The (Sort of) Idiot's Guide to Building a Robot
Jacob Bertels and Sam Novosad
Edwardsville High School

The (Sort of) Idiot's Guide to Building a Robot

Approaching a Challenge

Well, let us put it bluntly. In order to approach a challenge, you must first have one. Say you are walking to the kitchen and there is an orange on your table. What will you do with your newly found orange? Will you eat it? Will you throw it? Will you build a robot to pick it up and put it into the blue basket? Well, since this is a robotics paper, I hope you pick the last challenge. Now that you have the hard part done, let us go on to an easier task.

Strategizing

Now that you have settled on picking up your orange and placing it in a blue basket with a robot, you can approach your first line of attack. Your robot needs to somehow pick up the orange and place it in the basket. But how? Could you simply give the robot a claw, or maybe a scoop? Maybe you could build a ramp and push the orange up into the basket? All things considered, you want to pick the simplest design.

Several things to consider when strategizing a design:

- What parts are you limited to?
- How many motors do you have?
- Are there any obstacles in the way?
- How should I distinguish this orange from an apple?
- How can I keep it simple?

Strategizing is not as easy as it looks. Sometimes you have part limitations. For example, you might only have straight pieces or a certain number of connector pegs. In this example, you would actually have to plan out the entire design so you would not run out of parts. Another limitation could be motors. What if you need two and you only have one? Gearing one to do the work of two would be an answer. Obstacles on the course can include both physical and methodical. Sometimes there are physical obstacles such as walls and edges of tables, while other times there are programmable obstacles. These could include tracking of objects or just pure moving about the board. All in all, KEEP IT SIMPLE! The easier you make something, the easier it will be to troubleshoot.

Because the robot is determined by these questions, you must sit down and think about what you have to do. Then after you have spent time developing several solutions for the design of the robot, ask yourself the question, "What is the most promising design?" The best strategy is the strategy that does not fail and will accomplish the orange challenge multiple times. In this paper we will discuss a simple robot with a claw and wheel design. We have also chosen to use a camera in order to find the orange. This is the easiest way we have found to track objects by color.

Construction

Based on all of the design requirements and a detailed analysis of the task set in front of us, we have chosen to go with a simple arm, claw, and camera bot. In this robot we will use an arm attached to a chassis with a claw. We will then have camera center on the objects and then use the robot to move to them. With this design, we hope to capture and pick up the orange, and then drop it into the basket.

Chassis

The chassis is what holds the robot together. This is probably the most important section of building a robot. If your chassis is not sturdy or poorly built, then that will be reflected on your robot's performance. These days, children seem to have an iRobot Create lying around. Since the Create is already structurally sound, we will use this; the majority of your work is complete

To get started, you must first think about what you are going to be mounting onto the Create. Since we are grabbing the orange with a claw, you will need at least one base structure that will support an arm. Then, in order to track the orange and the basket you will need to have a camera. The camera attaches straight into the CBC and thus, the CBC must also be mounted (Recommendation: The camera should be attached to a structure for higher tracking accuracy).

If you cannot get a hold of a Create, here is an example of a Lego chassis that will work just fine: http://community.botball.org/articles/chassis-construction.[1] then click on the Chassis Construction link at the bottom.

Claw

The claw is what will actually accomplish the tasks. In order for the first task to be completed, we must first find the orange and have the claw grab it. Grabbing the ball is is not as easy as it looks. The most basic claw that is efficient will require at least two gears. Rule of thumb, the claw should be big enough to completely grasp the orange and hold it. The first gear should be positioned on the axle attached to the motor/servo. The second gear should have the teeth interlocked with the teeth of the first gear (No slippage). This allows both gears to turn at the same rate. Now you attach the sides of claw. Each gear will get one side. By attaching a side to each gear your servo turns each side inversely. Ta da, a working claw!

Arm

The purpose of the arm is to raise and lower the claw; which allows the orange to be placed into the basket. The simplest way to build an arm is to attach a servo soundly onto your base. Then screw long Lego pieces to the servo horn. The long Lego pieces must

be long enough to raise your claw over the top of the basket. After you have attached the arm pieces to the servo, attach your claw to the end of the arm.

Now that you are done building a simple robot, now you need to make it move. If you want to add onto your robot and improve upon it, have fun building.

Programming

The programming will be broken up into two parts. There is the motion of the robot and the positioning of the actuators. The robot movement is very simple because KISS C has built in functions. The actuators are a bit more difficult, but the camera is where you have to think.

Create Movement

The Create must have a couple basic functions in order to track and retrieve the orange. These functions include moving left, right, forward, and backward. The good thing about using KISS C is that it already includes some built in functions [2]. These include methods that move forward, backward, turn clockwise, counter-clockwise, and stop. With just these five commands, you can successfully navigate almost any obstacle.

Actuators

The actuators include the claw and the arm. In this exercise we used servos. This allows us to get accurate and consistent results each time. This is where more testing comes into play. The servo uses a set_servo_position[2] command that moves a servo to the same position every time. That means you may have one servo for your arm that sets it up and down. Then you may have one servo for your claw that sets it open and closed. By doing these four motions, you can successful pick up the orange and open the claw over the basket.

Camera

The camera is what we suggest you to use to successfully find the orange and blue basket. Using the camera, you can track the color of the orange and the position of the orange in your screen. Using the position of the orange “blob” on the screen, you can turn your robot to directly face the orange. The camera also allows you to track the size of the orange, thus allowing you to move toward the orange until the “blob” is a certain size. You can also use this technique on finding the blue basket.

Sample code: (This code is written in Kiss-C which has built in functions)[2]

```
#include "camlib.h" // uses a prewritten camera library
#define AS 3        // defines servo ports for arm and claw
#define CS 2

void openClaw()
{
       set_servo_position(CS, 1250);
       fsleep(0.5);
}
void closeClaw()
{
       set_servo_position(CS, 300);
       fsleep(0.5);
}
void raiseArm()
{
       set_servo_position(AS, 2000);
       fsleep(0.5);
}
void lowerArm()
{
       set_servo_position(AS, 1300);
       fsleep(0.5);
}
int main()
{
       enable_servos();
       set_servo_position(AS, 1400); // raises Arm
       openClaw();
       fsleep(0.5);
       center_on_BB_analog(1); //centers on orange
       move_until_big(1);      //moves until big
       lowerArm();
       moveForward(5, 500);
       closeClaw();
       raiseArm();
       center_on_BB_analog(2); //centers on basket
       move_until_big(2);
       openClaw();
       disable_servos();
}
```

Operational Testing

The testing is what brings everything you have previously done together. By testing your robot you can find any previous flaws in the hardware or software. Repeat testing until the robot is accurate and precise. Each time you test, you will probably change your code. Don't feel discouraged; this is the longest part.

Works Cited

1. *Welcome Botballers! | Botball Community*. Web. 17 June 2010. <http://community.botball.org/>.
2. Kiss-C Programmers Manual for the CBC Botball Controller