Autonomous Interactive Robot

for Air Hockey (AIR)

Validation and Verification

SFWR ENG 4G06 / MECHTRON 4TB6

GROUP 3

Nima Akhbari

Daniel Chaput

Nicholas Cianflone

Michael Rowinski

Joshua Segeren

Evan Skeete

Contents

Revisions

Version	Date	Authors	Revision Description
0.0	2015-02-18	Nima Akhbari Daniel Chaput Nicholas Cianflone Michael Rowinski Joshua Segeren Evan Skeete	Initial revision.
0.1	2015-02-26	Michael Rowinski	Hardware validation added
0.2	2015-02-28	Nima Akhbari Evan Skeete	Change “Affine Transform” to “Perspective Transform”. Changed accompanying result from Fail to Pass.
0.3	2015-03-01	Daniel Chaput	Grammar and formatting for submission
0.4	2015-03-02	Michael Rowinski	Table of contents formatting Removed table of figures Grammar fixes
0.5	2015-03-02	Joshua Segeren	Software validation changes; grammar, formatting fixes
0.6	2015-03-02	Daniel Chaput	Final authorization and review before submission. Minor grammar.
1.0	2015-04-04	Michael Rowinski	Added Requirement ID column to the mechanical and electrical verification tables Updated test results to include additional detail
1.1	2015-04-05	Michael Rowinski	Functional Validation table updated to show the validation of requirements and specifications for the various AIR subsystems Added additional verification tests which were completed
1.2	2015-04-06	Evan Skeete Joshua Segeren	Added software requirements referencing; additional verification tests
1.3	2015-04-06	Nima Akhbari	Modified software requirements actual behavior
1.4	2015-04-06	Nima Akhbari Michael Rowinski	Added response times and max speeds the system actually obtains
1.5	2015-04-06	Joshua Segeren	Formatting, wording, light changes for R1 submission

1 Purpose

The goal of this project is to design and develop an autonomous robot capable of playing air hockey against a human opponent. This document provides an analysis of whether or not the various components of the project satisfy the outlined goals and requirements. This validation and verification of the system uses various test cases to confirm design decisions and determine that the current implementation is aligned with the overall project goals.

2 Validation

2.1 Project Goals

The comprehensive goal of this project is to design and develop an autonomous robot capable of playing air hockey against a human component. This requires design considerations in the fields of mechanical, electrical/hardware, and software engineering in order to implement a fully functional mechatronics system. To achieve this goal the design team must find a solution to object detection, tracking, and response problems in the most effective manner. The design process will achieve this by investigating, determining, and implementing subsystems based on best practices across machine vision, motion planning, predictive analytics, control systems, mechanical design, and hardware-software interfacing. Goals for validation and verification can be extended to both the hardware and software components of the AIR system.

Software components required in the design of the AIR system include:

Component	Goals
Object Detection and Tracking	– Detect/filter for puck and determine centroid – Track puck position and speed – Supply data to path planning and strategy selection
Strategy Selection	– Determine strategy appropriate for user – Use the selected strategy's path planning module
Path Planning	– Determine the best path for the robot mallet to take in both defensive and offensive positions (influenced by strategy selection) – Supply control data to hardware components – Receive actual position data from hardware components (semi-closed loop)

These systems are paramount to the effective functioning of the AIR system. Together, these subsystems dictate the functioning of the robot, and the interactions of these subsystems supply control commands to the robot (hardware). Design goals include validation of the decision to separate and focus on the development of these specific subsystems.

Proper functionality will also need to be verified using test cases to ensure the output given by the system matches expectations. Incorporating proper validation and verification will ensure the software components function appropriately and provide proper support to the overall AIR system.

Hardware components required in the design of the AIR system include:

Component	Goals
Mechanics Controller	– Control motor speed and timing through motor driver (includes error/safety handling) – Read movement data from path planning subsystem – Supply mallet position data to path planning subsystem
Mechanics Calibration	– Determine if any positional error has occurred and supply data to correct error to the mechanics controller if needed
Goal Detection/Human Machine Interface (HMI)	– Determine when a goal has been scored – Update scoreboard

These systems are the acting controller for the robot's functions. The software intelligence supplies the hardware components with data it uses to control the movement and position of the robot. The hardware components work together to ensure expected movement occurs, positional accuracy is maintained, and the user is supplied with relevant information. These components will need to be validated and verified to ensure that the actual outputs match expected outputs.

Safety of users and spectators is the highest priority in the development of AIR. The main goal is to provide a safe and fun environment where user injury is prevented. Using limit switches, the system will be protected from over traveling or breaking of key components with the overall goal of protecting the user. Validation and verification that these components are functioning correctly is crucial during the development of AIR.

Overall, the projects goals and requirements can be referenced in Group 3's project choice and goals document, as well as the system goals and requirements document. Validation and verification of the specific subsystems as well as any significant isolated components is an important element in the design process of AIR and in the subsequent sections, testing and methodology will be further elaborated.

2.2 Functional Validation

Goal	Characteristics and Functionality of Current Implementation	Specification and Requirement Completion
Puck Location Sensor	– Camera grabs frames at an average rate of 120 frames per second – Camera position is accounted for using Perspective Transform to focus image on the table, regardless of the camera angle/orientation – Color calibration enables improved accuracy when detecting the object – Object color is filtered; calculated centroid of detected object is given to the path planning/strategy subsystems	– Camera frame rate allows the AIR robot to respond within the necessary time as specified by requirement 7.1.1.1 – Perspective transforms, color and object filtering all provide mechanism to improve puck detection and tracking – Detection mechanisms allow positional errors to comply with requirement 7.1.1.3 and 7.1.2.1
Strategy	– Determine correct strategy to use from a programmed selection – Decide appropriate positional movement corresponding to enabled strategy	– Strategy determination and associated robot movements are made to comply with requirement 7.1.2.2 and 7.1.3.1
Mechanics Controller	– Interfacing between the motor driver (TB6600) and Arduino allows communication between components (motor timings and speed) – Serial connection between Arduino and CPU allows communication between software components and mechanics controller; this facilitates real-time (i.e. time-accurate) operation – Limit switches protect the Y-axis carriages from over traveling or damaging system/user by turning off motor enable-all signal on TB6600 thus stopping all robot motion	– Interfacing between the Arduino, CPU and motor driver is necessary to the operation of the AIR system – All components must operate in unison to achieve a product that is capable of playing air hockey – Limit switches provide system safety to comply with requirement 7.1.2.6
Mechanics Calibration	– Limit switches placed on both axes provide a hard feedback system in order to reset the software position of the mallet with the actual position of the mallet	– Due to slipping in the motors, the mallet position in the software would differ from the actual position of the mallet – Resetting the software position upon limit switch actuation decreases the

		positional error of the mallet to comply with requirement 7.1.1.3
Human Machine Interface (HMI)	– 7-segment display provides a large and clean display for the current game score for the user to see – Error code functionality not yet implemented	– The HMI display is made to comply with requirement 7.1.2.3

2.3 Behavioral Validation

Behavioral validation evaluates the AIR's ability to meet behavioral requirements. While functional validation helps ensure subsystem functionality, behavioral validation analyzes the system's ability to behave as expected overall. In the current development stage of AIR, the subsystems have not yet been fully integrated, but the behaviors of the separate software and hardware systems can be individually validated. Currently the hardware system functions as expected though some specific subsystems—such as the HMI and mechanics calibration—have yet to be implemented. Hardware can currently control the robot within its desired workspace at desired speeds, while the overall software system has implemented the vision system, strategy and path planning subsystems. As development for AIR moves forward, integration of these two subsystems will facilitate a more complete behavioral validation.

3 Verification

The verification process for the AIR system is an iterative process that will test both hardware and software systems thoroughly. Initially the system components will be isolated and tested for independent functioning, moving forward to integrative testing as development reaches the appropriate stages.

For hardware verification, individual components making up the system will be tested and verified for proper functionality. This includes testing the various electrical, mechanical, and control system components. If the proposed components are verified to be working as expected, testing will move forward to the overall hardware system. This will test explicit inputs and outputs to verify the hardware system works as expected and there are no issues. The hardware must be tested independently, at first, to minimize potential risk; any integrative tests of the system which includes software tests, first and foremost depends on properly functioning hardware for both system and user safety. Full verification of hardware is achieved when each individual component is verified, followed by verification of the overall integrated hardware system.

Software verification will mirror the same iterative process that hardware verification endured. Individual systems making up the software components will be tested using explicit input-to-output mapping in order to ensure proper functionality. After each subsystem in the software component is verified, the entire software component will be tested for proper functionality. During the design process an integrated simulation was developed to test that each software subsystem is working as expected. After verification of both the isolated and integrated software components is confirmed, the AIR system is ready for fully integrated testing.

The software and hardware systems of AIR can be tested in parallel, though as stated previously for system and user safety, hardware will need to be verified before any software can be integrated with it. With verification of both systems overall integrative testing can be done. At the current development stage of AIR, full integration has not been achieved. As development moves forward, test cases for the entire system will be added to ensure proper functionality.

3.1 Hardware, Electrical, and Mechanical Verification

3.1.1 Isolated Mechanical Component Verification

3.1.1.1 V-Slot Carriages

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.1.1.1	7.1.1.4	– Verify the correct mounting and motion of the main robot carriages	– Manual force to slide the carriage	– Smooth motion with minimal friction	– Behaves as expected	Pass
3.1.1.1.2		– Verify the robot workspace	– Manual force to move both robot axes	– Robot is able to reach 90 - 100% of desired workspace	– Robot is able to reach 100% of desired workspace	Pass

3.1.1.2 Robot Mallet Mounting Mechanism

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.1.1.1	7.1.3.1	– Verify the mallet mounts to the carriage appropriately	– Move mallet along mount	– Smooth vertical motion with no horizontal give	– Mallet mount only moved in the vertical direction	Pass
3.1.1.1.2		– Verify the mallet is able to contact the table throughout the workspace	– Move carriage and mallet over entire robot workspace	– Mallet slides up and down and remains in contact with the table throughout workspace	– No binding on the mallet mount and mallet always contacts table surface	Pass

3.1.2 Isolated Electrical Verification

3.1.2.1 Stepper Motors

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.2.1.1	N/A	– Verify that the motor is capable of rotating in both directions	– 10 second forward stepping sequence followed by 10 second reverse stepping sequence from the TB6600 motor driver	– Smooth rotation without slipping in both directions	– Motor was able to accelerate and rotate in both directions without slipping (finishing precisely where it started)	Pass
3.1.2.1.2	N/A	– Verify that the motor is capable of rotating to precise positions	– Stepping sequences to rotate the motor 10 revolutions forward then 10 revolutions in reverse	– Smooth acceleration in both directions to reach max speeds – No slipping and original position should be achieved at the end of the sequence	– Motors accelerated very smoothly to various max speeds as set in the software. Slipping only occurred at the speed limits of the motors	Pass

3.1.2.2 Power Supply

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.2.2.1	N/A	– Verify that the correct voltage is outputted	– 120 VAC	– Output 32 – 39 VDC	– Measured 32V – 39V depending on the potentiometer setting	Pass

3.1.2.3 Limit Switches

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.2.3.1	7.1.2.5 7.1.2.6	– Verify the correct functioning of the limit switches	– Supply the limit switches pull up resistor with 3.3V	– Voltage across the switch goes from 0V to 3.3V when pressed	– Behaves as expected	Pass

3.1.2.4 S-R Latch

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.2.4.1	7.1.2.6	– Verify the correct functioning of the latch	– High and low pulses to the S and R inputs	– Output Q remains high when S is set and only resets when S is low and R goes high	– Behaves as expected	Pass

3.1.3 Isolated Hardware Verification

3.1.3.1 TB6600 Motor Driver

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.3.1.1	7.1.1.3	– Verify that the driver is able to properly rotate the motor	– Stepper motor stepping pulses from Arduino to rotate 200 steps (1 rev)	– Stepper motor rotates without slipping	– Behaves as expected	Pass
3.1.3.1.2	7.1.1.3	– Verify that the driver is able to rotate the motor in both directions	– Stepper motor stepping pulses and directional pulse from Arduino for one	– Stepper motor rotates in one direction, stops then rotates in the opposite direction	– Motor rotated correctly in both directions without slipping	Pass

			revolution in each direction			
3.1.3.1.3	7.1.1.2	– Verify that the driver correctly accelerates and decelerates the motor	– Stepper motor stepping pulses from Arduino	– Stepper motors accelerate and decelerate without slipping	– Acceleration limits were determined where the motor would slip under acceleration. Software limits were installed to prevent slipping	Pass
3.1.3.1.4	7.1.2.6	– Verify the correct functioning of the enable-all input to the driver	– Stepper motor stepping pulses and toggling the enable-all input to ground	– Stepper motors should rotate when the enable all is grounded. No motion in the stepper motors when the enable-all is not grounded	– No current going to the motors was measured when enable-all was grounded	Pass

3.1.3.2 7-Segmented Display HMI

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.3.3.2	7.1.2.3	– Verify that the 7-segment display is working correctly and displays all necessary digits	– Arduino program that loads numbers 0-9 into each digit on the display	– All digits on the display correctly display the numbers 0-9	– Behaves as expected	Pass

3.1.3.3 Arduino UNO Microprocessor

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.3.3.1	N/A	– Verify that the Arduino functions correctly and is able to be programmed	– Program and flash the Arduino with a test program that proves functionality	– Upload program successfully and observe execution of the program on the Arduino console	– Arduino software displayed successful programming of the unit and test program to flash lights ran successfully	Pass
3.1.3.3.2	N/A	– Verify the functionality of the serial connection to the computer	– Send data in both directions (duplex) between laptop and Arduino	– The data sent and received matches on both the Arduino and computer	– Serial monitoring window displayed the correct values as sent to and from the Arduino	Pass

3.1.4 Integrated Hardware Verification

3.1.4.1 Arduino to Motor Driver Interface

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.4.1.1	N/A	– Verify the correct functioning of the Arduino to motor driver interface	– Use the Arduino to accelerate and decelerate all three stepper motors independently	– Motors rotate at the intended speeds and in the correct directions	– Arduino was able to get all three motors to accelerate, decelerate and rotate independently	Pass

3.1.4.2 Limit Switch Safety Circuit

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.4.2.1	7.1.2.6	– Verify that the limit switches terminate all robot motion when activated within a safe distance and response time	– Press the limit switch – Stepper motor pulses to rotate motors continuously in one direction	– When any limit switch is pressed all robot motion is terminated instantly	– The distance the robot moves after detecting the limit switch is minimal and deemed safe based on limit switch positioning and Hazard analysis	Pass

3.1.4.3 Robot X & Y Axis Movement

Test ID	Requirement ID	Description	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.1.4.3.1	7.1.1.4	– Verify that the robot mallet is capable of moving in the X/Y-Axis	– Use the Arduino to accelerate and decelerate the X/Y-Axis in both directions	– X/Y-Axis accelerates and decelerates smoothly without the timing belt or motor slipping	– The robot mallet was able to reach 100% of its workspace while always maintain contact with the table surface	Pass
3.1.4.3.2	7.1.1.2	– Verify that the X/Y-Axis is capable of reaching the required velocity	– Arduino program used to determine the maximum speed of the axis	– X/Y-Axis should move at minimum of 2 m/s with an ideal velocity of 6 m/s	– Max velocities in the X and Y axis were determined to be 2.4 m/s and 4.5 m/s, respectively	Pass

					This is the point at which slipping begins to occur during motion	
3.1.4.3.3	**7.1.1.2**	– Verify the maximum puck shooting speed	– Arduino program to accelerate mallet to max velocity to strike the puck	– A minimum puck velocity of 4 m/s after collision with the robot mallet	– No offensive strategy implemented at this time.	Fail
3.1.4.3.4	**7.1.1.1**	– Verify the average robot mallet response time	– Puck motion on the table	– Mallet should adjust positioning due to puck movement – Mallet should move no later than 190 ms after puck has been moved	– Mallet reposition after the puck moves into the playable workspace. – The mallet's response time is approximately 500 ms	Fail

3.2 Software Verification

3.2.1 Isolated Vision Software Verification

For each of the isolated vision tests (3.2.1.1.1 – 3.2.1.4.1), the relevant requirement is ultimately 7.1.1.3 – Combined Mallet Position and Puck Detection Accuracy. From a system-level perspective, the individual accuracies are not appropriate discussion parameters for requirements, but are all necessary to meet this global requirement.

3.2.1.1 Perspective Transform Accuracy

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail

3.2.1.1.1	– Four coordinates representing the table's corners	– Transforms image to focus solely on the table	– The four corners of the table are properly stretched and mapped to the window; full table is in view for vision processing	Pass

3.2.1.2 Color Calibration

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.1.2.1	– Array of pixels	– Calculates the Hue, Saturation and Value (HSV) range to filter	– The proper range is calculated, to filter the desired color	Pass

3.2.1.3 Color Filtering

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.1.3.1	– Matrix of pixels, color range	– Filters the proper color range	– The desired color, described by the given range, is filtered	Pass

3.2.1.4 Object Centroid Calculation

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.1.4.1	– Filtered image	– Locates proper object, and outputs its centroid	– The center of the largest object detected within the filtered image is being properly calculated and subsequently returned to the caller	Pass

3.2.2 Isolated Interfacing Software Verification

3.2.2.1 CPU to Arduino Serial Interfacing

There is no explicit requirement against which the CPU to Arduino Serial Interfacing can be referenced; however, it is indirectly part of 7.1.1.1 – System-controlled Mallet Response Time, since interfacing must be completed within each system-level (physics loop) timestep. However, the interface timing is not tested, since it is done over serial USB for which the communication time is negligible given the size of messages being sent here (480 Mbit/s for USB 2.0).

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.1.1.1	– Target position vector representing absolute position steps in X, Y-axes	– Position vector is sent and processed by Arduino	– Behaves as expected	Pass

3.2.3 Isolated Strategy Selection Verification

There is a single documented requirement which entails a combination of defensive and offensive actions, of naïve to advanced complexity for different game situations (Requirement 7.1.2.2 – Strategy); here, some subcomponents of the strategy layer are documented and evaluated based on qualitative intended behaviors.

3.2.3.1 Naïve Defense

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.2.1.1	– Puck position and velocity	– Within the simulation, moves the mallet along the Y-axis in-line with the incoming puck	– Behaves as expected	Pass

3.2.3.2 Triangle Defense

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.2.2.1	– Puck position and velocity	– Within the simulation, moves the mallet to the closest point on the edge of the defensive triangle	– Behaves as expected	Pass

3.2.3.3 Way-Point Offense

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.2.3.1	– Puck position and velocity	– Within the simulation, moves the mallet such that a straight line can be drawn from the mallet, through the puck and into the center of the opponent's goal	– Behaves as expected	Pass

3.2.4 Integrated Software Verification

3.2.4.1 Strategy Selection

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.3.1.1	– Puck velocity is greater than MAX_VELOCITY_TO_ENGAGE* – Puck is not within TRIANGLE_STRATEGY_DEFENCE_DISTANCE* from the goal	– Uses naïve defense	– Behaves as expected	Pass
3.2.3.1.2	– Puck velocity is greater than MAX_VELOCITY_TO_ENGAGE – Puck is within TRIANGLE_STRATEGY_DEFENCE_DISTANCE from the goal	– Uses triangle defense	– Behaves as expected	Pass
3.2.3.1.3	– Puck velocity is less than MAX_VELOCITY_TO_ENGAGE	– Uses waypoint offense	– Behaves as expected	Pass

*MAX_VEOLCITY_TO_ENGAGE and TRIANGLE_STRATEGY_DEFENSE_DISTANCE are constants that define position and velocity ranges that engage specific strategies

3.2.4.2 Strategy Performance

For strategy performance (non-functional), offensive (delivering 90% of shots to desired target) and defensive (defending against 90% of shots received) metrics were defined in 7.1.3.1 – Performance.

Test ID	Input	Expected Behavior	Actual Behavior	Pass/Fail
3.2.4.2.1	– Puck position and velocity, within threshold range of defensive response and speed greater than MAX_VELOCITY_TO_ENGAGE (i.e. defensive mode conditions)	– Prevents puck from being scored with 90% probability	– To be determined (TBD) via experimental testing	N/A
3.2.4.2.2	– Puck velocity is less than MAX_VELOCITY_TO_ENGAGE (i.e. offensive mode conditions)	– Deliver 90% of offensive shots/actions with respect to desired target	– To be determined (TBD) via experimental testing	N/A