NYU TANDON SCHOOL OF ENGINEERING

General Engineering Department

Accushot – Pinnacle Sports Performance

Final Design Report

Project Team Members:

Alex Arora

Christina Curry

Andrew Hampton

Elijah Ager Luckett

Table of Contents

1. INTRODUCTION

1.1 Purpose of Project

Accushot is a sports simulator that allows users to practice soccer, baseball, and lacrosse. Accushot uses a combination of cameras and projectors to simulate and capture data, to enhance player performance. Accushot uses image processing to to calculate velocity position and accuracy. Additionally, Accushot keeps track of your performance over time so you can see your improvement. The simulator can be used to practice or to simulate real game time situations.

1.2 Background of the Experiment

Accushot was created out of the issue of not being able to effectively practice specific game situations. The idea was to create a simulator capable of tracking real time information and relaying it to the user like golf simulators on the market today with the added versatility of multiple sports. Some key components to the project included image processing, microcontrollers, and coding.

Image processing is a method of converting an image into a digital form then performing operations on it, to get an enhanced image or to extract information from it (Mary 2011). Accushot takes video of the of the ball as it flies towards the target. Using image processing the computer can locate and track the ball. From this the computer can do calculations to get speed, accuracy, and projected distance.

All the computing work done by Accushot is done through microcontrollers. Microcontrollers are small computers that often preform only one task and run one specific

program (Brain 2000). The program is stored in read-only memory and generally does not change (Brain 2000). Accushot uses an Arduino and two Raspberry Pi microcontrollers. The main difference of between the two microcontrollers is processing power. Due to Arduino's lack of processing power it is used in conjunction with a sensor to record the end time. The superior computing power of Raspberry Pi allows the system to handle multiple tasks, including image capture, image processing, and computations.

The coding language used by Raspberry Pi, and subsequently Accushot, is Python. Which is an object-oriented, high-level programming language used primarily for web and app development (Python for Beginners 2019). The other programming language used, to a much lesser extent, is specific to Arduino. Arduino IDE runs off a modified C++ code, another common programming language. In the case of python and Arduino vast libraries of code exist that decrease the time to create working code.

2. REQUIREMENTS

2.1 Physical Components

There are four major subsets that the physical components of Accushot fall under, images, processing, set up and display. Accushot utilizes two 5-megapixel cameras to capture the balls flight path and movement. Two Raspberry Pis handle the bulk of the processing with help from one Arduino. The Arduino relays data from the force sensor to the Raspberry Pi. Pieces used to set up the simulator include: a 3D printed ball stand, a camera mount and tarp. Display is also the target. A projector is used to project a target on the tarp.

2.2 Software Components

The software used in final design was the Raspberry Pi OS, Arduino IDE and the Python IDE.

3. PROCEDURES

3.1 Physical Construction

The physical aspects visible to the user is the ball stand, a camera box on the side, and the tarp. The tarp is 2.44 x 1.83 meters, and for our image processing code, must be green. The setup puts the tarp at 2.5 meters away from the ball stand and the user. The side camera is placed halfway between the user and the tarp, at 2.5 meters, positioned so the camera is facing the action from the side of the area it is happening in.

There are three separate components of circuitry: the main Raspberry Pi, the Raspberry Pi for the side camera, and the Arduino RedBoard for the Piezoelectric sensor. The main Raspberry Pi circuit contains the button, a receiver, and the front-view camera. The receiver will take in data from the side camera. The front view camera will capture the ball's motion from a head-on angle, assisting in collecting the accuracy data. The wiring is housed in the 3D printed ball stand, with the button exposed for the ball to press it when on the stand, and start the time once the button is released, meaning the ball has left the mount. The ball stand is a trapezoid shape from a side viewpoint, with a top side length of 7 centimeters, a bottom side length of 14 centimeters, and a height of 5 centimeters. The mount for the ball is a hole, which acts similarly to a tee, has a diameter of 2.5 centimeters. It is able to fit the button of a 2.25 centimeter diameter due to a 5 millimeter slanted fill cut. The top face of the stand is the 7 centimeters of the top side length by

a 5-centimeter width. The front of the box contains a hole that is 8 millimeters in diameter for the lens to have the tarp in view. This front face of the box is 5 x 7 centimeters. The bottom of the stand also includes four rings on the bottom, so the stand can be mounted into the ground using screws or pins.

The second Raspberry Pi only held the side camera and a transmitter to send information to the main Raspberry Pi. This component is placed in a 3D printed protective box that uses a hole of 8 millimeters in diameter for the lens to capture the motion of the ball from a side angle.

Finally, the Arduino RedBoard was used for the piezoelectric sensor that is attached to the back of the tarp. When it feels a vibration on the tarp, which would be the signal the ball has hit, it will send a signal to the main Raspberry Pi to stop the time.

3.2 Software Setup

Overview:

The main file for the software initializes the arduino piezo through bluetooth and the pushbutton connected directly to the raspberry pi. The bluetooth communication has been a little troublesome and hard to figure out but the circuitry code is straightforward and loops until an input is given. Then using the data from the push button both cameras are started using the pi camera library. Then the video is ended and saved using the piezo button hit. Then using the image processing code to track the ball in the video a coordinate list is developed. This image processing software breaks down the video frame by frame finding the colored ball and records its coordinate. The next part of the main code processes these coordinate lists to be useful. The side cameras coordinate list is manipulated to get the velocity and projected distance of the

ball. The front camera is used to see how close the ball gets to the target on the tarp. This accuracy data is recorded on to a text file and then all of the data is relayed back to the user.

Code:

Main:

```
#import all important libraries

import pushbuttoncode.py
import imageprocessing.py
import piezo.ino
import datetime
import dataprocessing.py
from picamera import PiCamera
push=pushbuttoncode() #start looking for when pushbutton is unpressed
with picamera.PiCamera() as camera and picamera.PiCamera() as camera1:
    if push==True: #When this is true start the recordings and find the start of time
        startTime=imageprocessing.StopWatch.start()
        camera.start_recording('sidevid.h264')
        camera1.start_recording('frontvid.h264')
     if piezo()==1: #when the arduino piezzo is hit stop recordings and end time
        endTime=imageprocessing.StopWatch.stop()
        camera.stop_recording()
        camera1.stop_recording()
coor_lst_side=imageprocessing.imageprocessing('sidevid.h264') #find coordinatlists for balls
#trajectory in each video
coor_lst_front=imageprocessing.imageprocessing('frontvid.h264')
#each of the rest of these codes are just calculating important information detailed in variable
names
horizontal_vel=dataprocessing.horizontal_vel(coor_lst_side, startTime,endTime)
vert_vel= dataprocessing.vert_vel(coor_lst_side, startTime,endTime)
projected_dist=dataprocessing.projected_dist(coor_lst_side, startTime, endTime)
target_loc=imageprocessing.target_location('frontvid.h264')
accu=dataprocessing.accu(coor_lst_front, target_loc)
avg=dataprocessing.write_to(accu)
#return information to user
print("Great shot! Your shot had a horizontal velocity of:",str(horizontal_vel),"A vertical velocity
of:",str(vert_vel),"Would have gone this far:",str(projected_dist),"Had an accuracy
of:",str(accu),"had an average accuracy of:",str(avg))
```

Image Processing:

```
import cv2
import imutils
import argparse
#import all necissary image processing imports and pins for raspberry pi

def imageprocessing(getvid):
```

```
    ap = argparse.ArgumentParser()
    ap.add_argument(getvid, "--video", required=True,
        help="get vid")#add argument for video
    args = vars(ap.parse_args())

    redupper=(20,90,50) #found by messing with http://colorizer.org/
    redlower=(0,55,25) #these are also hfv. Upper bound and lower bound of the search for ball.
    #Set up argument to grab video data
    vs = cv2.VideoCapture(args["video"])
    time.sleep(2.0)
    coordlst=[] #set data type where coordinates are going

    while True:
        frame=vs.read() #read frame by frame

        frame = imutils.resize(frame, width=600) #probably dont want to resize
        blurred = cv2.GaussianBlur(frame, (11, 11), 0)
        hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV) #make it easier to process

        mask = cv2.inRange(hsv, redLower, redUpper)
        mask = cv2.erode(mask, None, iterations=2)
        mask = cv2.dilate(mask, None, iterations=2)
        #create a way to easily and quickly find red and remove any errors

        cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE) #make sure works for all versions of cv
        cnts = imutils.grab_contours(cnts)

        center=None #initialize the center

        if len(cnts) > 0:
                # find the largest contour in the mask, then use
                # it to compute the minimum enclosing circle and
                # centroid
            c = max(cnts, key=cv2.contourArea)#largest contour area
            ((x, y), radius) = cv2.minEnclosingCircle(c) #initialize location of smallest circle that can
#completly fit over object
            M = cv2.moments(c)#places this in a moment dictionary for xy
            center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))#centroidx=(M10/M00)
centroidy=M1/M0
        coordlst.append(center)#put that in as a list item
        if coordlst[len(coordlst)-1][0]>=1.05*(coordlst[len(coordlst)-5][0]:
            break

    vs.release() #end video processing
    cv2.destroyAllWindows() #shut down cv
    return coordlst #return list

def target_location(getvid): #same as image processing except for finding the target on the tarp
    ap = argparse.ArgumentParser()
```

```
    ap.add_argument(getvid, "--video", required=True,
        help="get vid") #add argument for video
    args = vars(ap.parse_args())

    greenupper=(20,90,50) #found by messing with http://colorizer.org/
    greenlower=(0,55,25) #these are also hfv

    vs = cv2.VideoCapture(args["video"])
    time.sleep(2.0)
    frame=vs.read()
    frame = imutils.resize(frame, width=600) #probably dont want to resize
    blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    mask = cv2.inRange(hsv, greenLower, greenUpper)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE) #make sure works for all versions of cv
    cnts = imutils.grab_contours(cnts)
    center=None #initialize the center

    if len(cnts) > 0:
                # find the largest contour in the mask, then use
                # it to compute the minimum enclosing circle and
                # centroid
        c = max(cnts, key=cv2.contourArea) #largest contour area
        ((x, y), radius) = cv2.minEnclosingCircle(c) #initialize location of smallest circle that can
completly fit over object
        M = cv2.moments(c)#places this in a moment dictionary for xy
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"])) #centroidx=(M10/M00)
centroidy=M1/M0
    vs.release()
    cv2.destroyAllWindows()
    return center

class StopWatch: #determine the time passed using this class
    def __init__(self):
        pass

    def start(self):
        """Starts the timer"""
        self.start = datetime.datetime.now()
        return self.start

    def stop(self, message="Total: "):
        """Stops the timer.  Returns the time elapsed"""
        self.stop = datetime.datetime.now()
        return message + str(self.stop - self.start)
```

Data Processing:

```
#all functions needed to calculate important info to relay to user
import math
def horizontal_vel(coor_lst,startTime,endTime):
    start_coor=coor_lst[0][0] #first coordinate of side ball
    end_coor=coor_lst[len(coor_lst)-1][0] #last coordinate
    tot_dist=end_coor-start_coor #find pixel distance between
    foot_dist=tot_dist*0.0008680556 #convert to feet
    elapsed_time=endTime-startTime
    res=foot_dist/elapsed_time #find average velocity using time
    return res

def vert_vel(coor_lst, startTime, endTime): #same as horizontal velocity function but with two
#points close to each other at the beginning kick.
    start_coor=coor_lst[0][1]
    end_coor=coor_lst[5][1]
    tot_dist=end_coor-start_coor
    foot_dist=tot_dist*0.0008680556
    elapsed_time=5*((endTime-startTime)/len(coor_lst))
    res=foot_dist/elapsed_time
    return res

def projected_dist(coor_lst, startTime, endTime):
    for i in range(len(coor_lst)): #find the middle coordinate
        if coor_lst[i][1]>coor_lst[i-1][1]:
            top_coor=coor_lst[i]
            frame_num=i
    half_coor=top_coor[0]
    first_coor=coor_lst[0][0]
    proj_dist=(half_coor-first_coor)*2 #find the distance from start to middle and double to find
projected distance.
    return proj_dist #return

def accu(coor_lst,target): #use last instance of the ball to see how far it is from the target
    wall_hit=coor_lst[len(coor_lst)-1] #set target coordinate
    xdif=abs(wall_hit[0]-target[0]) #find x and y differences from balls ending position
    ydif=abs(wall_hit[1]-target[1]
    tot_dif=math.sqrt(xdif**2+ydif**2) #find distance apart
    return tot_dif #return

def write_to(accu):#save accuracy in text file and average previous cases
    if not os.path.exists('accu.txt'):
        file(filename, 'w').close()
    fa=open("accu.txt","a")
    fa.write(str(accu))
    fa.close()
    fr=open("accu.txt","r")
    lst=fr.readlines()
    res=0
```

```
    for i in range(len(lst)):
        res+=float(lst[i])
    res/=len(lst)
    return res
```

Arduino:

```
void global(){
int piezo_Pin= 0; #set to input pin

//Set the threshold levels
int threshold= 500; #set threshold vibration
}
//Wakeup the Serial Monitor
void setup()
{
pinMode(piezo_Pin, INPUT) #set the pin to input
Serial.begin(9600); #set communication to computer(should be changed to bluetooth
}

#if the reading is higher than the threshold value, then the LED is turned ON for a Second You
can edit to your specification
void loop()
{
int reading= analogRead(piezo_Pin);
Serial.println(reading);
if (reading > threshold)
{
  return true
}
}
```

3.3 Software Troubleshooting

The process of troubleshooting the code has been somewhat difficult both because of unfortunate circumstances and because the coders didn't have a background in the software developed. The circumstantial misfortunes mainly came in the fact that the circuitry components were not delivered until super late into the project. This meant that the code wouldn't even have a chance to be tested until way later into the project unless it was completely altered to be tested one function at a time. Another circumstantial mishap is the fact that the way we approach our project has changed completely at least 5 times during this semester. While this is part of the prototyping process the changing of the design lasted well into the second half of the semester.

This meant that multiple added features had to be added to the code as well as hard labored code being taken away from the project. Lastly the other difficulty was in our ambitious design and lack of experience in our design. While this could be supplemented by going in to meet with the open lab teacher assistance's to get help this only helps so much because most of the work would still be on us for the research.

4. MILESTONE AND FINAL PRODUCT REQUIREMENTS

Commented [EG1]: (1) Discuss the human resources and training

4.1 Benchmark A Requirements

Benchmark A required the initial drawings for the product design, the code flowchart, a project schedule of 20 to 30 tasks, and a materials list that included a cost estimate and links to order from. The initial drawings had to include the tangible components of the prototype, meaning a detailed visual representation of the setup of the product when in use. The code flowchart had to show the preliminary thought process behind the code that would be written. The project schedule showed all the tasks that needed to be completed and the dates they were meant to be completed by.

4.2 Benchmark B Requirements

The necessary components for benchmark B were working code for the image processing, CAD designs for the physical prototype, and device readings. The code was supposed to be able to represent some form of data or feedback. The CAD designs and the related 3D-printed components were completed. The device readings were supposed to have a

separate part of the product to display data to the user, however this was pivoted since our design changed.

4.3 Final Submission Requirements

The final product is meant to have the complete final prototype, sensors that can collect and recording data, and a simulation projection with the correct images.

5. RESULTS

5.1 Benchmark A

Benchmark A yielded the initial code flowchart, which showed the general processes that the camera and button did together in order to output data. In addition to a code flowchart, initial design sketches were also made. The materials listed in the cost estimate resulted in a estimated total of $5,000. The original project schedule contained 22 tasks, but eventually more were added as the design evolved. Both the original and final cost estimates and project schedules are shown below. The originals were what was submitted in benchmark A, then were edited later on, but not officially re-submitted once changed.

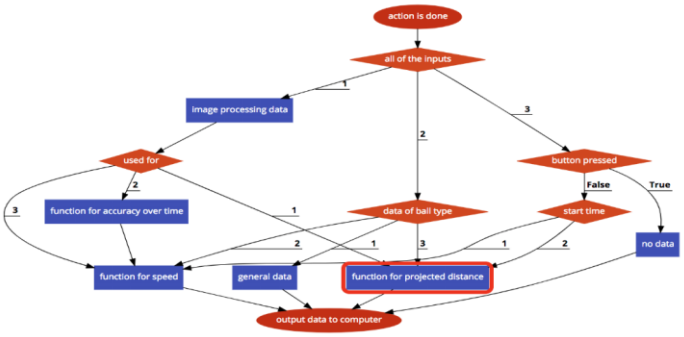

Figure 1. Initial Code Flowchart

Table 1. Initial Cost Estimate

Accushot Cost Estimate				
Bought Online				
Item	Cost	Quantity/Hours	URL	Total
Tarp	$10	1	tarp link	$10
Camera (image processing)	$14	1	Camera Link	$14
Force Sensitive Resistor	$17	1	Sensor Link	$16
Projector	$50	1	Projector Link	$50
Miscellaneous electronics	$10	1	None	$10
			Sub total	$100
Supplied by NYU				
Raspberry Pi	$0	1	None	None
LCD	$0	1	None	None
Miscellaneous electronics	$0	1	None	None
3D printed item	$0	1	None	None
Labour				
Total Labour	$50	81.25	None	4062.5
4162.5	4162.5	4162.5	4162.5	4162.5

Table 4. Original Project Schedule

Milestone 1	8 wks	Fri 1/4/19	Thu 2/28/19	
Brainstorm Project Ideas	20 hrs	Wed 2/20/19	Fri 2/22/19	
Collect Materials	**7.06 days?**	**Wed 2/20/19**	**Fri 3/1/19**	
Pick Up Kit from Model Sh	0.5 hrs	Wed 2/20/19	Wed 2/20/19	
LCD from Model Shop	0.25 hrs	Wed 2/27/19	Wed 2/27/19	
Order Materials from Amaz	0.5 hrs?	Fri 3/1/19	Fri 3/1/19	
Research and Development	3 days	Mon 2/25/19	Wed 2/27/19	
Milestone 2	2.4 wks	Thu 2/28/19	Thu 3/14/19	
Benchmark A	1 day?	Wed 3/6/19	Wed 3/6/19	
Program	**1.94 days?**	**Fri 3/1/19**	**Fri 3/15/19**	4
Store and Analyze Data	1 day?	Fri 3/1/19	Sat 3/2/19	
Calculations	**4 days?**	**Tue 3/12/19**	**Fri 3/15/19**	
Get derivatives of grap	4 days?	Tue 3/12/19	Fri 3/15/19	
Simulation	8 hrs	Wed 3/13/19	Wed 3/13/19	
Color Image Prcoessing	12 hrs	Wed 3/13/19	Thu 3/14/19	
Revise Project Outline	4 hrs	Sat 3/2/19	Sat 3/2/19	
Finances	8 hrs	Tue 3/5/19	Tue 3/5/19	
Wire Microprocessors	2 hrs?	Fri 3/8/19	Fri 3/8/19	
3D-Print	**1 day?**	**Wed 2/27/19**	**Wed 2/27/19**	
Ball Stand	1 day?	Wed 2/27/19	Wed 2/27/19	
Protective Box for Projecto	1 day?	Wed 2/27/19	Wed 2/27/19	
Image Processing	**1 day**	**Thu 2/28/19**	**Thu 2/28/19**	21
Automated Image Analysis	8 hrs	Thu 2/28/19	Thu 2/28/19	
Build Prototype	12 hrs	Mon 4/1/19	Tue 4/2/19	9
Milestone 3	14 days?	Mon 3/25/19	Thu 4/11/19	
Trial and Error	3 wks	Mon 4/15/19	Fri 5/3/19	12,25,22

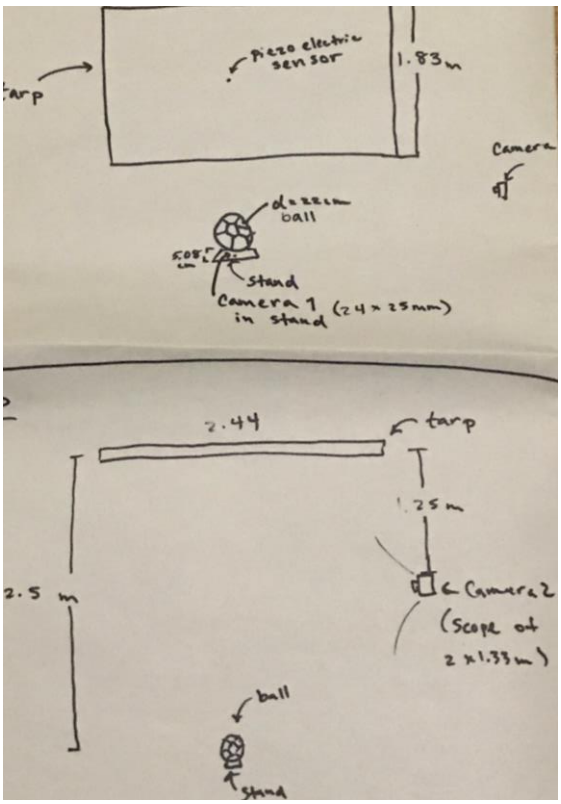

Figure 2. Initial Set Up Sketch

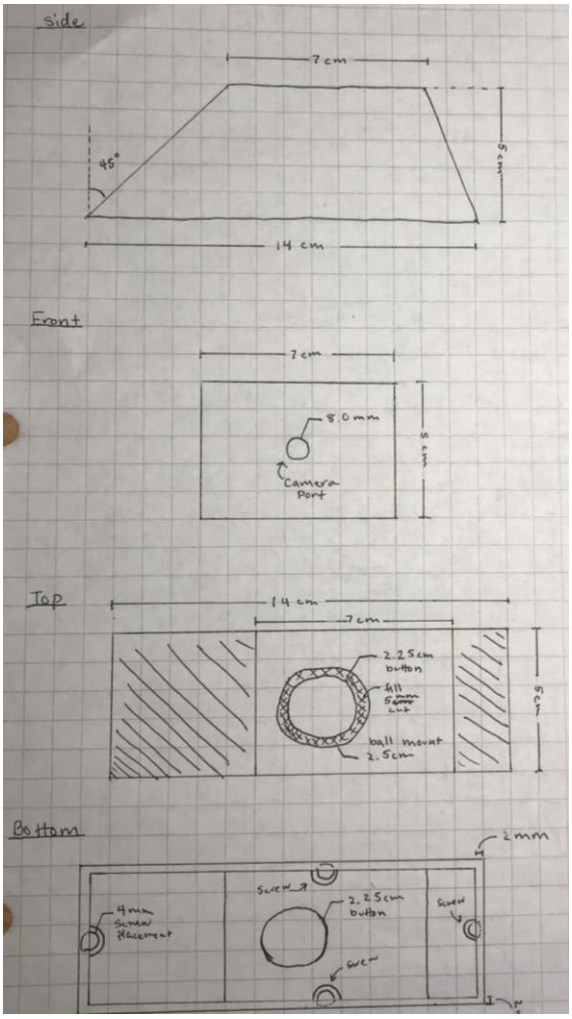

Figure 3. Ball Stand Initial Sketch

5.2 Benchmark B

The code flowchart changed drastically from Benchmark A to B, taking account for a generated video from the cameras, an accuracy text file, and all the data projected to the user after it has been processed by the program. The flowchart also goes into detail of the process which the raspberry pi is taking to find a list of coordinates of the ball. In addition to the code flowchart being completed, the actual Python code was also completed. A camera case and ball stand CAD drawing was also designed, and 3D printed for Benchmark B. For Benchmark B, the cost estimate rose $100 in account of the extra materials bought, resulting in a new total cost of $5,100.

Figure 4. Camera Front

Figure 5. Camera Case Top

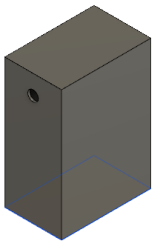

Figure 6. Camera Case Isometric

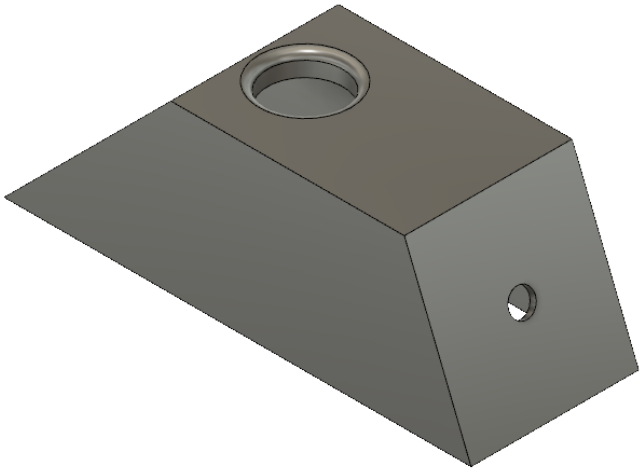

Figure 7. Ball Stand Isometric

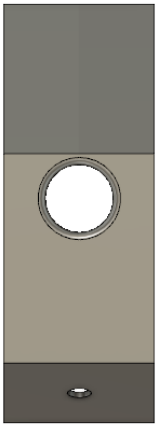

Figure 8. Ball Stand Top

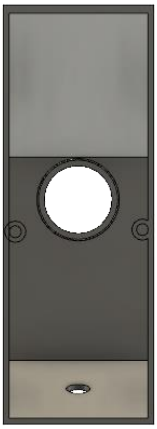

Figure 9. Ball Stand Bottom

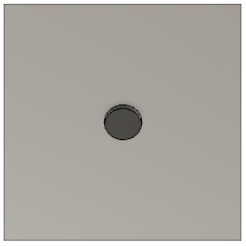

Figure 10. Ball Stand front

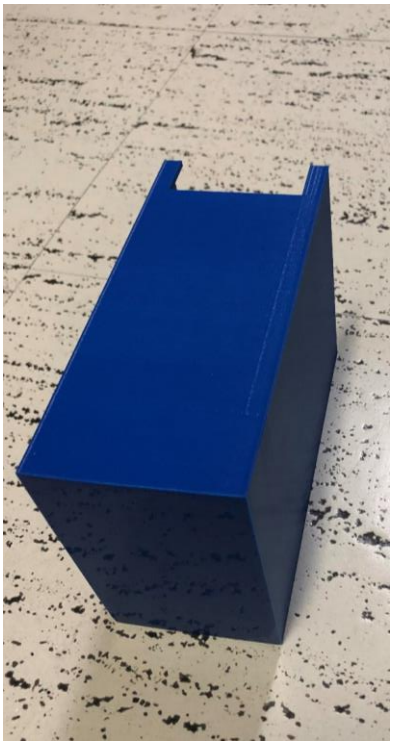

Figure 11. Camera Case Back

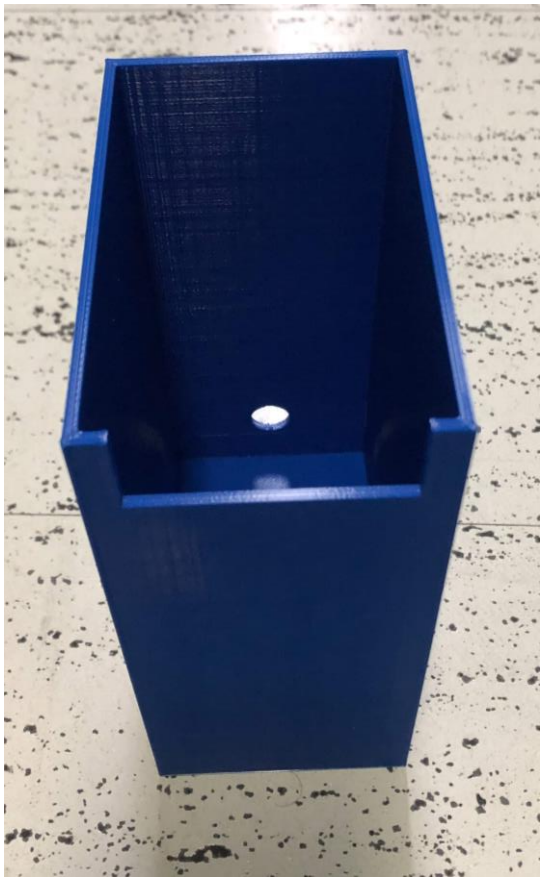

Figure 12. Camera Case Bottom

Figure 13. Camera Case Bottom-Isometric

Figure 14. Ball Stand Side

Figure 15. Ball Stand Isometric

Figure 16. Ball Stand Top

Figure 17. Ball Stand Bottom

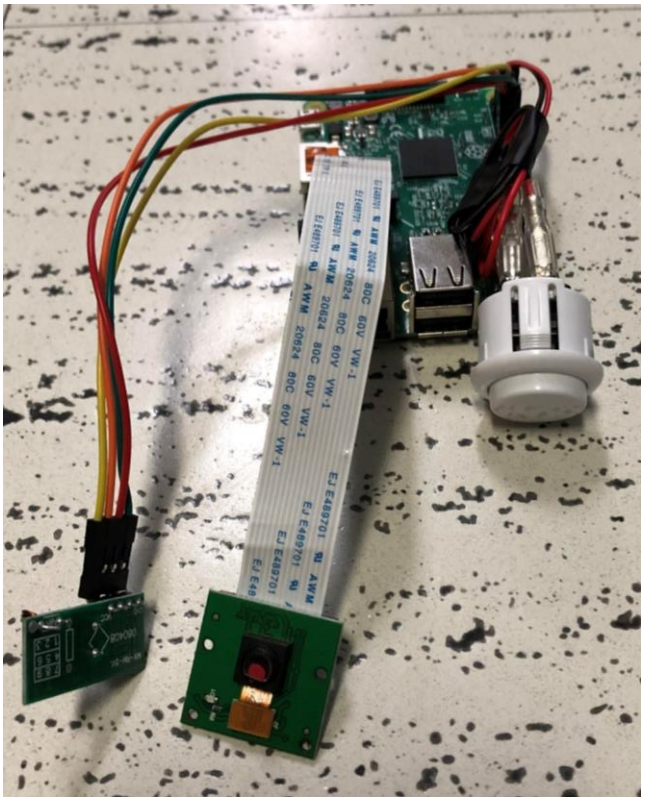

Figure 18. Ball Stand Raspberry Pi Circuit

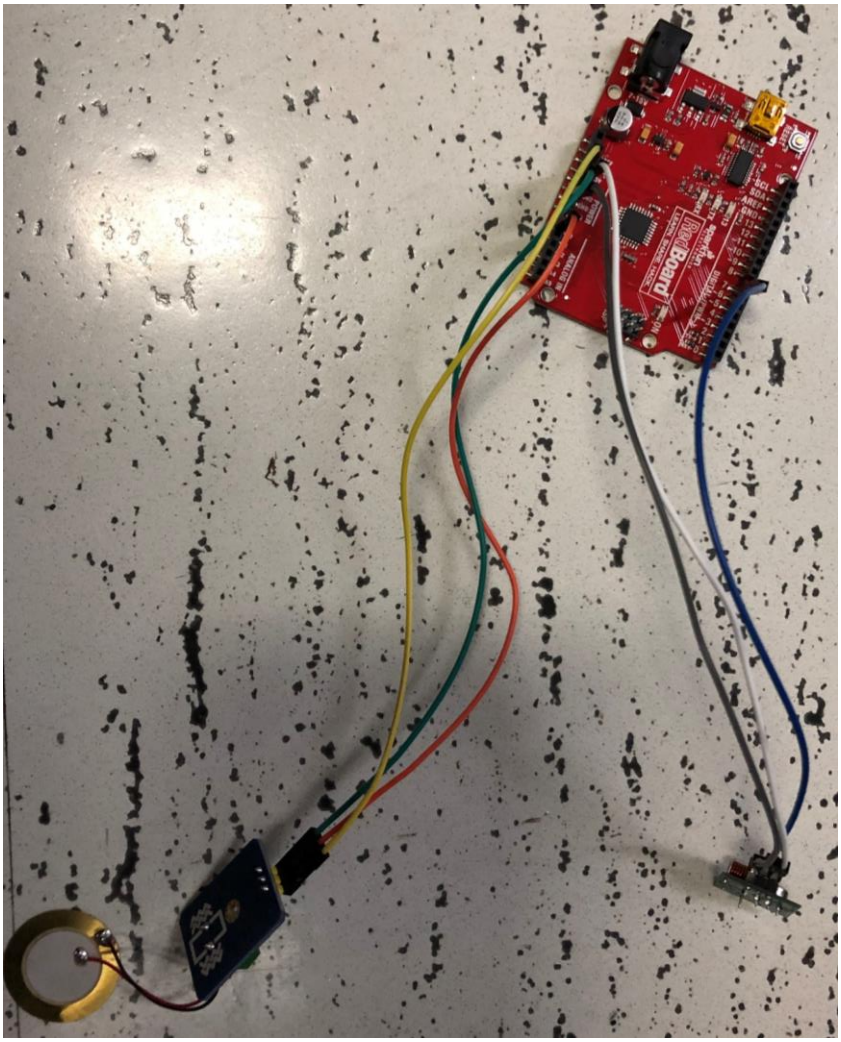

Figure 19. Arduino With Force Sensor

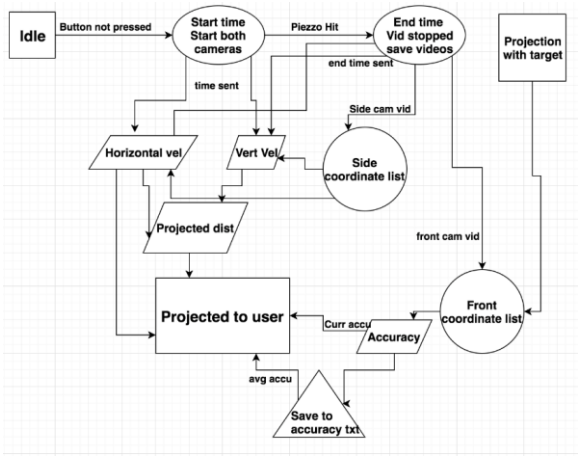

Figure 20. Final Code Flowchart

Table 3. Final Project Schedule

Milestone 1	1.8 wks	Mon 2/18/19	Thu 2/28/19
Brainstorm Project Ideas	20 hrs	Wed 2/20/19	Fri 2/22/19
Collect Materials	**7.06 days?**	**Wed 2/20/19**	**Fri 3/1/19**
Pick Up Kit from Model Shop	0.5 hrs	Wed 2/20/19	Wed 2/20/19
LCD from Model Shop	0.25 hrs	Wed 2/27/19	Wed 2/27/19
Order Materials from Amazon	0.5 hrs?	Fri 3/1/19	Fri 3/1/19
Research and Development	3 days	Mon 2/25/19	Wed 2/27/19
Benchmark A	**6.38 days**	**Wed 2/27/19**	**Wed 3/6/19**
Finances	1 day	Wed 2/27/19	Wed 2/27/19
Revise Project Outline	4 hrs	Sat 3/2/19	Sat 3/2/19
Code Flowchart	3 hrs	Wed 3/6/19	Wed 3/6/19
Initial Drawings	3 hrs	Wed 3/6/19	Wed 3/6/19
Milestone 2	**12 days**	**Thu 2/28/19**	**Thu 3/14/19**
Milestone 3	**14 days?**	**Mon 3/25/19**	**Thu 4/11/19**
Benchmark B	**8 days?**	**Mon 3/25/19**	**Tue 4/2/19**
Working Prototype	8 days?	Mon 3/25/19	Tue 4/2/19
CAD Drawings	8 days?	Mon 3/25/19	Tue 4/2/19
Program	**26 days?**	**Mon 3/25/19**	**Fri 4/26/19**
Color Image Processing	20 days	Mon 3/25/19	Fri 4/19/19
Store and Analyze Data	15 days?	Mon 4/8/19	Fri 4/26/19
Simulation	8 hrs	Mon 4/15/19	Mon 4/15/19
Wire Microprocessor Circuits	**11 days?**	**Sat 3/30/19**	**Fri 4/12/19**
Arduino	10 days	Sat 3/30/19	Fri 4/12/19
Side Camera	11 days?	Sat 3/30/19	Fri 4/12/19
Ball Stand	11 days?	Sat 3/30/19	Fri 4/12/19
Determine Necessary Calculations	**10 days?**	**Mon 4/1/19**	**Fri 4/12/19**
Get derivatives of graphed data	10 days?	Mon 4/1/19	Fri 4/12/19
3D-Print	**5 days?**	**Mon 4/1/19**	**Fri 4/5/19**
Ball Stand	5 days?	Mon 4/1/19	Fri 4/5/19
Protective Box for Projector	1 day?	Mon 4/1/19	Mon 4/1/19
Image Processing	**21 days?**	**Wed 4/3/19**	**Wed 5/1/19**
Automated Image Analysis	128 hrs	Wed 4/3/19	Wed 4/24/19
Build Physical Prototype	**1 day?**	**Wed 5/1/19**	**Wed 5/1/19**
Set Up Tarp	1 day?	Wed 5/1/19	Wed 5/1/19
Put Ball on Stand	1 day?	Wed 5/1/19	Wed 5/1/19
Trial and Error	2.4 wks	Mon 4/29/19	Tue 5/14/19
Comission	1 day?	Wed 5/1/19	Wed 5/1/19

Table 4. Final Cost Estimate

Accushot Cost Estimate					
Bought Online					
Item	Cost	Quantity/Hours	URL	Total	Reason
Tarp	$10	1	tarp link	$10	For the ball to hit, attched to force sensor
Camera (image processing)	$14	2	Camera Link	$28	Used for image processing
Soccer Ball	$0	1		$0	Used for testing the product
Projector	$50	1	Projector Link	$50	Project the background on the tarp
Pieszoelectric Sensor	$8	1	Sensor link	$8	Stop time when senses vibration
Miscellaneous				$4	
			Sub total	$100	
Supplied by NYU					
Raspberry Pi	$0	2	None	None	Microcontroller for the programming
Arduino	$0	1	None	None	For the button and vibration sensor
Button	$0	1	None	None	Start time when released
3D Printed Camera Box	$0	1	None	None	Holds side camera for image processing
3D printed Ball Stand	$0	1	None	None	Stand for the ball, incorporated button to start time
Transmitter and Receiver	$0	1	None	None	Send information from side camera to main raspberry pi
Wires	$0		None	None	Connect circuits
Labour					
Total Labour	$50	100		$ 5,000.00	
			Total	$ 5,000.00	

5.3 Difficulties Experienced

Difficulties were initially experienced deciding the orientation and location of the sensor and cameras, as well as deciding how the data received should be interpreted and manipulated to return useful information to the user. Difficulties were also experienced with micro SD card an initializing the Raspberry Pi operating system, as receiving functioning one took a considerable amount of time thus delaying our submission for Benchmark B.

6. CONCLUSION

The results of the project were the image processing, piezo sensor, and push button were all integrated and tested together and separately to test function and usefulness. They were all wired and used for location and accuracy purposes and resulted in manipulated data for the end

user for improvement in their craft. Future improvements for the idea include a bigger budget, better/functional technology that is already preconfigured with the tools necessary, and more resources and libraries to grow this project even further.

Works Cited

Brain, Marshall. 2000. *How Stuff Works*. 1 April. Accessed April 20, 2019. https://electronics.howstuffworks.com/microcontroller1.htm.

Mary, Rose. 2011. *Engineers Garage.* February. Accessed April 20, 2019. https://www.engineersgarage.com/articles/image-processing-tutorial-applications.

NYU SOE. 2019. *EG1003 Lab Manual.* Accessed April 20, 2019. https://manual.eg.poly.edu/index.php/Main_Page.

Python for Beginners. 2019. *https://www.pythonforbeginners.com/learn-python/what-is-python/.* January. Accessed April 20, 2019.

Project: Accushot

Group: Alexa Arora, Elijah AgerLuckett, Andrew Hampton, Christina Curry

Documentation	30	**30**
Introduction	10	**10**
Requirements	10	**10**
Procedures	15	**15**
Milestone/Final Product Requirements	9	**10**
Results	10	**10**
Conclusion	15	**15**
Subtotal	99	**100**
Formatting Deductions		
TOTAL	99	