Auto Laser Test Plan
Ian Schuster / James Garrett

Part 1: Description of Test Plan

Our project has two sides, a physical side which consists of servos and a laser connected to a raspberry pi, and a software side which consists of an AI which uses image recognition to find balloon locations. This will require three different testing styles, the physical side testing, the software side testing, and combined testing. The physical side tests will be using sample location data to ensure the servos can aim the laser properly, and that the laser will turn on and off properly. The software side testing will consist of sample images to ensure that the AI properly identifies balloons, and converts that location data into values the servos can use. Finally we will have combined testing which uses real time data and actually runs the system with both the physical and software sides combined to ensure the two interact as expected.

Part 2: Test Case Descriptions

PS1.1	**Physical Side Test 1 - Laser On**
PS1.2	Ensure that the laser can be powered on correctly.
PS1.3	This test will run the digitalWrite() command to set the power value for the laser to HIGH, and then will run the digitalRead() command to determine if the value was correctly set to HIGH.
PS1.4	Input(s): Pin number of the laser, “HIGH”
PS1.5	Expected Output(s): “HIGH”
PS1.6	Normal
PS1.7	Whitebox
PS1.8	Functional
PS1.9	Unit
PS2.1	**Physical Side Test 2 - Laser Off**
PS2.2	Ensure that the laser can be powered off correctly
PS2.3	This test will run the digitalWrite() command to set the power value for the laser to LOW, and then will run the digitalRead() command to determine if the value was correctly set to LOW.

PS2.4	Inputs(s): Pin number of the laser, “LOW”
PS2.5	Expected Output(s): “LOW”
PS2.6	Normal
PS2.7	Whitebox
PS2.8	Functional
PS2.9	Unit
PS3.1	**Physical Side Test 3 - Move Laser Up**
PS3.2	Ensure that the servos can move the laser upwards.
PS3.3	This test will run the read() command on the y-axis servo to get the angle of the shaft. Then it will run the write() command on the servo to set the angle to a greater value. Then it will run the read() command again on the servo to determine if the orientation equals the expected value.
PS3.4	Input(s): NA
PS3.5	Expected Output(s): True, indicating that the servo moved the direction we expected it to.
PS3.6	Normal
PS3.7	Whitebox
PS3.8	Functional
PS3.9	Unit
PS4.1	**Physical Side Test 4 - Move Laser Down**
PS4.2	Ensure that the servos can move the laser downwards.
PS4.3	This test will run the read() command on the y-axis servo to get the angle of the shaft. Then it will run the write() command on the servo to set the angle to a lesser value. Then it will run the read() command again on the servo to determine if the orientation equals the expected value.
PS4.4	Input(s): NA
PS4.5	Expected Output(s): True, indicating that the servo moved the direction we expected it to.
PS4.6	Normal
PS4.7	Whitebox

PS4.8	Functional
PS4.9	Unit
PS5.1	**Physical Side Test 5 - Move Laser Left**
PS5.2	Ensure that the servos can move the laser to the left.
PS5.3	This test will run the read() command on the x-axis servo to get the angle of the shaft. Then it will run the write() command on the servo to set the angle to a greater value. Then it will run the read() command again on the servo to determine if the orientation equals the expected value.
PS5.4	Input(s):
PS5.5	Expected Output(s): True, indicating that the servo moved the direction we expected it to.
PS5.6	Normal
PS5.7	Whitebox
PS5.8	Functional
PS5.9	Unit
PS6.1	**Physical Side Test 6 - Move Laser Right**
PS6.2	Ensure that the servos can move the laser to the right.
PS6.3	This test will run the read() command on the x-axis servo to get the angle of the shaft. Then it will run the write() command on the servo to set the angle to a lesser value. Then it will run the read() command again on the servo to determine if the orientation equals the expected value.
PS6.4	Input(s): NA
PS6.5	Expected Output(s): True, indicating that the servo moved the direction we expected it to.
PS6.6	Normal
PS6.7	Whitebox
PS6.8	Functional
PS6.9	Unit
S1.1	**Software Test 1 - Identify a Balloon Location**
S1.2	This test will ensure that the AI can identify a single balloon's location.

S1.3	This test will use a static image of a balloon via our camera system to simulate a basic scenario the system would find itself in and that the AI is functioning as expected.
S1.4	Inputs: Image of balloon via our camera.
S1.5	Expected Outputs: (x, y) coordinates of the balloon in the image.
S1.6	Normal
S1.7	Blackbox
S1.8	Functional
S1.9	Unit
S2.1	**Software Test 2 - Identify no balloons.**
S2.2	This will ensure the AI won't throw out an inaccurate guess if there is no balloon to be identified.
S2.3	This test will use a static image via our camera which does not contain a balloon to simulate a scenario where the system is active but not presented with a balloon target. This will ensure the AI is strict enough to not identify a non-balloon as a target.
S2.4	Input: image of an area with no balloon via our camera.
S2.5	Expected Output: null/falsy result indicating no balloon found.
S2.6	Abnormal
S2.7	Blackbox
S2.8	Functional
S2.9	Unit
S3.1	**Software Test 3 - Convert Image coords. Into servo rotation**
S3.2	This will ensure the AI can tell the servos how to move.
S3.3	This test will take balloon coordinate data from the AI and convert it into rotational commands for the servos so they can properly target the desired location.
S3.4	Input: Location data in format output by AI, bounds of image
S3.5	Expected Output: Servo rotational command equivalent to balloon location.
S3.6	Normal
S3.7	Whitebox

S3.8	Functional
S3.9	Unit
S4.1	**Software Test 4 - Multiple Balloons**
S4.2	This will ensure that the software can decide on which balloon to target first in the event that there are multiple balloons detected in an image.
S4.3	This test will use a confidence score for each balloon detection. The highest/most confident detection will be targeted with priority over the less confident ones.
S4.4	Input: Images that contain more than one balloon.
S4.5	Expected Output: List of confidence scores and servo rotational commands for each detection, ordered from highest to lowest confidence.
S4.6	Normal
S4.7	Whitebox
S4.8	Functional
S4.9	Unit
FS1.1	**Full System Test 1 - Run live scenario with 1 balloon**
FS1.2	This will ensure the systems functions as expected with a 1 balloon environment
FS1.3	This test will be run with a live environment with 1 balloon visible to the camera and the system should find the balloon and target it with the laser system.
FS1.4	Input: Environment with 1 balloon.
FS1.5	Output: The 1 balloon gets targeted by the laser system.
FS1.6	Normal
FS1.7	Blackbox
FS1.8	Functional
FS1.9	Integration
FS2.1	**Full System Test 2 - Run live scenario with multiple balloons**
FS2.2	This will ensure the systems functions as expected with a multiple balloon environment
FS2.3	This test will be run with a live environment with multiple balloons visible to the camera and the system should find the balloons and cycle through targeting them with the laser system

FS2.4	Input: Environment with multiple balloons.
FS2.5	Output: The laser system cycles through the balloons targeting each for a set period of time.
FS2.6	Normal
FS2.7	Blackbox
FS2.8	Functional
FS2.9	Integration

Part 3: Test Case Matrix

Test Case	Normal / Abnormal / Boundary	Blackbox / Whitebox	Functional / Performance	Unit / Integration
PS1	Normal	Whitebox	Functional	Unit
PS2	Normal	Whitebox	Functional	Unit
PS3	Normal	Whitebox	Functional	Unit
PS4	Normal	Whitebox	Functional	Unit
PS5	Normal	Whitebox	Functional	Unit
PS6	Normal	Whitebox	Functional	Unit
S1	Normal	Blackbox	Functional	Unit
S2	Abnormal	Blackbox	Functional	Unit
S3	Normal	Whitebox	Functional	Unit
S4	Normal	Whitebox	Functional	Unit
FS1	Normal	Blackbox	Functional	Integration
FS2	Normal	Blackbox	Functional	Integration