**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 | 2.0 |  |  |  |
| Date: | 10-Apr-2020 | 27-May-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**: | | 13 April |
| **Hardware Version:** | | 1.0 | | | **Time:** | | 1:00 AM |
| **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. | X |  |  |  | |
| 2 | Create varying PWM inputs. Measure PWM output. | N/A | X |  |  | Created input by mapping MPU6050 input data to PWM output. | |
| 3 | Measure PWM at 0% duty cycle | Frequency, and duty cycle of both boards should match within 5%. | X |  |  | 0% error. | |
| 4 | Measure PWM at 25% duty cycle | Frequency, and duty cycle of both boards should match within 5%. | X |  |  | <1% error on output. | |
| 5 | Measure PWM at 50% | Frequency, and duty cycle of both boards should match within 5%. | X |  |  | <1% error on output. | |
| 6 | Measure PWM at 75% | Frequency, and duty cycle of both boards should match within 5%. | X |  |  | <1% error on output. | |
| 7 | Measure PWM at 100% | Frequency, and duty cycle of both boards should match within 5%. | X |  |  | <1% error on output. | |
| **Overall test result:** | | | X |  |  | See images in same folder for more info. | |

### 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**: | | 14 April 2020 |
| **Hardware Version:** | | 1.0 | | | **Time:** | | 2:38AM |
| **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. | X |  |  | 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. | X |  |  | Modified outputSum += ki\*error to outputSum = ki\*error. MKR limit set to 255, then mapped magnitude to 1k, ESP limit set to 255. | |
| 3 | Physically move MPU6050 to setpoint tilt value. | Controller outputs should be within 5%. | X |  |  | <1% error on output. | |
| 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%. | X |  |  | <1% error on output. Loop process time for both boards is within 1%. | |
| 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%. | X |  |  | <1% error on output.  Loop process time for both boards is within 1%. | |
| **Overall test result:** | | | X |  |  | PASSED. | |

### 

### 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**: | | 25 May 2020 |
| **Hardware Version:** | | 2.0 | | | **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. | X |  |  |  | |
| 2 | Physically move MPU6050 to setpoint tilt value. | Output of C++ PID and MKR1010 PID libraries should be identical. | X |  |  |  | |
| 3 | Tilt MPU6050 on MKR1010 robot to one extreme of robot tilt. | Output of C++ PID and MKR1010 PID libraries should be identical. | X |  |  |  | |
| 4 | Tilt MPU6050 on MKR1010 robot to opposite extreme of robot tilt. | Output of C++ PID and MKR1010 PID libraries should be identical. | X |  |  | When finding the setpoint, the outputs of the serial are slightl different. i.e. Serial says we are 185 deg from setpoint while tcp says we are -175 degree away. This is technically correct either way. | |
| **Overall test result:** | | | X |  |  |  | |

### **5.1.4: TCP Data Accuracy (Test Id:RT-UT-04)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | Rust PID library | | | **Test ID #:** | | RT-UT-04 |
| **Description:** | | Compare serial data output to tcp data output. | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | | May 26th, 2020 |
| **Hardware Version:** | | 2.0 | | | **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 serial print data out data at the same time as sending it over tcp. | Could should compile. | X |  |  |  | |
| 2 | Move the robot through the ranges of motion for pitch. | N/A | X |  |  |  | |
| 3 | Compare results from database to tcp output over 30 second period. | Serial data and TCP data should be within 2% error value wise. And less than 2% missed data points. | X |  |  | At around 180 degrees, one output will print ~185 while the other prints ~-175. Technically both are correct. | |
| **Overall test result:** | | | X |  |  | PASS | |

### **5.1.5: Adafruit motor shield PWM accuracy (Test Id:RT-UT-05)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | |  | | | **Test ID #:** | | RT-UT-05 |
| **Description:** | | Compare the PWM output to the motors | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | | 27 May 2020 |
| **Hardware Version:** | | 2.0 and 1.0 | | | **Time:** | | 4:00PM |
| **Setup:** | | ESP boards connected to MPU6050 with oscilloscope measuring PWM outputs. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Program PWM output through arduino using methods in test case 5.1.2. Setup PWM output for adafruit shield. | Code should compile and upload. | X |  |  |  | |
| 2 | Setup oscilliscope to measure output of both MKR1010 PWM and adafruit shield. | N/A | X |  |  |  | |
| 3 | Physically move MPU6050 to setpoint tilt value. | Controller outputs should be within 5%. | X |  |  | Duty cycles is within 5%. Frequency is not. | |
| 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. | X |  |  | With a 55/255 input, duty cycle in the same for both at ~23% within 1% error. PWM duty cycles for both old and new robot are identical. Frequencies are not, this is a limitation of the adafruit motor board. | |
| 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. | X |  |  | With a 215/255 input, duty cycle in the same for both at ~83% within 1% error.  PWM duty cycles for both old and new robot are identical. Frequencies are not, this is a limitation of the adafruit motor board. | |
| **Overall test result:** | | | X |  |  | PASSED. | |

## **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. |  |  |  | No valid simulink model to compare | |
| **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. |  |  |  | No valid simulink model to compare | |
| **Overall test result:** | | |  |  |  |  | |

## 

## **5.3 Functionality Test**

### 5.3.1 Physical Robot Balance performance C++ vs Kind2 PID (Test Id: RT-FT-01)

PDS Related Requirement 1: Extend the existing code for the inverted pendulum robot to include a Rust control library

PDS Related Requirement 2: Modify Lustre/Kind2 Rust code generator to generate embedded friendly Rust, which can be directly imported in your library.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | PID C++ vs Rust | | | **Test ID #:** | | RT-FT-01 |
| **Description:** | | Test to compare preformance of PIDs implemented in C++ and Rust. Functinality should be near identical | | | **Type:** | | white box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | | 27 May 2020 |
| **Hardware Version:** | | 2.0 | | | **Time:** | | 5:00PM |
| **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 | Rust and C++ PID Hold the robot such that it has a tilt angle of approximately zero. | The robot is held upright. | X |  |  | Robot has no issue balancing at center. | |
| 2 | For Rust and C++ Turn the robot on and release it. | The robot may jerk around, but ultimately remains upright. | X |  |  | Robot has no issue balancing at center. | |
| 3 | With the C++ PID library enabled installed. Move the robot to both sizes of the tilt angle, and find the offset before the robot cannot recover. Record this offset. | N/A |  |  | X | PID , recovers from +/-40 degrees from offset. | |
| 4 | With the Rust PID library enabled installed. Move the robot to both sizes of the tilt angle, and find the offset before teh robot cannot recover. | The Rust PID point of no return should be within 5% of the C++ PID point of no return. | X |  |  | Rust PID is able to recover from +/- 40 degrees from offset. Can’t any further offset because of robot physical body. | |
| 5 | Pick robot off table. Hold robot by hand. See if robot is able to balance itself while in hand | Rust PID and C++ PID should have identical responses. | X |  |  | Robot will bounce between -30 to +30 degrees, but is able to balance itself the majority of the time. | |
| **Overall test result:** | | | X |  |  | Rust PID and C++ PID are functionally very similar. | |

### 5.3.2 Physical Robot Balance performance Rust PID vs Rust Fuzzy Cont. (Test Id: RT-FT-02)

PDS Related Requirement 1: Extend the existing code for the inverted pendulum robot to include a Rust control library

PDS Related Requirement 2: Modify Lustre/Kind2 Rust code generator to generate embedded friendly Rust, which can be directly imported in your library.

PDS Related Requirement 6: Develop and verify a controller with a wider range of stable input conditions and compare its perforamnce with the PID cnotroller through simulation and in the real system.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | PID Rust vs Fuzzy Rust | | | **Test ID #:** | | RT-FT-02 |
| **Description:** | | Compare the Rust Fuzzy controller for tilt input and Rust PID controller for various inputs. | | | **Type:** | | white box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | | 27 May 2020 |
| **Hardware Version:** | | 2.0 | | | **Time:** | | 7:00PM |
| **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 | For the Fuzzy logic controller hold the robot such that it has a tilt angle of approximately zero. | The robot is held upright. | X |  |  | Robot has no issue balancing at center. | |
| 2 | For the fuzzy logic controlled robot, turn the robot on and release it. | The robot may jerk around, but ultimately remains upright. | X |  |  | Robot has no issue balancing at center. | |
| 3 | With the Rust PID library enabled. Move the robot to both sizes of the tilt angle, and find the offset before the robot cannot recover. Record this offset. | N/A |  |  | X | PID , recovers from +/-40 degrees from offset. | |
| 4 | With the Rust Fuzzy logic controller enabled. Move the robot to both sizes of the tilt angle, and find the offset before teh robot cannot recover. | The Fuzzy logic controller should have a better response than the PID controller. |  |  | X | Fuzzy logic controller is unable to recover past about 15 degress offset from the setpoint. | |
| 5 | Pick robot off table. Hold robot by hand. See if robot is able to balance itself while in hand | The fuzzy logic controller should have a better response than the PID controller. |  |  | X | Fuzzy logic ontroller doesn’t have nearly as good performance. | |
| **Overall test result:** | | |  |  | X | Fuzzy logic controller isn’t as good as the PID. Potentially can get better with better tuning. | |

## **5.4 Stress Test**

### 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**: | | 27 May 2020 |
| **Hardware Version:** | | 2.0 | | | **Time:** | | 7:35PM |
| **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. |  |  | X |  | |
| 2 | Run the program for 2 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. |  |  |  | Start at 7:38PM, no issues.  7:43PM, no issues.  7:50PM, no issues. | |
| **Overall test result:** | | |  |  |  |  | |

### 5.4.2 Test 2: IMU and Serial data measurement test (Test Id: RT-ST-02)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:**Team 10(Ignacio Mejia-Rodriguez, Artem Kulakevich, Andrew Forsman, Yuqi Wang) | | | | | | | |
| **Test Case Name:** | | PWM Accuracy | | | **Test ID #:** | | RT-ST-02 |
| **Description:** | | Communication and PID calculation robustness. | | | **Type:** | | black box |
| **Tester Information** | | | | | | | |
| **Name of Tester:** | | Artem | | | **Date**: | |  |
| **Hardware Version:** | | 2.0 | | | **Time:** | |  |
| **Setup:** | | MKR1010 setup to stream MPU6050 data through the serial port. Leave MKR1010 running for a long duration. | | | | | |
| **Step** | **Action** | **Expected Result** | **Pass** | **Fail** | **N/A** | **Comments** | |
| 1 | Program MKR1010 with Wifi disabled, and serial data enabled. | Code should compile, Arduino should connect to wifi and start writing data. |  |  |  |  | |
| 2 | Run the program for 2 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. MKR1010 should not hang up losing communication with MPU6050. |  |  |  |  | |
| **Overall test result:** | | |  |  |  |  | |

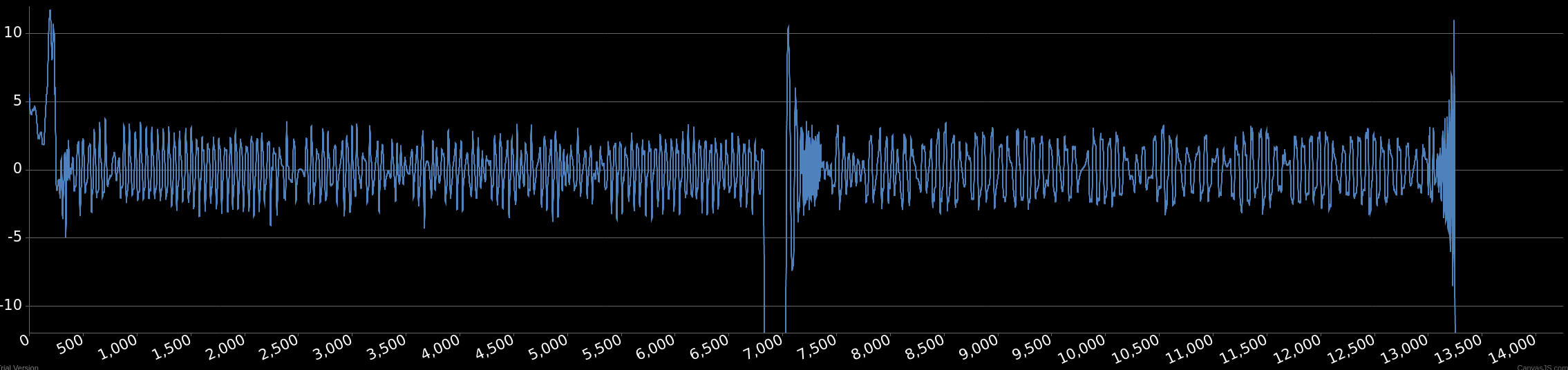
## **5.5 Parametric Test**

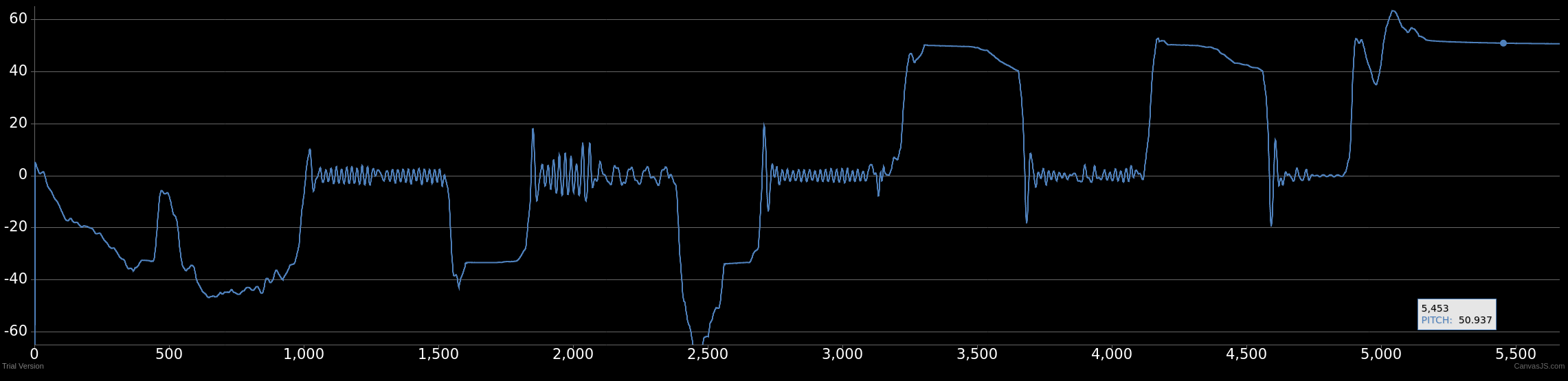
Verifying that our controller performs as simulated is a very important part of this project. This is a task in progress and is more detailed than what can be included in the tabular form. Separate documents will be written specifically for the verification of the controller.

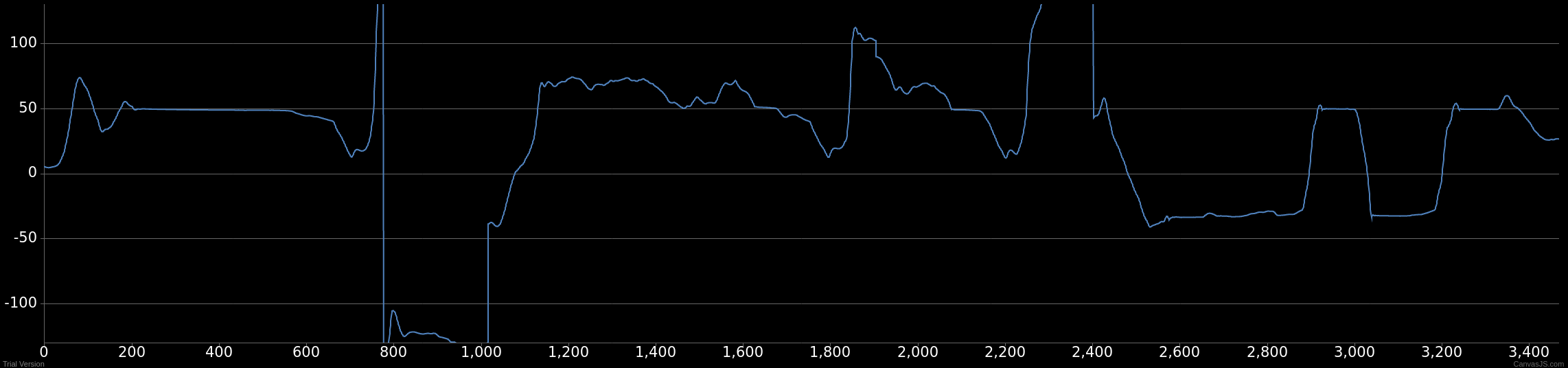
## **5.6 Acceptance Test**

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)

Figure 1: Tilt date for Fuzzy-I controller on the left, PID controller on the right after impact in the center.

Figure 2: PID controller recovering from laying on the floor poth in pooistive and negative.

Figure 3: Fuzzy controller failing to recover from laying down position. Usually overshooting the the other end.