Final Report

Mentor: Peter Henderson
Members: Craig Peden, Akela Drissner, Basile Henry, Kostadin Kalinov, Michael Lotkowski, Eugen Alpeza, Andrew Crawford

Introduction

Our goal for the course was to produce a reliable and robust robot with accompanying control software for football matches with the eventual goal of winning the final tournament. While it did not win, our robot performed exceptionally and achieved everything we could have wished for in a strong defender.

This report will outline the various aspects that went into the creation of our robot over the duration of the course and some of the challenges we had to overcome. It will outline our reasoning and contextualise our solutions to the problems we faced in achieving our goals.

Management/Organisation

We started our first meeting with a discussion about how we were going to manage our team. It was agreed upon that a remote repository and a Facebook group for discussion were essential. While most groups had a team structure with a team leader, we decided to go with a flat organization. Our team met almost on a daily basis to work on resolving various issues together.

As we progressed further into the project, we introduced pair programming. This allowed everyone to have a better overview of what individual people were working on and so allowed us to find the best possible solutions to issues we encountered. We soon realized our decision regarding organization was working very well for us.

Looking back after the final tournament, we could have chosen the more traditional team structure with an official group manager as our structure may have been difficult for other students to handle. However for us in particular we do not believe this would have improved our performance by much.

Hardware

The overarching goal of the hardware side of our robot was to create a simple design which could reliably maneuver around the pitch, defend the goal, and pass the

ball to our attacker. We are extremely proud of the outcome, but it was the result of a lot of failures, learning and interacting throughout the entire length of the course.
Our group used an Arduino Uno as the brains of the project connected to three lego NXT Servo motors, one 9V lego technic, and a custom built Infrared sensor. We used a very basic two-wheel drive chassis with two ball bearings in the rear for balancing.

Originally, we started with a mechanically complex design based on 2011's Group 9 which utilised two wheels - each with two motors - that could both individually turn 360 degrees and provide forward/backward drive for the robot. However, these wheels were overly complex and extremely different to program as well as heavily restricted the cannon-esque kicker design we had in mind. The space issues this created eventually resulted in a clean slate being necessary for the design and a move towards a more conventional chassis.

The design we then agreed upon was a simple two wheel rectangular chassis that was very similar to many robots both in this year and previous years. The movement worked very well but the same problem of making a functional kicker that fit in the box was a recurring trend that took many iterations and a lot of thought. After attempting our planned cannon-esque kicker on the new design, it quickly became apparent we would again have space issues. However, once we were told our role was a defender and thus we did not need too powerful a kicker, we were able to use a traditional foot-like kicker that solved all of our problems.

The catcher also went through a few iterations. After first trying a horizontal catcher that reached out with arms spread we then switched to a vertical grabber due again to space issues. Originally we wanted both the catcher and the grabber on a single motor but after that became practically difficult with the layout of the front of the robot we switched to a dual motor design. This allowed greater flexibility in our strategies and allowed us to independently control the kicker and the grabber. To make the grabber even better we added an infrared sensor which was able to detect if the ball was within grabbing range without having to rely on the vision system. The result of this iteration and innovation on the sensor was a robot which was able to grab and attempt to pass the ball 100% of the time during the final tournament - a feat we believe unmatched.

We used ball bearings on the back of the robot to balance the robot and through good design we never had any problems with weight distribution, traction or movement due to the mostly symmetrical design.

The main problem we had to overcome throughout was fitting in the box, which drove the design and ideology behind the rebuild and the new kicker. We began with designing around our wheel system which was our only major mistake however we managed to change it and produce a robot which performed admirably in the final tournament.

Communications

Our communication system needed to be simple, fast and robust. The communication was handled wirelessly using the xino RF arduino board. We chose to send low level commands to the robot to have good control of every aspect of the movements and actions executed. The communication protocol was chosen with the idea that the computer will be doing most of the communication in an imperative manner at a high frequency.

Each command represents a change in the state of the robot FSM and each is represented by a single byte. This makes sending and receiving extremely fast, thus allowing us to have real time communication with the robot. The command byte is constructed as illustrated in figure 1 in the appendix. When a command is received on the robot all the necessary checks are done (signature and checksum) before the command is evaluated and executed. We included the signature bit to reduce the risk of processing other team's messages.

The movement of the robot is done through the control of two motors - one for the left and one for the right wheel. The movement commands controlling the wheel speeds needed to have as much precision as possible to ensure we had the best possible accuracy. We noticed that the motors could not be used with under 30% power or the robot would not move. Thus we mapped non zero speeds in our communications module to between 30% and 100% power on the motor board. Reducing the range of values the module sent maximised the accuracy of the speeds we did cover, since our custom protocol limited us to sending only 7 distinct speeds - using three bits.

The kicker opcode was used with different values to request kicker/grabber actions to be executed. Once a command was received some non blocking timers were set up on the arduino in order to execute timed actions such as grabbing and kicking. Those actions are timed because we were using simple motors that needed to go one way for a given amount of time and then stop. The grabber command was only used to arm the grabber and the catching was done by the arduino itself when the infrared sensor detected the ball.

On the computer side, the communication sent commands in the main thread (the one used for vision and planning) and received in another thread. The receiving thread could then update the state of the robot.

Vision/World State

The vision system provided us with almost all of the contextual information required to build the world from which the robot's actions would be deduced. The vision and world system as with many other parts of the code was based upon Group 7's python code from 2014. The camera above the pitch provided us with frames which were processed into useful information using the OpenCV python library, which provided us with masking and contour detection allowing objects on the pitch to be identified.

The world model was based around zones on the pitch, which were obtained using a cropping tool developed by last year's Group 7. Knowing where the zones were on the pitch allowed us to evaluate which robot was which and to plan strategies.

Masking was done using hue-saturation-value channels, loaded from saved values and edited in the gui, allowing for real-time calibration of the system. For example the ball was found using a red hue, a high saturation, and values subject to lighting conditions. The centre of the contour created by this masking was then passed to the 'world' as a pixel location on the image. Using raw pixel locations had the advantage of simplicity since there was no need to convert values to real distances. The plates on top of the robots were used as a mask so that the offset of the dot on the plate could used to calculate the direction the robot was facing. We experimented with different methods of finding the dot, including gaussian filtering, since it was hard to differentiate from the plate especially in situations of uneven lighting as was seen on the second pitch.

Frame rate was also very important since the only feedback we got from the state of the pitch was through the camera. With the latency of the camera feed we needed to make an effort to ensure that we stayed as near as possible to the feed's frame rate in order to minimise our reaction times.

Another issue we encountered was that the vision system would only detect the centre of the ball to within accuracy of a few pixels, so it was necessary to sanity check the velocity of the ball using the past few frames to reduce extreme values of velocity and provide more reliable information for planning.

Strategy

We modelled our robot as a reactive agent, as our control flow consists of a hierarchy of task-accomplishing behaviours. The reason we chose this particular agent design was because in our particular environment, the robot was deterministic in regards to the set of environment variables that our software monitored. Furthermore, due to the limited number of random variables that we monitored and the relatively small number of condition statements that we used to express the

behaviour of the robot, our implementation exhibited all of the advantages and none of the disadvantages of the reactive agent design.

As we were the defensive robot of our team, we had three main goals to achieve: if the ball is in our zone we should catch the ball, if we have the ball we should pass to our teammate, and if the ball is not in our zone we should defend.

These three goals we modelled by using a single strategy for every one of them. We also had one additional strategy that was used to avoid getting stuck around the edges of the pitch when we were trying to catch the ball. When the situation on the pitch changed, we appropriately changed the strategy so that we were executing logic to match that.

Since we were using a reactive design with relatively few environment variables and condition statements for control of our robot, debugging the strategy and fine tuning it was relatively easy. That allowed us to achieve very intelligent and precise behaviour of our robot which in turn resulted in consistently performing any of our three goals. Furthermore, since we were able to send anywhere from 25 to 30 commands to the robot per second and the fact that we used relative measures referencing objects on the pitch, rather than absolute values of distance, our strategy component was able to correct for the inaccuracies in the movement of the robot.

In the end, we can say that our robot consistently achieved the goals it had with the strategy component that we implemented.

Conclusion

In conclusion, we functioned excellently as a unit which can be seen in our well balanced individual marks throughout the entire process.

Our group was very economical with our use of our budget, having spent less than a quarter of it. In a future revision, perhaps investing more time into a more complex and expensive design through the inclusion of holonomics could have resulted in a better robot.

We could also have worked on improving our vision system by developing a more flexible system which can handle inconsistent lighting conditions. We made an attempt at introducing gaussian filtering but this didn't make it into the completed build.

While we have these two areas we could have possibly improved upon, we believe they would not have made a game-winning difference to the final product and that bigger improvements could be found from improving the attacking aspects of our teammates.

We are proud to have produced a robot which we believe was a solid candidate for the best defender in the year.

Appendix

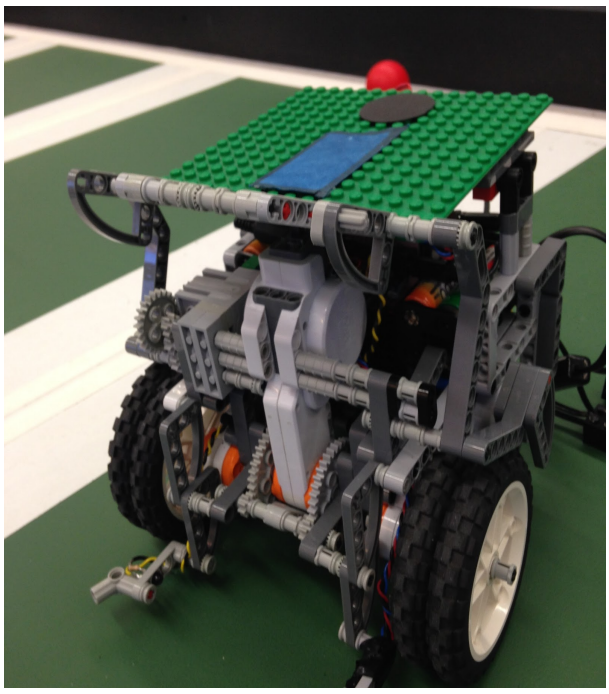
Figure 1 - Message Structure

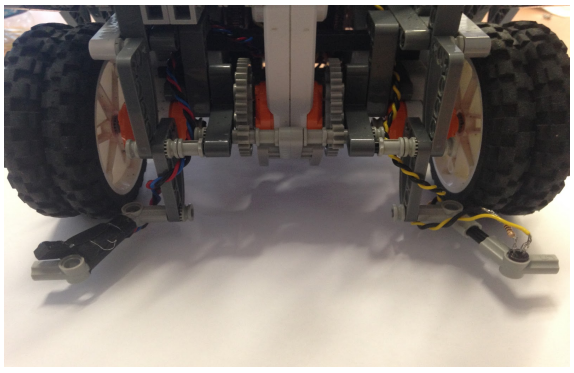
1 Bit	1 Bit	2 Bits	4 Bits
Signature	Checksum	Opcode	Argument

- 1 bit for signature set to 1 (this was to differentiate from simple ASCII characters used by most other teams)
- 1 bit for checksum (simple parity check)
- 2 bits for opcode (3 different actions: left and right wheels control and kicker control, 4th action was used as a test opcode)
- 4 bits for the argument (16 different possible values)

Resources

Our Robot, Botstian Schweinsteiger





Left: Picture of the main parts of the robot *Right:* Close shot of the infrared sensor

Git Repo
http://www.github.com/CraigDem/SDP