

The Soccer Guy



By
Marcus Hilgers
Faisal Ahmad
Jacob Borer

1.0 Abstract

The Soccer Guy automaton was created to provide a fun, interactive, simple game. It uses simple mechanical and computational elements to produce an entertaining interactive automaton. It primarily consists of a playing field with a goal and a goalie (See Figure 1). All the mechanical and computational elements reside underneath the playing field. The computational element is used to detect a shot on the goal and decide how the goalie should react. The mechanical element moves the goalie from side to side. The end result is a whimsical self-operating machine, an automaton.

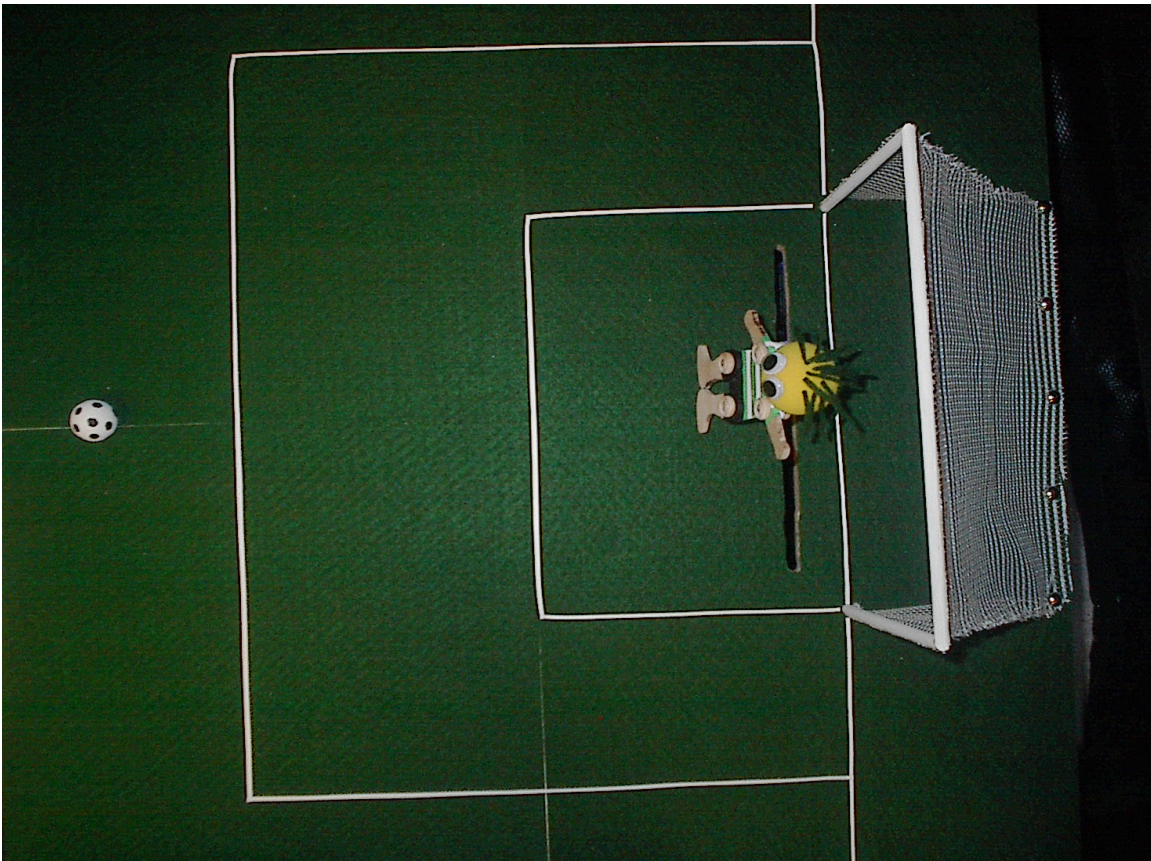


Figure 1: Overview of the Soccer Guy

2.0 Design

The three main components of our automata are the playing field, the drive mechanism for the goalie, and the computer based control. The playing field and control elements are basically the same as when we first sketched out our design. The design, prototyping, and construction phases were interlaced. The drive mechanism was revised several times. The use of prototypes allowed us to see and avoid potential pitfalls. By focusing on producing working components early we were able to build upon each success towards our final goal. There was very little work to be done integrating components because of this design while build approach.

Our original design was rather ambitious and included sensors for predicting the ball's motion, a mechanical foot to kick the ball, score sensing, ball tracking, and a crowd that would react to goals. (See **Error! Reference source not found.** and **Error! Reference source not found.**) Our initial proposal included a specific list of checkpoints. The first checkpoint consisted of creating a goalie and having him randomly jump to block the ball. The other five checkpoints built incrementally upon that to include all our proposed elements. After consulting with Ann and Mike Eisenberg we decided to limit the scope of the project and focus on the first checkpoint. This turned out to be a good decision.

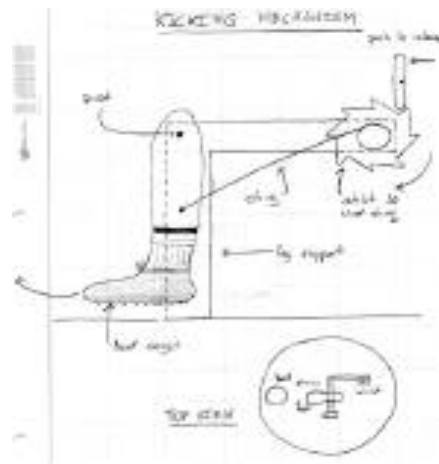


Figure 2: Proposed Kicking Mechanism

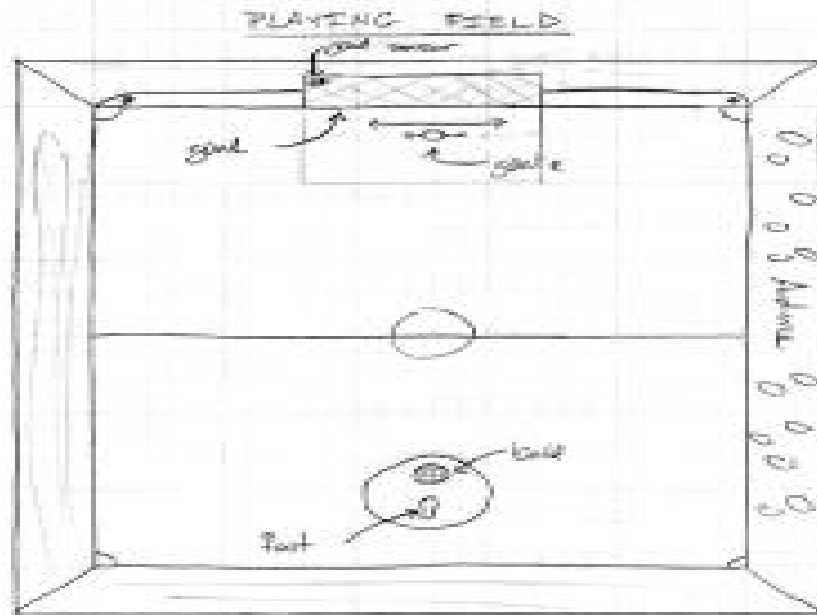


Figure 3: Proposed Field with All Elements

The creation of the playing field proceeded rapidly. We first decided on appropriate dimensions for the field and constructed a set of goal posts based on these dimensions. Our feeling was that this would help solidify the scale of the project in our minds. We then fabricated the playing field and a prototype goalie. We quickly learned that balsa wood was the wrong material for the goalie. Small and thin pieces of balsa wood often split during cutting on the scroll-saw. The prototype balsa wood goalie promptly had its legs destroyed when it was tested trying to stop a finger flicked ping pong ball. We decided to use bass wood for the final goalie. The use of a dremel was instrumental in working with the bass wood. It allowed us to cut and shape the wood without having it split.

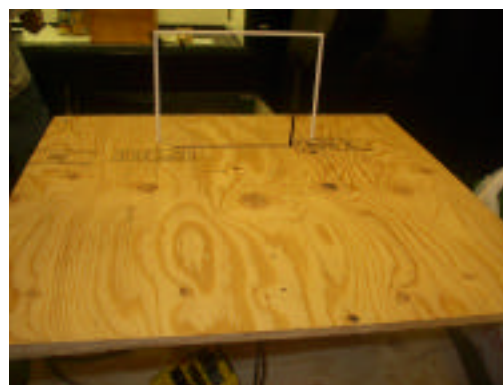


Figure 4: Unfinished field with goal posts

The playing field was constructed like a table with a base to provide room for the mechanism below. A slot was cut in front of the goal posts to provide a track for the goalie control arm to pass through. A hole was also drilled in the surface where the ball was to be “kicked” from. Under this hole a light sensor was eventually mounted.

Finishing the field and goalie was then a trivial exercise in decorating the goalie, felting the field, taping the field markings, and fashioning a net out of a laundry bag.

The mechanical mechanism to move the goalie side to side was by far the greatest challenge. There were a few different mechanisms prototyped and discussed. The original design was composed of a crank slider oriented vertically with a slide placed close to the crank to create an up and over motion for the goalie to follow (See Figure 5). We constructed three or four models out of Lego and balsa wood based on a commercially produced model put together by a classmate. While this design did provide vertical and horizontal motion, it didn't do so in an easy to control or fluid way. We wanted a significant amount of freedom at the top for movement but this led to an inability to control the motion consistently. This mechanism also didn't provide the amount of horizontal motion we wanted.

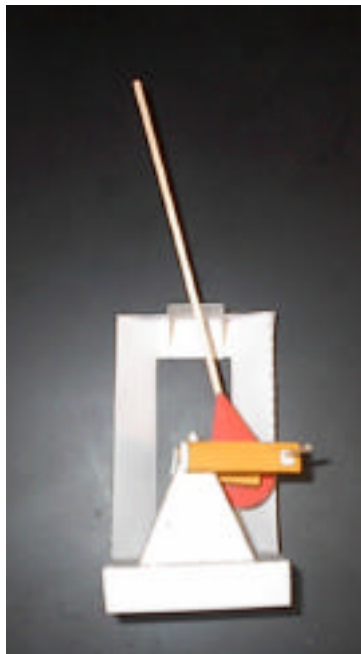


Figure 5: Model the original mechanical design was based on

This led us to look at just moving the goalie side to side. We took as inspiration another paper model (See Figure 6). The new mechanism would be mounted horizontally and provide movement in only one direction. We worked on this mechanism for a week until we had a status meeting with the other Things That Think groups. At this meeting we heard of multiple problems with driving cranks using the Lego motors. Most the problems revolved around trying to create a linkage between the wooden mechanism and the plastic motor. Other problems dealt with the wooden linkages and the friction they experienced. Taking this as an omen we decided to switch our driving mechanism once again.

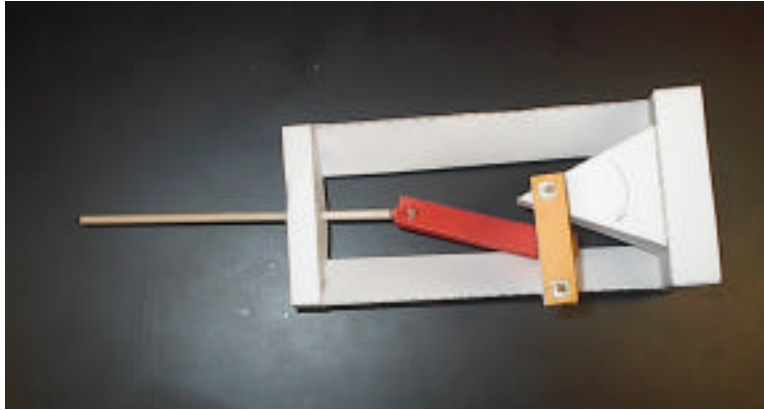


Figure 6: Crank Slider with only one axis of movement

The next and final mechanism consisted of just a Lego motor connected to a Lego gear wheel that meshed with a Lego shaft lined with Lego teeth. This provided the side to side motion and was very simple to build. The construction was all in Lego and was nailed to the base. The decision was made to use Lego exclusively because of the need for fast assembly. We worried that the wooden dowels we were using previously for shafts would not mesh consistently with the Lego pieces. A simple wheel assembly and arm for the goalie was added on the other end of the shaft (See Figure 7). After some testing we added weight to the assembly to stop the gear from slipping. (Notice the box of nails on the shaft's midpoint.) Although we gave up vertical motion we succeeded in providing consistent and easily controlled horizontal motion.

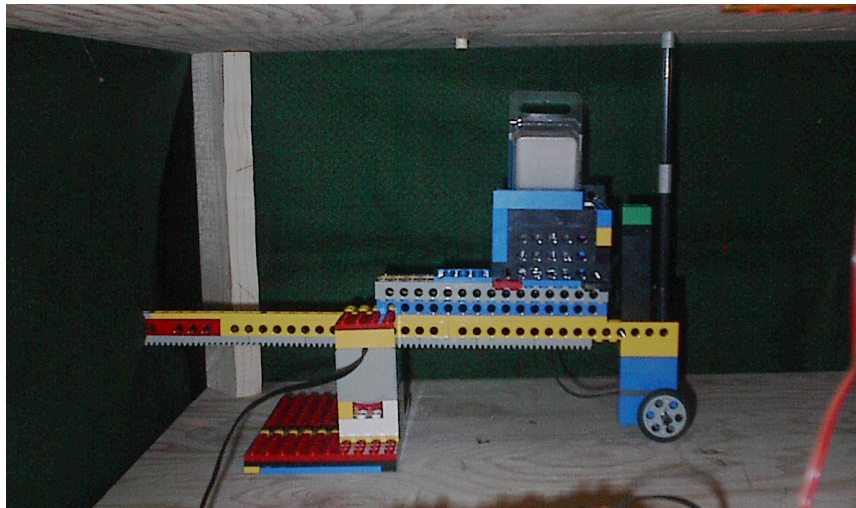


Figure 7: Final drive mechanism

The computational aspect consists of a control module and a light sensor. Given that the original design called for three sensors to detect ball movement and scoring and a cricket only has two sensor ports we chose to go with a Lego mindstorm for control. In the end we only needed one sensor port. The job of the control module is to detect an increase in light and randomly choose a direction for the goalie to move. The five possible choices are far left, left, shimmy, right, and far right. After a timeout the goalie is reset back to its

original position for the next attempt. The duration of motor movement was determined through trial and error. All code is contained in appendix A.

In trying to design a realistic goalie we originally believed that some sort of direction and speed sensors were needed. But we decided just to go for “kick” sensing and random movement at stage one. This turned out to be a good decision. By pure luck the length of the field provides enough time for the goalie to move and the motor was just the right strength to swiftly move the goalie. There was no need for us to determine ball velocity. The interactive experience is also helped by the fact that the goalie moves randomly. If the goalie always moved exactly in the path of the ball it would not feel “real”. It would instead feel like you were playing an unbeatable machine and no one would want to play.

3.0 Evaluation / Education

The overall goal of our project was to create an interactive automaton based on a game that many children play. It is easy for someone to understand how it works just by inspection. No advanced knowledge is needed to understand the mechanics and the actual control logic is trivial. While our automaton is composed of some very simple components they combine to create a fun, engaging diversion.

On its own this automaton really doesn't provide much of an educational experience. Mechanically and computationally it is trivial. Our original design had a much more interesting mechanical system that may have provided some information about linkages. One thing of interest to inquisitive types may be the light sensor. It is easily manipulated and experimented with.

One could take copies of this automaton and allow kids to add sensors to them. The mindstorm programming software is simple and children could easily adjust our code to use these sensors. Touch, angle, color, infrared, and other sensors could be used. A contest to see who could build a better goalie would provide impetus for experimentation. The goalie that blocked the most shots would be the winner. In this way the automaton becomes a tool for teaching about programming and physical sensors.

Our primary goal was to make something entertaining. To this end we succeeded. When we play against the goalie there is a feeling of playing a game. This is the response we wanted to elicit. To improve the response we could add several components. The one that stands out is a better prediction capability for the goalie. We would not want to make it so good that it was impossible to beat the goalie, just harder than random chance.

The addition of score keeping and a response to goals would enrich the experience further. It is normal for people to want to know how well they are doing at a game. It would be nice if the automaton responded to a point in a positive manner. A song or a cheering mechanical crowd could make people smile when they scored. In these ways we could evoke a more involved response.

Appendix A: Mindstorm Code

```

program test {
  #include <RCX2.h>
  #include <RCX2MLT.h>
  #include <RCX2Sounds.h>
  #include <RCX2Def.h>
  sensor light2 on 2
  light2 is light as percent
  macro LOOP {
    forever {
      if light2 > 5{
        SIDE
      }
      else
      {
      }
    }
  }
  macro SIDE {
    counter1 = ( Random 0 to 1000 )
    if counter1 > 500{
      LEFT
    }
    else
    {
      RIGHT
    }
    while light2 > 1 {
    }
  }
  macro LEFT {
    if counter1 > 750{
      if counter1 > 950{
        direction [ ] [ C ]
        on [ C ] for 5
        direction [ C ] [ ]
        on [ C ] for 10
        direction [ ] [ C ]
        on [ C ] for 5
      }
      else
      {
        direction [ ] [ C ]
        on [ C ] for 30
        wait 100
        direction [ C ] [ ]
      }
    }
  }
}

```

```

        on [ C ] for 30
    }
}
else
{
    direction [ ] [ C ]
    on [ C ] for 20
    wait 200
    direction [ C ] [ ]
    on [ C ] for 20
}
}
macro RIGHT {
    if counter1 > 250{
        if counter1 > 450{
            direction [ ] [ C ]
            on [ C ] for 5
            direction [ C ] [ ]
            on [ C ] for 10
            direction [ ] [ C ]
            on [ C ] for 5
        }
        else
        {
            direction [ C ] [ ]
            on [ C ] for 30
            wait 200
            direction [ ] [ C ]
            on [ C ] for 30
        }
    }
    else
    {
        direction [ C ] [ ]
        on [ C ] for 20
        wait 100
        direction [ ] [ C ]
        on [ C ] for 20
    }
}
}

main {
    ext InterfaceType "kFreestyle"
    rcx_ClearTimers
    bbs_GlobalReset([A B C])
}

```

```
try {  
  LOOP  
} retry on fail  
}
```

Appendix 2: Images



Figure 8: Our Tester



Figure 9: Side Shot



Figure 10: Setting up the shot