ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO di
INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE
"Guglielmo Marconi"
DEI

CORSO DI LAUREA MAGISTRALE IN

0934 – INGEGNERIA ELETTRONICA: Electronics for Intelligent Systems, IoT and Big-Data

TESI DI LAUREA
in
Lab of Electronic Circuit Design

SETUP AND APPLICATION OF A DECISION TREE MACHINE LEARNING IN A REAL HUMAN MOTION TRACKING SCENARIO

CANDIDATO

Rohan Khan

RELATORE

Chiar.mo Prof. Luca De Marchi

CORRELATORE/I

Pietro Garofalo

Anno Accademico
2023/2024

Sessione
I

I would like to dedicate this thesis to my project supervisor, honorable faculty, my respectable parents, friends and loved ones who have been the constant source of inspiration for me. Without their continuous support and counselling, I would not have been able to complete the project successfully.

ACKNOWLEDGEMENTS

First of all, a huge thanks to both of my supervisors (Pietro Garofalo and Professor Luca De Marchi) for their continuous guidance and support throughout this internship and thesis process. I was lucky to have you both as my mentor and counselor that allowed me to demonstrate my work in the best possible manner.

Thank you to my parents Shehnaz and Khurram, and my sister Hijab for all the support that you provided. Special thanks to my two cute little nephews Yahya and Abdur Rehman for just existing.

Thanks to Ahsan, Muneeb and Ubaid for being here with me and for your emotional support not only during my whole degree, but also way back before it all started. Thanks to Maryam and Eesha as well. Thanks to the friends that I made in Norway during my exchange. Especially Mudasir and Faisal.

Thanks to all my other friends for just being there with me.

1 Table of Contents

2 List of Tables

3 List of Figures

4 Abstract

Human motion analysis is becoming an increasingly used technology nowadays with the invention of smarter sensor technologies and with its applications in the field of health, ergonomics, sports, medicine, and so on. Turingsense EU Lab is one of the many companies around the globe working on this technology. The purpose of this thesis was to evaluate the SensorTile.Box PRO kit (one of the smart IoT devices which is available in the market with its possible applications in motion analysis) as a possible replacement of the standard Accelerometer/Gyroscope sensors that they use. The purpose of this possible replacement is the benefits that this box kit offers, allowing us to implement an A.I. based machine learning algorithm that could save both time and effort in the prediction of correct arm movements.

The approach to investigate this evaluation was based on evaluating the performance of the box kit on predicting certain arm movements. This approach was based on first data collection, then training the decision-tree based ML algorithm by providing that data with their labels, and then finally testing the resulting configuration. By going through this process and modifying certain things along the way based on certain outcomes, the result was that the box kit was able to distinguish between the two different types of arm movements when it was attached to the wrist, and the accelerometer and the gyroscope data was used as the training data.

Conclusions that were drawn from the series of these experiments were that the SensorTile.Box PRO kit is no doubt a very promising device with a lot of potential for applications in the field of movement analysis. However, particularly in this application for Turingsense EU Lab, its potentials are kind of limited as we do not have complete freedom in terms of what kind of training data we want to use, or what kind of ML algorithm we want to use. If these and other related issues can be resolved, then this box kit can come very handy to Turingsense EU Lab.

5 Introduction

In the field of medicine, sports, and physiotherapy, human movement analysis, also known as motion analysis, has become an investigative and diagnostic tool. Movement analysis involves studying the movement patterns of humans, animals, and/or machines to understand the mechanics and dynamics involved. This field has applications in various domains including sports, medicine, ergonomics, animation, robotics, and so on. [1] [2] [3] [4] [5]

Along these lines, lies the motivation of Turingsense EU LAB. Turingsense EU LAB is a US-based startup, whose mission is to transform the way movement analysis is performed in sports and rehabilitation using wearable technologies to adapt to non-expert end-user scenarios. To achieve this mission, Turingsense EU LAB is guided by an innovative work plan for human motion capture that starts from the creation of easy-to-use hardware up to a customized application for the end user. Turingsense EU LAB works in close collaboration with its customers, from defining the specifications for the final application to designing a customized solution that includes specific biomechanical protocols, sensor fusion algorithms and the creation of software development platforms. [1]

5.1 Key Aspects of Movement Analysis

Movement analysis has some key concepts that are necessary to understand to be able to implement it successfully. These include data collection, data processing, analysis techniques, and its applications.

5.1.1 Data Collection

Collecting data is the first important step in motion analysis. If the data collection is not carried out correctly, movement analysis will never succeed. There are multiple ways to collect the data for training the algorithms for movement analysis.

- Optical Cameras: uses sensors or cameras to track movement. It works on technologies such as optical systems with markers, inertial sensors, and depth cameras. [1] [4] [5]

[1] https://www.turingsense.eu/

- Video Analysis: records movement with standard or high-speed cameras and then analyzes the video. [1] [6] [7] [8]
- Wearable devices: use accelerometers, gyroscopes, and magnetometers to gather data on movement. [3] [4] [7] [9] [10] [11] [12] [13] [14] [15] [16]

5.1.2 Data Processing

After data has been collected, it is also important to process it before using it for movement analysis.

- Kinematics: Studying the geometry of motion without considering the forces. This includes analyzing position, velocity, and acceleration. [1] [3] [4] [6] [10]
- Kinetics: Analyzing the forces and torques that cause motion, often using force plates or instrumented treadmills. [1] [3] [4] [6] [10] [17]
- Biometrics: Measuring biological data such as heart rate, muscle activity (EMG), and oxygen consumption to correlate with movement. [1] [6]

5.1.3 Analysis Techniques

Analysis techniques include:

- Qualitative Analysis: Visual assessment by experts to identify movement patterns and potential issues. [5] [7] [17]
- Quantitative Analysis: Using mathematical and computational methods to objectively measure and analyze movement parameters. [1] [3] [6] [7]
- Machine Learning: Applying algorithms to recognize patterns, classify movements, and predict outcomes based on the data. [9]

5.1.4 Applications

Some key applications of motion analysis include:

- Sports: In sports, it can be used to improve players' performance and to prevent injuries by analyzing the techniques of the players and their biomechanics. For injured players, it can also be used to guide rehabilitation programs. [1] [5] [7] [9] [12] [13] [19]

- Medicine: In medicine, it can be used to diagnose and treat movement disorders, to plan surgeries, and for patients' rehabilitation with conditions like stroke or Parkinson's disease. [1] [5] [9] [12] [18]
- Ergonomics: Designing workplaces and tools to reduce the risk of injury and for increasing efficiency. [1] [2] [5] [9] [18]
- Animation and Gaming: Creating realistic animations for films and video games by capturing and replicating exact human movements. [5]
- Robotics: Developing robots that can move and interact with their environment in a human-like manner, and for industrial robots to improve efficiency and precision. [4] [6]
- Research and Development: It is fundamental in studying the mechanics of biological systems, offering new insights and advancements in the field of kinesiology, orthopedics, and physical therapy. It also drives the development of new technologies such as advanced prosthetics, wearable sensors, and exoskeletons. [5]

Movement analysis provides a scientific basis for understanding and optimizing human and machine movement. Its applications are diverse and impactful, ranging from improving athletic performance and enhancing patient care to driving innovations in technology and design. By leveraging advanced techniques and tools, movement analysis continues to play a pivotal role in advancing knowledge and improving outcomes across multiple disciplines.

5.2 Example Workflow of Movement Analysis

An example workflow of movement analysis is written below:

1. Capture Movement: Use a motion capture system to record an athlete performing a specific activity.
2. Process Data: Extract kinematic data such as joint angles, velocities and accelerations.
3. Analyze Movement: Compare the data against normative values or the athlete's past performances to identify inefficiencies or potential injury risks.
4. Provide Feedback: Offer recommendations to improve technique, such as adjusting posture or altering training routines.
5. Monitor Progress: Regularly repeat the analysis to track improvements and make ongoing adjustments.

5.3 Challenges and Considerations

Movement analysis offers some challenges too and has some consideration that are listed below:

- Accuracy and Precision: Ensuring that the data collected is accurate and that the analysis is reliable. [1] [2] [3] [5] [17]
- Complexity: Human movement is highly complex and varies significantly between individuals, making standardization difficult. [1] [6] [19]
- Huge amount of Data: Extending on the previous point, to standardize it, a huge amount of data is required, which is diverse, covering all possibilities. [9]
- Integration: Combining data from various sources and sensors can be challenging but is often necessary for a comprehensive analysis. [1] [2] [10]
- Ethical Considerations: Ensuring the privacy and consent of individuals being analyzed, especially in medical and sports settings. [1] [9]

5.4 Movement Analysis at Turingsense EU LAB

Turing Motion represents the new working paradigm in the field of human movement analysis and has been successfully applied to products for the mass market, allowing anyone to track their movements and verify their pose in any place. The new paradigm includes ingredients that, combined together, enable new applications in the research sector. The Turingsense platform is constantly expanding and customizing based on different applications. [2]

5.4.1 Turingsense Technology

Thanks to more than 15 years of experience in the sector, Turingsense has developed proprietary algorithms for capturing human movement, completely based on inertial sensors, through solutions that do not use magnetometers, but which still allow continuous and prolonged recordings. The technology is called **MAG-FREE** TECHNOLOGY. [3]

[2] https://www.turingsense.eu/turing-motion
[3] https://www.turingsense.eu/

From a practical point of view, this means being able to capture human movement with the following advantages:

1. No need to calibrate magnetic sensors at each recording (magnetic sensors are not used)
2. No limitations in the type of environment within which the motion capture is performed:
    - Laboratory
    - Home
    - Physiotherapy room
3. No limitation in the application scenario:
    - Gait analysis even in the presence of electrical structures nearby
    - I walk on platforms of strength
    - Prosthetic devices or orthopedic aids with metal supports
    - Ferromagnetic objects present in the surrounding environment.

5.4.2 Smart Clothing

The smart clothing at Turingsense is based on MEMS (Micro Electromechanical Systems) technology to meet requirements both in terms of accuracy in motion capture and in terms of end-user applications. Some important properties of these wearable smart clothing are: 3

- Fully washable
- Customizable fashion design
- Different sizes
- Wireless technology
- Rechargeable via USB
- Scientifically proven sensor positioning using ISEO and OUTWALK measurement protocols
- 16 inertial units (3D Accelerometers, 3D Gyroscopes)
- Customizable configuration (Upper Limb and Lower Limb)
- No magnetometer used

5.4.3 Calibrations

Turingsense offers motion capture by using the above-mentioned smart clothing that has sensors embedded within it. These sensors include 3D Accelerometers and 3D Gyroscopes. Before using the sensors directly for motion capture, it is important to calibrate the sensors with pre-defined body movements. One of these movements includes the movement of the arm along the sagittal plane, in which the arm is raised completely up and then back down. In this movement, the sensors are positioned on the biceps. To achieve this, initially, the accelerometer and the gyroscope data are collected corresponding to this arm movement. And then an algorithm is trained on this data that should be able to detect this movement. Technically, the algorithm should be able to detect the movement no matter the placement of the sensor along the biceps (be it front, back, or sides). And it should discard any other arm movement and characterize it as an incorrect movement.

Currently, the user calibrations and Motion A.I. developed at firmware level at Turingsense products are made with algorithms that are calibrated on some data taken from different people. The algorithms are refined, by providing additional data, to distinguish between a person standing straight and a person not standing straight. If the person is standing straight, then the algorithm tells if the calibration passes or not. These algorithms have the following assumptions and limitations:

- It is assumed that the sensor is placed on a certain area on the body in a certain position.
- It is currently impossible to distinguish the not straight condition if it is because of the person not standing straight or because of the sensor not placed straight.
- It is therefore assumed that the sensors are placed upright correctly.
- It is also impossible to distinguish between the straight state among different people who have different backs. If a person, for example, has a curved back but standing straight, the algorithm fails and considers it as if the person is not standing straight.
- When the calibration fails, the algorithm cannot tell the reason why it failed. It could be because the user is not making the right movement, or the user is wearing the sensor wrongly, or the sensor is moving too much. But the algorithm cannot specify which of these situations. Its feedback is only to LOOK AT THE INSTRUCTIONS AGAIN and TRY AGAIN.

5.4.4 A.I. based on Machine Learning

By using AI based on machine learning, the first task would be to recognize the user's correct movement. Additionally, if the user's movement is wrong, then it could suggest to the user something more than simply infinitely repeating a movement until he succeeds, but rather possible adjustments to make. With further intelligence, it is also possible to understand where the sensor is on the body, then give suggestions to the user accordingly, for example, if the sensor placement is wrong.

5.4.5 SensorTile.Box PRO

Based on previous statements, it was suggested to use STMicroelectronics' SensorTile.Box PRO for the verification of correct user's movement that is necessary for calibration. The STEVAL-MKBOXPRO (SensorTile.box PRO) is the new ready-to-use programmable wireless box kit for developing any IoT application based on remote data gathering and evaluation.[4] But before making any decisions, it was necessary to evaluate this box kit and to see if it fits in this application.

To evaluate the box kit, it is necessary to see how it performs in predicting the correct arm movement. The first step int this evaluation would be the data collection for training the ML algorithm. The data was collected corresponding to the calibration movements to provide input for the training of machine learning algorithms. This data could either be collected using SensorTile.Box PRO or the standalone accelerometer and gyroscope sensors. The second step would then be to use the collected data for the training of the decision tree - based ML algorithm using the tools provided by ST, such as, UNICO GUI. Then the third and final step would be to test the resulting algorithm to see if it can predict both the correct and incorrect arm movements. After the algorithm has been trained correctly, we can also use the LED and BUZZER functions that are available on the SensorTile.Box PRO kit to give feedback to the user, for example, on correct or incorrect movement during calibration.

[4] https://www.st.com/en/evaluation-tools/steval-mkboxpro.html

6 Methodology

6.1 Exploring Hardware and Software Tools

When moving towards implementation of detecting user's correct arm movement on SensorTile.Box PRO, it was important to get familiar with necessary hardware and software tools, mostly from STMicroelectronics.

6.1.1 STEVAL-MKBOXPRO

STEVAL-MKBOXPRO (SensorTile.Box PRO) is a new ready-to-use programmable wireless box kit with multi-sensors and wireless connectivity for any intelligent IoT node. The box kit can be used according to three different modalities, based on the user's level of expertise. [5]

- **Entry Mode:** In entry mode, a wide range of already embedded IoT applications can be run on the box. STBLESensor App can be downloaded on the smartphone and board can be programmed with any of the applications that have been specifically designed to work with the board sensors. Some practical applications include Compass, Free-fall detection, Pedometer, Barometer, Data Recorder, Human activity recognition, and so on.
- **Expert Mode:** In expert mode, custom applications can be built through the STBLESensor App by selecting specific input data, functions/algorithms to be performed on that data, and then deciding how to display the data.
- **Pro Mode:** In pro mode, the user can make their own IoT by taking advantage of STM32 Open Development Environment (ODE) and ST function pack libraries, including sensing AI function pack with neural network libraries, without the need to perform any coding activity.

Detailed important features of the box kit are:

- Ultra-low-power with FPU Arm-Cortex-M33 with TrustZone® microcontroller (STM32U585AI)
- microSD™ card slot for standalone data logging applications

[5] https://www.st.com/en/evaluation-tools/steval-mkboxpro.html

- High precision sensors to gather high-quality data:
    - Low-voltage local digital temperature sensor (STTS22H)
    - Six-axis inertial measurement unit (LSM6DSV16X)
    - Three-axis low-power accelerometer (LIS2DU12)
    - 3-axis magnetometer (LIS2MDL)
    - Pressure sensor (LPS22DF)
    - Digital microphone/audio sensor (MP23DB01HP)
- User Interface:
    - Hardware power switch
    - Green and orange system LED to display the power supply state
    - 4 programmable status LEDs (green, red, orange, blue)
    - 2 programmable push-buttons
    - Audio buzzer
    - Reset button
    - Qvar with electrodes for user interface experience
    - Interface J-Link/SWD debug-probe
    - Interface for extension board
    - Socket for DIL24 sensor adapters
- Power and charging options: USB Type-C® charging and connecting, 5 W wireless charging and rechargeable long-life 480 mAh battery.
- STBLESensor App on the smartphone (both on the Google Play and Apple Store) allows you to immediately connect to the box kit.
- Firmware over-the-air (FOTA) upgrade

In the context of this internship, this box kit was usually used in the Entry and the Expert modes. In the entry mode, it was used for data collection. In the expert mode, it was used to create custom applications that run on UNICO generated ML configurations.

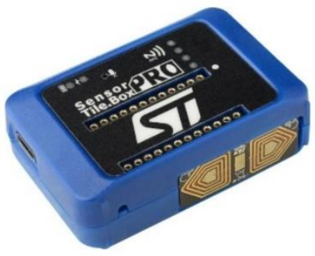


Figure 1: SensorTile.Box PRO kit

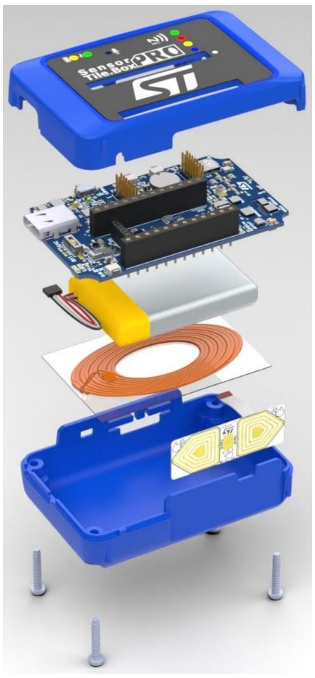


Figure 2: Inside view of STEVAL-MKBOXPRO

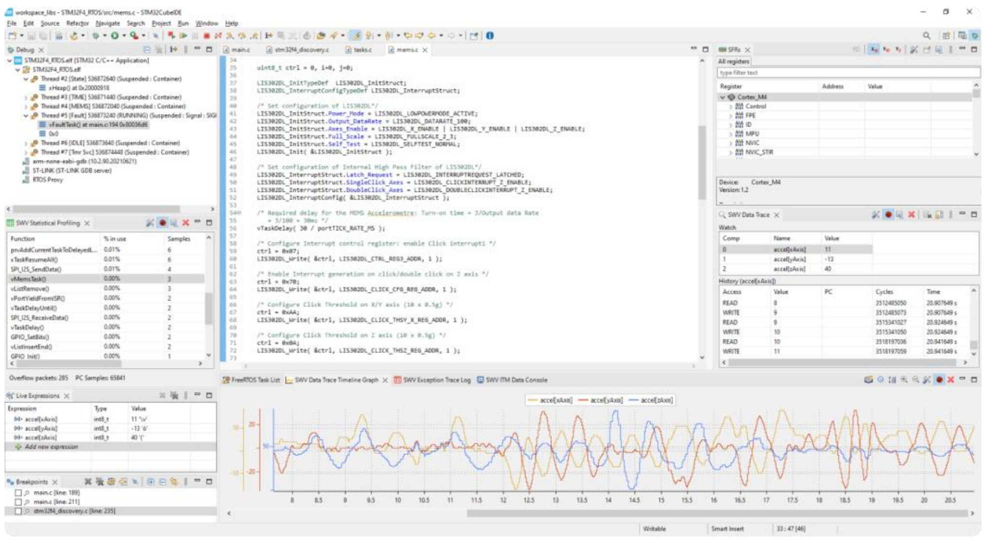
6.1.2 STM32CubeIDE

STM32CubeIDE is an Integrated Development Environment (IDE) for STM32. STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors. [6]

In the context of this internship, STM32CubeIDE was used to work on the Activity Recognition Algorithm project. It was used to make some modifications to the already existing project. The

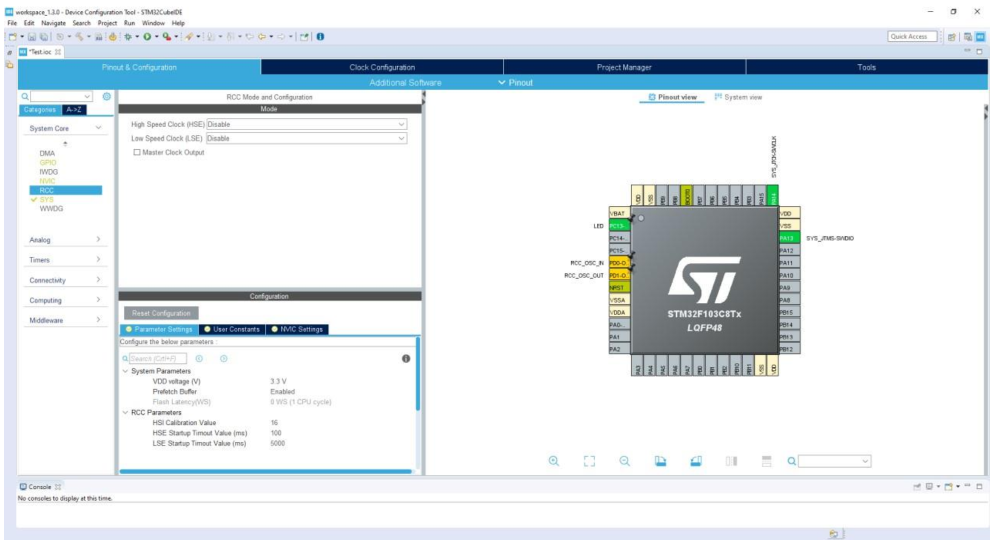
[6] https://www.st.com/en/development-tools/stm32cubeide.html

modifications included blinking of some LEDs and a buzzer sound when certain conditions are met.



```
uint8_t ctrl = 0, i=0, j=0;

LIS302DL_InitTypeDef  LIS302DL_InitStruct;
LIS302DL_InterruptConfigTypeDef LIS302DL_InterruptStruct;

/* Set configuration of LIS302DL*/
LIS302DL_InitStruct.Power_Mode = LIS302DL_LOWPOWERMODE_ACTIVE;
LIS302DL_InitStruct.Output_DataRate = LIS302DL_DATARATE_100;
LIS302DL_InitStruct.Axes_Enable = LIS302DL_X_ENABLE | LIS302DL_Y_ENABLE | LIS302DL_Z_ENABLE;
LIS302DL_InitStruct.Full_Scale = LIS302DL_FULLSCALE_2_3;
LIS302DL_InitStruct.Self_Test = LIS302DL_SELFTEST_NORMAL;
LIS302DL_Init( &LIS302DL_InitStruct );

/* Set configuration of Internal High Pass Filter of LIS302DL*/
LIS302DL_InterruptStruct.Latch_Request = LIS302DL_INTERRUPTREQUEST_LATCHED;
LIS302DL_InterruptStruct.SingleClick_Axes = LIS302DL_CLICKINTERRUPT_Z_ENABLE;
LIS302DL_InterruptStruct.DoubleClick_Axes = LIS302DL_DOUBLECLICKINTERRUPT_Z_ENABLE;
LIS302DL_InterruptConfig( &LIS302DL_InterruptStruct );

/* Required delay for the MEMS Accelerometre: Turn-on time = 3/Output data Rate
   = 3/100 = 30ms */
vTaskDelay( 30 / portTICK_RATE_MS );

/* Configure Interrupt control register: enable Click interrupt1 */
ctrl = 0x07;
LIS302DL_Write( &ctrl, LIS302DL_CTRL_REG3_ADDR, 1 );

/* Enable Interrupt generation on click/double click on Z axis */
ctrl = 0x70;
LIS302DL_Write( &ctrl, LIS302DL_CLICK_CFG_REG_ADDR, 1 );

/* Configure Click Threshold on X/Y axis (10 x 0.5g) */
ctrl = 0xAA;
LIS302DL_Write( &ctrl, LIS302DL_CLICK_THSY_X_REG_ADDR, 1 );

/* Configure Click Threshold on Z axis (10 x 0.5g) */
ctrl = 0x0A;
LIS302DL_Write( &ctrl, LIS302DL_CLICK_THSZ_REG_ADDR, 1 );
```

Figure 3: STM32CubeIDE Example Screenshot 1



High Speed Clock (HSE) Disable
Low Speed Clock (LSE) Disable
Master Clock Output

System Parameters
VDD voltage (V) 3.3 V
Prefetch Buffer Enabled
Flash Latency(WS) 0 WS (1 CPU cycle)
RCC Parameters
HSI Calibration Value 16
HSE Startup Timout Value (ms) 100
LSE Startup Timout Value (ms) 5000

No consoles to display at this time.

Figure 4: STM32CubeIDE Example Screenshot 2

6.1.3 STM32CubeProg

STM32CubeProgrammer (STM32CubeProg) is a software tool for programming STM32 products. It provides an easy-to-use and efficient environment for reading, writing and verifying device memory through both the debug interface (JTAG and SWD) and the bootloader interface (UART, USB DFU, I2C, SPI, and CAN). [7]

In the context of this internship, STM32CubeProg was used to upload the binaries on the box kit in DFU mode. These binaries include the basic firmware binary of SensorTile.Box PRO kit and other binaries obtained after successfully compiling a project on STM32CubeIDE.

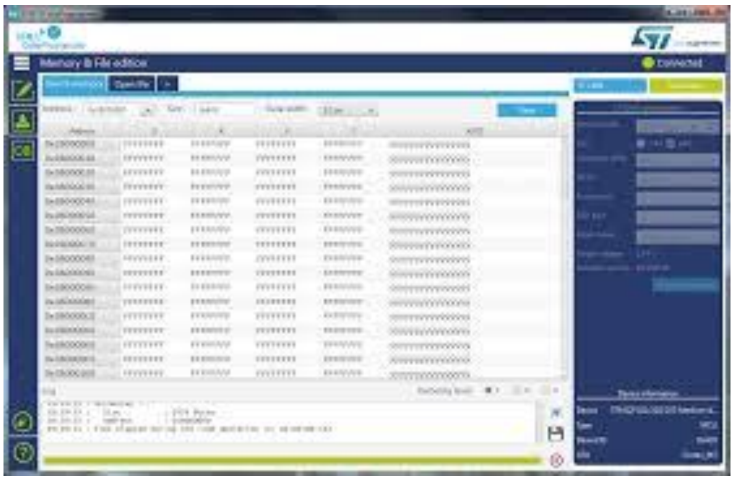


Figure 5: STM32CubeProg Example Screenshot

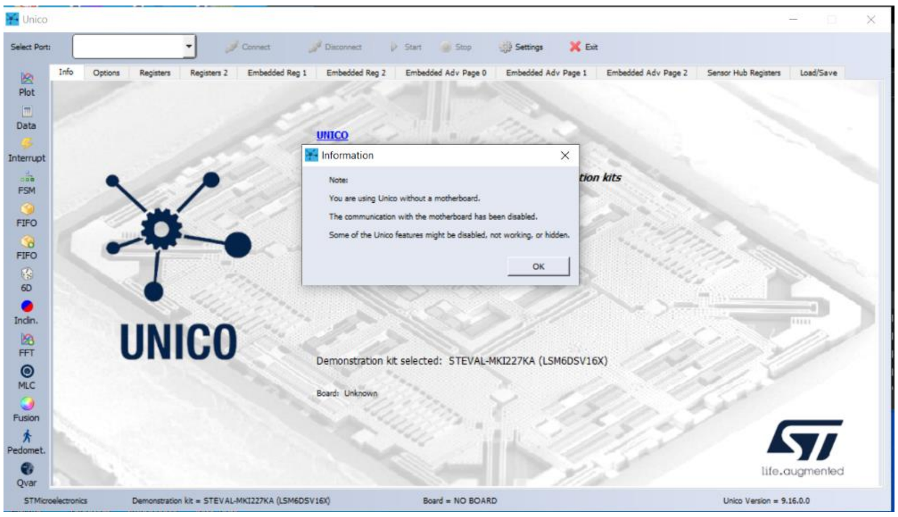
6.1.4 Unico-GUI

Unico-GUI is a MEMS evaluation kit software package. It is a cross-platform graphical user interface interacting with STEVAL-MKI109V3 (Professional MEMS tool) which is the motherboard compatible with all ST MEMS adapter boards. It is also possible to run UNICO offline (without the motherboard) for generating configurations of advanced features like the Machine Learning Core, Finite State Machine, and pedometer. [8]

Examples of tools which support the advanced features are the following: Machine Learning Core tool that allows the user to configure a machine learning core starting from the management of data patterns and labeling to setting and generating the configuration file to run the algorithm.

[7] https://www.st.com/en/development-tools/stm32cubeprog.html
[8] https://www.st.com/en/development-tools/unico-gui.html

In the context of this internship, this is the most important software. It was used to create all the machine learning configurations that were eventually programmed on the box kit. The aim of this internship is to create a machine learning core able to distinguish the correct arm movements. Well, this is the tool to create that machine learning configuration.



Note:

You are using Unico without a motherboard.

The communication with the motherboard has been disabled.

Some of the Unico features might be disabled, not working, or hidden.

Demonstration kit selected: STEVAL-MKI227KA (LSM6DSV16X)

Board: Unknown

STMicroelectronics   Demonstration kit = STEVAL-MKI227KA (LSM6DSV16X)   Board = NO BOARD   Unico Version = 9.16.0.0

Figure 6: Unico-GUI Example Screenshot

6.1.5 AlgoBuilder

AlgoBuilder is a graphical design application to build and use algorithms. The application facilitates the process of implementing proof of concept using a graphical interface without writing the code. [9]

In the context of this internship, this software was used only a couple of times to get familiar with it by creating some very basic firmware with very basic building blocks. Later on, it was not used for the purpose of this internship.

[9] https://www.st.com/en/development-tools/algobuilder.html

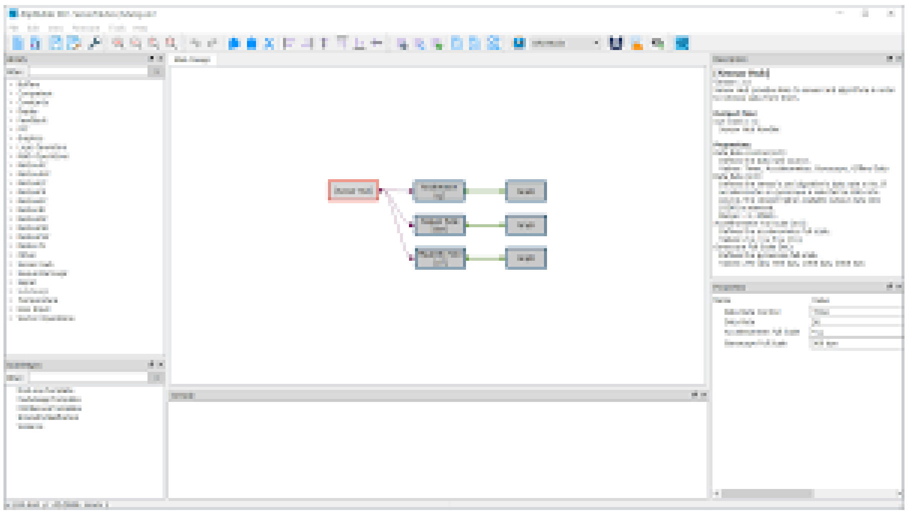


Figure 7: AlgoBuilder Example Screenshot

6.1.6 Unicleo-GUI

Unicleo-GUI is a graphical user interface (GUI) to demonstrate the functionality of ST sensors and algorithms. Unicleo-GUI is able to cooperate with firmware created by AlgoBuilder application and display data coming from the running firmware. [10]

As this tool is mostly used to read the live sensor values, it was not used at all as it required a base motherboard to be connected in between the laptop and the sensors board to be able to read those sensor values. As we did not have that base motherboard, there was no point in using this software at all.

Figure 8: Unicleo-GUI Example Screenshot

[10] https://www.st.com/en/development-tools/unicleo-gui.html

6.1.7 STBLESensor

STBLESensor mobile application is used for connecting with the box kit via Bluetooth. It can then read the sensors data being sent by the box kit. It is also used to program the board in Expert mode. All the data received by the app can be logged into CSV files and exported by e-mail. [11]

In the context of this internship, this mobile application was used to collect the training data using the SensorTile.Box PRO kit. It was also used to upload the machine learning configurations on the box kit and then used to test those configurations by creating new applications in the expert mode.

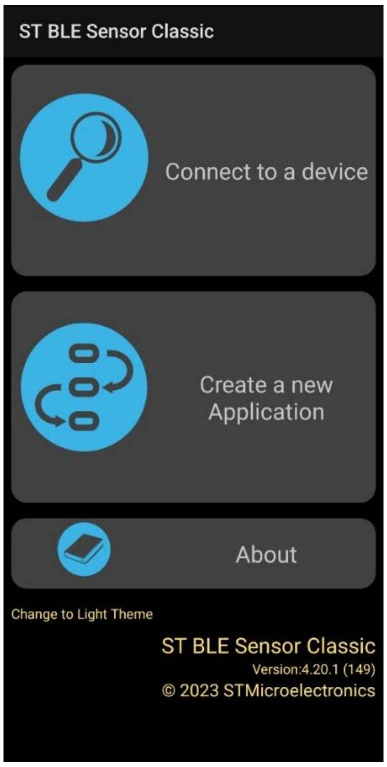


Figure 9: STBLESensor App Example Screenshot 1

[11] https://www.st.com/en/embedded-software/stblesensor.html

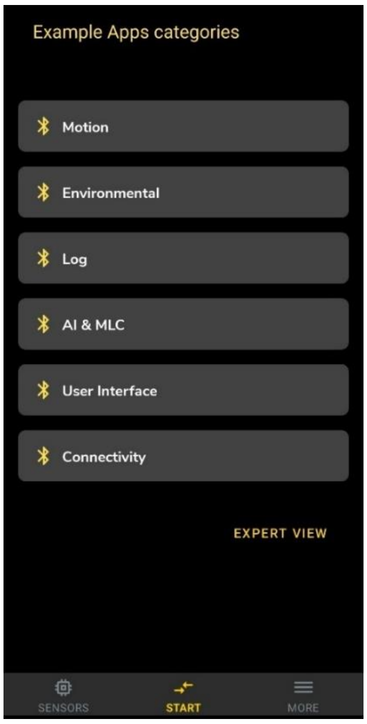


Figure 10: STBLESensor App Example Screenshot 2

6.1.8 LSM6DSV16X

LSM6DSV16X is a 6-axis inertial measurement unit (IMU) and AI sensor with embedded sensor fusion. It is a high-performance, low-power 6-axis small IMU, featuring a 3-axis digital accelerometer and a 3-axis digital gyroscope, that offers the best IMU sensor. [12]

The LSM6DSV16X enables processes in edge computing, leveraging embedded advanced dedicated features such as a finite state machine (FSM) and a machine learning core (MLC) for IoT applications.

This LSM6DSV16X IMU is embedded within the SensorTile.Box PRO kit. It is a very important hardware unit in the context of this internship as the MLC unit on it is responsible for implementing the algorithm that will be able to detect the correct arm movement.

[12] https://www.st.com/en/mems-and-sensors/lsm6dsv16x.html

6.1.9 Activity Recognition Algorithm

After getting familiar with the above-mentioned hardware and software components, it was time to implement something using those tools, in order to get familiar with them. The first thing that was done was implementing the Activity Recognition MLC example project. This example project is available as an STM32CubeIDE project in STM32 ODE function pack. The project is directly built on STM32CubeIDE resulting in a binary file. This binary file can then be programmed on the board (STEVAL-MKBOXPRO) using STM32CubeProgrammer in the DFU mode, in which case, the board is directly connected to the PC via USB/Type-C cable. Alternatively, the board can be programmed by using the STM32CubeIDE debugger, in which case, the board is connected to the PC via STLINK-V3SET (The STLINK-V3SET is a modular debugging and programming probe for the STM8 and STM32 Microcontrollers). This activity recognition program is intended to work with ST BLE Sensor app. The activities recognized in this example are: Stationary, Walking, Jogging, Biking and Driving. After the board is connected with the app using Bluetooth, it is displayed on the app which of the activities the MLC recognizes. This was the standard unmodified version of the Activity Recognition program.

6.2 Setting up Real-time Feedback

Real-time feedback would be able to inform the user about its arm movement by means of audio and/or visual aids. For example, the box kit can produce a specific buzzer sound or blink a specific LED if the user’s arm movements are correct and some other sound or some other LED if the movements are incorrect.

6.2.1 LED Blinking and Buzzer

After implementing the standard Activity Recognition program on the box kit, some modifications were made to the program by using the STM32CubeIDE. In the project, the code was edited such that the box kit will blink some LEDs and will also produce a buzzer sound if the MLC recognizes any activity. This modification was successful and was also tested after compiling the project and uploading it on the board.

6.3 Workflow Setup

After implementing the Activity Recognition Algorithm and then modifying it ourselves, it was time to create a general workflow setup for implementing our own “arm movement recognition” algorithm. The general workflow would first involve collecting the data, then analyzing it by plotting the data (preferably using python libraries), and then cleaning/filtering and processing it to use for training. This data would be then fed to STM’s Unico-GUI software which uses decision trees to create a Machine Learning Configuration (MLC) which can then be programmed on the box kit, and then later tested for the correct performance. This is a broad overview of the MLC pipeline implemented, whose detailed implementation is discussed in the sections to follow.

6.3.1 MLC Configuration

After the successful attempt with LEDs and buzzer, the focus was towards using the MLC to detect a particular arm movement which is necessary in calibrating the sensors for motion capture. In this case, the sensor is attached to the biceps on the arms.

The first step towards this task was the data collection. Theoretically, the more the data, the better the MLC would be trained. The data was collected for 9 different combinations of sensor placements and arm movements for each of the arms. Practically, this data must be collected for as many people as possible in order to generalize the algorithm, so that it will be able to predict the movements of the unknown people. The MLC configuration generation process uses the decision-tree algorithm to match the datasets with the assigned labels. Several MLC configurations were created in which different labels were assigned to different datasets, to test the performance of the decision-tree generation process.

This configuration is created by using Unico-GUI. The result of this UNICO step is a configuration file, which can then be used for programming the board in Expert mode. When programming the board, by selecting the Output ‘Stream to Bluetooth’, it is possible to observe the decision tree output on the STBLESensor app.

6.3.2 MLC Pipeline

The whole MLC generation process can be broken down in three major steps. These steps are described below.

6.3.2.1 *Data Collection using ST BLE:*

The data was collected using the SensorTile.Box PRO kit. There are two ways to collect data. The first method writes the data to the csv files on the SD-card mounted on the kit. The second method writes it in the internal memory of the mobile phone.

a) In the first method, the Data recorder example app was uploaded on the board using ST BLE Sensor Classic App as shown in the images below.

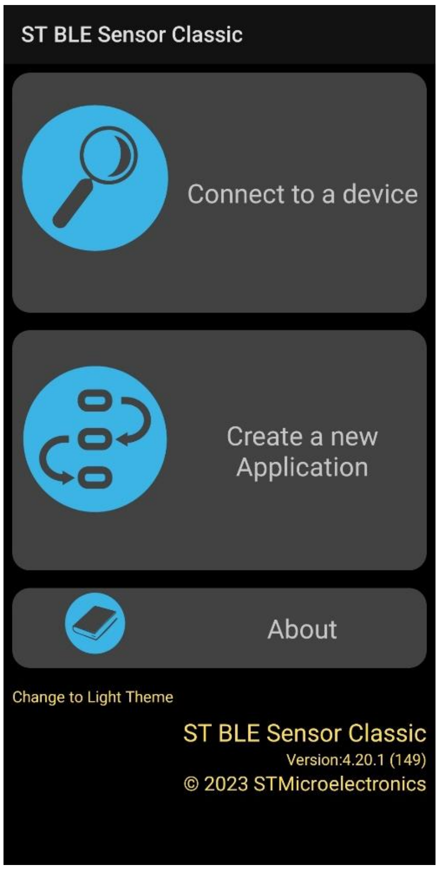


Figure 11: ST BLE Sensor App Main Screen

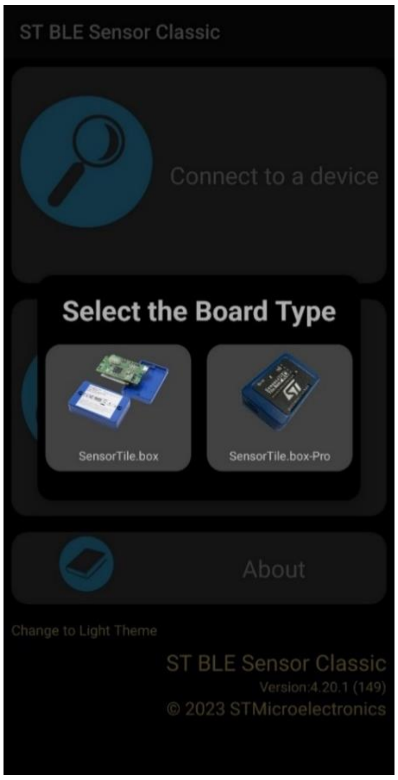


Figure 12: ST BLE Sensor App Screenshot 2

Select ‘Create a new Application’ -> Select the Board Type.

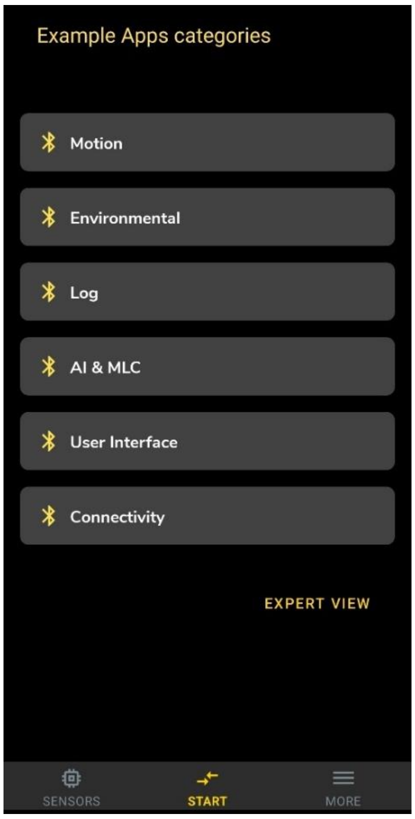


Figure 13: ST BLE Sensor App Screenshot 3

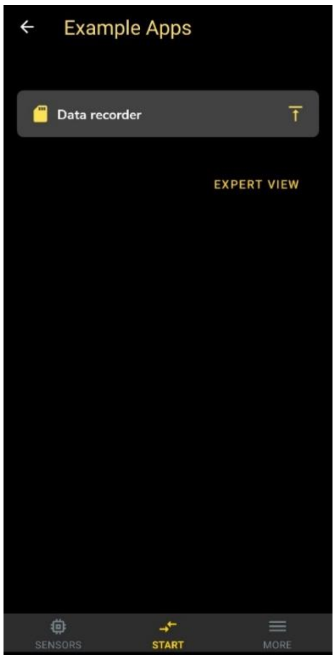
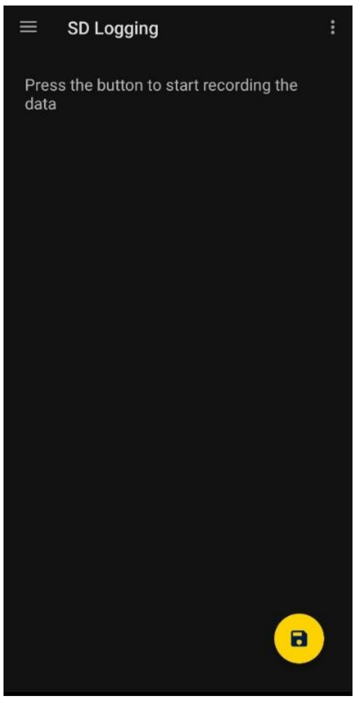


Figure 14: ST BLE Sensor App Screenshot 4

Select ‘Log’ -> Upload Data recorder app on the board.

After the app is loaded successfully, connect to the device. Then, data logging can be done as shown in the images below.



Press the button to start recording the data

Figure 15: ST BLE Sensor App Screenshot 5

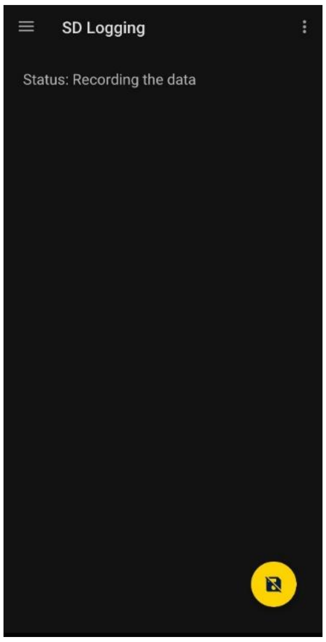


Figure 16: ST BLE Sensor App Screenshot 6

Select Start Logging and Stop Logging option at the bottom right of the screen to start and stop logging the data. The data is saved in a csv file, which is stored on the SD card mounted on the kit and can be retrieved from there.

b) In the second method, a new application was created using the ST BLE Sensor Classic app.

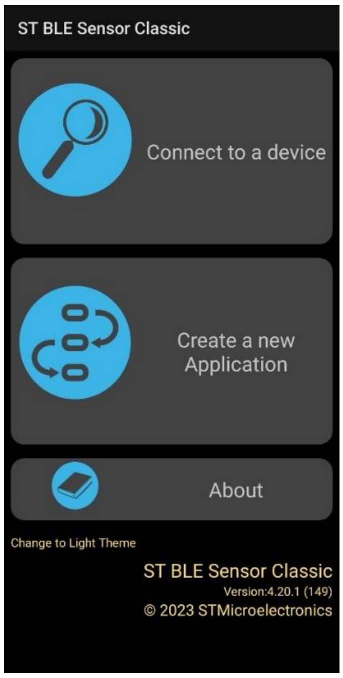


Figure 17: ST BLE Sensor App Screenshot 7

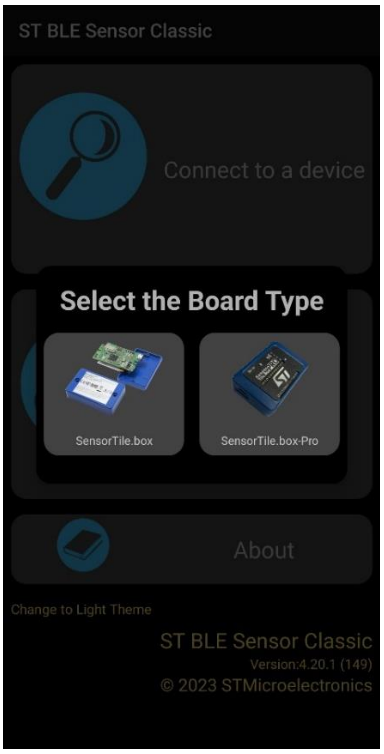


Figure 18: ST BLE Sensor App Screenshot 8

Select 'Create a new Application' -> Select the Board Type.

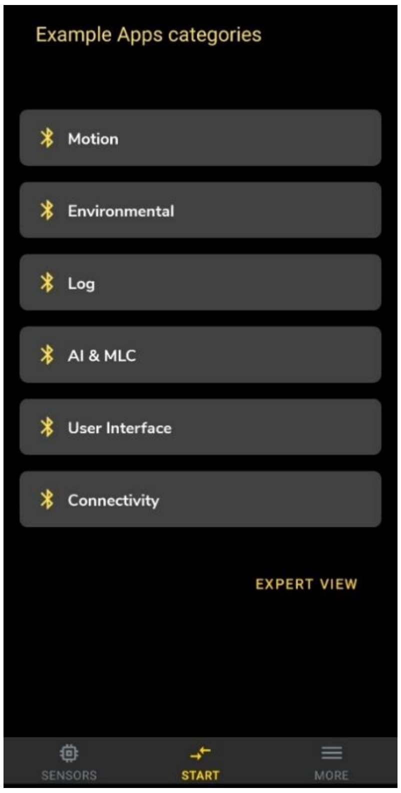


Figure 19: ST BLE Sensor App Screenshot 9

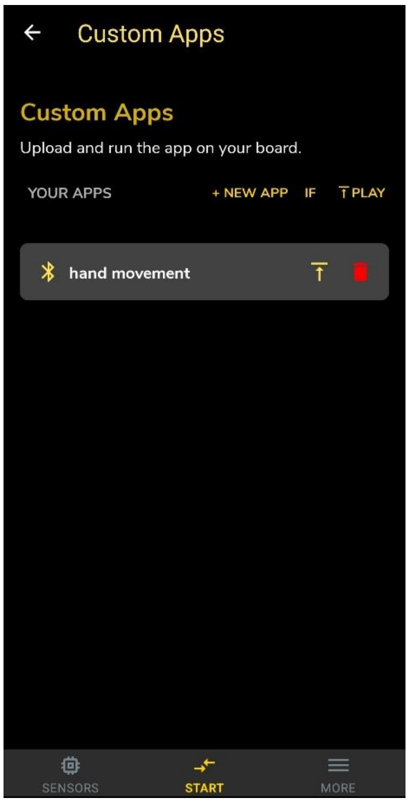
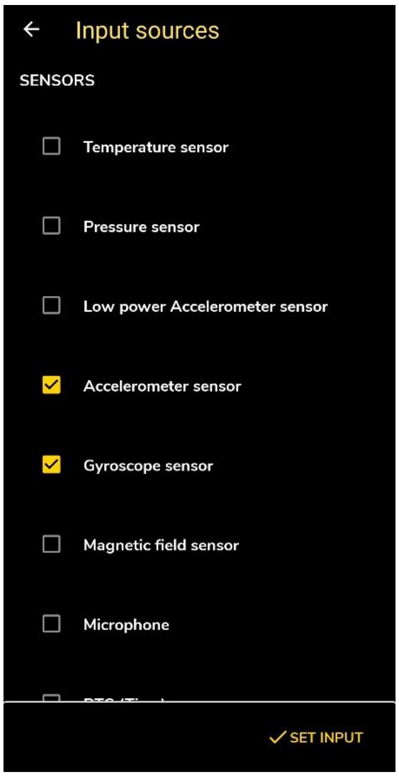


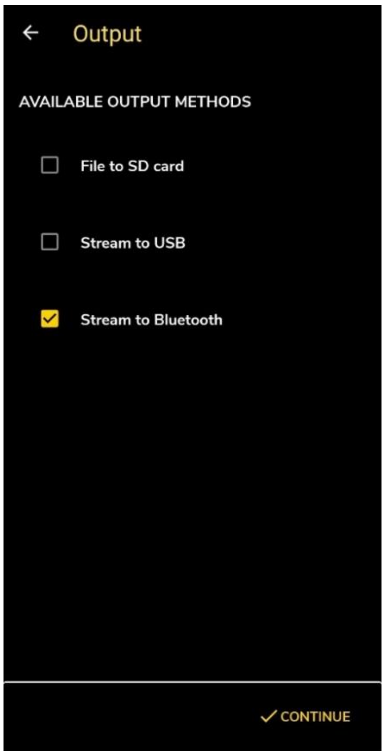
Figure 20: ST BLE Sensor App Screenshot 10

Press 'EXPERT VIEW' and then press '+ NEW APP'.



- [ ] Temperature sensor
- [ ] Pressure sensor
- [ ] Low power Accelerometer sensor
- [x] Accelerometer sensor
- [x] Gyroscope sensor
- [ ] Magnetic field sensor
- [ ] Microphone

Figure 21: ST BLE Sensor App Screenshot 11


Output

AVAILABLE OUTPUT METHODS

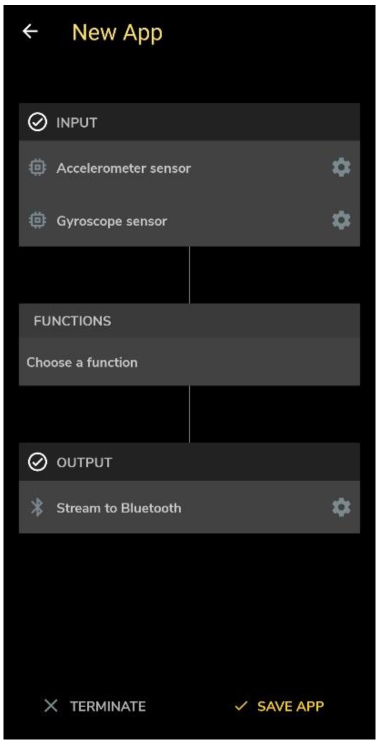
☐ File to SD card

☐ Stream to USB

☑ Stream to Bluetooth

Figure 22: ST BLE Sensor App Screenshot 12

Select Input sources and Output.

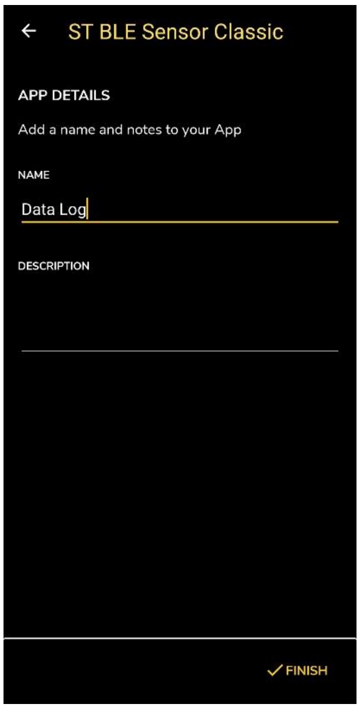

New App

INPUT
Accelerometer sensor
Gyroscope sensor

FUNCTIONS
Choose a function

OUTPUT
Stream to Bluetooth

Figure 23: ST BLE Sensor App Screenshot 13


ST BLE Sensor Classic

APP DETAILS

Add a name and notes to your App

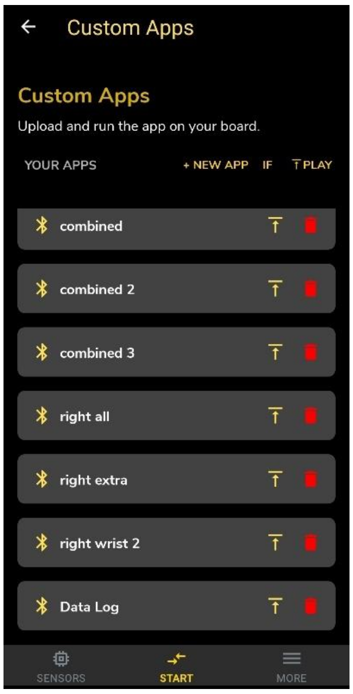
NAME

Data Log

DESCRIPTION

Figure 24: ST BLE Sensor App Screenshot 14

Save app. Give it a suitable name and an optional description. Press Finish.


Custom Apps

Custom Apps

Upload and run the app on your board.

combined

combined 2

combined 3

right all

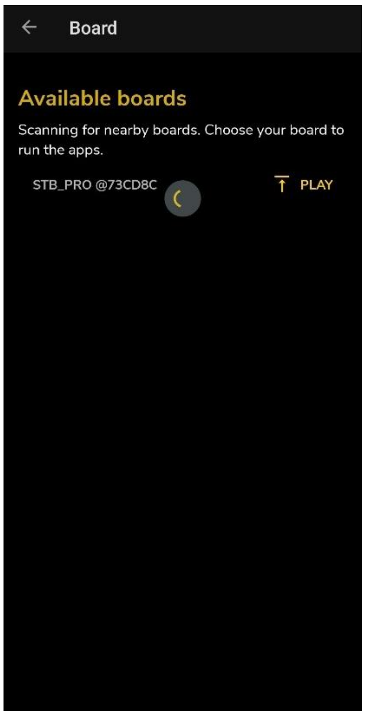
right extra

right wrist 2

Data Log

Figure 25: ST BLE Sensor App Screenshot 15

Upload app by pressing the upload symbol.

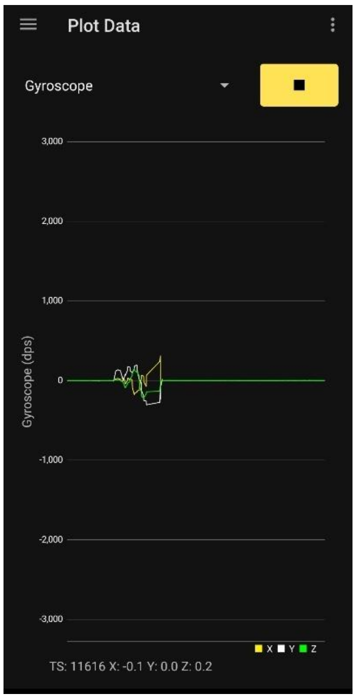


Scanning for nearby boards. Choose your board to run the apps.

STB_PRO @73CD8C

Figure 26: ST BLE Sensor App Screenshot 16

Select the board.



TS: 11616 X: -0.1 Y: 0.0 Z: 0.2

Figure 27: ST BLE Sensor App Screenshot 17

Once the app is uploaded, Gyroscope and Accelerometer data (which were selected as inputs initially) can also be observed on the app. Press the three vertical dots icon at the top right of the screen to start and stop logging.

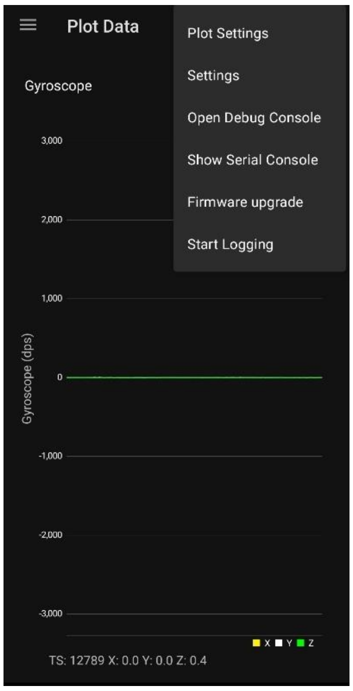


Figure 28: ST BLE Sensor App Screenshot 18

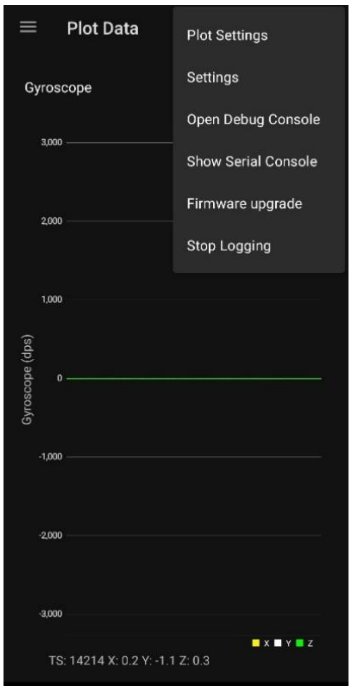
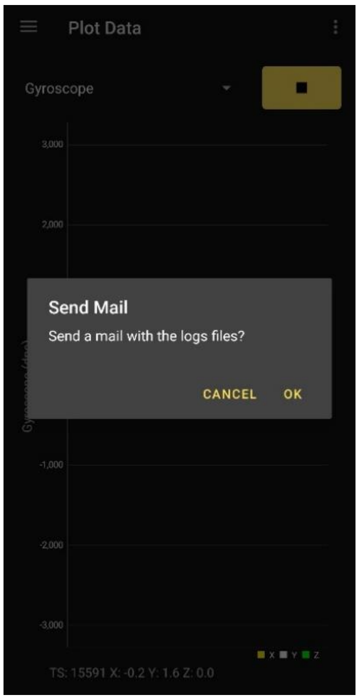


Figure 29: ST BLE Sensor App Screenshot 19

Start and Stop Logging accordingly.



Send Mail

Send a mail with the logs files?

TS: 15591 X: -0.2 Y: 1.6 Z: 0.0

Figure 30: ST BLE Sensor App Screenshot 20

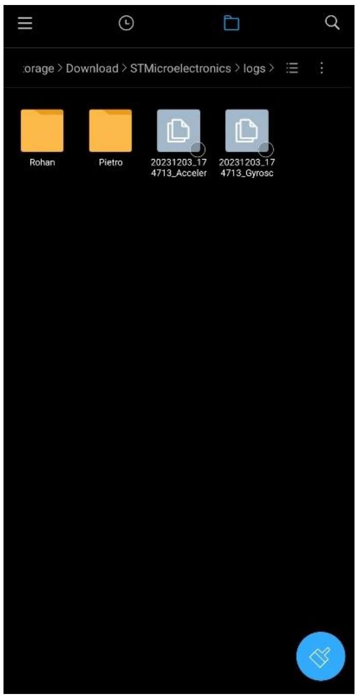
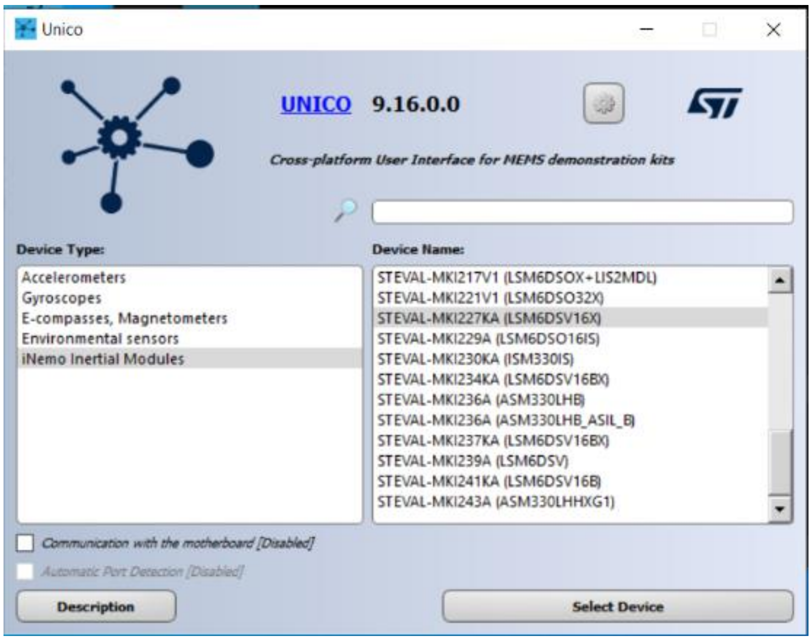


Figure 31: ST BLE Sensor App Screenshot 21

Select OK to save the files internally. Files can be found in the device's internal memory.

6.3.2.2 *Generating UCF file for MLC using UNICO-GUI:*



Device Type:

- Accelerometers
- Gyroscopes
- E-compasses, Magnetometers
- Environmental sensors
- iNemo Inertial Modules

Device Name:

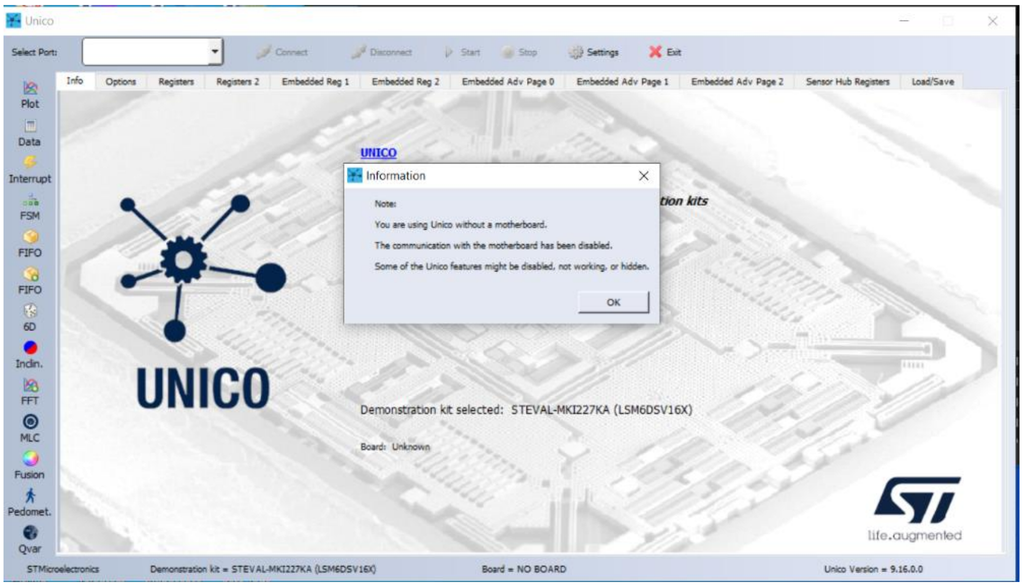
- STEVAL-MKI217V1 (LSM6DSOX+LIS2MDL)
- STEVAL-MKI221V1 (LSM6DSO32X)
- STEVAL-MKI227KA (LSM6DSV16X)
- STEVAL-MKI229A (LSM6DSO16IS)
- STEVAL-MKI230KA (ISM330IS)
- STEVAL-MKI234KA (LSM6DSV16BX)
- STEVAL-MKI236A (ASM330LHB)
- STEVAL-MKI236A (ASM330LHB_ASIL_B)
- STEVAL-MKI237KA (LSM6DSV16BX)
- STEVAL-MKI239A (LSM6DSV)
- STEVAL-MKI241KA (LSM6DSV16B)
- STEVAL-MKI243A (ASM330LHHXG1)

☐ Communication with the motherboard [Disabled]

☐ Automatic Port Detection [Disabled]

Figure 32: UNICO GUI Main Screen

Open UNICO-GUI. Select ‘iNemo Inertial Modules’ -> ‘STEVAL-MKI227KA (LSM6DSV16X)’. Unselect ‘Communication with the motherboard [Disabled]’ to use it offline. Click ‘Select Device’.



Information

Note:

You are using Unico without a motherboard.

The communication with the motherboard has been disabled.

Some of the Unico features might be disabled, not working, or hidden.

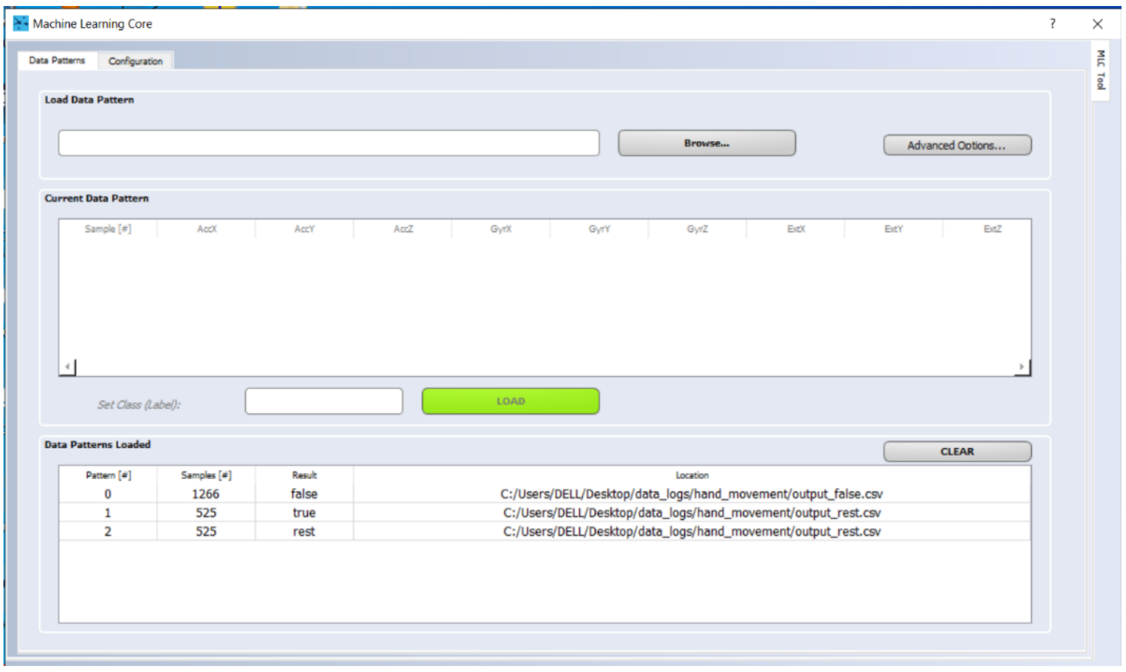
Demonstration kit selected: STEVAL-MKI227KA (LSM6DSV16X)

Board: Unknown

STMicroelectronics   Demonstration kit = STEVAL-MKI227KA (LSM6DSV16X)   Board = NO BOARD   Unico Version = 9.16.0.0

Figure 33: UNICO GUI Screenshot 2

Select MLC option from the left panel.



Machine Learning Core

Data Patterns | Configuration

Load Data Pattern

Browse... | Advanced Options...

Current Data Pattern

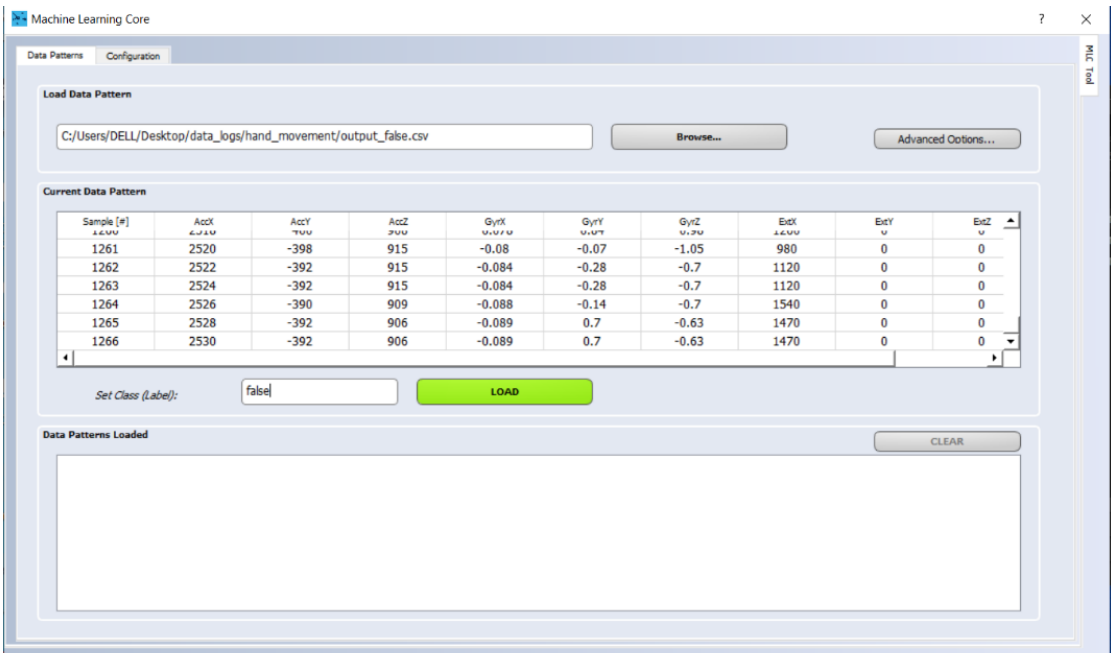
Sample [#]	AccX	AccY	AccZ	GyrX	GyrY	GyrZ	ExtX	ExtY	ExtZ

Set Class (Label): | LOAD

Data Patterns Loaded | CLEAR

Pattern [#]	Samples [#]	Result	Location
0	1266	false	C:/Users/DELL/Desktop/data_logs/hand_movement/output_false.csv
1	525	true	C:/Users/DELL/Desktop/data_logs/hand_movement/output_rest.csv
2	525	rest	C:/Users/DELL/Desktop/data_logs/hand_movement/output_rest.csv

Figure 34: UNICO GUI Screenshot 3



Machine Learning Core

Data Patterns | Configuration

Load Data Pattern

C:/Users/DELL/Desktop/data_logs/hand_movement/output_false.csv

Browse... | Advanced Options...

Current Data Pattern

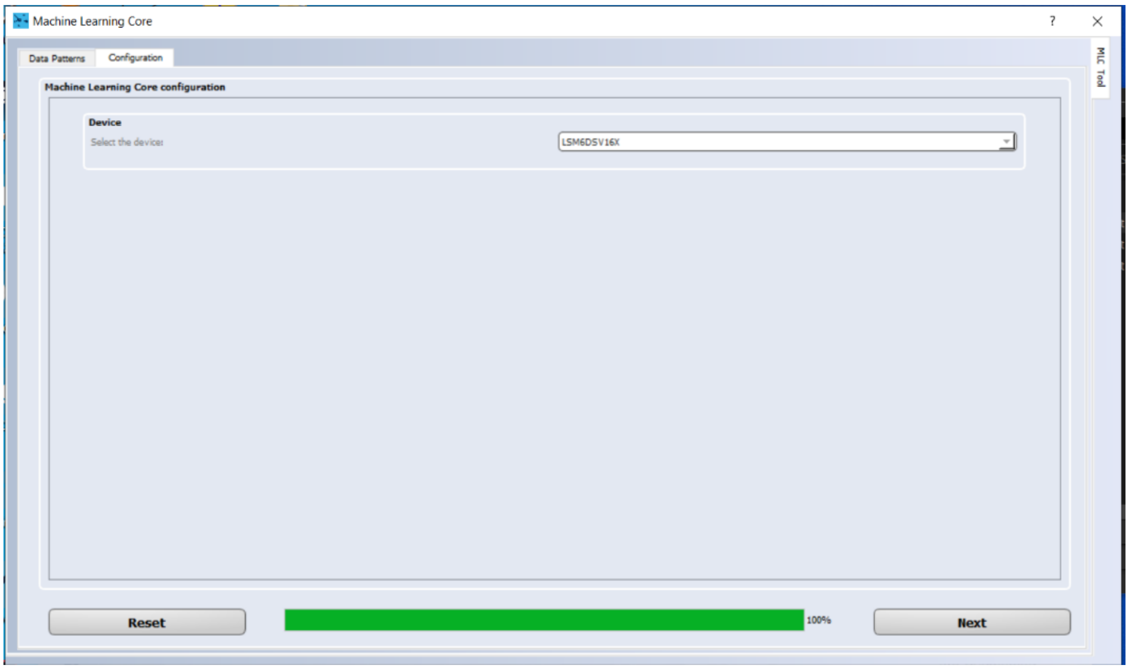
Sample [#]	AccX	AccY	AccZ	GyrX	GyrY	GyrZ	ExtX	ExtY	ExtZ
1261	2520	-398	915	-0.08	-0.07	-1.05	980	0	0
1262	2522	-392	915	-0.084	-0.28	-0.7	1120	0	0
1263	2524	-392	915	-0.084	-0.28	-0.7	1120	0	0
1264	2526	-390	909	-0.088	-0.14	-0.7	1540	0	0
1265	2528	-392	906	-0.089	0.7	-0.63	1470	0	0
1266	2530	-392	906	-0.089	0.7	-0.63	1470	0	0

Set Class (Label): false | LOAD

Data Patterns Loaded | CLEAR

Figure 35: UNICO GUI Screenshot 4

Browse csv files, assign labels, and load them.



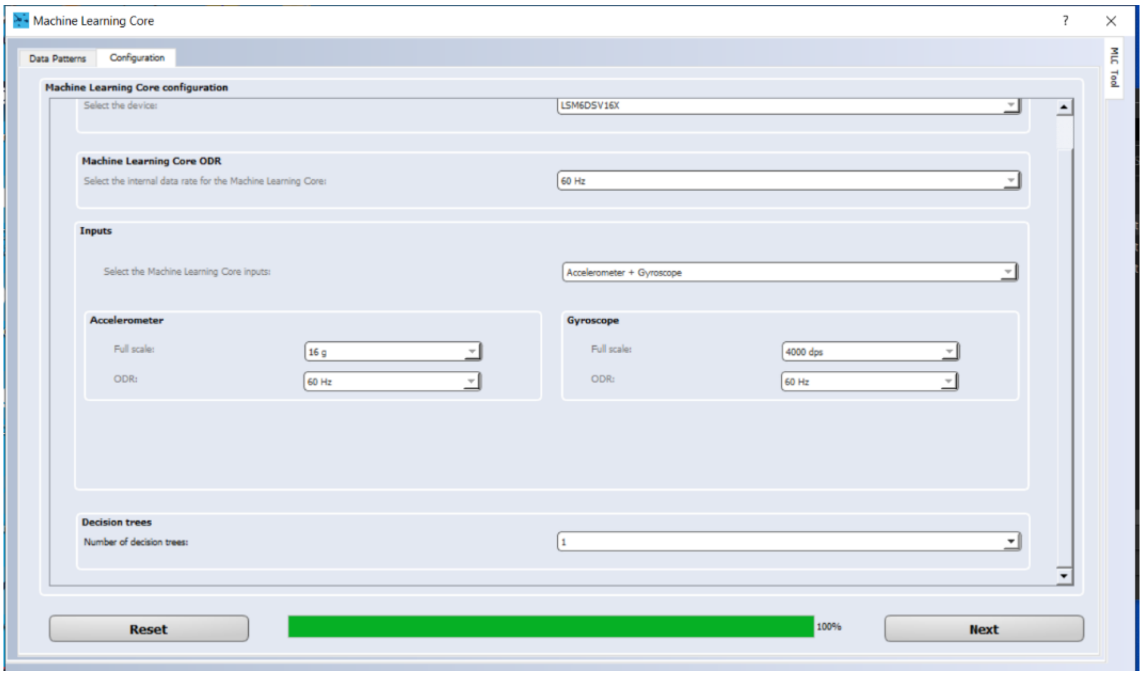
Machine Learning Core configuration

Device

Select the device: LSM6DSV16X

Figure 36: UNICO GUI Screenshot 5

Go to the ‘Configuration’ tab.



Machine Learning Core configuration

Select the device: LSM6DSV16X

Machine Learning Core ODR

Select the internal data rate for the Machine Learning Core: 60 Hz

Inputs

Select the Machine Learning Core inputs: Accelerometer + Gyroscope

Accelerometer

Full scale: 16 g

ODR: 60 Hz

Gyroscope

Full scale: 4000 dps

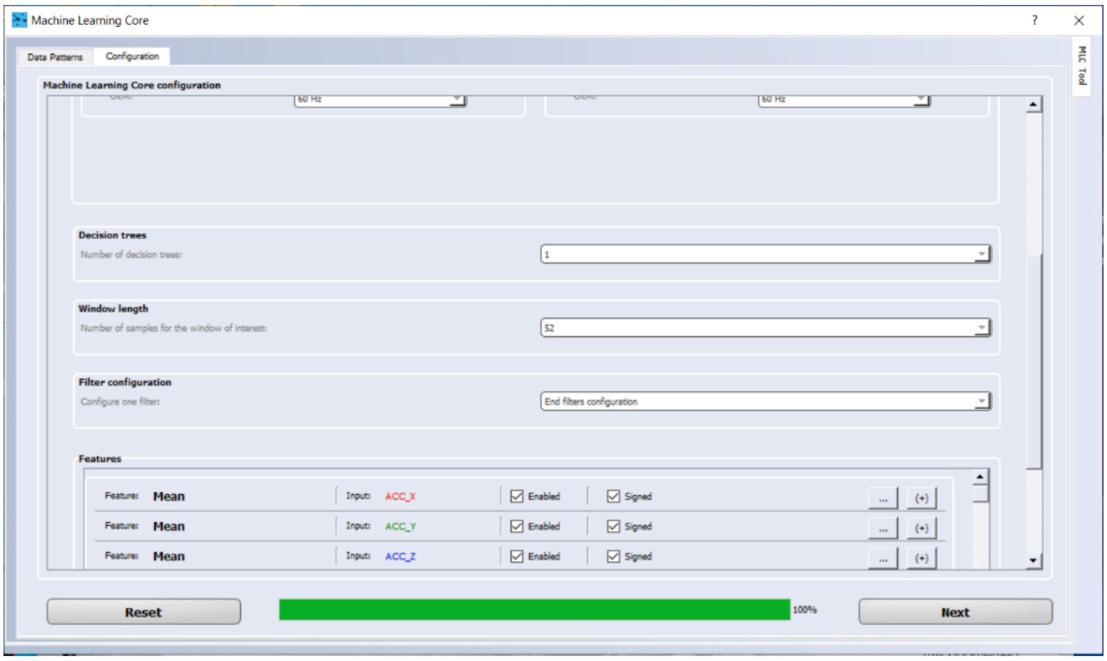
ODR: 60 Hz

Decision trees

Number of decision trees: 1

Figure 37: UNICO GUI Screenshot 6

Select ‘Device’, ‘Machine Learning Core ODR’, and ‘Inputs’. Set the inputs’ Full scale and ODR.


Machine Learning Core

Data Patterns | Configuration

Machine Learning Core configuration

ODR: 60 Hz | ODR: 60 Hz

Decision trees
Number of decision trees: 1

Window length
Number of samples for the window of interest: 52

Filter configuration
Configure one filter: End filters configuration

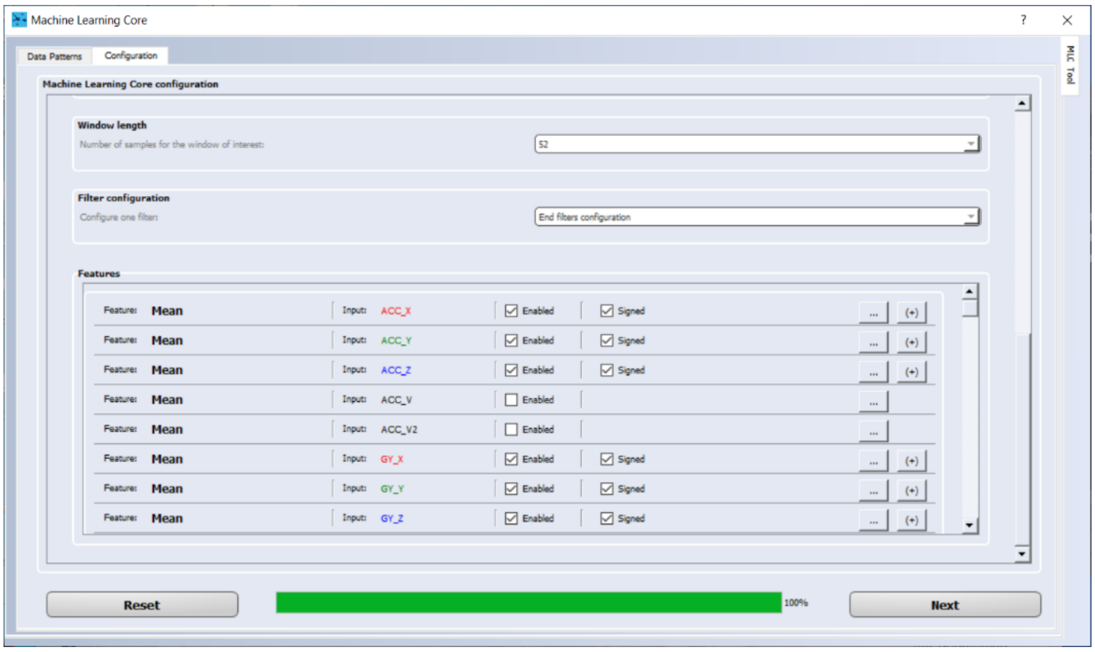
Features

Feature: Mean	Input: ACC_X	☑ Enabled	☑ Signed
Feature: Mean	Input: ACC_Y	☑ Enabled	☑ Signed
Feature: Mean	Input: ACC_Z	☑ Enabled	☑ Signed

Reset | 100% | Next

Figure 38: UNICO GUI Screenshot 7

Set number of ‘Decision trees’, ‘Window length’, and ‘Filter configuration’.


Machine Learning Core

Data Patterns | Configuration

Machine Learning Core configuration

Window length
Number of samples for the window of interest: 52

Filter configuration
Configure one filter: End filters configuration

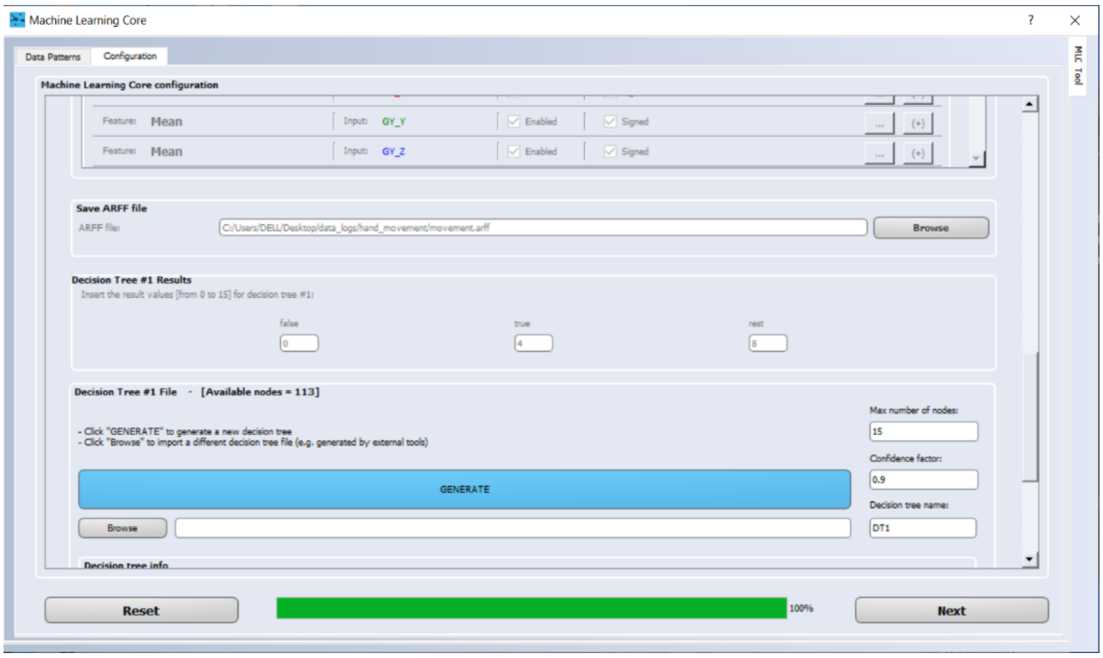
Features

Feature: Mean	Input: ACC_X	☑ Enabled	☑ Signed
Feature: Mean	Input: ACC_Y	☑ Enabled	☑ Signed
Feature: Mean	Input: ACC_Z	☑ Enabled	☑ Signed
Feature: Mean	Input: ACC_V	☐ Enabled	
Feature: Mean	Input: ACC_V2	☐ Enabled	
Feature: Mean	Input: GY_X	☑ Enabled	☑ Signed
Feature: Mean	Input: GY_Y	☑ Enabled	☑ Signed
Feature: Mean	Input: GY_Z	☑ Enabled	☑ Signed

Reset | 100% | Next

Figure 39: UNICO GUI Screenshot 8

Select ‘Features’ that you want for the decision tree(s).


Machine Learning Core

Machine Learning Core configuration

Feature: Mean | Input: GY_Y | ✓ Enabled | ✓ Signed

Feature: Mean | Input: GY_Z | ✓ Enabled | ✓ Signed

Save ARFF file

ARFF file: C:/Users/DELL/Desktop/data_logs/hand_movement/movement.arff

Decision Tree #1 Results

Insert the result values [from 0 to 15] for decision tree #1:

false: 0 true: 4 rest: 8

Decision Tree #1 File - [Available nodes = 113]

- Click "GENERATE" to generate a new decision tree
- Click "Browse" to import a different decision tree file (e.g. generated by external tools)

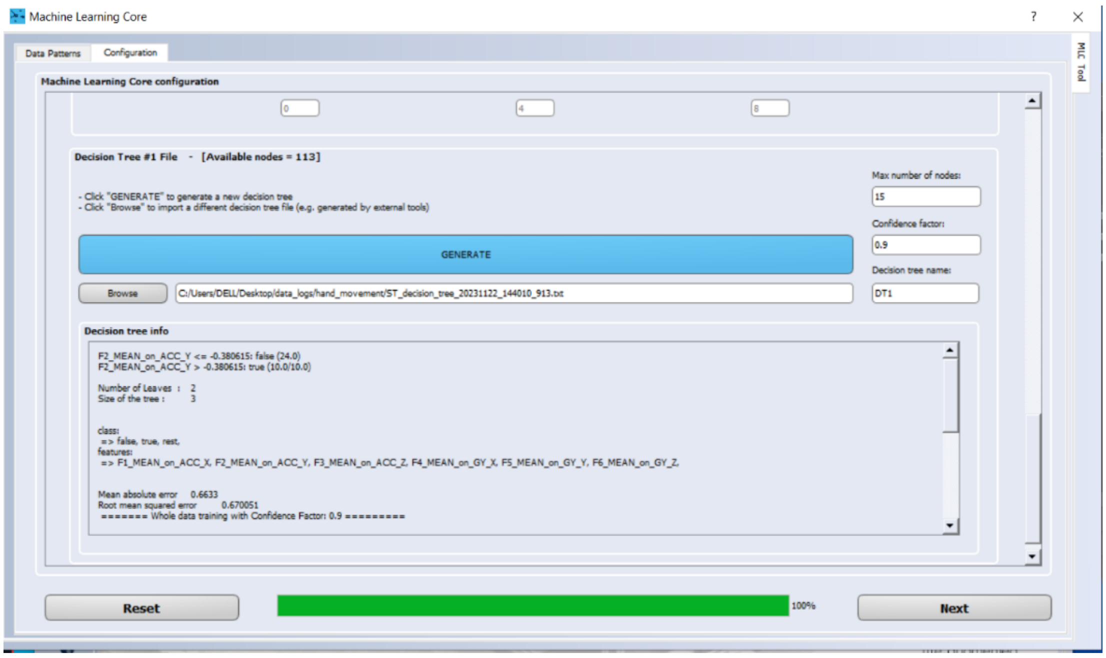
Max number of nodes: 15

Confidence factor: 0.9

Decision tree name: DT1

Figure 40: UNICO GUI Screenshot 9

Save ARFF file. And then assign numerical values to the Decision tree Results.


Machine Learning Core

Machine Learning Core configuration

0 4 8

Decision Tree #1 File - [Available nodes = 113]

- Click "GENERATE" to generate a new decision tree
- Click "Browse" to import a different decision tree file (e.g. generated by external tools)

Max number of nodes: 15

Confidence factor: 0.9

Decision tree name: DT1

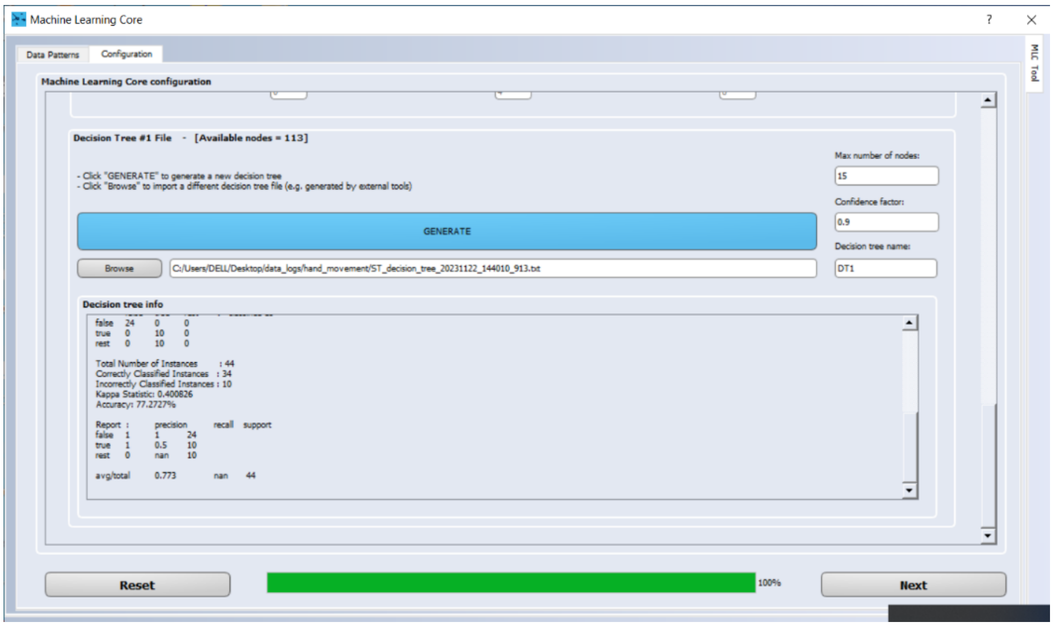
C:/Users/DELL/Desktop/data_logs/hand_movement/ST_decision_tree_20231122_144010_913.txt

Decision tree info

```
F2_MEAN_on_ACC_Y <= -0.380615: false (24.0)
F2_MEAN_on_ACC_Y > -0.380615: true (10.0/10.0)

Number of Leaves  :   2
Size of the tree :    3


class:
 => false, true, rest,
features:
 => F1_MEAN_on_ACC_X, F2_MEAN_on_ACC_Y, F3_MEAN_on_ACC_Z, F4_MEAN_on_GY_X, F5_MEAN_on_GY_Y, F6_MEAN_on_GY_Z,


Mean absolute error    0.6633
Root mean squared error       0.670051
 ======= Whole data training with Confidence Factor: 0.9 =========
```

Figure 41: UNICO GUI Screenshot 10

Click ‘GENERATE’ to generate the decision tree. You can aslo set ‘Max number of nodes’, ‘Confidence factor’, and ‘ Decision tree name’ before generating the decision tree.



Machine Learning Core configuration

Decision Tree #1 File - [Available nodes = 113]

- Click "GENERATE" to generate a new decision tree
- Click "Browse" to import a different decision tree file (e.g. generated by external tools)

Max number of nodes: 15

Confidence factor: 0.9

Decision tree name: DT1

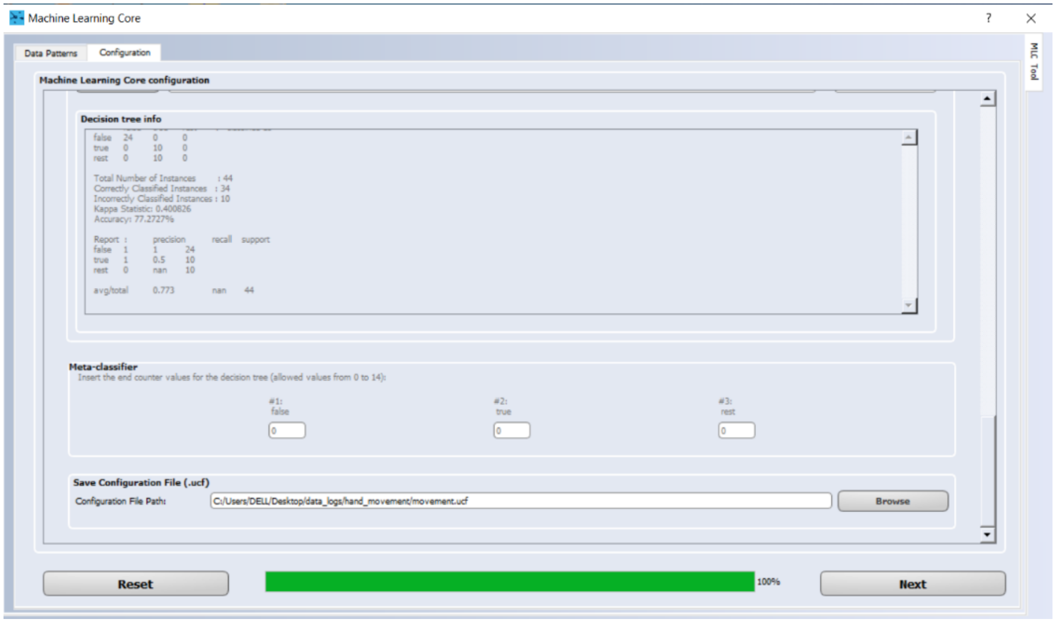
C:/Users/DELL/Desktop/data_logs/hand_movement/ST_decision_tree_20231122_144010_913.txt

Decision tree info

```
false   24   0    0
true    0    10   0
rest    0    10   0

Total Number of Instances       : 44
Correctly Classified Instances   : 34
Incorrectly Classified Instances : 10
Kappa Statistic: 0.400826
Accuracy: 77.2727%

Report :      precision      recall   support
false   1     1        24
true    1     0.5      10
rest    0     nan      10

avg/total     0.773         nan     44
```

Figure 42: UNICO GUI Screenshot 11

You can see the text file to see the decision tree generated. This info is also displayed on the GUI.



Machine Learning Core configuration

Decision tree info

```
false   24   0    0
true    0    10   0
rest    0    10   0

Total Number of Instances       : 44
Correctly Classified Instances   : 34
Incorrectly Classified Instances : 10
Kappa Statistic: 0.400826
Accuracy: 77.2727%

Report :      precision      recall   support
false   1     1        24
true    1     0.5      10
rest    0     nan      10

avg/total     0.773         nan     44
```

Meta-classifier

Insert the end counter values for the decision tree (allowed values from 0 to 14):

#1: false 0

#2: true 0

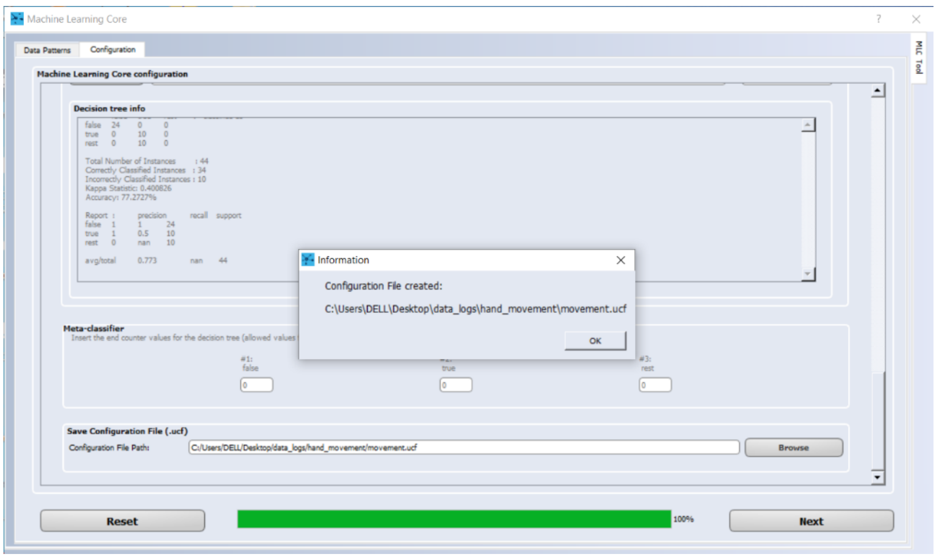
#3: rest 0

Save Configuration File (.ucf)

Configuration File Path: C:/Users/DELL/Desktop/data_logs/hand_movement/movement.ucf

Figure 43: UNICO GUI Screenshot 12

Assign ‘Meta classifier’. Finally, save the configuration file.



Machine Learning Core configuration

Decision tree info

```
false   24   0    0
true    0    10   0
rest    0    10   0

Total Number of Instances      : 44
Correctly Classified Instances  : 34
Incorrectly Classified Instances : 10
Kappa Statistic: 0.400826
Accuracy: 77.2727%

Report :       precision      recall   support
false   1      1        24
true    1      0.5      10
rest    0      nan      10

avg/total      0.773          nan      44
```

Information

Configuration File created:

C:\Users\DELL\Desktop\data_logs\hand_movement\movement.ucf

Meta-classifier

Insert the end counter values for the decision tree (allowed values

#1: false: 0
true: 0
#3: rest: 0

Save Configuration File (.ucf)

Configuration File Path: C:/Users/DELL/Desktop/data_logs/hand_movement/movement.ucf

Figure 44: UNICO GUI Screenshot 13

Configuration file is created.

6.3.2.3 *Uploading Configuration file to the Board using ST BLE:*

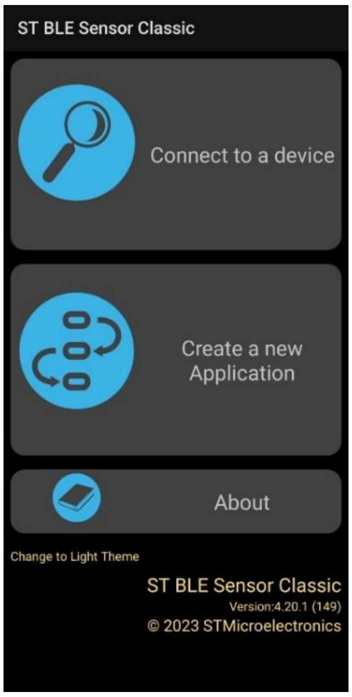


Figure 45: ST BLE Sensor App Screenshot 22

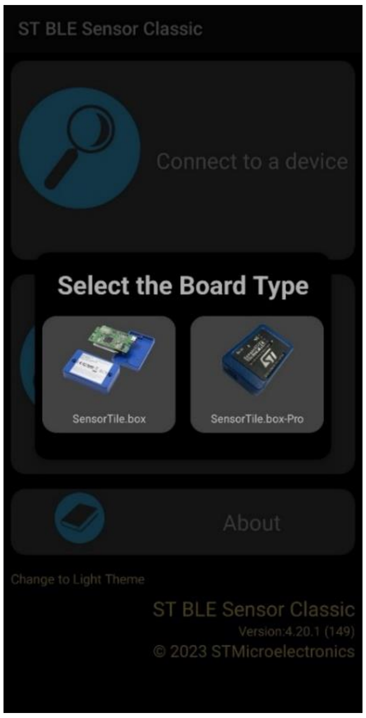


Figure 46: ST BLE Sensor App Screenshot 23

Open the ST BLE Sensor Classis app. Press 'Create a new Application', select the Board Type.

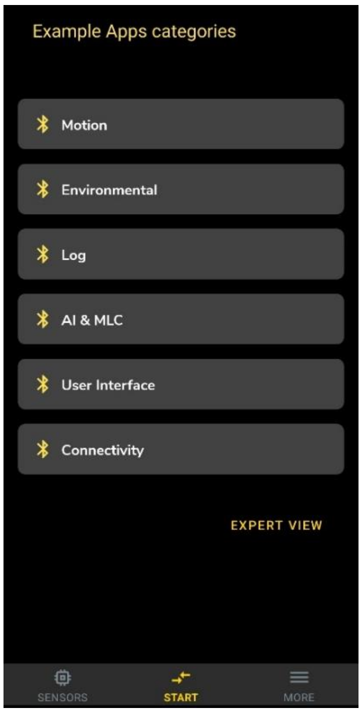
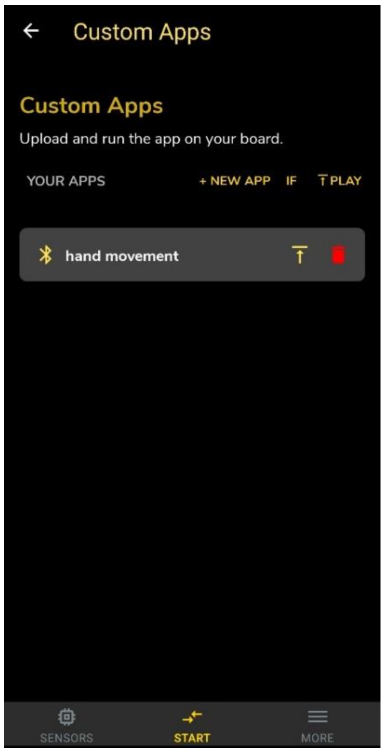


Figure 47: ST BLE Sensor App Screenshot 24

Press ‘EXPERT VIEW’.



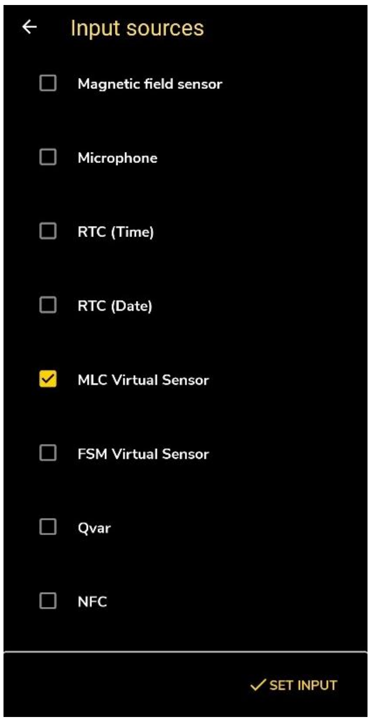
Custom Apps

Upload and run the app on your board.

hand movement

Figure 48: ST BLE Sensor App Screenshot 25

Press ‘+NEW APP’.



- [ ] Magnetic field sensor
- [ ] Microphone
- [ ] RTC (Time)
- [ ] RTC (Date)
- [✓] MLC Virtual Sensor
- [ ] FSM Virtual Sensor
- [ ] Qvar
- [ ] NFC

Figure 49: ST BLE Sensor App Screenshot 26

Select ‘MLC Virtual Sensor’ from the Input sources.

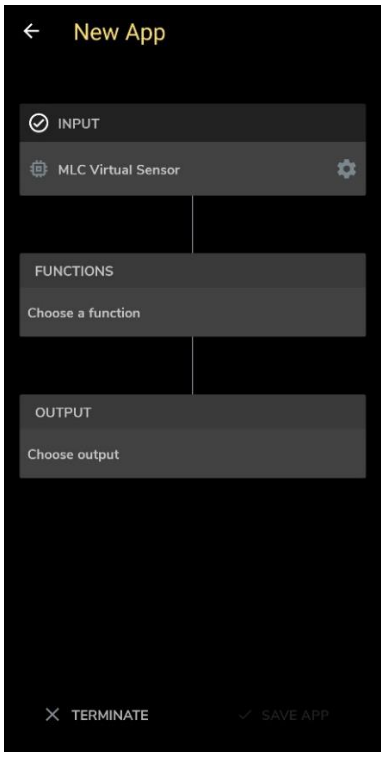


Figure 50: ST BLE Sensor App Screenshot 27

Press the settings icon next to ‘MLC Virtual Sensor’.

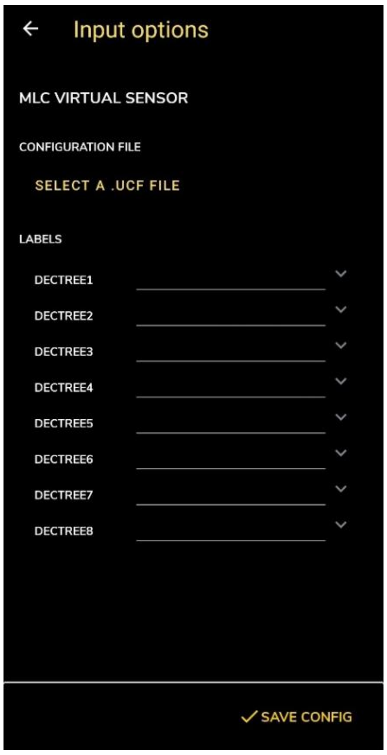
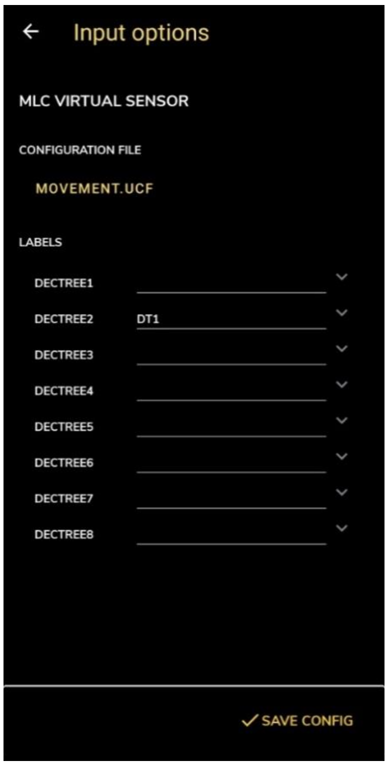


Figure 51: ST BLE Sensor App Screenshot 28

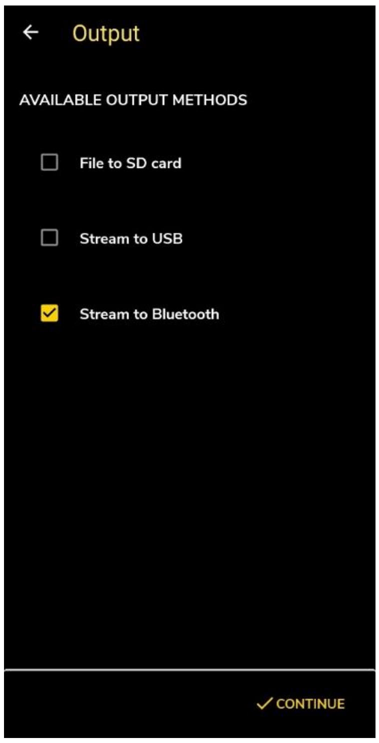
Select a ucf file.



DECTREE1	
DECTREE2	DT1
DECTREE3	
DECTREE4	
DECTREE5	
DECTREE6	
DECTREE7	
DECTREE8	

Figure 52: ST BLE Sensor App Screenshot 29

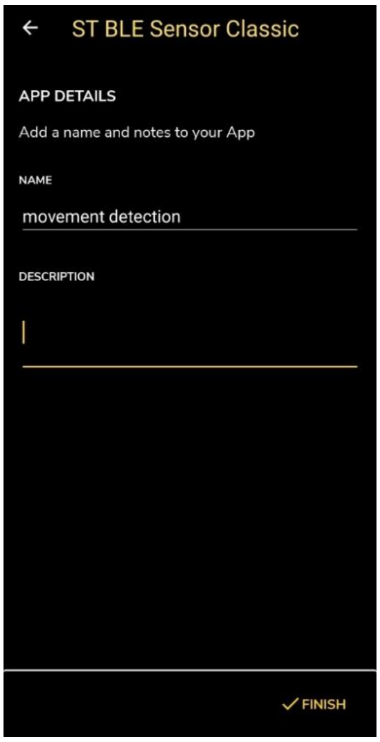
Save Config after uploading the ucf file.



- [ ] File to SD card
- [ ] Stream to USB
- [✓] Stream to Bluetooth

Figure 53: ST BLE Sensor App Screenshot 30

Select ‘Stream to Bluetooth’ option in the Output tab and press Continue.


ST BLE Sensor Classic

APP DETAILS

Add a name and notes to your App

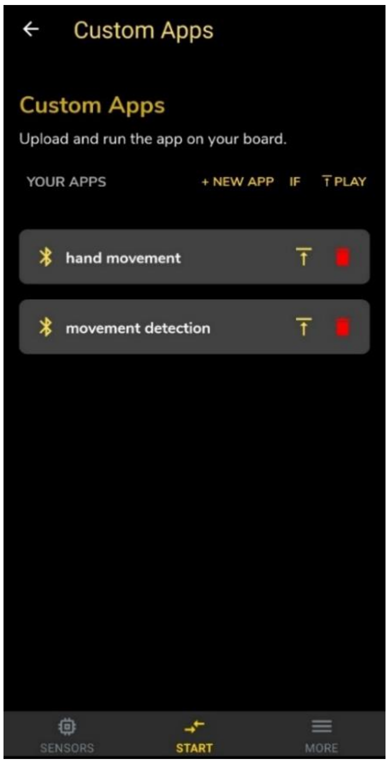
NAME

movement detection

DESCRIPTION

Figure 54: ST BLE Sensor App Screenshot 31

Save app. Give a name and an optional description and press Finish.


Custom Apps

Custom Apps

Upload and run the app on your board.

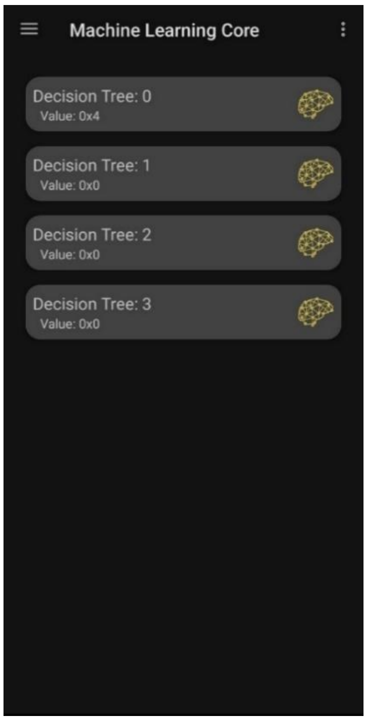
YOUR APPS

hand movement

movement detection

Figure 55: ST BLE Sensor App Screenshot 32

Upload the app by pressing the upload option and selecting the board.



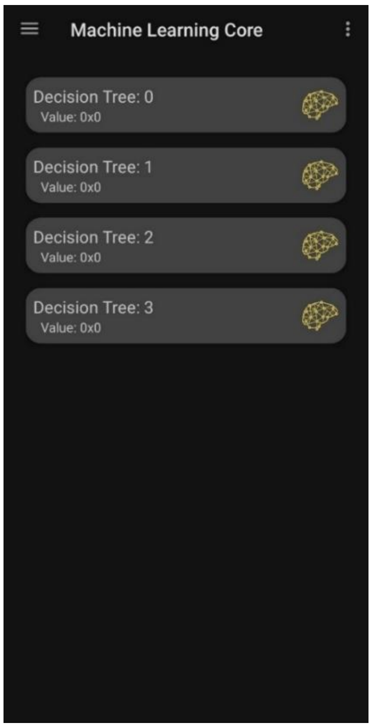
Decision Tree: 0
Value: 0x4

Decision Tree: 1
Value: 0x0

Decision Tree: 2
Value: 0x0

Decision Tree: 3
Value: 0x0

Figure 56: ST BLE Sensor App Screenshot 33



Decision Tree: 0
Value: 0x0

Decision Tree: 1
Value: 0x0

Decision Tree: 2
Value: 0x0

Decision Tree: 3
Value: 0x0

Figure 57: ST BLE Sensor App Screenshot 34

You can observe the output changing on the ‘Decision Tree: 0’.

6.3.3 MLC Experiments

Once the MLC pipeline was established, some MLC experiments were tried as a starting point.

6.3.3.1 Data Collected

The data was collected for 9 different combinations of sensor placement and arm movement (3 different sensor positions **times** 3 different arm movements). For each combination, 3 data samples were collected.

Sensor Position	**Arm Movement**	**Sagittal**	**Frontal**	**45º**
Sagittal		3	3	3
Frontal		3	3	3
Backward		3	3	3

Table 1: Number of Data Samples

This data corresponds to 27 samples. It is per hand per person. So, for one person, there will be 54 samples in total, 27 for the right hand and 27 for the left hand.

Below is the image showing the different planes (sagittal, horizontal and frontal) for better understanding.

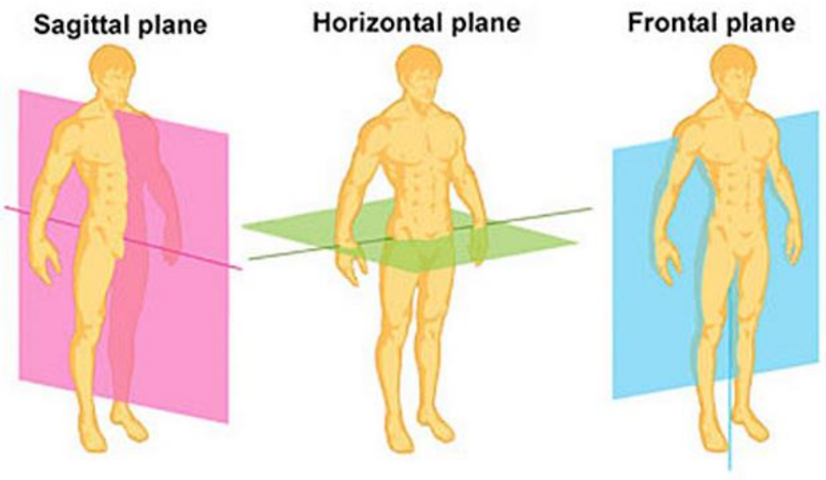


Figure 58: Different planes [13]

[13] https://www.teachpe.com/anatomy-physiology/planes-of-movement

Below are the images showing the sagittal sensor placement on both right and left hands.

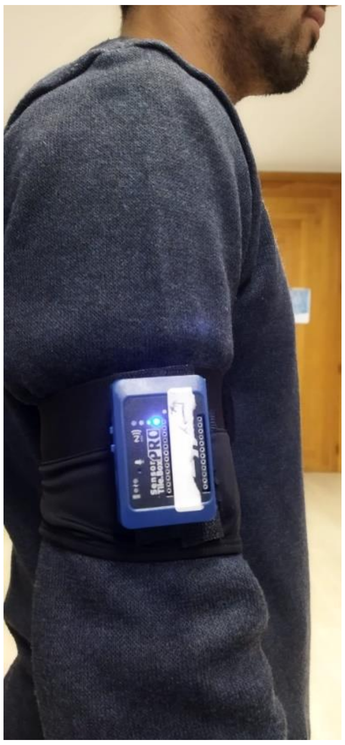


Figure 59: SensorTile.Box PRO kit mounted on the right arm

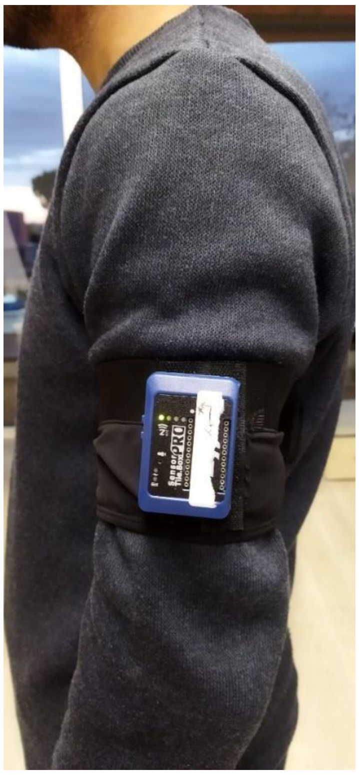


Figure 60: SensorTile.Box PRO kit mounted on the left arm

6.3.3.2 Models Created

There were several models created and tested. For simplicity and understanding, let’s assign simple variables to the data categories above.

Sensor Positions: Sagittal = A, Frontal = B, Backward = C

Arm Movements: Sagittal = D, Frontal = E, 45^o = F

So, for example, case AD would mean the data collected when sensor position was sagittal and arm movement was also sagittal. Case CE would mean the data collected when sensor position was backward, and the arm movement was frontal.

1) In the first case of decision tree generation, the case AD was given ‘true’ label, and the rest of the cases (AE, AF, BD, BE, BF, CD, CE, and CF) were given ‘false’ labels. So, if the sensor is mounted in sagittal position and the arm moves along the sagittal plane, the model should identify it as a correct movement, otherwise, not.
2) In the second case, AD, BD, and CD were given true labels, and the rest (AE, AF, BE, BF, CE, and CF) were given the false label. So, no matter the sensor placement, if the arm movement is sagittal, the model should identify it as a correct movement, otherwise, false.
3) In the third case, AD was given the true label, and AE and AF were given the false labels. So, this case was restricted to only sagittal sensor placements.
4) In the fourth case, two decision trees were generated in the configuration. One for identifying the correct movement of right arm, and another for the left arm. AD, BD, and CD were given true labels, and the rest (AE, AF, BE, BF, CE, and CF) were given the false label. Both for the right-hand and the left-hand datasets.
5) In this case, again, two decision trees were generated in the configuration. One for identifying the correct movement of right arm, and another for the left arm. This time, AD was given the true label, and AE and AF were given the false labels. So, this case was restricted to only sagittal sensor placements, but for both hands.
6) All the previous configurations were created based on just one person’s data and tested on the same person. In this case, the data for two people was used to create the configuration. AD was given the true label, and AE and AF were given the false labels. Then, it was tested on one person.

For each of these models, the following configuration settings were set when creating the MLC:

- MLC Frequency: 30 Hz
- Sensors Frequencies: 60 Hz
- Window Length: 52
- Feature: Mean

More details about setting these parameters will come later. As of now, these values were set by default.

6.3.3.3 Performance

The performance was observed by testing the model against the cases for which it was created. None of the models were accurate enough to consider them good. There were some cases in which models were able to identify the correct movement, but most of the time they gave wrong predictions.

Below is the summary of the models' characteristics in tabular form.

S.No.	**True Label**	**False Label**	**Hands**	**No of people in the dataset**
1	AD	AE, AF, BD, BE, BF, CD, CE, CF	Right	1
2	AD, BD, CD	AE, AF, BE, BF, CE, CF	Right	1
3	AD	AE, AF	Right	1
4	AD, BD, CD	AE, AF, BE, BF, CE, CF	Both	1
5	AD	AE, AF	Both	1
6	AD	AE, AF	Right	2
7	AD	AE	Right	1

Table 2: MLC models' characteristics

In the last table entry, when nothing seemed to be working, it was decided to focus on just one arm right now (for example, right arm) and limit the True/False cases only to AD and AE initially, so that the resulting model will be very simple. Then after we have a working model that differentiates between the AD and AE cases, we can extend it with the remaining cases, and for the left arm too. But even the simplest model was not giving any good results at this moment.

Seeing the performance of the above models, it was noticed that using the accelerometer and the gyroscope data as it is might not be a good option in this case. One possible alternative was to use Quaternions / Euler Angles.

6.4 Correcting the Input Data Type

After noticing that the accelerometer and gyroscope data might not be sufficient to train the decision tree, it was time to look at the other possible options available within the box kit. It was the only restriction that we had when using the SensorTile.Box PRO kit that we can only use the resources available inside the kit and only the ST provided software (possibly some external software as well that makes working with ST devices possible). One of those possibilities was to use the Quaternion data belonging to the arm movements, which is possible thankfully with the help of ST BLE Sensor app.

6.4.1 Using Quaternions instead of Accelerometer /Gyroscope Data

The ST BLE Sensor Classic app offers a ‘Sensor Fusion – Quaternion’ application that uses the LSM6DSV16X iNemo 6-axis inertial measurement unit and the LIS2MDL 3-axis compensated magnetometer to show the orientation estimation of SensorTile.boxPRO in the 3D space. The algorithm used in the sensor fusion algorithm (MotionFX library) embedded in the SensorTile.box MCU which uses the sensors data as inputs (9-axis) and calculates the quaternion coefficient describing the rotation of 3D cube model. The sensor fusion algorithm is basically a Kalman filter that minimizes the sensors’ inaccuracies based on peculiarities of other sensors; it includes gyroscope calibration and magnetometer calibration (to compensate for the magnetometer offset).

The only problem with the Quaternion data is that it contains 4 data sequences. A quaternion is represented as $q = a + bi + cj + dk$, where a, b, c, and d are real numbers representing the components of the quaternion, and i, j, and k are the quaternion units. Here is what each component represents:

- a: The real part of the quaternion

- b: The coefficient of the *i* unit, which is one of the imaginary units of the quaternion.
- c: The coefficient of the *j* unit, another imaginary unit of the quaternion.
- d: The coefficient of the *k* unit, third imaginary unit of the quaternion.

Together, these components form a four-dimensional vector that represents a point in the space of quaternions. The real part (a) represents the scalar part of the quaternion, while the imaginary parts (b, c, d) represent the vector part. This combination allows quaternions to represent rotations and translations concisely and efficiently. [20]

The problem with the four-dimensional quaternion data comes from a limitation of MLC (Machine Learning Core) training tool of Unico-GUI. It only expects a 3-dimensional data for each sensor data used in the training. It does not make sense to use 3 dimensions of the 4-dimensional data for training as it would be inaccurate. So, we needed a way to convert this 4-dimensional data into 3 dimensions.

6.4.2 Euler Angles

Thankfully, we have Euler angles, the solution to our previous problem. Euler angles are a set of three angles (usually called roll, pitch and yaw) used to represent the orientation of an object or coordinate system in three-dimensional space. The three angles typically represent rotations about the three principal axes of the coordinate system. There are several conventions for defining Euler angles, such as XYZ, ZYX, ZXY, XZY, XZX, YXY, and so on. In total, there are 12 of these conventions. Each convention specifies the order in which the rotations are applied, and about which axes. However, Euler angles have a very important limitation known as Gimbal lock, which occurs when two of the three angles align, leading to a loss of one degree of freedom. [20] [21]

To counteract the Gimbal lock, if we apply the rotation sequence in which the first axis is the axis of rotation of the arm, we will never run into the problem of Gimbal lock. Applying the rotation sequence with the first axis as the axis of rotation of the arm helps counteract Gimbal lock by stabilizing the primary reference frame and preventing intermediate axis alignment issues. By following this step, the risk of intermediate steps leading to a loss of degree of freedom is minimized because the initial step already establishes a stable reference frame, thus preserving the necessary degrees of freedom and avoiding Gimbal lock. [20]

To convert quaternions into Euler angles, SciPy's Spatial Transformations (scipy.satial.transform) package was used. This package contains a Rotation class (scipy.spatial.transform.Rotation) which provides the functionality to convert a 4-dimensional quaternion into a 3-dimensional Euler angle.[14] When converting a quaternion into a Euler angle, a rotation sequence needs to be specified which tells along which axis the rotation was performed first. Only AD and AE cases are being considered right now to keep the resulting model as simple as possible, as explained before as well. AD data was converted using the ZXY sequence since the arm moves around the z-axis of the sensor (or it can be said along the x-y plane of the sensor) in the case of AD. And the AE data was converted using the XZY sequence since the arm moves around the x-axis of the sensor (or along the y-z plane of the sensor) in the case of AE.

In the table below, different rotation sequences are mentioned for different scenarios.

Sensor Position	Movement Plane	Forward	Lateral	45°
Lateral		zxy	xzy	zxy
Front		xzy	zxy	xzy
Back		xzy	zxy	xzy

Table 3: Different rotation sequences for different scenarios

To get the Euler angles, first Quaternions data was collected. 11 measurements were taken for each case (AD and AE). So, 11 measurement files containing Quaternions for the AD case, and 11 more measurement files containing Quaternions for the case of AE. After we have the Quaternions data, Euler angles were calculated according to their respective rotation sequences.

After the Euler angles were calculated, the offsets were also removed from them for the data to make sense. So, for each arm movement, all three of the Euler angles were starting from 0 degrees, and if the arm goes all the way up, one of the angles (depending on the sensor

[14] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.html

orientation) can be seen changing from 0 degrees to 180 degrees. This offset removal step was done on all the collected data.

After removing the offsets, the angles were plotted for the AD and AE cases to get the visual representation of the angles. All the 11 measurements were plotted on a single graph to see the consistency of those measurements. These plots are shown in the results section. After plotting these graphs, it was noticed that some of these measurements are not consistent with each other what was not expected. For example, we know that one of the angles should approach 180 degrees during the arm movement, but it was not the case with some measurements. So, it was decided to discard those measurements and to not use them in the training of the decision tree, as they could be misleading and might affect the training step. These plots are shown in the results section under the Change of Data Type subsection.

After discarding inconsistent measurements, the data columns were arranged in an order before using it for training. This is necessary because for different cases, different rotation sequences were applied when converting to Euler angles. So, the cases for which ZXY rotation sequence was used, their roll, pitch and yaw columns would be rearranged as pitch, yaw and roll. Similarly, for the cases, for which XZY rotation sequence was used, their roll, pitch and yaw columns would be rearranged as roll, yaw and pitch. This is to keep the column orders consistent in the order XYZ.

In the case of AD, the ZXY rotation sequence was used to convert the Quaternions into Euler angles. So, in this case, roll corresponds to the rotation along Z-axis, pitch along the X-axis and yaw along the Y-axis. Whereas, in the case of AE, the XZY rotation sequence was used. Now in this case, the roll corresponds to rotation along the X-axis, pitch along the Z-axis and yaw along the Y-axis.

Therefore, before using this data for training, it was necessary that the data columns of all the cases (AD and AE for right now) were in the same order. Therefore, the column order was changed such that they follow the XYZ order. Hence, in the case of AD, the roll-pitch-yaw order was changed to pitch-yaw-roll to follow the XYZ order. Similarly, in the case of AE, the roll-pitch-yaw order was changed to roll-yaw-pitch to follow the XYZ sequence. Now the data was ready for training.

6.5 Adapting Training to the Real Scenario

The MLC was trained using the data obtained in the previous step to distinguish between the AD and AE movements. The AD movements were labeled as True, whereas the AE movements were labeled as False. After training the MLC, there was still no success. The MLC still could not differentiate between the AD and the AE movements, not 100% correctly. There would be instances in which the output would be correct, predicting the correct arm movement, but there will be other instances as well in which it would not predict accurately, or it would be stuck on just output (output not changing with any arm movement). So, it was very inconsistent and not reliable at all.

After not getting any success, it was decided to do a little experiment. This experiment was limited to the AD movements only. For AD, those measurements whose roll values were going beyond 100 degrees were considered as True and the rest of them were considered false. This experiment, unfortunately, had no success either. It had the same effect on the output that it was most of the time stuck at one value and was not changing. The point of this experiment was to make the decision-making process as easy as possible, so that MLC should at least be able to predict this one correctly, but as usual, it failed.

After these repeated failures, it was necessary to make sure that the machine was interpreting the live data the same way as the data it was trained on. Because when the machine was trained, it was not trained on the actual Quaternions data that was collected in the beginning. Those Quaternions were changed to Euler angles and then the offsets were removed as well. This was the data that was used for training, and it was important to make sure that the machine was also able to see the data the same way when it was making predictions. If the machine could not interpret the interpreted data, then how would it have made the correct predictions? So, we needed a way to first run the machine offline, where it could be provided with the interpreted data instead of the live data so that it could be seen if the trained algorithm was able to make the correct predictions on the interpreted data to verify the correctness of the algorithm. In this way, it could be confirmed that at least the machine was trained correctly.

Fortunately, ST mentions about offline analysis in one of its documents to verify the classification performance of MLC. ST provides a python script that could be run to verify the performance. The script verifies the performance by making a comparison of the decision tree and the ARFF file. The script was run on the first ML configuration that was created for distinguishing between the AD and AE movements. After running this script, it gave 100%

accuracy on the model. Still, it was not the exact thing which was intended to be tried. The actual intention was to see if the model performs accurately on the new unseen data, but this python script from ST does not even use any training/testing data to test the model. It just compares the contents of ARFF file and decision tree file. Nevertheless, it still gave 100% accuracy for the model, which means that it is accurate, at least, according to ST criteria.

6.6 Customizing Machine Learning Decision Parameters

After these repeated failures, there were not a lot of things left to try. One of the things that were left was to customize some of the decision tree training parameters, such as ODR (Output Data Rate), window length, and feature selection.

6.6.1 MLC and Sensor(s) ODR

ODR is the output data rate. It is the sampling frequency at which the device runs and samples the readings. The MLC (Machine Learning Core) ODR and the sensors ODR are not too important, individually, but together their combination is important to understand. If the sensors' ODR is twice as much as MLC's ODR, then it implies that sensors are running at twice as speed as MLC. It means that MLC is sampling only every other reading (not all readings) available from the sensors, as its speed is exactly half as the sensors' speed. So, for example, if there are 100 readings available from the sensors, MLC is only sampling 50 of them, ignoring one sample after reading each sample. On the other hand, if both the MLC ODR and sensors ODR are same, then they both are running at the same speed and MLC is reading every sample that is provided from the sensors. The sensors include the accelerometer and the gyroscope sensors.

It is important to understand this in the context of window length, and how many readings there are available in a measurement file. If there are, for example, 150 readings available in a measurement file, and let's say that the sensors ODR is twice as much as the MLC ODR, then there are exactly 75 readings (half of the actual number of readings) available to the MLC for the training.

6.6.2 Window Length

One of the decision tree training parameters that could possibly affect the outcome by changing it was the window length. So, it was decided to test it by changing the window length parameter which is set before the model training. This parameter tells how many data values are going to be used in a window and then the selected feature is applied on those values in each window. So, for example, if a window length of 13 is selected, it will group the values in the dataset in multiple windows, with each window having 13 non-overlapping samples. And then, for example, if mean is selected as the feature, it will calculate the mean of the 13 values in the window for each window.

Up until this point, the window length of 52 was selected which was a default setting from the ST. The results obtained so far were from this setting. So, it was decided to play around with this setting and see how it affects the outcome. Setting the window length to 13 had the most promising results so far, in which the sensor output was changing according to the expected arm movement, but it was still inconsistent and unreliable.

13, 26, 39, 52, 65, 78 and 91 are the different window lengths that were tried, along with different combinations of features. Window lengths above 91 were unable to be tested because the software was unable to generate a decision tree for those scenarios as there were not enough samples in the measurement file. The detailed results of the outcomes of different window lengths are reported in the results section.

6.6.3 Feature Selection

This is another important parameter to test after the window length. This parameter applies, whatever feature that is selected, to the values in the window. Up until this point, only the mean values were used as feature (both in accelerometer/gyroscope case and the Euler angles case). There are other features that can be tried such as minimum, maximum, variance, energy, and many more. The best way to decide which feature to use is by plotting the data and observing which feature can better grasp the underlying data. The detailed results about the outcome of different features are reported in the results section.

6.7 Going Back to The Accelerometer/Gyroscope Data

After getting repeated failures even when using Euler angles, and that too after trying different options, the only possibility left was to try the accelerometer and gyroscope data again, but this time with customizable machine learning parameters. Like before, the parameters that could be changed are window length, feature selection, and MLC and/or sensors ODR.

As before, the focus was to have a working algorithm that can at least differentiate between the AD and AE movements. At some point, a third state was included as well other than AD and AE states, called the Idle state. Moreover, there were initially only 3 measurement files for each hand movement as described in the MLC Experiments section, but after some iterations, more measurement files were included to have more data for the better ML training. To identify which feature selection could be best in a particular case, the corresponding graphs were plotted of the accelerometer and the gyroscope data, which are shown in the results section.

All the results, that were tried with different options and different configurations, are reported in the Accelerometer/Gyroscope Data with Tunable ML Parameters subsection under the Results section.

6.8 Using Sensor on the Wrist instead of on the Arm

When using the sensor on the arm did not give any promising results, even with customizable ML decision parameters, there was still one thing that could be tried. It was to use the sensor on the wrist instead of on the arm. The reason for doing it would be prominent after looking at the plots of accelerometer and gyroscope data. When using the sensor on the wrist, the AD and AE have more differentiable data as compared to the case when the sensor was on the arm. These plots are shown in the results section.

Like the approach described in the previous Going Back to The Accelerometer/Gyroscope Data section, different options and configurations were tried in this case as well, such as different window lengths, different feature selections, and different MLC and/or sensors ODRs. The results for all these different configurations are reported in the Sensor on the wrist subsection under the results section.

7 Results

All the different results from different subsections in the Methodology chapter are included in this chapter.

7.1 MLC Experiments Performance

In the MLC Experiments subsection under the Workflow Setup section, some MLC models were created to test different cases and distinguish between different arm movements. Unfortunately, this exercise was not successful at all as it was not able to recognize any movement correctly. Their performance is reported in the table below.

S.No.	True Label	False Label	Hands	No of people in the dataset	Performance
1	AD	AE, AF, BD, BE, BF, CD, CE, CF	Right	1	Not able to distinguish between the true movements and the false movements.
2	AD, BD, CD	AE, AF, BE, BF, CE, CF	Right	1	Not able to distinguish between the true movements and the false movements.
3	AD	AE, AF	Right	1	Not able to distinguish between the true movements and the false movements.
4	AD, BD, CD	AE, AF, BE, BF, CE, CF	Both	1	Not able to distinguish between the true movements and the false movements.
5	AD	AE, AF	Both	1	Not able to distinguish between the true

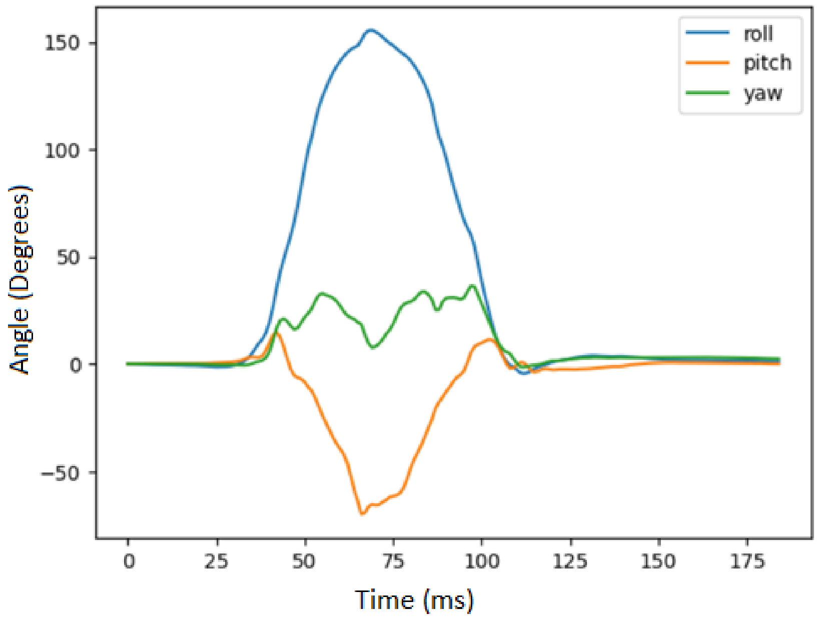
					movements and the false movements.
6	AD	AE, AF	Right	2	Not able to distinguish between the true movements and the false movements.
7	AD	AE	Right	1	Not able to distinguish between the true movements and the false movements.

Table 4: MLC models' performance

7.2 Change of Data Type

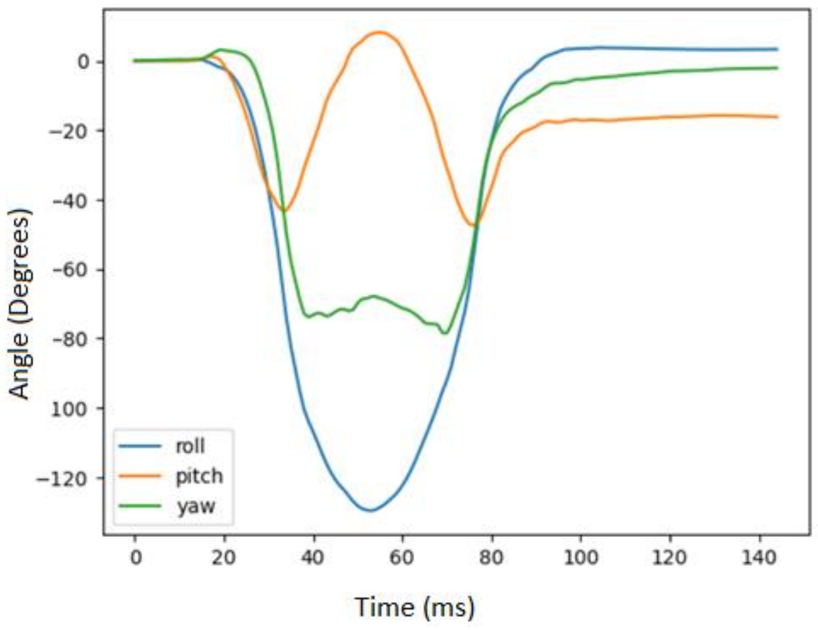
In the section Correcting the Input Data Type, it is mentioned how the data type was changed from Accelerometer/Gyroscope to Quaternions and then eventually to Euler Angles. After calculating Euler angles, they were first plotted to get the visual representation of the data before using it in the training process.

Below are the plots of the Euler angles of one of the 11 measurements for both the case of AD and AE.



Euler angles in the case of AD

In the plot above, Euler angles are plotted in degrees (after removing the offsets) for the case of AD. It can be seen in the plot that the roll is crossing 150 degrees which satisfies the arm movement condition of AD that it should approach 180 degrees.



Euler angles in the case of AE

In the plot above, Euler angles are plotted in degrees (after removing the offsets) for the case of AE. It can be seen in the plot that the roll is crossing the -120 degrees which satisfies the arm movement condition of AE that it should approach -180 degrees. (It is negative because of the sensor orientation which is not important and can be ignored)

To verify the integrity of the collected data, we calculated the min, max and range of all three Euler angles of all 11 measurements for each of the 9 cases. If these statistics are not consistent among the 11 measurements for any certain case, it means that the measurements that are inconsistent have some problems in the original data and therefore, it is better if these measurements should not be used in the MLC training because it could contribute towards the improper training of the MLC. Below are the statistics for both AD and AE cases.

7.2.1.1 AD

Roll:

Measurement File No.	Roll Minimum	Roll Maximum	Roll Range	Roll Average	Roll Variance
0	-4.489397	155.398591	159.887987	37.347883	3107.493016
1	- 20.588897	144.516195	165.105092	38.225601	3585.076437
2	- 34.780051	154.545851	189.325902	55.543096	3799.889076
3	- 98.940774	76.011203	174.951977	- 12.627701	3603.296925
4	- 4.578309	145.965063	150.543372	42.700858	3424.963224
5	- 14.167902	149.444853	163.612756	46.647889	3919.386788
6	- 77.015355	26.428775	103.444130	- 23.339770	1296.144546
7	- 36.734936	35.850878	72.585814	1.289612	457.298503
8	- 21.771149	25.128225	46.899374	0.203756	197.052888
9	- 79.944679	51.072129	131.016808	- 2.042009	1589.107385
10	- 79.068447	15.932596	95.001043	- 26.378531	1200.495868

Table 5: Roll statistics for 11 measurement files in the case of AD

Pitch:

Measurement File No.	Pitch Minimum	Pitch Maximum	Pitch Range	Pitch Average	Pitch Variance
0	-69.984536	14.711747	84.696282	-9.002772	415.219902
1	-60.697133	9.365794	70.062927	-8.550391	378.191783
2	-74.414243	7.097643	81.511886	-13.501869	629.583228

3	-77.174331	1.850664	79.024995	-16.565927	552.212524
4	-72.995200	9.860994	82.856194	-14.599861	533.831940
5	-70.880741	4.695503	75.576244	-16.783718	535.537761
6	-66.212732	1.364536	67.577269	-14.224223	465.278813
7	-64.197848	0.367527	64.565374	-15.052183	443.078269
8	-57.416693	4.654139	62.070832	-11.606306	410.368598
9	-62.163317	0.800303	62.963620	-14.144839	393.196654
10	-62.515554	5.888517	68.404071	-11.940338	467.782123

Table 6: Pitch statistics for 11 measurement files in the case of AD

Yaw:

Measurement File No.	Yaw Minimum	Yaw Maximum	Yaw Range	Yaw Average	Yaw Variance
0	- 1.551632	36.309673	37.861305	9.405395	135.430361
1	- 15.415170	71.718241	87.133411	8.808493	423.460539
2	- 35.734448	115.701739	151.436187	16.233989	909.622705
3	- 99.485431	42.153388	141.638820	- 48.055325	1371.263712
4	-11.248516	34.104834	45.353349	4.398950	96.686227
5	- 15.678918	92.645691	108.324609	7.549187	411.314338
6	- 138.71615	12.884332	151.600484	- 62.390997	1617.369415
7	- 131.01892	18.832076	149.850995	- 39.462012	1800.931919
8	- 145.82738	9.474301	155.301679	- 40.671490	2401.546488

9	- 116.11692	28.973043	145.089964	- 42.172099	1436.294694
10	- 149.8152	7.633346	157.448542	- 61.927365	2084.179049

Table 7: Yaw statistics for 11 measurement files in the case of AD

7.2.1.2 AE

Roll:

Measurement File No.	Roll Minimum	Roll Maximum	Roll Range	Roll Average	Roll Variance
0	- 170.32182	6.656484	176.978301	- 46.114114	4982.901961
1	- 79.325571	265.600637	344.926208	62.945668	10040.190314
2	- 196.29223	6.358872	202.651103	- 67.920153	7562.612526
3	- 89.483829	254.964345	344.448174	70.484855	6403.271946
4	- 129.77568	3.827288	133.602973	- 34.526037	2532.447607
5	- 75.286624	281.932272	357.218896	41.658678	11481.381431
6	- 117.52575	5.190158	122.715911	- 29.380059	2025.688585
7	- 74.624606	283.377021	358.001626	31.754657	10455.056386
8	- 79.773611	278.824877	358.598488	29.538770	10549.816496
9	- 98.197295	6.792605	104.989900	- 23.598798	1476.584440
10	- 79.150467	278.675478	357.825944	40.948097	1200.495868

Table 8: Roll statistics for 11 measurement files in the case of AE

Pitch:

Measurement File No.	Pitch Minimum	Pitch Maximum	Pitch Range	Pitch Average	Pitch Variance
0	- 51.139762	15.245324	66.385086	-13.405650	220.151933
1	- 16.467538	37.406586	53.874124	9.503592	185.833718
2	- 28.747716	37.213613	65.961329	- 0.004949	396.464597
3	- 39.768946	26.414497	66.183443	1.567019	470.183698
4	- 47.450739	8.259586	55.710325	- 15.358958	189.212408
5	- 1.428872	56.641645	58.070517	19.021048	257.628909
6	- 55.035261	0.321924	55.357185	- 18.341376	224.103538
7	- 0.115809	55.128147	55.243956	21.090121	238.185647
8	- 0.395301	63.779175	64.174475	26.028255	310.856466
9	- 52.383915	2.981422	55.365337	- 15.127074	214.831610
10	- 0.288509	58.580543	58.869052	18.726928	256.852659

Table 9: Pitch statistics for 11 measurement files in the case of AE

Yaw:

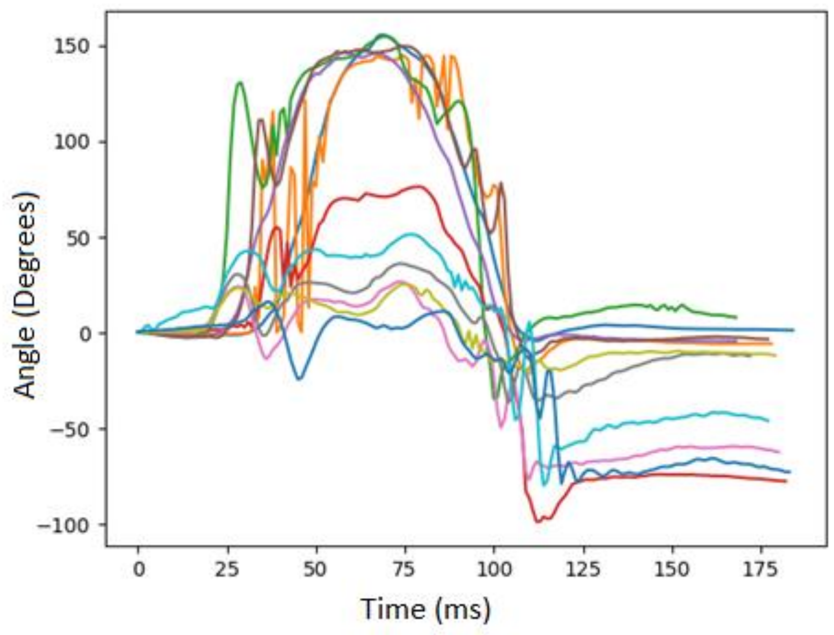
Measurement File No.	Yaw Minimum	Yaw Maximum	Yaw Range	Yaw Average	Yaw Variance
0	- 109.77023	9.735430	119.505659	- 27.323833	1717.612344
1	- 348.38225	9.560465	357.942714	- 108.09169	19937.414371
2	- 141.15161	9.435283	150.586896	- 40.890038	2694.950905
3	- 327.89875	15.928750	343.827505	- 104.09318	13353.762437

4	- 78.568890	3.193969	81.762859	- 24.739707	940.468884
5	- 341.68708	15.648473	357.335553	- 95.726873	20171.592677
6	- 80.347598	1.414416	81.762014	- 24.302663	1008.454505
7	- 345.44622	13.385107	358.831325	- 119.19072	22687.186277
8	- 347.07191	12.197960	359.269868	- 110.50293	21254.110404
9	- 82.691795	3.077206	85.769000	- 25.484108	1024.065347
10	- 346.22333	13.308520	359.531853	- 109.27046	21617.649393

Table 10: Yaw statistics for 11 measurement files in the case of AE

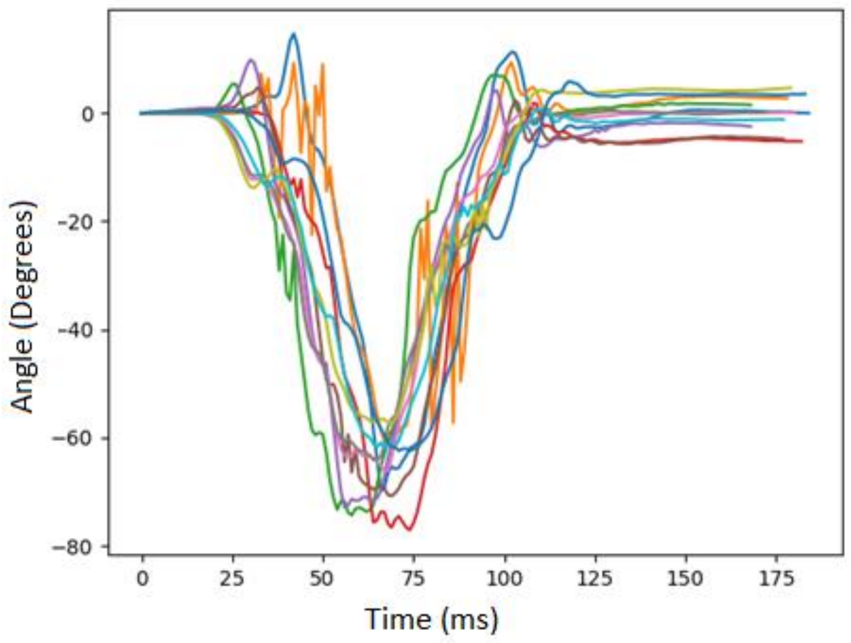
The roll, pitch and yaw were also plotted separately, but combinedly for the different measurements. In this way, it is easier to see the above-mentioned effect of inconsistency on each angle separately with respect to the different measurements. In this way, the roll values of all measurements for a specific case (for example AD) were plotted on one graph. Similarly, the pitch and yaw values were plotted separately for that specific case. In the case of AD, the roll values should approach 180 degrees for each measurement. But there were some measurements for which it was not even crossing 100 degrees. So, a 100-degree threshold was set, and the measurements that were not crossing this threshold were not used in the training. It was a similar situation in the case of AE. For AE, the roll values should approach –180 degrees (negative because of sensor orientation). But for some measurements, it was not the case, so those measurements were discarded too.

Below are the plots of the roll, pitch and yaw, plotted separately, but combinedly for all 11 measurements, and for each case, AD and AE. The first case is AD.



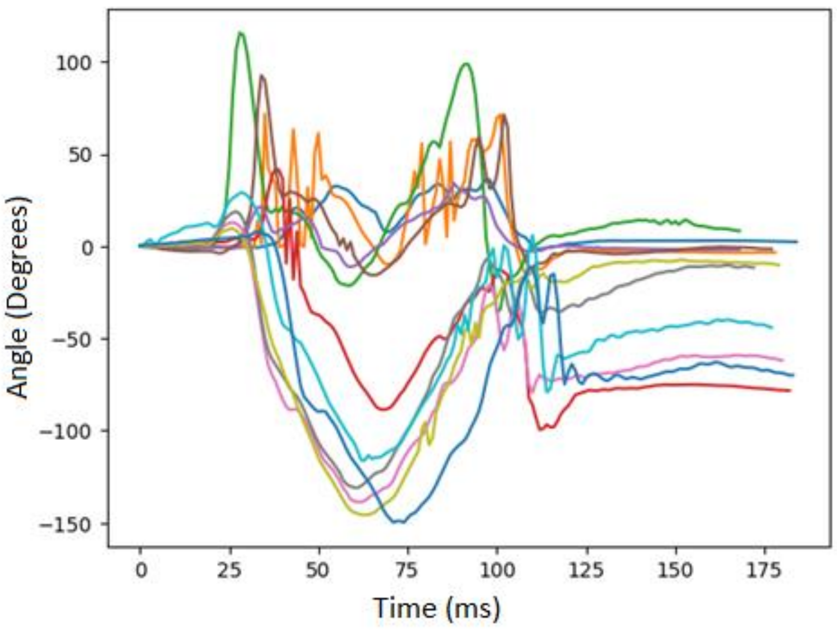
Roll angles for 11 measurements of AD

In the graph above, the roll angles are plotted for all 11 measurements for the case of AD. With the help of this graph, the consistency can be seen easily among different measurements with respect to one specific angle. It also helps to see which measurements are accurate and are approaching 180 degrees.



Pitch angles for 11 measurements of AD

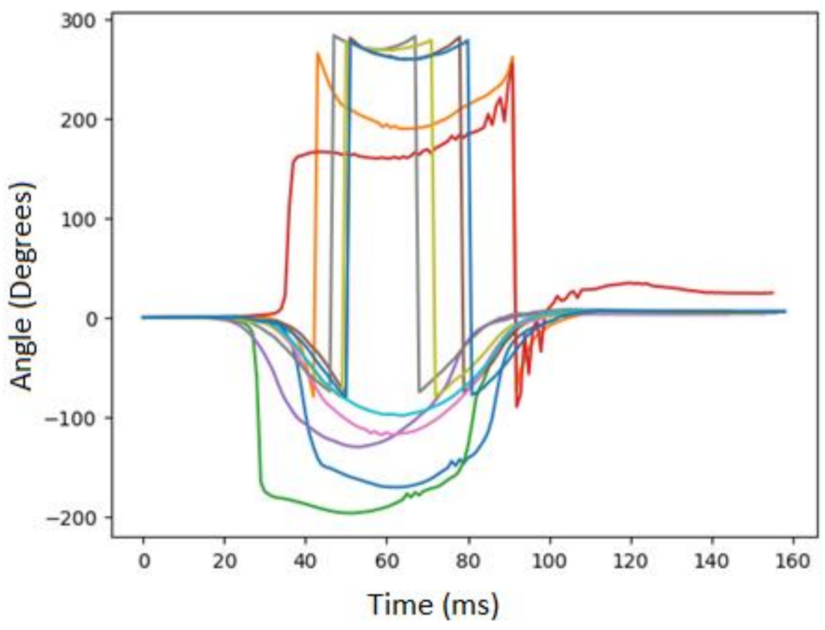
In the graph above, the pitch angles are plotted for all 11 measurements for the case of AD. But this plot is not as important as the roll one. Because roll is the angle that gives direct information about the arm movement/rotation in terms of angles in the direction we are interested in.



Yaw angles for 11 measurements of AD

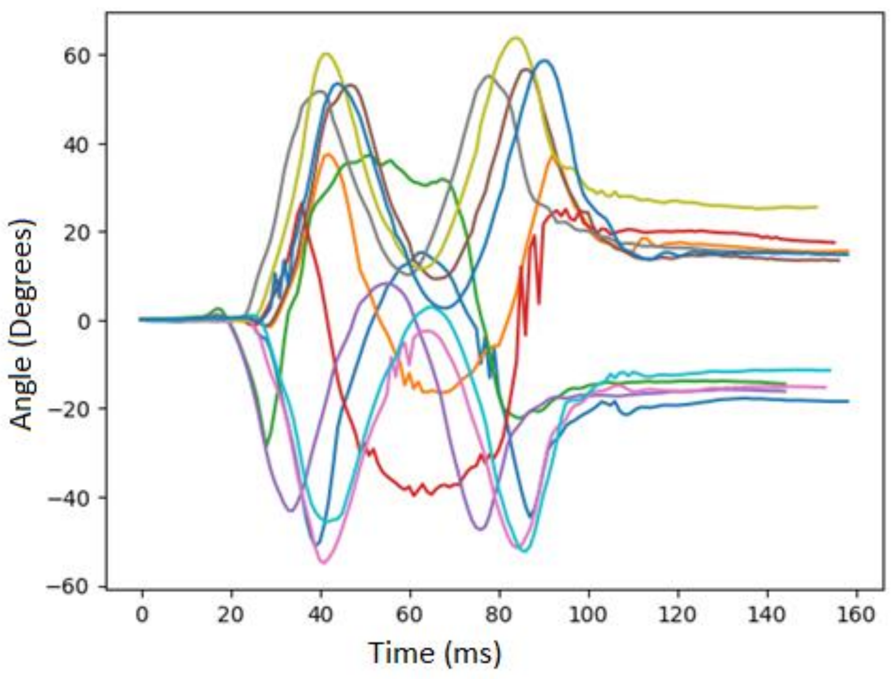
In the graph above, the yaw angles are plotted for all 11 measurements for the case of AD. Similarly, this plot is not as important as the roll one because of the reason stated before.

Now, below are the plots of the roll, pitch and yaw, plotted separately, but combinedly for all 11 measurements, for the case of AE.



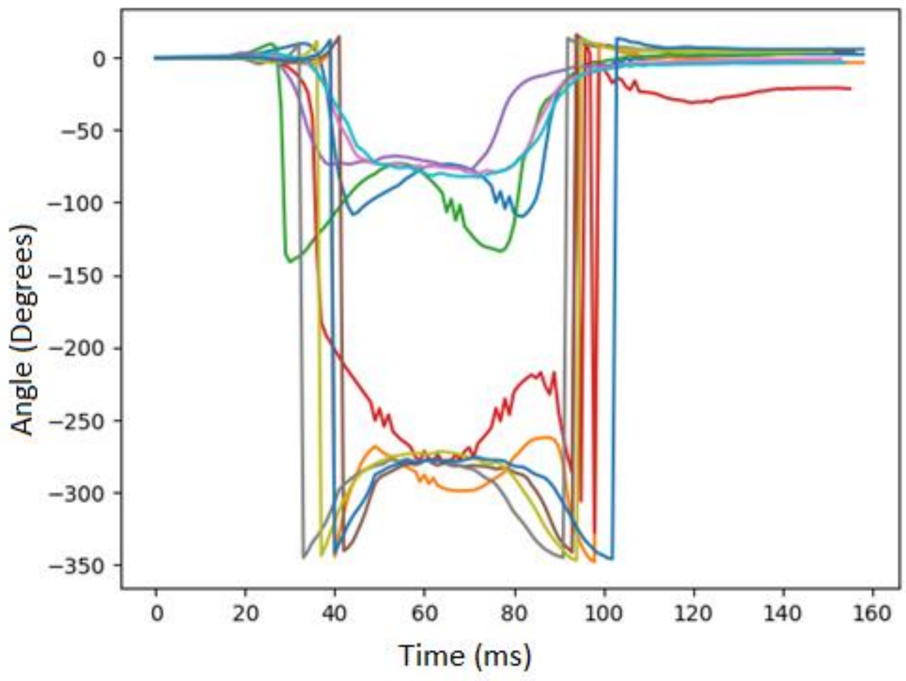
Roll angles for 11 measurements of AE

In the graph above, the roll angles are plotted for all 11 measurements for the case of AE. With the help of this graph, the inconsistency can be seen easily among different measurements with respect to one specific angle. It also helps to see which measurements are accurate and are approaching -180 degrees.



Pitch angles for 11 measurements of AE

In the graph above, the pitch angles are plotted for all 11 measurements for the case of AE. But this plot is not as important as the roll one. Because roll is the angle that gives direct information about the arm movement/rotation in terms of angles in the direction that we are interested in.

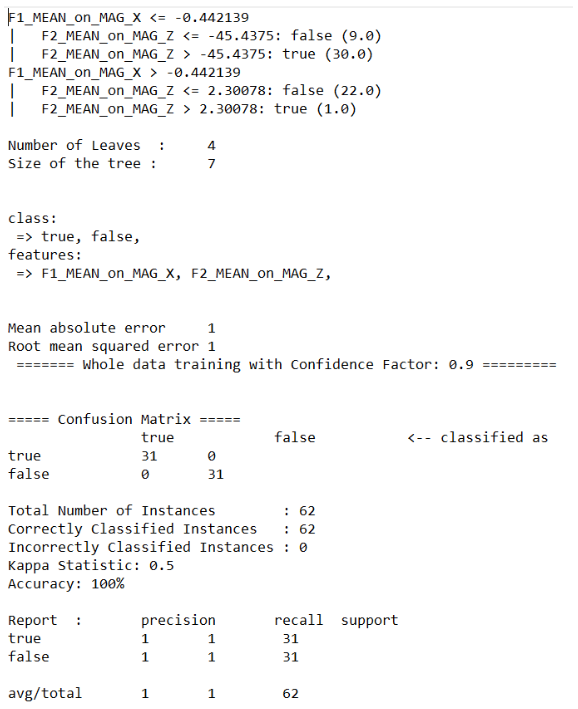


Yaw angles for 11 measurements of AE

In the graph above, the yaw angles are plotted for all 11 measurements for the case of AE. Similarly, this plot is not as important as the roll one because of the reason stated before.

7.3 Real Scenario Training

As discussed in the section Adapting Training to the Real Scenario, the decision tree training to distinguish between the AD and AE movements was no success. Below is the decision tree report for this trained MLC.

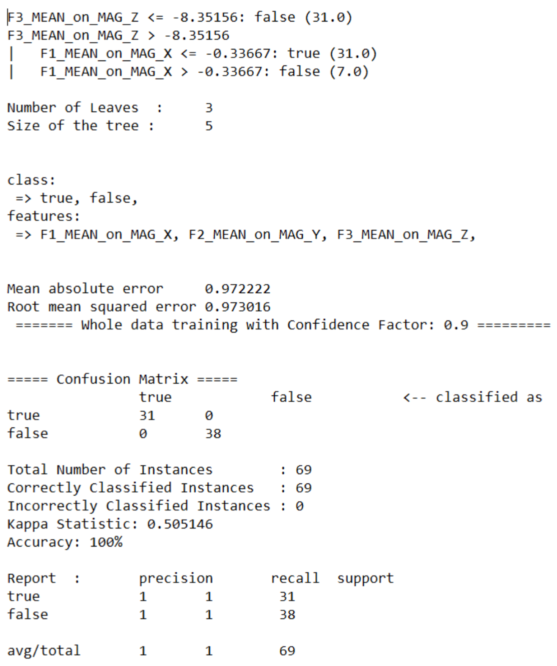


```
F1_MEAN_on_MAG_X <= -0.442139
|   F2_MEAN_on_MAG_Z <= -45.4375: false (9.0)
|   F2_MEAN_on_MAG_Z > -45.4375: true (30.0)
F1_MEAN_on_MAG_X > -0.442139
|   F2_MEAN_on_MAG_Z <= 2.30078: false (22.0)
|   F2_MEAN_on_MAG_Z > 2.30078: true (1.0)

Number of Leaves  :     4
Size of the tree :      7


class:
 => true, false,
features:
 => F1_MEAN_on_MAG_X, F2_MEAN_on_MAG_Z,


Mean absolute error     1
Root mean squared error 1
 ======= Whole data training with Confidence Factor: 0.9 =========


===== Confusion Matrix =====
                true            false           <-- classified as
true            31      0
false           0       31

Total Number of Instances        : 62
Correctly Classified Instances   : 62
Incorrectly Classified Instances : 0
Kappa Statistic: 0.5
Accuracy: 100%

Report  :       precision       recall   support
true            1       1        31
false           1       1        31

avg/total       1       1        62
```

Figure 61: Decision Tree report for trained MLC to distinguish between AD and AE movements

Although it can be seen in the report that the resulting decision tree has the 100% accuracy on the trained data, it was still unable to predict the correct movements.

Then, the AD experiment was tried which was no success either. Its resulting decision tree report is shown below.

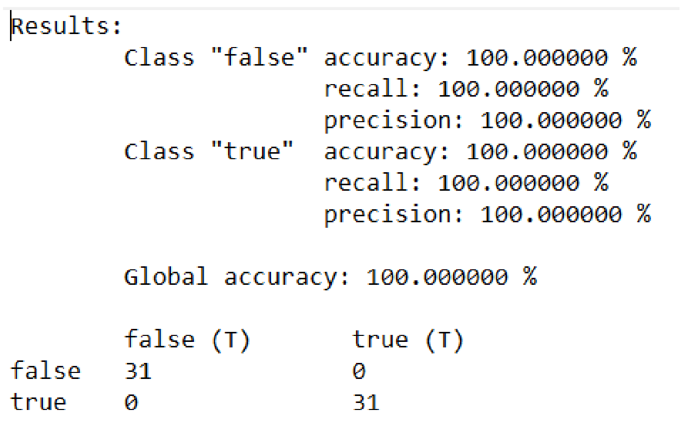


```
F3_MEAN_on_MAG_Z <= -8.35156: false (31.0)
F3_MEAN_on_MAG_Z > -8.35156
|   F1_MEAN_on_MAG_X <= -0.33667: true (31.0)
|   F1_MEAN_on_MAG_X > -0.33667: false (7.0)

Number of Leaves  :     3
Size of the tree :      5


class:
 => true, false,
features:
 => F1_MEAN_on_MAG_X, F2_MEAN_on_MAG_Y, F3_MEAN_on_MAG_Z,


Mean absolute error     0.972222
Root mean squared error 0.973016
 ======= Whole data training with Confidence Factor: 0.9 =========


===== Confusion Matrix =====
                true            false           <-- classified as
true            31      0
false           0       38

Total Number of Instances        : 69
Correctly Classified Instances   : 69
Incorrectly Classified Instances : 0
Kappa Statistic: 0.505146
Accuracy: 100%

Report  :       precision       recall   support
true            1       1        31
false           1       1        38

avg/total       1       1        69
```

Figure 62: Decision Tree report for AD experiment

In this case too, the resulting decision tree had 100% accuracy but still no success.

Then, the ST provided offline analysis was performed to verify the classification performance of MLC. It gave 100% accuracy for the model which was trained to distinguish the AD and AE movements. Below is the test report for accuracy.



```
Results:
        Class "false" accuracy: 100.000000 %
                      recall: 100.000000 %
                      precision: 100.000000 %
        Class "true"  accuracy: 100.000000 %
                      recall: 100.000000 %
                      precision: 100.000000 %

        Global accuracy: 100.000000 %

        false (T)        true (T)
false   31               0
true    0                31
```

Figure 63: Offline analysis report to test MLC accuracy

7.4 ML Decision Parameters

7.4.1 AD Experiment

The ML parameters such as window length and feature selection were customized to improve the performance. It was initially tried on AD experiment which is mentioned in the section Adapting Training to the Real Scenario. The results are reported below.

Below are the results of different window lengths and different features on AD experiment when the simple AD arm movement was performed after placing the sensor on the arm. In the AD experiment, the true cases are those cases for which roll value is greater than or equal to 100 degrees. Whereas the rest of the cases are false.

Window Length	**Feature**	**Mean**	**Min**	**Max**
13		Stuck at 0 (true)	Not changing. Stuck at whatever the initial value was.	Stuck at 0 (true)
26		Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 0 (true)
39		Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 4 (false)
52		Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 4 (false)
65		Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 4 (false)
78		Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 4 (false)
91		Stuck at 4 (false)	Stuck at 0 (true)	Stuck at 4 (false)

Table 11: Performance result of AD experiment with varying window lengths and features

In the table above, it can be seen that the output was mostly stuck at just one value. For mean and minimum features, it is mostly stuck at true even though when half AD movement was performed in which case roll value should be less than 100 degrees. And in the maximum feature, it is mostly stuck at false.

Below are the results of the different window lengths and different features on AD experiment when the sensor was randomly moved and rotated by holding it in hand. This was important to see if the sensor output was even changing or not when it was not changing in the typical AD movement.

Window Length	**Feature**	**Mean**	**Min**	**Max**
13		Stuck at 0 (true)	Randomly changing between 0 (true) and 4 (false). Not consistent.	Stuck at 0 (true). Started at 4 (false) and then immediately

			changed to 0 (true) and then stayed true.
26	Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 0 (true)
39	Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 4 (false)
52	Stuck at 0 (true)	Stuck at 0 (true)	Randomly changing between 0 (true) and 4 (false).
65	Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 4 (false)
78	Stuck at 0 (true)	Stuck at 0 (true)	Stuck at 4 (false)
91	Stuck at 4 (false)	Stuck at 0 (true)	Randomly changing between 0 (true) and 4 (false).

Table 12: Output of AD experiment when the sensor is randomly moved by hand

In the above table, it can be seen that the sensor output was somewhat changing under the influence of random sensor movements by hand, which shows that at least there was some working algorithm down there, but it just did not seem to be working correctly.

7.4.2 AD/AE Movements

The effect of changing the ML parameters to distinguish the AD/AE movements are reported below. Note that the AD movements are labeled as true, whereas AE movements are labeled as false. In this subsection, the results belonging to the Customizing Machine Learning Decision Parameters section are reported, in which case the Euler angles were used as the training data.

Below are the results of different window lengths and different features on the algorithm when the simple AD movement was performed after placing the sensor on the arm.

Window Length	Feature	Mean	Min	Max
13		Stuck at 0 (false)	Stuck at 4 (true)	Stuck at 4 (true)
26		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
39		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
52		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
65		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
78		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)

Table 13: Sensor outcome when performing simple AD movement

Below are the results of different window lengths and different features on the algorithm when the simple AE movement was performed after placing the sensor on the arm.

Window Length	Feature	Mean	Min	Max
13		Stuck at 0 (false)	Stuck at 4 (true)	Stuck at 4 (true)
26		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
39		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
52		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
65		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
78		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)

Table 14: Sensor outcome when performing simple AE movement

It can be seen in both of the tables above that the sensor output was usually stuck at one value, which was not even correct. For mean and minimum features, it was mostly stuck at true regardless of the AD or AE movement. And for maximum feature, it was stuck at false.

Below are the results of the different window lengths and different features when the sensor was randomly moved and rotated by holding it in hand. Again, this was important to see if the sensor output was even changing or not when it did not change in the typical AD and AE movements.

Window Length	**Feature**	**Mean**	**Min**	**Max**
13		Stuck at 0 (false)	Randomly changing between 4 (true) and 0 (false).	Stuck at 4 (true)
26		Randomly changing between 4 (true) and 0 (false)	Stuck at 4 (true)	Stuck at 0 (false)
39		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
52		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
65		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)
78		Stuck at 4 (true)	Stuck at 4 (true)	Stuck at 0 (false)

Table 15: Sensor output when rotating it randomly

The above table shows the output was in fact changing, but only in a limited number of cases. It means that there was some underlying algorithm that was working somewhat (because of the changing output values), but it just did not seem to be working correctly.

7.5 Accelerometer/Gyroscope Data with Tunable ML Parameters

As discussed in the Going Back to the Accelerometer/Gyroscope Data section, when nothing seemed to be working, the only option left worth trying was to go back to the accelerometer and gyroscope data, and train the decision tree again but this time with varying ML parameters such as window length, feature selection, and MLC and/or sensors ODR. The focus was on distinguishing between only the AD and AE movements, with AD movement labeled as true and AE as false.

The first configuration that was tried as a starting point was with the following settings: Window Length: 13, Feature Selection: Mean, MLC ODR: 30 Hz, Sensors ODR: 60 Hz. The decision tree was trained to distinguish between the AD and AE movements.

The result was that it seemed to be working as it was almost able to properly distinguish between the two movements. However, it was not consistent, like there were only a few instances in which it would predict the movements correctly. So, it still needed to be improved. The options for improvements were to different window lengths, or different feature selection, or different ODRs for MLC and sensors. Another possibility was to include a third state as well, called the Idle state in which the arm is just resting and not moving at all.

7.5.1 Plots

Before presenting any results, below are the plots of the accelerometer and the gyroscope data when the AD and AE movements were performed while the sensor was placed on the right bicep.

7.5.1.1 AD

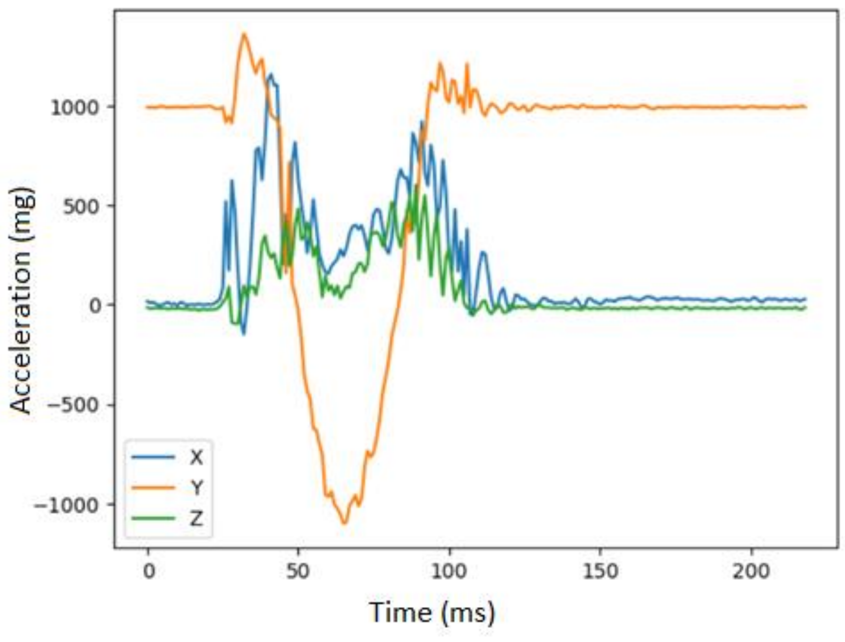


Figure 64: AD Accelerometer

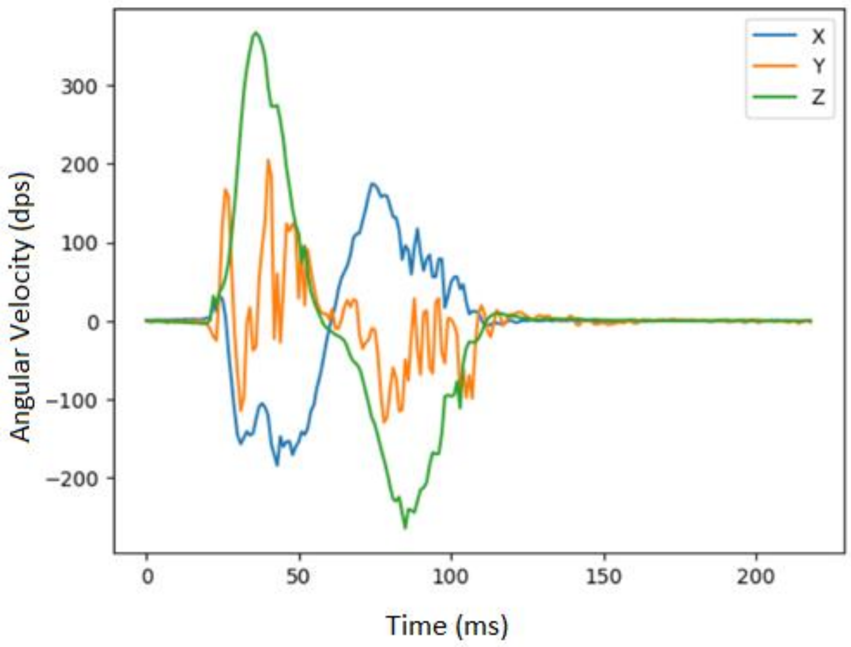


Figure 65: AD Gyroscope

7.5.1.2 AE

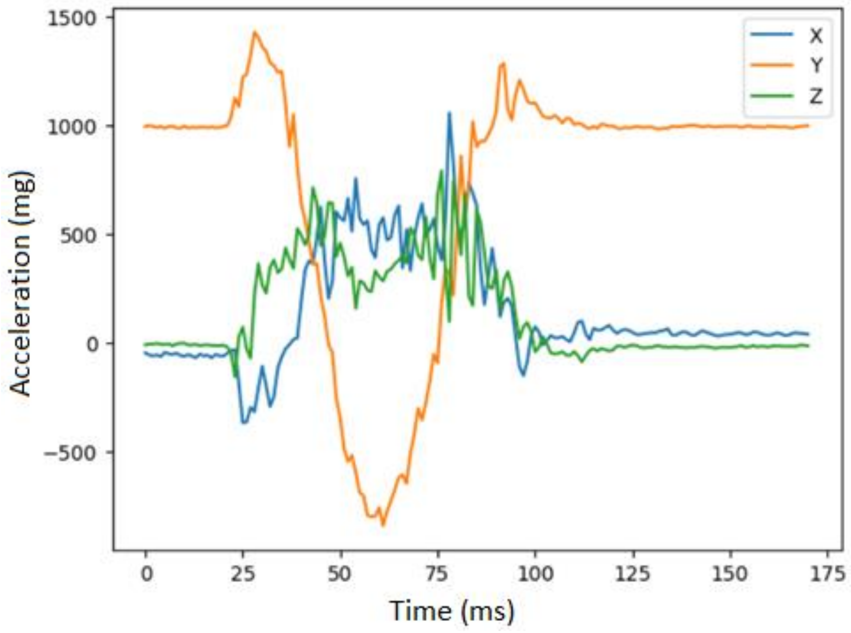


Figure 66: AE Accelerometer

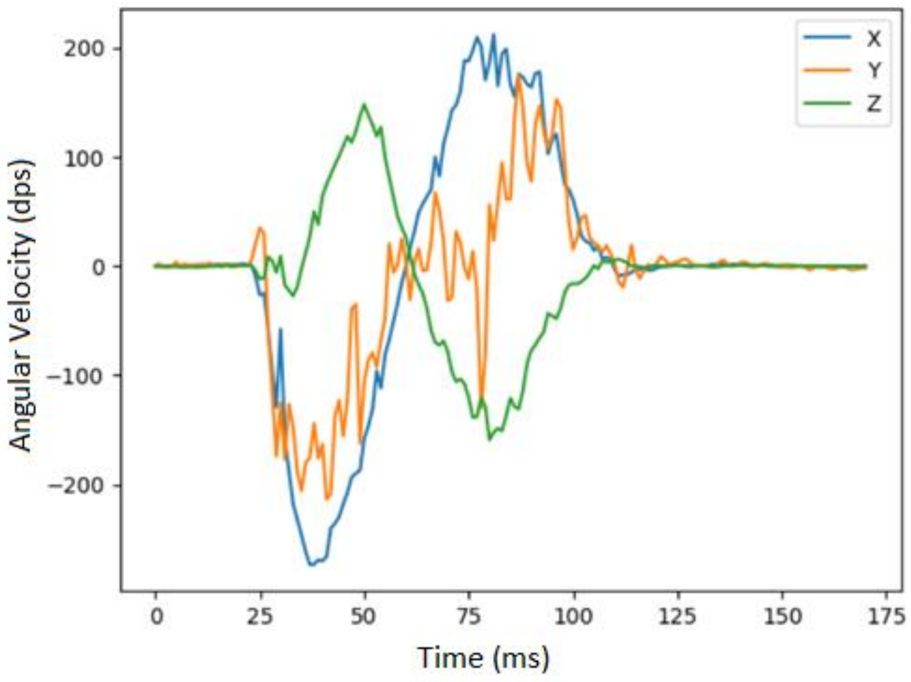


Figure 67: AE Gyroscope

7.5.2 ODR

The first thing that was tried was changing the MLC and sensors ODRs. Whereas the window length and feature selection were kept constant at 13 and mean, respectively. In the first configuration, both MLC and sensors ODR was 30 Hz. In the second configuration, MLC ODR was 30 Hz and the sensors ODR was 120 Hz. In the third configuration, both MLC and sensors ODR was 60 Hz. And, in the fourth and final configuration, MLC ODR was 60 Hz and the sensors ODR was 120 Hz. The outcomes of these four configurations are reported in the table below.

Configuration	**Result**
30-30	Output does not seem to be changing at all with the arm movement.
30-120	Generally false at rest, but goes to true whenever hand is

	raised, doesn't matter in which direction.
60-60	Output doesn't seem to be changing at all.
60-120	Output is changing too fast probably because of the high frequencies of MLC and sensors. It is difficult to see under what conditions the output is changing.

Table 16: Outcome of trying different combination of ODR

From the above table, it can be seen that changing the ODR did not seem to have any meaningful impact on the problem. Seeing that, it was noted that perhaps the best ODR combination is 30 Hz for the MLC and 60 Hz for the sensors.

7.5.3 Idle state

Observing no progress in trying multiple ODR, it was suggested that the problem could be defined better by defining a third state as well in addition to the AD and AE states. This third state was the idle state in which the arm is at rest and not moving at all.

Now, the configuration was tried with this additional idle state and with window length 13, feature mean, MLC ODR 30 Hz, and sensors ODR 60 Hz. The result was that it was an improvement to the previous similar configuration without the idle state, but still it was not even near perfect and was very inconsistent as well with its output, especially with the AD and AE cases.

7.5.4 More data

To target the inconsistencies between the AD and AE states, it was decided to provide it with more training data for the AD and AE cases. Initially, there were only 3 measurement files for each case, which was then increased to 11 files each by collecting more data. The resulting configuration gave the following outcome:

Idle	**AD**	**AE**
Stays at 4 (true)	Changes to 8 (false) when hand goes up and then to 4 (true) when the hand is down back.	Changes to 8 (false) when hand goes up and then to 4 (true) when the hand is down back.

Table 17: Performance of MLC when provided with more data

7.5.5 Window length

The next thing that was tried to get better results was to play around with different window lengths. In these configurations, the MLC ODR, sensors ODR and the feature selection remained same (30 Hz, 60 Hz and mean, respectively). The window lengths that were tried are 52, 91, 104, and 110.

Result:

Window Length	**Movement**	**Idle**	**AD**	**AE**
52		Stays at 0 (idle). Though changes to 4 (true) and 8 (false) with slight arm movement/rotation.	Stays at 0 (idle)	Stays at 0 (idle)
91		Stays at 4 (true)	Stays at 4 (true)	Stays at 4 (true)
104		Stays at 4 (true)	Stays at 4 (true)	Stays at 4 (true)
110		Stays at 0 (idle)	Stays at 0 (idle)	Stays at 0 (idle)

Table 18: Performance of different MLCs with different window lengths

There was still no good outcome even with trying with multiple window lengths. The best window length considered after this experiment was still 13.

7.5.6 Feature Selection

The next thing that was tried was different features other than just mean. The features that were tried are peak-to-peak, variance, energy, recursive, minimum, and maximum. In these configurations, the MLC ODR was 30 Hz, sensors ODR was 60 Hz, and the window length was 13.

Result:

Feature | **Movement**	**Idle**	**AD**	**AE**
Mean	Stays at 4 (true)	Changes to 8 (false) when hand goes up and then to 4 (true) when the hand is down back.	Changes to 8 (false) when hand goes up and then to 4 (true) when the hand is down back.
Peak-to-peak	Stays at 4 (true)	Changes to 8 (false) when arm is going up and then to 4 (true) when the arm stays up. Changes to 8 (false) again when the arm is going down and then to 4 (true) when the arm is completely down.	Changes to 8 (false) when arm is going up and then to 4 (true) when the arm stays up. Changes to 8 (false) again when the arm is going down and then to 4 (true) when the arm is completely down.
Variance	Not stable. Continuously changing between 0 (idle), 4 (true) and 8 (false)	Not stable. Continuously changing between 4 (true) and 8 (false)	Not stable. Continuously changing between 4 (true) and 8 (false)

Energy	Not stable. Continuously changing between 0 (idle), 4 (true) and 8 (false)	Not stable. Continuously changing between 4 (true) and 8 (false). But stays at 8 (false) when the arm is completely up.	Not stable. Continuously changing between 4 (true) and 8 (false). But stays at 8 (false) when the arm is completely up.
Recursive	Not stable. Continuously changing between 4 (true) and 8 (false)	Not stable. Continuously changing between 4 (true) and 8 (false)	Not stable. Continuously changing between 4 (true) and 8 (false)
Minimum	Stays at 4 (true)	Changes to 0 (idle) and then to 4 (true) when the arm is back down.	Changes to 0 (idle) and then to 4 (true) when the arm is back down.
Maximum	Stays at 4 (true)	Changes to 0 (idle) and then to 4 (true) when the arm is back down.	Changes to 0 (idle) and then to 4 (true) when the arm is back down.

Table 19: Performance of different MLCs with different features selection

It can be seen in the table above that even trying different features did not have a positive outcome on the output.

In summary, we tried

- Different MLC and sensors ODR
- Additional Idle state
- More training data
- Different window lengths
- Different features

But none of it had a positive outcome. From here on, there was not much that can be tried within the box kit or ST tools. But there were some external things that could be tried, for example, changing the position of sensor placement.

7.6 Sensor on the Wrist

As explained in the Using Sensor on the Wrist instead of on the Arm section, when nothing else worked, there was still this last thing that could be tried. It was to place the sensor on the wrist instead of on the biceps. By placing the sensor on the wrist, it was easier to differentiate between the AD and AE cases as compared to the case when the sensor was placed on the biceps, as can also be seen in the plots of Accelerometer and Gyroscope. Like the previous section, different configurations were tried and tested with different ML decision parameters.

7.6.1 Plots

Before presenting any results, below are the plots of accelerometer and gyroscope data when AD and AE movements were performed by placing the sensor on the right wrist.

7.6.1.1 AD

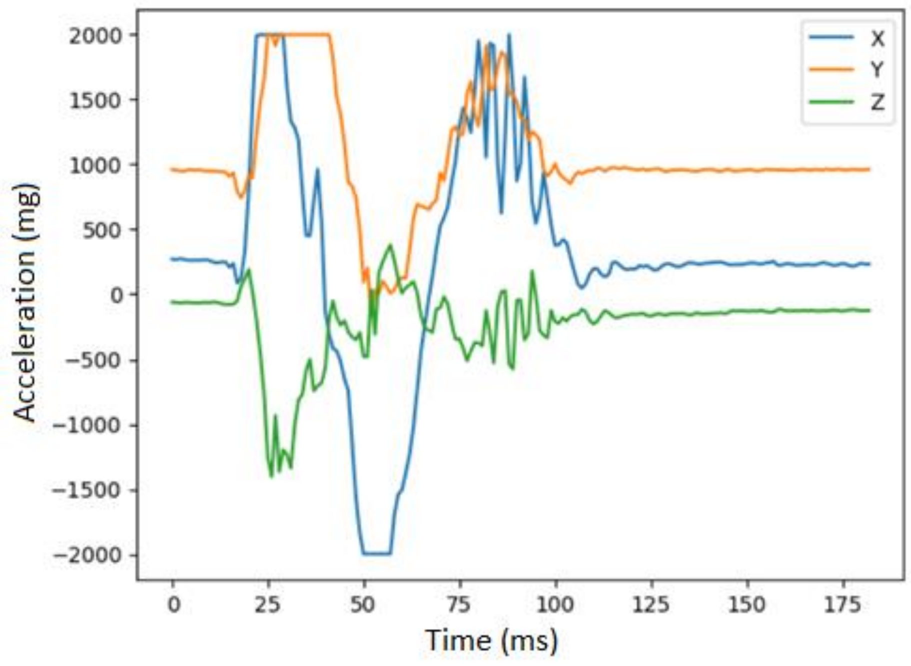


Figure 68: AD Accelerometer Wrist

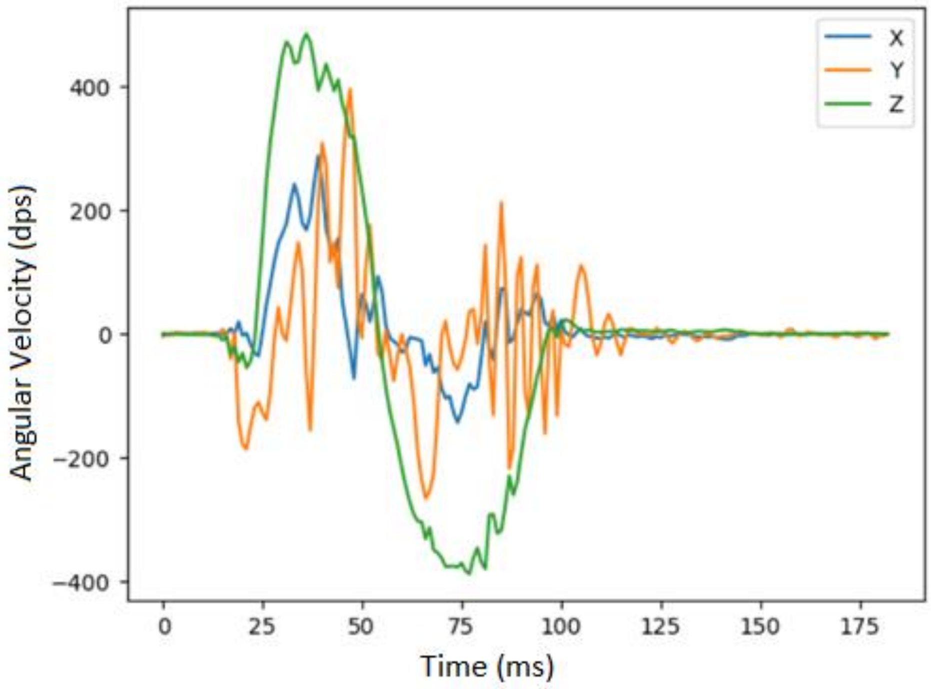


Figure 69: AD Gyroscope Wrist

7.6.1.2 AE

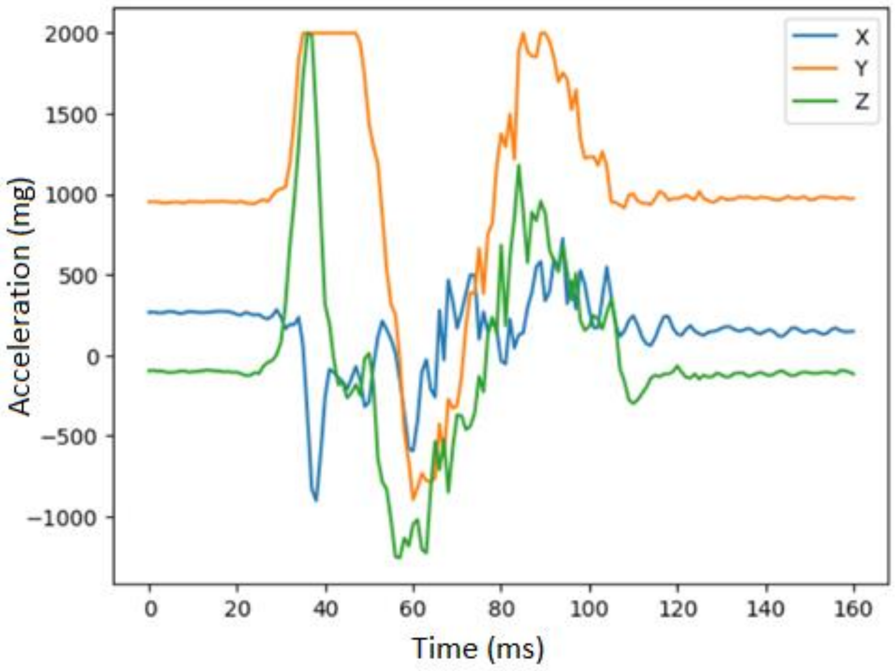


Figure 70: AE Accelerometer Wrist

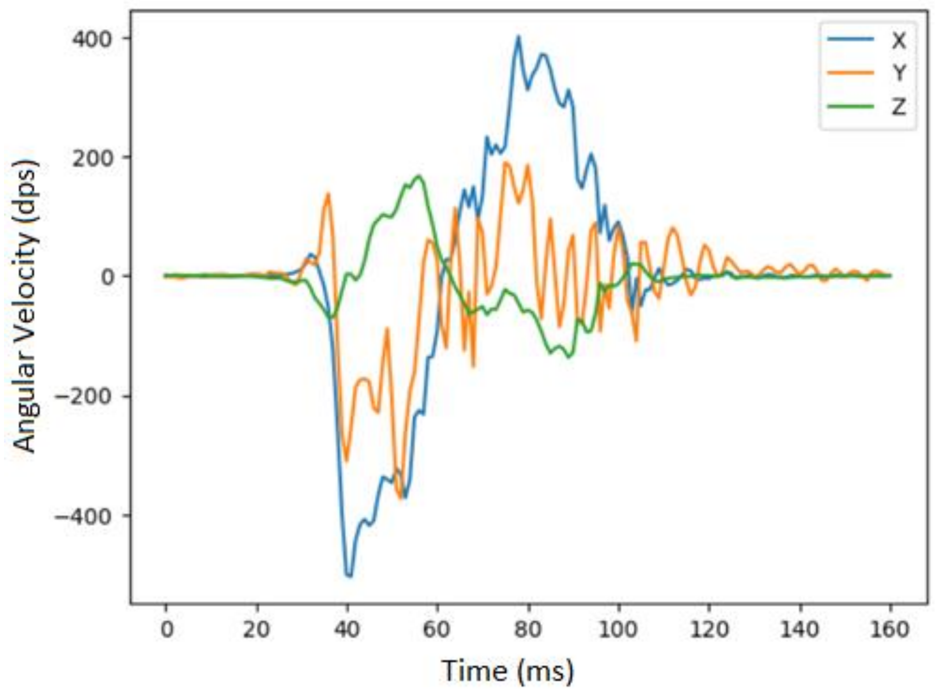


Figure 71: AE Gyroscope Wrist

7.6.2 Window length

First, different window lengths were tried by keeping the other parameters constant. The MLC ODR was set at 30 Hz, sensors ODR at 60 Hz, and the feature was mean. 3 different window lengths were tried in this case: 13, 16, and 80.

Result:

Window Length	**Movement**	**Idle**	**AD**	**AE**
13		Randomly changing between 4 (true) and 8 (false) by slight hand rotation.	Changes to 8 (false) when hand goes up and stays there for a while, but back to 4 (true) as soon as the hand comes back down.	Changes to 8 (false) when hand goes up and stays there for a while, but back to 4 (true) as soon as the hand comes back down.

16	Randomly changing between 0 (idle), 4 (true) and 8 (false) by slight hand rotation.	Randomly changing between 4 (true) and 8 (false).	Randomly changing between 0 (idle), 4 (true) and 8 (false).
80	Stuck at 8 (false)	Stuck at 8 (false)	Stuck at 8 (false)

Table 20: Performance of different MLCs with different window lengths when the sensor is on the wrist

7.6.3 Single reading

When it did not seem to be working, it was decided to set the window length in a way such that the whole data file would correspond to one single reading in the training. The configuration that was tried was with MLC ODR 30 Hz, sensors ODR 60 Hz and the window length 83. Since the sensors ODR is twice the MLC ODR, it would be seeing only half the number of actual readings in the file. As there were around 166 readings in the file, the window length was chosen as 83. Before trying this configuration, it was noted from the accelerometer and the gyroscope graphs that the peak-to-peak would be the best feature to capture the complete essence of the data. So, the feature that was selected was peak-to-peak.

On the next iteration, the idle case was omitted from the experiment as it seemed to be misguiding the training process, and it was not even important to predict the idle case, it was just a nice to have feature if it could have been incorporated. So, it could be dropped if wanted.

Result:

Configuration	**Movement**	**Idle**	**AD**	**AE**
With Idle		Stays at 8 (false)	Gives 4 (true) almost every time.	Gives 8 (false) almost every time.
Without Idle		N/A	Gives 0 (true) almost every time.	Gives 4 (false) almost every time.

Table 21: Performance comparison of two MLCs with and without idle case when the sensor is on the wrist

Finally, it gave some meaningful results as can be seen in the table that it was able to predict both AD and AE movements correctly almost every time.

7.6.4 Single file

Now that we had a working algorithm which was able to predict the correct arm movements. There was still one more thing that could be tried to see if it improves the algorithm. In the last algorithm, it was trained by using multiple measurement files. What if we train the algorithm by using single data file, one for AD and another one for AE. So, we began by using just a single file for the training. Different configurations were tried that are listed below in the table. In most of these configurations, the window length was set such that one file had just one reading.

The configurations are written in aa-bb-cc-dd format in the table below where aa is the MLC frequency, bb is the sensors frequency, cc is the window length, and dd is the feature used.

Result:

Feature	**Movement**	**Idle**	**AD**	**AE**
30-60-13-Mean		Remains at 0 (idle) at a specific hand position, but changes to both 4 (true) and 8 (false) at other positions with slight hand rotations.	Gives 4 (true) when the arm is raised in a specific hand position, otherwise gives 8 (false) as well.	Gives 8 (false) when the arm is raised in a specific hand position, otherwise gives 4 (true) as well.
60-60-13-Mean		Remains at 0 (idle) at a specific hand position, but changes to 8 (false) at other positions with slight hand rotations.	Gives both 4 (true) and 8 (false). Very uncertain.	Mostly gives 8 (false), but sometimes gives 4 (true) as well.
30-60-83-Mean (also single reading)		Stays at 8 (false).	Stays at 8 (false).	Stays at 8 (false).

30-60-83-Peak (also single reading)	Stays at 8 (false).	Gives 4 (true) every time.	Gives 8 (false) almost every time (once gave 4 (true)).
30-60-83-Peak Wo Idle (also single reading)	N/A	Gives 0 (true) every time.	Gives 4 (false) almost every time. You just have to keep your wrist strict.
60-60-168-Mean (also single reading)	Stays at 8 (false)	Stays at 8 (false)	Stays at 8 (false)
60-60-168-Peak (also single reading)	Stays at 8 (false)	Gives 4 (true) but not every time.	Stays at 8 (false) almost every time.
30-30-168-Peak (also single reading)	Stays at 8 (false)	Gives 4 (true) almost every time. Though the response is very delayed (perhaps because of low frequency?)	Gives 8 (false) almost every time (once gave 4 (true)). Its response is also delayed.

Table 22: Performance comparison of different MLCs with different configurations when the sensor is on the wrist

From the table, it can be seen clearly that peak-to-peak was the best feature to use when using window length such that there was just one reading in one measurement file. Then, the configuration in which sensors ODR is twice the MLC ODR was better than the configurations in which both ODRs are same. Then, with and without idle case did not have a much difference with respect to predicting AD and AE movements. However, with idle case, it was unable to predict the idle condition, rather it predicted it as either AD or AE conditions.

8 Conclusion and Future Work

To conclude, setting up a decision-tree based Machine Learning for real-life human motion tracking based on STMicroelectronics' STEVAL-MKBOXPRO (SensorTile.Box PRO) kit was not an easy task as one could have expected initially. The first step in getting started was to explore all the hardware and software tools provided by STM that could come handy in setting up this application, including SensorTile.Box PRO kit and several software such as STM32CubeIDE, Unico-GUI, etc., and STBLESensor mobile application. Learning those tools was not that difficult at all as STM provides detailed datasheets and application notes to get familiar with these tools.

After this phase, all the focus of the internship was towards the generation and use of the MLC (Machine Learning Core). Getting to know the MLC generation process was easy and straightforward. This generation process also allows you to choose some ML training decision parameters. The first approach was to use the Accelerometer and the Gyroscope data while the sensor was mounted on the right arm on the biceps. But it was not any success when building the decision-tree based on those data. Forcing us to switch to other alternatives, leading to the use of Euler angles as the training data.

Honestly speaking, using Euler angles for an application like this in which there are certain arm movements in different directions makes more sense than using the accelerometer and gyroscope data standalone. Because these arm movements are more well represented with Euler angles than accelerometer/gyroscope data. But again unfortunately, this step was not fruitful in setting up the well-running decision tree to predict the arm movements.

I think that the problem behind this lies in the reason that Euler angles are not available to us directly from the sensors. They are the result of transformation from the Quaternions which itself is not available directly to us from the sensors. We obtain the Quaternions by using the sensor-fusion of accelerometer, gyroscope and magnetometer data. So, when we train the decision tree on the Euler angles, we provide it as external data as compared to providing it as internal data. When we were using accelerometer/gyroscope data in the previous step, we could tell the software that this is the accelerometer data and this is the gyroscope data, so when we run the resulting MLC, it knows how to make predictions on which sensors data. But in this case where Euler angles are given as external data, there is no way to tell during the training process that this data is the Euler angles data. It just treats it as external data without any label. I think that is the root problem for this not working, which seems to be a flaw in the STM's design. This issue has been communicated with the STM and together by discussing it with them, we should be able to identify the root cause of this problem.

Nevertheless, this step made us familiar with the tunable training parameters that we decided to try also on the accelerometer/gyroscope data. Tuning those parameters gave some good signs of progress, so we knew that we were onto something. The limitation was that the resulting algorithm was not successfully able to distinguish between the AD and AE movements when the sensor was placed on the biceps, which is true as there was not much of a difference between the two movements from the sensors' point of view as it was placed very close to the rotation axis. Therefore, when the sensor was placed instead on the wrist, it gave sensors a much better understanding of what was really happening. Resulting in better performance (almost perfect) in distinguishing the AD and the AE movements.

From the point of view of Turingsense and their applications, typically they would have wanted their sensors to be placed on the biceps. As the point of this internship was to explore the functionalities that STM has to offer with its SensorTile.Box PRO kit and the possibility of using this box kit in Turingsense's applications, we got to know that this box kit has some potential. And if Turingsense wants to use it, they would have to most likely place it on the wrist instead of the biceps for it to work, or to use a network of these sensors in which more than one sensor are placed at different positions, but it would only make it more complex. But using this box kit has some limitations too, such as, it does not give freedom to use any machine learning algorithm, we are bound to use only decision-tree algorithm. Then we are also limited by the data types that we can train the decision tree on, as explained earlier, which would not have been the case if they had their own hardware with their own firmware. Then they would be free to program it any way they like, and I believe that they can get good results with this approach. But, writing your own hardware/firmware from scratch is not easy at all, and it requires a team of experienced professionals to create something like this. So, both approaches have their own benefits and drawbacks. In the end, there is no free lunch.

For future work, there are two ways to look at it. If they want to continue with STMicroelectronics, then it is probably better to work closely with STM engineers to better understand their limitations and to better communicate their requirements. With this collaboration, it is still possible to use the SensorTile.Box PRO kit in their applications by resolving the issues that were discovered during this internship. Another way to look at the future work is the possibility of finding other alternatives to SensorTile.Box PRO kit that might be already available in the market and then accessing their working and performance for Turingsense’s applications.

9 Bibliography

1. Lu, Tung-Wu, and Chu-Fen Chang. "Biomechanics of human movement and its clinical applications." The Kaohsiung journal of medical sciences 28.2 (2012): S13-S25.
2. Bortolini, Marco, et al. "Motion Analysis System (MAS) for production and ergonomics assessment in the manufacturing processes." Computers & Industrial Engineering 139 (2020): 105485.
3. Kavanagh, Justin J., and Hylton B. Menz. "Accelerometry: a technique for quantifying movement patterns during walking." Gait & posture 28.1 (2008): 1-15.
4. Field, Matthew, et al. "Human motion capture sensors and analysis in robotics." Industrial Robot: An International Journal 38.2 (2011): 163-171.
5. Petrosyan, Tigran, Arayik Dunoyan, and Hasmik Mkrtchyan. "Application of motion capture systems in ergonomic analysis." Armenian journal of special education 4.2 (2020): 107-117.
6. Kulić, Dana, et al. "Anthropomorphic movement analysis and synthesis: A survey of methods and applications." IEEE Transactions on Robotics 32.4 (2016): 776-795.
7. Ortega, Basilio Pueo, and José M. Jiménez Olmedo. "Application of motion capture technology for sport performance analysis." Retos: nuevas tendencias en educación física, deporte y recreación 32 (2017): 241-247.
8. Hsieh, Jun-Wei, et al. "Video-based human movement analysis and its application to surveillance systems." IEEE Transactions on Multimedia 10.3 (2008): 372-384.
9. Zago, Matteo, Ana Francisca Rozin Kleiner, and Peter Andreas Federolf. "Machine learning approaches to human movement analysis." Frontiers in bioengineering and biotechnology 8 (2021): 638793.
10. Iosa, Marco, et al. "Wearable inertial sensors for human movement analysis." Expert review of medical devices 13.7 (2016): 641-659.
11. Di Nardo, Francesco, and Sandro Fioretti, eds. "Recent Advances in Motion Analysis." (2021).
12. Meng, Zhaozong, et al. "Recent progress in sensing and computing techniques for human activity recognition and motion analysis." Electronics 9.9 (2020): 1357.

13. Worsey, Matthew TO, et al. "A systematic review of performance analysis in rowing using inertial sensors." Electronics 8.11 (2019): 1304.
14. Kim, Byong Hun, et al. "Measurement of ankle joint movements using IMUs during running." Sensors 21.12 (2021): 4240.
15. Pau, Massimiliano, et al. "Smoothness of gait in healthy and cognitively impaired individuals: a study on Italian elderly using wearable inertial sensor." Sensors 20.12 (2020): 3577.
16. Filippeschi, Alessandro, et al. "Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion." Sensors 17.6 (2017): 1257.
17. Wong, Wai Yin, Man Sang Wong, and Kam Ho Lo. "Clinical applications of sensors for human posture and movement analysis: a review." Prosthetics and orthotics international 31.1 (2007): 62-75.
18. Duffy, Vincent G., ed. Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Anthropometry, Human Behavior, and Communication: 13th International Conference, DHM 2022, Held as Part of the 24th HCI International Conference, HCII 2022, Virtual Event, June 26–July 1, 2022, Proceedings, Part I. Vol. 13319. Springer Nature, 2022.
19. Stanescu, Monica, and Marius Stoicescu. "New Competencies for Physical Education Teachers: Software for Movement Analysis." The International Scientific Conference eLearning and Software for Education. Vol. 1. " Carol I" National Defence University, 2012.
20. Dunn, Fletcher. 3D math primer for graphics and game development. CRC Press, 2011.
21. Henderson, D. M. Shuttle Program. Euler angles, quaternions, and transformation matrices working relationships. No. NASA-TM-74839. 1977.