Team 7 Test Plan
Flextimus-Prime

Authors:
Max Schweitzer
Julian Saunders
John Anderson
Grant Vesely

Revision Number	Date
1.0	11/30/17

Table of Contents

Introduction

This document is intended to ensure the system functions as a whole by testing from the ground up. By testing from the bottom up we can ensure any single point of failure due to hardware or software is quickly identified and fixed.

Objectives

The overall objective of this test plan is to test the flextimus-prime posture detector in a piece by piece fashion, using a combination of unit testing, functional testing, and error testing to ensure a robust product. The unit testing allows us to individually test the functionality of one module by itself, ensuring it will behave as expected before we integrate it with the other modules. Functional testing will be used to ensure we meet the features and abilities defined within our requirements specification, linked below. Error testing will help to ensure a robust system, and to minimize final bugs in the finished product.

Reference Documentation

Document Name	Description	Link
Requirements Specification	Official requirements specification document detailing various needs, mays, and shoulds of the project.	https://github.com/pdxjohnny/flextimus-prime/wiki/Requirements
Block Diagram	High level block view of total system and subsystems	https://github.com/pdxjohnny/flextimus-prime/blob/master/Block%20Diagram.pdf
Final Schematic	Eagle schematic file.	https://github.com/pdxjohnny/flextimus-prime/blob/master/flextimus-prime.sch
Final Board	Eagle board layout file.	https://github.com/pdxjohnny/flextimus-prime/blob/master/flextimus-prime.brd

Testing Plan

Hierarchical Test Outline

Unit Testing

Unit testing involves testing each individual part before testing it in any sort of combination with other parts. This ensures that we can find any single point of error due to hardware faults early on in the testing process.

Objectives:

- Component meets individual functional requirements
- Black box / white box tests pass
- Component meets design requirements for system

Test cases:

- Test power supply [UT-PWR1]
    - System is being powered by USB port with correct voltages.
- Test power supply connections [UT-PWR2]
    - Each individual component is receiving power from the power supply.
- Test Config button [UT-BTN1]
    - Button sends signal when asserted.
- Test Pause button [UT-BTN2]
    - Button sends signal when asserted.
- Test Config LED [UT-LED1]
    - LED lights when signal received.
- Test Pause LED [UT-LED2]
    - LED lights when signal received.
- Test Buzzer [UT-BZR1]
    - Buzzer sounds when signal received.
- Test LCD screen [UT-LCD1]
    - LCD screen turns on and visibly works, can be sent messages to display text.
- Test Flex sensor [UT-FLX1]
    - Flex sensor shows voltage change when bent.

Functional Testing

Functional testing involves testing how one or more parts interact with each other. Ideally we want to test at the lowest levels of functionality, just above unit testing. Once we know that a sub-function works, we can rule it out as a potential source of error in the higher level functionality. Additionally, we are testing for basic functionality of the system as well.

Objectives:

- Verify that components in subsystems function properly.
- Fix any hardware issues between components.
- Fix any software bugs that cause complete failures of the subsystem.
- Verify basic functionality

Test cases:

- Test Config mode [FT-CFG1]
  - Configuration mode begins when config button is pressed
- Test Pause mode [FT-PSE1]
  - Pause mode begins when pause button is pressed
- Test posture recording [FT-PST1]
  - Values for good posture are recorded correctly
- Test posture violation [FT-PST2]
  - Program detects suboptimal posture
- Test alert on a violation [FT-PST3]
  - Buzzer sounds when posture violation signal is received

Error Testing

Error testing involves interacting with the device in a way that it is not meant to function. We want to ensure that the system can continue to function properly when unexpected or undesired user input is detected.

Objectives:

- System functions as intended when used correctly.
- System does not fail when unexpected inputs are entered.

Test Cases:

- Test flex sensor errors [ET-TST1]
  - No warnings if flex sensor is bent the wrong way
- Test multiple simultaneous button input [ET-TST2]
  - Arbitration if pause and config buttons are pressed simultaneously
- Test incorrect button sequence [ET-TST3]
  - No errors if wrong button sequence is pressed, i.e. config and then pause
- Test config timeout [ET-TST4]
  - Config function times out after not receiving user input after some time limit

Resources

Personnel:

- Max Schweitzer
- Julian Saunders
- John Anderson
- Grant Vesely

Equiptment:

- Solder iron
- Reflow oven
- Microscope
- Multimeter
- ST-Link v2

Components/Boards:
Flextimus-prime board - Revision 1.2
Bill of Materials

Software:

- EAGLE
- ARM toolchain
- GDB
- St-util
- ST Link Utility

PCs:

- Personal PC's and Laptops

Operating Systems:

- Linux
- Windows 7/10

Detailed Test Cases

Test Writer: Julian Saunders			
Test Case Name:	Test Config mode	Test ID #:	FT-CFG1
Description:	Configuration mode begins when config button is pressed and iterates through the config sequence correctly when subsequent buttons are pressed.	Type:	White Box
Tester Information			
Name of Tester:		Date:	
Hardware Ver:	1.0	Time:	
Setup:	Power up the system with the flex sensor and configuration button enabled in the software. Only input will be through the sensor and button themselves. View outputs on LCD screen.		

Step	Action	Expected Result	Pass	Fail	Comments
1	First press of config button.	“Mode: Config” appears on screen.			
2	Adjust (bend) sensor to desired min and max position.	-			
3	Second press of config button.	Config mode is exited, min and max values are new monitor values. “Mode: Monitoring” appears on screen.			
Overall test results:					

Test Writer: Julian Saunders			
Test Case Name:	Test button sequence	Test ID #:	ET-TST3
Description:	System does not fault when incorrect buttons are pressed for different sequences.	Type:	White Box
Tester Information			
Name of Tester:		Date:	
Hardware Ver:	1.0	Time:	
Setup:	Whole system is booted up. Only inputs tested will be user input by config and pause buttons. System should continue to function without faults, showing outputs on screen.		

Step	Action	Expected Result	Pass	Fail	Comments
1	Press config button	“Mode: Config” appears on screen.			
2	Press pause button	System stays in config mode			
3	Finish config sequence by pressing config button again.	Sequence configures correctly and resumes to Monitoring mode, displaying “Mode: Monitoring” on screen.			
4	Press pause button	“Mode: Paused” appears on screen.			
5	Press config button	System stays in pause mode			
6	Press pause button	Sequence ends pause mode correctly and			

		resumes Monitoring mode. Shows “Mode: Monitoring” on screen.			
Overall test results:					