programmed using assembly, C and C++ however, a small developer community known as STM32DUINO had created libraries for this microcontroller and thus, it could be programmed using simplified Arduino language. Due to the easy programming over USB port, a bootloader created for this board should be downloaded into the flash memory of the MCU.

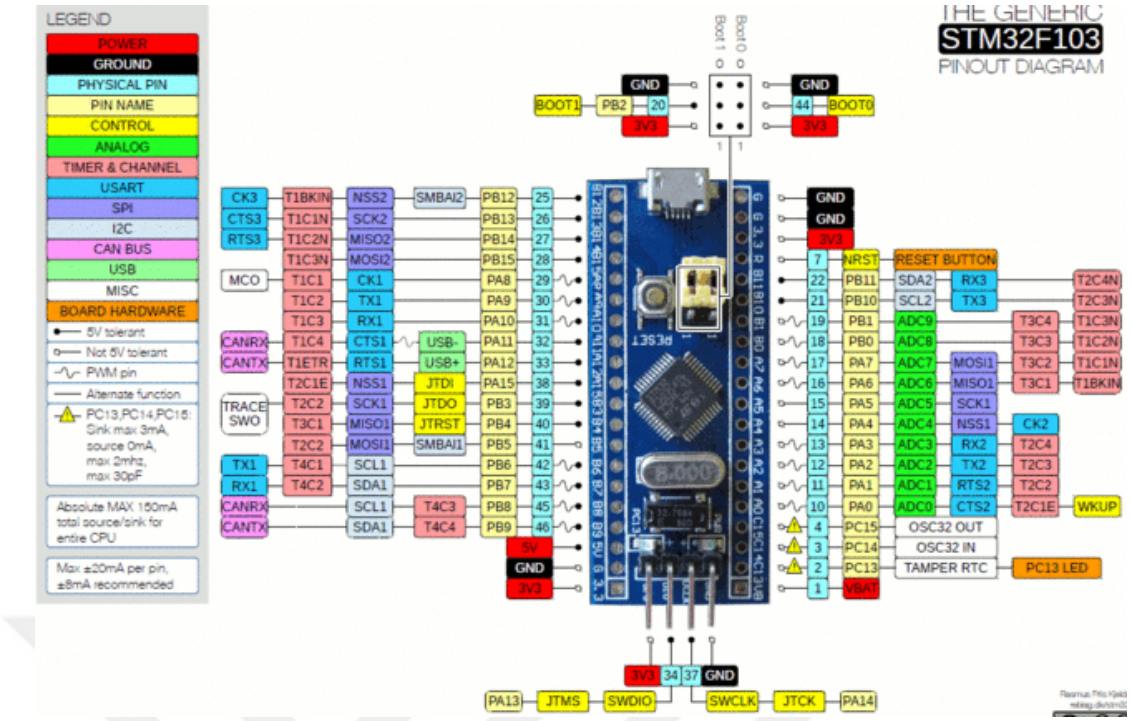


Figure 4.8 Blue Pill Board and pin specifications[29]

4.7 Circuit Scheme

As seen in Figure 4.9, the wiring schematic was presented. These wire colors also represent the wire colors used in the real-world framework itself. All red wires represent the 5V power line while the black ones represent the ground line. Yellow wires represent the I2C SCL while the green ones represent the I2C SDA. White wired connections are used for the I2C address selection for the I2C sensors. The orange wire connected to the EMG sensor represents the preprocessed analog data output of the EMG sensor. The maroon, gray, blue and purple wires connected to the SD card module represents the CS, SCK, MOSI, MISO lines. The purple, orange and blue wires connected to the LED through resistors represents the red, green and blue anodes of the LED. The remaining blue wires connected to the buttons are the line between the MCU and the GND line through a pulldown resistor.

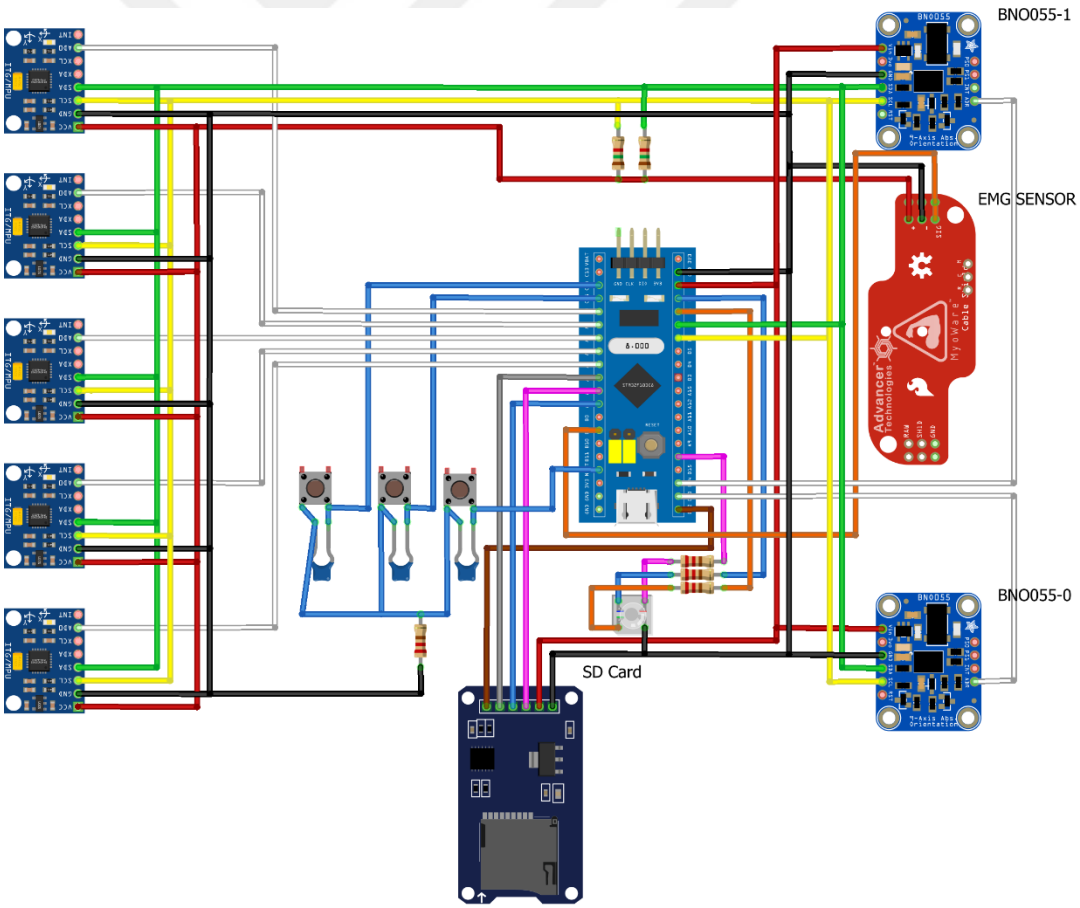


Figure 4.9 Wiring scheme

In Figure 4.10, it is understood that, capacitors C1, C2, C3 were used for hardware debouncing of the buttons. Due to the open collector structure of I2C protocol (mentioned in 5.1), the I2C pull up resistors R1 and R2 were used in the SDA and SCL lines. For the RGB LED, the resistors R3, R4 and R5 were used for current limitation.

In Figure 4.8, it could be seen that, each pin has different characteristics such as being PWM, analog/digital input/output, SPI, I2C, USART pin etc. The pin characteristics are the unique feature of that MCU and for each purpose, that specific pin should be used. For instance, instead of using the dedicated PWM pins, the LED control could be done by any digital output pin. However, it requires a polling operation on that pin which reduces the performance significantly. For the buttons, inertial pull up resistors of the MCU were activated therefore no need the external ones.

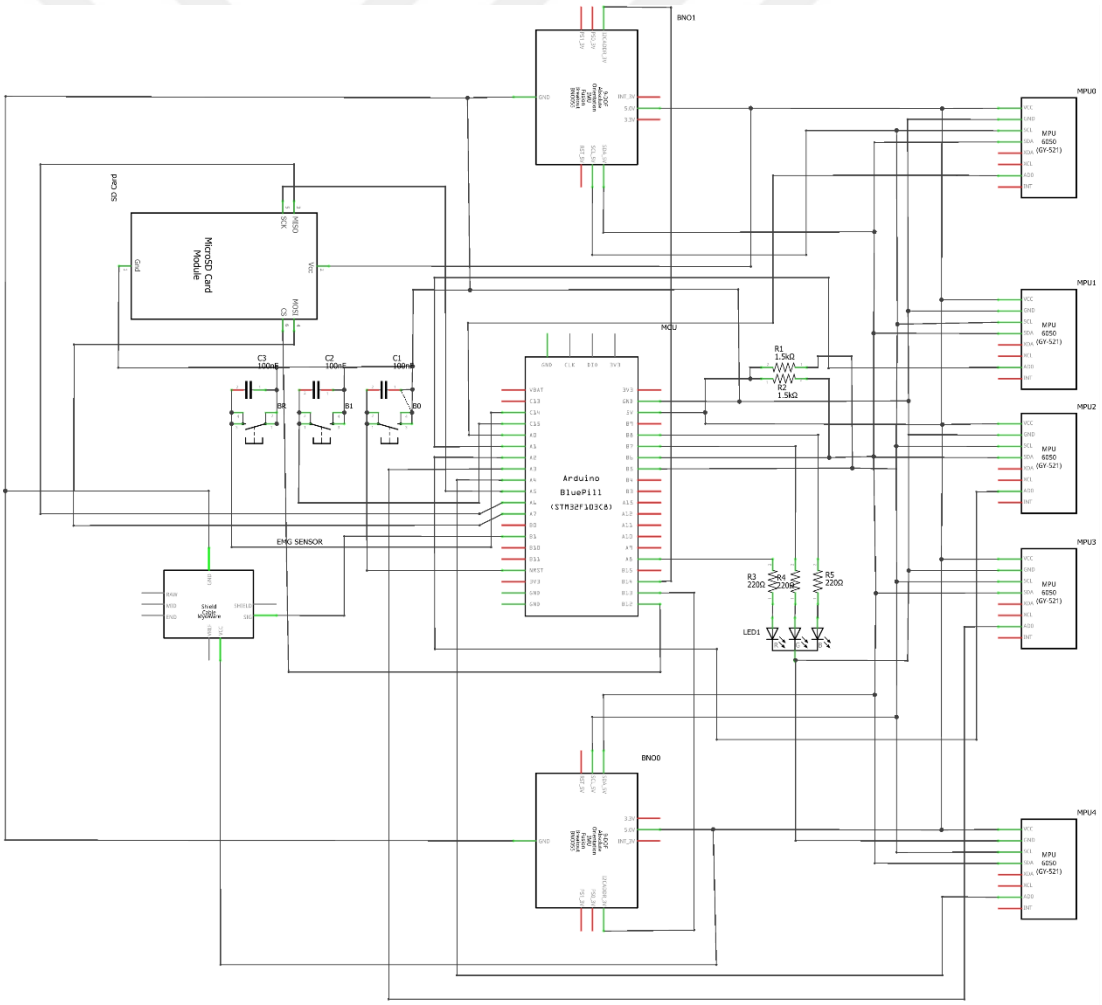


Figure 4.10 Circuit scheme

CHAPTER 5

COMMUNICATION PROTOCOLS

5.1 I2C Protocol

I2C is a communication protocol invented by Philips Semiconductors which uses two wires named SDA and SCL. SDA pin is responsible for the data transfer while SCL is responsible for the clock pulse which is generated by the master device. It is very easy to connect lots of devices through these two wires by connecting them as parallel (Figure 5.1). The communication is maintained by the master device as seen in Figure 5.2.

The SDA and SCL pins have open drain feature that's why we need to supply the voltages to these pins through pull up resistors.

As seen in Figure 5.2, I2C protocol carries 7 bits of I2C slave address and 8 bits of register address and 8 bits of data for register write operation; 7 bits of address, 8 bits of register address, 8 bits of register address and then 8 bits of data for read operation. This feature of I2C protocol makes it slower comparing it by the other protocols. The clock frequency of this protocol is 100KHz for standard mode and 400KHz for fast mode. For the best performance, the fast mode I2C is decided.

One of the important things about the I2C protocol is to know that, the components having the same manufacturer and the model number will have the same I2C address which come up with a problem named "I2C address conflict". In order to deal with this problem, an address pin was used. By applying logical 0 or 1 to that pin, the I2C address of that device will change into another state. Due to the fact that, each I2C devices have two alternative addresses, the address of the device concerned at that time should be changed into the state 0 and the remaining ones should stay as 1 or the vice versa. In other words, only one device should have different address than others at once.

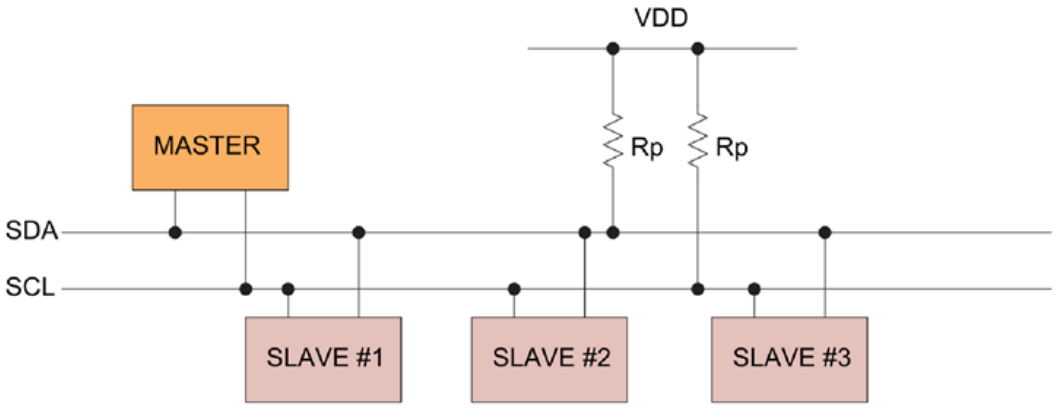


Figure 5.1 I2C connection structure[30]

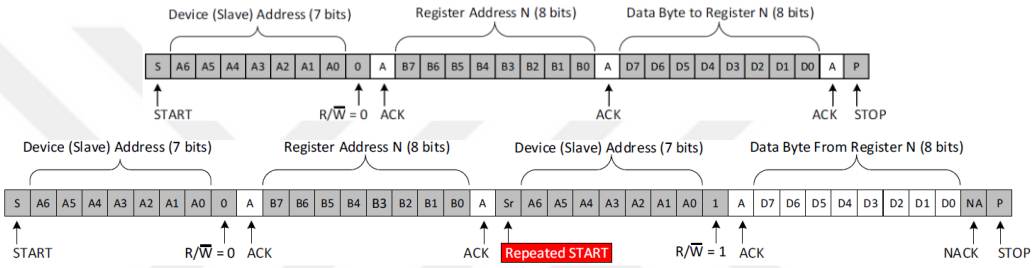


Figure 5.2 I2C data structure (write mode on top read mode on bottom)[31]

5.2 SPI

SPI is a communication protocol invented by Motorola. Which is very fast according to I2C (up to 100MHz) and full duplex which means, data could be transferred from master to slave and slave to master at the same time. There is no limit with the data length to be read and to be written. However, it uses 4 connections (see Figure 5.3) for each device while I2C requires 2 or 3 (to avoid address conflict). SCK is generated by the master device. MOSI and MISO means, the data sent from master to slave and slave to master respectively. CS which sometimes named as SS (Slave Select) pin, is for selecting the device to be communicated.

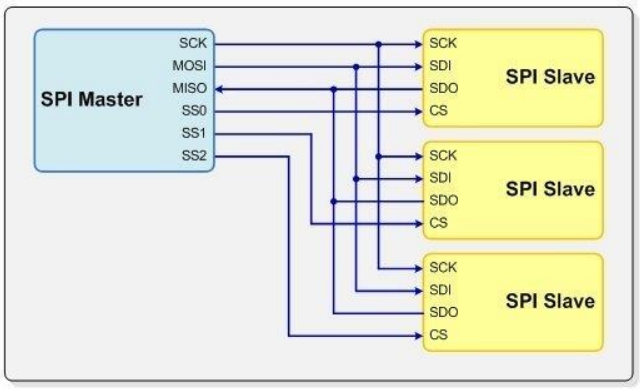


Figure 5.3 SPI communication structure[32]

5.3 USART

The USART protocol is very fast (up to 4.5Mbit/s) and very easy to use protocol which uses two wires for communication as seen in Figure 5.4. The data frame of this protocol could be seen in Figure 5.5. In this protocol, the clock signal could be restored from the data sequence, therefore there is no need an additional clock pin or no need to specify the baud rate to the decoder side, while for the UART communication require this specification.

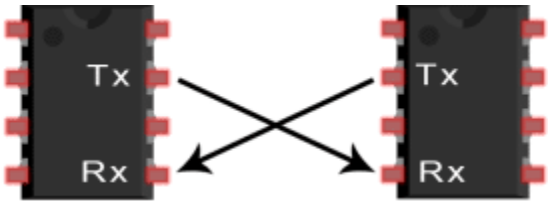


Figure 5.4 USART communication structure[33]

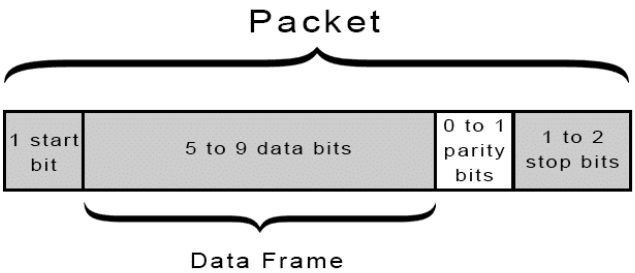


Figure 5.5 USART data frame[33]

CHAPTER 6

HARDWARE CREATION

In Figure 6.1, the full framework of the device could be seen. In order to fix the sensors over the fingertips, hand and upper arm, using the velcro tapes were considered. These velcro tapes were fixed below the sensors using hot glue. The cables were isolated using the heat shrink tubing. Two programmable buttons and one hardware reset button and one RGB LED were placed to the velcro tape positioned below the MCU. The 6-DOF IMUs are positioned on the thumb, index, middle, ring and little finger with respect to S_0, S_1, S_2, S_3, S_4 notations. The 9 DOF IMUs are positioned on the upper arm and back hand with respect to N_0 and N_1 notations.

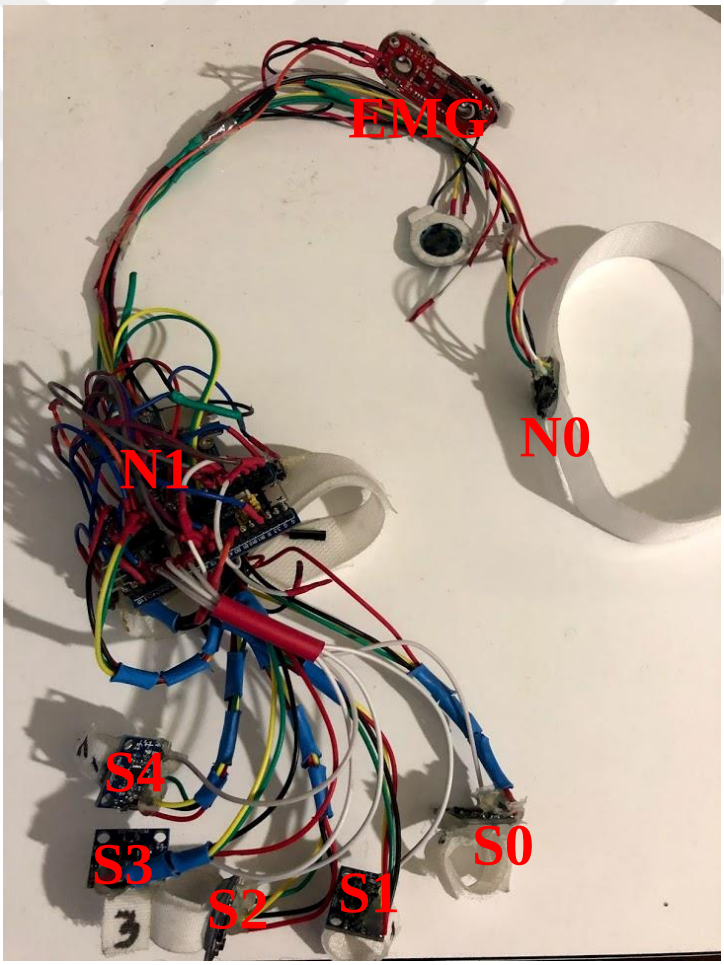


Figure 6.1 Full framework

CHAPTER 7

DATA COLLECTION AND ANALYSIS

The gyroscope provides the rotational acceleration of the sensor in degrees per seconds. The accelerometer provides the linear acceleration of the sensor in m/s^2. The magnetometer provides the magnetic effect to which the sensor is exposed, in terms of microteslas, and will be used in this study because of the effect of the Earth's magnetic field on the sensor. The EMG sensor also provides the ion exchange rate in the muscles analogously in terms of voltage. The data provided are raw in this state. However, since these raw data require a large cost and increase the size of the data, they are interpreted by the microcontroller inside the sensors to provide individual orientation data. Orientation can be presented as Euler angles or quaternion data. In order to provide this orientation data, some algorithms that are embedded in the sensors are involved. In a Kalman Filter design [34], which is very popular in the literature and frequently used in such applications, a 9-DOF inertial sensor can obtain orientation data with minimal drift error. The signal produced by the EMG sensor is an analog low frequency signal with a range of 0V-5V and a reference point of about 2.5 V. This signal is first rectified by taking the absolute value of less than 2.5 volts and the signal is scaled between 0-5 Volts. After that, the envelope of this rectified signal appears as a smooth function. Thanks to this process, the power of this generated signal could be estimated with a low-cost approach.

In this study, the collected gestures will accumulate and construct phrases. Therefore, the **gestures** could be considered as **phrases**.

7.1 Data Types

From the 6 and 9-DOF sensor units, the quaternion data is obtained, the linear acceleration in the xyz axes in 3-dimensional space from each of these sensors and the "approximate energy consumed by the arm muscle" component obtained from the EMG sensor. In the notation d_{nt}^{im}, d points out the quaternion (q) or acceleration (a) or the raw data type taken (EMG) from the EMG sensor, n points out the component number (quaternion 0, acceleration 0 etc.), i points out 9 axes (N) BNO055 or 6 axes

(S) MPU6050 measurement unit, m: ID of the measurement unit (S0, N1 etc.), t: time instance of the chunk. This data block can be seen in Figure 7.1.

Data is collected as one **chunk (C_t)**. Each chunk has the sensor measurements C_m. By the combination of each chunk, a **phrase** is obtained as seen in Figure 7.1. Each chunk has a time label in terms of milliseconds which the time instance of the chunk created which is indicated as *t*. While collecting the data, the phrase and the repetition of each phrase is counted by pressing the buttons. The button on the left-hand side is responsible for the repetition counter and the right-hand side is responsible for the phrase counter. In the notation rep_t, *rep* shows the number of repetitions of that phrase and in the notation id_t, *id* shows the ID number of that phrase. The repetition number and the ID number of that phrase is controlled by pressing the left and right push buttons on the hardware respectively.

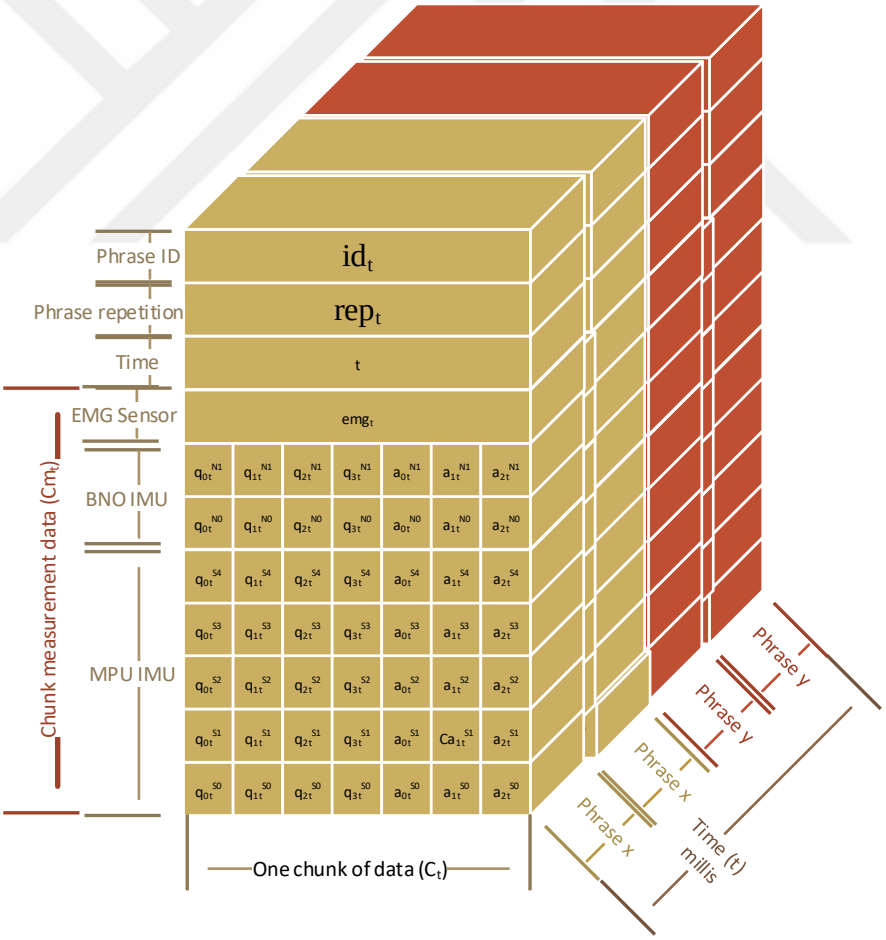


Figure 7.1 Data structure

The variable types for q_{nt}^{Sm} and q_{nt}^{Nm} are 32 bits of floating number. a_{nt}^{Sm} is signed 16 bits integer while a_{nt}^{Nm} is 32 bits of double number. emg_t is 16 bits of unsigned integer number. t is 32 bits of unsigned integer number while id_t and rep_t are 16 bits of unsigned integer numbers. However, while logging the data via SD card or communication terminal window, it is a fact that, they are stored as ASCII characters and thus, each character occupies 1 byte of memory location. In order to reduce the data storage cost and standardize the data, 8 digits after the floating point were taken into the account. In addition to these data, also a "TAB" character is used as separator. Consequently, size of one chunk is approximately 490-500 bytes according to the ASCII stream.

7.2 Data Collection

The collection of data is made by user. These phrases were recorded when the framework is on the right hand. For this data collection session and test, 3 phrases with 3 repetitions for each were considered. Test phrases could be seen in Figure 7.2. Between each **phrase** and **repetition**, the arm and hand were positioned at the **idle** position and all the phrases started and finished by beginning with idle position and returning to the idle position. The starting and ending points of each phrase indicated as red and green colored arrows respectively. Phrase 0 is simply spinning the hand one time in air when fingers are free. Phrase 1 is made by making the hand into a fist slowly. Phrase 2 is made by making the hand into a fist and making the thumb finger touch to the lip. When the phrase is finished, the buttons on the framework were pressed in order to give an ID and repetition information to that phrase.

Figure 7.2 Idle, Phrase 0, Phrase 1 and Phrase 2 respectively

7.3 Preprocessing

Since the inertial sensors do not produce data in real time, these data need to be uniformly sampled. The sampling frequency of each inertial unit is approximately 100-200 Hz. However, due to the restrictions of the microprocessor and some calculation costs, the sampling frequency reduced into approximately 20Hz. A destination sampling period *s* is set before the collection of the chunks start. In Figure 7.3, a chunk stream could be seen. In notation, *t* denotes the time instance of the chunk created. *Cm* denotes the measured sensor value. T denotes the Time buffer, B denotes the buffer which could be *L*, *D* and *U*. *L* stands for the low buffer, *D* stands for the destination buffer, in td_t, td denotes the current destination time, *s* is sampling period, *U* stands for the upper buffer. For Figure 7.3, instance *t=1* overlaps with the destination period; however, no chunk was captured in that time instance. In order to deal with this problem, two chunk data nearest to the destination period are evaluated according to the Equation 6. $Tt_{d,t}^{L}$ and $Tt_{d,t}^{U}$ values are taken directly from the nearest two values. $Cm_{td,t}$ value is the expected value (but not found). This value is taken from the buffer $Cm_{td,t}^{D}$.

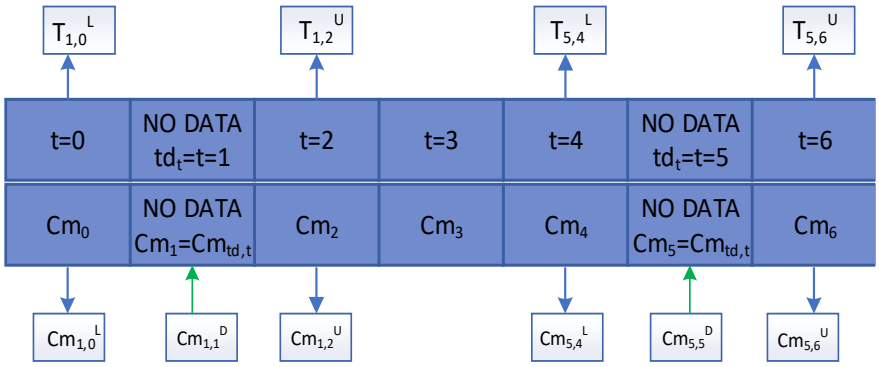


Figure 7.3 Uniformly sampling process

$$Cm_{td_t} = Cm_{td_t}^{D} = \begin{cases} Cm_{td_t}^{L} + \left(Cm_{td_t}^{U} - Cm_{td_t}^{L}\right) \times \dfrac{td_t - T_{td_t}^{L}}{T_{td_t}^{U} - T_{td_t}^{L}}, & t = \dfrac{td_t}{2} \wedge t = \dfrac{3 \times td_t}{2} \\ Cm_0, & td_t = 0 \\ Cm_t, & td_t = t \end{cases} \quad (6)$$

After the uniform sampling process, seen that, some impulse noises occur in the samples. These impulse noises are very high values compared to the nominal values taken from the IMUs. During the normalization process, these impulse noises will have the highest or lowest values of our covariance matrix and this could generate a

misleading information while calculating our covariance matrix; consequently, the feature vectors. In order to deal with this problem, a 1-D 5th order median filter is used to eliminate these impulse noises. After the elimination of the impulse noises, all the sensor readings should be normalized between 0 and 1 to avoid the misleading information of high valued readings of the sensor on the covariance matrix. Then, it is very efficient to apply a PCA on that uniformly sampled, median filtered and normalized data.

7.4 Principal Component Analysis (PCA)

PCA is a dimension reduction technique in order to obtain the feature vector for classification applications. The main idea behind the PCA is to obtain the covariance matrix of the dataset and finding the eigenvectors and eigenvalues of it. The biggest eigenvalue corresponds to its eigenvector means, that eigenvector is the most important feature vector while reconstructing the data.

The values used in covariance matrix are uniformly sampled, median filtered and normalized ones. For the covariance matrix, there exists 50 sensor readings and 664 time instances for each. Thus, the size of the covariance matrix is 50 by 50. Each element of the covariance matrix is calculated by comparing two sensor readings. In equation 7, X denotes the first sensor reading to be compared, Y denotes the second sensor reading to be compared and $\bar{X}, \bar{Y}$ denotes the mean value of the corresponding sensor readings. Since there are 664 time instances for each sensor readings, the value of n=664. After comparing each sensor readings two by two, our covariance matrix could be generalized as seen in Table 7.1. It should be noted that, this table is the generalized version of covariance matrix which has the size 50 by 50; not 7 by 7.

$$cov(X,Y) = \sum_{i=1}^{n} \frac{(X_i - \bar{X}) * (Y_i - \bar{Y})}{(n-1)} \tag{7}$$

Table 7.1 Generalized covariance matrix

cov(emg,emg)	cov(emg,N^0)	cov(emg,N^I)	cov(emg,S^0)	cov(emg,S^I)	cov(emg,S^2)	cov(emg,S^3)
cov(N^0,emg)	cov(N^0,N^0)	cov(N^0,N^I)	cov(N^0,S^0)	cov(N^0,S^I)	cov(N^0,S^2)	cov(N^0,S^3)
cov(N^I,emg)	cov(N^I,N^0)	cov(N^I,N^I)	cov(N^I,S^0)	cov(N^I,S^I)	cov(N^I,S^2)	cov(N^I,S^3)
cov(S^0,emg)	cov(S^0,N^0)	cov(S^0,N^I)	cov(S^0,S^0)	cov(S^0,S^I)	cov(S^0,S^2)	cov(S^0,S^3)
cov(S^I,emg)	cov(S^I,N^0)	cov(S^I,N^I)	cov(S^I,S^0)	cov(S^I,S^I)	cov(S^I,S^2)	cov(S^I,S^3)
cov(S^2,emg)	cov(S^2,N^0)	cov(S^2,N^2)	cov(S^2,S^0)	cov(S^2,S^I)	cov(S^2,S^2)	cov(S^2,S^3)
cov(S^3,emg)	cov(S^3,N^0)	cov(S^3,N^I)	cov(S^3,S^0)	cov(S^3,S^I)	cov(S^3,S^2)	cov(S^3,S^3)

As seen in equation 8, *A* is a matrix, *v* is an eigenvector, λ is an eigenvalue. That means, that matrix could be decomposed into its eigenvectors and eigenvalues. In this study, *A* will be the covariance matrix. The eigenvalues and eigenvectors of the covariance matrix are found by using equations 9 and 10.

$$Av = \lambda v \quad (8)$$

$$|A - \lambda I| = 0 \quad (9)$$

$$(A - \lambda I)v = 0 \quad (10)$$

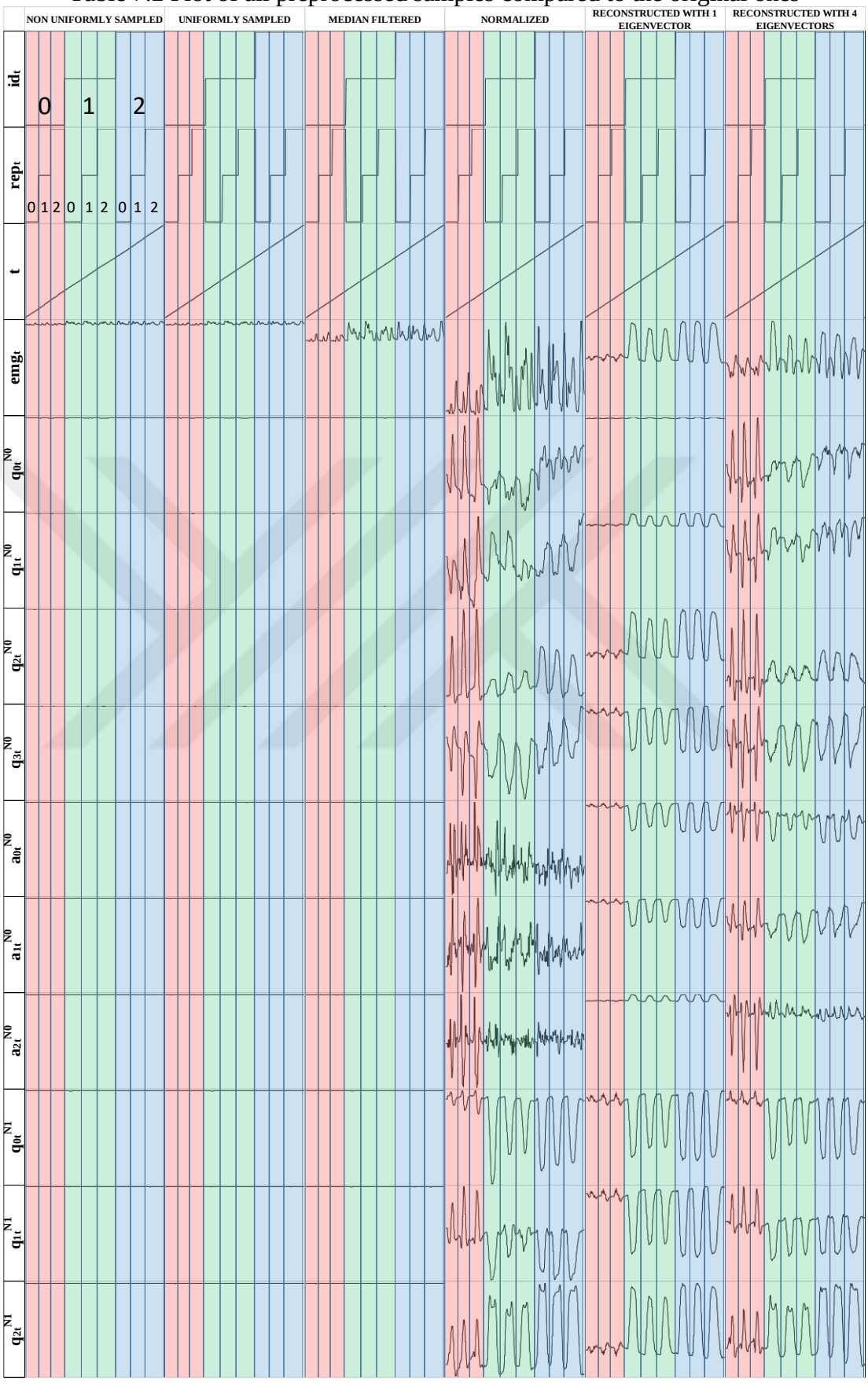
The multiplication of the uniform sampled, median filtered, normalized and mean subtracted 664*50 dataset with the unsorted 50*50 eigenvector matrix will give the **score matrix**.

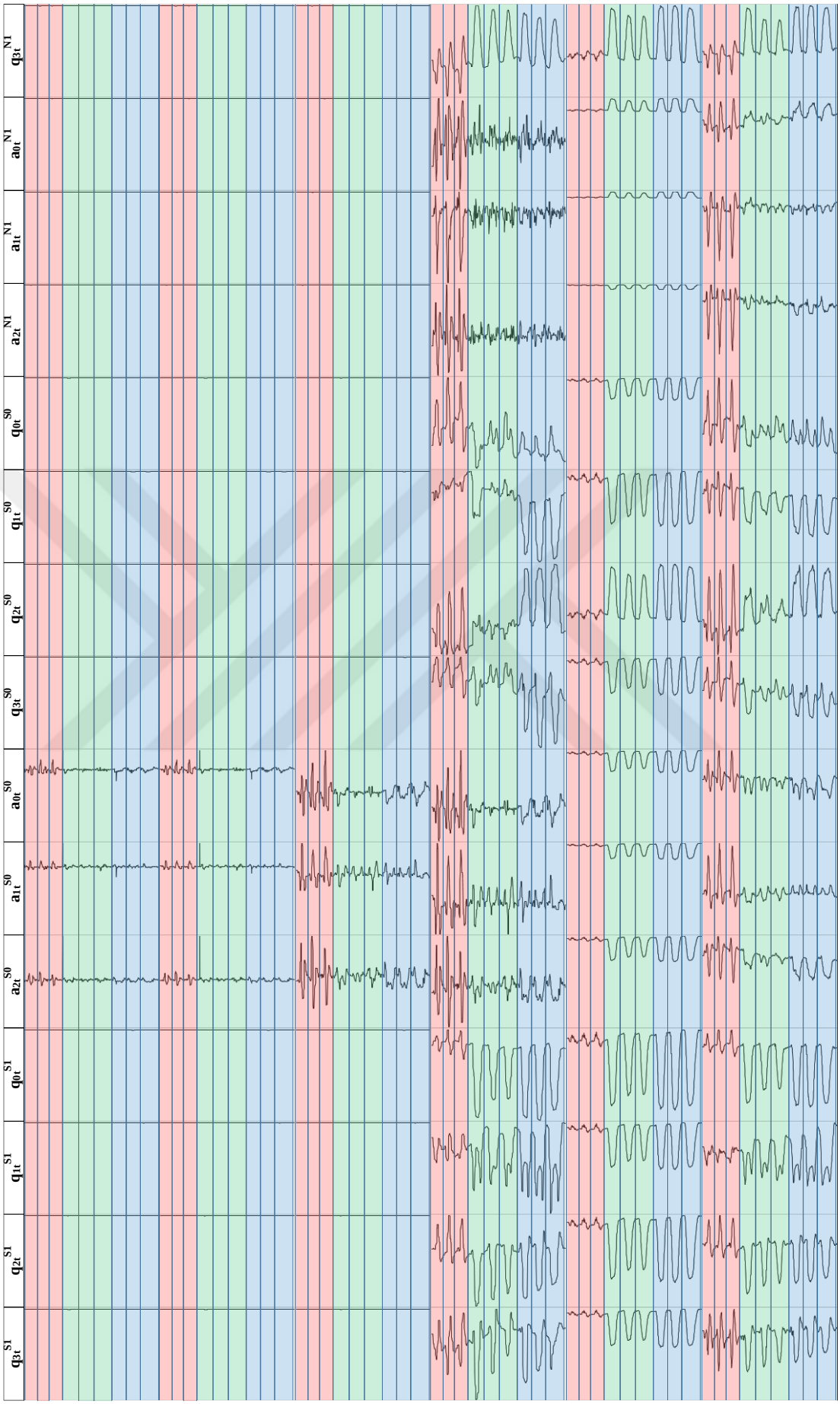
After finding the eigenvectors and eigenvalues of the covariance matrix, these eigenvectors and corresponding score matrix columns should be sorted according to their eigenvalues in descending order. Each of these eigenvectors are now our **feature vectors.** The first eigenvector is our most important feature vector for data reconstruction.

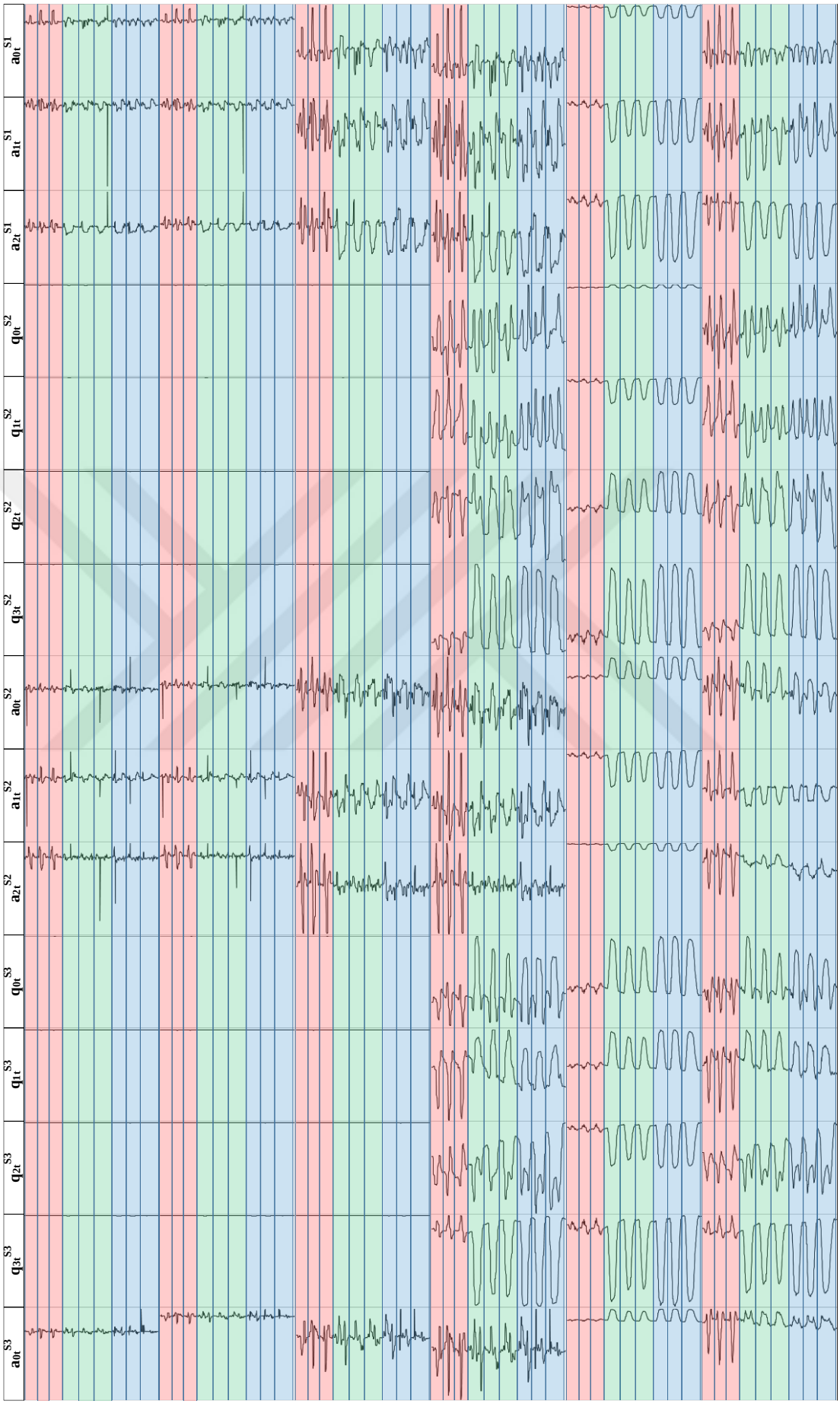
Reconstruction of data is made by multiplying the score matrix with the corresponding transpose of the eigenvectors and adding the mean values of the uniformly sampled, median filtered and normalized dataset. The data size reduction is made by taking the most important eigenvectors into the account and eliminating the remaining ones. All

the process of preprocessing and the reconstruction of each sensor readings could be seen in Table 7.2.

Table 7.2 Plot of all preprocessed samples compared to the original ones




q_{3t}^{N1}
a_{0t}^{N1}
a_{1t}^{N1}
a_{2t}^{N1}
q_{0t}^{S0}
q_{1t}^{S0}
q_{2t}^{S0}
q_{3t}^{S0}
a_{0t}^{S0}
a_{1t}^{S0}
a_{2t}^{S0}
q_{0t}^{S1}
q_{1t}^{S1}
q_{2t}^{S1}
q_{3t}^{S1}


S1 a0t
S1 a1t
S1 a2t
S2 q0t
S2 q1t
S2 q2t
S2 q3t
S2 a0t
S2 a1t
S2 a2t
S3 q0t
S3 q1t
S3 q2t
S3 q3t
S3 a0t

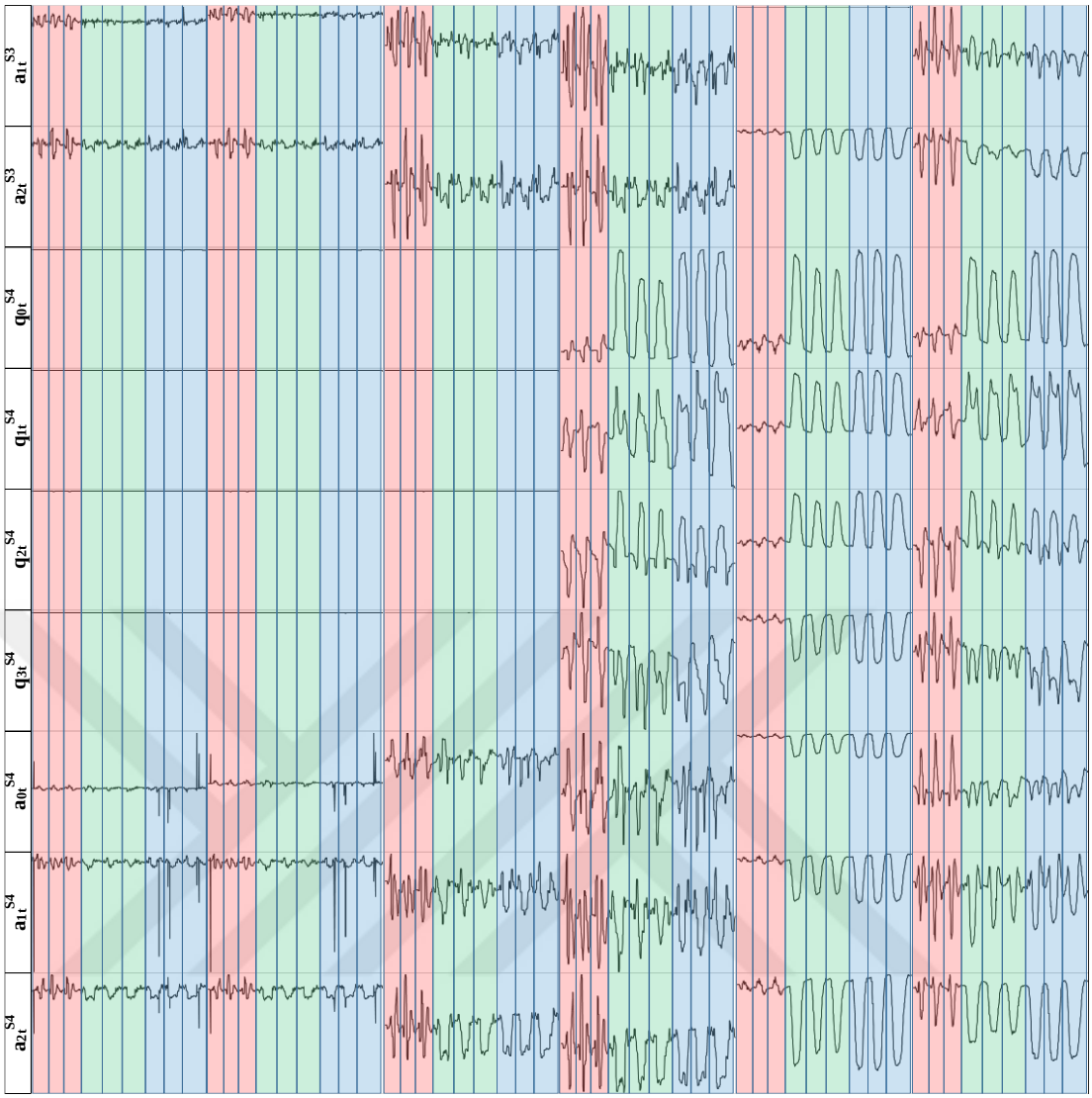

a_{1t}^{S3}
a_{2t}^{S3}
q_{0t}^{S4}
q_{1t}^{S4}
q_{2t}^{S4}
q_{3t}^{S4}
a_{0t}^{S4}
a_{1t}^{S4}
a_{2t}^{S4}

Table 7.3 Importance and the importance percentage of each eigenvector

Eigenvector	Eigenvalue	Percentage
1	1.057611455	47.33%
2	0.285467973	12.77%
3	0.245179671	10.97%
4	0.157654429	7.05%
5	0.096394830	4.31%
6	0.069378591	3.10%
7	0.047133631	2.11%
8	0.039059124	1.75%
9	0.037002841	1.66%
10	0.030958708	1.39%
11	0.019355383	0.87%
12	0.015494884	0.69%
13	0.014155975	0.63%
14	0.012636637	0.57%
15	0.010070038	0.45%
16	0.008566713	0.38%
17	0.007991926	0.36%
18	0.007224386	0.32%
19	0.006437630	0.29%
20	0.006105915	0.27%
21	0.005614336	0.25%
22	0.004912390	0.22%
23	0.004552224	0.20%
24	0.004315847	0.19%
25	0.004170871	0.19%
26	0.003467997	0.16%
27	0.003353078	0.15%
28	0.003249749	0.15%
29	0.002915858	0.13%
30	0.002628223	0.12%
31	0.002428110	0.11%
32	0.002303797	0.10%
33	0.002022267	0.09%
34	0.001780567	0.08%
35	0.001691687	0.08%
36	0.001538165	0.07%
37	0.001412818	0.06%
38	0.001250142	0.06%
39	0.001210383	0.05%
40	0.001092220	0.05%
41	0.001016238	0.05%
42	0.000875673	0.04%
43	0.000868541	0.04%
44	0.000594872	0.03%
45	0.000414261	0.02%
46	0.000390353	0.02%
47	0.000270360	0.01%
48	0.000265648	0.01%
49	0.000149682	0.01%
50	0.000091800	0.00%

After the reconstruction of data with 4 eigenvectors, 4 score matrix columns and 4 mean values, 78.12% of the information of original data pattern could be restored.

7.5 About Program Code

Some of the program code is shown as flowchart in APPENDIX A.

C++ based Arduino language has two base functions named main() and loop(). These two functions are gathered together as one main() function in the flowchart in order to describe them easily. Some of the libraries belonging to sensors, LED animation and buttons were taken from the GitHub sources.

CHAPTER 8

RESULTS AND DISCUSSION

A smart glove system has been created for the solutions that can be developed in this regard by analyzing the studies in the literature and identifying the weak points. As concluded from Table 7.2, each phrase has unique pattern. Implementing a uniformly sampling algorithm does not significantly change the characteristics of this pattern. After the uniformly sampling of data, there are 664 samples (time instances) for each sensor measurement. This means, 664*50=33200 samples in total. However, due to the calculation cost of the classifiers, it is not efficient to take all these samples as feature vectors. With median filtering and normalization processes, the misleading bias factor of covariance matrix was eliminated. After the PCA, seen that, by checking the eigenvalues, approximately 78% of the information generated by the sensors could be represented by 4 eigenvectors with size 50 samples each. Consequently, the 33200 samples in total could be reduced into 200 samples by taking 4 eigenvectors as feature vectors with a small loss of information. Consequently, a data collection system was implemented, the non-uniform sampling issues, noises and biases of the signals were eliminated and the PCA which is a data reduction technique was applied and the sensors were fused into the feature vectors. A small dataset for the classification and other applications was generated and analyzed.

This smart glove system is planned to be used for the development of the first inertial and EMG sensor based Turkish Sign Language dataset in the future. In addition, for different data types which can be obtained from these inertial and EMG sensors, different classification methods will be analyzed and the results of these analyzes will be obtained.

BIBLIOGRAPHY

[1] A. Chaudhary, J. . Raheja, K. Das, and S. Raheja, “Intelligent Approaches to interact with Machines using Hand Gesture Recognition in Natural way: A Survey,” *Int. J. Comput. Sci. Eng. Surv.*, vol. 2, no. 1, pp. 122–133, 2011.

[2] B. Demircioğlu, G. Bülbül, and H. Köse “Turkish Sign Language Recognition With Leap Motion,” pp. 0–3, 2016.

[3] R. R. Itkarkar, “A Survey of 2D and 3D Imaging Used in Hand Gesture Recognition for Human-Computer Interaction (HCI),” no. December, pp. 19–21, 2016.

[4] S. Mitra and T. Acharya, “Gesture recognition: A survey,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, 2007.

[5] U. Orhan, K. E. Hild II, D. Erdogmus, B. Roark, B. Oken, and M. Fried-Oken, “RSVP KEYBOARD: AN EEG BASED TYPING INTERFACE.”

[6] “15. Öğrenci Proje Sergisi Fakülte Proje Üçüncüsü, Elektrik ve Elektronik Mühendisliği Bölümü - YouTube,” 2018. [Online]. Available: https://www.youtube.com/watch?v=Byo2CmzxpXI. [Accessed: 01-May-2019].

[7] Erhan Akan, “Görme engelliler için akıllı eldiven geliştirdi - Anadolu Ajansı,” *2015*. [Online]. Available: https://www.aa.com.tr/tr/vg/video-galeri/gorme-engelliler-icin-akilli-eldiven-gelistirdi. [Accessed: 01-May-2019].

[8] M. M. Zaki and S. I. Shaheen, “Sign language recognition using a combination of new vision based features,” *Pattern Recognit. Lett.*, vol. 32, no. 4, pp. 572–577, 2011.

[9] S. Abbas Memi¸Albayrak, “Kinect RGB Görüntülerde ve Derinlik Haritalarında Uzam-zamansal Özellikleri Kullanarak Türk ˙ I ¸ saret Dili Tanıma Turkish Sign Language Recognition Using Spatio-temporal Features on Kinect RGB Video Sequences and Depth Maps,” 2013.

[10] A. Golliwar, H. Patil, R. Watpade, S. Moon, S. Patil, and V. D. Bondre, “Sign Language Translator Using Hand Gloves,” *Int. J. Electr. Electron. Comput. Syst.*, vol. 2, no. 1, pp. 88–91, 2014.

[11] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, “uWave: Accelerometer-based personalized gesture recognition and its applications,” *Pervasive Mob. Comput.*, vol. 5, no. 6, pp. 657–675, Dec. 2009.

[12] E. Akan, H. Tora, and B. Uslu, “Hand gesture classification using inertial based sensors via a neural network,” in *2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2017, pp. 140–143.

[13] Jian Wu, L. Sun, and R. Jafari, “A Wearable System for Recognizing American Sign Language in Real - time Using IMU and,” vol. 20, no. 5, pp. 1–10, 2016.

[14] N. Jatupaiboon, S. Pan-Ngum, P. Israsena, B.-W. Chen, S. Hsieh, and C.-H. Wu, “Real-Time EEG-Based Happiness Detection System,” *Sci. World J.*, 2013.

[15] J. S. Kim, W. Jang, and Z. Bien, “A dynamic gesture recognition system for the Korean Sign Language (KSL),” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 26, no. 2, pp. 354–359, 1996.

[16] F. T. Liu, Y. T. Wang, and H. P. Ma, “Gesture recognition with wearable 9-axis sensors,” in *IEEE International Conference on Communications*, 2017, pp. 1–6.

[17] H. P. Gupta, H. S. Chudgar, S. Mukherjee, T. Dutta, and K. Sharma, “A Continuous Hand Gestures Recognition Technique for Human-Machine Interaction Using Accelerometer and Gyroscope Sensors,” *IEEE Sens. J.*, vol. 16, no. 16, pp. 6425–6432, 2016.

[18] “An introduction to Euler angles.” [Online]. Available: https://www.wikiwand.com/en/Euler_angles. [Accessed: 06-May-2019].

[19] “Gimbal Lock.” [Online]. Available: https://www.wikiwand.com/en/Gimbal_lock.

[20] J.-L. Blanco, “A tutorial on SE(3) transformation parameterizations and on-manifold optimization,” 2013.

[21] “Conversion between quaternions and Euler angles.” [Online]. Available: https://www.wikipedia.org/en/Conversion_between_quaternions_and_Euler_angles.

[22] “Pro Micro & Fio V3 Hookup Guide - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide/hardware-overview-pro-micro. [Accessed: 07-Jun-2019].

[23] “MPU-6000 and MPU-6050 Product Specification Revision 3.4 MPU-6000/MPU-6050 Product Specification,” 2013.

[24] “BNO055 Intelligent 9-axis absolute orientation sensor,” 2014.

[25] Sparkfun, “MyoWare Muscle Sensor.” .

[26] S. B. LAXMI SHAW, “Online EMG Signal Analysis for diagnosis of Neuromuscular diseases by using PCA and PNN.”

[27] “MyoWare Muscle Sensor Datasheet,” 2015.

[28] “STM32F103C8T6 Datasheet,” 2015.

[29] Stm32Duino, “STM32F103C8T6 Blue Pill.” .

[30] “I2C Connection Structure.” [Online]. Available: https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html.

[31] J. Valdez, “Understanding the I2C Bus,” 2015.

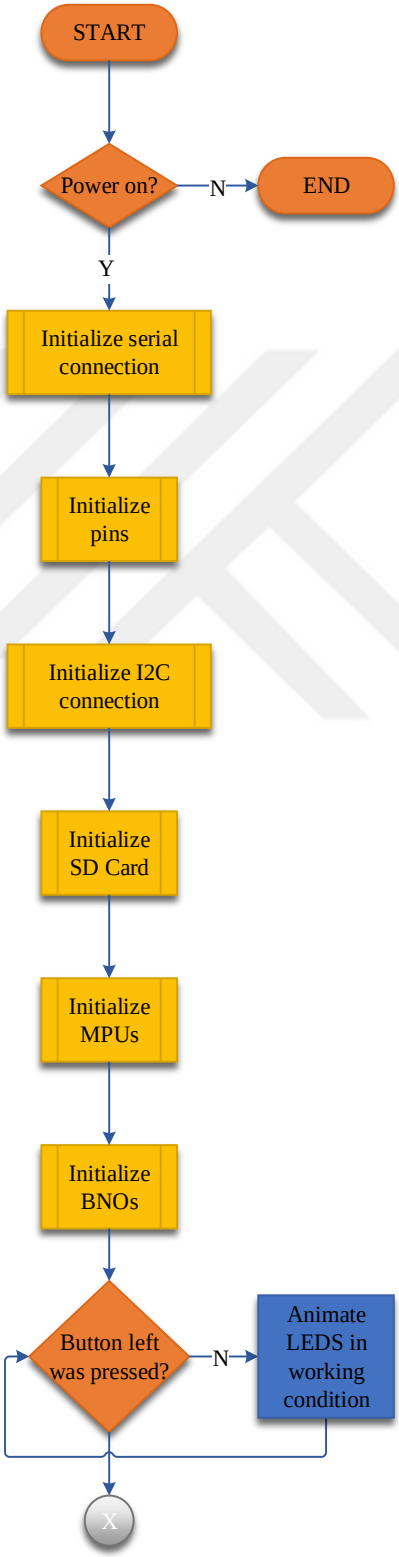
[32] “SPI Tutorial – Serial Peripheral Interface Bus Protocol Basics.” [Online]. Available: https://www.corelis.com/education/tutorials/spi-tutorial/. [Accessed: 10-Jun-2019].

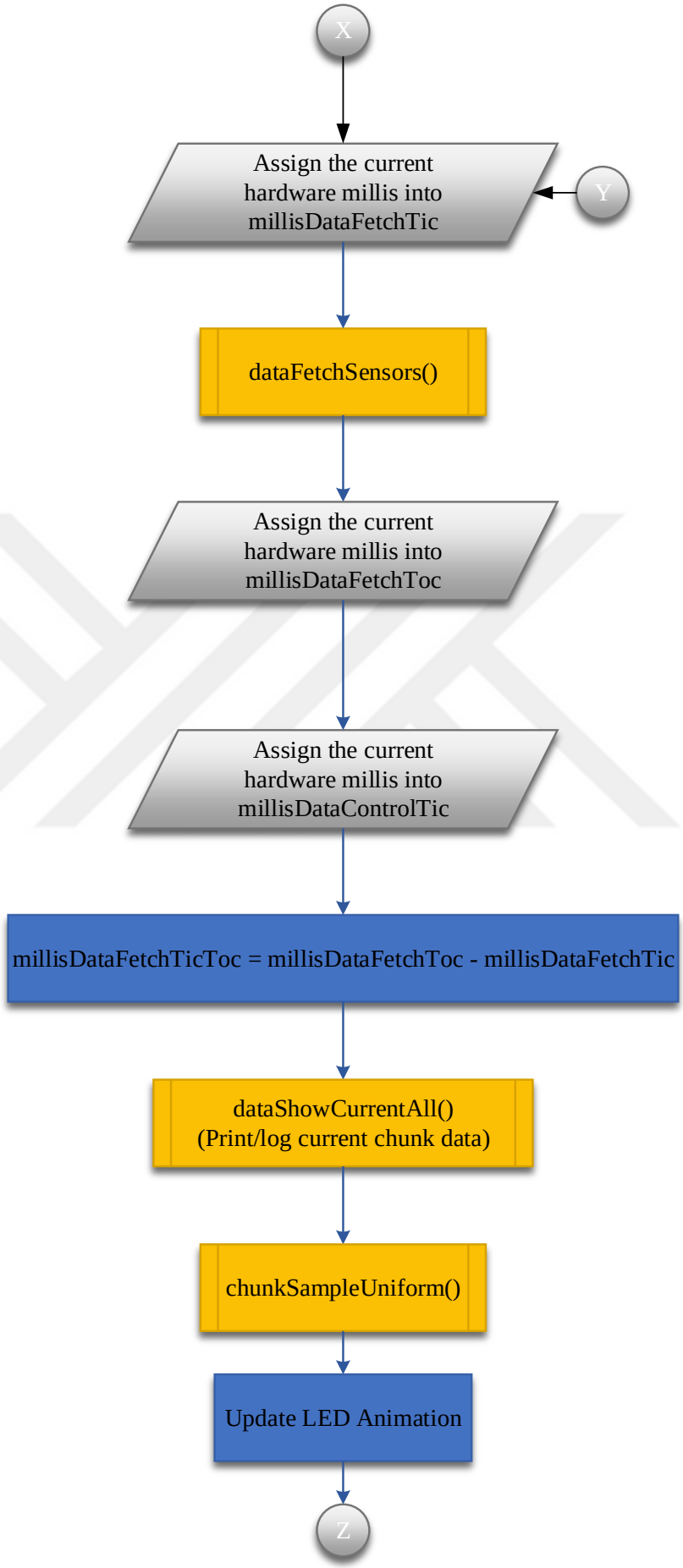
[33] “Basics of UART Communication.” [Online]. Available: http://www.circuitbasics.com/basics-uart-communication/. [Accessed: 10-Jun-2019].

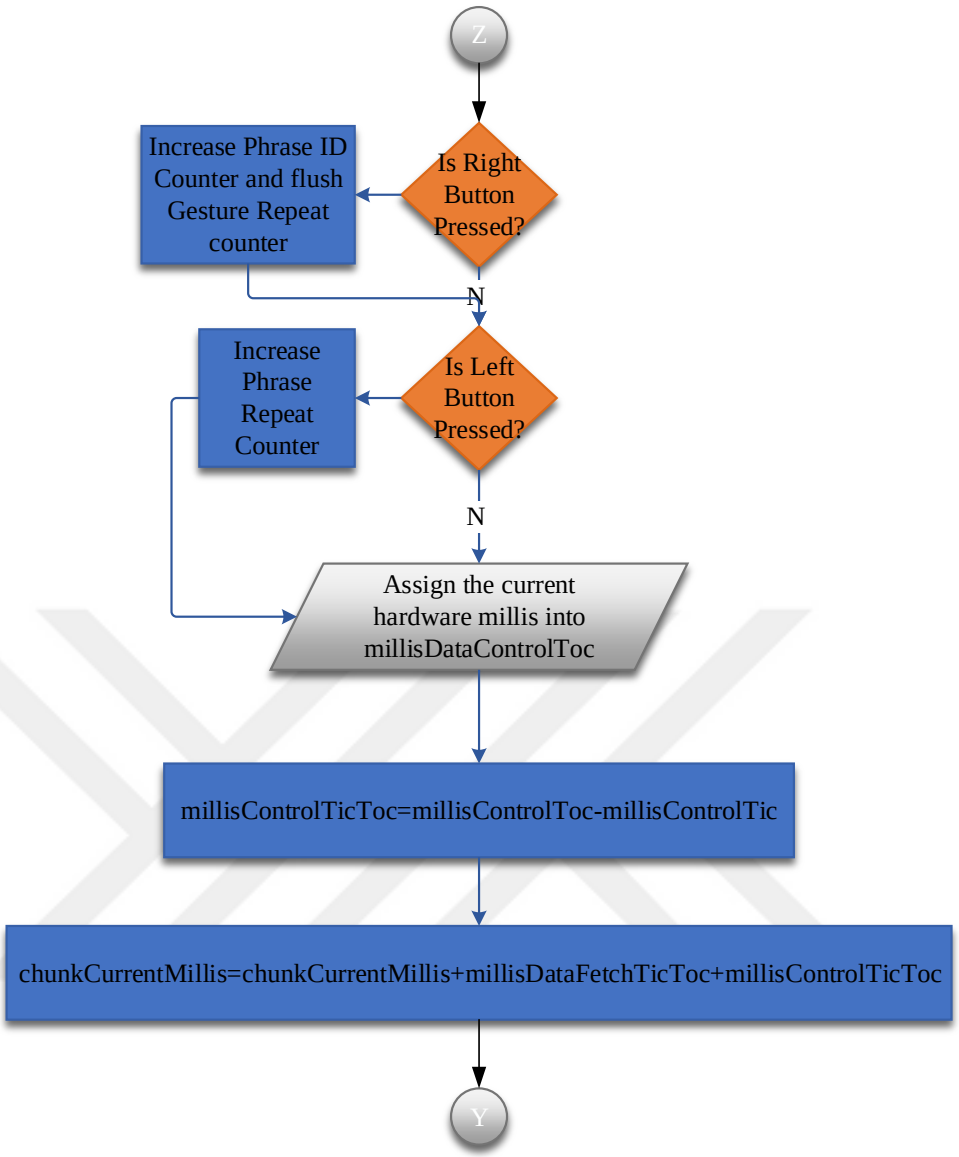
[34] D. Roetenberg, H. J. Luinge, C. T. M. Baten, and P. H. Veltink, “Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, 2005.

APPENDIX A FLOWCHART OF SOME IMPORTANT FUNCTIONS

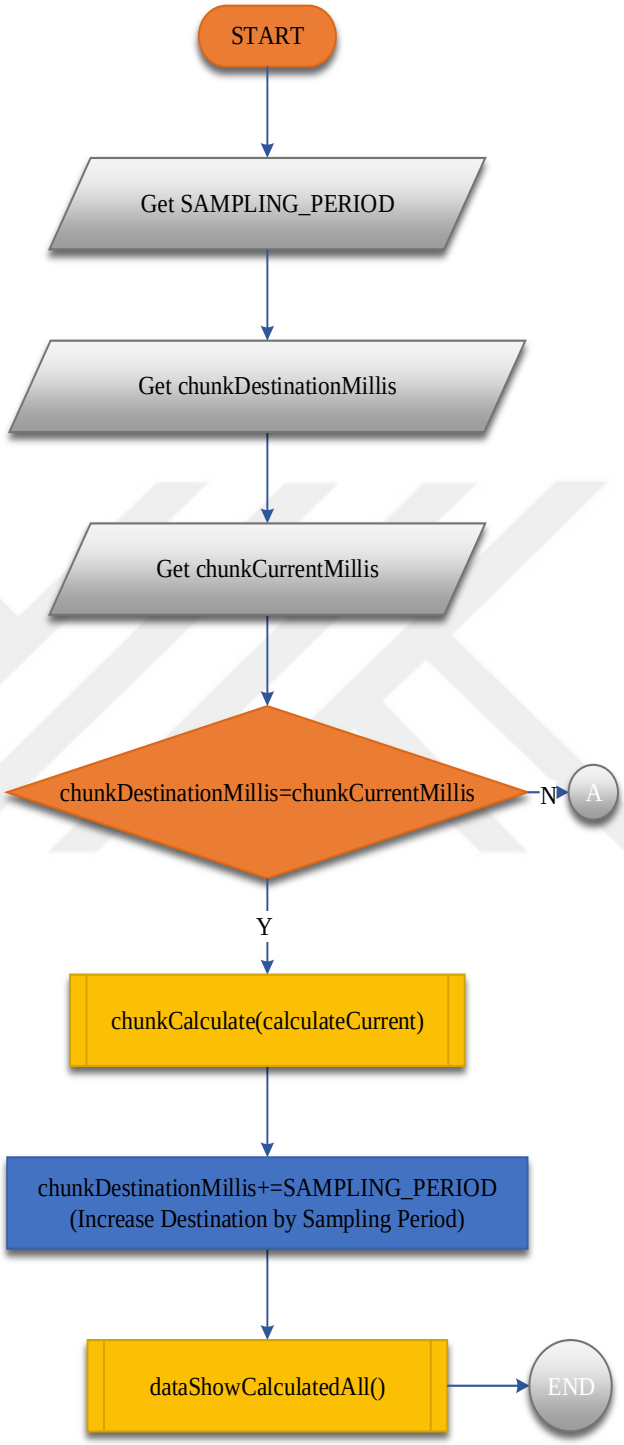
FLOWCHART OF main()

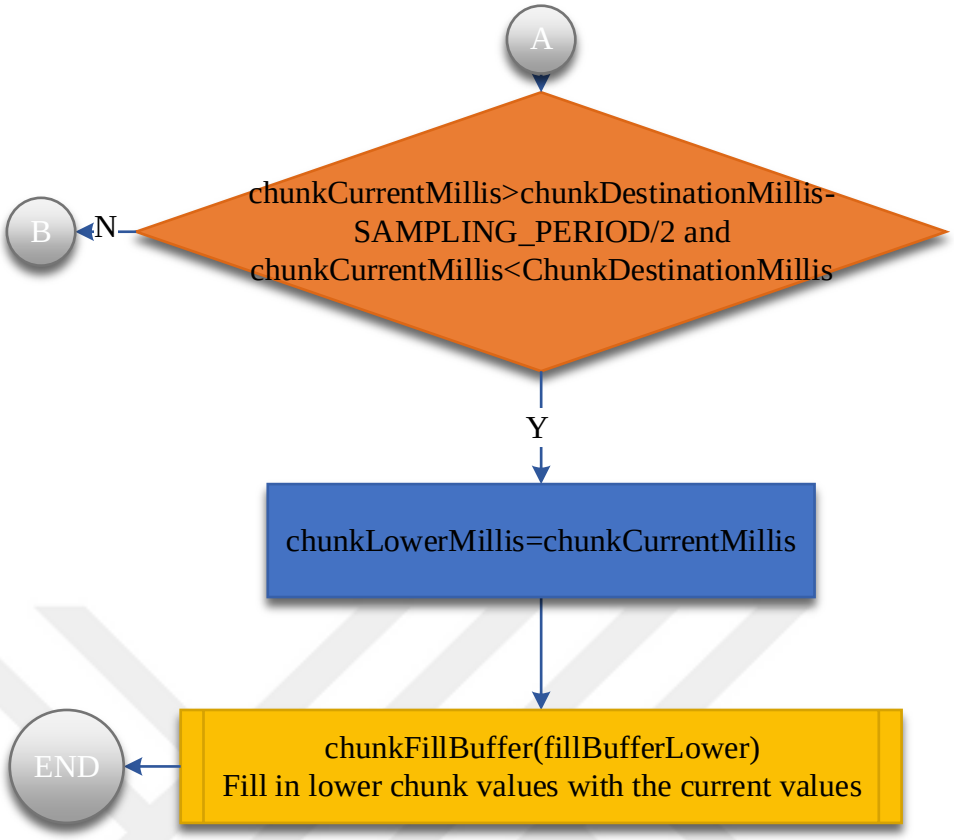


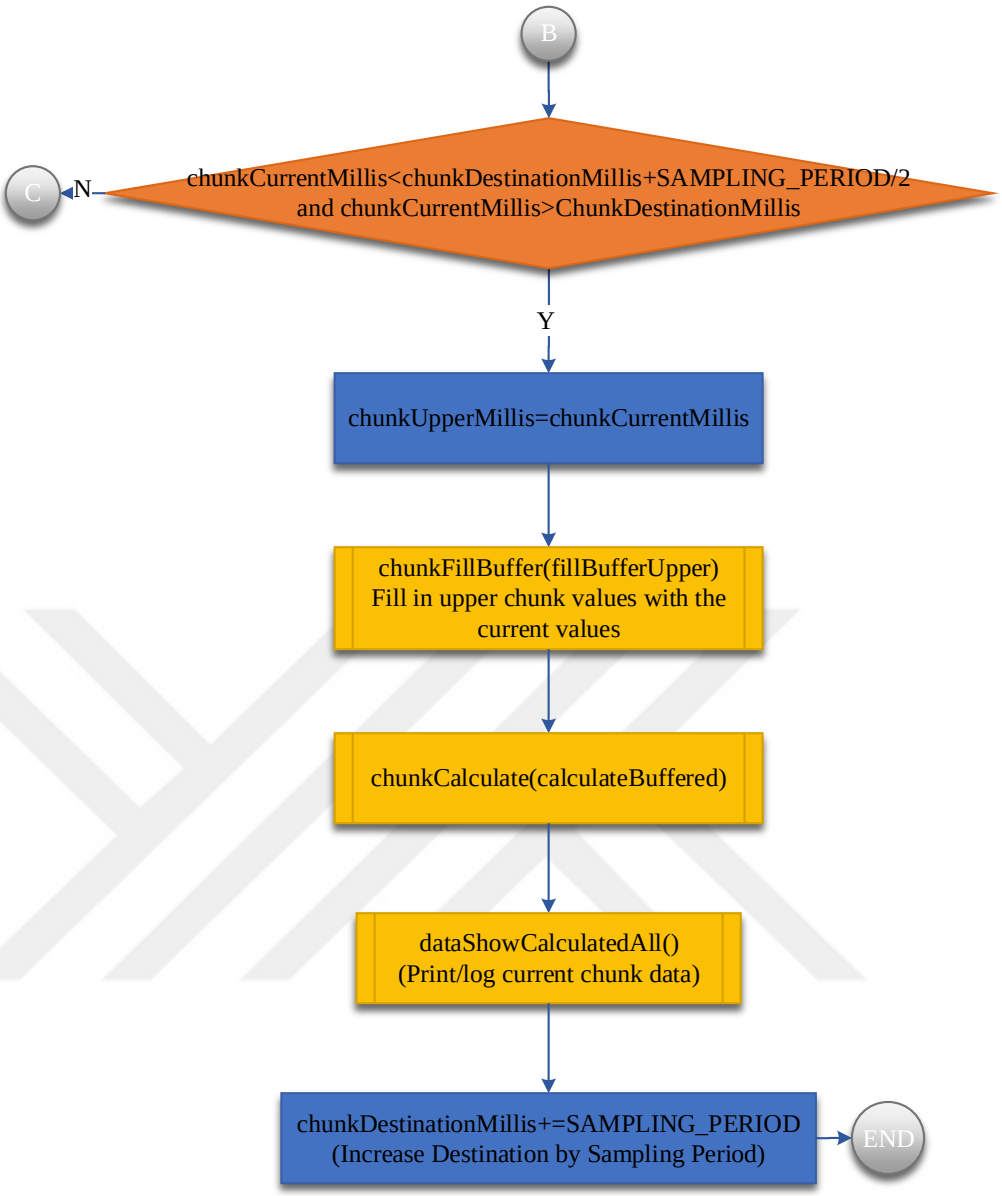

X
Assign the current hardware millis into millisDataFetchTic
Y
dataFetchSensors()
Assign the current hardware millis into millisDataFetchToc
Assign the current hardware millis into millisDataControlTic
millisDataFetchTicToc = millisDataFetchToc - millisDataFetchTic
dataShowCurrentAll() (Print/log current chunk data)
chunkSampleUniform()
Update LED Animation
Z

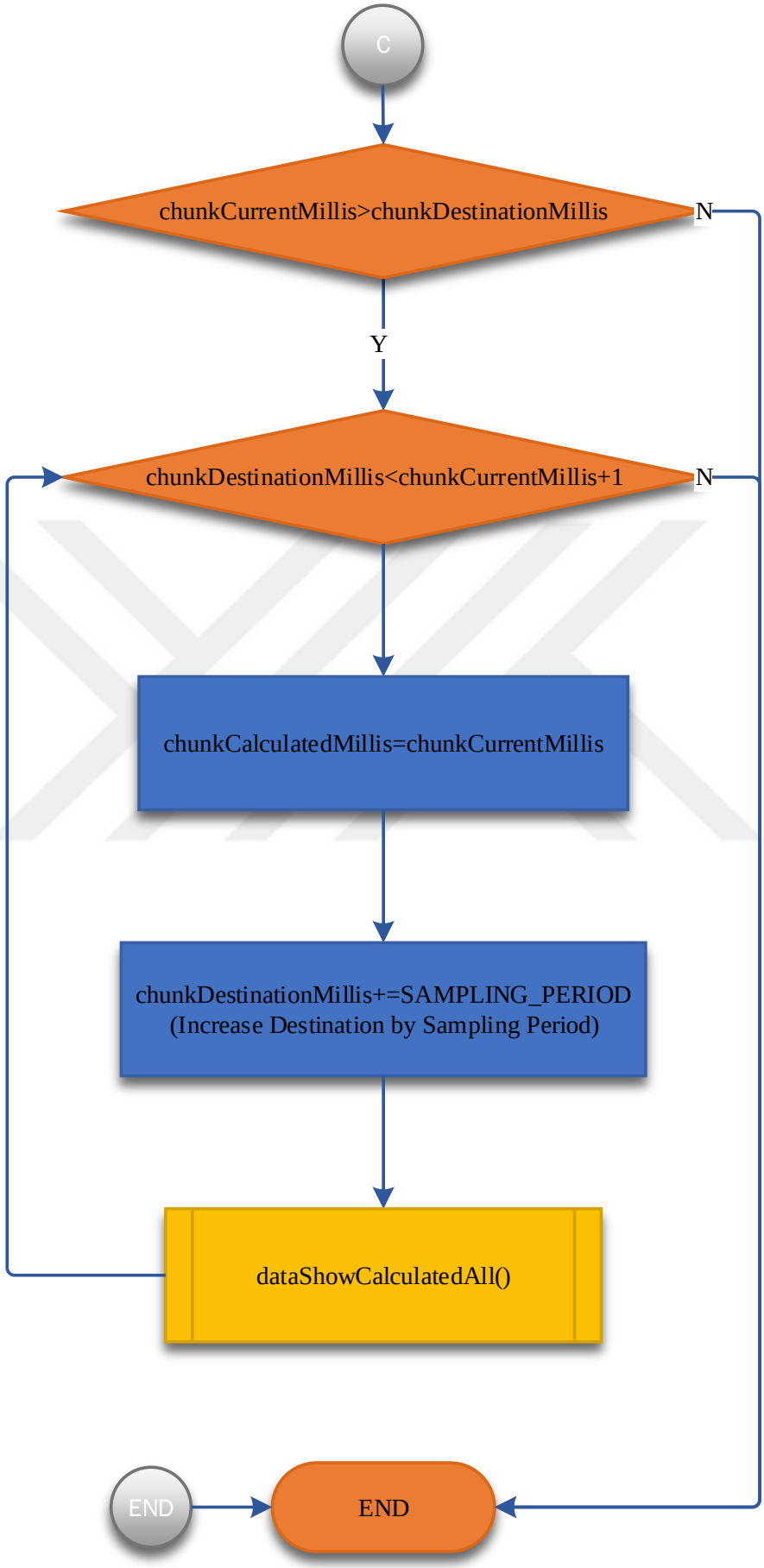

Z
Is Right Button Pressed?
Increase Phrase ID Counter and flush Gesture Repeat counter
N
Is Left Button Pressed?
Increase Phrase Repeat Counter
N
Assign the current hardware millis into millisDataControlToc
millisControlTicToc=millisControlToc-millisControlTic
chunkCurrentMillis=chunkCurrentMillis+millisDataFetchTicToc+millisControlTicToc
Y

FLOWCHART OF chunkSampleUniform()




A
chunkCurrentMillis>chunkDestinationMillis-SAMPLING_PERIOD/2 and chunkCurrentMillis<ChunkDestinationMillis
B
N
Y
chunkLowerMillis=chunkCurrentMillis
chunkFillBuffer(fillBufferLower)
Fill in lower chunk values with the current values
END


B
chunkCurrentMillis<chunkDestinationMillis+SAMPLING_PERIOD/2
and chunkCurrentMillis>ChunkDestinationMillis
C
N
Y
chunkUpperMillis=chunkCurrentMillis
chunkFillBuffer(fillBufferUpper)
Fill in upper chunk values with the current values
chunkCalculate(calculateBuffered)
dataShowCalculatedAll()
(Print/log current chunk data)
chunkDestinationMillis+=SAMPLING_PERIOD
(Increase Destination by Sampling Period)
END


C
chunkCurrentMillis>chunkDestinationMillis
N
Y
chunkDestinationMillis<chunkCurrentMillis+1
N
chunkCalculatedMillis=chunkCurrentMillis
chunkDestinationMillis+=SAMPLING_PERIOD
(Increase Destination by Sampling Period)
dataShowCalculatedAll()
END
END

FLOWCHART OF chunkCalculate()