BALLSPOT

System Design Document

Version 0.1

12/08/2017

Document Number: <0.1>

Contract Number: <0001>

Table of Contents

1. Introduction

BallSpot is an autonomous robotic tennis ball retrieval system. Currently retrieval of tennis balls is performed manually by tennis enthusiasts who vary in age from 5 to 85. This task, which could be harmful to some participants due to fitness levels and age, would instead be performed by BallSpot.

This system would integrate into current tennis facilities and programs with no need for infrastructure or system changes.

1.1 Purpose of the SDD

The purpose of this System design document is to outline the plan and procedures of the project at hand. Our goal is to provide more than just a general overview development methodology for the reader to remove any obscurities that may be surrounding our overall plans. We want to provide enough specification for the project so anyone involved (i.e. the project manager, project team, and development team) in the future can easily replicate the project if needed.

2. General Overview and Design Guidelines/Approach

2.1 General Overview

Picking up tennis balls takes the fun out of playing tennis and is extremely tedious. In order to eliminate the need for an individual to have to remove these balls after a practice session, we propose the following solution. Our aim is to build a robotic tennis ball collector that will be able to detect tennis balls on an open court and retrieve these balls. The end product will be fully automated and need no supervision to perform this task.

2.2 Assumptions/Constraints/Risks

2.2.1 Assumptions

The robot will be in place upon the tennis court when the time comes for it to retrieve the tennis balls. The environment of the tennis court will remain static; weather should never be an issue. The final product should be able to operate without any human interaction and little to no supervision.

2.2.2 Constraints

Light conditions will be tuned to indoor lighting. The robot should carry its power internally.

2.2.3 Risks

When it comes to a robot that retrieves tennis ball on a tennis court, there shouldn't be any major risks that should be extensively covered. However, there is one hazard that may be worth mentioning. There will be moving parts and motors on the robot; such as the wheels and also on the front of the robot that will be doing the collecting of the tennis balls. There should be some sort of safety mechanisms put in place that will mitigate the possibility of injury. The robot will be designed to stop when there is an obstruction and only small ball-sized items will be able to make it into the opening for the ball intake.

3. Design Considerations

3.1 Goals and Guidelines

The overall goals for the robot are to move around a tennis court at a reasonable pace, reliably identify and move to a tennis ball, and to efficiently pick up and carry the balls. These will all be achieved ideally without any user input on the robot besides the initial start up.

3.2 Development Methods & Contingencies

The team will begin by building the actual body of the robot. One option that we may end up using is lego mindstorm. There are a lot of possibilities when it comes to lego mindstorm and it seems like it may be pretty promising. Once the actual body of the collector has been built, we're going to need some way for the vehicle to recognize the tennis balls on the tennis court so it knows where to go to retrieve them. We've decided image recognition was the best way for this to be accomplished. PixyCam seems like an excellent product and has shown some great results for tracking a tennis ball. Once the image recognition is trained to detect a tennis ball, correct programming will be required to successfully direct the robot to the ball and also how to retrieve it once it gets near. The actual retrieving mechanism to grab the tennis balls is still an obstacle. A few ideas for the collection of the tennis balls have been thrown around (such as two spinning wheels that essentially suck the ball in once it comes in contact, or an actual vacuum that sucks the ball through a tube). With these ideas have also come complications that may arise. We'll have to experiment with a few different options to find the one that will provide the most optimal results. The idea to build a remote controlled vehicle to begin has been suggested so we can get the mechanics down for exactly how the robot is going to be operated. Once this is completed, implementing a fully A.I. machine shouldn't be extremely difficult to accomplish.

3.3 Architectural Strategies

- *Use of a particular type of product (programming language, database, library, commercial off-the-shelf (COTS) product, etc.)*
  Python, Numpy, OpenCV, PixyCam
- *Future plans for extending or enhancing the software*
  Allow for ability for software to be used in other retrieval robots
- *User interface paradigms (or system input and output models)*
  The user should be able to give one simple command, whether it be a button (physical or virtual) or an audible command, and the robot runs.
- *Error detection and recovery*
  Software should detect hardware malfunctions (e.g. ball jams, stuck robot)
- *Control over a network*
  Allow for simple remote controls for testing and error handling
- *Generalized approaches to control*
  Reflex agent
- *Communication mechanisms*
  BlueTooth, Wireless, USB

4. System Architecture and Architecture Design

4.1 Hardware Architecture

Hardware implementation will be left to the discretion of the manufacturers.

4.2 Software Architecture

An operating system for the image processing and robot control (e.g. Linux, OpenCV, python).

4.3 Information Architecture

4.3.1 Records Management

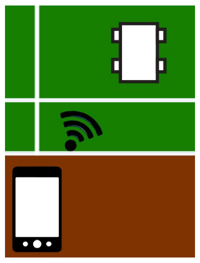
4.3.1.1 Data

Electronic data

- Image data
    - Light intensity
    - Color space
    - Resolution
- Location data
    - GPS
    - Compass
    - Localization
    - Image analysis

4.4 Internal Communications Architecture

The diagram below depicts the flow of data to and from the robot.



5. System Design

5.1 Business Requirements

An autonomous or semi-autonomous robot that picks up and retrieves tennis balls.

5.2 Database Design

If a database is utilized in the implementation of this project, it will be fully CRUD capable and standards compliant.

5.2.1 Data Objects and Resultant Data Structures

Containers for storing images

5.2.2 File and Database Structures

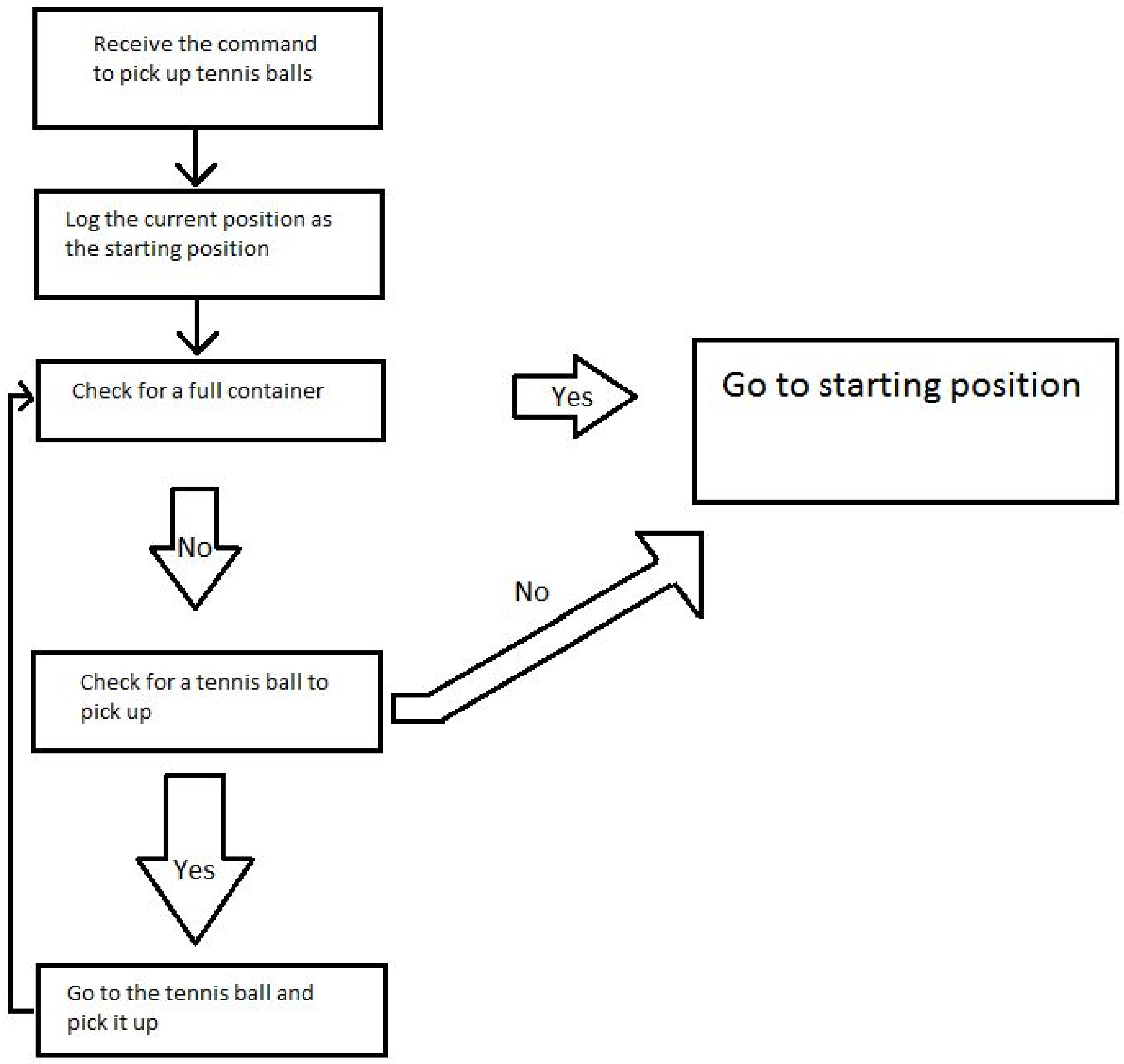
5.2.2.1 Non-Database Management System Files

- *Record structures, and data elements referenced within the records*
  An array of two ints for the starting position
- *Estimate of the file size or volume of data within the file, including overhead resulting from file access methods*
  8 bytes

5.3 User Machine-Readable Interface

The bot will either have a button to start the process, or it will get the command wirelessly from an app. Once the robot is done, it will send a message to a screen that it has finished.

6. Operational Scenarios


Receive the command to pick up tennis balls
Log the current position as the starting position
Check for a full container
Yes
Go to starting position
No
No
Check for a tennis ball to pick up
Yes
Go to the tennis ball and pick it up

This flowchart explains the operational scenario of the robot.

7. Detailed Design

7.1 Hardware Detailed Design

- *Power input requirements for each component*
  Power for the motors and for the processing unit

- *Signal impedances and logic states*
  Consideration for how materials affect signal transmission
- *Connector specifications (serial/parallel, 11-pin, male/female, etc.)*
  Ensure interoperability of all related components
- *Memory and/or storage space specifications*
  Ensure hardware will support the software that is required (e.g. Linux, OpenCV, python, Raspberry Pi)
- *Processor requirements (speed and functionality)*
  Ensure hardware will support the software that is required (e.g. Linux, OpenCV, python, Raspberry Pi)
- *Cable type(s) and length(s)*
  Ensure interoperability of all related components
- *User interfaces (buttons, toggle switches, etc.)*
  Ensure user interface is minimal and intuitive

7.2 Software Detailed Design

- *Service Identifier - The unique identifier and/or name of the software service*
  Linux, OpenCV, python
- *Classification - The kind of service (e.g., application, data service, etc.)*
  Application
- *Requirements - The specific functional or nonfunctional requirements that the service satisfies*
  Identify tennis balls, and control the robot

Appendix A: Record of Changes

Table 1 - Record of Changes

Version Number	Date	Author/Owner	Description of Change
<0.1>	<12/08/2017>	BallSpot	<Initial Setup>