2016 RFC Cambridge Team Description Paper

Eric Anschuetz[1], Jonathan Cruz[1], Kate Donahue[1], Sandra Schumann[1], Rico Stormer[1], and Ezra Zigmond[1]

[1] Harvard University
[2] Massachusetts Institute of Technology
robocup-exec@mit.edu

Abstract. This paper illustrates advances the RFC Cambridge team made in the 2015-2016 year and describes the changes all robots have and will undergo for the 2016 competition. Improvements were made across the board: mechanically, we are improving our robots' reliability and dribbling capabilities; electronically, we are finalizing our new circuit board design; and programmatically, we are shifting to future-proof code and looking towards future projects in implementing dribbling algorithms.

1 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 [6]. In 2013, we also tested a few cross sections for the dribbler including both concave and convex cross sections. Our preliminary findings indicate that the concave shape gives 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 design was first mentioned in [2] but no results were found on the effectiveness of it. In [2] we also made

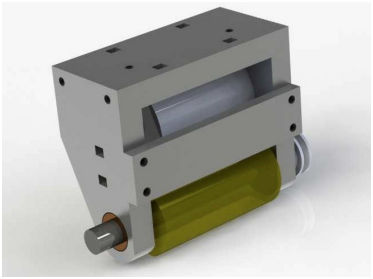

Fig. 1. A CAD rendering of the current dribbler design.

note of the fact that the material first used was not durable enough and thus we 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. We will then test the effectiveness of each shape. We have 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.

We have found through testing that the best two shapes are straight and concave cross sections. Further testing is needed and will be concluded soon in order to determine which of the two current designs is better.

Kicking Mechanism

One of the major projects we 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 [1], [4], and [7]. In many cases they described building

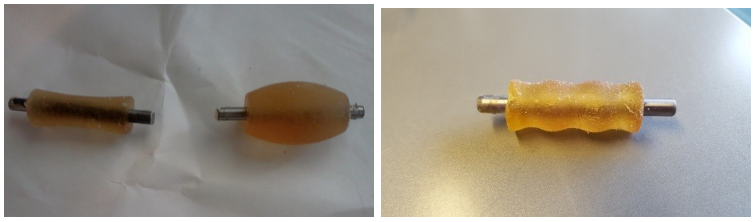

Fig. 2. The concave, convex, and sine wave finished casted parts, respectively. Notice the rough edges on the dribblers.

their own solenoid; however, we determined that for continuity and repeatability of the kickers after existing team members had graduated that it made more sense to buy solenoids. Further testing then revealed that, due to the existing front design of our robots, are our robots are tight on space for a chip kicking mechanism. In the end, we referred to the rules of the game which currently do not allow a goal to be scored on a chip kicker, and our experience at different competitions which indicated that chip kickers are not usually a main determining factor in winning games. Because of this we decided to forgo adding a chip kicker.

We now turned to modifying our existing design of our standard kicking mechanism. Our existing kicker design was created before any of the team members of the current team joined, around 2011, and have issues with consistency; through the years, our kickers had been heavily modified and customized for each robot. This not only leads to the usual inconveniences of maintain multiple designs but also means that each kicker behaves differently and is therefore difficult to program uniformly. Therefore, our number one priority of our new kicker design was to primarily rely on standard store-bought materials to replace our hand built ones. We are extremely fortunate that those that originally designed our kicking mechanism left 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. Using these data, we were able to focus on a kicker design that integrated well with our current design as our existing robots were designed and built around a particular shape and configuration of that solenoid.

To improve our kicking consistency, we also chose to move from a "pull" mechanism to a "push" kicking mechanism. Below we have included the 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

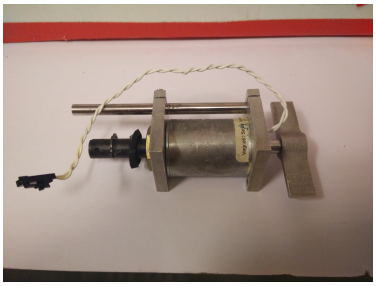

Fig. 3. The current kicker design with a hex plunger.

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 one 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 one 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 one have chosen. One 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.

We are still in the process of choosing our desired spring. Though we have not yet tested the numerical speed of our new kicker design, it qualitatively delivers the same power as our previous kicking mechanism with much improved consistency.

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, we 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.

2 Electrical Engineering

Electrical Engineering

We have been continuing our work from last year regarding the redesign of our electronics system. Our goals have been three-fold:

1. Create a new revision of our circuit boards to improve modularity and reliability
2. Update our capacitor discharge circuitry to improve safety and ease of use
3. Improve firmware robustness 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 using an XBEE - acting as the SPI master - a motor board which received commands from the communications board to drive all four motors, and a kicker board that handled the charging and discharging of capacitors for kicking, as well as delivering power to the other boards. One of main problems with our previous revision was an overall lack of modularity present in the motor board design: if any of the four motor drivers on the motor board ran into problems, then the entire motor board would need to be replaced - an expensive and time consuming endeavor.

One of the main desires with our redesign of the board architecture was an improvement in the modularity of the motor boards. Our current design has more but smaller boards: A mainboard that handles power and signal distribution as well as capacitor charging and discharging, a communications board that handles XBEE communication with the main computer and distribution of signals to the mainboard, and four motorboards each with one microcontroller and a motor driver for one of the four motors. There are several advantages of this new architecture over our previous system. Most importantly, motor modules and microcontrollers are relatively easy to swap out. Now, there is only one relatively large board, with the remaining five smaller modular boards being much smaller, located on above the mainboard. The communication protocol between the SPI master and the motorboard slaves is also simplified because each motorboard now only drives one motor rather than four.

Along with our improvements to the overall board architecture, we also decided to shift to different types of board to board connectors as well as a new

motor controller part. In order to save board space and improve reliabiliy, we switched from the through-hole connectors used in our previous revision to a shrouded SMT connector. The change from through hole to SMT connectors saves space on each of the boards by only taking up space on a single side of the board. The shrouded SMT connectors, in particular, also increase board reliability by making it much more difficult to accidentally plug in connectors incorrectly - a common problem with the previous board achitecture. Along with our upgrades to the overall board architecture, we decided to upgrade our old motor controller part to a better part with increased functionality and reliability (DRV8307) - one of the main improvements being integrated thermal shutdown.

Updates to Capacitor Discharge Circuitry

A major concern with our previous capacitor discharge circuitry was that it could only discharge the capacitors while power was applied to the system - if power was lost, the capacitors would remain fully charged at dangerously high voltage levels of 250V. This lead to several unsafe situations, often necessitating the use of a metallic object to short the capacitor leads, discharging the capacitors back down to safe voltages.

We explored numerous potential solutions to this design issue, eventually settling on two distinct designs that would allow the capacitors to be charged and discharged with microcontroller commands as well as discharging the capacitors with no power applied to the system. The first design focuses on using the capacitors own voltage when charged to drive a discharge circuit, even without any external power sources. The second design focuses on taking advantage of the unique relationship between gate-source voltage and drain current of depletion-mode MOSFETs, namely the fact that depletion-mode MOSFETs conduct with an applied gate-source voltage of 0. Each design uses an opto-coupler to isolate the microcontroller from the high voltage, analog signals liable to be present in the discharge circuitry.

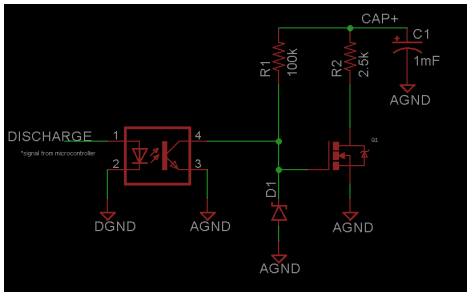

Fig. 4. Capacitor discharge circuitry design number 1.

The first design (see figure 4) allows the microcontroller to drive the gate of the n channel MOSFET, and actively control discharging of the capacitors. This design also allows discharge with no external power by using current across R1 to bias the MOSFET's gate, using the zener diode as a voltage reference - in essence, allowing discharge through R2 as long as the voltage across the capacitors is large enough to drive the zener diode. We ultimately did not pursue this design, due in large part to concerns about constant power dissipation through this circuitry when the capacitor is being charged (although the power loss is small) and inability to completely discharge the capacitor (it can only easily discharge until the transistor's gate-source voltage threshold).

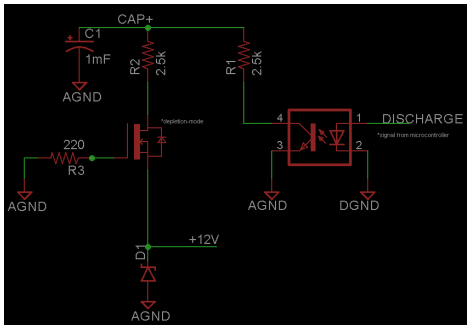

Fig. 5. Capacitor discharge circuitry design number 2.

The second design (see figure 5) has a branch that allows the microcontroller to actively control capacitor discharge through the BJT present in the opto-coupler, while also containing a passive discharge branch that will discharge the capacitors when power is no longer applied. When external power is applied, the 12V power supply present on board keeps the gate-source voltage at -12V, preventing the depletion-mode n channel MOSFET from conducting. If external power is lost, the gate-source voltage will drop to about 0V, allowing the transistor to conduct and discharge the capacitors. A zener diode is added to clamp any fluctations that may occur on the power line to safe levels - while likely unecessary, we added this feature out of an abundance of caution. We ultimately decided to pursue this design due to its ability to completely discharge capacitors without external power as well as its lack of constant discharging.

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 mainboard 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 API's 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 communications board can send commands to the charging and discharging circuits directly. Secondly, there is now a single microcontroller for each motor driver circuit. With our previous revision, 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.

3 Computer Science

Due to the significant amount of work that needed to be accomplished on the electrical engineering subteam this year, most of our computer science manpower has been focused on aiding the electrical engineering team on their tasks; nevertheless, our computer science team has been able to perform small maintenance tasks such as creating new debugging tools and moving to modern libraries, and has explored avenues for future projects, particularly on some dribbling algorithms.

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 us 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; this feature also means in the future it will be straightforward to add not only a direct match recorder to our field drawing system but also a match recorder that allows us to determine how our code evaluated the field throughout the match to help debug inconsistencies of our AI during the match.

A little more on the maintenance side, we have cleaned up much of our legacy code. Not only have we written many lines of code to take advantage of the latest C♯ 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 we are currently running, allowing us to ensure we are running the correct play based on signals from the Referee Box.

Fig. 6. A demonstration of our new heat map drawing capabilities. Highest priority locations are mapped in purple.

Dribbler based AI and the future

Looking towards the future, we have investigated ways to utilize our dribbler more effectively in our plays. Our current naïve heuristic is that, while our ball carrier is undefended, it pushes towards the goal to keep up our assertive offense from [3]—in the future, we intend to implement methods for effective ball

handling as in [5]. Furthermore, we are looking next year towards ways to juke around defenders using the dribbler, allowing us to get a quick shot in undefended or a bounce shot 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.

References

1. Adachi, Y., Kusakabe, H., Yamanaka, Y., Ito, M., Murakami, K., Naruse, T.: RoboDragons 2015 Extended Team Description. (2015)
2. Adkins, C., Anschuetz, E., Donahue, K., Gaddy, D., Khan, O., Liu, L., Logan, B., Park, E., Ramirez, A., Robo, E., Schanne, E., Schluntz, E., Yau, M.: 2014 Team Description Paper RFC Cambridge. (2014)
3. Anschuetz, E., Donahue, K., Gaddy, D., Park, E., Schluntz, E., Schumann, S.: 2015 Team Description Paper. (2015)
4. Barulic, M., Buchanan, J., Iachonkov, B., Jones, E., Jones, J., Mansour, A., Osawa, R., Strat, R.: RoboJackets 2015 Team Description Paper. (2015)
5. Biswas, J., Cooksey, P., Klee, S., Mendoza, J.P., Wang, R., Zhu, D., Veloso, M.: CMDragons 2015 Extended Team Description. (2015)
6. Donahue, K., Gaddy, D., Mattinson, B., Ramirez, A., Jin, R., Liu, L., Kanev, S., Park, E., Logan, B., Schanne, E.: 2013 RoboCup Team Qualification Paper: RFC Cambridge. (2013)
7. Shamsi, M., Waugh, J., Williams, F., Ross, A., Llofriu, M., Weitzenfeld, A.: RoboBulls 2015: RoboCup Small Size League. (2015)