General Information:

Instructor: Darren Dawson (ddarren@clemson.edu)
Office: Riggs 105
Office hours by appointment only

TAs: Bill Kolodzey (wkolodz@clemson.edu)
Office: Fluor Daniel 323
Available during scheduled class times or by appointment

Location: Holtzendorff Project Lab (Room B-01)
Meeting Time: TBD
Course Webpage: http://www.clemson.edu/ces/departments/ece/undergrad/mindstormslab.html
Credits: 1

Course Description:

In this ECE robotics project, student teams will learn engineering and design principles by building and programming robots with Lego Mindstorms and MathWorks MATLAB. The Mindstorms kits allow for easy construction of advanced robots complete with servo motors and a variety of sensors. Using MATLAB, a mathematical programming tool widely used in industry and academia, to control the robots opens up a vast feature set beyond that provided by the basic Mindstorms software. In the first part of the class, students will go through several preliminary experiments to learn the basics of controlling the robot’s motors and reading from its sensors with MATLAB. Those sensors include touch, light, color, and ultrasonic. Also, MATLAB and its built in image processing functionality makes possible the use of a standard webcam as an additional sensor. After gaining a firm understanding of the fundamentals of design and the use of the Mindstorms and MATLAB toolbox, these student teams will complete a design challenge by designing, building, and programming their own two-legged walking robot.

Attendance Policy:

The Holtzendorff project lab will be available during the scheduled class meeting time for students to work on projects. The TA will be there during those times to answer questions and provide assistance. Students can stop in at any time during those hours. Students will be also able to take the Mindstorms kits with them and work on the projects outside of class if they choose, so attendance is not required. However, students will have to come during one of the scheduled meetings each of the first five weeks to get their work on the lab for that week checked off.
Schedule (Subject to Change):

<table>
<thead>
<tr>
<th>Item</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekly progress &amp; individual participation</td>
<td>Semester long</td>
</tr>
<tr>
<td>Introduction to class</td>
<td>Week of 9/3</td>
</tr>
<tr>
<td>Sample robot construction &amp; Lab 1 (Connections)</td>
<td>Week of 9/10</td>
</tr>
<tr>
<td>Lab 2 (Motors)</td>
<td>Week of 9/17</td>
</tr>
<tr>
<td>Lab 3 (Touch Sensor)</td>
<td>Week of 9/24</td>
</tr>
<tr>
<td>Lab 4 (Color &amp; Light Sensors)</td>
<td>Week of 10/1</td>
</tr>
<tr>
<td>Lab 5 (Ultrasonic Sensor)</td>
<td>Week of 10/8</td>
</tr>
<tr>
<td>Introduction to final project</td>
<td>Week of 10/15</td>
</tr>
<tr>
<td>Robot Prototype &amp; Progress Report</td>
<td>Week of 11/5</td>
</tr>
<tr>
<td>Finished robot &amp; Final Report/Presentation</td>
<td>Week of 12/3</td>
</tr>
</tbody>
</table>

Grading:
Completion of all assignments is required. Grades will be heavily based on participation and effort put forth in each part of the class. Since attendance is not required, the two reports and presentation are the place to demonstrate effort. Reports and presentations that clearly describe the students’ methods for working through and solving problems will be scored high.

The prototype should demonstrate progress towards the final goal. Each group should have finished building their robot and should have begun the programming phase. The progress report should be no more than 1 page and the final report no more than 2. Both should include specific examples of challenges met and how they were overcome.

The final presentation will be submitted by email. Instructions will be given for recording narration in PowerPoint. The presentation must also include a video of the robot in action. It should be approximately 5 minutes long and cover much the same material as the reports.

Academic Integrity
Clemson University’s academic integrity policy reads: “As members of the Clemson University community, we have inherited Thomas Green Clemson's vision of this institution as a "high seminary of learning." Fundamental to this vision is a mutual commitment to truthfulness, honor, and responsibility, without which we cannot earn the trust and respect of others. Furthermore, we recognize that academic dishonesty detracts from the value of a Clemson degree. Therefore, we shall not tolerate lying, cheating, or stealing in any form.” Students will be held to this standard.
Summary
The final project for this class is to build and program a robot that solves a Rubik’s cube. Your
team will be provided with instructions for building the robot shown above, as well as MATLAB
code to have it scan a Rubik’s cube and generate a solution. Your task will be to write an
algorithm to have the robot manipulate the cube to solve it.

The programming of the robot is completely up to you. Any MATLAB code submitted must
have been written by your team. You are free to base your code on control algorithms found
on the Internet or elsewhere, but you must cite your sources and explain your choices in your
reports.
Due Dates

Week of 11/5       Robot prototype and progress report

Week of 12/3      Finished robot, final report and presentation

Reports

There are two reports due for this project. These reports should document how you went about the design and programming processes. They should include problems that arose and how you adapted your design or program to deal with them. Also, they should explain how your MATLAB program works to keep the robot balanced. The progress report should be no longer than 1 page and the final report no longer than 2 pages.

Presentation

In the final presentation, your team will present your project to the instructor. As with the reports, you should explain how your program works and also any major challenges that you encountered during the project and how you dealt with them. The presentation should last about 5 minutes. You should create a PowerPoint presentation and record narration over it. Instructions on how to do that are available on the course webpage. The presentation must also include a video of your robot in action.
Provided Code
You will be provided with MATLAB code for the following functions. These can be downloaded from clemson.edu/ces/departments/ece/undergrad/mindstormslab.html.

cubeSolver

The main program.

initCubeSolver

Resets all parts of the robot (color sensor, turn table, and arm) to their starting positions.

scanCube

Scans each face of the cube with the color sensor. Returns a 6x3x3 cell array containing the color of each square of each face. This function is protected because it contains code for manipulating the cube.

correctCube

Sometimes, the color sensor incorrectly reads a color. This function opens a GUI that allows you to manually set the color for each square of each face of a cube. Returns a 6x3x3 cell array containing the color of each square of each face. Can also take a 6x3x3 cell array as input to initialize the color of each square in the GUI.

solveCube

Generates a series of moves to solve the current cube. Takes a 6x3x3 cell array as input and returns a string containing a sequence of faces to turn and the direction to turn them.

For example, if solveCube returns L R’ U’ D, you would turn the left face clockwise, the right face counterclockwise, the top face counterclockwise, and the bottom face clockwise.

CS_Options

Allows ports for motors and sensors to be set. Also allows motor power for cube flipping arm to be changed.
doSolution

Manipulates the cube to solve it. Contains code for how to split up the string returned by solveCube into steps. As is, it just prints out each step.

turnToStop

Turns a motor until that motor runs into an obstruction and cannot turn any further. Takes as arguments the motor, direction, and power. Read the comments at the top of the file for more information.

Using the Provided Code

The cubeSolver function contains the basic sequence of the program. However, it is incomplete and you will need to add your own code to the doSolution function.

The scanCube function will have the robot check each square of each face of the cube with the color sensor. It is important to call initCubeSolver before calling scanCube, as the movement of the color sensor depends on it starting at the position that initCubeSolver leaves it in. The scanCube function finishes with the cube and color sensor in the same positions as they started.

Also, the color sensor does not distinguish between red and orange. To get around this, if it detects a square as being red, it will switch from color mode to a light intensity mode. This allows it to (sometimes) tell the difference between the brighter orange and the darker red. However, it needs some reference measurements to know how bright each color is. Thus, the first time per MATLAB session that you call scanCube, it will require you to calibrate the sensor. It will prompt you to place the cube on the turntable so that the face with the red/orange center is pointed up and then it will scan the center square of that face. It stores the measured values in global variables so that they can be accessed later on while scanning the other squares.

Still, the colors read may not be 100% accurate, especially the oranges/reds. Thus, the correctCube function opens up a GUI that allows you to manually set the color of each square on the cube. The correctCube folder contains the actual figures and code for the GUIs, and the correctCube function just runs either the Windows or Mac version, depending on which type of computer you have.

The correctCube function can be used to correct errors made by the scanCube function with the code:

```matlab
variable = scanCube();
variable = correctCube( variable );
```
Alternatively, to save time and battery life while testing, you can just use the `correctCube` function to manually enter the whole cube. Calling `correctCube` with no input, as below, starts the GUI with every square blank instead of colored in.

```matlab
variable = correctCube();
```

If you don’t use `scanCube` while testing, be sure to call `initCubeSolver` before running your code, as your code must assume that all parts of the robot start in the positions set by `initCubeSolver`.

The `solveCube` function calls an external program, either `solver_win.exe` or `solver_mac` depending on which type of computer you have, to generate a solution. The first time one of these programs is run, it generates an approximately 500 megabyte file to speed up the solving process. Thus, it is recommended that you run the program outside of MATLAB once and wait for it to generate the file and then exit. This will cut way down on the time it takes for `solveCube` to run the first time. For more information on why this file is required, see [http://kociemba.org/cube.htm](http://kociemba.org/cube.htm).

The solution generated comes back as a string, with each letter specifying the face to turn. Turns are clockwise unless the letter is followed by an apostrophe, in which case the turn is counterclockwise. For example, if `solveCube` returns `L R’ U’ D`, you would turn the left face clockwise, the right face counterclockwise, the top face counterclockwise, and the bottom face clockwise. Keep in mind that which face is considered the front/top/left/... is defined by the starting orientation of the cube, not the current orientation.

The `CS_Options` function allows you to change a few parameters used in the given functions without having to access the source code for those functions. First, it allows you to edit the ports used for each motor and the color sensor. This means that you don’t have to worry about plugging the cables into the correct ports; you can just define what the correct ports are. Also, it allows you to set the power used for the flipper arm in the `scanCube` function. You may have to increase the power as the battery charge decreases.
Record a PowerPoint Slide Show (Office 2010)

PowerPoint allows you to easily record timing and narration for a slide show. You simply click “Record Slide Show” and then start talking (assuming your computer has a microphone). The timing of when you advance slides will also be recorded.

Instructions

From the “Slide Show” menu, select “Record Slide Show.” You can choose to either start recording from the beginning of the presentation or at the current slide.

This should bring up the full screen slide show view along with a small window in the top left corner of the screen.

This window controls the recording. The arrow button on the left advances to the next slide, the pause button pauses recording, and the U turn discards the recording for the current slide and starts it over. The time on the left is the time recorded for the current slide and the time on the right is for the entire presentation.

When you are finished recording your presentation, close the recording window and exit the slide show. You can review what you’ve recorded by playing the slide show from the “Slide Show” toolbar.
You can also have your mouse pointer show up in the recorded slide show by holding both the CTRL key and the left mouse button while moving the mouse around. The pointer will be displayed as a “laser pointer.”

After recording a slide, in the left column of the “Normal” view, there will appear a star icon under the slide number and an audio icon in the bottom right corner of the slide to indicate that the slide has a transition and narration set.

![Image of presentation view](image)

The audio icon also appears in the bottom right corner of the slide itself in “Normal” view.

![Audio icon](image)

You can mouse over this to play the narration or adjust its volume. Also, you can select the icon and delete it to remove the narration and timing from that slide.
**Microphone Settings (Windows 7)**

To adjust the volume on your microphone, open the Control Panel. Click on “Hardware and Sound” and then click on “Sound.”

![Hardware and Sound](image)

In the window that opens, select the “Recording” tab.

![Sound](image)

Select your microphone device, and click the “Properties” button. In the window that opens, select the “Levels” tab. From here, you can adjust the microphone volume and boost.

![Microphone Properties](image)
Export to Video

After you have recorded all your narration, you can convert your PowerPoint presentation to a video. In PowerPoint, open the “File” menu and select “Save & Send.” Under “File Types,” select “Create a Video.” In the options on the right side of the screen, select “Internet & DVD” and “Use Recorded Timings and Narrations.” When you are ready to start, click “Create Video.” This process can take several minutes, depending on the length of your presentation and the size of embedded videos and narration.

You will be prompted for a location to save the video. This will create a Windows Media Video (.wmv) file, which is playable in Windows Media Player.