**ECE 412**

**High Assurance Controller of Self-balancing Robot: Test Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Authors: | Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang | | | | Group # 10 |
| Github | <https://github.com/Artem1199/ECE-HACoSR2/tree/85f0d0fe3be21060c7b8cf96b7f15a4fa912cec2> | | | | |
| Version #: | 1.0 |  |  |  |  |
| Date: | 10-Apr-2020 |  |  |  |  |

**Table of Contents**

[**1.0 Introduction**](#_heading=h.454sypogesv6) **1**

[**2.0 References**](#_heading=h.logw7luts7cu) **1**

[2.1 Documents](#_heading=h.rumt1tgsvnxo) 1

[**3.0 Resources**](#_heading=h.ctv4ik1cga29) **1**

[3.1 Equipment](#_heading=h.sa97aw3d8php) 1

[3.2 Project Specific Hardware](#_heading=h.a2zy0ygu0yze) 1

[3.3 Software](#_heading=h.x7vcyooxepa) 2

[**4.0 Objectives**](#_heading=h.ybfoonigu7d7) **2**

[4.1 Unit Test](#_heading=h.vqs996uvmfwn) 2

[4.2 Integration Test](#_heading=h.rsd46trr1vn0) 2

[4.3 Functionality Test](#_heading=h.2t80hwrt408q) 2

[4.4 Stress Test](#_heading=h.j1hvka5w7u0s) 2

[4.5 Parametric Test](#_heading=h.577ypmo5gc3l) 2

[4.6 Acceptance Test](#_heading=h.x64bpsrdyxhn) 3

[**5.0 System Tests**](#_heading=h.82hi4x139omn) **3**

[5.1 Unit Tests](#_heading=h.g20tn1y95w67) 3

[5.1.1: PWM Accuracy test (Test Id:RT-UT-01)](#_heading=h.j9p2g16uw9yc) 3

[5.1.2: PWM Response Rate (Test Id:RT-UT-02)](#_heading=h.romkcg8p728) 4

[5.1.3: PWM Response Rate (Test Id:RT-UT-03)](#_heading=h.ay2upimvfkj4) 6

[5.2 Integration Test](#_heading=h.l36gvh2ylhh7) 7

[5.2.1 Tilt Controller test (Test Id: RT-IT-01)](#_heading=h.bfytpasawtdt) 7

[5.2.2 Yaw Controller Test (Test Id: RT-IT-02)](#_heading=h.mldp4ukn41wb) 8

[5.3 Functionality Test](#_heading=h.jfagrlt2ghll) 9

[5.3.1 Physical Robot Balance and Yaw performance (Test Id: RT-FT-01)](#_heading=h.s64c3i7sdg15) 9

[5.3.2 Test 1: IMU and Web server measurement test (Test Id: RT-UT-01)](#_heading=h.pj6wrsj5d95e) 10

[5.4 Stress Test](#_heading=h.as4b4r3h8t4m) 12

[5.3.2 Test 1: IMU and Web server measurement test (Test Id: RT-ST-01)](#_heading=h.du4vm1wieurr) 12

[5.5 Parametric Test](#_heading=h.ihjkfybj2wuz) 13

[5.6 Acceptance Test](#_heading=h.it8t3ri9p29a) 13

[5.6.1: PWM Response Rate (Test Id:RT-AT-02)](#_heading=h.rxamd94j5myp) 13

# 

# **1.0 Introduction**

The test plan was developed to document and track the essential information required to soundly define methods for testing the two wheeled inverted pendulum robot (TWIP) and ensure that it functions correctly based on the specifications defined in PDS. The team broke down the test planning based on the component modules, applied unit testing, and integration tests. The intended audience for the test plan is the project team, student peers, and professors.

# **2.0 References**

## **2.1 Documents**

### 

|  |  |  |
| --- | --- | --- |
| Product Design Specification | Version 1 | February 18, 2020 |
| <https://github.com/Artem1199/ECE-HACoSR2/blob/master/Docs/PDS/2020_PSU_Capstone_10_PDS.pdf> | | |

### 

|  |  |  |
| --- | --- | --- |
| Galois Project Proposal |  |  |
| <https://github.com/GaloisInc/HighAssuranceControllerOfSelfBalancingRobotCapstone/blob/master/ECE_Casptone_High_assurance_controller_II.pdf> | | |

# **3.0 Resources**

## **3.1 Equipment**

* Oscilloscope
* Digital Multimeter
* Soldering Iron
* Protractor

## **3.2 Project Specific Hardware**

* Arduino MKR WiFi 1010.
* Arduino ESP-32
* TWIP (Robot built by previous team)
* TWIP charging unit.

## **3.3 Software**

* Arduino IDE (1.8.10) and related drivers
* MATLAB/Simulink
* Lustre

# **4.0 Objectives**

The objective of this document is to test the two-wheeled inverted pendulums (TWIP) against all of our requirements. This will be done by evaluating modules in our system and performing some of the types of tests listed below.

## **4.1 Unit Test**

Multiple individual units will have to be tested using various equipment to verify the functionality of individual modules.

## **4.2 Integration Test**

Integration tests involve combining multiple units and testing them together to verify that the modules will interact as expected before the whole system is entirely closed up inside the box.

## **4.3 Functionality Test**

Functionality testing will be used to confirm basic functionality, without strict parametric test. The modules should be able to turn on and switch between states after certain inputs.

## **4.4 Stress Test**

Stress testing will verify the stability and reliability of the system to determine robustness.

## **4.5 Parametric Test**

Parametric testing will verify that the observed data is distributed according to our design parameters.

## **4.6 Acceptance Test**

The purpose of this test is to evaluate the system’s compliance with the design requirements and assess whether it is acceptable or not.

# **5.0 System Tests**

## **5.1 Unit Tests**

## 5.1.1: PWM Accuracy test (Test Id:RT-UT-01)

PDS Related Requirement: change the “otis\_arduino” code so it works with Arduino MKR WiFi 1010.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | PWM Accuracy | | | **Test ID #:** | | RT-UT-01 |
| **Description:** | | Compare PWM expected output consistency of old board and new board. | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | ESP board and MKR1010 boards with PWM output code. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Setup MKR1010 Cortex m0+ boards and old ESP board with identical PWM control code and test vectors. | Code should compile and upload. |  |  |  |  | |
| 2 | Create varying PWM inputs. Measure PWM output. | N/A |  |  |  |  | |
| 3 | Measure PWM at 0% duty cycle | Frequency, and duty cycle of both boards should match within 5%. |  |  |  |  | |
| 4 | Measure PWM at 25% duty cycle | Frequency, and duty cycle of both boards should match within 5%. |  |  |  |  | |
| 5 | Measure PWM at 50% | Frequency, and duty cycle of both boards should match within 5%. |  |  |  |  | |
| 6 | Measure PWM at 75% | Frequency, and duty cycle of both boards should match within 5%. |  |  |  |  | |
| 7 | Measure PWM at 100% | Frequency, and duty cycle of both boards should match within 5%. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

### 5.1.2: PWM Response Rate (Test Id:RT-UT-02)

PDS Related Requirement: change the “otis\_arduino” code so it works with Arduino MKR WiFi 1010.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | PWM Response | | | **Test ID #:** | | RT-UT-02 |
| **Description:** | | Compare PID controller output on MKR1010 and ESP processor boards and response time comparison. | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | ESP boards connected to MPU6050 with oscilloscope measuring PWM outputs. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Assemble ESP boards on breadboard with MPU6050 communication. MKR1010 reuse current processor on B.B. | ESP: establish communication w/ I2C, MKR1010: N/A. |  |  |  |  | |
| 2 | Upload current working TWIP code with “PID.h” library for both ESP and MKR1010 boards. Match setpoints, and P,I,D constants, and SampleTimes on both boards. | Code should compile and upload. |  |  |  |  | |
| 3 | Physically move MPU6050 to setpoint tilt value. | Controller outputs should be within 5%. |  |  |  |  | |
| 4 | Tilt MPU6050 on MKR1010 robot to one extreme of robot tilt, record value, match value on ESP-32. Record PID output. Set an oscilloscope to measure PWM output continuously. | Controller outputs should be within 5%. Compare PWM output change response time. Response time should be within 10%. |  |  |  |  | |
| 5 | Tilt MPU6050 on MKR1010 robot to opposite extreme of robot tilt, record value, match value on ESP. Record PID output. Set an oscilloscope to measure PWM output continuously. | Controller outputs should be within 5%. Compare PWM output change response time. Response time should be within 10%. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

### 

### 5.1.3: PWM Response Rate (Test Id:RT-UT-03)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | Rust PID library | | | **Test ID #:** | | RT-UT-03 |
| **Description:** | | Compare C++ library and Rust library. | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | MKR1010 board with Rust PID library generated by Lustre and C++ PID libraries. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Modify code to implement both Rust and C++ PID libraries. Upload to MPU6050. | Code should upload to MKR1010. |  |  |  |  | |
| 3 | Physically move MPU6050 to setpoint tilt value. | Output of C++ PID and MKR1010 PID libraries should be identical. |  |  |  |  | |
| 4 | Tilt MPU6050 on MKR1010 robot to one extreme of robot tilt. | Output of C++ PID and MKR1010 PID libraries should be identical. |  |  |  |  | |
| 5 | Tilt MPU6050 on MKR1010 robot to opposite extreme of robot tilt. | Output of C++ PID and MKR1010 PID libraries should be identical. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

## **5.2 Integration Test**

### 5.2.1 Tilt Controller test (Test Id: RT-IT-01)

PDS Related Requirement: Develop and verify a controller with a wider range of stable input con- ditions and compare its performance with the PID controller through both simulation and in the real system.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:** Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | Tilt controller | | | **Test ID #:** | | RT-IT-01 |
| **Description:** | | The tilt controller is the heart of the self balancing robot. It is tested to ensure that the control input (output of the controller) is as expected for several different inputs. There are 2 different tilt controllers: PID and Fuzzy Mamdani. The same type of test is conducted for both. | | | **Type:** | | white box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | |  | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | Disconnect the tilt control input from the robot and probe this point with an oscilloscope. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Disconnect control input from robot and attach an oscilloscope probe to the control input. | Will see a PWM signal generated on the screen of the oscilloscope. |  |  |  |  | |
| 2 | Move the device to a known tilt angle. | A PWM of different duty cycle should appear on the oscilloscope. |  |  |  |  | |
| 3 | Translate the control signal to a motor torque given the manufacturers data. | The torque from the motors is the same as simulated in Simulink. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

### 5.2.2 Yaw Controller Test (Test Id: RT-IT-02)

PDS Related Requirement: Develop and verify a controller with a wider range of stable input con- ditions and compare its performance with the PID controller through both simulation and in the real system.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | Yaw controller | | | **Test ID #:** | | RT-IT-02 |
| **Description:** | | The yaw controller allows the orientation of the robot to remain at a certain angle. It is tested to ensure that the control input (output of the controller) is as expected for several different inputs. The yaw control system uses PD control. | | | **Type:** | | white box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | |  | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | Disconnect the yaw control input from the robot and probe this point with an oscilloscope. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Disconnect control input from robot and attach an oscilloscope probe to the control input. | Will see a PWM signal generated on the screen of the oscilloscope. |  |  |  |  | |
| 2 | Move the device to a known yaw angle. | A PWM of different duty cycle should appear on the oscilloscope. |  |  |  |  | |
| 3 | Translate the control signal to a motor torque given the manufacturers data. | The torque from the motors is the same as simulated in Simulink. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

## 

## **5.3 Functionality Test**

### 5.3.1 Physical Robot Balance and Yaw performance (Test Id: RT-FT-01)

PDS Related Requirement: Develop and verify a controller with a wider range of stable input con- ditions and compare its performance with the PID controller through both simulation and in the real system.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | Basic balance and yaw Functionality | | | **Test ID #:** | | RT-FT-01 |
| **Description:** | | This test is to ensure the robot remains upright and rotates approximately to its initial orientation when physically moved. | | | **Type:** | | white box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | |  | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | No external measurement tools are needed. Just set the robot upright and turn it on. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Hold the robot such that it has a tilt angle of approximately zero. | The robot is held upright. |  |  |  |  | |
| 2 | Turn the robot on and release it. | The robot may jerk around, but ultimately remains upright. |  |  |  |  | |
| 3 | Move the robot such that its yaw angle is different from the initial yaw angle. | The robot should go approximately back to its starting point with respect to yaw while remaining balanced. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

### 5.3.2 Test 1: IMU and Web server measurement test (Test Id: RT-UT-01)

PDS Related Requirement: change the “otis\_arduino” code so it works with Arduino MKR WiFi 1010.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | IMU and Webserver Measurements | | | **Test ID #:** | | RT-FT-02 |
| **Description:** | | Test data sent to the Webserver is accurate to data displayed in the serial stream. | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | MKR1010 with wifi access. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Modify Arduino code to print serial data equivalent to Web Server data. | NA |  |  |  |  | |
| 2 | Upload code and run robot, connect to Web Server. | Robot should connect to web server |  |  |  |  | |
| 3 | Max out tilt in one direction, max out tilt in opposite direction. | Data should be output in the serial monitor, and on the website. |  |  |  |  | |
| 4 | Max out Yaw in 1 direction, max out yaw in the opposite direction | Data on the server should respond to change. |  |  |  |  | |
| 5 | Copy Serial data and SQL data to a spreadsheet. Compare Serial data and SQL data using excel conditional formatting. | Values should match within 5%. Check for any missing data or hang up issues. Hang ups or lost should occur less than 10% of the time. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

## **5.4 Stress Test**

Given the current situation (COVID-19) and the fact that we only have one robot to develop we will avoid most stress tests for now. This is a living document and changes may be made in the future.

### 5.4.1 Test 1: IMU and Web server measurement test (Test Id: RT-ST-01)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | PWM Accuracy | | | **Test ID #:** | | RT-ST-01 |
| **Description:** | | Communication and PID calculation robustness. | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | MKR1010 board with TCP communication enabled and Rust Heap memory based PID. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Program MKR1010 with Wifi TCP enabled and Rust heap allocated PID. | Code should compile, Arduino should connect to wifi and start writing data. |  |  |  |  | |
| 2 | Run the program for 8 hours. Monitor data output to the server. After 8 hour verify data is still live by moving the robot. | MKR1010 should consistently output live data to the server. MKR1010 should not hang up losing communication with MPU6050 or with a web server. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

## **5.5 Parametric Test**

Verifying that our controller performs as simulated is a very important part of this project. This is a task in progress and is more detailed than what can be included in the tabular form. Separate documents will be written specifically for the verification of the controller.

## **5.6 Acceptance Test**

It is important to note that this project is research based rather than focused on creating a product. The system's acceptance is based on the ability to establish formal verification for the TWIP. In other words, the basic functionality of the robot allows us to have a complex system that we can formally verify.

### 5.6.1: PWM Response Rate (Test Id:RT-AT-02)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | Rust PID library | | | **Test ID #:** | | RT-AT-01 |
| **Description:** | | Compare Lustre generated code vs. code written manually. | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | |  |
| **Hardware Version:** | | 1.0 | | | **Time:** | |  |
| **Setup:** | | Create Rust testbench with Lustre generated and Rust code. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Create a library with test vectors. Modify Lustre and self written programs to allow for testbench inputs. | Program should compile. |  |  |  |  | |
| 3 | Test input vectors based on values measured from MPU6050 real world movement. | Create assert functions to compare inputs and outputs. |  |  |  |  | |
| 4 | Test input values at setpoint, one tilt extreme and other tilt extreme. | All assert statements should pass. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |