Table Tennis Fault Serve Detection

By

Katelyn Riehl

Shraddha Dangi

Vishesh Verma

Final Report for ECE 445, Senior Design, Spring 2017

TA: Zipeng Wang

03 May 2017

Project No. 9

Abstract

In this paper, we discuss the design, implementation, and verification of a system to determine and report the correct call on the service in competitive doubles table tennis as an assistive tool to the umpire. The system consists of Power, Ultrasonic Sensor, Camera, and Output Display modules. Each module still has improvements that need to be made before a final product is ready; however, we were able to demonstrate basic functionality of each module independently as proof-of-concept for our design.

Contents

1 Introduction

In table tennis, the service is the most crucial part of every point. It dictates the style and pace of play for the point. In doubles, service becomes even more restrictive where the player can only serve from the right half of his side to the diagonally opposite side of the opponent. Often players try to "jam" the opponent by serving along the centerline of the table, making the serve challenging to return. However it is often difficult for the umpire to determine if a serve was "in" or "out" from the side of the table. With organizations like the National Collegiate Table Tennis Association, a doubles match is used as a tiebreaker between two teams who had split results in the singles stage. In such high-stakes matches, even one mistaken call can affect which teams move on in the competition. Our design addresses the issue of calls that may be difficult to determine "in" or "out" by detecting which side of the line the ball bounced.

2 Design

2.1 System Block Diagram

The table tennis fault serve detection system consists of four modules as seen in Fig. 1. Modules include the power supply, ultrasonic sensor module, camera module, and output display. The power supply module consists of two batteries, one of which is stepped down from 9 to 5 Volts as required by the ultrasonic sensor circuits and ATmega.

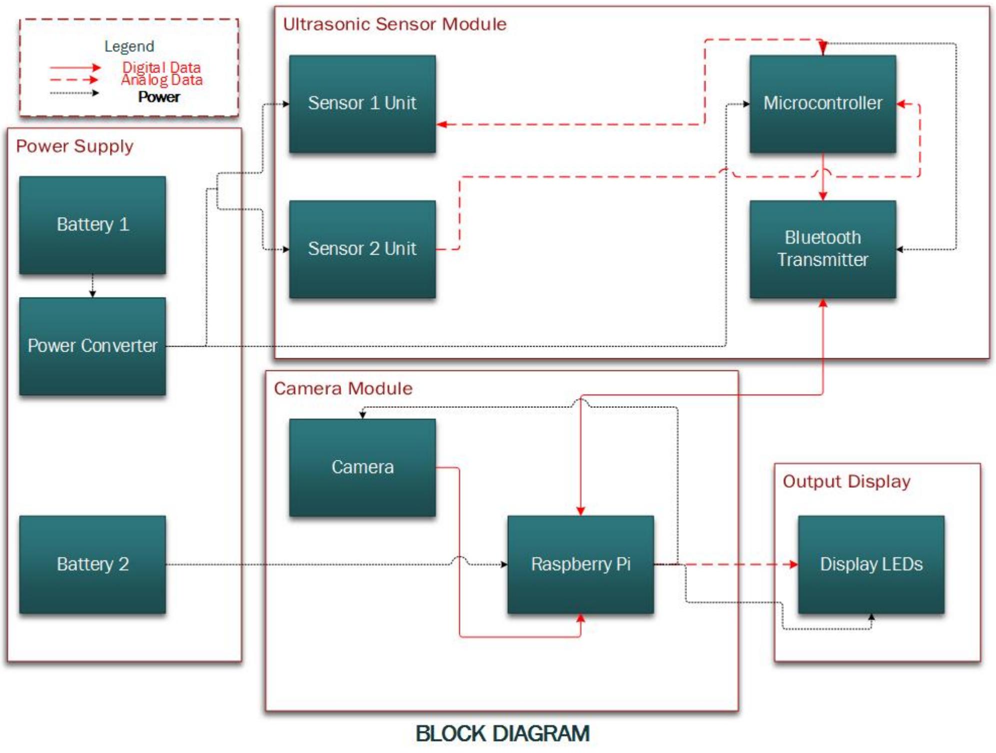


Figure 1: System block diagram

Located in the ultrasonic sensor module are the ultrasonic sensor circuits, ATmega microcontroller, and Bluetooth transmitter. The ultrasonic sensors continuously read in distance data from the object closest to them, which in our case would be the table tennis ball and relay this information to the ATmega which computes the correct call of the play.This data is wirelessly transmitted through Bluetooth to the Raspeberry Pi. The camera module consists of the Raspberry Pi microcontroller and camera. As the Raspberry Pi is receiving data from the ATmega, the camera is simultaneously tracking the ball and table using blob detection. When the ball reaches a certain height above the table, the corresponding sensor readings are used to determine whether the ball was "in" or "out". This information is sent to the output display where the LED lights up green for "in" or red for "out". The umpire can then press a button to reset the LED for the next play.

2.2 Physical Design

The design for the system as seen in Fig. 2 is solely for one player′s side of the table. For a complete system, a replicated design is needed for the opponent′s side. The detection system uses two ultrasonic sensors. The sensors are placed on opposing sides of the table and are attached to the net post. Utilizing the net posts helps ensure the sensors are aligned correctly as net post alignment is required for tournaments. Both sensors are angled towards the middle of the table to view at least 3.5 ft from the end of the table as a serve very rarely lands in the remaining 1 ft. This was verified through experimentation. Fig. 3 shows empirically that serves don′t generally land within the first ft. from the net. The ATmega, Bluetooth transmitter, and power supply are all attached to the underside of the table as to not hinder gameplay.

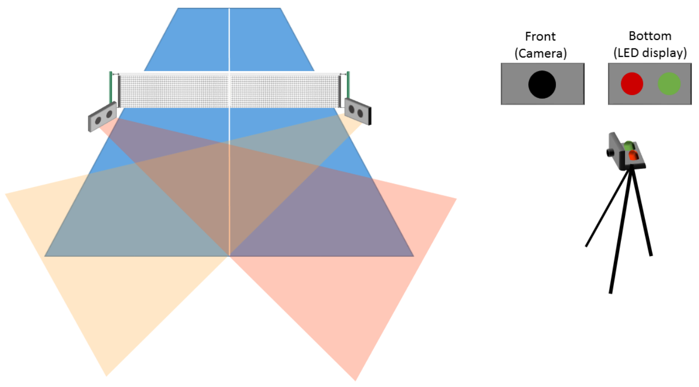


Figure 2: Physical design

The camera module and output display are located near the umpire approximately 12 ft. away from the table. The camera is mounted level with the table, approximately 2.5 ft. above the ground. By choosing such a height, the viewing angle is similar to that of Fig. 4. The output display is located behind the camera for easy observation and accessibility to the button for the umpire. The battery and Raspberry Pi are located with the output display.

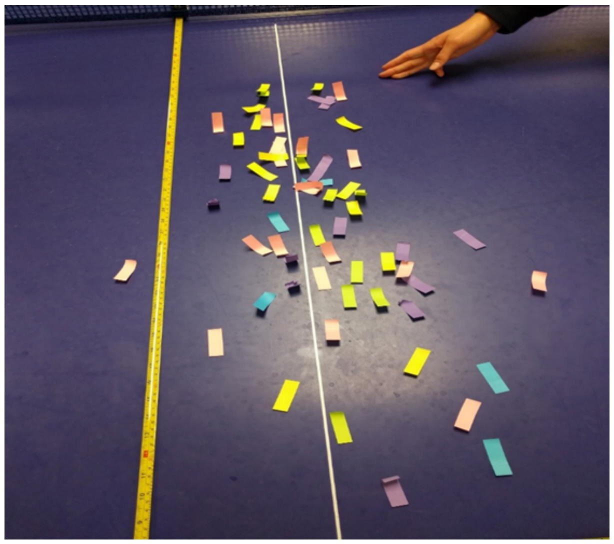


Figure 3: Experimental serve bounce locations

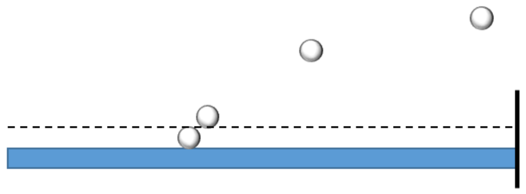


Figure 4: Camera view of table

2.3 Power Supply

2.3.1 Battery

As the sensor module is located approximately 12 ft. from the camera module and output display, two power supplies are used in the implementation of the design. The camera module and output display are powered by a 5V/2.4A (6400mAh) rechargeable power bank. Because the Raspberry Pi is a relatively large consumer of power, choosing a battery of this particular scale allows the Raspberry Pi and output display to last approximately 2.5 hours. The sensor module is powered by a 9V to 5V buck converter. The 9V supply is provided by a 9V (500 mAh) battery.

2.3.2 Power Converter

Because the ultrasonic sensor circuit and ATmega require 5V, a 9V to 5V buck converter is used. The buck converter has a voltage tolerance of ±5% and a maximum current requirement of 70 mA. This current requirement was decided based upon the current draws from the two ultrasonic sensor circuits and ATmega as seen in Table 1. The buck converter utilizes the Texas Instruments

TPS54202 integrated circuit (IC). Features of the IC include over-current protection, over-voltage protection, and a feedback loop help ensure the intended output voltage is achieved [1]. Fig. 5 displays the circuit schematic for the buck converter.

Table 1: Ultrasonic sensor module voltage and current consumption

Component	**Voltage (5)**	**Current (mA)**
Ultrasonic Sensors (x2)	5	∼30
ATmega	5	4.1

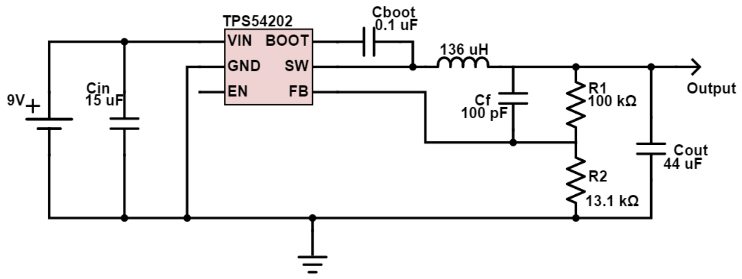


Figure 5: Buck converter circuit schematic

Found in the datasheet [1] were recommendations of 10.1 μF (minimum), 0.1 μF, 75 pF (minimum), and 100 kΩ for C_{in}, C_{boot}, C_f, and R_1 respectively. Based on these suggestions, as well as availability of components, values of 15 μF, 0.1 μF, 100 pF, and 100 kΩ were chosen for the design.

The value of R_2 was calculated using Eq. (1). This equation makes use of a voltage divider used for the feedback of the IC. A given value of 0.596 was used for the reference voltage, V_{ref} [1]. As the ultrasonic sensor circuit and ATmega both use 5V, that value was used as V_{out}. Using Eq. (1), a value of 13.5 kΩ is calculated for R_2. After testing, this value was reduced to 13.1 kΩ.

$$R_2 = \frac{R_1 * V_{ref}}{V_{out} - V_{ref}} \tag{1}$$

The output capacitance was then calculated using Eq. (2). A given value of 500 kHz was used for f_{sw}, 0.25V for ΔV_{out} based on design parameters, and 70mA for ΔI_{out} based on the design. Using these values, C_{out} is determined to be 1.12 μF. Because a larger output capacitance reduces ripple, a value of 44 μF was chosen given additional testing.

$$C_{out} = \frac{2\Delta I_{out}}{f_{sw} * V_{out}} \tag{2}$$

To determine a value for the feed-forward capacitor, C_f, the crossover frequency is first calculated using Eq. (3). This results in a value of approximately 18 kHz which is then used in Eq. (4) to calculate C$_f$. Using a value of 100 kΩ for R$_1$, the minimum feed-forward capacitance value equates to 88.6 pF. For the design, a value of 100 pF was chosen.

$$f_0 = \frac{3.95}{V_{out} * C_{out}} \tag{3}$$

$$C_f = \frac{1}{2\pi f_0}\frac{1}{R_1} \tag{4}$$

For the inductance, a value of 15 μH was suggested for an output of 5V. Based on testing results, a final value of 136 μH was chosen. Eqs. (1-4) were all given in the datasheet [1].

2.4 Ultrasonic Sensor Module

2.4.1 Ultrasonic Sensor Unit

Design

The two ultrasonic sensor units consists of transmitting and receiving sensors and the circuitry that drives them. These sensors are fixed on the net post, angled towards the table so as to have a detection range as specified in Fig. 2.

The principle of ultrasonic ranging is shown in Fig. 6. The transmitting sensor emits a sound wave in the ultrasonic range and when this wave hits an object, it gets reflected back. This reflection is picked up by the receiving sensor and the time difference between the sent and received pulse is measured.Using Eq. (5), with speed of sound in dry air = 343m/s, we are able to compute the distance to the object in range. The time is divided by 2 to account for forward and reverse wave propagation.

$$distance = \frac{time * 343}{2} m \tag{5}$$

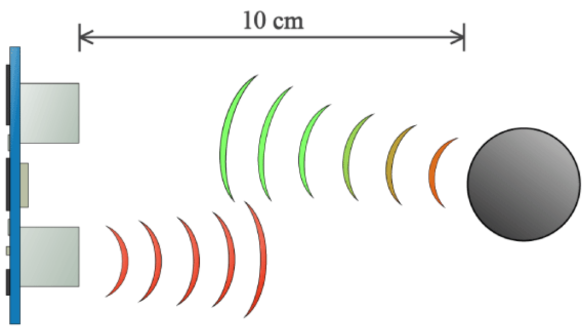


Figure 6: Ultrasonic Ranging[2]

In our design, the transmitter unit will emit a fixed number of square wave pulses using a LM555 timer circuit described in detail in transmitter section below. The frequency of these pulses will be 40 KHz which is in the ultrasonic range. For our implementation, we use differencing to determine which sensor the ball bounced closer to. Both the sensors will transmit at the same time and are placed on opposite ends of the table. From a high level standpoint, this problem comes down to which sensor received the reflected signal first. If sensor 1 received the reflected pulse first, the ball is closer to that side of the table.

A high level block diagram of the sensor module is shown in Fig. 7. The operation of the transmitter and receiver is controlled by the ATmega Chip. The ATmega controller is connected to the reset pin of the both the transmitter LM555 timers through Hex inverters and also accepts the amplified input from the sensor receivers.

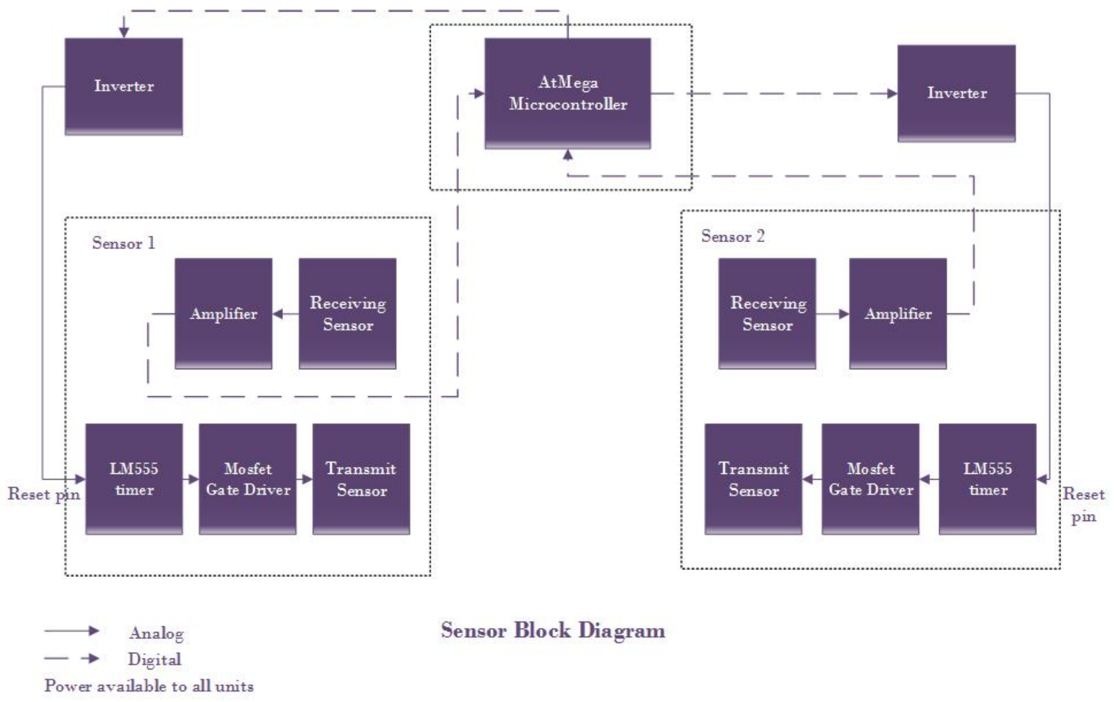


Figure 7: Sensor Unit Block Diagram

<u>Transmitter Circuit</u>

The complete transmitter circuit consists of the LM555 timing circuit with a MOSFET driver IC for the transmitting sensor. The driver is necessary as the sensor is a large capacitive load. It also includes the hex inverter IC, which is controlled by the ATmega328p IC and connected to the reset pin of the LM555 timer. The reset pin of the LM555 timer is active low. This means that when the input is low, the output of the timer is reset to 0V and does not transmit a pulse.The circuit schematic and EAGLE board layout are shown in Figs 8 and 9 respectively.

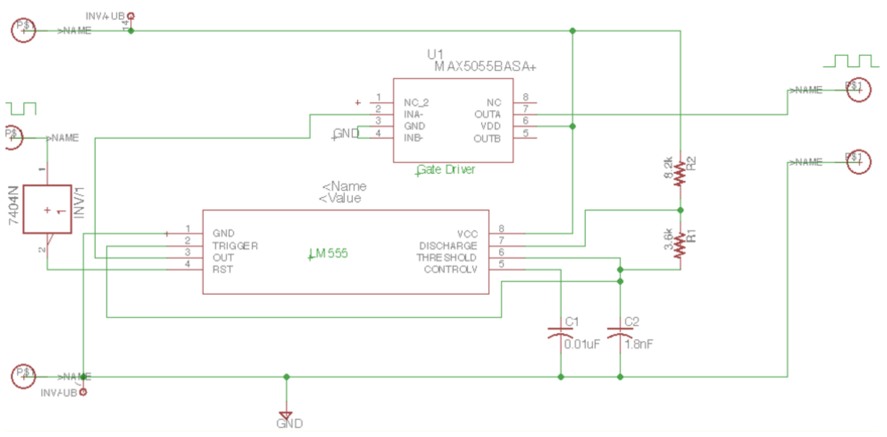


Figure 8: Ultrasonic sensor transmitter circuit schematic

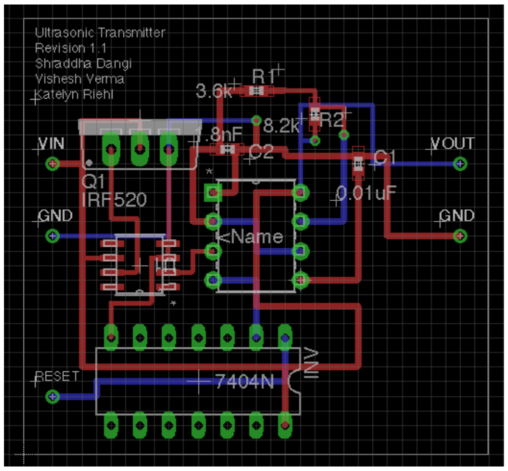


Figure 9: Ultrasonic sensor circuit PCB

The LM555 timer is being used in the astable mode where it acts as a multivibrator continuously cycling between a high and low output value. This means that it will continuously transmit a square wave pulse until it is reset. The frequency and time period of the waveform is calculated by the equations below. We chose our R1, R2, and C to satisfy Eq. (6) to get a 40 kHz square waveform in the ultrasonic frequency range. Based on commonly available capacitor and resistor values, we chose R2=3.8 kΩ, R1=8.2k Ω, and C2= 1.8nF.

$$f_c = \frac{1.44 * C}{R_2 + 2R_1} \quad \text{where } f_c \text{ is the oscillation frequency} \tag{6}$$

The time period of the output square wave pulse is calculated by Eqs. (7-9) where t_h is the time for which the pulse is high and t_l the time which the pulse is low. Together, they add up to give the time period of the waveform with a duty cycle of 59.8%.

$$t_h = 0.693 * (R_2 + R_1) * C \quad (7)$$

$$t_h = 0.693 * R_2 * C \quad (8)$$

$$T = t_h + t_l = 0.0249ms \quad (9)$$

Lastly, the output of the LM555 module is fed into a MOSFET gate driver which drives the large capacitive load which is the sensor.

Timing Diagram

The timing diagram for our implementation is shown in Fig. 10. For our design we chose to send 4 pulses for ease of detection. This equates to a transmit time of 100μs. The reset time is determined by the furthest distance the sensor can "see". For example, if the ball is at the farthest corner of the table from the sensor diagonally 6.7ft away, a wave reflected from that ball will reach the receiver in 11.9 ms at room temperature. We add a buffer of 3 feet to this distance as the player stands within that range and his/her movements could also reflect back to the sensor. Therefore, our total distance of concern is 9.7 feet, and the corresponding time between transmit and receive for that distance is 17.2 ms at room temperature. Therefore, to remove delayed reflections,we set our reset time to 17.2 ms.

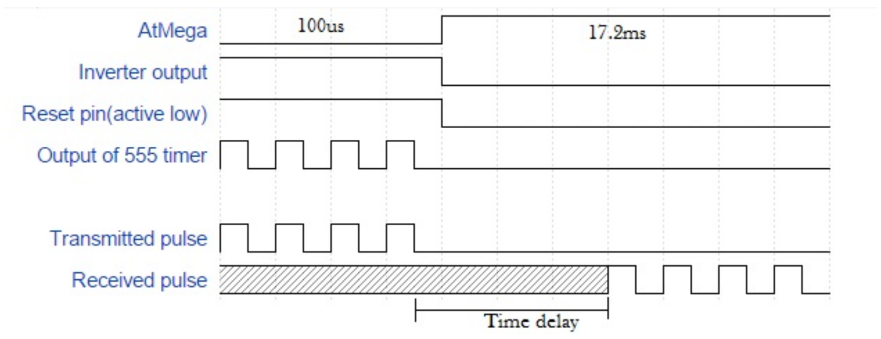


Figure 10: Timing Diagram [3]

Receiver Circuit

The receiver circuit consists of the receiver sensor and an amplifier. The amplifier is set up with an operational amplifier in feedback to achieve a specified gain. The desired gain will be determined by observing the strength of the received signal from the ultrasonic sensor and comparing it to the op-amp's saturation voltage (5V). To implement the amplifier with the op-amp, we needed to compute the resistor values for R_1 and R_2 For the amplifier design, the gain G can be computed as shown in Eq. 10.

$$G = 1 + \frac{R_1}{R_2} \quad (10)$$

This circuit schematic and EAGLE PCB layout is shown in Fig. 11. The resistor values are specified for a gain of 100 and can be easily changed for a different gain if needed.

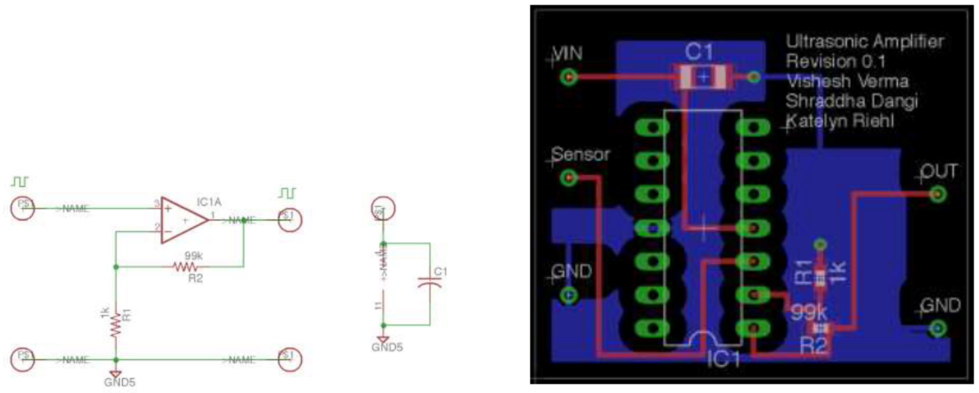


Figure 11: Receiver Circuit (left) and EAGLE Board schematic(right)

2.4.2 ATmega Microcontroller

Because a complex microcontroller is not necessary, the ATmega328p is used in the ultrasonic sensor module at 16MHz. The ATmega continuously receives time delay readings via digital pins from both ultrasonic sensors. It then analyzes the information to determine the distance of the ball from each sensor. These distance readings are then compared to detect which sensor the ball bounced closer to. This information is sent to the Bluetooth transmitter via the TX pin. Fig. 12 shows the circuit schematic for the microcontroller [4].

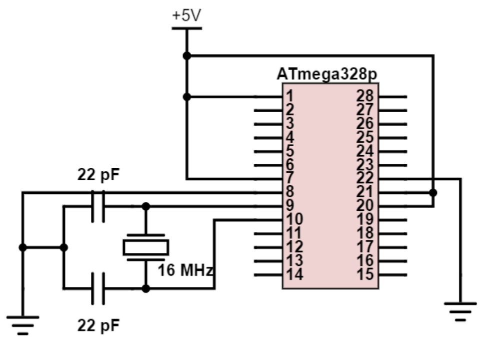


Figure 12: ATmega circuit schematic

The ATmega is also used to provide a trigger to the reset pin of the LM555 timer. This is done by continuously outputting a “LOW” signal for 100 μs followed by a “HIGH” signal for 17.2 ms (code found in Appendix B). This allows the timer to produce a 4 pulse square wave with a 17.2 ms delay before the next set of pulses are sent.

2.4.3 Bluetooth Transmitter

The HC-05 is used as the Bluetooth transmitter. Once the comparison is made in the ATmega, the Bluetooth module sends this result to the Raspberry Pi. The circuit schematic for the Bluetooth transmitter can be seen in Fig. 13. A voltage divider is used to lower the voltage to 3.3V as required by the RX pin of the HC-05.

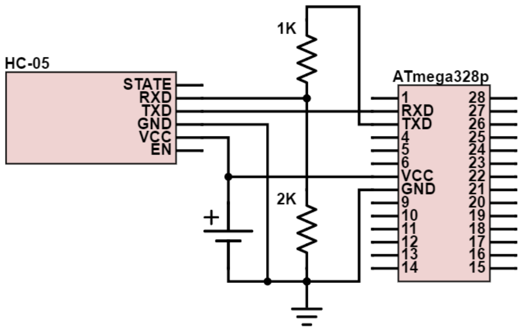


Figure 13: HC-05 circuit schematic

2.5 Camera Module

2.5.1 Raspberry Pi Microcontroller

The Raspberry Pi microcontroller is used to process video feed from the camera. Using OpenCV, the Raspberry Pi searches each frame for colors within a certain pre-specified range corresponding to the ball or table. This is converted into a binary image file, with "1" representing pixels with value inside the aforementioned range, and "0" representing pixels with value outside the range. This was performed for two color ranges corresponding to the ball and the table. A sample of the original and binary images can be seen in Figure 14. Once this comparison has been performed, the Raspberry Pi determines the largest area of "1"s and calls this the object of interest. [5]

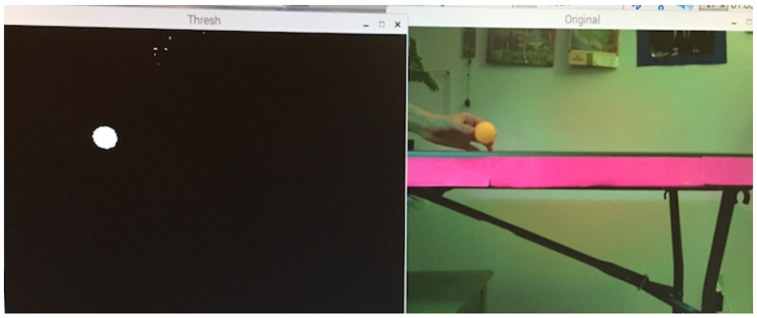


Figure 14: A single frame from the camera (right) and generated binary image for the ball (left)

In the case of the ball the object must be a circle, and in the case of the table the object must be a rectangle. Using the locations of the center of the ball and top of the table, the Raspberry Pi can determine the height of the ball above the table. Once within a certain threshold height of the table, the Raspberry Pi will read the appropriate value from the sensor readings (sent via bluetooth) to make a determination of whether the serve was “in” or “out”. A flowchart for the described software can be seen in Fig. 15. The implemented code can be seen in the Appendix C.

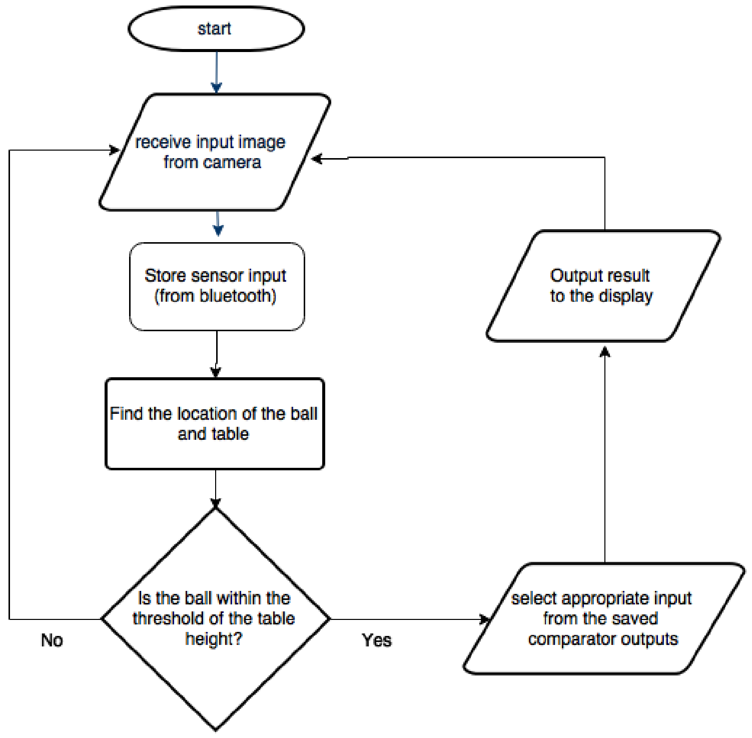

start
receive input image from camera
Store sensor input (from bluetooth)
Find the location of the ball and table
Is the ball within the threshold of the table height?
No
Yes
select appropriate input from the saved comparator outputs
Output result to the display

Figure 15: Software flowchart

2.5.2 Camera

The camera module used to capture the video of the ball bouncing was specified to be capable of operating with resolution 640x480 at up to 90 frames per second. The camera is situated near the umpire′s table at the same height as the playing table. This allows the camera to observe the ball as it approaches the table, bounces, and leaves the table. To determine an appropriate frame rate for the camera, we performed preliminary testing using a GoPro camera, which operates at 30 frames per second. The GoPro camera was able to capture frames showing the descent, bounce, and ascent of the ball. This confirmed that, with a higher frame rate camera, the selected camera would be able to detect the bounce of the ball as desired, and would also show more frames close to the time when the ball contacts the table.

2.6 Output Display

The initial implementation of the output LED display board was with a finite state machine built in hardware. The inputs are any valid sensor reading when the ball contacts the table, regardless of whether or not the bounce was a serve. As shown in Fig. 16, the state machine would transition on the input it receives (on the serve), and holds its output until the button is pressed to reset it.

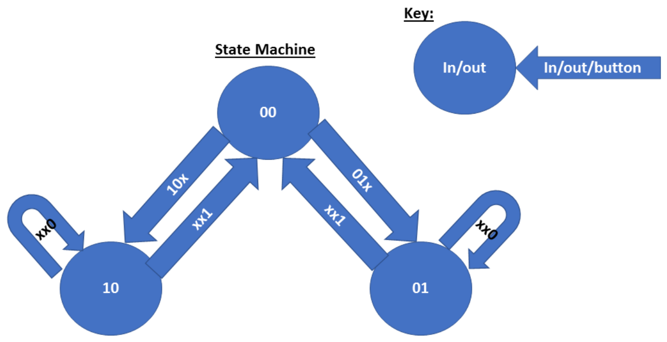


Figure 16: Output display finite state machine

3 Design Verification

3.1 Power Supply

3.1.1 Battery

As table tennis matches last at most 45 minutes, the battery was tested to ensure that the buck converter could be powered for the whole match using a single 9V battery. To verify this requirement, a 9V battery was connected as the input voltage to the buck converter. A load representing the ultrasonic sensor module was placed as the output of the buck converter. Results were recorded for the input voltage, output voltage, and output voltage ripple. Data for the test can be seen in Table 2. According to the results, the battery voltage drops approximately 0.5V over the time frame. It can be noted that the starting voltage of the battery is 8.7V due to prior testing.

Table 2: Battery voltage over 45 minutes

Minutes	$\mathbf{V}_{dc,battery}$	$\mathbf{V}_{DC,load}$	$\mathbf{V}_{p-p,load}$
0	8.7 V	5.03 V	40 mV
5	8.6 V	5.03 V	42 mV
10	8.5 V	5.03 V	59 mV
15	8.5 V	5.03 V	54 mV
20	8.4 V	5.03 V	40 mV
25	8.4 V	5.03 V	54 mV
30	8.3 V	5.03 V	31 mV
35	8.3 V	5.03 V	67 mV
40	8.2 V	5.03 V	33 mV
45	8.2 V	5.03 V	48 mV

3.1.2 Power Converter

To be viable in the design, the buck converter was tested to ensure it could supply 5V ±5% and a current of at most 70 mA. The input voltage of 9V was used from the power supply, and a load of 75 Ω was placed as the load. The voltage across the output is approximately 5.03V as seen by Fig. 17. Using this voltage and the resistance value of 75 Ω, an output current of 67.1 mA is calculated using Eq. (11).

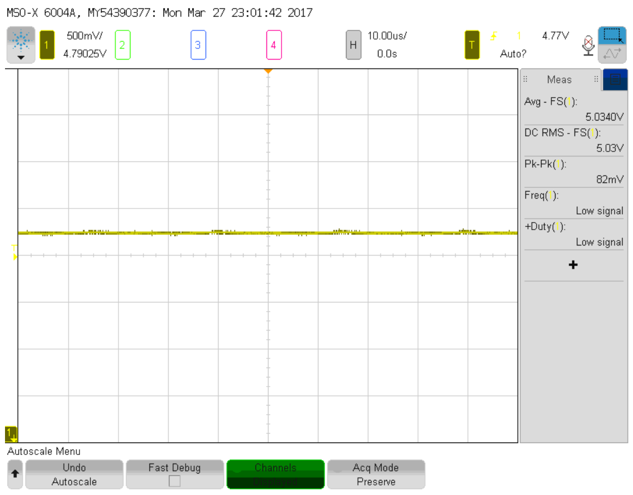


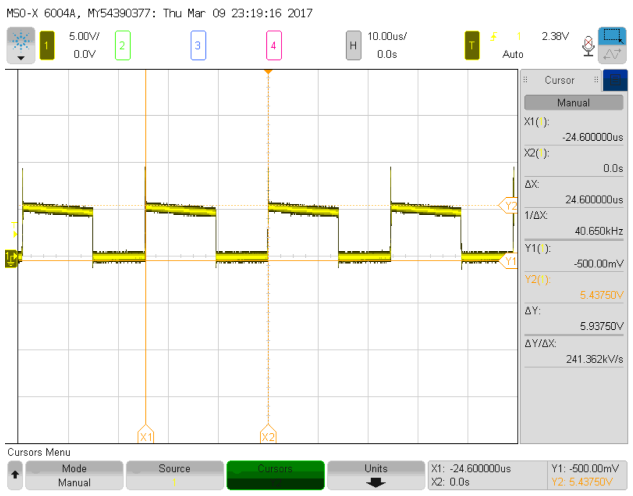
Figure 17: Buck converter output voltage

$$I = \frac{V}{R} \tag{11}$$

3.2 Ultrasonic Sensor Module

3.2.1 Ultrasonic Sensors

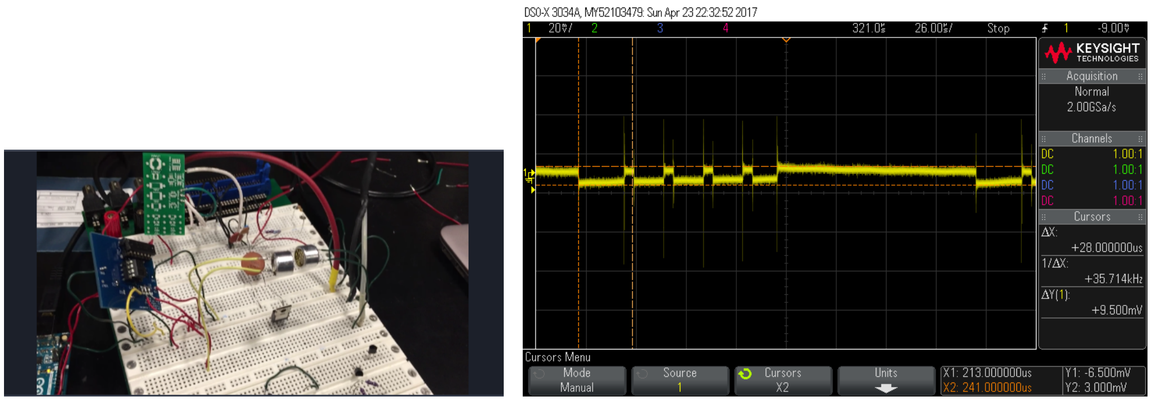
We assembled the LM555 timing circuit on a breadboard with the LM555 chip and the required resistor and capacitor values. The output of this lab test is shown on the oscilloscope in Fig. 18. The frequency is 40.65 kHz, and the time period is 0.02496ms which is only slightly different than the calculated value. This was expected due to tolerance issues with passive components. A 10% tolerance of R1, R2 and C, the frequency will fluctuate from 33.97 kHz to 49.38 kHz. This is not an issue as these frequencies are still in the ultrasonic range and will travel at the speed of sound. The spikes seen in the graph are due to capacitive coupling which occurs in breadboard connections. This will not occur when the circuit is implemented in a PCB.



```
MSO-X 6004A, MY54390377: Thu Mar 09 23:19:16 2017
5.00V/  0.0V   10.00us/  0.0s   2.38V  Auto
Cursor  Manual
X1( ): -24.600000us
X2( ): 0.0s
ΔX: 24.600000us
1/ΔX: 40.650kHz
Y1( ): -500.00mV
Y2( ): 5.43750V
ΔY: 5.93750V
ΔY/ΔX: 241.362kV/s
Cursors Menu
Mode Manual   Source 1   Cursors   Units
X1: -24.600000us  X2: 0.0s   Y1: -500.00mV  Y2: 5.43750V
```

Figure 18: Tested waveform of assembled LM555 timing circuit

We chose the UT-1240K-TT-R[6] and the UT-1240K-TT-R[7] transmitter and receiver sensors, and the MAX5055 Dual MOSFET DRIVER because of its compactness and fast switching speed[8]. The complete assembled transmitter circuit and the waveform generated (without amplification) at the output of the receiver sensor is shown in Figure 19.



```
DSO-X 3034A, MY52103479: Sun Apr 23 22:32:52 2017
20mV/   321.0us  26.00us/  Stop   1  -9.00mV
KEYSIGHT TECHNOLOGIES
Acquisition  Normal  2.00GSa/s
Channels  DC 1.00:1  DC 1.00:1  DC 1.00:1  DC 1.00:1
Cursors
ΔX: +28.000000us
1/ΔX: +35.714kHz
ΔY(1): +9.500mV
Cursors Menu
Mode Manual   Source 1   Cursors X2   Units
X1: 213.000000us  X2: 241.000000us   Y1: -6.500mV  Y2: 3.000mV
```

Figure 19: Assembled Transmitter Circuit (left) and output at reciever sensor(right)

In the setup displayed in Figure 19, we placed the sensors end to end to test their functionality. The waveform received on the receiving sensor is also displayed to the right of Figure 19. It is inverted compared to the desired waveform as was specified in Fig. 10. This is due to the fact that the MAX5055 is an inverting MOSFET Driver[8]. To correct this, we would invert the signal at the output of the LM555 timer, before it is fed into the Driver.

Another aspect related to the waveform is that the amplitude of the received signal is about 10mV even when the sensors are facing each other. This amplitude is very low and and would not be detectable by the ATmega. A possible solution to this problem is using multi-stage amplifier, the gain of which will be decided by the lowest possible valid reflection.

To display proof of concept, we used the Arduino ping sensors, the HC - SR04 which have a sensing range of 2cm-4m and accuracy of 3mm [9]. This was good for our implementation as the center line on a table tennis table is 3mm wide. So, to accurately determine the location of the ball, we would need to make sure that our own sensor circuit achieved that accuracy of upto 3mm. The code used to run these ping sensors is included in Appendix A.

The Integration of the ping sensors with the table is shown in Fig.20. Placement of the sensors was critical as they have to be equal distance form a center point on the table and have to be angled the same way to get the most accurate results. Through testing with these ping sensors we were able to read distance values of the ball and determine which side of the table the ball bounced.

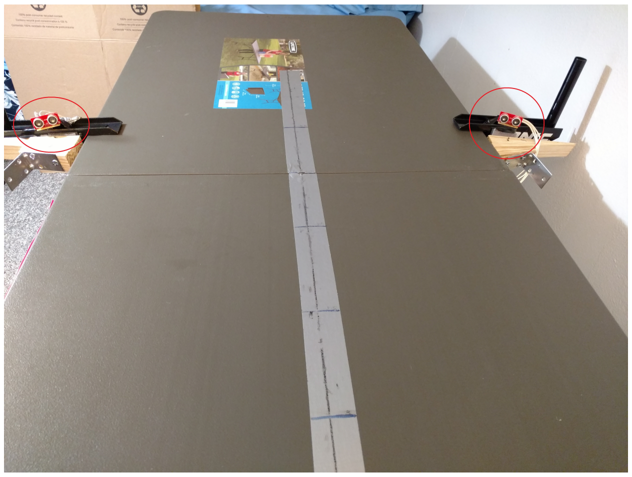


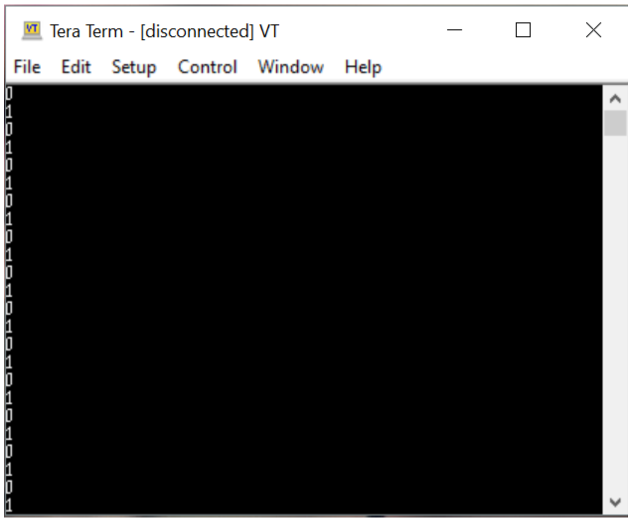
Figure 20: Integration of ping sensor with the system

3.2.2 ATmega Microcontroller

Due to time constraints and a focus on the other modules, the ATmega board microcontroller was not implemented. Components whose functionality depends on the ATmega were connected to an Arduino Uno. The Arduino Uno can provide as the proof-of-concept of the ATmega board.

3.2.3 Bluetooth Transmitter

Because information between the ultrasonic sensor module and Raspberry Pi are located approximately 12 ft. away from each other, the Bluetooth transmitter was tested accordingly. After connecting the HC-05 to an Arduino similarly to Fig. 13, the device was paired with a laptop. Keeping the laptop at least 12 ft. away from the HC-05, data was then transmitted and observed. Fig. 21 displays the received data from the Bluetooth transmitter which corresponds to the closer sensor.



```
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
```

Figure 21: Bluetooth transmitted data 12 feet away

3.3 Camera Module

3.3.1 Raspberry Pi Microcontroller

An essential aspect of the Raspberry Pi block is the ability to track the ball and the top of the table. Both objects are reliably able to be tracked. For simplicity of calibration, we attached pink paper to the side of our table. The results of this tracking can be seen in Fig. 22. It can be seen that there is a yellow circle surrounding the ball, with a red dot indicating its center. There is also a blue marker at the left end of the pink part of the table indicating the level of the table. Using the maximum y-position gives us the height of the table.

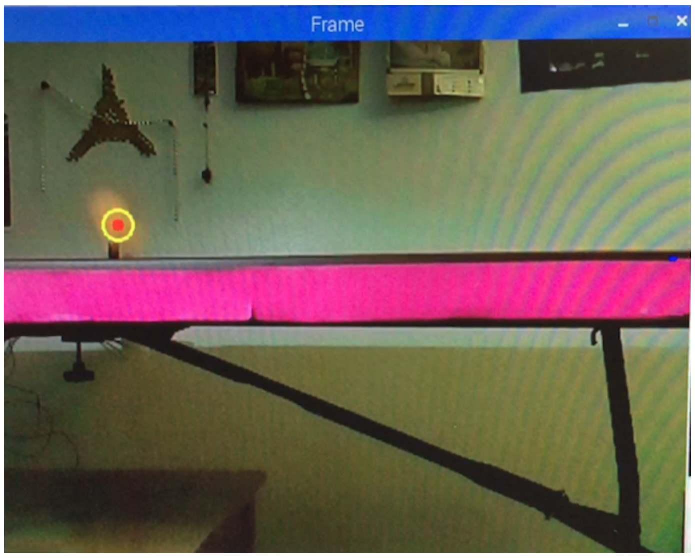


Figure 22: Ball detection and tracking

Though the tracking works as desired, the frame rate of the video was about 10 frames per second, which is much less than the desired 90 frames per second. Additionally, in order to complete implementation of our design, Bluetooth receiving would need to be integrated into the software.

3.3.2 Camera

The camera did not meet the desired specifications for the project. The frame rate was tested by having the software count the number of analyzed frames while running for 30 seconds, and print on exiting. Even with minimal additional processing, the camera displayed images with maximum frame rate of about 55 frames per second. Upon integration with the computer vision code, the camera displays images with frame rate of about 10 frames per second.

3.4 Output Display

The verification of the LED board was quite simple. A GPIO pin is set high to indicate an "in" reading, and the green LED turned on. Another GPIO pin is set high to indicate an "out" reading, and the red LED turned on. The button is connected to a third GPIO pin. However, the reading of the input was not yet functional by the time of the demo.

The original board′s with the hardware logic did not function as desired, and debugging yielded no results. Therefore we decided to implement this in software instead and focus on the more critical aspects of the project.

4 Cost

4.1 Parts

Table 3: Parts Costs

Part	Manufacturer/Distributor	Unit Cost ($)	Quantity	Total ($)
HC-05	DSD Tech	7.99	1	7.99
TPS54202	TI	2.04	1	2.04
ATmega328p	Atmel	2.14	1	2.14
Dip Socket	Digikey	0.33	1	0.33
Camera Mount	Scorpi	12.95	1	12.95
Ultrasonic Transmitter	Digikey	4.95	2	9.90
Ultrasonic Receiver	Digikey	4.95	2	9.90
Resistor 8.2K 0402	Digikey	0.02	1	0.02
Resistor 3.6K 0402	Digikey	0.02	1	0.02
Capacitor 10000pF 0402	Digikey	0.01	1	0.01
Capacitor 1800pF 0402	Digikey	0.01	1	0.01
MAX5055BASA	Maxim Integrated	9.31	2	18.62
Capacitora 0.1uF	Digikey	0.13	1	0.13
Misc. Ics	ECE Store	6.61	1	6.61
9V Connector	ECE Store	0.25	1	0.25
9V Battery	ECE Store	1.34	1	1.34
Mount Supplies	Lowe′s	17.71	1	17.71
Camera	LoveRPi	21.99	1	21.99
Raspberry Pi	Adafruit	47.09	1	47.09
LED	Digikey	3.23	1	3.23
			Total	**162.28**

4.2 Labor

Table 4: Labor Costs

Name	Rate ($/hr)	Hours	Total ($)
Shraddha	30	250	7,500
Katelyn	30	250	7,500
Vishesh	30	250	7,500
		Total	**22,500**
		Grand Total	**22,662.28**

5 Conclusion

5.1 Accomplishments

We were able to accomplish much in our design this semester. Although our design was not fully integrated, we were able to show functionality of most of the modules throughout the semester. Our power supply was fully functional and met voltage and current requirements. By using the off-the-shelf sensors, it could be determined which side of the line the table tennis ball bounced. This was a big step in our design as it helped prove that ultrasonic sensors were a viable option. It showed us that if we could adequately amplify the transmitted pulses we received in our designed ultrasonic sensor circuit, our sensors could be a substitute for the off-the-shelf sensors. Another big accomplishment in our project was the ability to track both the ball and table as this was one of the main components in our design.

5.2 Uncertainties

There are a few issues with the current state of the system that may affect the intended purpose of the design. The main uncertainty is with the ultrasonic sensors. Because we were unable to complete the receiver circuit, it is hard to determine the accuracy of the ultrasonic sensors. To confirm that the sensors would be accurate enough for our intended use, experiments would need to be conducted. Another uncertainty with the system is whether the frame rate of the camera could be increased adequately. Although parallel programming is the option that would need to be explored, we cannot say that it would be a significant enough increase.

5.3 Ethical and Safety Considerations

Design of the table tennis detection system required the considerations of several ethical and safety issues. The following statements from the IEEE Code of Ethics [10] were relevant in the design of the system.

IEEE Code of Ethics #1 - “to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment”

It is important that the design is as safe as possible for the end user. This included minimizing the risk of overheating components, determining ways to fixate units to their predetermined location, and stating any possible safety hazards. Components were chosen keeping voltage and current limits in consideration. By doing so, the risk of components combusting, which could result in harm to anyone nearby as well as the equipment, is reduced. Another concern in the safety of the user is the risk of objects acting as projectiles if hit. To minimize this concern, components are thoroughly secured at their locations. Lastly, there is also potential safety concerns with the use of the lithium-ion rechargeable battery including both chemical and electrical hazards. Lithium-ion batteries consist of lithium for the anode as well as a material for the cathode which is usually nickel or manganese and cobalt [11]. Although the battery should not release any chemicals, there is the potential risk of leakage. This could be due to corrosion or damage to the battery. Also, there is a possibility that the battery may overheat and/or combust if it is overcharged. The increase in temperature may cause instability in the battery.

IEEE Code of Ethics #3 - "to be honest and realistic in stating claims or estimates based on available data"

It must be stated that our system is likely not 100% accurate. Because the system is not manufactured, there exists the possibility of error in the accuracy. Also, the design should not be altered in any way. Changes may affect accuracy if not recalibrated.

5.4 Future Work

One of the most important improvements needed is to improve the frame rate of the Raspberry Pi camera. It was determined that the primary cause of this delay between frames is the the time taken by the computation needed for tracking the ball and table. This can be dramatically sped up by utilizing the onboard GPU and parallelizing the program. This process would require not only parallelizing the code we created but also the underlying libraries used providing many-fold improvement on runtime.

Although it was proven that the HC-05 had the ability to transmit data the appropriate distance for the design, communication was not established between the ATmega and Raspberry Pi. The next step for complete functionality of the Bluetooth transmitter is to set up a Bluetooth connection between the HC-05 and Raspberry Pi. This requires launching the Raspberry Pi Bluetooth tool and pairing the device manually [12]. Once paired, the HC-05 will be able to transmit data to the Raspberry Pi.

Adding the functionality of the button is the last step towards completing the LED board. This provides the feedback back to the Raspberry Pi that the next point is about to start.

Finally, this unit could be considered the first step towards an automated score-keeping or umpiring system. Additional systems that would need to be designed and implemented for this include: score-tracking, general point detection (not specific to serves), serve legality detection, interference detection (i.e. loose balls entering the field of play), and net service detection.

References

[1] "TPS54202 4.5-V to 28-V Input, 2-A Output, EMI Friendly Synchronous Step Down Converter," Datasheet, April 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/tps54202.pdf

[2] "Ultrasonic Sensor HC-SR04 and Arduino Tutorial," Web page, accessed May 2017. [Online]. Available: http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

[3] "Timing Diagram Generator," Web page, accessed May 2017. [Online]. Available: http://wavedrom.com/editor.html

[4] "From Arduino to a Microcontroller on a Breadboard," Web page, accessed April 2017. [Online]. Available: https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard

[5] "Ball Tracking with OpenCV," Web page, accessed April 2017. [Online]. Available: http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/

[6] "Ultrasonic transmitter 30vrms, ut-1240k-tt-r," Datasheet, April 2016. [Online]. Available: http://media.digikey.com/pdf/Data%20Sheets/Projects%20Unlimited%20PDFs/UT-1240K-TTR.pdf

[7] "Pui audio, inc, ultrasonic receiver 30vrms, ut-1240k-tt-r," Datasheet, April 2016. [Online]. Available: http://www.puiaudio.com/pdf/UR-1240K-TT-R.pdf

[8] "4a, 20ns, dual mosfet drivers," Datasheet, April 2016. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX5054-MAX5057.pdf

[9] "Ultrasonic ranging module hc - sr04," Datasheet, April 2016. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[10] "IEEE Code of Ethics," Web page, accessed May 2017. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html

[11] R. aisbl, "Safety of Lithium-Ion Batteries," The European Association for Advanced Rechargeable BAtteries, Brussels, Belgium, PDF, June 213.

[12] "Everything You Need to Set Up Bluetooth on the Raspberry Pi 3," Web page, accessed April 2017. [Online]. Available: http://lifehacker.com/everything-you-need-to-set-up-bluetooth-on-the-raspberr-1768482065

[13] "NewPing Library for Arduino," Web page, accessed April 2017. [Online]. Available: https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home

[14] "Ball Tracking with OpenCV," Web page, accessed April 2017. [Online]. Available: http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/

[15] "Range-Detector," Web page, accessed April 2017. [Online]. Available: https://github.com/jrosebr1/imutils/blob/master/bin/range-detector

Appendix A ATmega Ping Sensor Code

based on code from [13]

```
#include <NewPing.h>
#define SONAR_NUM 2      // Number of sensors.
#define MAX_DISTANCE 100// Maximum distance (in cm) to ping.

NewPing sonar[SONAR_NUM] = {   // Sensor object array.
  NewPing(9, 8, MAX_DISTANCE), // Each sensor$\textquotesingle$s trigger pin, echo pin,
  and max distance to ping.
  NewPing(11, 10, MAX_DISTANCE),
};

float storedValue[2];
int closer;

void setup() {
  Serial.begin(9600);
}

void loop() {
  for (uint8_t i = 0; i < SONAR_NUM; i++)  { // Loop through each sensor and display results.
    Serial.print(i);
    Serial.print(" = ");
    delayMicroseconds(1);
    if (i == 1) {
      if ((sonar[i].ping() == 0)) {
        storedValue[i] = sonar[i].ping()*0.034/2;
      }
      else {
        storedValue[i] = (sonar[i].ping()+1)*0.034/2;
      }
      //Serial.print(storedValue[i]/12/2.54);
      //Serial.println();
    }
    else {
      storedValue[i] = sonar[i].ping()*0.034/2;
      //Serial.print(storedValue[i]/12/2.54);
      //Serial.println();
    }

  for (uint8_t i = 0; i < SONAR_NUM; i++) {
    if ((storedValue[i] > 49) && (storedValue[i] < 58) )
    //1 inches old val. 53.3, 58.4,oldval 50, previous 55
    //if ((storedValue[i] > 66 && storedValue[i] < 71) ) //1.5 inches
    //if ((storedValue[i] > 78 && storedValue[i] < 81) ) //2 inches
    //if ((storedValue[i] > 94 && storedValue[i] < 99) ) //3inches {
      Serial.print(i);
      Serial.print("=");
      Serial.print(storedValue[i]/12/2.54);
      Serial.print("  ");
      Serial.println();
    }
  }
}
delay(17);
}
```

Appendix B ATmega LM555 Timer Code

```
int control555 = 11;



void setup() {
  // put your setup code here, to run once:
  pinMode(control555, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
//  noInterrupts();
  //digitalWrite(control555, HIGH);
  digitalWrite(control555, LOW);   // Put reset high so IC555 transmits
  delayMicroseconds(100);                        // do this for 100 microseconds
  digitalWrite(control555, HIGH);    // Put reset to 0 so 555 turns off
  delayMicroseconds(100);      //17.2 milliseconds




}
```

Appendix C Raspberry Pi Blob Detection Code

based on code from [14]

```
# import the necessary packages
from collections import deque
import numpy as np
import argparse
import imutils
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera
import time

# initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (320,240)
image= PiRGBArray(camera, size=(320,240))
camera.framerate=40
# print frame[1,1,1]

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--picamera",
help="Raspberry Pi camera ")
ap.add_argument("-b", "--buffer", type=int, default=64,
help="max buffer size")
args = vars(ap.parse_args())


# allow the camera to warmup
time.sleep(0.1)

# define the lower and upper boundaries of the "green"
# ball in the HSV color space, then initialize the
# list of tracked points
# greenLower = (29, 86, 6)
# greenUpper = (64, 255, 255)
# greenLower = (0, 0, 174)
# greenUpper = (255, 255, 255)
```

```
greenLower = (0, 28, 225)
greenUpper = (85, 255, 255)
pts = deque(maxlen=args["buffer"])
ctr=0

# keep looping
while True:
# grab the current frame
#(grabbed, frame) = camera.read()
camera.capture(image, format=$\textquotesingle$bgr$\textquotesingle$, use_video_port=True)
frame=image.array
        # cv2.imshow("Frame",frame)
        # cv2.waitKey(0)
# if we are viewing a video and we did not grab a frame,
# then we have reached the end of the video
#if args.get("PiCamera") and not grabbed:
#break

# resize the frame, blur it, and convert it to the HSV
# color space
#frame.truncate(600)
frame = imutils.resize(frame, width=300)
blurred = cv2.GaussianBlur(frame, (11, 11), 0)
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

# construct a mask for the color "green", then perform
# a series of dilations and erosions to remove any small
# blobs left in the mask
mask = cv2.inRange(hsv, greenLower, greenUpper)
mask = cv2.erode(mask, None, iterations=2)
mask = cv2.dilate(mask, None, iterations=2)

# find contours in the mask and initialize the current
# (x, y) center of the ball
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]
center = None

# only proceed if at least one contour was found
```

```
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)
#print cnts
((x, y), radius) = cv2.minEnclosingCircle(c)
#print "radius = ", radius
M = cv2.moments(c)
center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

# only proceed if the radius meets a minimum size
if radius > 5:
# draw the circle and centroid on the frame,
# then update the list of tracked points
cv2.circle(frame, (int(x), int(y)), int(radius),
(0, 255, 255), 2)
cv2.circle(frame, center, 5, (0, 0, 255), -1)

# update the points queue
pts.appendleft(center)



# loop over the set of tracked points
for i in xrange(1, len(pts)):
# if either of the tracked points are None, ignore
# them
if pts[i - 1] is None or pts[i] is None:
continue

# otherwise, compute the thickness of the line and
# draw the connecting lines
thickness = int(np.sqrt(args["buffer"] / float(i + 1)) * 2.5)
cv2.line(frame, pts[i - 1], pts[i], (0, 0, 255), thickness)

# show the frame to our screen
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
ctr = ctr+1
```

```
# if the $\textquotesingle$q$\textquotesingle$ key is pressed, stop the loop
if key == ord("q"):
                print ctr
break

        image.truncate(0)
# cleanup the camera and close any open windows
camera.close()
cv2.destroyAllWindows()
```

Appendix D Raspberry Pi Range Detection

based on code from [15]

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# USAGE: You need to specify a filter and "only one" image source
#
# (python) range-detector --filter RGB --image /path/to/image.png
# or
# (python) range-detector --filter HSV --webcam

import cv2
import argparse
from operator import xor
from picamera import PiCamera
from picamera.array import PiRGBArray

def callback(value):
    pass



def setup_trackbars(range_filter):
    cv2.namedWindow("Trackbars", 0)

    for i in ["MIN", "MAX"]:
        v = 0 if i == "MIN" else 255

        for j in range_filter:
            cv2.createTrackbar("%s_%s" % (j, i), "Trackbars", v, 255, callback)



def get_arguments():
    ap = argparse.ArgumentParser()
    ap.add_argument($\textquotesingle$-f$\textquotesingle$, $\textquotesingle$--filter$\textquo
                    help=$\textquotesingle$Range filter. RGB or HSV$\textquotesingle$)
    ap.add_argument("-p", "--picamera", required=True,
                    help="Raspberry Pi camera ")
    args = vars(ap.parse_args())


def get_trackbar_values(range_filter):
    values = []

    for i in ["MIN", "MAX"]:
        for j in range_filter:
            v = cv2.getTrackbarPos("%s_%s" % (j, i), "Trackbars")
            values.append(v)

    return values


def main():
    args = get_arguments()

    range_filter = args[$\textquotesingle$filter$\textquotesingle$].upper()
    camera = PiCamera()
    image= PiRGBArray(camera, size=(640, 480))

    setup_trackbars(range_filter)

    while True:
        #if args[$\textquotesingle$webcam$\textquotesingle$]:
        if args[$\textquotesingle$picamera$\textquotesingle$]:
            #ret, image = camera.read()
            camera.capture(image, format=$\textquotesingle$bgr$\textquotesingle$)

            if not ret:
                break

            if range_filter == $\textquotesingle$RGB$\textquotesingle$:
                frame_to_thresh = image.copy()
            else:
                frame_to_thresh = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

        v1_min, v2_min, v3_min, v1_max, v2_max, v3_max = get_trackbar_values(range_filter)
        thresh = cv2.inRange(frame_to_thresh, (v1_min, v2_min,
        v3_min), (v1_max, v2_max, v3_max))

        cv2.imshow("Original", image)
        cv2.imshow("Thresh", thresh)

        if cv2.waitKey(1) & 0xFF is ord($\textquotesingle$q$\textquotesingle$):
            break


if __name__ == $\textquotesingle$__main__$\textquotesingle$:
    main()
```

Appendix E Requirement and Verification Table

Requirements	**Verification**	**Pts.**
Ultrasonic Sensors		
LM555 astable multivibrator circuit should emit a frequency of 40 kHz ±10%.	1. Connect microcontroller to LM555 to enable reset pin to high (reset is active low). This will generate a continuous square wave. 2. Connect output to oscilloscope. 3. Verify frequency is within 40 kHz ±10%.	2
Both sensors must be able to view the 3.5 feet of the middle line from the end of the table closest to the player.	1. Set ultrasonic sensors in place for design specifications. 2. Connect sensors to oscilloscope. 3. Serve ball along middle line approximately 1 ft from the net. 4. Verify sensors receive signal on oscilloscope. 5. Repeat step 4 for serves approximately 2 ft, 2.5 ft, 3 ft, 3.5 ft, 4ft, and 4.5 ft from net.	7
Ultrasonic receiving sensors must detect reflected wave at a time delay corresponding to the distance of the ball from the sensor.	1. Set ultrasonic sensors in place for design specifications. 2. Connect receiver sensor output to Arduino (for time measurement). 3. Drop ball a specified distance away from sensor. 4. Verify distance using microcontroller by measuring time delay in received pulse.	5
ATmega Microcontroller		
Must repeatedly send a low signal for 100 us and a high signal for 17.2 ms	1. Connect microcontroller output pin to oscilloscope. 2. Run reset signal code which should send a low signal for 100 us and a high signal for 17.2 ms. 3. Verify signal is low for 100 μs and high for 17.2 ms.	1

Requirements	**Verification**	**Pts.**
Ultrasonic Sensors		
Must be able to determine which receiver acquired the reflected signal first.	1. Start pulsing the LM555 timers using the ATmega microcontroller 2. Bounce ball close to sensor 1 3. Observe that ATmega interprets received 4 consecutive pulses in order to return that the ball is closer to sensor 1 4. Bounce ball close to sensor 2 5. Observe that ATmega interprets received 4 consecutive pulses in order to return that the ball is closer to sensor 2	7
Bluetooth Transmitter		
Must be able to correctly send information with 99% accuracy from sensor to a bluetooth receiving device 12 ft away.	1. Program bluetooth transmitter to send a loop of bits of data representing which sensor detected object first. 2. Pair bluetooth transmitter with a bluetooth receiver 12 ft away. 3. Verify correct information is sent.	3
Raspberry Pi		
Must be able to receive 640x480p images at 90 fps from the camera module.	1. Read input image from camera. 2. Store in local memory and increment a frame counter 3. Output to computer 4. Repeat for 1 second 5. Observe continuous images on computer 5. Verify that frame counter reaches 90.	3
Must be able to send a correct signal to the display to cause LED to light green for "in" and red for "out".	1. Hard code a signal to transmit to the display. 2. Observe if the correct LED is green for "in" and red for "out".	1
Must distinguish the table tennis ball from the non-white background and table.	1. Return the location of the bottom-most pixel of the ball 2. Return dimensions of the ball (total number of pixels) 3. Display the image on the screen. 4. Compare returned values with raw image	7

Requirements	**Verification**	**Pts.**
Must distinguish the level of the top of the table tennis table with 90% accuracy.	1. Draw a line at the level of the top of the table tennis table on top of the image. 2. Display the image to the screen. 3. Compare returned values with raw image	5
	Output Display	
The display must clearly show if ball is "in" or "out" for the play.	1. Hard-wire input from the ARM microcontroller. 2. Check if LED displays green for "in" and red for "out".	1
The display must be reset after each play.	1. Trigger LED. 2. Press button to manually reset LED. 3. Check if LED is off.	2
	Battery	
The 9V battery must be able to feed the buck converter a minimum of 5V for at least 45 minutes.	1. Connect battery to buck converter and multimeter. 2. Measure voltage every 5 minutes. 3. Verify the voltage never falls below 5V.	3
	Buck Converter	
Buck converter must output a voltage of 5V with a ±5% voltage ripple and a minimum current of 70mA	1. Connect resistive load to buck converter. 2. Supply converter with 9V. 3. Connect output to multimeter. 4. Verify voltage is 5V with a ±5% and current is a minimum of 70mA.	3
	Total Points	**50**