# Executive Summary

# Introduction

## Background of Project & Problem Statement

Any sport requires practice if you want to improve your abilities. But also, it is important to practice efficiently: one does not become a weight champion by lifting pillows constantly nor starts training by lifting hundreds of kilos. Table tennis is no exception, although it is not always possible to train with opponents at the same skill level as you. As AntiaTech, we aim to remove this problem and make it possible for our customers to reach their potential as soon as possible with our product TrainingBuddy. TrainingBuddy allows you to set its difficulty level by giving you control of its spin, speed adjustment and creating ball routines to make sure you can prepare for any scenario! It will show you where your weakness resides and will be ready for whenever you are ready.

## Current Status

In order to achieve our goal of creating the perfect TrainingBuddy, we are marching in very solid steps. In two months we have created our prototypes for mechanical systems, detection and speech recognition systems, all in working conditions, seperately. Moreover, these subsystems are being exposed to tests and their weaknesses are analyzed so that we can improve them more and more. What's next, we conducted research on how we can create convenient user interfaces and control algorithms so that we can progress in a fast manner in the coming days.

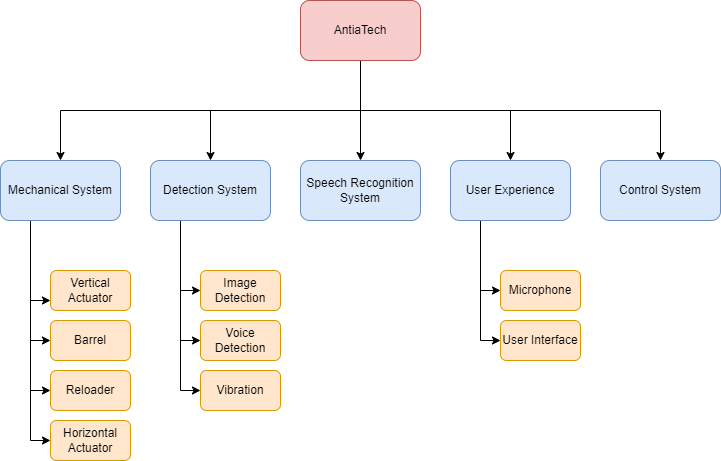
## Scope & Organization

This report contains a clear explanation of the problem (above) and contains this an that …

// TO DO

# Design Objective & Requirements

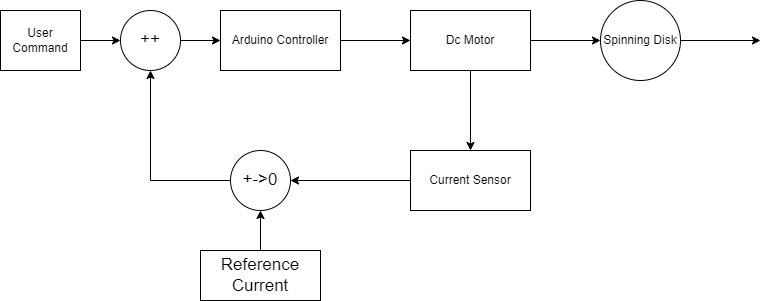
# System Design



### Mechanical Subsystem

The mechanical system is a system that includes all of the moving parts of the Training Buddy project. This system is made up of the reloader, barrel, horizontal actuator, and vertical actuator. The system's goal is to launch balls from a ping pong ball dispenser at various speeds and angles in different directions, based on user commands.

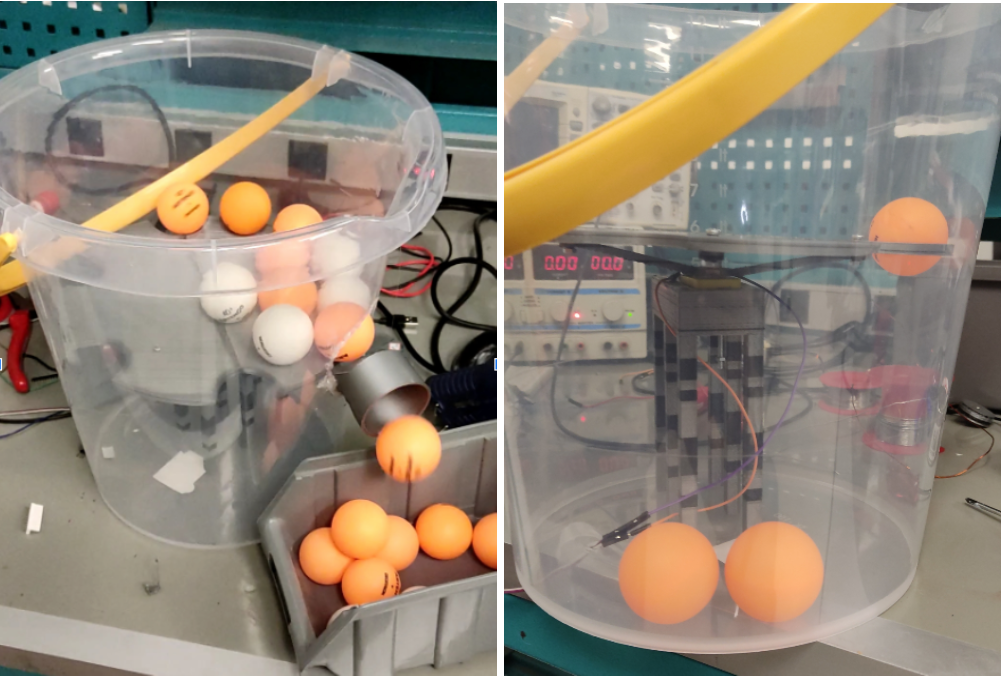
#### Reloader



The part of Training buddy that takes up the most space in terms of volume is the reloader. Reloader consists of a bucket and the mechanical parts inside. In addition, the empty spaces at the bottom of the bucket are intended to be used for placing other components.

In the simplest terms, the task of the Reloader is to push the balls into the barrel quickly or slowly according to the commands it receives from the user. After making some designs to achieve this goal, we made our first attempts by printing these designs on a 3D printer. Based on the bad results according to our test results, we will make some improvements.

At this point, we also created a plan B in case we could not solve the problems we experienced.



**Plan A:**

As seen in the pictures above, there is a rotating disk inside our bucket, and the motor and feet carrying this disk. This disc is intended to push the balls into the barrel as it rotates. The disc does this by squeezing the balls to the side of the bucket as it rotates. By looking at the results of our initial tests, we proved that we can push balls into a barrel with this method. However, we still have some problems.

1-) Our first problem is that the torque of the first engine we use is difficult to turn the bucket completely full. Since the motor rotates very fast, we have to operate it with low voltage and in case of any stuck, the motor cannot heal itself.

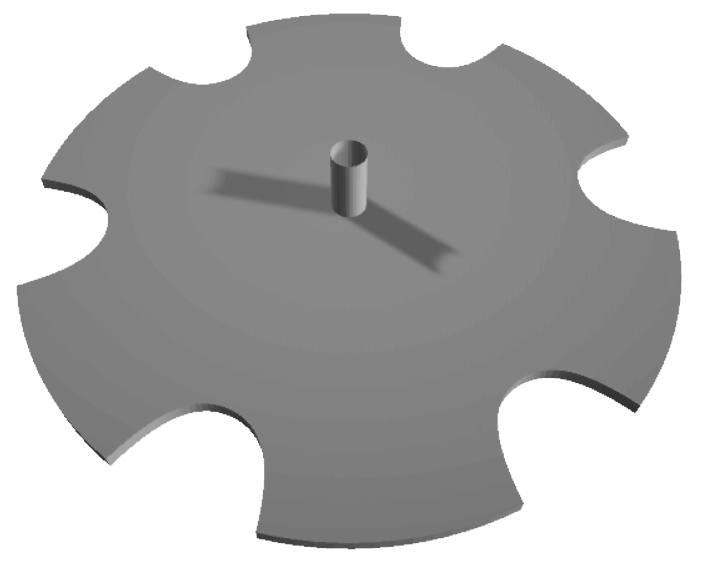
2-) In addition, while the bucket is rotating at low speeds, the balls can sometimes get stuck between the bucket and the barrel. We aim to solve this problem by using a higher torque motor. Also, we can detect a problem with current sensors. We noticed that a sudden acceleration and deceleration of the engine could solve this problem. Therefore, we will develop a healing mechanism based on this feedback we receive.

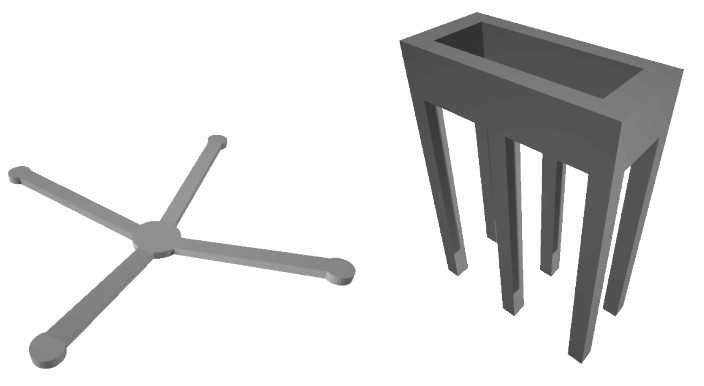
3-) Another problem that we are likely to encounter in the future is that the inclination caused by the vertical movements of the bucket prevents us from reloading the ball. We plan to use a curved disc to solve this problem.

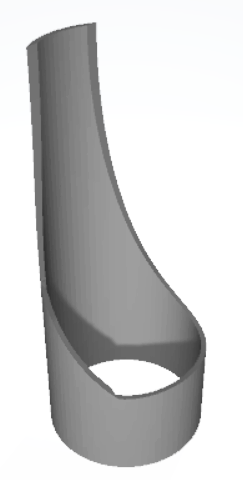
**Plan B:**

The most difficult issue to solve from the above-mentioned problems is that the balls get stuck between the barrel and the bucket. According to our tests, this jam is more likely to occur when there is more than one ball in the bucket. We have a plan to throw the balls one by one on the rotating disc in case the healing mechanism we mentioned above does not work efficiently. That means we'll have to add one more layer between the barrel and the balls.

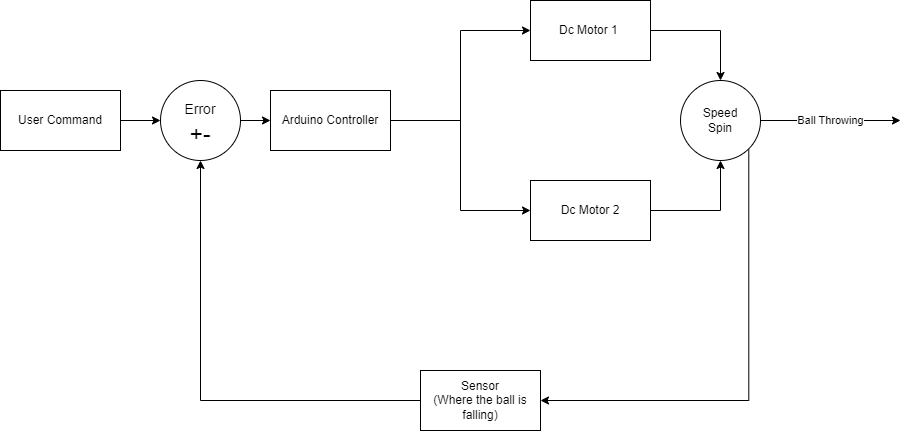
After replacing the engines and installing the feedback mechanism, we will discuss the possibility of realizing this plan by looking at the results of the tests we will do.







#### Barrel



The purpose of the Barrel System is to throw the balls from the reloader in the desired spin, either quickly or slowly according to the commands received from the user.

Our system consists of a barrel and rotating wheels facing inside this barrel. Our plan is to control the speed and spin of the ball by spinning these wheels at different speeds.

**Plan A:**

According to our first design, we have a barrel and 2 high speed rotating dc motors attached to this barrel. These motors will be compressed, accelerated and thrown when the ball touches the wheels while rotating at high speeds. We also expect the spin applied to the ball to change when the wheels spin at different speeds.

In order to see that this goal is achievable, we made a primitive design and printed our designs on a 3D printer. Next, we did ball launch tests and proved that we can throw the ball at different speeds. While conducting these tests, we considered the current values we applied to the motors. We observed how fast the ball went at different current levels.

Also, instead of using primitive methods to improve the accuracy of our tests, we enlisted the help of image processing technology. After watching the ball coming out of the muzzle with a camera, we were able to obtain the full velocity of the ball by converting frame/second > meter/second. As a result, according to our initial tests, our design worked as expected and the ball could be launched at the speed we wanted.

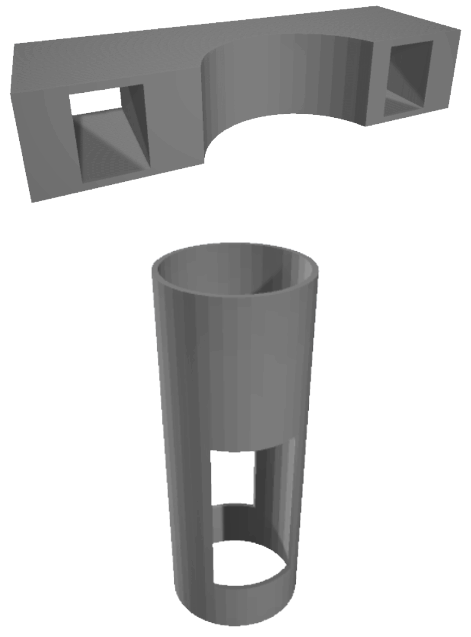
After successfully solving the issue of throwing the ball, we need to decide how fast the ball should be thrown according to the commands from the user. For this purpose, we will detect where the ball lands on the opposite table by using the image processing method. Next, we'll set up a feedback mechanism that slows the ball down if it's going fast enough to go off the table, and accelerates the ball if it's not going the other way. We will achieve this by changing the PWM values we give to the motors according to the feedback we get.

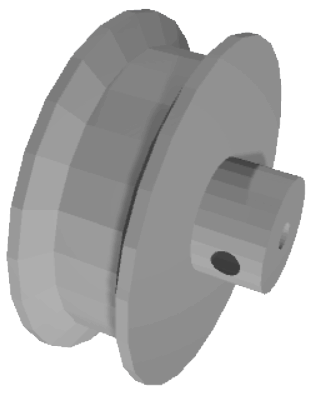
In our tests on spin, we did not get as good results as we expected. Especially the fact that the wheels we used did not stick to the balls statically well, failed us in giving spin to the balls. However, when we tracked the ball by measuring the horizontal deviation of the ball by rotating the barrel by hand, we saw that the ball had a certain amount of spin. In order to solve this problem completely, we decided to change the material we use in the wheels and use a softer plastic.

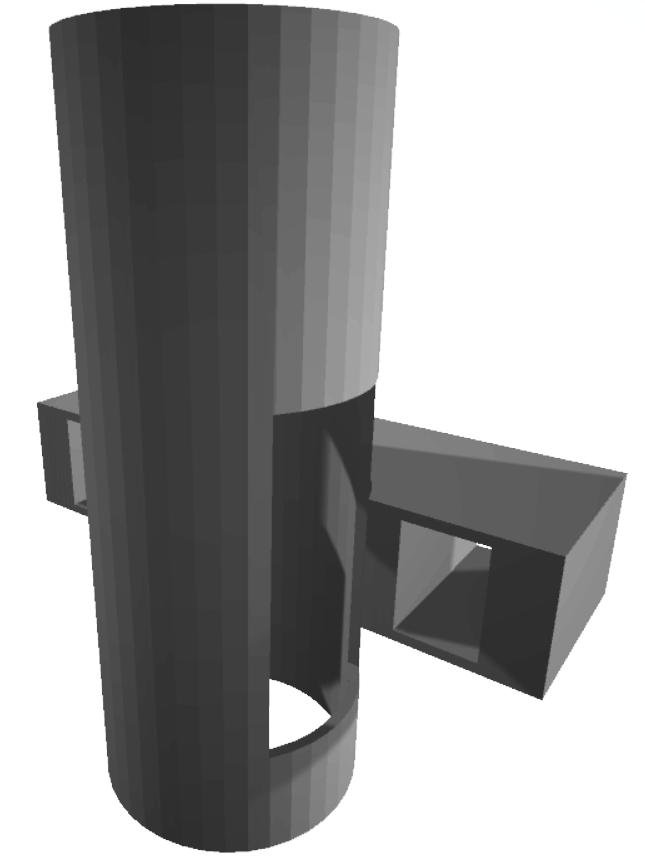
**Plan B:**

Among the things we have mentioned above, the design has not been finalized yet and the most important thing that may force us is where the ball lands on the opposite side. According to Plan A, we plan to throw the ball to the opposite side by processing images. However, since the Raspberry Pi processor will have other tasks, it can be overloaded while doing so much work. So we can solve this problem by putting vibration sensors on various sides of the table and installing a system that predicts the point where the ball will fall.

Other applications of the ball launch system will not be changed in Plan B as their viability has already been proven.



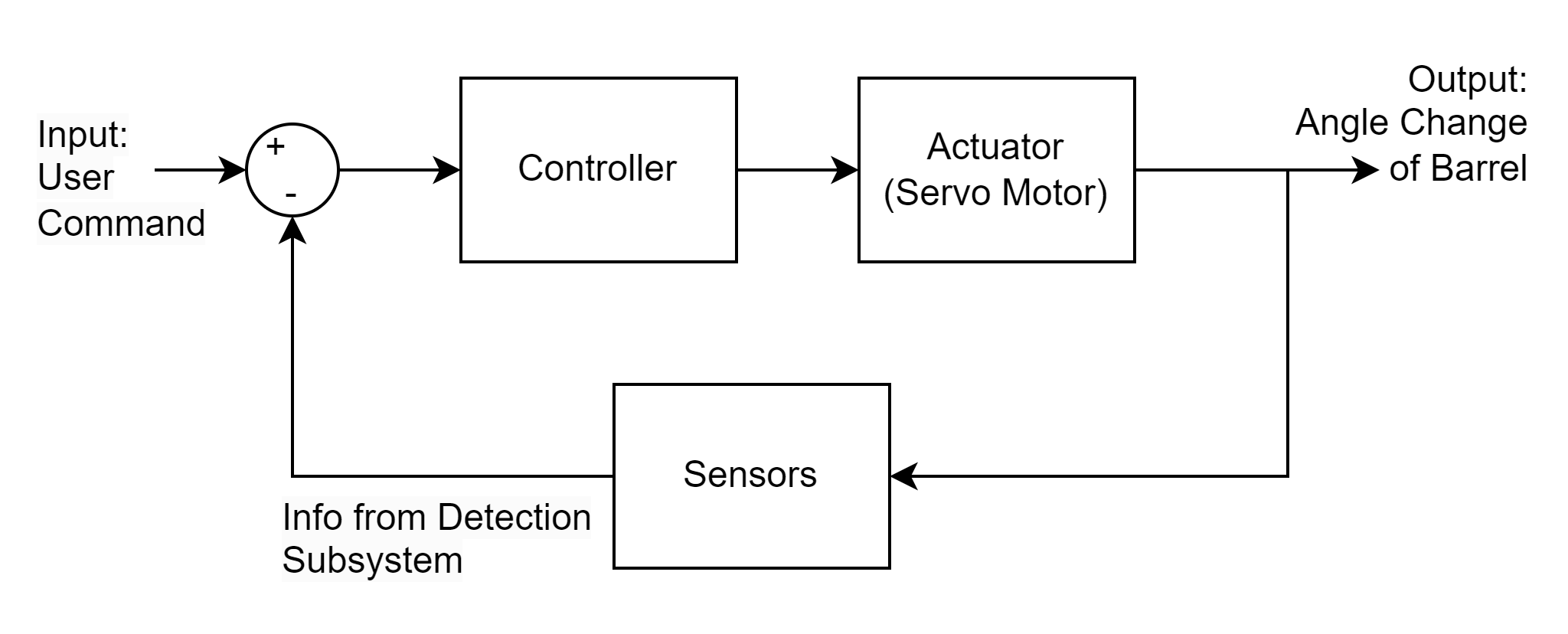




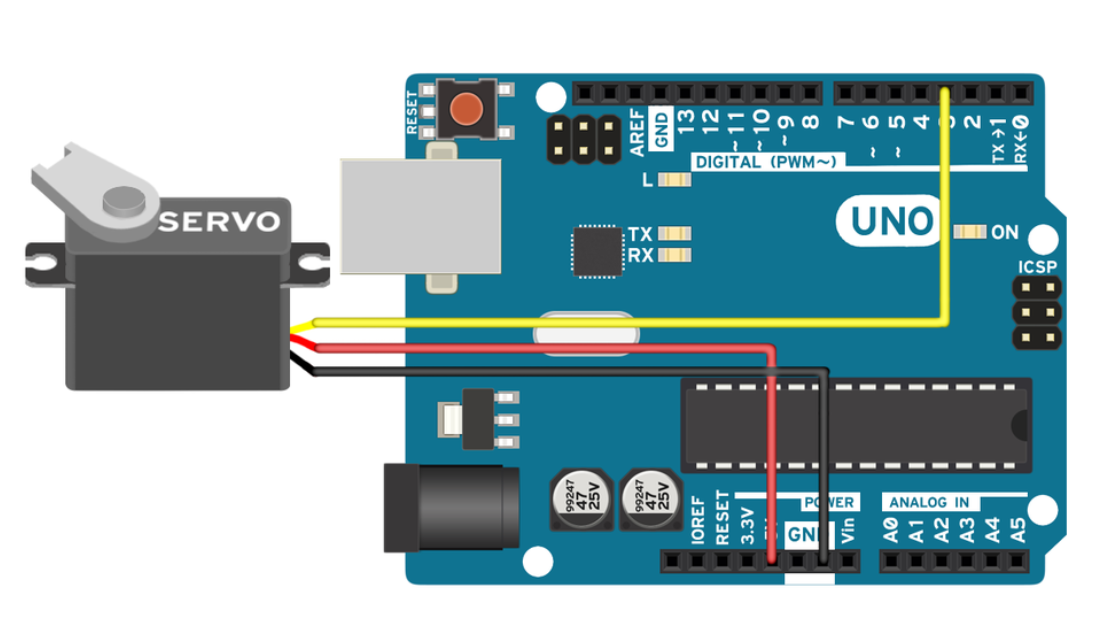
#### Horizontal Actuator

Horizontal actuation is done near the barrel system. This system takes the user commands as reference and changes the barrel's angle horizontally, in other words, it applies the panning to the ball launching. The panning angle is limited to 76 degrees, therefore it can be done by a servo motor.

The subsystem can also take input from detection systems, and change any launching angle error with control algorithms.



A servo motor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors. In our case, the controller is Arduino or Raspberry Pi, hence, there are a variety of servo motors in the market. Servo motor compatible with Arduino has everything built in: a motor, a feedback circuit, and a motor driver.



*Figure* .. Connections of Servo Motor-Arduino

In above figure, the connection between servo motor -and Arduino is depicted. Vcc and Ground connections are to supply power to the motor. Thus, the power consumption is very low for the component. Third connection is to control and give angle information to the motor via an analog pin.

**Controller Basics - Arduino Uno**

Arduino Uno is a microcontroller board based on the ATmega328﻿ by Atmel. The Arduino Uno pinout consists of 14 digital pins, 6 analog inputs, a power jack, USB connection and ICSP header. The utility of the pinout provides many different options such as driving motors, LEDs, reading sensors and more.

* There are 3 ways to power the Arduino Uno:

DC Power Jack (Barrel Jack) - DC Power Jack can be used to power the Arduino board. It is usually connected to a wall adapter. The board can be powered by 5-20 volts but the manufacturer recommends to keep it between 7-12 volts. Above 12 volts, the regulators might overheat, and below 7 volts, might not suffice.

VIN Pin - This pin is used to power the Arduino Uno board using an external power source. The voltage should be within the range same as in the Power Jack.

USB cable - when connected to the computer, provides 5 volts at 500mA.

* 5v and 3v3

They provide regulated 5 and 3.3V to power external components according to manufacturer specifications.

* GND

In the Arduino Uno pinout, there can be find 5 GND pins, which are all interconnected.

* Analog In

The Arduino Uno has 6 analog pins, which utilize ADC (Analog to Digital converter). These pins serve as analog inputs but can also function as digital inputs or digital outputs. Arduino Pins A0-A5 are capable of reading analog voltages. On Arduino the ADC has 10-bit resolution, meaning it can represent analog voltage by 1,024 digital levels. The ADC converts voltage into bits which the microprocessor can understand.

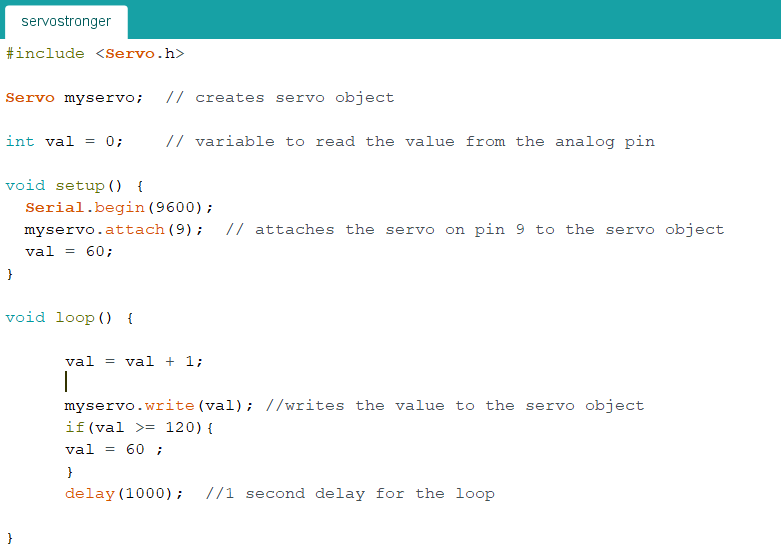
* Digital Pins

Pins 0-13 of the Arduino Uno serve as digital input/output pins. Pin 13 of the Arduino Uno is connected to the built-in LED. In the Arduino Uno - pins 3,5,6,9,10,11 have PWM capability. These pins can give analog outputs as well as digital.

* Serial (TTL) - Digital pins 0 and 1 are the serial pins of the Arduino Uno. They are used by the onboard USB module.
* SPI - SS/SCK/MISO/MOSI pins are the dedicated pins for SPI communication. They can be found on digital pins 10-13 of the Arduino Uno and on the ICSP headers.
* I2C - SCL/SDA pins are the dedicated pins for I2C communication. On the Arduino Uno they are found on Analog pins A4 and A5.

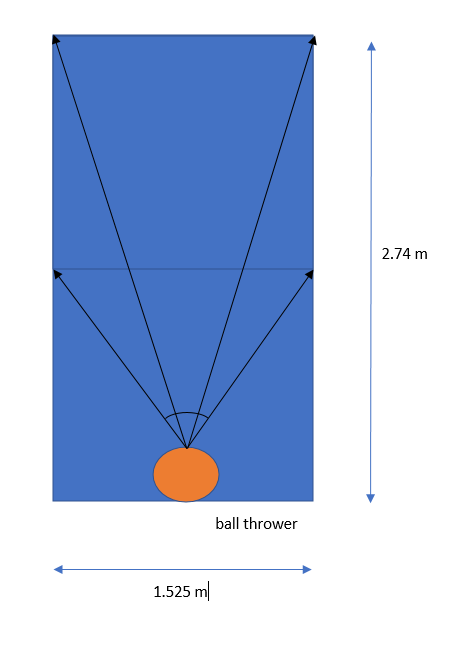
Afrduino is programmable with the [Arduino IDE](https://en.wikipedia.org/wiki/Arduino#Software) (Integrated Development Environment), via a type B [USB cable](https://en.wikipedia.org/wiki/USB_cable). In the IDE, programming language is C++, and there are libraries for external components, functions etc.

Below figure shows the Arduino code for servo motor driving. In the loop, the servo motor changes 1° angle every second between 60° and 120°.



*Figure* .. Code of Servo Motor-Arduino

In the project, panning is done by one servo motor (currently chosen as Tower Pro MG995 servo motor with 10 kg.cm torque and 0,20 sec / 60 ° angular speed) from the first part of the barrel. The attached servo pans the second part of the barrel system which contains the heavy side with two spin motors. Because of the weight of the second part of the barrel, servo is selected by considering the requirement of higher torque to move this part.

Sub-system requirements are to change the barrel’s angle in maximum 2 seconds, and give any angle between 52 and 128 according to current calculations. Servo motor speed is much higher than the required speed, and angle range is also wider. 

The overall design of this subsystem is shown in below figure.

//Buraya 3d çizim gelecek

#### Vertical Actuator

### Detection Subsystems

Purpose of the detection subsystem is making the decision on whether the player has successfully returned the ball (additionally whether the robot successfully launched the ball?). To achieve this, we have come up with 3 several solutions. Interestingly, instead of choosing one solution and improving only on the chosen solution, we are planning to implement all 3 solutions and make a decision at the end by combining the outputs of the 3 solutions. We choose this method since all the three solutions do not demand additional heavy resources such as time or money and therefore it doesn’t hurt us to implement all.

//TO DO: ADD OUTPUTS

*Image Detection*

Detection of a successful or failed return can be achieved by continuously tracking the ball with an image sensor. Assuming we have successfully implemented ball tracking software, we have the ability to extrapolate the data on locations where the ball currently is and where it has been.

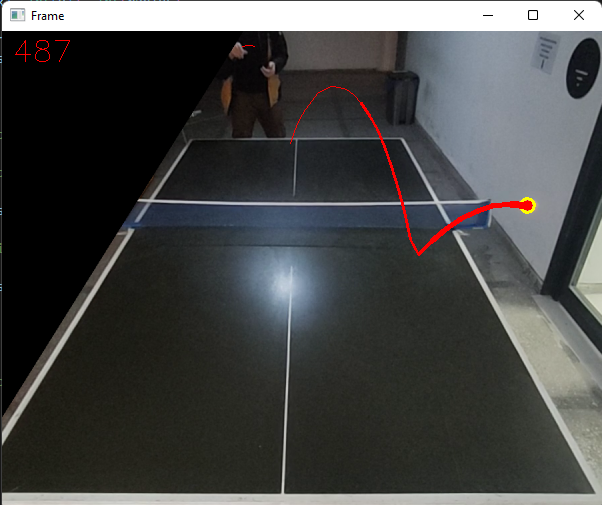


Figure X. Our prototype ball detection method.

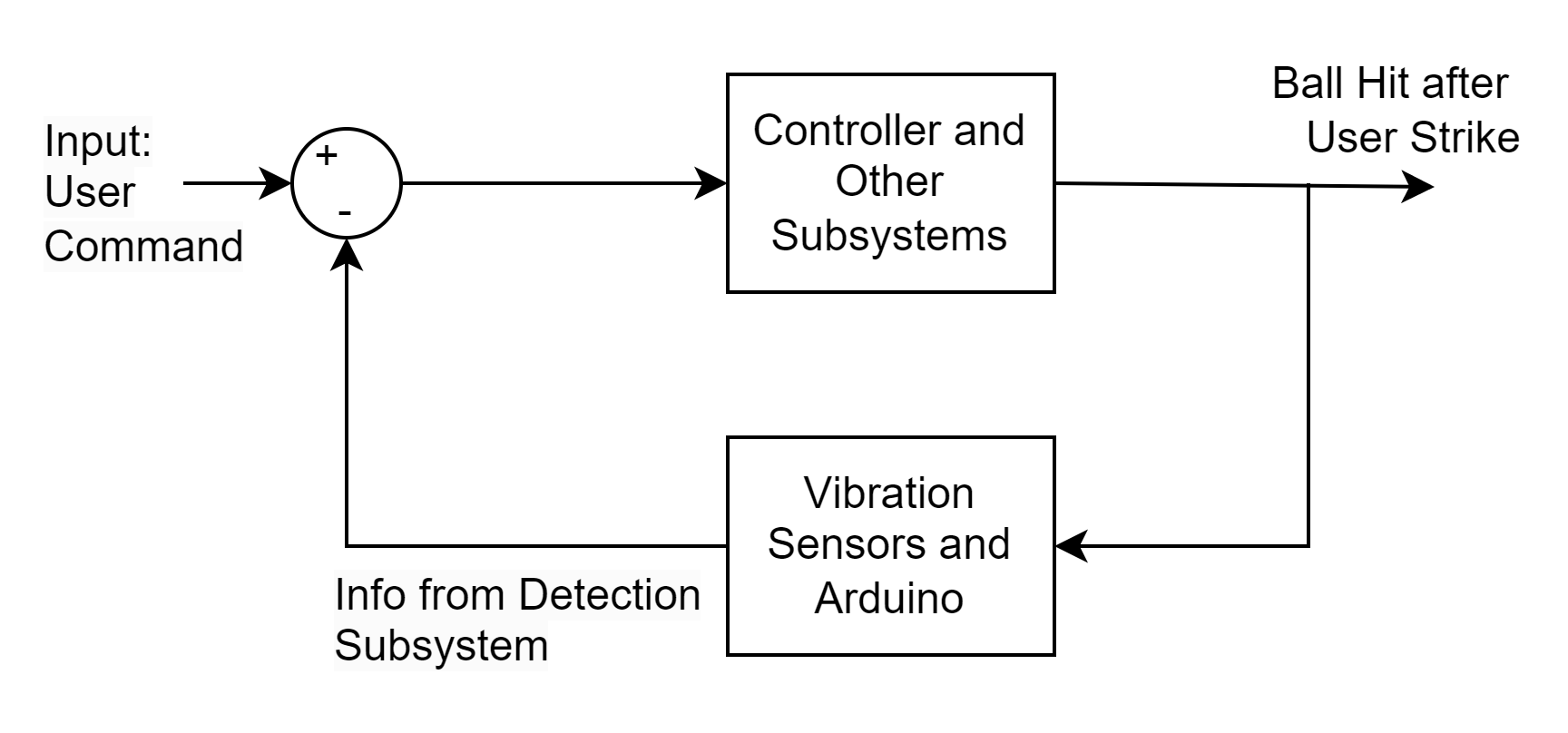
As seen on the figure above, when we can keep track of the path the ball has followed, it is easy to make assumptions on it’s action. In this case, we would like to detect whether it has hit the table, and whether it has hit the correct half of it. Clearly when the ball hits the table, it’s speed in x direction doesn’t change substantially. Its speed on the y-axis on the other hand, changes very abruptly, to the point it almost changes sign but holds the same magnitude. In short, we plan to detect return success by observing the sudden changes in y-axis speed of the ball.

#### Vibration

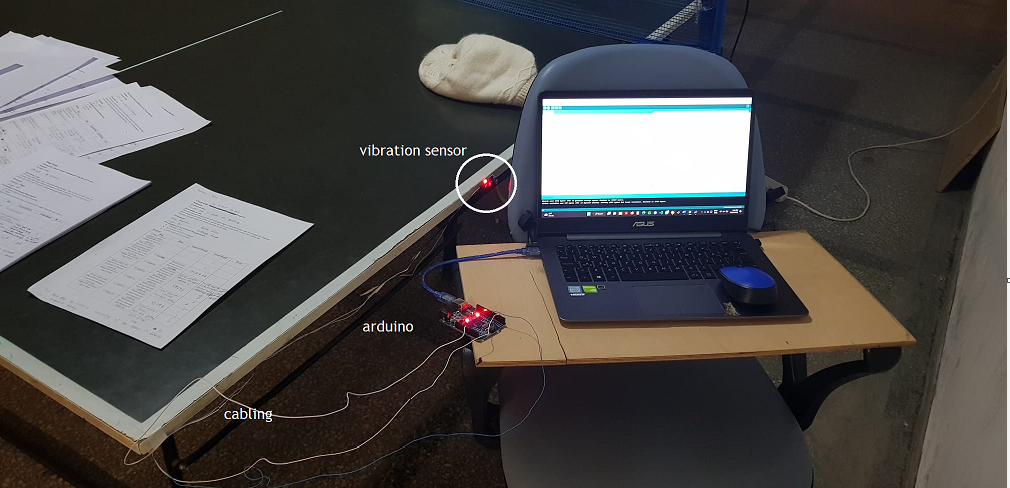
Detection of a hit can be done in many ways. When a ball hits the table after the user’s hit, it vibrates, and this impact can be measured with vibration sensors.

Vibration sensors are a type of sensor that can detect motion or changes in vibration. The working principle of a vibration sensor is that when a vibration or motion is detected, it produces an electrical signal related to the magnitude of the vibration.

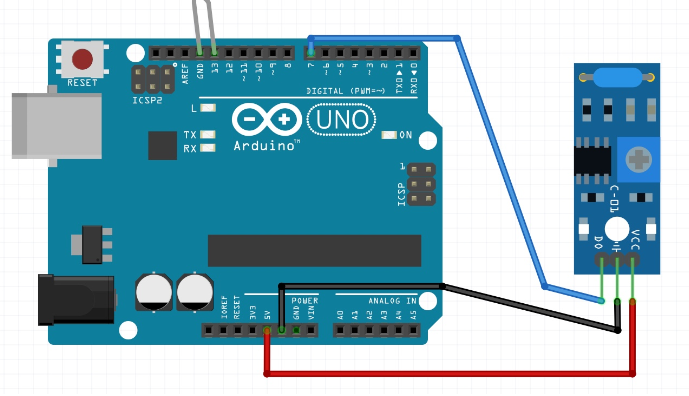
As can be seen in Figure A, detection information of the hit coming from the vibration sensors can be used as a raw data in image detection algorithms, or it can be a feedback to the launching correction algorithms. Horizontal, vertical angle changes, speed and frequency changes can be corrected according to the vibration magnitude and location detection of the ball hit on the tennis table.



In our solution approach, vibration sensors are placed near the tennis table as shown in Figure A. Two or four vibration sensors can be used for this purpose. The communication between the Arduino used for vibration sensing and Raspberry Pi can be done by cabling or wifi connection. Therefore, a wifi module can be adapted to the system in the later process.



In the subsystem, vibration sensors are connected directly to Arduino which the connection includes both data transfer and power supply as can be seen in Figure C.



Coding algorithm includes vibration sensor libraries and sensing functions. The location detection of the ball hasn’t been made with these sensors, however the subsystem have passed the test on hit detection by detecting the hits 99.99% of the time.

### Speech Recognition Subsystem

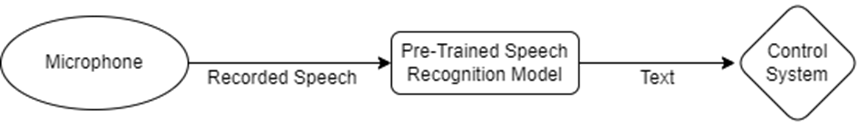
One of the main project specifications is that Training Buddy should be controlled by verbal commands, which makes the speech recognition subsystem a must. Speech recognition is not a new technology; it has advanced significantly in the past. Now, even simple devices used daily can understand human speech and convert it to text. Therefore, there are multiple ways this requirement can be fulfilled, online or offline. We plan to construct this subsystem based on two different approaches, one is online, and the other is offline. The end product's approach to performing speech recognition will be determined based on the performance comparison of those two approaches in the designed tests. We also consider using a hybrid method (online + offline) if the test results indicate a significant improvement in that case. Later in this part, the details about these two approaches and the envisioned algorithm of the hybrid method will be given.

The online approach we plan to implement requires not much processing power or storage capability but a good-quality stable internet connection. Because Training Buddy is a stationary project, providing a stable internet connection to the device is not difficult. As a result, we expect a good performance from this approach. The online engine we will use is the Google speech recognition engine, which can be easily accessed from the python speech recognition library over the internet. The block diagram representing the simple algorithm of this method is given in Figure A.

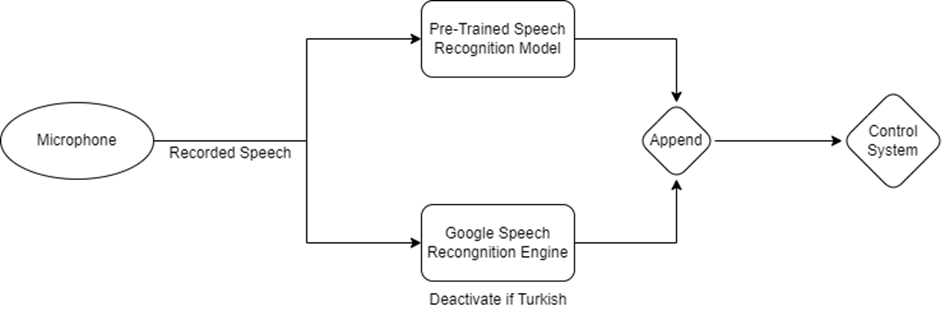


In this algorithm, the input sound to the microphone is recorded every 2 seconds and saved as a sound file. Then, these sound files are sent to the Google speech recognition engine and converted into text. These two processes work in parallel so that no command is lost during the processing time durations. That is, while the engine is processing one audio file, the microphone starts to record the next one. The sound files are removed after processing to achieve the minimum storage requirement.

The offline method is based on a speech recognition toolkit, Vosk. The advantage of this offline method is that it supports 20+ languages, including Turkish. We plan for Training Buddy to have a language option where the user can switch between English and Turkish. Unfortunately, the online method does not support languages other than English, but Vosk does. This method requires storing pre-trained speech recognition models in Raspberry Pi. We think Pi can easily handle it. Other than that, the algorithm for this method is not different from that of the online method. The main idea will be the same, but this time recorded sound files are processed by the pre-trained speech recognition model instead of the online engine. The block diagram representing the algorithm of this method is given in Figure B.



There is also a third design solution for the speech recognition subsystem: the hybrid model. The hybrid model is the integration of online and offline methods. We plan to construct and employ this model if the methods' performance alone is insufficient according to the test results. Moreover, if the performance of the online method is sufficient, but we decide on including the language option where the user can switch between Turkish and English, this model will again be necessary since only the offline method supports languages other than English. Figure C shows the block diagram representing the hybrid method's algorithm.



This is an envisioned algorithm and may change during the implementation period. However, it is basically based on the parallel operation of the online and offline methods. The sound file recorded by the microphone is fed to both the online engine and the pre-trained model. Then, the outputs of each method are compared in the block named "append." If the output texts are different, this block appends them. If not, it does nothing and gives one of the texts to the output. When the preferred language is set to Turkish, the online engine is deactivated since it does not support Turkish.

The method that will be implemented in the speech recognition subsystem (only offline, only online, or hybrid method) of the end product will be determined according to the results of the planned tests and our language setting decision. So far, we have implemented only the online method and performed its test.

### Ball Tracking Subsystem

Training buddy promises to its customers the analytics related to the training session. These analytics are measurements such as: speed, spin intensity , location where the ball has landed and the path it followed etc. To make the necessary measurements and obtain the analytics, we decided to create a program that tracks the ball movements and make necessary extrapolations so that the player can be informed about the gameplay and make necessary adjustments and improve his/her game. As a result of our researches, we summarized the possible algorithms that we can use in order to track the ball below:

#### Color based algorithms

Color-based object detection algorithms rely on the inherent color characteristics of the objects in an image to locate and classify them. These algorithms work by first defining a set of color thresholds or ranges that are used to segment the image into different colored regions. Then, the algorithm searches for connected components or clusters of pixels within these regions that meet certain size and shape criteria, which are used to identify and classify the objects in the image.

Color-based object detection algorithms can be effective in certain scenarios, such as when the objects to be detected have distinctive and unique colors, or when the background of the image is relatively uniform. However, these algorithms can be sensitive to lighting and other environmental conditions, and may not perform well in situations where the objects have similar colors to the background, or where the lighting is highly variable.

#### Shape Based Algorithms

Shape-based object detection algorithms rely on the inherent shape characteristics of the objects in an image to locate and classify them. These algorithms work by first applying image processing techniques to extract the shape features of the objects, such as edges, corners, and contours. The extracted shape features are then used to identify and classify the objects in the image.

Shape-based object detection algorithms can be effective in certain scenarios, such as when the objects to be detected have distinctive and unique shapes, or when the background of the image is relatively uniform. However, these algorithms can be sensitive to variations in the shape of the objects, and may not perform well in situations where the objects have similar shapes or where the shape of the objects is highly variable.

#### 

#### 

#### Feature based object algorithms

Feature-based object detection algorithms rely on the inherent features or characteristics of the objects in an image to locate and classify them. These algorithms work by extracting a set of features from the image, such as color, shape, texture, or depth, and then using these features to identify and classify the objects in the image.

Feature-based object detection algorithms can be effective in a wide range of scenarios, as they can be robust to variations in the appearance of the objects and the background of the image. However, these algorithms can be computationally intensive, and may require a large amount of training data to achieve good performance.

This algorithm branches into several different methods. Some are:

* Template matching
* Machine learning such as support vector machines (SVMs) or decision trees

Kalman Filter: This is a popular algorithm that uses a predictive model to estimate the state of an object over time. It is commonly used for object tracking because it can handle noisy measurements and predict the object's future location.

Particle Filter: This algorithm represents the object being tracked as a set of particles, each of which represents a possible location for the object. The particles are updated based on the measurements obtained from the tracking system.

Mean Shift: This is a non-parametric method that uses the statistical mode of the object's location to find the object. It works by shifting the object's location to the mode of the distribution at each time step.

CamShift: This is an extension of the Mean Shift algorithm that estimates the object's location and orientation based on the object's color distribution.

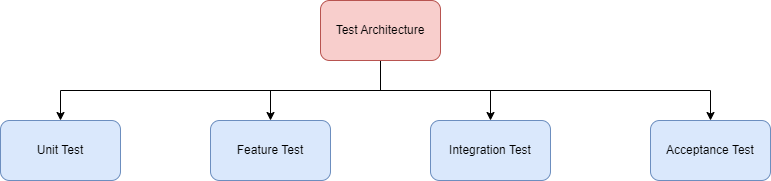
Optical Flow: This algorithm estimates the motion of the object by analyzing the changes in pixel intensity between consecutive frames.

Deep Learning-Based Tracking: These algorithms use convolutional neural networks (CNNs) to learn to track objects. They are usually trained on a large dataset of annotated images and can achieve very good performance, but require a lot of computational resources.

### User Interface

### Control System

# Test Architecture



## Unit Test

Unit tests test whether the smallest components of the system are working properly for their intended purpose. For example, the test that checks whether a dc motor works in suitable current and voltage ranges is a unit test.

## Feature Test

Feature test is a type of test that checks the functionality of the parts that make up the sub-system.

For example, barrel-related tests that are part of the mechanical subsystem are in the feature test category.

These tests are tested independently of other parts and subsystems. For example, there is no need to have a barrel in the mechanism that pushes the ball into the barrel. If the ball comes out of the bucket in the defined amount in a certain time, this test is successful.

## Integration Test

Integration tests check the integration of parts containing a subsystem with each other and the relationships of smaller subsystems with other subsystems.

For example, in a feature test scenario, the ball entering the barrel is pushed into the barrel by hand, while in the integration test, the throwing of a ball pushed into the barrel through the reloader mechanism is controlled.

In addition, the relationship of one subsystem to another subsystem can also be included in the scope of integration testing.

For example, a test case involving throwing the ball to the right or left with a voice command from the user is an integration test.

## Acceptance Test

The Acceptance test is the type of test that checks whether all subsystems work in harmony with each other.

The Acceptance test also includes a checklist. This checklist checks how well the final product meets the requirements.

## First Test Results & Comments

In the section below, you can see the first test results and relevant comments of our first prototype.

In addition, the test results of the features we consider most critical for our system are listed below in the charts.

| **Test Case** | **Result** | **Comment** |
| --- | --- | --- |
| Check if DC Motors Work Continuously with Rated Current | Fail | L298N DC Motor driver is heating. We’ll put a fan to solve this problem. |
| Check If Maximum Power Consumption of All Motor Units is In Proper Limits | Pass | - |
| Check If Mechanical Components Fits Our Purpose After Printing Them | Pass | - |
| Check if Servo Motor of Ball Thrower Can Work as Expected |  |  |
| Check if Step Motor of Ball Thrower Can Work as Expected |  |  |
| Check if Microphones Work Properly |  |  |
| Check if System Ball Tracking Can Work Continuously |  |  |
| Check if the Vibration Sensor Can Work Properly | Pass | - |
| Check if ball-thrower can throw ball with manually entered input speeds | Pass | - |
| Check if balls are sent to barrel with manually entered input speeds | Fail | The balls get stuck between the bucket and the barrel. |
| Check if Barrel and Motors Are Connected Properly | NA |  |
| Check if Ball Thrower Changes Horizontal Angles Correctly |  |  |
| Check if Ball Thrower Changes Vertical Angles Correctly |  |  |
| Check if System Can Detect the Ball Accurately |  |  |
| Check if System Can Track Ball Accurately |  |  |
| Check if Raspberry Pi Understands the Verbal Commands |  |  |
| Check if ball-thrower mechanism with all components can work properly |  |  |
| Check if immediate photo capture is possible by signaling |  |  |

## Critical Test Results

### Ball Speed w.r.t Different Current Rates

### Reloading Rate w.r.t Different Spinning Disc Speed

### Temperature of Heatsink w.r.t Different Current Rates for 1 Minute of Motor Driving

# Planning

## Schedule

## Risk Analysis

# Conclusion

# 

# References

<https://www.circuito.io/blog/arduino-uno-pinout/>

<https://www.instructables.com/Arduino-Servo-Motors/>

https://github.com/alphacep/vosk-api/