12-2-2014

Dynamically Reconfigurable Miniature Golf Course

James Lind
Western Michigan University, jamesplind@gmail.com

Dynamically Reconfigurable Miniature Golf Course

Mohammed Abukabbos
Ryan Feist
James Lind

Advisor: Dr. Steven M. Durbin

ECE 4820 Electrical/Computer Engineering Design II
December 5th, 2014

Abstract

Dynamically Reconfigurable Miniature Golf Course
Mohammed Abukabbos, Ryan Feist, James Lind
Western Michigan University - College of Engineering

The purpose of this report is to inform the reader of the Senior Design Project lead by 3 students in the Electrical and Computer Engineering department at Western Michigan University. The project is in needed for use within the ECE department as a tool for showcasing to prospective students the attributes of electrical and computer engineering.

At the heart of the project lies an Arduino microcontroller acting as the commanding brain of the system. Its task is to manipulate various types of obstacles throughout the playing field with a goal of delivering a dynamic experience to the player(s). Additionally, a Raspberry Pi microcontroller is utilized to display a friendly and easy-to-use graphical user interface for keeping track of player rankings and game commands. The overall project will bear fruit from multiple disciplines within the ECE department namely schematic design methodology, power management and programming.

Future advancements in the project would include the integration of the automatic stroke-counting putter developed in a previous Senior Design Project.

DISCLAIMER

Acknowledgments/Permission

Special thanks to Dr. Steven Durbin and Western's Electrical and Computer Engineering department for providing time, money, and resources for the construction of the project.

Special thanks to David Florida for providing components such as the TV monitor with stand, power supplies, and assistance in ordering parts through online retailers.

Special thanks to Erivelton Gualter Dos Santos for providing example Python code for GUI development through the Raspberry Pi microcontroller.

Table of Contents

Background & Need Statement

There is currently a lack of projects that can be used to showcase to prospective students what they will learn from the electrical and computer engineering program in the Western Michigan University College of Engineering and Applied Sciences. More engaging, fun examples to be used during open houses are needed. A number of projects to do that have been commissioned by Dr. Steven Durbin, Professor and Chair of the Electrical and Computer Engineering (ECE) Department. In this project, a dynamically reconfigurable miniature golf course was created.

Due to a lack of space, a full-sized miniature golf course cannot fit within a small demonstration area for prospective students and visitors. Therefore, a single physical hole was used in conjunction with a microcontroller to reconfigure obstacles around the hole. This simulates a 9-hole miniature golf course, while taking up only a fraction of the space that a full-sized course would. The project will be used to provide a fun, interactive demonstration for future College of Engineering and Applied Sciences students and open house visitors, showing what the Western Michigan University Electrical Engineering program has to offer.

Sponsor ID:

The sponsor for this project is *Dr. Steven M. Durbin*, Professor and Chair of the Western Michigan University CEAS Electrical and Computer Engineering Department.

Project Construction

A description of each part of the project's construction is outlined below.

Frame

Base:

The base of the frame was constructed by arranging 2x4's in a ladder-like structure. Two pieces cut to 88" were laid lengthwise and seven smaller pieces were cut to 30" to serve as supports for mounting the obstacles. The top of the frame was constructed from a piece of plywood with holes cut and drilled in it to mount the obstacles. This top portion was then covered with a layer of turf-like material and 2x2 pieces of wood were used as rails to prevent the ball from leaving the playing area. A detailed drawing of the frame is shown in Fig. 1.

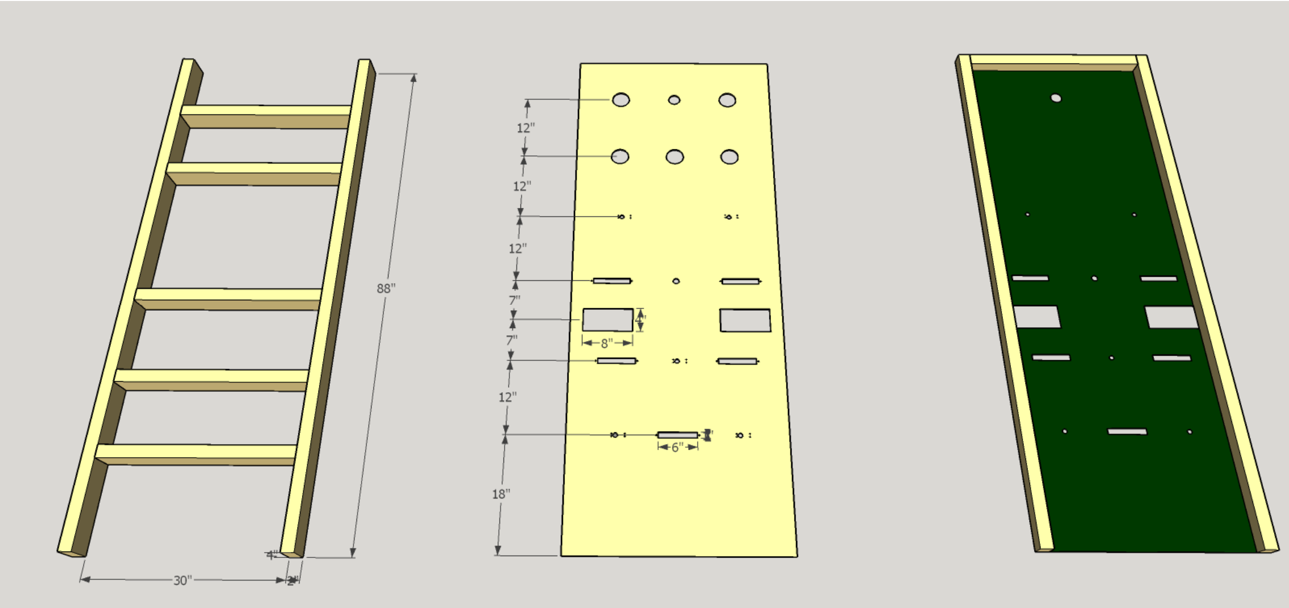


Figure 1: Parts of frame construction. Base (left), top (center), and felt/rails (right).

<u>Hole:</u>

The hole was modeled after a standard golfing hole as seen on many golf courses. Modification was necessary to ensure the system would be made aware of a ball's presence in the hole. This was done by placing a modified arcade button at the bottom of the hole, triggered when a force of 20 grams or more was applied to its surface. In this fashion, every time the golf ball (46 grams) fell into the hole it would push down the button, acting as a switch and sending the Arduino an active-low signal, indicating that the user had scored.

Obstacles

<u>Rotating Walls:</u>

Powering each rotating wall is a servo motor mounted directly to the underside of the plywood playing field. The mounting of the 5 servo motors and their relative positions throughout the underside of the playing field was carefully determined as to ensure a symmetrical and uniform design layout seen from above by the players. The rotating obstacle itself was then constructed out of wood to a length of 6" then fastened with a servo accessory for direct attachment to the servo spline for angular manipulation.

<u>Pop-up Walls:</u>

The basic concept of the pop-up wall was to raise and lower an object, revealing itself to the player in one position while hiding from the player in the other position. This process was built from multiple components including a 6"x3" piece of plexiglass used as the visible up-and-down obstacle, ½" diameter plastic cylinder sliced in half to form two channels for the plexiglass to fluidly slide along, and two 90 degree brackets for added stability and reliability.

The up and down motion was provided by a servo motor mounted on its side with a slider-crank mechanism to transition the rotational motion to linear vertical motion.

Adjustable Ramps:

The adjustable ramps were constructed from a lightweight foam material cut to resemble a half-cylinder whose flat side could then be adjusted to provide a slight angle noticeable on the playing field. ¼” thin wood dowels matching the length of the half-cylinder ramps were inserted into each side allowing for a backbone to the ramp. Servo accessories were then attached to the backbones of our obstacles and then attached to an accompanying servo motor for powering their movement.

Terrain Modifiers:

The goal of the terrain modifiers was to establish a slight hill in the playing field, replicating uneven surfaces most likely seen on actual golfing greens. This was accomplished by mounting a servo motor on its side to allow the conversion from rotational motion into linear motion needed for pushing up on the green fabric from below. An accessory piece was used to fasten a bolt to the servo motor which was then fed through a hole in the plywood, quietly lying flat until instructed to rise, forming a miniature hill on the playing field.

Spinning Wall:

This spinning wall is powered by a DC motor with continuous rotation, unlike the servo-powered rotating walls, which only operate within a specific range of motion. Given that this DC motor is not a stepper motor there is no direct way of keeping track of the position of the shaft. To accomplish this task, a continuous rotation potentiometer was coupled with the DC motor via same sized gears. The Arduino can then take the resulting potentiometer values and easily track the exact position of the DC motor shaft.

Electronics

Block Diagram:

A block diagram outlining the interactions between the different parts of the system is shown in Fig. 2. The Arduino is used to check whether or not the ball is in the hole and also handles the movement of the obstacles. The Raspberry Pi deals with user input as well as displaying information to the screen. The Arduino and Raspberry Pi are connected with a USB cable, which allows for communication between the two. This cable is also used to provide the Arduino with power from the Raspberry Pi. Each element is discussed in detail below.

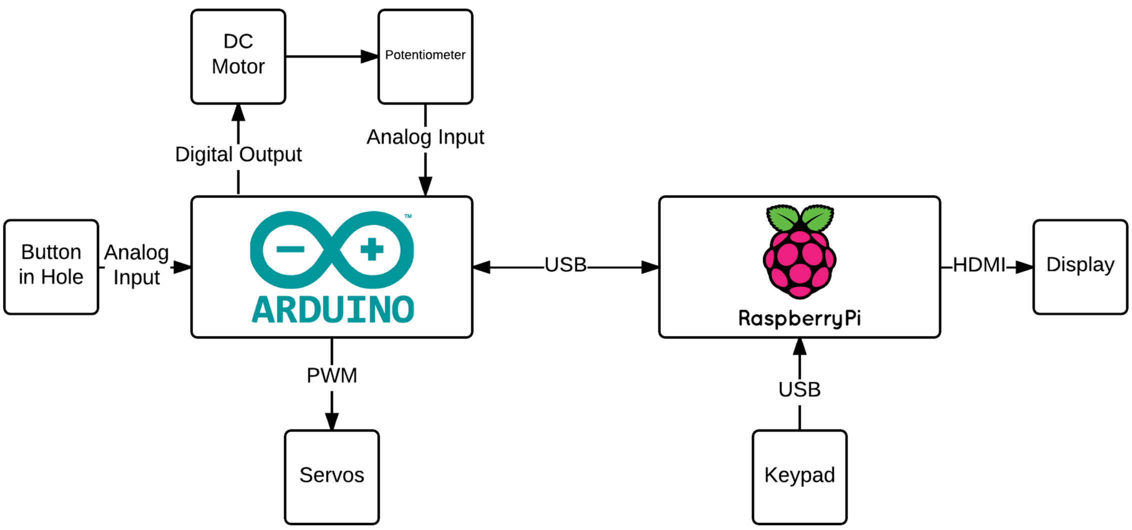


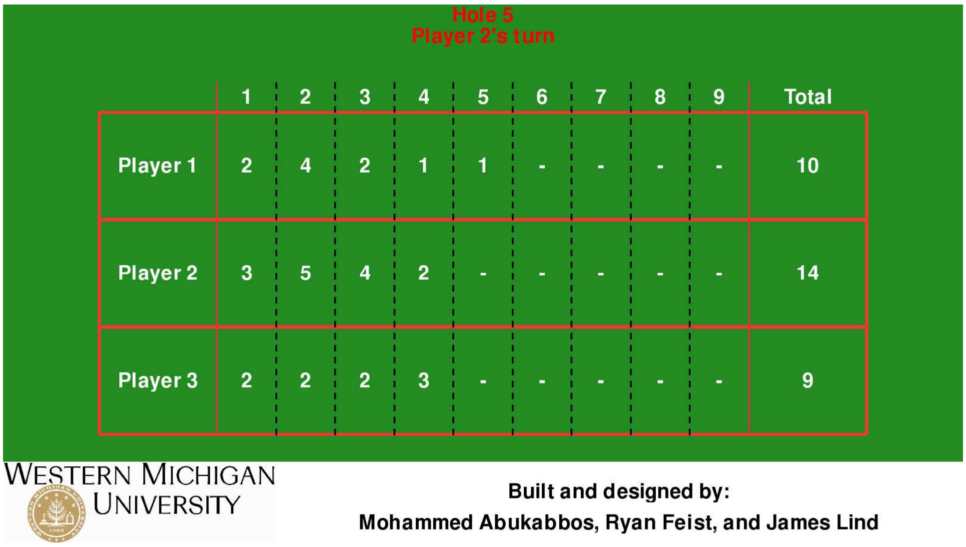
Figure 2: Block diagram

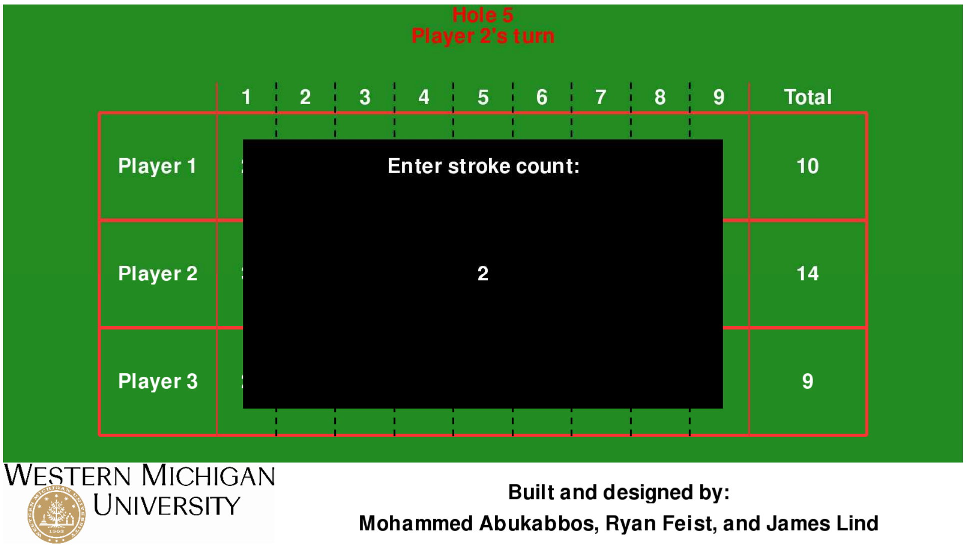
Arduino:

An Arduino Mega microcontroller was used to control the servos, DC motor, and detect when the ball was in the hole. Each of the servo's signal lines were connected to one of the Arduino's digital output pins. Then, the Arduino servo library was used to set the servos to the proper positions. By varying the pulses sent to the servos from 700µs to 2300µs, the servos can be set from anywhere between 0° and 180°. One of the Arduino's digital output pins was used to turn on the transistor in the DC motor's circuit. By setting the output pin to high (5V) or low (0V), the motor is turned on or off, respectively. The motor's position was read by using two gears to connect the shaft of the DC Motor to the stem of a continuous-rotation potentiometer. One of the Arduino's analog inputs was then used to read the value from the potentiometer, giving an indication as to the current angle of the spinning wall. This was used to set the spinning wall back to its default position of 0° on the holes when it wasn't spinning.

Raspberry Pi and Display:

A full-sized display was used to show the current hole number, score, and course status to the player. This information was displayed on the screen using a Raspberry Pi with the Raspbian operating system installed on it. A python script was run from the Pi using the Pygame python module to draw the necessary graphics. A USB numeric keypad was connected to one of the Pi's USB ports to allow the players to enter their scores. USB serial communication was used to transfer information between the Arduino and Raspberry Pi when necessary. This was made possible through a serial communication module for python, pySerial. Screenshots of the graphical user interface running on the Raspberry Pi are shown in Fig. 3.





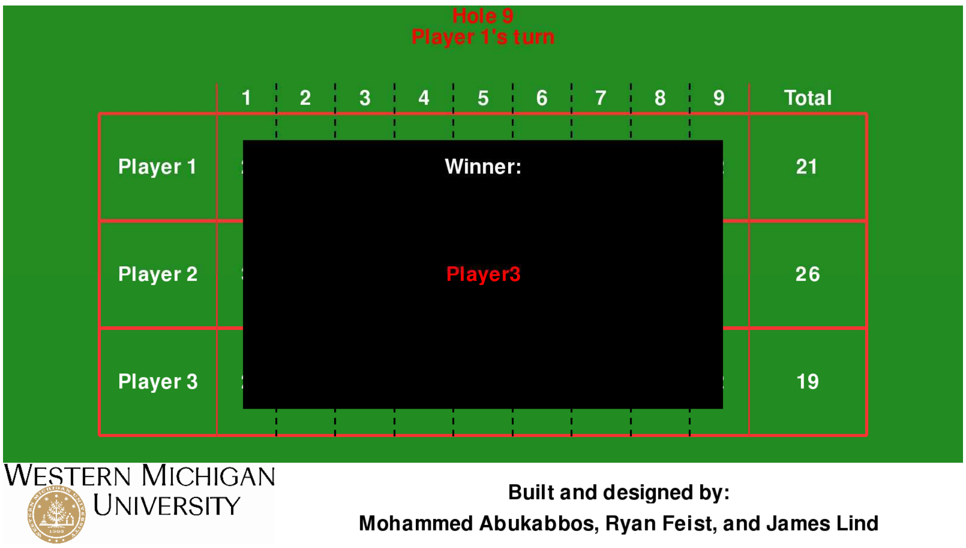


Figure 3: Graphical user interface showing main scorecard screen (Top), stroke entry dialog box (Middle), and winner declaration (Bottom)

<u>DC Motor Circuit:</u>

A simple transistor circuit, shown in Fig. 5 was used to turn the DC motor on and off. A model MJE3055T transistor was used. According to the MJE3055T's data sheet, it has a DC current gain of $\beta = 70$ and a base-emitter voltage of $V_{BE} = 1.8V$. The DC motor draws around 400mA when operating at 12V. For the collector current to provide 400mA, a base current of $\frac{400mA}{70} = 5.7mA$ is needed. The Arduino's digital output pin connected to the base of the transistor provides 5V. Therefore, a resistor of $\frac{5V-1.8V}{5.7mA} = 560\Omega$ placed between the Arduino's output pin and the base of the transistor would provide the necessary current. Two 270Ω resistors were placed in series, resulting in 540Ω.

<u>Process control:</u>

When the system is initially started up, the Arduino defaults to the configuration for the first hole and the Raspberry Pi asks for one of the users to enter the number of players. Once the input is received, the Pi sends the information to the Arduino and draws a scoreboard with the appropriate number of players on the display. Then, the Arduino and Pi enter into the control loop described in Fig. 4. The blue sections represent steps associated with the Arduino, while the red sections represent steps associated with the Raspberry Pi. The Arduino constantly checks the switch in the hole for the ball. Once the ball is detected, it sends a signal to the Raspberry Pi enabling score entry. The Pi pops up the dialog box shown in Fig. 3 and waits until it receives an entry from the player. Once the input is received, the Pi sends a signal to the Arduino letting it know that the user has entered his or her stroke count. This lets the Arduino know that the user's turn is finished. If the player hasn't removed his or her ball from the hole yet, the Arduino waits until the ball is removed and then continues. If the player who just scored wasn't the last player, the Pi updates the display for the next player. If the player who just scored was the last player, the Pi updates the display to read "Configuring Next Hole" and the Arduino moves the obstacles to the positions for that hole. When the Arduino is finished, the Pi goes back to the normal display of the player scores, letting the players know that the hole is ready. After nine holes, the player with the lowest score is declared the winner, as shown in Fig. 3.

Circuit design:

A schematic for the electronics is shown is Fig. 5.

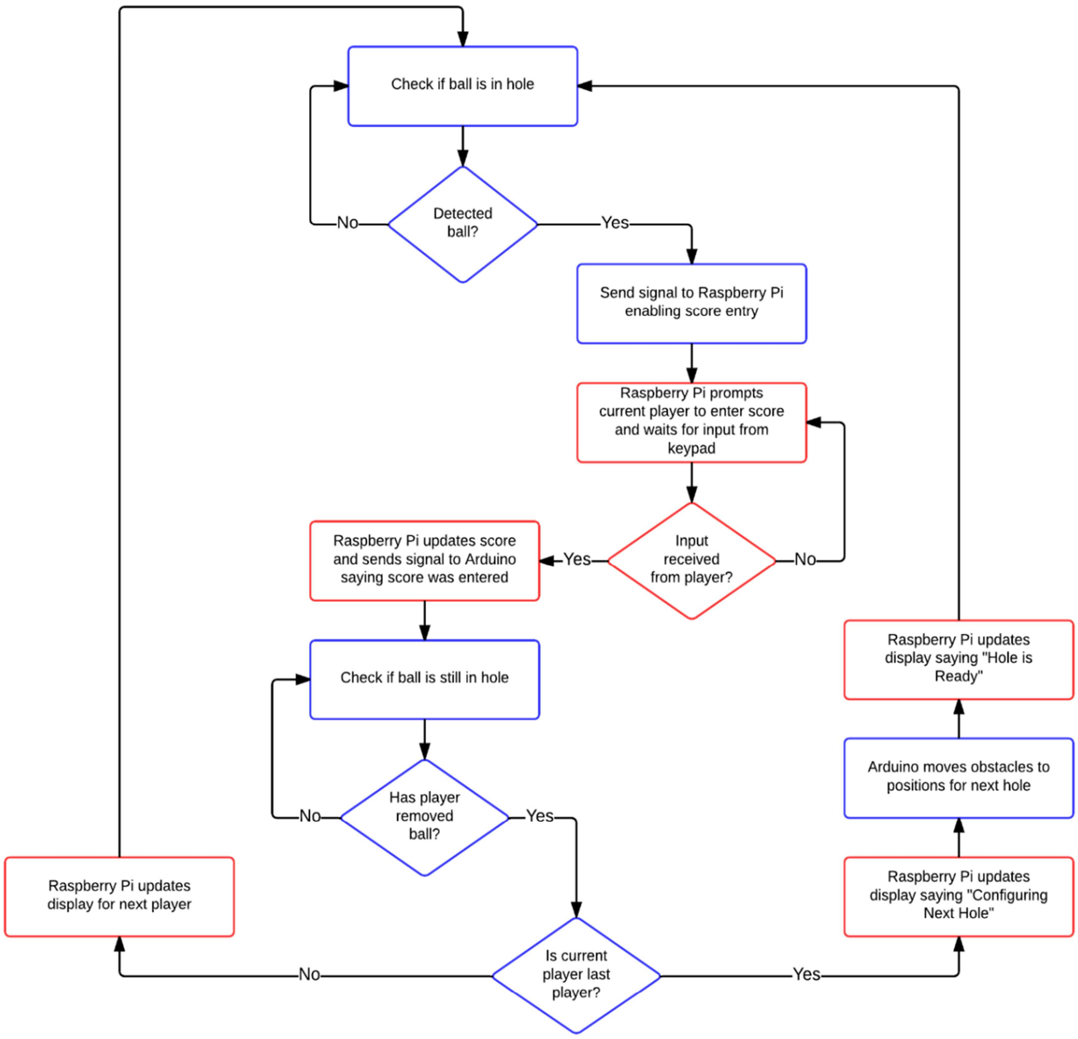


Figure 4: Process control loop for Arduino and Raspberry Pi

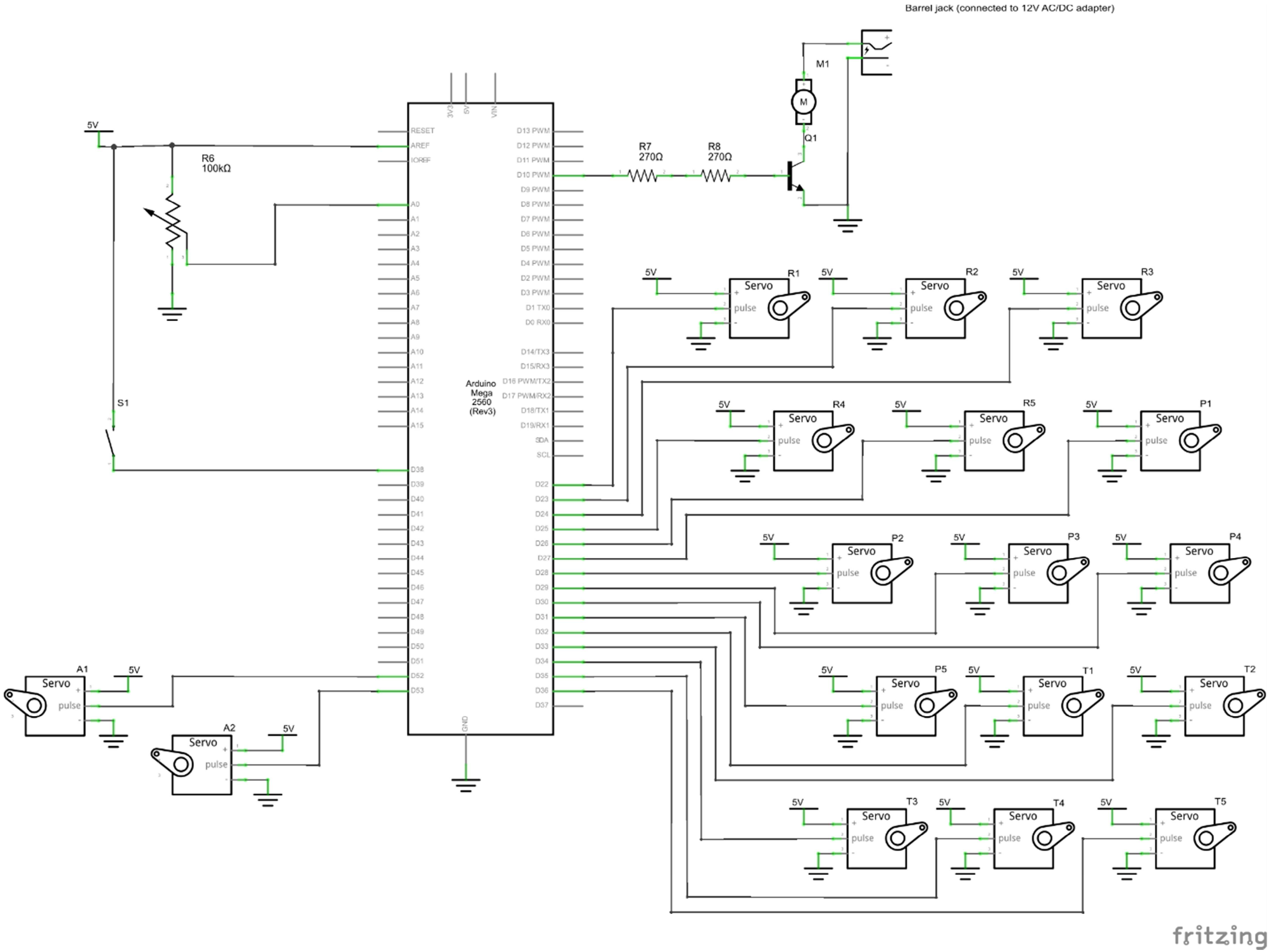


Figure 5: Circuit Schematic

Bill of Materials

A detailed list of the costs involved in the project is given below in Table 1.

Table 1: Project Components and Costs

Final Project Budget

Item	Quantity	Total Price
General Items		
2"x4" wood	4	$19.88
2"x2" wood	2	$6.00
Screws 2" long	50 count	10.54
Screws 1.5" long	50 count	$8.00
Plywood	1	30
Turf	1	$16.00
Carpet Adhesive	1	$5.00
Adjustable Feet	1	$2.50
Grip Pads	1	$2.50
Dremel Cutting Discs	5	7.98
Handles	2	$3.00
Wheels	2	$7.00
Power Strip	1	$8.00
Screws w/ locking nut	15	$10.00
Continuous Spinning Wall		
DC Motor	1	$29.37
Set screw shaft coupler	1	12
Spur Gear	2	$17.50
Potentiometer	1	$13.89
Golf Ball Hole		
Arcade Button	1	$8
Golf Ball Hole	1	$6.99

Item	Quantity	Total Price
Adjustable Ramps		
Floral foam cylinder	4	$12.00
Wood 1/4" dowel rode	1	$2.00
Terrain Modifiers		
Bolts w/ pre-drilled holes	5	$5.45
Pop-Up Walls		
Servo Arm (Aluminum)	5	$25.00
Servo Arm (Plastic)	pack of 10	$12.50
1/2" black plastic couplers	5	$3.15
Metal Angle Brackets	5 packs of 2	$15.00
Electronics		
Wires	6 spools	$20.00
Wire U-nails	100 count	$2.50
Arduino Mega 2560	1	$38.00
Raspberry Pi	1	$37.00
TV monitor w/ stand	1	DONATED
Servos	17	$169.15
5V 1A AC/DC power adapter	3	DONATED
12V 1A AC/DC power adapter	1	$6.00
Pin headers	2	$3.00
Solder-able breadboard	1	$2.95
DC barrel jack adapter	4	$7
USB numeric keypad	1	$4.19
Total Budget		**$589.44**

Final Product

Pictures of the final product are shown on the next two pages in Fig. 6 and Fig. 7.

Figure 6: Display and Course

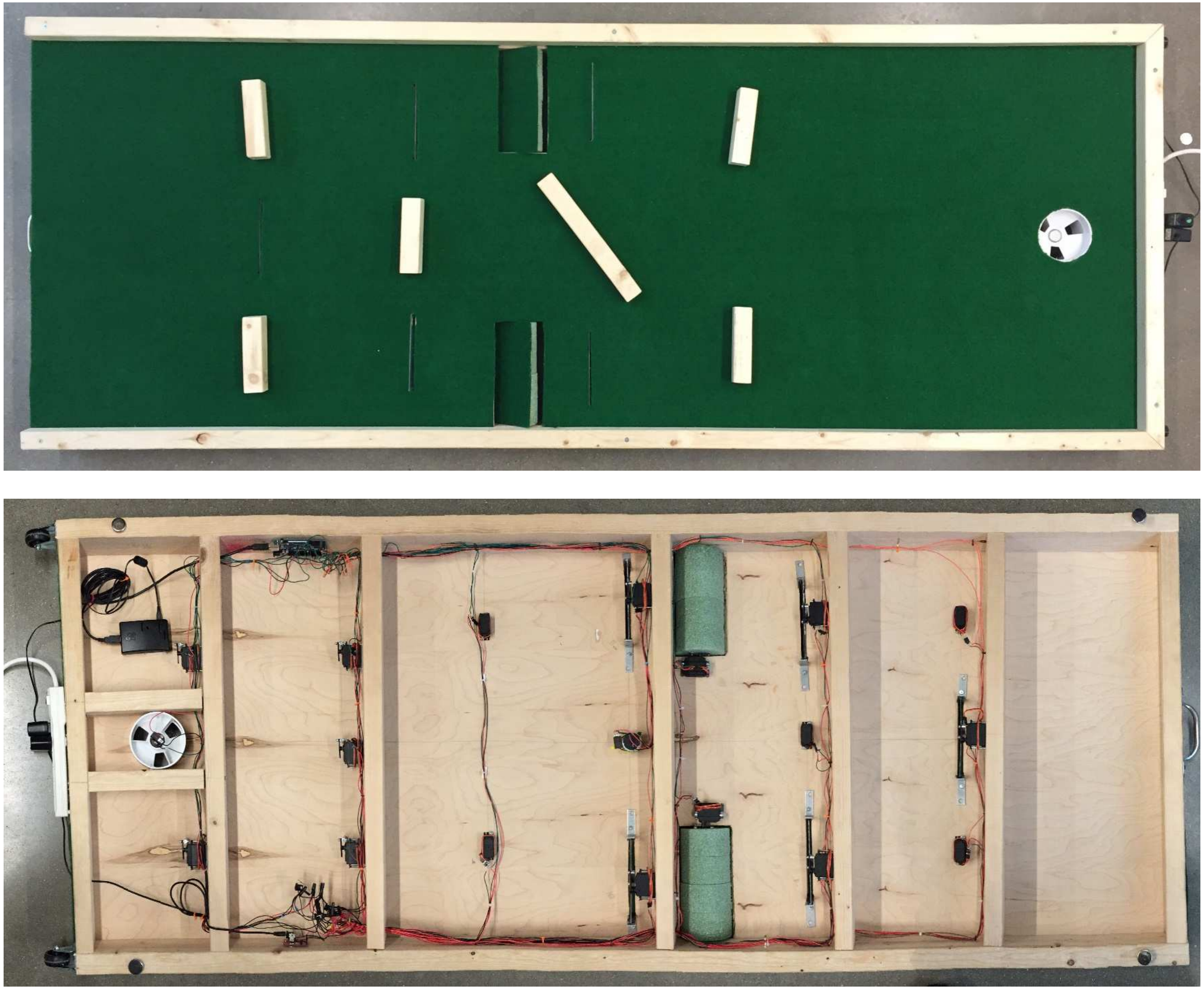


Figure 7: Top view of course showing obstacles (Top) and underside of course showing electronics (Bottom)

Evaluation of Specifications

The original specifications laid out for the project last semester are listed below, followed by an evaluation of whether or not each specification was met.

1. Physical Characteristics

1.1. <u>Width must be no larger than half the width of main Parkview corridor:</u>

The width of Parkview's main corridors is 9'8''. The width of the miniature golf course is 2'10". Therefore, the specification was met.

1.2. <u>Final product must be easily transported around ECE department:</u>

The miniature golf course was outfitted with wheels and a handle for easily transporting the system throughout the department. Therefore, the specification was met.

2. Functionality

2.1. <u>Course must use only 1 physical hole:</u>

The course only uses one physical hole. Therefore, this specification was met.

2.2. <u>Objects must be reconfigurable into 9 different holes, shown in Table 2 and Fig. 8:</u>

This specification was met by using servos to move the obstacles to the required positions, as described in the construction section above.

2.3. <u>Electronics must be low-voltage DC:</u>

Both the Arduino and Raspberry Pi operate at 5V DC. The servo motors operate at 5V DC while the DC motor operates at 12V DC. Therefore, the specification was met.

2.4. <u>A microcontroller must be used to reconfigure the course obstacles:</u>

This specification was met by using the Arduino to reconfigure all obstacles.

2.5. <u>The scores for each player, the current player's turn, and the current hole must be displayable to the player:</u>

This specification was met by using the Raspberry Pi to draw the current hole, player turn, and scores to the display screen.

2.6. <u>Ideally, the course would be able to communicate with the putter from the previous senior design group:</u>

Due to trouble getting the pervious senior design group's putter to work, it was decided to allow the players to enter their stroke count manually instead. However, the code that was written to take the input from the numeric keypad could be switched out to allow a possible implementation of the putter in the future.

3. Financial

3.1. <u>The total budget for the project must remain under $400:</u>

This specification was not met. The final budget for the project exceeded $400. The final total costs of all components, excluding the items that were donated, was $589. This was $189 over budget.

Table 2: Obstacle Configurations for Each Hole

Obstacle	Hole 1	Hole 2	Hole 3	Hole 4	Hole 5	Hole 6	Hole 7	Hole 8	Hole 9
T1	flat	flat	flat	hill	flat	flat	flat	flat	hill
T2	flat	flat	flat	hill	flat	flat	flat	flat	hill
T3	flat	flat	flat	flat	flat	flat	hill	flat	hill
T4	flat	flat	hill	hill	flat	flat	hill	hill	hill
T5	flat	flat	flat	flat	flat	flat	hill	flat	hill
A1	0°	0°	0°	45°	180°	45°	45°	0°	180°
A2	0°	0°	0°	45°	180°	45°	0°	180°	180°
P1	down	up	down	up	down	up	down	up	up
P2	down	up	down	up	down	up	up	down	up
P3	down	down	up	down	down	up	up	up	up
P4	down	down	up	down	down	up	down	up	up
P5	down	up	down	down	up	up	down	down	up
R1	0°	90°	45°	0°	60°	0°	45°	45°	315°
R2	0°	90°	315°	0°	300°	0°	45°	0°	45°
R3	0°	90°	90°	90°	90°	0°	45°	90°	0°
R4	0°	45°	90°	90°	150°	210°	45°	150°	0°
R5	0°	315°	90°	90°	210°	150°	45°	150°	0°
S1	off	on	on	off	on	off	on	off	on

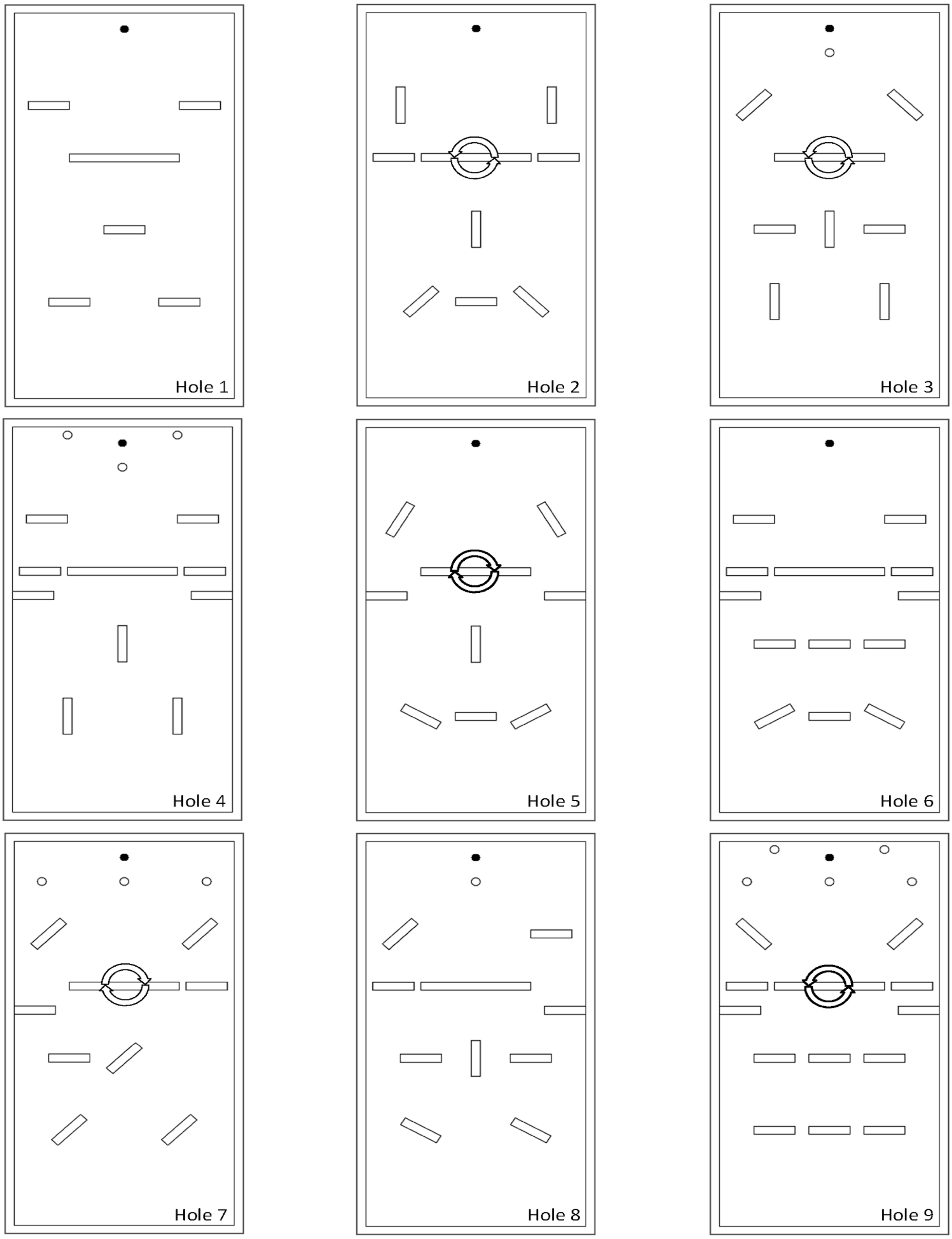


Figure 8: Hole configurations

Recommendations

If more time had been allowed for development of the miniature golf course we would have implemented the automatic stroke counting putters from last semesters senior design project. Additional game modes would be programmed for the player to choose from. For instance, a mode in which the hole configurations would be randomly generated, surprising the players with new “holes”. A top scores page to display the top performers over a period of time would have also been a nice addition.

References

Richardson, M., & Wallace, S. (2012). *Getting Started with Raspberry Pi* (1st ed.). Sebastopol, CA: Maker Media.

Code Libraries Used:

Pygame, (1999). Free Software Foundation, Inc. Boston, MA.

pySerial, (2013). Chris Liechti.

Arduino Servo Library, http://arduino.cc/en/reference/servo