EX-BOT
Table of Contents

Table of Contents..................................................................................2
Introduction ............................................................................................3
Ex-Bot’s Functionalities .........................................................................4
Strategy and Performance.....................................................................7
Programming ..........................................................................................8
Appendix ..................................................................................................<page>
**Introduction**

Egg Hunt is a furious fun strategy, mind-challenging contest that involves robots competing against each other. In this year’s event, there were six teams and we were one of them. All the programming was based on the previous labs, which indeed facilitated all the programming procedures.

The rules are very simple. As shown on Figure 1, the robots have to capture their good eggs and somehow score it towards the single goal. There were 10 eggs for each color (yellow and black). Compared to last year where it had two goals, this year’s contest had become much more challenging for having a single goal. A strong yellow light was implemented right above the goal. The light made it easier for the robots to locate the goal. We had one month to get prepared for the contest.

![Figure 1: Representation of the Egg-Hunt Arena.](image)
Ex-Bot's Functionalities

Ex-Bot was the smallest robot in this year’s competition. It is 12 inches wide by 11 inches width. Our goal was to make the robot small and low, so that it could be agile and capable of sliding under the goal to drop the eggs.

Ex-Bot was made of several sensors, such as touch, sonar and light sensors. One servo (Figure 2) was used to control the main egg entrance. IR LED was implemented to improve the reading for the reflecting light sensor (Figure 3) to decide whether it is a “good egg” or a “bad egg”. It also had a select egg switch that could choose what is going to be the good egg and the bad egg.

The front of the robot had a gate, which opened as long as the reflecting light sensor detected that there was a good egg. We put a sonar sensor (Figure 3) located on the edge of the egg’s entrance. The sonar sensor actually tells the robot that an egg is approaching the gate. The reason for us putting the sonar sensor was to eliminate the color ambiguity problem we were having before. Since the goal’s light was pointing directly down to the table, the reflecting light sensor was having problems identifying the yellow egg. The values for the floor and the yellow egg were identical. The sonar sensor and some LEGO adjustments fixed this problem.

Also, on the front of the robot, we implemented one sliding wheel on each side (Figure 4), therefore, when the robot hits a dead spot, it will be able to get out without any problems. Furthermore, we implemented one touch sensor on each side (Figure 4), so while the robot is on “wonder state” and it happens to hit a wall, it will actually just drive backwards and go to a different path.

After the robot gathered 3 good eggs, Ex-Bot will change it’s mode to light seeking. For that, we implemented two light sensors on the back. The robot will actually drive backwards towards the goal, and then drop the eggs. Ex-Bot will be able to get out of this state when the “shoulder touch sensors” (Figure 5) are being pressed for about one second. The shoulder touch sensors are located on the highest spot of the robot, so they will be high enough for the goal’s top bar.

Figure 2: The servo gate.
Figure 3: Light reflecting and sonar sensors.

Figure 4: Sliding wheel, touch sensor, and color select switch (red arrow).
Figure 5: Shoulder touch sensor
Initially, our plan was to collect four eggs and drive to the goal as many times as possible. We ended up changing our plans, because the robot could take a lot of time just looking for eggs and never go to the point of drive towards the goal to the eggs.

Since, there was a possibility of the eggs getting spread close to the walls; we used the help of our two sliding wheels and of our touch sensors (Figure 4). If the robot had gotten to a point that the touch sensor is inefficient on a determined spot, the side wheels will automatically drive robot out of that dead spot, using the power of the motors.

We were very disappointed with the performance of our robot. We were very confident of our software, but mainly our frame, which turned out to be the cause of our robot’s failure. We also ignored couple of features that could have helped, but due to lack of time, we finished the main parts and decided to run the robot as it was.

The first time we entered in the arena, we set up four eggs to be stored inside the robot, which obviously was not the correct choice. Ex-Bot, before it broke, was looking for eggs forever. Our robot already had two eggs stored inside it. But when the software was running smooth, the right part of the frame came off and the motor started to spin. After we noticed the strange noise, the whole frame completely broke in half. That was the end of our first match.

Before the second match, we reinforced the frame, so it will not break in half again. We were confident again of our robot. We also noticed that we could have implemented one touch sensor on the back, in case the robot gets stuck while in the light seeking state.

The second match came and we had the obligation to winning in order to follow through the competition. The match started, and within seconds our robot already had one egg inside. We thought that this time, we would be the winners. Through the mid-time, one part of our robot fell, but it still had the entire frame was intact, but the worse came right after the little piece came off. The right motor just gave up working, and the robot started to drive in circles. After the contest, we took our robot a part and we came to know that right motor was getting too much pressure due to one bent LEGO stick.

After all the work we have put in, we failed on testing the robot running for some period of time. We could have spent more time testing the frame of the robot, instead of just worrying about the codes and all the elements attached to the robot.
Programming

ex-bot-main.vi

Connector Pane

Block Diagram
Hold is full or time has expired. Deliver to goal.
ex-init.vi

Connector Pane

Block Diagram
GateControl.vi

Connector Pane

Block Diagram
LCD Display Driver.vi

Connector Pane

Block Diagram
LCD String Prep.vi

Connector Pane

Line String: abc 1 4

Output Line: abc 1 4

Block Diagram

If greater than 15 characters, truncate the string.

How many space characters are needed to fill the line.

Adds a space per iteration.

Output Line: abc 1 4
ex-deinit.vi

Connector Pane

error in

Block Diagram

Bumper-Check-Motors.vi

Connector Pane

Block Diagram
No bumpers pressed, pass normal values.

Left Bumper Pressed, back up right side

Right Bumper Pressed, back up left side
Both bumpers pressed, do straight back up.

**SignedMotorControl.vi**

**Connector Pane**

- Motor Number
- Directional Motor Power
- Motor Ctrl
- Speed (readout)
- isForward (readout)
- error in (no error)
- error out

**Block Diagram**
CaseMaker.vi

Connector Pane

Block Diagram

Takes two binary inputs and returns the decimal equivalent.

GlobalMotorControl.vi

Connector Pane

Block Diagram
EggDetector.vi

Connector Pane

Block Diagram
If an egg is detected, it takes a color reading, then use the accept or reject function.

If no egg is detected, this will simply exit.
**egg-presence-detect.vi**

**Connector Pane**

- Output Active
- $x < y$?
- error in (no error)
- error out

**Block Diagram**
**egg-IsDesiredColor.vi**

**Connector Pane**

**Block Diagram**

Returns true if egg presence is detected
ColorDetector.vi

Connector Pane

Block Diagram
Returns true if egg is yellow

yellow or black egg????
yellow<100
Black>700
Else = BAD READ
egg-accept.vi

Connector Pane

Block Diagram
egg-reject.vi

Connector Pane

Block Diagram
unit-Left-turn.vi

Connector Pane

Block Diagram

Produces approx. 30 degrees of left rotation.

goal-align-rough.vi

Connector Pane

Block Diagram
ReadLightSensors.vi

Connector Pane

Output Active

Left Light Sensor

Right Light Sensor

error in

error out

Block Diagram

goal-deliver-straight.vi

Connector Pane

error in

error out
Block Diagram
Time Expired. Run fine-tune goal finding.

Although right now there isn't one, so we'll use the rough goal find.

Left Shoulder Pressed:
Cut left motor.

Reduce right motor speed by half
GlobalMotorControl-LEFT.vi

Connector Pane

Block Diagram

GlobalMotorControl-RIGHT.vi
Connector Pane

Right Motor — Motor Ctrl

Block Diagram

Right Motor

Right Motor