2016 RFC Cambridge Team Description Paper

Eric Anschuetz, Kate Donahue, Rico Stormer, Ezra Zigmond[*]
RFC Cambridge
(Dated: January 23, 2016)

This paper illustrates advances the RFC Cambridge team made in the 2015-2016 year and describes the changes all robots will undergo for the 2015 competition. Improvements were made across the board, improving our robots' reliability.

I. MECHANICAL ENGINEERING

This year the mechanical team focused on several projects that were discontinued in the past, including:

1. The dribbler.
2. The kicking mechanism.
3. The shield.

We chose to work on the dribbler shape in hopes of creating a dribbler that allowed our robots to better handle and control the ball. The kicking mechanism was elected because the solenoid was heavily modified, making it difficult to replace when necessary, and the current retraction system involving rubber bands was not consistent. Finally, the shields were chosen as a project because the current models where created from two separate pieces of plastic held together by an epoxy which frequently broke.

Dribbler

The team is currently still using the design for the dribbler mechanism that was created in 2013 and is referenced in the 2013 TDP. In 2013 the team also tested a couple of shapes for the dribbler including both convex and concave shapes. They found initially that the concave shape gave better control over the ball. We have not yet confirmed this to be true but are in the process of testing these shapes in addition to a sine wave. This shape was brought up in the 2014 TDP but no results were found on the effectiveness it. In the 2014 TDP the team also made note of the

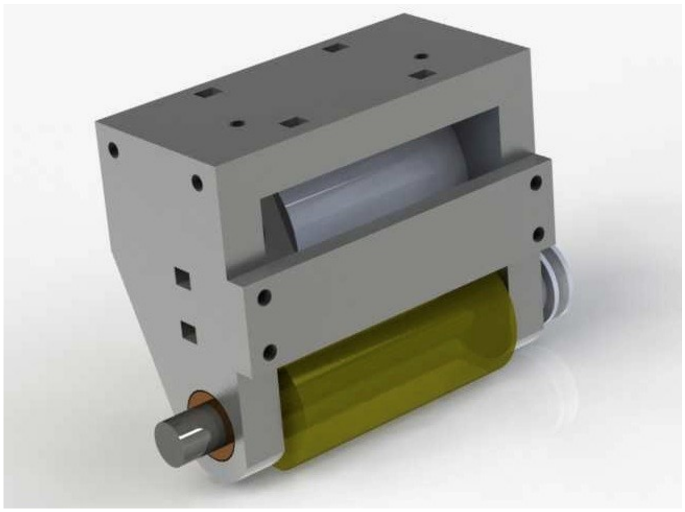
FIG. 1. A CAD rendering of the current dribbler design.

[*] robocup-exec@mit.edu

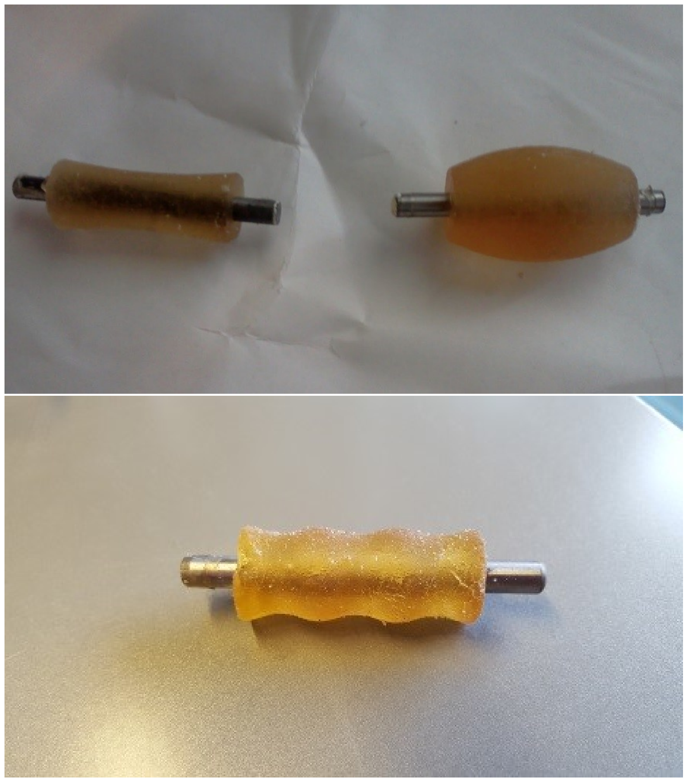

FIG. 2. The convex, concave, and sine wave finished casted parts. Notice the rough edges on the dribblers.

fact that the material they were first using was not durable enough and thus switched from urethane rubber to a low durometer (Shore OO-30) silicone rubber. We have also noticed that the concave and convex dribblers made from urethane rubber were cast in a 3D printed mold. These molds have ridges on them and in turn create an uneven surface on the dribbler. This uneven surface catches and tears resulting in flaking and a loss of durability for the dribbler. In order to solve this issue and improve the dribbler mechanism we have come up with the following plan:

1. The desired dribbler shape will first be 3D printed in order to create the complex shape. Then we will thermoform plastic around the 3D printed part creating two halves. Once the two halves are combined a full mold is complete and the ridges from 3D printing will have been removed.

2. The next step is to then test the effectiveness of each shape. We created a standalone dribbling mechanism for this so that we can quickly test each dribbler shape without having to work with an entire robot.

3. Once we have determined the best shape we will scale up the production of the dribbler and include that dribbler on our fleet of robots.

Kicking Mechanism

One of the major projects the team worked on this year was investigating the current design of the kicker. There were two major questions we investigated:

1. Should we add a chip kicker?

2. Should we modify our existing regular kicker or keep the same design?

For both problems we looked at research from existing team description papers. Of particular help were the 2015 team description papers from the Robobulls, the RoboJackets, and the RoboDragons. In many cases they described building their own solenoid. However, we determined that for continuity and repeatability of the kickers after existing Team members had graduated made more sense to buy a solenoid. In this case so we came across another problem

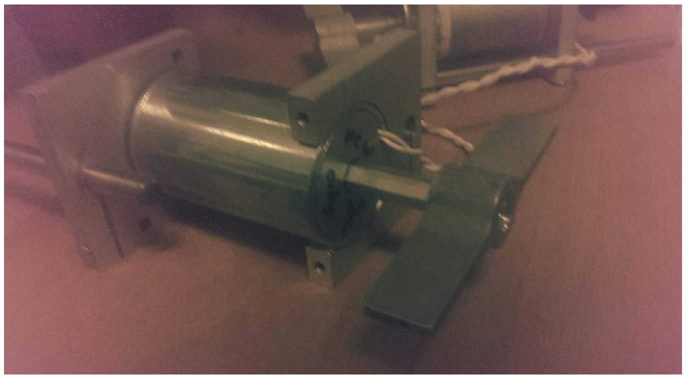

FIG. 3. The current kicker design with a hex plunger.

because the existing front design of the robot does not allow much space. This is especially problematic in the case of the chip kicker. In the end we referred to the rules of the game which currently does not allow a goal to be scored on a chip kicker and our experience at different competitions which is indicated that she figures are not usually a main determining factor in winning games. Because of this we decided to forgo adding a kicker. However, this still left the problem of modifying our existing design of the regular kicker. Our existing kicker design was created before any of the team members of the current team joined, around 2011. However we are extremely fortunate that they decided to leave us with detailed data on different solenoids they tested and in particular how far the inner rod needed to be extended in order to have maximal kicking ability. This Excel spreadsheet we would be glad to share with interested teams. It allowed us to decide to save the same model roughly. This was also due to mechanical concerns. The remainder of the existing robot was designed and built around a particular shape and configuration of that solenoid. However there are additional problems, namely that the kicker design had been heavily modified and customized for each kicker. This is not only labor intensive but also meant that kicker is frequently varied in their properties. The main changes this design made was to identify and buy store-bought materials that replaced hand built ones. One of the key difficulties was converting the solenoid from pull to push. Below we have included specific steps we took in order to accomplish this:

1. Purchase all necessary materials. The solenoid is from electromechanics online, part number SOTUH032051. Springs are also available from the same company. In addition we bought threaded male to female hex standoff from McMaster. You should also either purchase or design a kicker face to go in the end of the shaft. This should be approximately the same mass as the golf ball so as to achieve the maximally efficient transfer of energy. Our preferred method was to design one using the water jet we had access to and then drill out the remaining parts ourselves using a press fit keep it on top of the shaft.

2. Disassemble the solenoid. Modify the shaft so that you can screw the hex standoff on to the end. Although the shaft is typically made of a very hard material in practice this did not turn out to be as difficult as you would anticipate. Modify one end of the solenoid so that the shaft can fit through it. This will entail making the circular opening at one end larger to accommodate the shaft.

3. Assemble the solenoid along with the spring you have chosen. You could design alternate pullback mechanism perhaps involving rubber bands. However our experience with this has shown us that springs are the most compact and reliable method and are fairly cheap so they can be replaced if broken.

Shield

This is a continuation of the project from 2014 and was described in that TDP. The basic issue is that the current shield design is made of two parts. There is a top plate that is attached to a heat bent sheet of plastic using epoxy. This epoxy can be quite brittle and breaks when struck by an object such as another robot during a soccer match. The quick fix solution in the past was to just add more epoxy. We have come to the point where this is no longer a reasonable solution. In 2014 a project began to make the shield out of one solid piece of plastic thus eliminating the need for epoxy at all. In order to accomplish this, the team decided to use thermoforming. A reusable foam core model was created so that multiple molds could be made.

It was noted that the plastic used to create the shield when thermoforming was thinner than the previous plastic used and its durability was questioned. We have not yet had the opportunity to test the durability yet but another design was proposed. It involved reusing the two part shield system. Instead of using epoxy the two parts would instead be attached by four L brackets.

It is still not fully decided which design will be better but further testing will tell us which is more durable and which requires more time to create on a large scale which will in turn help us with our design decision.

II. ELECTRICAL ENGINEERING

We have been continuing our work from last year on redesigning our electronics system. Our goals have been twofold:

1. Create a new revision of our circuit boards to improve modularity and reliability.

2. Improve robustness firmware for the electronics.

Board Revision

In order to address the first point, we first rethought our board architecture. Last year, we had three boards: a communications board which interacted with a computer with XBEE and acted as the SPI master, a motor board which received motor commands from the communications board and drove all four motors, and a kicker board that handled charging and discharging capacitors for kicking as well as delivering power to the other boards. One of main problems with our previous revision was that the design of the motor board was such that if there was a problem with one of the four motor driving circuits, the entire board would need to be replaced.

To address this concern, one of the main points in redesigning the architecture was to increase the modularity of the motor driving circuits in particular. Our current design has more but smaller boards: A motherboard that handles XBEE communication with the computer and powers the other boards, a microcontroller board to make it easy to replace just the microcontroller, a kicker board that controllers charging the capacitors on the motherboard and discharging for kicking, and four motorboards each of which has one microcontroller and has the driving circuit for one of the four motors. There are several advantages of this new architecture over our previous system. Most importantly, motor modules and microcontroller are relatively easy to swap out, while we are able to combine most of what used to be on the communication and kicker board into one mother board. Now, there is only one relatively large board, with the remaining six smaller modular boards being much smaller. The communication protocol between the SPI master and the motorboard slaves is also simplified because each motorboard now only drives one motor.

Firmware Improvments

One of our major concerns about the firmware going into this year was getting accurate input to the motorboard from our quadrature encoders. Previously, we handled this by attaching the output pins of the quadrature encoders to external interrupts. Then, in the firmware, whenever one of the pin change interrupts (PCI) fired, we would check to see which pins had changed since the last PCI and then update the direction of the wheel motion. The problem we ran into was that if the robots were moving at a moderate to fast speed, the PCI interrupts would happen too fast and some of the interrupts would be dropped. This would sometimes result in the firmware being unable to distinguish a robot moving very slow and very fast. Due to the problems with the PCI approach, we decided to change the microcontroller model to one that has built in hardware quadrature encoder interrupt (QEI) handling. We decided to switch all of our microcontrollers from the ATMEGA165 to the ATXMEGA16A4 (in theory, we could have kept the motherboard using the ATMEGA165, but this would have necessitated maintaining two separate codebases). The primary challenge we have faced switching microcontrollers has been adapting our lowest level firmware to use the appropriate registers for the ATXMEGA16A4. Fortunately, when we designed the firmware last year, most of the details dealing directly with the microcontroller APIs were encapsulated within a low level library. Thus, we were able to reuse most of the existing code and only needed to modify the low level details in one place.

The revisions to board architecture have also allowed us to simplify the protocol used to communicate between the motherboard (SPI master) and the SPI slaves. Firstly, we no longer have a microcontroller specifically for controlling the kicking circuitry, so the motherboard can send commands to the charging and discharging circuits directly. Secondly, now there is one microcontroller for each motor driving circuit. Before, when the single motorboard received a command from the communications board, the command had to specify which motor the command was for. Thus, the command we send no longer have to explicitly contain information about which motor the message is for.

III. COMPUTER SCIENCE

This past year our computer science team has focused on creating debugging tools, moving to more modern libraries and C$^\sharp$ coding practices, and writing effective AI to take advantage of our dribbling capabilities.

Debugging tools

One of the first debugging tools we added this year was a dynamic visualization of our written AI in action. By mapping where in the field our AI designates as “high value” territory that our robots should move to and “low value” territory that our robots should avoid, we can fine-tune our strategy such that the AI sends robots to the most strategically viable positions. Though we could statically view what parts of the field previous versions of the code deemed “high value,” we now can watch values across the field develop as plays develop and fine-tune not only spatially where our robots should be sent but also where they should be sent to act on some possible future field state. We plan on using this latter feature to help our team write and debug a strategy that sends robots to ideal locations based on predictions of where enemy robots will be, not necessarily where they are at the time of field evaluation.

A little more on the maintenance side, our team has cleaned up much of our legacy code. Not only have we written many lines of code to take advantage of the latest C$^\sharp$ and .NET library features for ease of readability and modularity, but we have also completely reworked our field visualization system to use Windows Forms, future-proofing it as the older graphical libraries we have used become deprecated. During this rewrite of our field visualization system we also added information useful when tracking what play our team is currently running, allowing us to ensure we are running the correct play based on signals from the Referee Box.

Dribbler based AI

Our plays have been modified to take advantage of our dribbler. While our ball carrier is undefended, it pushes towards the goal to keep up the assertive offense we have been trying to keep in place since last season. Furthermore, we have added ways to juke around defenders using the dribbler, allowing us to get a quick shot in undefended or bounce shot the ball off of a teammate. Though these lanes are in general only open until the defender reacts to our movements, it allows us to aggressively move the ball closer to the opposing goal and get around defenders, something we have had trouble with in past practice and competition.

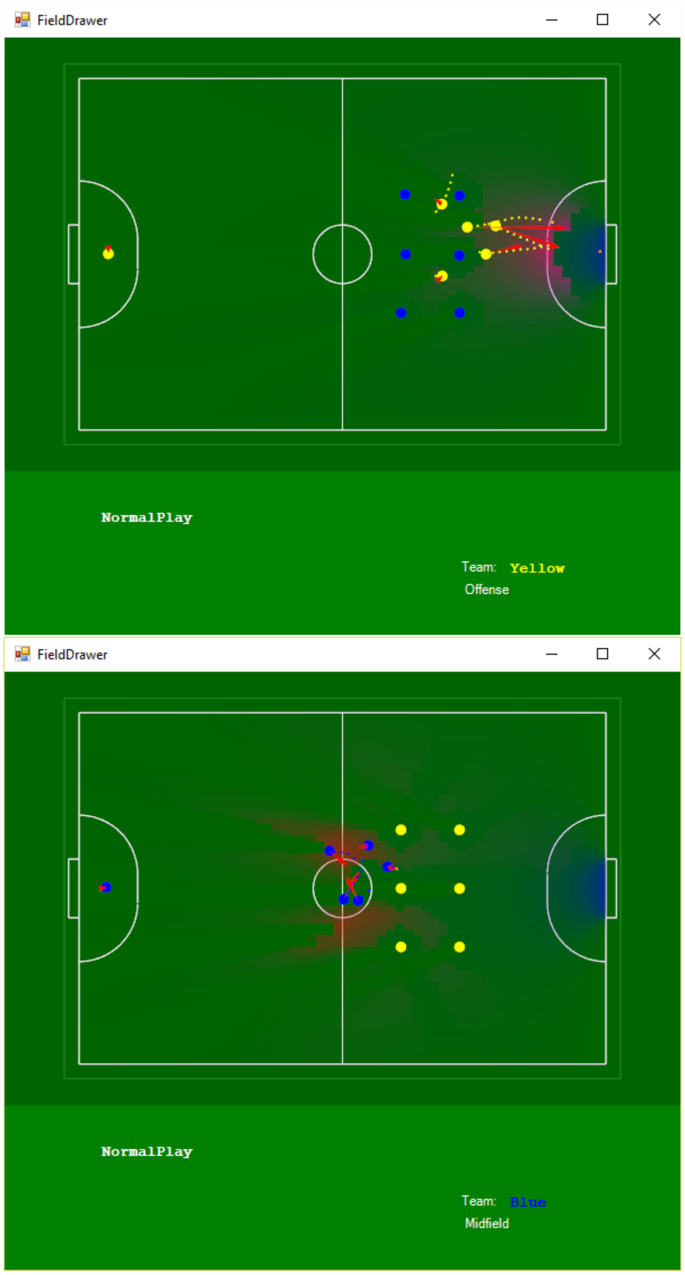

FIG. 4. A demonstration of our new field drawing capabilities.