MECHTRON 4TB6: System Design

LifeLine

Group 30
Emily Crowe, crowee
Arthur Faron, farona
Danushka Fernando, fernad12
Yerin Thevarajah, thevaryn
Phillip Truong, truonp1

January 4, 2021

Revision History

Revision	Date	Author(s)	Description
0	04/01/2021	EC, AF, DF, YT, PT	Document Created

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of LifeLine is to provide first-aiders with a device to quantitatively monitor a casualty's vital signs to assist in first aid situations.The following document will provide details of the system design of the LifeLine device. The document will serve as a record of the design variables and considerations.

Decomposing a system into modules is a commonly accepted approach to development. A module is a work assignment for a developer or development team [1]. We advocate a decomposition based on the principle of information hiding [2]. This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by Parnas et al. [1], as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is used in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

1.2 Scope

The project, named LifeLine, intends to help a first-aider monitor a casualty's vital signs to keep them stable until emergency medical services (EMS) arrives. This project also intends to assist a first-aider by retrieving vital signs quantitatively. These results can then guide the first-aider to adapt and improve their treatment plan. The device is able to record and save the vital sign data which can later be exported for more advanced analysis by medical professionals.

LifeLine is a product that carries out four (4) primary functions: quantitative measurement of primary vital signs through sensor readings, logging the primary vitals data, displaying the primary vitals data, and exporting the primary vitals data. The primary vital signs this project will be measuring are body temperature, blood pressure and heart rate.

The device will instruct the user where to place the different sensors for each vital sign in order to measure them safely and efficiently. While these procedures are taking place, the device will present the results on a display. The first-aider can use this to further their treatment and continue monitoring the vital signs until medical assistance arrives. The product will then log the measurements and export this data into an accessible file to be used for medical records and for further examination by medical professionals.

2 Variables

2.1 Monitored Variables

Table 2: Monitored variables

Name	Description	Type	Units
m_BT_casualty	Pre-processed Body Temperature measurement recorded by IR temperature sensor	Digital	degrees Celsius
m_BP_casualty	Pre-processed Blood Pressure measurement recorded by MAX32664 sensor	Digital	mmHg
m_HR_casualty	Pre-processed Heart Rate measurement recorded by MAX32264 sensor	Digital	BPM

2.2 Controlled Variables

Table 3: Controlled Variables

Name	Description	Type	Units
c_screenDisplay	Displays data and menus requested by the user on the LCD screen of the device.	Various. Depends on what is being displayed.	N/A

3 Constants

Table 4: Constant Variables

Variable name	Description	Type of Variable	Units
k_BT_min	Acceptable minimum body temperature	Constant	Degree Celsius
k_BT_max	Acceptable maximum body temperature	Constant	Degree Celsius
k_BP_min	Acceptable minimum blood pressure	Constant	mmHg
k_BP_max	Acceptable maximum blood pressure	Constant	mmHg
k_HR_min	Acceptable minimum heart rate	Constant	BPM
k_HR_max	Acceptable maximum heart rate	Constant	BPM

4 Module Components

Module 1: Body Temperature Measurement
Module 2: Blood Pressure Measurement
Module 3: Heart Rate Measurement
Module 4: Casualty Module Interface Specification (MIS)
Module 5: Body Temperature Data Processing
Module 6: Blood Pressure Data Processing
Module 7: Heart Rate Data Processing
Module 8: Data Storage
Module 9: Data Exporting
Module 10: User Interface- On/Off
Module 11: User Interface- Display Values
Module 12: User Interface- Display System Status
Module 13: User Interface- System Settings

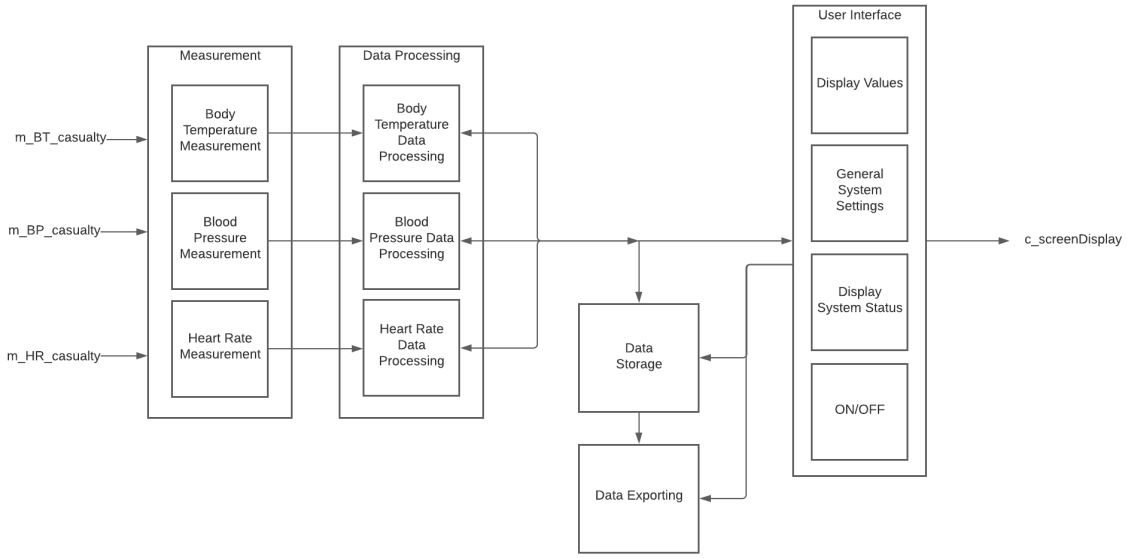

Figure 1: Module overview
Outlines the modules including in the LifeLine system and how they interact with each other.

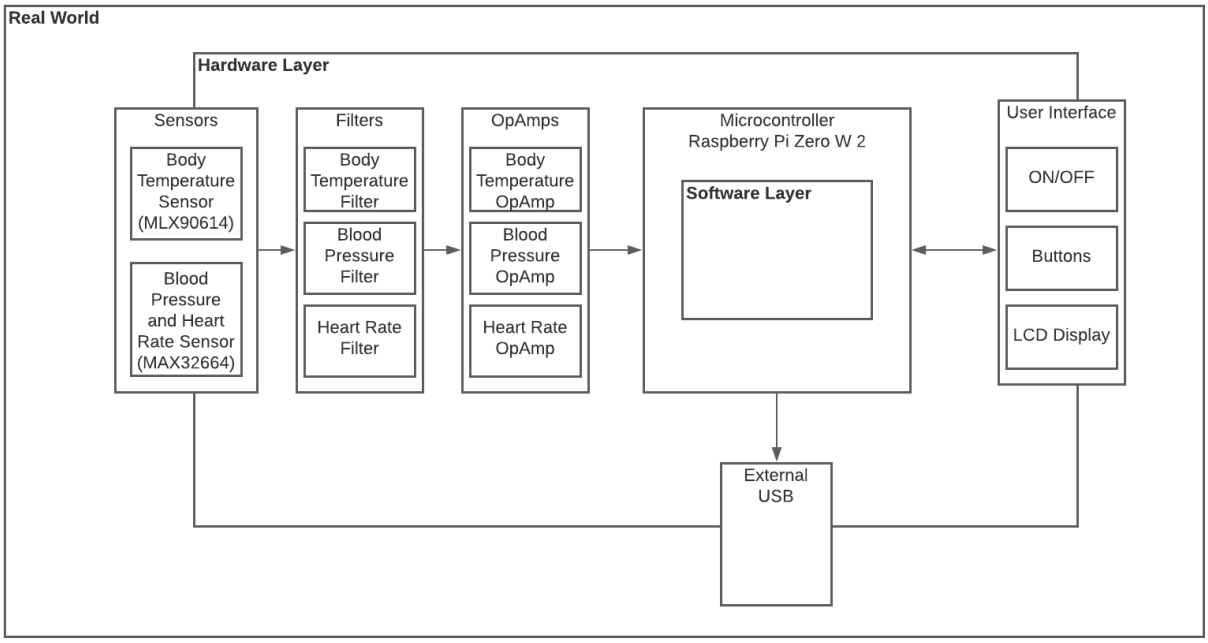

Figure 2: Hardware overview
Outlines the main hardware components used in the LifeLine device.

4.1 Body Temperature Measurement

M1: Body Temperature Measurement

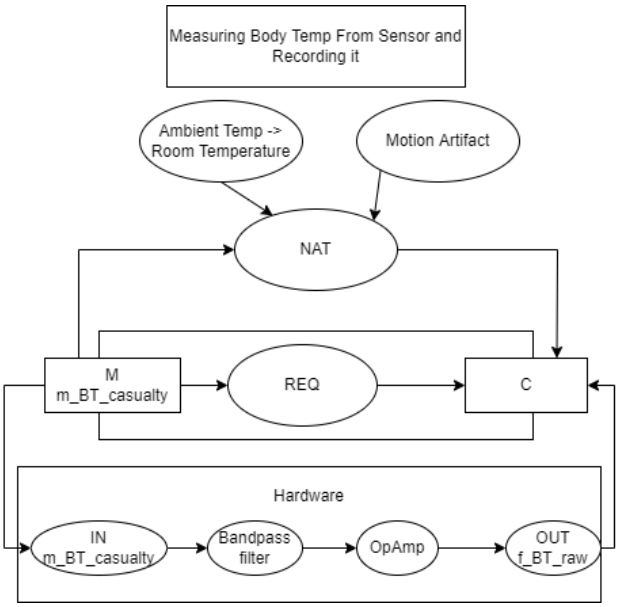

Figure 3: Body Temperature Measurement Mapping
NAT: Ambient temperature can affect the performance of the body temperature component since the MLX90614 IR thermometer operates best at room temperature. Motion artifact caused by any movements of the casualty has the potential to impact the body temperature measurement.

4.1.1 Services

The body temperature measurement module uses the MLX90614 IR thermometer (sensor) to measure the body temperature of the casualty.

4.1.2 Secrets

The body temperature measurement module handles raw data input from the body temperature sensor and passes it to the 'Body Temperature Data Processing' (M5). The body temperature module performs analog filtering of the input signal from the sensor. This raw temperature value is sent to the 'Body Temperature Data Processing' (M5) and 'Data Storage' (M8) modules.

4.1.3 Inputs and Outputs

Sensor is placed on the casualty's body for body temperature measurement.

Table 5: Body Temperature Measurement Input Variables

Input Variable	**Description**	**Data Type**	**From Module**
m_BT_casualty	Casualty's temperature as received by MLX90614 IR thermometer	Digital	N/A

Table 6: Body Temperature Measurement Output Variables

Output Variable	**Description**	**Data Type**	**To Module**
f_BT_raw	Raw body temperature data processed by bandpass filter and operational amplifier	Double	Body Temperature Data Processing (M5), Data Storage (M8)

4.1.4 Software Components

None.

4.1.5 Hardware Components

- MLX90614 IR thermometer
- Bandpass filter
- Operational amplifier

4.1.6 Timing Constraints

There must be no more than 1 second to receive measurement from sensor to display (NFR4 in System Requirements). The MLX90614 IR thermometer takes 0.25 seconds to acquire data from start-up [3].

4.2 Blood Pressure Measurement

M2: Blood Pressure Measurement

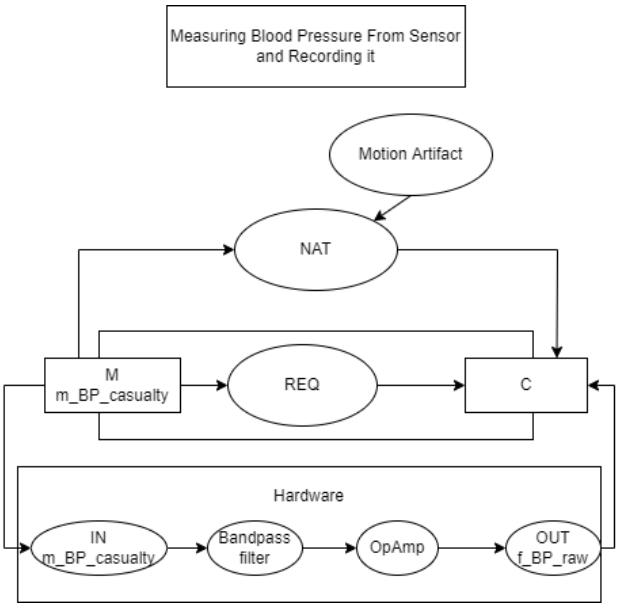

Figure 4: Blood Pressure Measurement Mapping
NAT: Motion artifact caused by any movements of the casualty has the potential to impact the blood pressure measurement.

4.2.1 Services

The blood pressure measurement module uses the Pulse Express Pulse-Ox & Heart Rate Sensor with MAX32664 (sensor) to measure the blood pressure trending of the casualty.

4.2.2 Secrets

The blood pressure measurement module handles raw data input from the blood pressure sensor and passes it to 'Blood Pressure Data Processing' (M6). The blood pressure measurement module performs analog filtering of the input signal from the sensor. This raw temperature value is sent to the 'Blood Pressure Data Processing' (M6) and 'Data Storage' (M8) modules.

4.2.3 Input

Table 7: Blood Pressure Measurement Input Variables

Input Variable	**Description**	**Data Type**	**From Module**
m_BP_casualty	Casualty's blood pressure as received by MAX32664	Analog	N/A

4.2.4 Output

Table 8: Blood Pressure Measurement Output Variables

Output Variable	Description	Data Type	To Module
f_BP_raw	Raw blood pressure data processed by bandpass filter and operational amplifier	Double	Blood Pressure Data Processing (M6), Data Storage (M8)

4.2.5 Software Components

None.

4.2.6 Hardware Components

- MAX32664 Optical Biometric Hub
- bandpass filter
- operational amplifier

4.2.7 Timing Constraints

There must be no more than 1 second to receive measurement from sensor to display (NFR4 in System Requirements). The MAX32664 has a clock speed of 96MHz [4].

4.3 Heart Rate Measurement

M3: Heart Rate Measurement

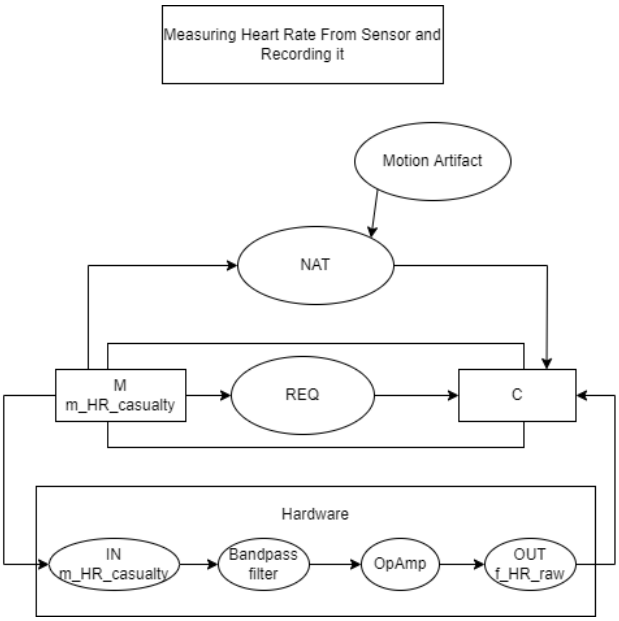

Figure 5: Heart Rate Measurement Mapping
NAT: Motion artifact caused by any movements of the casualty has the potential to impact the heart rate measurement.

4.3.1 Services

The heart rate measurement module uses the Pulse Express Pulse-Ox & Heart Rate Sensor with MAX32664 (sensor) to measure the heart rate of the casualty.

4.3.2 Secrets

The heart rate measurement module handles raw data input from the heart sensor and passes it to 'Heart Rate Data Processing' (M7). The heart rate measurement module performs analog filtering of the input signal from the sensor. This raw temperature value is sent to the 'Heart Rate Data Processing' (M7) and 'Data Storage' (M8).

4.3.3 Input

Table 9: Heart Rate Measurement Input Variables

Input Variable	**Description**	**Data Type**	**From Module**
m_HR_casualty	Casualty's heart rate as received by MAX32664	Analog	N/A

4.3.4 Output

Table 10: Heart Rate Measurement Output Variables

Output Variable	Description	Data Type	To Module
f_HR_raw	Raw heart rate data processed by bandpass filter and operational amplifier	Double	Blood Pressure Data Processing (M7), Data Storage (M8)

4.3.5 Software Components

None.

4.3.6 Hardware Components

- MAX32664 Optical Biometric Hub
- bandpass filter
- operational amplifier

4.3.7 Timing Constraints

There must be no more than 1 second to receive measurement from sensor to display (NFR4 in System Requirements). The MAX32664 has a clock speed of 96MHz [4].

4.4 Casualty Data Component

M4: Casualty Data Component

Casualty Module Interface Specifications

Module

CasualtyT

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
new_CasualtyT		CasualtyT	
getRawBodyTemp		f_BT_raw: double	
getRawBloodPres		f_BP_raw: double	
getRawHeartRate		f_HR_raw: double	
getBodyTemp		f_BT_proc: double	measureAgain
getBloodPres		f_BP_proc: double	measureAgain
getHeartRate		f_HR_proc: double	measureAgain
setRawBodyTemp	f_BT_raw: double		
setRawBloodPres	f_BP_raw: double		
setRawHeartRate	f_HR_raw: double		
setBodyTemp	f_BT_raw: double		measureAgain
setBloodPres	f_BP_raw: double		measureAgain
setHeartRate	f_HR_raw: double		measureAgain

Semantics

State Variables

casualtyID: int
f_BT_raw: double
f_BP_raw: double
f_HR_raw: double

f_BT_proc: double
f_BP_proc: double
f_HR_proc: double
f_BT_sensorOK: bool
f_BP_sensorOK: bool
f_HR_sensorOK: bool

State Invariant

None

Assumptions

The state variables will be set before they are used.

Access Routine Semantics

new CasualtyT():

- transition: *casualtyID* := generateID()
- output: *out* := *self*
- exception: none

getRawBodyTemp():

- output: *out* := *f_BT_raw*
- exception: none

getRawBloodPres():

- output: *out* := *f_BP_raw*
- exception: none

sgetRawHeartRate( ):

- output: *out* := *f_HR_raw*
- exception: none

getBodyTemp():

- output: *out* := *f_BT_proc*
- exception: measureAgain()

getBloodPres():

- output: *out* := *f_BP_proc*
- exception: measureAgain()

getHeartRate():

- output: *out* := *f_HR_proc*
- exception: measureAgain()

setRawBodyTemp(f_BT_raw):

- exception: none

setRawBloodPres(f_BP_raw):

- exception: none

setRawHeartRate(f_HR_raw):

- exception: none

setBodyTemp(f_BT_raw):

- transition:

  if checkInRange(smoothData(f_BT_raw, min, max) **then**
  f_BT_proc := smoothData(f_BT_raw)
  f_BT_sensorOK := TRUE
  else
  f_BT_sensorOK := FALSE
  measureAgain()
  end if

- exception: measureAgain()

setBloodPres(f_BP_raw):

- transition:

  if checkInRange(smoothData(f_BP_raw, min, max) **then**
  f_BP_proc := smoothData(f_BP_raw)
  f_BP_sensorOK := TRUE
  else
  f_BP_sensorOK := FALSE
  measureAgain()
  end if

- exception: measureAgain()

setHeartRate(f_HR_raw):

- transition:

  if checkInRange(smoothData(f_HR_raw, min, max) **then**
  f_HR_proc := smoothData(f_HR_raw)
  f_Hr_sensorOK := TRUE
  else
  f_HR_sensorOK := FALSE
  measureAgain()
  end if

- exception: measureAgain()

Local Functions

smoothData: rawData → *processedData*
smoothData(*rawData*)

checkInRange: checkInRange(processedData, min, max)

if smoothData(f_BP_raw, min, max) > min smoothData(f_BP_raw, min, max) < min) **then**
return TRUE
else
return FALSE
end if

generateID: → *intgenerateID*()

measureAgain: → *stringmeasureAgain*() : *print*("*PleaseMeasureAgain*")

4.5 Body Temperature Data Processing

M5: Body Temperature Data Processing

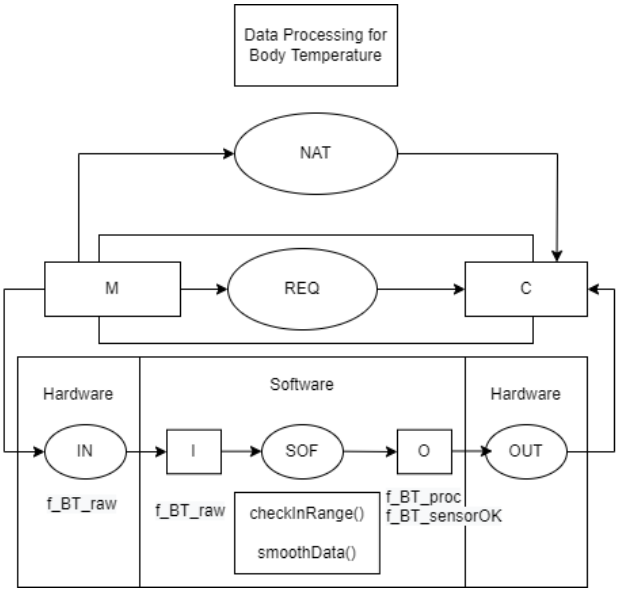

Figure 6: Body Temperature Data Processing Mapping

4.5.1 Service

The body temperature data processing module reduces noise and smooths the body temperature data. The body temperature data processing module additionally determines whether the readings are within an acceptable range.

4.5.2 Secrets

The body temperature data processing module handles output of the 'Body Temperature Measurement' (M1) module. Once smoothed, it is passed to the 'Data Storage' (M8) module. Additionally, the smoothed data is checked against a constant range of acceptable body temperatures and the result is stored in variable f_BT_sensorOK. This variable is also passed to the 'Data Storage' (M8) module.

4.5.3 Input

Table 11: Body Temperature Data Processing Input Variables

Input Variable	Description	Data Type	From Module
f_BT_raw	Raw body temperature data processed by bandpass filter and operational amplifier	Double	Body Temperature Measurement (M1)

4.5.4 Output

Table 12: Body Temperature Data Processing Output Variables

Output Variable	Description	Data Type	To Module
f_BT_proc	Processed body temperature data after digital smoothing	Double	Data Storage (M8)
f_BT_sensorOK	Body temperature sensor status indicator. True if body temperature is in acceptable range. False if out of range.	Boolean	Data Storage (M8)

4.5.5 Software Components

Refer to: M4 - Casualty Software Module

Table 13: Body Temperature Data Processing Software Components

Function	Parameters	Returns	Description
smoothData()	f_BT_raw	f_BT_proc	Removes noise and smooths data
checkInRange()	f_BT_proc, k_BT_min, k_BT_max	f_BT_sensorOK	Determines whether the processed data is in range

4.5.6 Hardware Components

- Raspberry Pi Zero W 2

4.5.7 Timing Constraints

There must be no more than 1 second to receive measurement from sensor to display (NFR4 in System Requirements).

4.6 Blood Pressure Data Processing

M6: Blood Pressure Data Processing

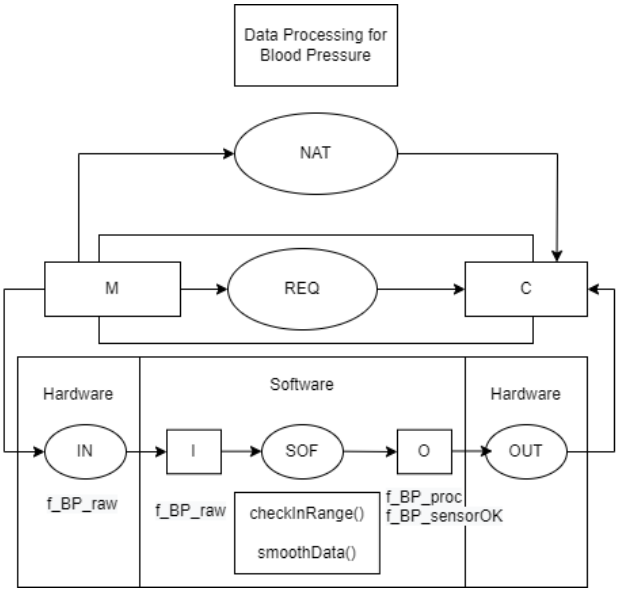

Figure 7: Blood Pressure Data Processing Mapping

4.6.1 Service

The blood pressure data processing module reduces noise and smooths the blood pressure data. The blood pressure data processing module additionally determines whether the blood pressure readings are within an acceptable range.

4.6.2 Secrets

The blood pressure data processing module handles output of the 'Blood Pressure Measurement' (M2) module. Once smoothed, it is passed to the 'Data Storage' (M8) module. Additionally, the smoothed data is checked against a constant range of acceptable blood pressure measurements and the result is stored in variable f_BP_sensorOK. This variable is also passed to the 'Data Storage' (M8) module.

4.6.3 Input

Table 14: Blood Pressure Data Processing Input Variables

Input Variable	Description	Data Type	From Module
f_BP_raw	Raw blood pressure data processed by bandpass filter and operational amplifier	Double	Blood Pressure Measurement (M2)

4.6.4 Output

Table 15: Blood Pressure Data Processing Output Variables

Output Variable	Description	Data Type	To Module
f_BP_proc	Processed blood pressure data after digital smoothing	Double	Data Storage (M8)
f_BP_sensorOK	Blood pressure sensor status indicator. True if blood pressure is in acceptable range. False if out of range.	Boolean	Data Storage (M8)

4.6.5 Software Components

Refer to: M4 - Casualty Software Module

Table 16: Blood Pressure Data Processing Software Components

Function	Parameters	Returns	Description
smoothData()	f_BP_raw	f_BP_proc	Removes noise and smooths data
checkInRange()	f_BP_proc, k_BP_min, k_BP_max	f_BP_sensorOK	Determines whether processed data is in range

4.6.6 Hardware Components

- Raspberry Pi Zero W 2

4.6.7 Timing Constraints

There must be no more than 1 second to receive measurement from sensor to display (NFR4 in System Requirements).

4.7 Heart Rate Data Processing

M7: Heart Rate Data Processing

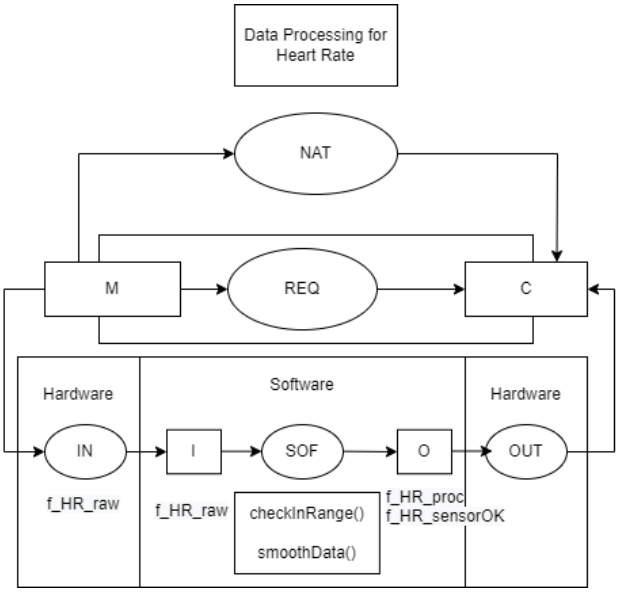

Figure 8: Heart Rate Data Processing Mapping

4.7.1 Service

Heart rate data processing module reduces noise and smooths the heart rate data. The heart rate data processing module additionally determines whether the heart rate readings are within an acceptable range.

4.7.2 Secrets

The heart rate data processing module handles output of the 'Heart Rate Measurement' (M3) module. Once smoothed, it is passed to the 'Data Storage' (M8) module. Additionally, the smoothed data is checked against a constant range of acceptable blood pressure measurements and the result is stored in variable f_BP_sensorOK. This variable is also passed to the 'Data Storage' (M8) module.

4.7.3 Input

Table 17: Heart Rate Data Processing Input Variables

Input Variable	Description	Data Type	From Module
f_HR_raw	Raw heart rate data processed by band-pass filter and operational amplifier	Double	Heart Rate Measurement (M3)

4.7.4 Output

Table 18: Heart Rate Data Processing Output Variables

Output Variable	Description	Data Type	To Module
f_HR_proc	Processed heart rate data after digital smoothing	Double	Data Storage (M8)
f_HR_sensorOK	Heart rate sensor status indicator. True if heart rate is within acceptable range. False if out of range.	Boolean	Data Storage (M8)

4.7.5 Software Components

Refer to: M4 - Casualty Software Module

Table 19: Heart Rate Data Processing Software Components

Function	Parameters	Returns	Description
smoothData()	f_HR_raw	f_HR_proc	Removes noise and smooths data
checkInRange()	f_HR_proc, k_HR_min, k_HR_max	f_HR_sensorOK	Determines whether processed data is in range

4.7.6 Hardware Components

- Raspberry Pi Zero W 2

4.7.7 Timing Constraints

There must be no more than 1 second to receive measurement from sensor to display (NFR4 in System Requirements).

4.8 Data Storage

M8: Data Storage

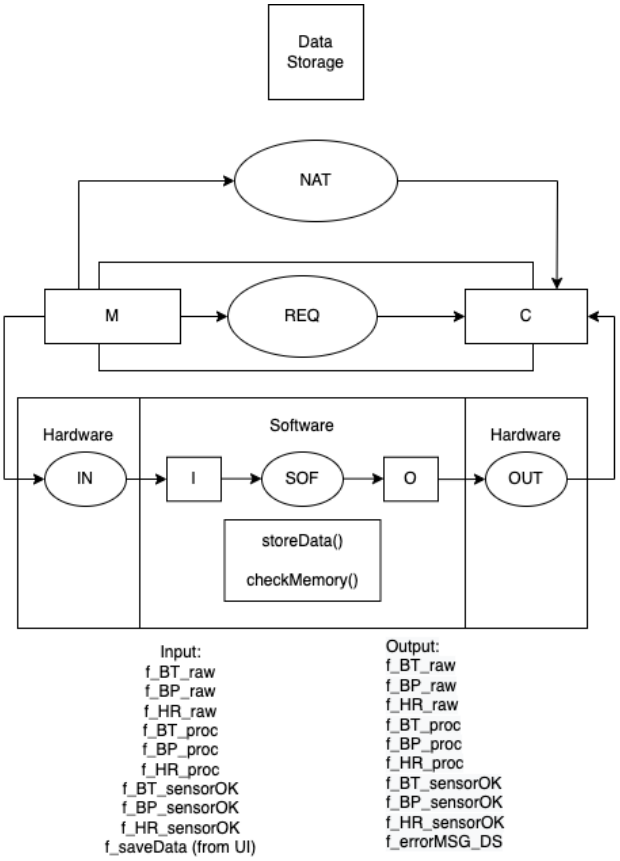

Figure 9: Data Storage Mapping

4.8.1 Services

Logs raw and processed sensor data as it is being collected and saves it to a patient profile. This data can later be exported to an external drive through the 'Data Exporting' (M1) module. The data storage module stores both the processed and unprocessed data. Additionally, it has the ability to separate the data saved for different patients. If the data isn't exported, it will be deleted on a power cycle.

4.8.2 Secrets

This module handles raw and processed data from the sensors. It detects any errors in the processed sensor data and sends a binary true/false value to the 'User Interface - Display System Status' (M12) module. Additionally, it saves the user's confirmation when prompted to either save or delete data from the local device.

4.8.3 Input

Table 20: Data Storage Input Variables

Input Variable	**Description**	**Data Type**	**From Module**
f_BT_raw	Measured body temperature reading	Double	Body Temperature Measurement (M1)
f_BP_raw	Measured blood pressure reading	Double	Blood Pressure Measurement (M2)
f_HR_raw	Measured heart rate reading	Double	Heart Rate Measurement (M3)
f_BT_proc	Processed body temperature reading	Double	Body Temperature Data Processing (M5)
f_BP_proc	Processed blood pressure reading	Double	Blood Pressure Data Processing (M6)
f_HR_proc	Processed heart rate reading	Double	Heart Rate Data Processing (M7)
f_BT_sensorOK	Body temperature sensor status	Boolean	Body Temperature Data Processing (M5)
f_BP_sensorOK	Blood pressure sensor status	Boolean	Blood Pressure Data Processing (M6)
f_HR_sensorOK	Heart rate sensor status	Boolean	Heart Rate Data Processing (M7)
f_saveData	Saves user's choice to save or delete collected data	Boolean	UI-ON/OFF (M10)

4.8.4 Output

Table 21: Data Storage Output Variables

Output Variable	Description	Data Type	To Module
f_BT_raw	Measured body temperature reading	Double	Data Export (M9)
f_BP_raw	Measured blood pressure reading	Double	Data Export (M9)
f_HR_raw	Measured heart rate reading	Double	Data Export (M9)
f_BT_proc	Processed body temperature reading	Double	Data Export (M9), User Interface
f_BP_proc	Processed blood pressure reading	Double	Data Export (M9), User Interface
f_HR_proc	Processed heart rate reading	Double	Data Export (M9), User Interface
f_BT_sensorOK	Body temperature sensor status	Boolean	Data Export (M9), User Interface
f_BP_sensorOK	Blood pressure sensor status	Boolean	Data Export (M9), User Interface
f_HR_sensorOK	Heart rate sensor status	Boolean	Data Export (M9), User Interface
f_errorMSG_DS	Sends thrown error messages in Data Storage module to the UI Display Status module	String	Data Export (M9), UI Display System Status (M12)

4.8.5 Software Components

Uses: Refer to: M4 - Casualty Software Module

Table 22: Data Storage Software Components

Function	Parameters	Returns	Description
storeData()	f_saveData	N/A	If f_saveData == true then data collected from sensors is saved locally on device. Otherwise data is deleted.
checkMemory()	N/A	f_errorMSG_DS	Monitors the data storage module and returns errors if any are found

4.8.6 Hardware Components

- Raspberry Pi Zero W 2
- SD card

4.8.7 Timing Constraints

N/A

4.9 Data Exporting

M9: Data Exporting

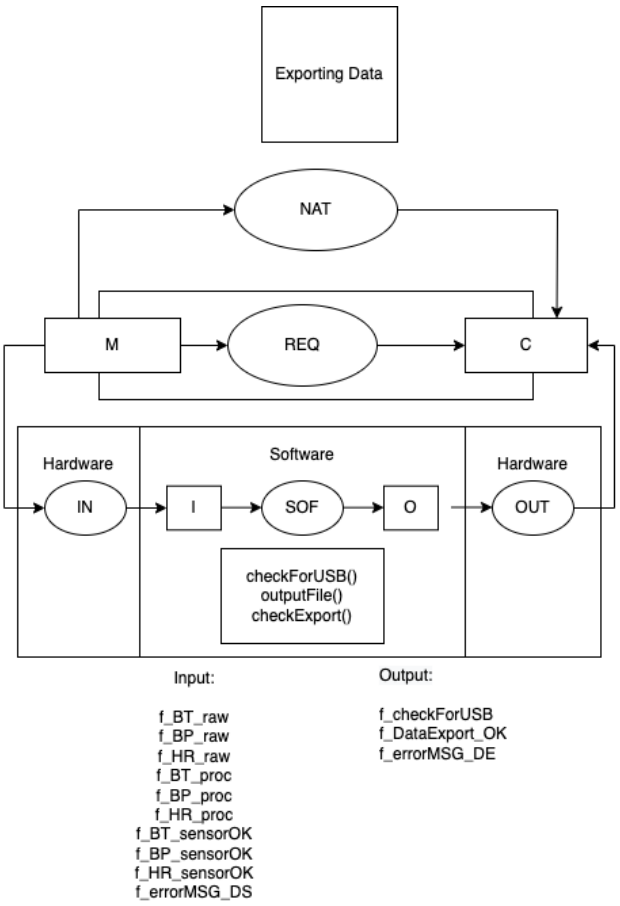

Figure 10: Data Export Mapping

4.9.1 Services

Exports selected saved vital sign measurement data to external drive. The user can choose to export raw data, processed data, or both.

4.9.2 Secrets

If the user chooses to export the data, the 'Data Export' (M9) module first checks if a USB drive has been inserted in the device. Additionally, it checks in background to see if the exporting process is carried out successfully. If it has not, it throws an error message to be displayed in the 'User Interface - Display System Status' (M12) module.

4.9.3 Input

Table 23: Data Exporting Input Variables

Input Variable	Description	Data Type	From Module
f_BT_raw	Measured body temperature reading	Double	Data Storage (M8)
f_BP_raw	Measured blood pressure reading	Double	Data Storage (M8)

Table 23: Data Exporting Input Variables

Input Variable	**Description**	**Data Type**	**From Module**
f_HR_raw	Measured heart rate reading	Double	Data Storage (M8)
f_BT_proc	Processed body temperature reading	Double	Data Storage (M8)
f_BP_proc	Processed blood pressure reading	Double	Data Storage (M8)
f_HR_proc	Processed heart rate reading	Double	Data Storage (M8)
f_BT_sensorOK	Body temperature sensor status	Boolean	Data Storage (M8)
f_BP_sensorOK	Blood pressure sensor status	Boolean	Data Storage (M8)
f_HR_sensorOK	Heart rate sensor status	Boolean	Data Storage (M8)
f_errorMSG_DS	Consists of thrown error messages in Data Storage module	String	Data Storage (M8)

4.9.4 Output

Table 24: Data Exporting Output Variables

Output Variable	**Description**	**Data Type**	**To Module**
f_CheckUSB	Detects if an external drive (USB) has been inserted into the device	Boolean	UI- Display System Status (M12)
f_DataExport_OK	Detects if the export process was carried out successfully	Boolean	Data Exporting (M9)
f_errorMSG_DE	Sends thrown error messages in the Data Export module	String	UI-Display System Status (M12)

4.9.5 Software Components

Table 25: Data Exporting Software Components

Function	Parameters	Returns	Description
checkforUSB()	N/A	f_CheckUSB	Check to see if a USB has been inserted in the device
outputFile()	N/A	patientRAW.txt, patientPROC.txt	Output files to be stored in the USB Drive
checkExport()	N/A	f_DataExport_OK	Ff an export issue has been detected, generate an error message and store it in f_errorMSG_DE

4.9.6 Hardware Components

- Raspberry Pi Zero W 2
- USB drive

4.10 User Interface - ON/OFF

M10: User Interface- On/Off

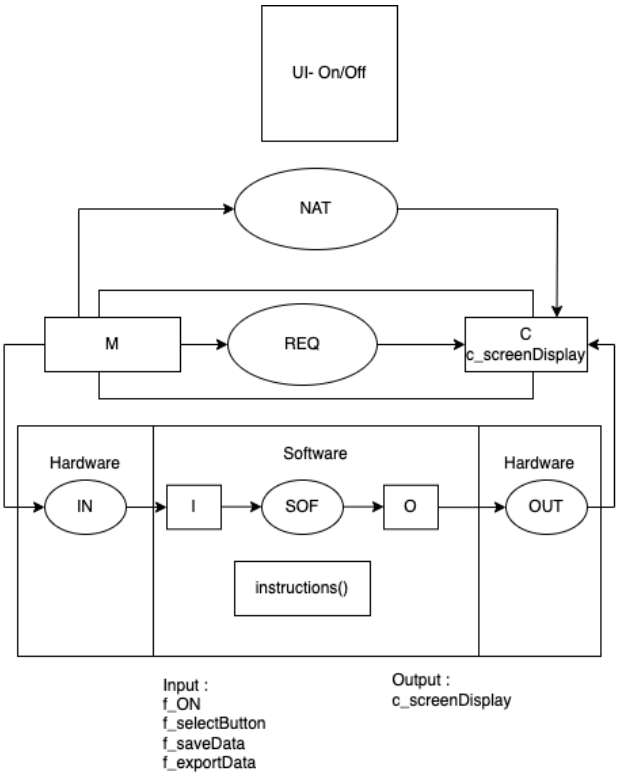

Figure 11: User Interface - On/Off Mapping

4.10.1 Services

A module of the user interface that is initialized by the user turning on the device to prompt placement instructions and to start measuring data. The turn off protocol prompts options to export and save data before turning off the device.

4.10.2 Secrets

When user turns on the device it will automatically prompt instructions to the user. When user selects to turn off the device it will automatically prompt the user to export and/or save data before fully turning off the device.

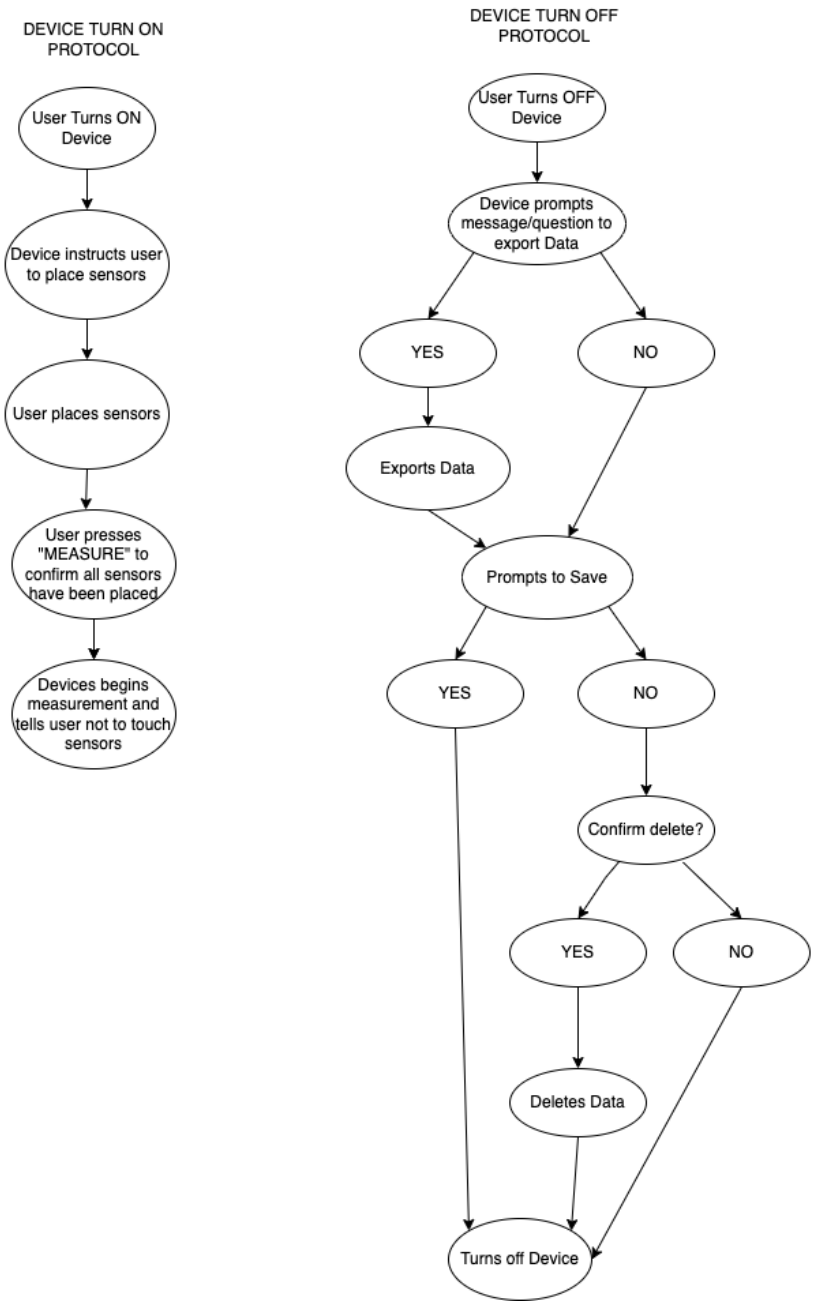

Figure 12: Flowchart of User Interface On and Off Controls

4.10.3 Input

Table 26: UI - ON/OFF Input Variables

Input Variable	**Description**	**Data Type**	**From Module**
f_ON	State Variable if the ON button is pressed	Boolean	Set by user input
f_startMeasure	State Variable indicating if the 'MEASURE' button is pressed	Boolean	Set by user input
f_saveData	Contains user's choice to save or delete the data from the current session	Boolean	Data Storage (M8)
f_exportData	Variable containing user's choice to export raw and/or processed data from the current session	Boolean	Data Export (M9)

4.10.4 Output

Table 27: UI - ON/OFF Output Variables

Output Variable	Description	Data Type	To Module
c_screenDisplay	Displays data and menus requested by the user on the LCD screen of the device.	Various. Depends on what is being displayed.	N/A

4.10.5 Software Components

Table 28: UI - ON/OFF Software Components

Function	Parameters	Returns	Description
instructions()	Prompted to give out instructions when user turns on the device	String	Outputs auditory and/or instructions on ideal placements of sensors for measurement

4.10.6 Hardware Components

- Raspberry Pi Zero W 2
- LCD display

4.11 User Interface - Display Values

M11: User Interface- Display Values

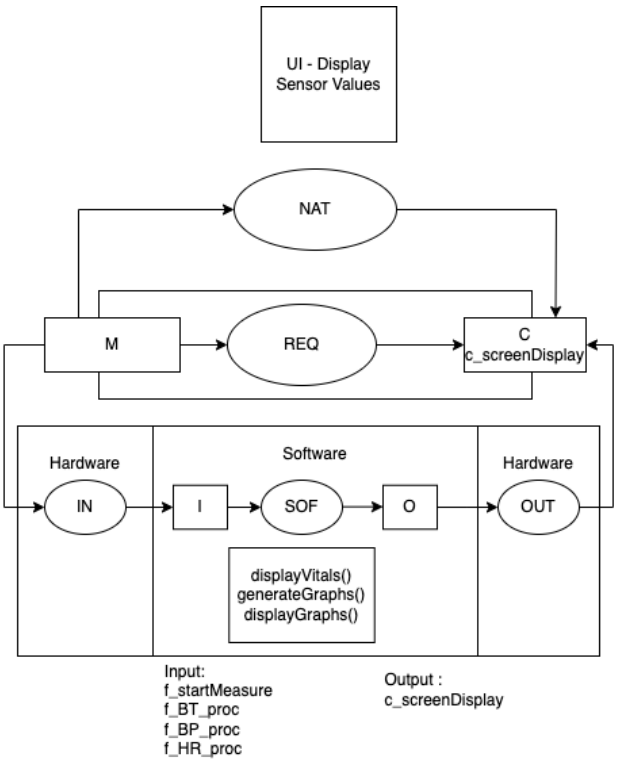

Figure 13: Mapping of User Interface - Display Values

4.11.1 Services

A module of the user interface that displays processed vitals data and plots graphs of the data on an LCD.

4.11.2 Secrets

Graphs will be generated to be displayed on the User Interface and processed Vitals or Errors will be generated and then displayed on the User Interface

4.11.3 Input

Table 29: UI - Display Values Input Variables

Input Variable	Description	Data Type	From Module
f_startMeasure	State Variable indicating if the 'MEASURE' button is pressed	Boolean	Set by user input
f_BT_proc	Processed body temperature reading	Double	Body Temperature Data Processing (M5)
f_BP_proc	Processed blood pressure reading	Double	Blood Pressure Data Processing (M6)
f_HR_proc	Processed heart rate reading	Double	Heart Rate Data Processing (M7)

4.11.4 Output

Table 30: UI - Display Values Output Variables

Output Variable	Description	Data Type	To Module
c_screenDisplay	Displays data and menus requested by the user on the LCD screen of the device.	Various. Depends on what is being displayed.	N/A

4.11.5 Software Components

Table 31: UI - Display Values Software Components

Function	Parameters	Returns	Description
displayVitals()	f_BT_proc f_BP_proc f_HR_proc	N/A	Displays numeric values of processed sensor readings on the LCD screen.
generateGraphs()	f_BT_proc f_BP_proc f_HR_proc	Plotting data	Generates graphical representation of processed sensor readings over time.
displayGraphs()	Plotting data output by generateGraphs()	N/A	Displays graphs generated in generateGraphs() function on the LCD screen.

4.11.6 Hardware Components

- Raspberry Pi Zero W 2
- LCD display

4.12 User Interface - Display System Status

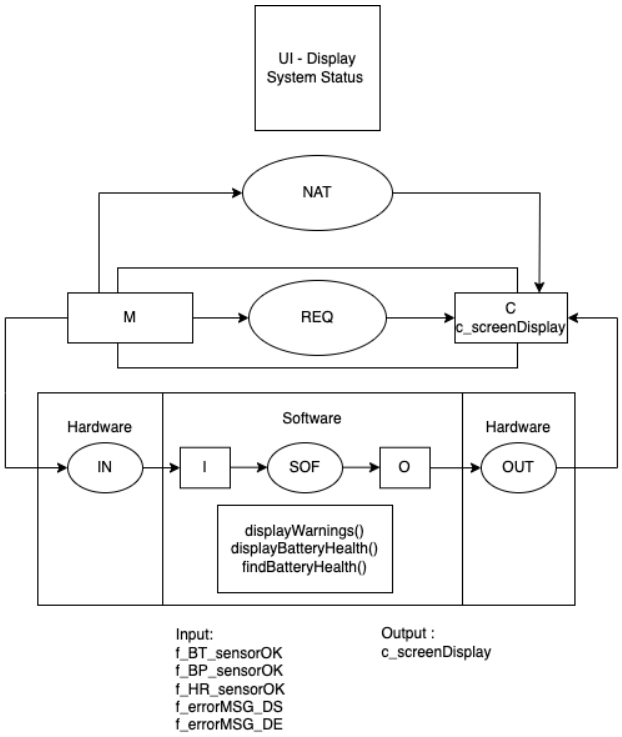

Figure 14: Mapping of User Interface - Display System Status

M12: User Interface- Display System Status

4.12.1 Services

A module of the user interface that displays system warnings and battery health of the device.

4.12.2 Secrets

Checks will be done consistently to monitor any issues with the system and battery health. If a certain issue has been returned then it will generate a warning to be displayed on the User Interface.

4.12.3 Input

Table 32: UI - Display System Status Input Variables

Input Variable	Description	Data Type	From Module
f_BT_sensorOK	Body temperature sensor status	Boolean	Body Temperature Data Processing (M5)
f_BP_sensorOK	Blood Pressure sensor status	Boolean	Blood Pressure Data Processing (M6)
f_HR_sensorOK	Heart Rate sensor status	Boolean	Heart Rate Data Processing (M7)
f_errorMSG_DS	Consists of thrown error messages in Data Storage module	String	Data Storage (M8)
f_errorMSG_DE	Consists of thrown error messages in Data Export module	String	Data Export (M9)

4.12.4 Output

Table 33: UI - Display Values Output Variables

Output Variable	Description	Data Type	To Module
c_screenDisplay	Displays data and menus requested by the user on the LCD screen of the device.	Various. Depends on what is being displayed.	N/A

4.12.5 Software Components

Table 34: UI - Display System Status Software Components

Function	Parameters	Returns	Description
displayWarnings()	N/A	N/A	Displays any warnings on the User Interface
displayBatteryHealth()	N/A	N/A	Displays Battery Percentage on the User Interface
findBatteryHealth()	N/A	N/A	Finds Battery Percentage of the Microcontroller

4.12.6 Hardware Components

- Raspberry Pi Zero W 2
- LCD display

4.13 User Interface - System Settings

M13: User Interface- System Settings

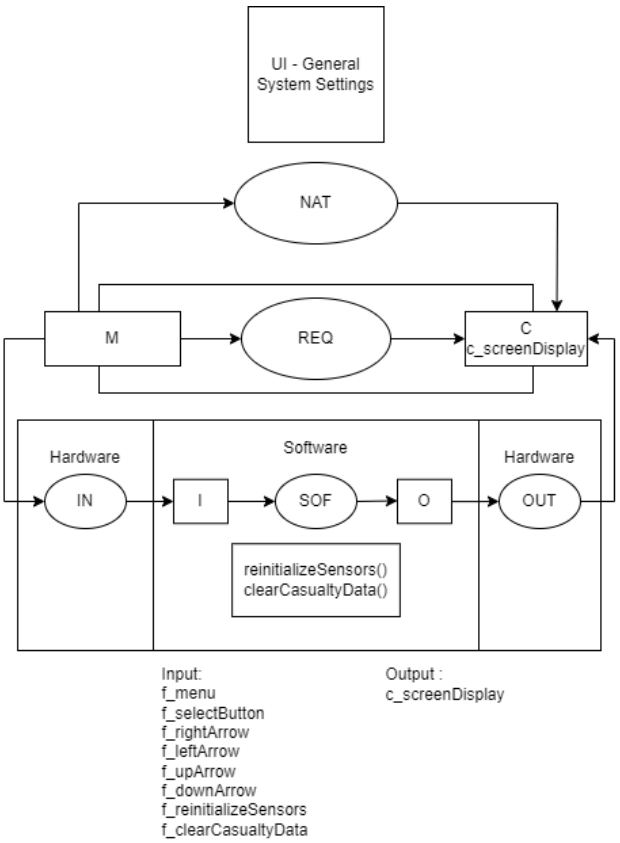

Figure 15: Mapping of User Interface - System Settings

4.13.1 Services

A module of the user interface that prompts options for re-initialization of data processing modules and options for deletion of saved casualty data.

4.13.2 Secrets

Re calibrating sensors, and resetting data processing algorithms and setting processed data to be zero and recalculating them.

4.13.3 Input

Table 35: UI - System Settings Input Variables

Input Variable	Description	Data Type	From Module
f_menu	Indicates if the Menu has been selected by the user	Boolean	Set by user input
f_selectButton	Press button that is currently selected in Menu	Boolean	Set by user input
f_rightArrow	Navigate UI to the right through Menu	Boolean	Set by user input
f_leftArrow	Navigate UI to the left through Menu	Boolean	Set by user input

Table 35: UI - System Settings Input Variables

Input Variable	Description	Data Type	From Module
f_upArrow	Navigate UI upwards through Menu	Boolean	Set by user input
f_downArrow	Navigate UI upwards through Menu	Boolean	Set by user input
f_reinitalizeSensors	Indicates if the user has chosen to reinitialize sensors	Boolean	UI local variable
f_clearCasualtyData	Indicates if the user has chosen to clear all casualty data currently stored on the device	Boolean	UI local variable

4.13.4 Output

Table 36: UI - Display Values Output Variables

Output Variable	Description	Data Type	To Module
c_screenDisplay	Displays data and menus requested by the user on the LCD screen of the device.	Various. Depends on what is being displayed.	N/A

4.13.5 Software Components

Table 37: UI - System Settings Software Components

Function	Parameters	Returns	Description
reinitializeSensors()	N/A	N/A	Resetting data processing algorithms
clearCasualtyData()	N/A	N/A	Setting processed Data to 0 and then recalibrating them

4.13.6 Hardware Components

- Raspberry Pi Zero W 2
- LCD display

5 System Behavior

5.1 Normal Operation

The LifeLine device collects human vital sign data via a series of sensors placed on a person's body, filters noisy data to be less erratic, and displays the data on a screen in a clear, easy to understand format. The device is also capable of logging the data it collects and exporting it in a variety of formats for further processing and analysis externally.

The device does not require any special training to operate. It is designed to be effective when used by people without knowledge of the device. The device is designed to be small, lightweight and portable so that it can be easily carried around everyday in a backpack, purse, or car. It is powered by a battery and can be entirely operated by a single person.

5.2 Undesired Event Handling

5.2.1 Impossible Sensor Values

In the event that the sensor readings are outside the possible range for a human the device will inform the user which sensor is giving erroneous readings. The device will recommend that the user ensure the sensor positioning is correct and give them the option to ignore the problematic sensor.

5.2.2 Motion Artifacts

In the event that the one or more sensors are not securely mounted on the casualty or the casualty is moving, the sensors may return abnormal readings. If these readings are consistently outside of acceptable ranges, the device will warn the user. However, if the motion artifacts present themselves as sporadic spikes the device will not be able to detect them. It is up to the user to ensure that the sensors are securely mounted on the casualty and that the casualty's movements do not dislodge them.

5.2.3 Physiological Noise

The sensor readings are expected to have a certain level of noise due to the characteristics of the sensors themselves as well as the variability in the physiological signals that are being read. The device will perform signal processing on data collected from each sensor to smooth out the incoming data however this process is not perfect. A certain level of variability in the data is expected.

5.2.4 Use in Abnormal Environments

Due to the intended use of the device it is expected to be used outdoors where the weather and temperature can vary drastically. That being said extreme heat or cold may impact the accuracy of the sensor readings and the operation of the micro-controller. They may also impact the battery life of the device.The device is also not designed to be submerged under water for long periods of time although it will have some water resistance. The device should be stored in a dry, climate controlled area to ensure consistent operation.

5.2.5 Casualty in Undesired Position

Depending on the medical situation a casualty may be in a position where it is difficult or impossible to place all sensors correctly on their body and moving them is not an option. Sensors may be placed in a variety of positions on the body and not all sensors need to be placed in order for the device to function, albeit at a reduced capacity.

5.2.6 Sensors Obstructed by Dirt or Residue

If abnormal sensor readings are detected, the device will inform the user to check that the sensors are not obstructed by any dirt or debris.

6 Appendix

References

[1] D. Parnas, P. Clement, and D. M. Weiss, "The modular structure of complex systems," in *International Conference on Software Engineering*, 1984, pp. 408–419.

[2] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Comm. ACM*, vol. 15, no. 2, pp. 1053–1058, December 1972.

[3] "Datasheet for mlx90614." [Online]. Available: https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90614

[4] "Max32664 ultra-low power biometric sensor hub: Maxim integrated." [Online]. Available: https://www.maximintegrated.com/en/products/interface/signal-integrity/MAX32664.html