Varia:

Session: Starts with login and ends with logout, drawback: each task might only be done once per session

TDMS logging -> download to local folder (~1sec) -> copy to tmp folder (AppData/Local../ETH/ETH\_MIKE.. shortcut available in front end directory) -> zip compression (after this it is crash safe) -> upload to webdav server (polybox)

Config Files Default values can overwritten by specifying them in json file

Assessments:

* Range of Motion: 1. Active Trials: Press Start to start trial -> **move to start position** -> let patient move finger with gravity compensation  
   -> patient presses finish button -> repeat trial until number of trials reached  
   2. Passive Trials: Press Start to start trial -> **move to start position** -> let therapist move finger with gravity compensation  
   -> patient presses finish button -> repeat trial until number of trials reached  
   3. Movement: Display message that finger will be moved -> Press Start -> display relax message -> compute movement  
   -> **move to start position** -> **perform movement** until completion -> display results
* Force: 0. Display message that external part must be fastened

1. Flexion let them press ok -> press start to start trial  
 -> count down -> measure & display force -> repeat trial until finished  
 2. Extension let them press ok -> press start to start trial  
 -> count down -> measure & display force -> repeat trial until finished

3. Display message that external part must be removed -> display result

* Position Matching: Press start to start trial -> **move to starting position** & indicate start position -> **move to a target** -> let patient indicate target position -> press validate  
   -> repeat trial until finished -> display results
* Fast reaching: Press start to start trial -> **move to a start position** -> **display target on screen** -> let user move to actively to target -> stop after ? time ? -> repeat trial

(strangly it moves towards the middle when the start position is on the outside -> gravity compensation?) -> display results

* Target Following: 1. Slow: Press start to start trial -> **Move to start position** -> follow displayed needle -> skip option -> repeat trial until finished

2. Fast: Press start to start trial -> **Move to start position** -> follow displayed needle -> skip option -> repeat trial until finished

When finished or skipped -> display result

Exercises

* Passive Matching: Training: Press start to start training trial -> **move to start position** -> **move to** one of the multiple **targets** -> user confirms   
   -> repeat training trial until all targets are shown once.  
   Exercise: Press Button to start -> **move to start position** -> **move to** one of the multiple **targets** -> user has to select which target it is  
   -> Display feedback -> repeat exercise trial
* Active Matching: Training: Press start to start training trial -> **move to start position** -> show a target on the screen   
   -> user has to move actively to the location with visual feedback -> confirm by button press -> repeat training trial  
   Exercise: Same as training without visual feedback
* Teach&Reproduce: Training: Press start to start training trial -> **move to start position, then to a target, pause there and move back to start** -> user has to move actively to the teached target with visual feedback -> confirm by button press -> repeat training trial  
   Exercise: Same as training without visual feedback

**Back End – Interface to motor, sensor, stop button**

**Front End – Interface to user**

How does a task look like (start, target, )

High speed logging

Run controller, which controller is needed

Where is the middle of ROM

Calculate min and max range of motion during passive ROM task

Does not know if it is practice run

User registration

Should we start a task?

Which task should we run?

How many repetitions should we do?

Left or right hand?

Is this a practice run?

For Exercises: Where is the target.

**Report current state**

RobotState: uint Counter  
float Time  
float Position  
float StartingPosition  
float TargetPosition  
float Force  
byte Trial  
enum RomState  
bool TargetState  
bool Finished  
bool Flexion

ControlData: bool EmergencyStop  
 bool Start  
 bool Restart  
 bool FrontendStarted  
 bool Close  
or  
PatientData: bool LeftHand  
 enum AssessmentMode  
 string SubjectNr  
 string DateTime  
 string PhaseTrialCount  
 string StudyName

**Backend - Task State – State Diagram**

**if** RestartSignal

**if** StartAssessment

StartTask

**if** start position reached

**if** skip phase

Start Trial

Close Log file if: -Restart Signal  
 -StopRequested  
 -AssessmentFinished

**if** RestartSignal or  
**if** AssesmentFinished

Resetting

**if** RestartSignal or  
**if** SkipToFinish

**if** RestartSignal

Unblock TCP thread

SkipSignal = false  
TargetState = false

**if** TrialFinished && !AssessmentFinished

Show Target

RestartSignal = false  
AssessmentFinished = false

**if** RestartSignal && !LogFileOpen

selectedHand = false

Wait For Restart

Hand Choice

StandardMikeUserInterface

Skip -> SkipSignal  
or  
ControlData: bool EmergencyStop -> StopSignal  
 bool Start -> StartSignal (starts a single trial)  
 bool Restart ->RestartSignal  
 bool FrontendStarted (only for logging)  
 bool Close (Shutdown backend)  
 float TargetPosition  
or  
PatientData: bool LeftHand (which hand)  
 enum AssessmentMode (which assesment)  
 string SubjectNr  
 string DateTime  
 string PhaseTrialCount (number of total trials)  
 string StudyName

Needs to receive “PatientData” first. This chooses the assessment to perform. And after setting “SelectedHand”, the state machine gets into a state ready to start the assessment.  
After receiving a controlData message with the StartSignal=true, one trial of the assessment is performed.

TCP Listener Loop

**TCP communication**

Load configurations from json file:  
number of trials per task, …

Config.Get\* ()

Config

TaskState

AssessmentMode

Hand Selection

State machine

Safety

Store all data without risk of loosing anything

Main Loop

UDP sending loop

Process()

Process()

BlockingCollection msgQueue  
ConcurrentQueue pendingsCallbacks

sendMessage()  
sendControlData()  
sendPatientData()

DownloadBackendLogAsync()

BackendClient

RobotState: uint Counter  
float Time  
float Position  
float StartingPosition  
float TargetPosition  
float Force  
byte Trial  
enum RomState  
bool TargetState  
bool Finished  
bool Flexion

Connect()

Patient CurrentPatientMetadata

ConcurrentQueue ResultStoreQueue

CreateAssessment()  
CreateSession()  
ChangeActiveHand()  
RegisterPatient()  
PatientExists()  
TryGetLastSessionForPatient()  
LoadAllResultsForCurrentPatient()  
…….

WebDavUploader

Load database from file db.db

Database

TrySetLanguage()

Provides label in chosen language via  
Strings.<enum\_lable\_name>.Get()

Static class

Read from all files in StreamAssests/Localization

LocalizationManager()

LocalizationManager

WriteLogEntry()

OnTick()

PatientData CurrentPatient  
RobotState CurrentRobotState  
BackendClient backendClient  
DataReceiver robotStateReceiver  
WebDavUploader uploader

OnMainButtonClick()  
EmergencyStop()  
FinalizePatientSetup()  
TransitionTo\*()

AbstractMikeUserInterface

Log to csv file in folder structure

RobotState currentRobotState  
IDataProcessor currentDataProcessor

Runs on separate thread.

ReceiveData()

DataReceiver

TDMS logging

**UDP stream**

**Back End**

**Front End**