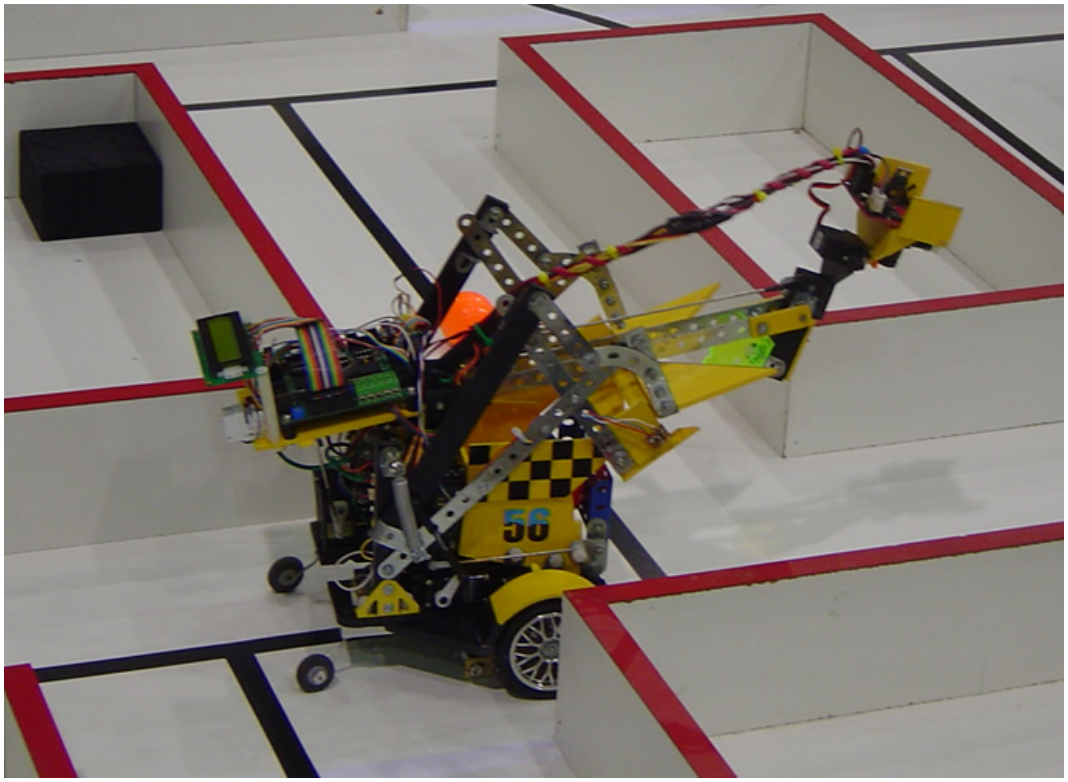
Caddy

A 2005 Roborodentia entry with vision and path planning abilities



By: Taylor Braun-Jones

Advisor: Dr. John Seng

Computer Engineering

California Polytechnic State University - San Luis Obispo

March 2013

Contents

1 Introduction

1.1 Problem Summary

Caddy is a robot entered into the 2005 Roborodentia. Roborodentia is an annual autonomous robotics competition held during Cal Poly's Open House by the IEEE Computer Society. Robot entries must navigate a maze searching for three randomly placed golf balls, collect them, and then deposit the balls in the "nest" at the end of the maze. The 2005 competition also included a new aspect. Two bonus balls were placed on a platform behind the wall in two predetermined corners of the maze such that the top of the golf ball was flush with the top of the wall.

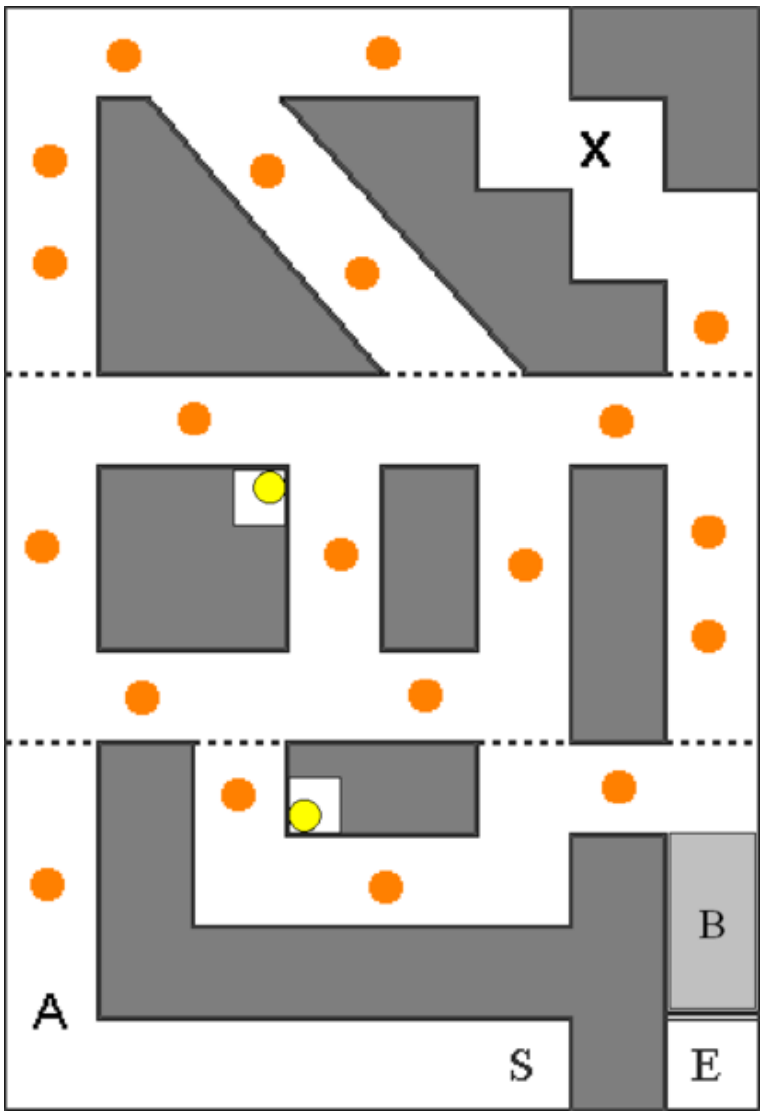


Figure 1: Arena map showing the two fixed bonus ball locations and the potential locations of the randomly placed ground balls

The competition scoring breakdown is as follows:

Point Value	Task
1	Passing the first turn in the maze - Point A
1	Triggering “nest” solenoid by activating optical sensor - Point X
3	Touching each ground ball (1 point per ball)
3	Collecting and possessing each ground ball (1 point per ball)
3	Bringing each ground ball to “nest” - Area B (1 point per ball)
9	Placing each ground ball in “nest” - Point E (3 points per ball)
10	Collecting and possessing a bonus ball - 2 Yellow Balls (5 points per ball)
6	Placing a bonus ball in the “nest” (3 points per ball)
36	Total possible points

In the case of a tie, the robot with the fastest time wins.

2 Goals and Requirements

The rules of the competition dictated a baseline set of goals that need to be met to be successful:

- **Line following** - The corridors of the maze were constructed with a black electrical tape line down the center, meant as an aid for the autonomous navigation of the pathways - and we saw no reason not to take advantage of it.

- **Junction detection** - To navigate the maze, Caddy would need the ability to detect when a junction was reach and to identify the type of junction (e.g. "T" junction, straight-or-right-turn junction, etc).

- **Ground ball collection** - The maximum score without collecting any ground ball is 5 out of 36 possible points - so a ground ball collection system is a must to score well.

- **Bonus ball collection** - Since the scoring distribution weighted bonus balls so heavily (16 of the 36 possible points are awarded for bonus ball tasks), we also decided that Caddy would need bonus ball collection capability in order to be competitive.

- **Ball release-into-nest system** - For an additional 3 points per ball, having the ability to release balls into the nest seemed worthwhile and comparatively simple to implement.

In addition to this baseline set of goals, we decided to focus on the autonomous aspect of the Roborodentia competition. In particular we wanted a robot that could *actively adapt* to the random ball locations (unlike any previous entry had ever done). This was the driver for an additional two requirements:

- **Path planning** - Caddy needed a way to map the arena and a shortest path algorithm that could find the best path through a sequence of goals.

- **Ball finding** - To make the best use of the path planning algorithm, we needed a way to actively search for balls down untraveled corridors.

3 Design

3.1 Collaborative Team Process

The team for this project was formed from interested members of the the Cal Poly Robotics Club.

To organize the tasks and identify critical paths in the (short) project time line, we used GanttProject to create a Gantt chart.

For code control and collaboration we used Concurrent Versions System (CVS). Since this project had a competitive nature, we chose to setup and host our own private CVS server rather than use a free, Internet-based hosting service.

Between face-to-face team meetings we used Drupal to host a private forum for discussing ideas, sharing progress, etc.

The inline code documentation and this project report were both managed using Doxygen. Keeping the documentation in plain text and means that the documentation can be version controlled the very same way as the source code. The documentation also tends to stay more up to date since it can be more conveniently updated at the same time as the source code.

3.2 System Architecture

When taking a holistic look at the project goals and requirements, it is clear that a camera-based vision system can satisfy line following, junction detection and ball-finding requirements. The image processing required for these task can all be done with a camera that is low resolution, low power, and low cost. The ball finding task, in particular, has few other options that are both low cost and simple to implement. The CMUcam2 developed by students at Carnegie Mellon University and sold through distributors as a packaged vision system, met our needs well.

Since the CMUcam can handle all the computationally intensive image processing as well as drive 5 servo control outputs, our requirements on the main microcontroller were fairly relaxed. The most computationally demanding task for the main microcontroller is the shortest path algorithm, but with a relatively small map even this could be handled by a low-end microcontroller.

3.3 Electrical Design

3.3.1 Power Regulation and Motor Controller

We opted to design and build our own power regulation and motor control circuitry over purchasing one of the more expensive off-the-shelf solutions. To ensure that we were not debugging software and electrical issues at the same time (difficult!) made sure to include robust regulation and decouple in the design. Our power sub-component power needs were:

- +5V regulated power for the ATMega32 and the rest of the electronics
- +6-15V unregulated power for the CMUcam2
- +6V for each motor, controllable via logic-level signal

Unregulated voltage was connected to the CMUcam2 (it had a built-in regulator) and to the motors. Although not ideal, connecting the motors to unregulated power meant we could save on cost by using a smaller, cheaper voltage regulator. We opted for a linear regulator to supply the +5V to the ATMega32 microcontroller.

In the choice between a linear regulator and a switching regulator, a linear was chosen over a switching regulator for the following reasons:

- Lower output noise - We wanted to take every precaution we could to ensure the EMF voltage generated by the motors did not affect the electronics.

- Simpler to implement - Switching regulators typically require more passive components to filter the inherently higher noise they generate.

- Cheaper

Switching regulators are more efficient, however any efficiency gains would be dwarfed in comparison with the power demands of the unregulated components (DC motors and CMUcam2). The circuit diagram for our implementation is shown below:

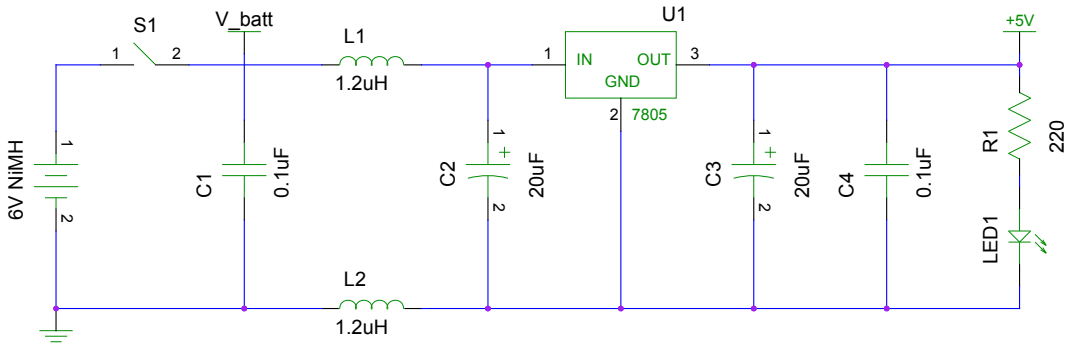


Figure 2: Power regulation circuit

To control the motors via digital signal we used an H-bridge circuit. For added protection of the electronics from the back-EMF voltage of the motors, additional diodes were connected as shown in the schematic below.

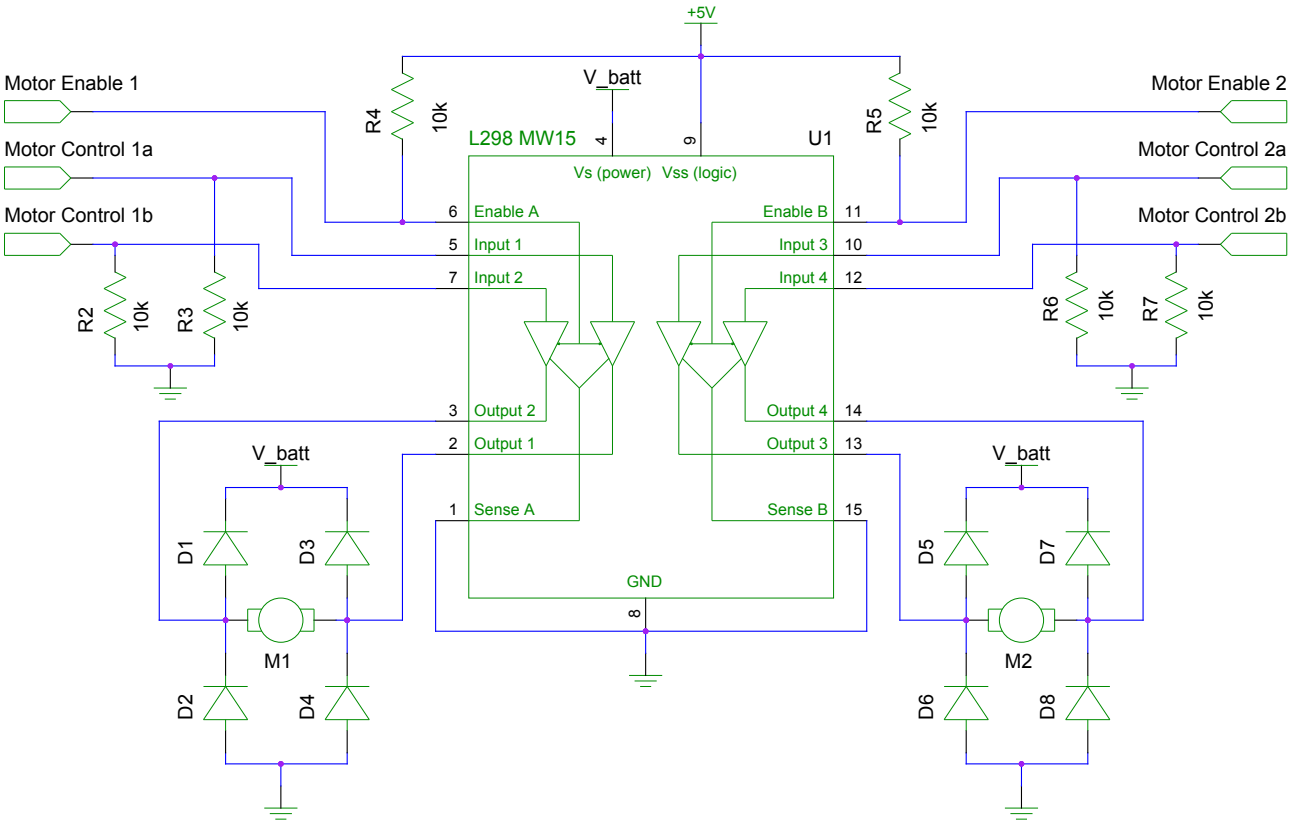


Figure 3: Motor control circuit

The power regulation and motor control circuits were fabricated together on a single board using a copper-plated board, etching solution, and a laser print out of the circuit layout. We made sure to use polarized headers for all the connections to avoid making any reverse polarity mistakes (we still managed to make a couple!).

3.3.2 IR Break Beam

In order to capture ground balls, Caddy needed a way to detect when a ball was within the grasp of its lift mechanism.

Originally planned to use the centroid tracking feature of the camera since the camera would always be facing down for line tracking during the times it needed to detect ground balls. This turned out to be difficult mainly due to the limitations of the camera. When the camera is configured to track a black line, glare from the overhead lighting and red golf balls have the same effect on what the camera detects – a gap in the line. We tried to simply change modes whenever a gap was detected, determine if the gap was due to a glare or a due to a ball, and then act accordingly. Unfortunately, the CMUcam did not handle rapid mode and parameter changes well, taking too long to switch from one mode to another. This lead to a failure in our carefully tuned PID line tracking algorithm which relied on frequent, regular updates over time. We considered and experimented with some ways of solving this problem but none were the quick, elegant solution we were looking for.

With a fast approaching deadline, we opted for a quick, but less elegant, solution. We installed an IR break beam sensor looking across the front of the lift arm so that when the arm was down and fully open a ground ball would break the beam as is passed under the arm.

3.3.3 Servo Reverser

The mechanical design of Caddy required 6 servos:

- Ball pickup, left side
- Ball pickup, right side
- Boom control
- Ball hopper
- Tilt action
- Pan action

This meant that the original plan to use the five servo control outputs of the CMUcam would be inadequate.

The following approaches were considered for accommodating the 6th servo output:

- **Mechanical:** Modify the mechanical design so that the ball pickup mechanism could be controlled by just one high-torque servo. The lift mechanism had already been iteratively refined to the point that it could be actuated by just one mechanical motion. Going this route would likely mean some major rework of the mechanical design - not ideal since we were otherwise happy with the mechanics of the robot.

- **Software:** Use some of the extra pins on the ATmega32 to generate a servo PWM signal. Unfortunately we were already using the two PWM peripherals on the ATmega32 so we would have to do this in software. We had limited timer resources on our chip and weren't sure how we might need to use them in the future so this was not an ideal solution.

- **Electrical:** Leverage the fact that the 2 servos controlling the ball pickup were the same signal, 180 degrees out of phase. This seemed like a perfect application for a simple 555 timer circuit.

Searching the Internet, we found we were not the only ones to think of using a 555 timer to create a servo reverser. Using the well documented plans from C. Dane [1] we fabricated a board the size of a postage stamp.

3.3.4 Wheel Encoders

The maneuvers needed at junctions and for the bonus ball pick up sequences needed to be accurate and repeatable. To achieve this we used a black and white encoder disk that we printed out and glued to the inside edge of each drive wheel. [2]

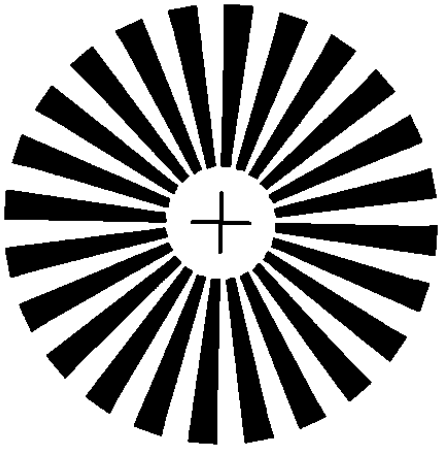


Figure 4: Reflective IR wheel encoder pattern

3.4 Software Architecture

3.4.1 Computing Platform

For our our computing platform we chose an ATmega32 microcontroller from Atmel's 8-bit AVR line of microcontrollers because it was C-programmable with free open-source tools and because we had a readily available development board, the ERE EMBMega32.

Figure 5: EMBMega32 development board from ERE CO.,LTD

3.4.2 PID Line Tracking

To track the black electrical tape line, we implemented a proportional–integral–derivative (PID) controller. In PID theory, the output of a PID controller, $c(t)$, is defined as:

$$c(t) = P_E e(t) + P_I \int e(t)dt + P_D \frac{de}{dt}$$

Where $e(t)$ is some error function and P_E, P_I, and P_D are adjustment coefficients for the observed error, the integrated error and the derivative of the error, respectively. We define our error term:

$$e(t) = \frac{dx}{dt}$$

By substitution, we get:

$$c(t) = P_E \frac{dx}{dt} + P_I x(t) + P_D \frac{d^2x}{dt^2}$$

Broken down and interpretted for the task of line tracking, these terms are:

- $P_E \frac{dx}{dt}$ ← How fast are we drifting from the center line?
- $P_I x(t)$ ← How far are we from from the center line?
- $P_D \frac{d^2x}{dt^2}$ ← How fast is the drift rate accelerating?

Provided that there is a way to measure or compute each of these terms, this is a more robust form of the equation because it eliminates the integral term which can cause problems due to accumulated error.

The figure below shows how the camera was used to measure $P_E \frac{dx}{dt}$:

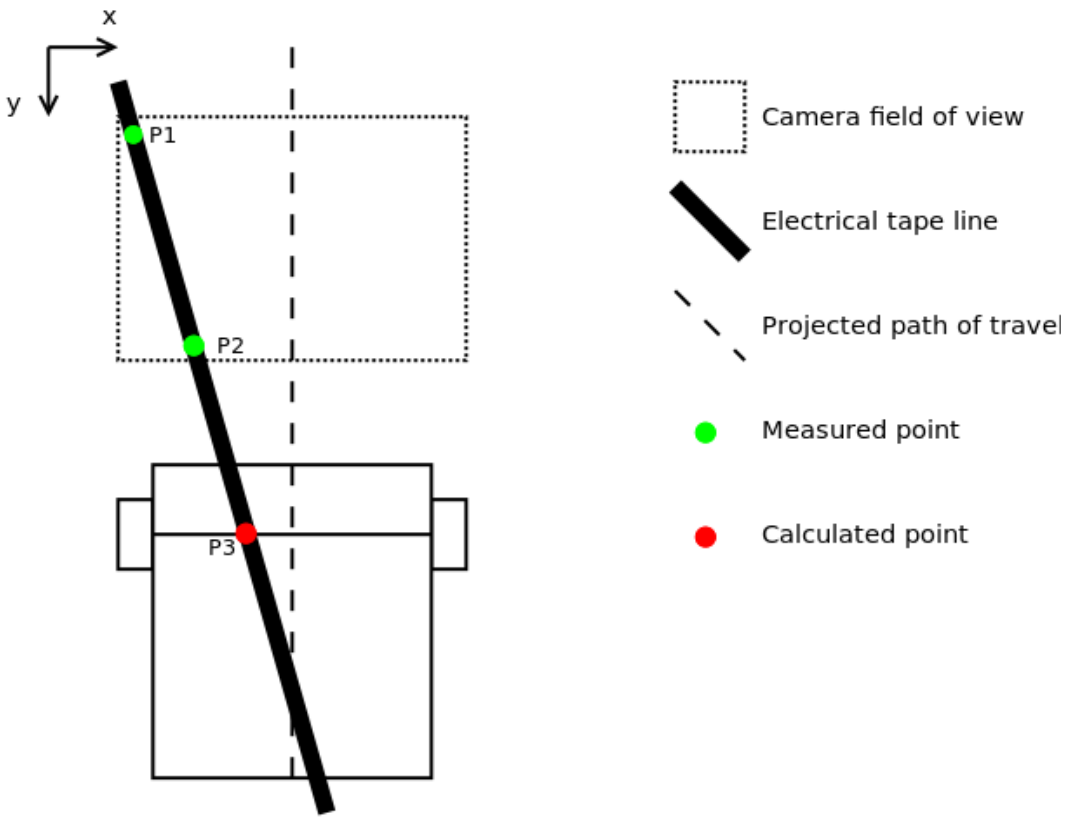


Figure 6: Diagram of line tracking geometry (NOT to scale)

The drift rate is the slope of the black line with respect to the center line of the robot. For points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ from the diagram above we define:

$$\frac{dx}{dt} = \frac{y_2 - y_1}{x_2 - x_1}$$

And for point $P_3 = (x_3, y_3)$ with constant, measurable value for y_3 and for constant, measurable line center x_{center}

$$x_3 = x_{center} - \frac{dx}{dt}(y_3 - y_1) + x_1;$$

Lastly, by storing the previously computed value of $\frac{dx}{dt}$, we can compute the third term:

$$\frac{d^2x}{dt^2} = \frac{dx}{dt} - \frac{dx}{dt}_{previous}$$

The coefficients for each of these terms were determined by trial and error using a tethered remote and stored persistently in EEPROM.

3.4.3 Maneuvering

When turning our bot by a certain number of ticks, we experienced overshoot despite actively applying DC motor braking. We addressed the problem with the following software solution.

After turning for the desired number of ticks, we applied braking and counted the number of excess ticks that occurred from the instant braking was commanded. After a fixed delay, we drove the wheels in the opposite direction for that same number of ticks.

This worked well for the most part, however, with different battery charges, turn amounts, and turn types, the amount of time to brake was never the same. If we did not brake the motors for a long enough delay, our bot would stop counting excess ticks and begin to drive the motors in the opposite direction, too soon. With our unsophisticated encoders that cannot detect the direction of wheel motion this resulted in "reverse ticks" being counted before the wheel had actually started moving in the reverse direction.

3.4.4 Ball Detection and Localization

3.4.5 Path Planning

4 Conclusion

4.1 Future Work

4.1.1 Regulated Motor Voltage

If we end up working with higher voltage motors again, it may be worth losing some voltage in order to send regulated voltage to the motors. This allows for more consistent operation across fresh and low batteries. This should also condition our batteries better, because the battery voltage can drop until the regulator's threshold is reached. As long as we drop the voltage down enough, it should be obvious when batteries are dead. As the regulator cuts out, the bot should slow down more dramatically. A common solution in ME 405 (mechatronics) is to regulate 14.4V down to 12V with an adjustable regulator (LM1084) for each motor. One regulator was not able to provide enough current for both motors. When driving the motors at the same pulse width, we were able to see a difference in how straight the bot drove, if the motors received voltages a few hundredths apart. Yet, just having the ability to precisely and consistently set the voltage to the motors was very useful.

4.1.2

Since we were using timer 1 (a 16-bit timer) for PWM, we could have used 16-bit PWM. 8-bit resolution seemed to be sufficient with the original 6 volts motors, but when we switched to 12 volt motors, more precise control of the PWM signal would likely have improved the PID line tracking. As it was, we had to use a PID offset constant of 1 which means that we would have required division to decrease the proportional coefficient parameter of our PID control algorithm.

4.1.3 Quadrature Wheel Encoders

Quadrature wheel encoders would have required more mechanical work (to mount the reflective IR sensors 90 degrees out-of-phase) and electrical work (wiring for twice as many sensors) but it would have helped solve some challenges with maneuvering the robot through precise sequences such as the bonus ball pickup.

Quadrature encoders would have allowed us to perform overshoot correction easily and accurately.

5 Appendix

5.1 Gantt Chart

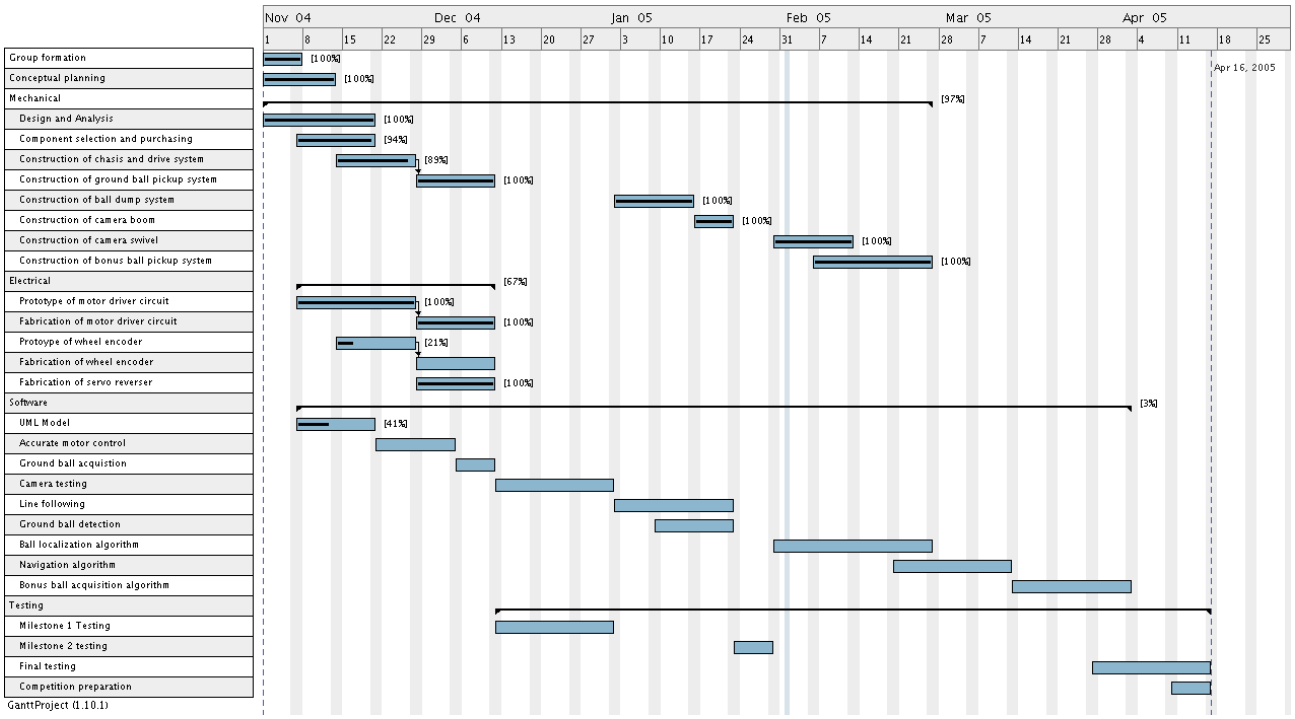


Figure 7: Caddy project gantt chart

5.2 Individual Contributions

Caddy was a joint effort between Taylor Braun-Jones, Logan Kinde, Tyson Messori, Scott Barlow, Michael Shelley, and Patrick McCarty. Primary contributors were Taylor, Logan, and Tyson.

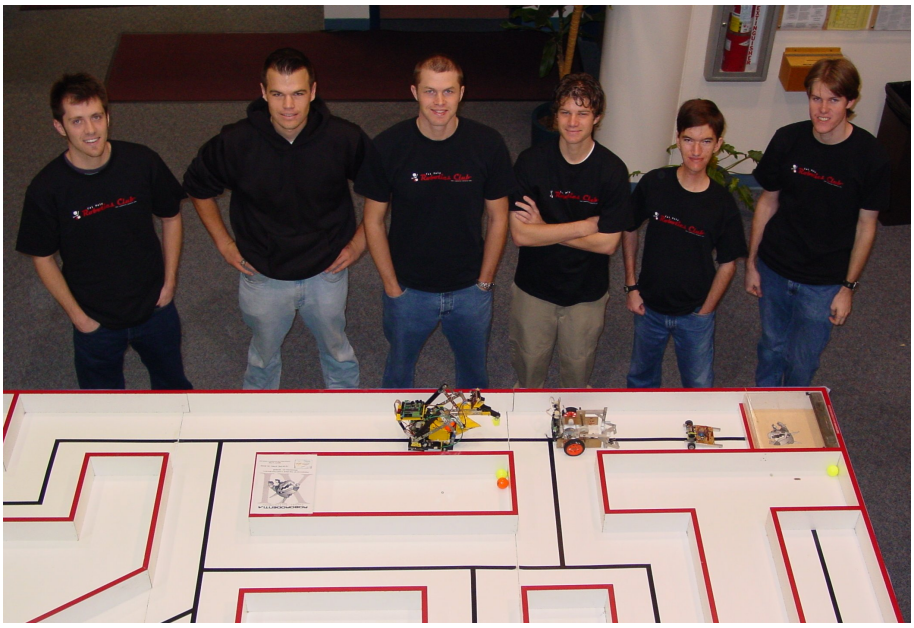


Figure 8: Team Photo. Left to right: Logan Kinde, Tyson Messori, Taylor Braun-Jones, Scott Barlow, Michael Shelley, Patrick McCarty

5.2.1 Contributions of Taylor Braun-Jones

Taylor was responsible for overall project coordination and administration including:

- Gantt chart creation and tracking
- MESFAC grant proposal
- Part ordering and budget management
- Creation and administration of the code repository

Taylor's electrical and mechanical contributions include:

- Custom-made battery packs from shrink wrap and five +1.2V rechargable NiMH AA batteries
- The design and implemenation of the bracket used to mount the CMUcam2 to the panning servo
- The concept and implementation of a detachable tethered remote for debugging and run-time parameter adjustment

The software contributions are attributed as follows:

- Code structure, high level architecture, and build system - Taylor Braun-Jones
- Path planning algorithm and implementation - Logan Kinde
- EEPROM reading and writing - Patrick McCarty
- PWM motor controller - Michael Shelly

The rest of the code and the majority of the code base was developed between Taylor and Logan together using the pair programming technique. These pieces include:

- PID line tracking
- Ball detection and seeking
- Course traversal
- Ball collection maneuvers

5.3 Acknowledgments

This project was made possible by a generous $525 grant from the Cal Poly Mechanical Engineering Student Fee Allocation Committee (MESFAC) and from various part and monetary contributions of the the Cal Poly Robotics Club.

6 Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

7 File Index

7.1 File List

Here is a list of all documented files with brief descriptions:

8 Data Structure Documentation

8.1 nodeStruct Struct Reference

Data Fields

- uint8_t **numAdjNodes**
- uint8_t **adjNodes** [MAX_ADJ_NODES]
- uint8_t **adjCosts** [MAX_ADJ_NODES]
- int8_t **adjHeadings** [MAX_ADJ_NODES]

8.1.1 Detailed Description

Definition at line 69 of file nodeList.h.

The documentation for this struct was generated from the following file:

- nodeList.h

8.2 PathList Struct Reference

Collaboration diagram for PathList:

Data Fields

- uint8_t **nodeNum**
- struct PathList * **nextNode**

8.2.1 Detailed Description

Definition at line 23 of file linkedList.h.

The documentation for this struct was generated from the following file:

- linkedList.h

8.3 searchNode Struct Reference

Data Fields

- uint8_t **parent**
- uint8_t **pathCost**
- bool **visited**

8.3.1 Detailed Description

Definition at line 38 of file updatePath.h.

The documentation for this struct was generated from the following file:

- updatePath.h

8.4 struct_EncoderState Struct Reference

Encoder state structure.

`#include <encoder.h>`

Data Fields

- uint16_t position

  position

8.4.1 Detailed Description

Encoder state structure.

Definition at line 115 of file encoder.h.

The documentation for this struct was generated from the following file:

- encoder.h

9 File Documentation

9.1 botCntrl.c File Reference

```
#include "botCntrl.h"
#include "trackLine.h"
#include "trackColor.h"
#include "junctionCode.h"
#include "updatePath.h"
#include "motorCntrl.h"
#include "camera.h"
#include "servos.h"
#include "buttons.h"
#include "nodeList.h"
#include "tetherUI.h"
#include "eeProm.h"
#include "ourLCD.h"
#include "helperFunctions.h"
#include <string.h>
```

Include dependency graph for botCntrl.c:

Macros

- #define **BEAM_IGNORE_COUNT** 6
- #define **CORRAL_COUNT** 3
- #define **LIFT_DONE_COUNT** 8

Functions

- void **runRoborodentiaCourse** (void)
- void **initBotGlobals** (void)
- bool **positionBot** (void)
- void **bbPositioning** (int8_t bbHeading, int8_t nextHeading)
- void **moveToJunction** (uint8_t numJunctions, bool justTurned)
- void **nestSequence** (void)

Variables

- uint8_t **botNode** = START_NODE
- int8_t **botHeading** = START_HEADING
- uint8_t **numUnreachedGoals** = NUM_GOALS

9.1.1 Detailed Description

Definition in file botCntrl.c.

9.2 botCntrl.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
```

```
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "botCntrl.h"
00019 #include "trackLine.h"
00020 #include "trackColor.h"
00021 #include "junctionCode.h"
00022 #include "updatePath.h"
00023 #include "motorCntrl.h"
00024 #include "camera.h"
00025 #include "servos.h"
00026 #include "buttons.h"
00027 #include "nodeList.h"
00028 #include "tetherUI.h"
00029 #include "eeProm.h"
00030 #include "ourLCD.h"
00031 #include "helperFunctions.h"
00032
00033 // avr-libc
00034 #include <string.h>
00035
00036 #define BEAM_IGNORE_COUNT      6
00037 #define CORRAL_COUNT           3
00038 #define LIFT_DONE_COUNT        8
00039
00040 // Global variables
00041 uint8_t botNode = START_NODE;
00042 int8_t botHeading = START_HEADING;
00043 uint8_t numUnreachedGoals = NUM_GOALS;
00044
00045 static bool liftDown;
00046 static uint8_t upComingBallNum;
00047
00048 static inline int8_t getNextHeading(uint8_t nextBotNode);
00049
00050 inline void runRoborodentiaCourse(void)
00051 {
00052     bool justTurned = false;
00053     bool firstRun = true;
00054
00055     updatePath();
00056
00057     // run through first leg, skipping positionBot
00058     junctionCode();
00059     moveToJunction(1, justTurned);
00060
00061 #if DEBUGGING
00062     if (lcdMode == NAV_LCD_MODE)
00063     {
00064         lcdWriteStr("                ", 0, 0);
00065         lcdWriteStr("                ", 1, 0);
00066         lcdPrintDecU08(botNode, 1, 0);
00067         lcdPrintDecS08(botHeading, 1, 3);
00068     }
00069 #endif
00070
00071     // run through arena
00072     while (pathList[pathListIndex + 1] != STOP_NODE)
00073     {
00074         junctionCode();       // ball search, bonous ball pickup, best path
       code
00075
00076         justTurned = positionBot();      // turning, preparing for linetracking
00077
00078         if (firstRun)
00079         {
00080             firstRun = false;
00081             setServo(LIFT, LIFT_OPEN); // Lower lift, on first run, b/c
       skipping seek at node 21
00082             msDelay(30);
00083             upComingBallNum = 1;
00084             liftDown = true;
00085         }
00086
```

```
00087 #if DEBUGGING
00088         if (lcdMode == NAV_LCD_MODE)
00089         {
00090             lcdPrintDecS08(botHeading, 1, 3);
00091         }
00092 #endif
00093
00094         moveToJunction(1, justTurned);   // linetracking, ground ball pickup
00095
00096 #if DEBUGGING
00097         if (lcdMode == NAV_LCD_MODE)
00098         {
00099             lcdPrintDecU08(botNode, 1, 0);
00100         }
00101 #endif
00102     }
00103
00104     if (pathList[pathListIndex + 1] == STOP_NODE)
00105     {
00106         positionBot();
00107         nestSequence();
00108     }
00109 }
00110
00111 /*
00112  *  Initializes some of bot's global variables
00113  */
00114 inline void initBotGlobals(void)
00115 {
00116     // init bot's path to INITIAL_PATH_LIST
00117     pathListIndex = 0;
00118     pathListSize = INITIAL_PATH_LIST_SIZE;
00119     // (pathList initialized in updatePath.c)
00120
00121     initGoalList();
00122     numKnownGoals = NUM_FIXED_GOALS;
00123
00124     liftDown = false;
00125     upComingBallNum = 0;
00126 }
00127
00128 /*
00129  * @brief Turn bot at junctions and, if necessary, ball nodes
00130  *
00131  * Maintains botHeading. Performs bonus ball pickup liftDown actions
00132  *
00133  * @return True when bot just turned. (Used to tell moveToJunction to
00134  * begin looking for next junction immediately.)
00135  *
00136  * PRE: camera is not streaming
00137  */
00138 inline bool positionBot(void)
00139 {
00140     bool justTurned = true;
00141
00142     int8_t nextHeading = getNextHeading(pathList[pathListIndex + 1]);
00143     int8_t bradsToTurn = nextHeading - botHeading;
00144
00145     // BB PICKUP CHECK
00146     if (botNode == BONUS_BALL_1 && isInGoalList(BONUS_BALL_1))
00147     {
00148         bbPositioning(BB1_HEADING, nextHeading);
00149         removeFromGoalList(BONUS_BALL_1);
00150     }
00151     else if (botNode == BONUS_BALL_2 && isInGoalList(BONUS_BALL_2))
00152     {
00153         bbPositioning(BB2_HEADING, nextHeading);
00154         removeFromGoalList(BONUS_BALL_2);
00155     }
00156
00157     // TURN/STRAIGHT CHECK
00158     else if (bradsToTurn != 0)
00159     {
00160         int8_t ticksToTurn;
00161         switch ((int8_t) bradsToTurn)
00162         {
00163         case -128: // U-turn
00164             if (botNode == 37)
00165             {
00166                 moveToJunction(1, false);
00167                 tickWheels(20, 20, 255);
```

```
00168                 msDelay(0x50);
00169                 moveStraight(-20, 255);
00170             }
00171             //tankTurn(245, -58);
00172             tickWheels(-29, 29, 250);
00173             tankTurn(245, -58);
00174             break;
00175         case -105: // Hard Diagonal
00176             tickWheels(28, 28, 250);     //28
00177             tractorTurn(255, -tempTweak4);
00178             tankTurn(250, -70);          //-80
00179             break;
00180         case 23: // Soft Diagonal
00181             tractorTurn(255, 23);        //23
00182             break;
00183         case -23:
00184             tractorTurn(255, -28);
00185             break;
00186         case 105:
00187             tickWheels(17, 17, 250);
00188             tankTurn(250, 80);           //80
00189             break;
00190         default:
00191             // fixed ticks forward here?
00192
00193             // convert brads to turn to ticks and turn
00194             if (bradsToTurn < 0)
00195             {
00196                 ticksToTurn = bradsToTurn + turnSubtract;
00197             } else
00198             {
00199                 ticksToTurn = bradsToTurn - turnSubtract;
00200             }
00201             tractorTurn(255, ticksToTurn);
00202             break;
00203         }
00204     }
00205     else
00206     {
00207         justTurned = false;
00208     }
00209
00210     if (botNode == SENSOR_NODE)
00211     {
00212         removeFromGoalList(SENSOR_NODE);
00213     }
00214
00215     // update botHeading
00216     botHeading = nextHeading;
00217
00218     // GB PICKUP CHECK: lower lift, if bot knows it will travel over ball
00219     upComingBallNum = getUpcomingBallNum();
00220     if (upComingBallNum != 0)
00221     {
00222         setServo(LIFT, LIFT_OPEN);
00223         msDelay(30);
00224         liftDown = true;
00225     }
00226
00227     return justTurned;
00228 }
00229
00230 /*
00231  * Returns absolute heading of next node given botNode and the next botNode.
00232  */
00233 static inline int8_t getNextHeading(uint8_t nextBotNode)
00234 {
00235     NODE nextNode;            // info about nodes adjacent to botNode
00236     int8_t nextNodeIndex;        // nextNode offset to nextBotNode
00237     int8_t nextHeading;          // absolute direction to nextBotNode
00238
00239     // get absolute direction of nextBotNode from node list
00240     getNode(botNode, &nextNode);
00241     nextNodeIndex = findValue(nextNode.adjNodes,
00242                               nextNode.numAdjNodes,
00243                               nextBotNode);
00244
00245     // get next heading or report error
00246     if (nextNodeIndex == -1)
00247     {
00248 #if DEBUGGING
```

```
00249         lcdWriteStr("pathList error  ", 0, 0);
00250 #endif
00251         brake(BOTH);
00252         while (1) ;
00253     }
00254     nextHeading = nextNode.adjHeadings[nextNodeIndex];
00255 
00256     return nextHeading;
00257 }
00258 
00259 /* Rotates bot before and after Bonus Ball grab
00260  *    bbHeading   - heading bot must have for bb pickup.
00261  *    nextHeading - heading bot must have after bb pickup
00262  */
00263 inline void bbPositioning(int8_t bbHeading, int8_t nextHeading)
00264 {
00265     // move forward (camera will be over junction at this point)
00266     // May or may not need to move foward (requires testing)
00267     //    Some are fine without foward, some seem to need it
00268     //    Right now, only -32 case moves forward
00269 
00270     // rotate by (bbHeading - botHeading)
00271     switch ((int8_t) (bbHeading - botHeading))
00272     {
00273     case 96:
00274         // example of 96 brad rotation
00275         tickWheels(28, 0, 255); // allows fluid motion (no overshoot
      correction)
00276         tankTurn(255, 58);      //58
00277         break;
00278     case -96:
00279         tickWheels(0, 28, 255); // allows fluid motion (no overshoot
      correction)
00280         tankTurn(255, -64);     //-64
00281         break;
00282     case -32:
00283         tickWheels(10, 10, 255); //10 Move bot forward a few ticks to make it
      correctly aligned
00284         tickWheels(0, 32, 255);                   //28
00285         break;
00286     default:
00287 #if DEBUGGING
00288         lcdWriteStr("ERROR:          ", 0, 0);
00289         lcdWriteStr("Turn Amt =      ", 1, 0);
00290         lcdPrintDecS08(bbHeading - botHeading, 1, 11);
00291         brake(BOTH);
00292         while (1) ;
00293 #endif
00294         break;
00295     }
00296 
00297     grabBonusBall(); // Grab the BB
00298 
00299     // Rotate by (nextHeading - bbHeading)
00300     // (This should only be 32, -32, or -96)
00301     switch ((int8_t) (nextHeading - bbHeading))
00302     {
00303     case 32:
00304         tankTurn(250, 32);
00305         break;
00306     case -32:
00307         tankTurn(250, -32);
00308         break;
00309     case -96:
00310         tankTurn(250, -90);
00311         break;
00312     default:         // Error, this should only be 32, -32, or -96
00313 #if DEBUGGING
00314         lcdWriteStr("ERROR:          ", 0, 0);
00315         lcdWriteStr("nH - bbH =      ", 1, 0);
00316         lcdPrintDecS08(bbHeading - botHeading, 1, 11);
00317         brake(BOTH);
00318         while (1) ;
00319 #endif
00320         break;
00321     }
00322 }
00323 
00324 /*
00325  * Moves to next junction in pathList.
00326  */
```

```
00327 inline void moveToJunction(uint8_t numJunctions, bool justTurned)
00328 {
00329     bool onLine = true;
00330     bool juncApproaching = false;
00331     uint8_t juncCount = 0;
00332 
00333     uint8_t ignoreJuncCount;
00334     if (!justTurned)
00335     {
00336         ignoreJuncCount = 3;
00337     } else
00338     {
00339         ignoreJuncCount = 0;
00340     }
00341 
00342     uint8_t pickingUp = false;
00343     uint8_t pickingUpCount = 0;
00344 
00345     uint8_t ignoreBreakBeamCount = BEAM_IGNORE_COUNT;
00346 
00347     trackLineInit();
00348 
00349     // Linetrack, until bot is at junction or nest.
00350     // If see ground ball, pickup it up and continue linetracking.
00351     while (onLine)
00352     {
00353         while (lineStatsProcessed) ;
00354 
00355         analyzeLineStats();
00356         adjustPWM();
00357 
00358         // CURRENT JUNCTION IGNORE
00359         if (ignoreJuncCount > 0 && junctionY == 0)
00360         {
00361             ignoreJuncCount--;
00362         }
00363 
00364         // JUNCTION CHECK
00365         if (ignoreJuncCount == 0 && junctionY != 0)
00366         {
00367             if (junctionY < turnPoint)
00368             {
00369                 juncApproaching = true;
00370             } else if (juncApproaching)
00371             {
00372                 juncApproaching = false;
00373                 juncCount++;
00374 
00375                 // set botNode to next junction in pathList
00376                 do
00377                 {
00378                     pathListIndex++;
00379                     botNode = pathList[pathListIndex];
00380                 } while (!isJunction(botNode));
00381 
00382                 // Break out of line tracking
00383                 if (juncCount >= numJunctions)
00384                 {
00385                     onLine = false;
00386                 }
00387             }
00388         }
00389 
00390         // STOP IGNORING BEAM CHECK
00391         if (liftDown && ignoreBreakBeamCount != 0)
00392         {
00393             ignoreBreakBeamCount--;
00394         }
00395 
00396         // BEGIN PICKUP CHECK
00397         if (liftDown && ignoreBreakBeamCount == 0 && BREAK_BEAM_TRIGGERED)
00398         {
00399             streamModeOff();
00400             setServo(LIFT, LIFT_CORRAL); // Perhaps raise it slowly if there
      are pick-up problems
00401             msDelay(30);
00402             trackLineInit();
00403 
00404             liftDown = false;
00405             pickingUp = true;
00406             pickingUpCount = 0;
```

```
00407         }
00408
00409         // COMPLETE/STOP LIFTING CHECK
00410         if (pickingUp)
00411         {
00412             pickingUpCount++;
00413
00414             if (pickingUpCount == CORRAL_COUNT)
00415             {
00416                 streamModeOff();
00417                 setServo(LIFT, LIFT_UP);
00418                 trackLineInit();
00419             }
00420
00421             if (pickingUpCount == LIFT_DONE_COUNT)
00422             {
00423                 pickingUp = false;
00424
00425                 // Set current botNode to node where this ball is
00426                 botNode = upComingBallNum;
00427                 removeFromGoalList(upComingBallNum);
00428
00429                 if (upComingBallNum == 1) // account for ball not found by
    camera prior to pickup
00430                 {
00431                     numKnownGoals++;
00432                 }
00433
00434                 // Find correct pathListIndex
00435                 while (botNode != pathList[pathListIndex])
00436                 {
00437                     pathListIndex++;
00438                 }
00439
00440                 streamModeOff();      // Turn off line tracking
00441                 disableServo(LIFT);
00442                 positionBot(); // In case we want to make a -128 brad turn
    after picking up ball
00443                 ignoreBreakBeamCount = BEAM_IGNORE_COUNT;
00444                 trackLineInit();      // Turn line tracking back on
00445             }
00446
00447         }
00448     }
00449
00450     streamModeOff();
00451
00452     // Make sure lift is up (in case we missed a ball or incorrectly thought
    one was there)
00453     if (liftDown)
00454     {
00455         brake(BOTH);
00456 #if DEBUGGING
00457         lcdWriteStr("No ball         ", 0, 0);
00458 #endif
00459         setServo(LIFT, LIFT_UP);    // Raise the lift
00460         msDelay(700);
00461         disableServo(LIFT);
00462         liftDown = false;
00463
00464         // correct goal state
00465         removeFromGoalList(upComingBallNum);
00466         numUnreachedGoals--;
00467         numKnownGoals--;
00468     }
00469 }
00470
00471 void nestSequence(void)
00472 {
00473     // line track, until NEST_BUTTON is pressed
00474     trackLineInit();
00475
00476     while (!justPressed(NEST_BUTTON))
00477     {
00478         if (!lineStatsProcessed)
00479         {
00480             analyzeLineStats();
00481             adjustPWM();
00482         }
00483
00484         debounceButtons();
```

```
00485     }
00486
00487     brake(BOTH);
00488     streamModeOff();
00489     setServo(LIFT, LIFT_UP);       // Turn lift on
00490     msDelay(300);
00491
00492     // Open door, back up, close door
00493     setServo(DOOR, DOOR_OPEN);
00494     moveStraight(-1, 255);              // Back up to take pressure off button
00495     brake(BOTH);
00496     //myDelay(25);                     // Let balls roll out
00497     msDelay(3000);
00498     setServo(DOOR, DOOR_CLOSED); // Leaves door closed, so lift and door don't
       colide on power up.
00499     //myDelay(10);                     // Wait for door to close
00500     msDelay(1000);
00501
00502     // Disable all servos
00503     disableServo(PAN);
00504     disableServo(TILT);
00505     disableServo(BOOM);
00506     disableServo(LIFT);
00507     disableServo(DOOR);
00508 }
```

9.3 botCntrl.h File Reference

High-level logic controlling Caddy's actions.

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for botCntrl.h: This graph shows which files directly or indirectly include this file:

Functions

- void **runRoborodentiaCourse** (void)
- void **initBotGlobals** (void)
- bool **positionBot** (void)
- void **moveToJunction** (uint8_t numJunctions, bool justTurned)
- void **bbSequence** (void)
- void **nestSequence** (void)
- void **bbPositioning** (int8_t bbHeading, int8_t nextHeading)

Variables

- uint8_t **botNode**
- int8_t **botHeading**
- uint8_t **numUnreachedGoals**

9.3.1 Detailed Description

High-level logic controlling Caddy's actions.

See Also

Problem Summary

Definition in file botCntrl.h.

9.4 botCntrl.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00023 #ifndef BOTCNTRL_H_
00024 #define BOTCNTRL_H_
00025
00026 // avr-libc
00027 #include <stdint.h>
00028 #include <stdbool.h>
00029
00030 // Global variables
00031 extern uint8_t botNode;
00032 extern int8_t botHeading;
00033 extern uint8_t numUnreachedGoals;
00034
00035 inline void runRoborodentiaCourse(void);
00036 inline void initBotGlobals(void);
00037 inline bool positionBot(void);
00038 inline void moveToJunction(uint8_t numJunctions, bool justTurned);
00039 void bbSequence(void);
00040 void nestSequence(void);
00041 inline void bbPositioning(int8_t bbHeading, int8_t nextHeading);
00042
00043 #endif // #ifndef BOTCNTRL_H_
```

9.5 buttons.c File Reference

```
#include "buttons.h"
#include "avrlibdefs.h"
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for buttons.c:

Macros

- #define **DEBOUNCE_COUNT** 3

Functions

- void **waitFor** (uint8_t button)
- bool justPressed (uint8_t button)
- bool justReleased (uint8_t button)
- void debounceButtons (void)

  Maintains wasEvent[] and toggles isDown[].
- bool isPressed (uint8_t button)
- bool **bothRightButtonsPressed** (void)
- bool **bothLeftButtonsPressed** (void)

9.5.1 Detailed Description

Definition in file buttons.c.

9.5.2 Function Documentation

9.5.2.1 bool isPressed (uint8_t *button*) [inline]

Returns

true when button is currently down (does no debouncing!)

Definition at line 114 of file buttons.c.

9.5.2.2 bool justPressed (uint8_t *button*) [inline]

Returns

true when confirmed rising edge at last debouncing.

Definition at line 50 of file buttons.c.

9.5.2.3 bool justReleased (uint8_t *button*) [inline]

Returns

true when confirmed falling edge at last debouncing.

Definition at line 58 of file buttons.c.

9.6 buttons.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "buttons.h"
00019
00020 // AVRLIB
00021 #include "avrlibdefs.h"
00022
00023 // avr-libc
00024 #include <stdint.h>
00025 #include <stdbool.h>
00026
00027 #define DEBOUNCE_COUNT   3   // should be equal to 2 or greater
00028
00029 static bool isDown[NUM_BUTTONS]   = { false, false, false,
00030                                       false, false, false };
00031 static bool wasEvent[NUM_BUTTONS] = { false, false, false,
00032                                       false, false, false };
00033 static uint8_t upCount[NUM_BUTTONS] = { DEBOUNCE_COUNT, DEBOUNCE_COUNT,
00034                                         DEBOUNCE_COUNT, DEBOUNCE_COUNT,
00035                                         DEBOUNCE_COUNT, DEBOUNCE_COUNT };
```

```
00036 static uint8_t downCount[NUM_BUTTONS] = { 0, 0, 0, 0, 0, 0 };
00037
00038 void waitFor(uint8_t button)
00039 {
00040     debounceButtons();
00041     while (!justReleased(button))
00042     {
00043         debounceButtons();
00044     }
00045 }
00046
00050 inline bool justPressed(uint8_t button)
00051 {
00052     return wasEvent[button] && isDown[button];
00053 }
00054
00058 inline bool justReleased(uint8_t button)
00059 {
00060     return wasEvent[button] && !isDown[button];
00061 }
00062
00066 void debounceButtons(void)
00067 {
00068     uint8_t button;
00069     for (button = 0; button < NUM_BUTTONS; button++)
00070     {
00071         // count times buttons have been consecutively up/down (upto
     DEBOUNCE_COUNT).
00072         if (isPressed(button))
00073         {
00074             downCount[button] = MIN(downCount[button]+1, DEBOUNCE_COUNT);
00075             upCount[button] = 0;
00076         }
00077         else
00078         {
00079             upCount[button] = MIN(upCount[button]+1, DEBOUNCE_COUNT);
00080             downCount[button] = 0;
00081         }
00082
00083         // check for confirmed up/down event
00084         if (isDown[button])
00085         {
00086             if (upCount[button] >= DEBOUNCE_COUNT)
00087             {
00088                 isDown[button] = false;
00089                 wasEvent[button] = true;
00090             }
00091             else
00092             {
00093                 wasEvent[button] = false;
00094             }
00095         }
00096         else
00097         {
00098             if (downCount[button] >= DEBOUNCE_COUNT)
00099             {
00100                 isDown[button] = true;
00101                 wasEvent[button] = true;
00102             }
00103             else
00104             {
00105                 wasEvent[button] = false;
00106             }
00107         }
00108     }
00109 }
00110
00114 inline bool isPressed(uint8_t button)
00115 {
00116     switch (button)
00117     {
00118     case RED_BUTTON:       return RED_BUTTON_DOWN;
00119     case L_UP_BUTTON:      return L_UP_BUTTON_DOWN;
00120     case L_DOWN_BUTTON:    return L_DOWN_BUTTON_DOWN;
00121     case R_UP_BUTTON:      return R_UP_BUTTON_DOWN;
00122     case R_DOWN_BUTTON:    return R_DOWN_BUTTON_DOWN;
00123     case NEST_BUTTON:      return NEST_BUTTON_DOWN;
00124     default:               break;
00125     }
00126
00127     return false;
```

```
00128 }
00129
00130 inline bool bothRightButtonsPressed(void)
00131 {
00132     return (justPressed(R_UP_BUTTON) && justPressed(
      R_DOWN_BUTTON)) ||
00133            (justPressed(R_UP_BUTTON) && isDown[R_DOWN_BUTTON])
      ||
00134            (justPressed(R_DOWN_BUTTON) && isDown[R_UP_BUTTON]);
00135
00136 }
00137
00138 inline bool bothLeftButtonsPressed(void)
00139 {
00140     return (justPressed(L_UP_BUTTON) && justPressed(
      L_DOWN_BUTTON)) ||
00141            (justPressed(L_UP_BUTTON) && isDown[L_DOWN_BUTTON])
      ||
00142            (justPressed(L_DOWN_BUTTON) && isDown[L_UP_BUTTON]);
00143
00144 }
```

9.7 buttons.h File Reference

Button debouncing, start bot logic.

```
#include <avr/io.h>
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for buttons.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **RED_BUTTON** 0
- #define **L_UP_BUTTON** 1
- #define **L_DOWN_BUTTON** 2
- #define **R_UP_BUTTON** 3
- #define **R_DOWN_BUTTON** 4
- #define **NEST_BUTTON** 5
- #define **NUM_BUTTONS** 6
- #define **RED_BUTTON_DOWN** bit_is_clear(PIND,6)
- #define **L_UP_BUTTON_DOWN** bit_is_clear(PINA,0)
- #define **L_DOWN_BUTTON_DOWN** bit_is_clear(PINA,1)
- #define **R_UP_BUTTON_DOWN** bit_is_clear(PINA,2)
- #define **R_DOWN_BUTTON_DOWN** bit_is_clear(PINA,3)
- #define **NEST_BUTTON_DOWN** bit_is_clear(PINB,0)
- #define **BREAK_BEAM_TRIGGERED** bit_is_set(PINB,1)

Functions

- void **initButtons** (void)
- void **waitFor** (uint8_t button)
- bool justPressed (uint8_t button)
- bool justReleased (uint8_t button)
- void debounceButtons (void)

  Maintains wasEvent[] and toggles isDown[].
- bool isPressed (uint8_t button)
- bool **bothRightButtonsPressed** (void)
- bool **bothLeftButtonsPressed** (void)

9.7.1 Detailed Description

Button debouncing, start bot logic.

Definition in file buttons.h.

9.7.2 Function Documentation

9.7.2.1 bool isPressed (uint8_t *button*) [inline]

Returns

true when button is currently down (does no debouncing!)

Definition at line 114 of file buttons.c.

9.7.2.2 bool justPressed (uint8_t *button*) [inline]

Returns

true when confirmed rising edge at last debouncing.

Definition at line 50 of file buttons.c.

9.7.2.3 bool justReleased (uint8_t *button*) [inline]

Returns

true when confirmed falling edge at last debouncing.

Definition at line 58 of file buttons.c.

9.8 buttons.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00021 #ifndef BUTTONS_H_
00022 #define BUTTONS_H_
00023
00024 #include <avr/io.h>
00025 #include <stdint.h>
00026 #include <stdbool.h>
00027
00028 #define RED_BUTTON       0
00029 #define L_UP_BUTTON      1
00030 #define L_DOWN_BUTTON    2
00031 #define R_UP_BUTTON      3
00032 #define R_DOWN_BUTTON    4
00033 #define NEST_BUTTON      5
00034 #define NUM_BUTTONS      6    // change isPressed(uint8_t button) when adding a
       button
00035
```

```
00036 #define RED_BUTTON_DOWN       bit_is_clear(PIND,6)
00037 #define L_UP_BUTTON_DOWN      bit_is_clear(PINA,0)
00038 #define L_DOWN_BUTTON_DOWN    bit_is_clear(PINA,1)
00039 #define R_UP_BUTTON_DOWN      bit_is_clear(PINA,2)
00040 #define R_DOWN_BUTTON_DOWN    bit_is_clear(PINA,3)
00041 #define NEST_BUTTON_DOWN      bit_is_clear(PINB,0)
00042
00043 #define BREAK_BEAM_TRIGGERED  bit_is_set(PINB,1)
00044
00045 void initButtons(void);
00046 void waitFor(uint8_t button);
00047 inline bool justPressed(uint8_t button);
00048 inline bool justReleased(uint8_t button);
00049 void debounceButtons(void);
00050 inline bool isPressed(uint8_t button);
00051 inline bool bothRightButtonsPressed(void);
00052 inline bool bothLeftButtonsPressed(void);
00053
00054 #endif // #ifndef BUTTONS_H_
```

9.9 caddy.c File Reference

Caddy's main loop and Atmel initialization.

```
#include "botCntrl.h"
#include "motorCntrl.h"
#include "camera.h"
#include "servos.h"
#include "encoder.h"
#include "buttons.h"
#include "eeProm.h"
#include "helperFunctions.h"
#include "ourLCD.h"
#include <avr/io.h>
```

Include dependency graph for caddy.c:

Macros

- #define START_DELAY 5

Functions

- int main (void)

  Caddy's power-on entry function.

9.9.1 Detailed Description

Caddy's main loop and Atmel initialization.

Definition in file caddy.c.

9.9.2 Macro Definition Documentation

9.9.2.1 #define START_DELAY 5

Short delay wait for finger to be fully removed from start button (or tether cable to be disconnected)

Definition at line 38 of file caddy.c.

9.10 caddy.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00021 #include "botCntrl.h"
00022 #include "motorCntrl.h"
00023 #include "camera.h"
00024 #include "servos.h"
00025 #include "encoder.h"
00026 #include "buttons.h"
00027 #include "eeProm.h"
00028 #include "helperFunctions.h"
00029 #include "ourLCD.h"
00030
00031 // avr-libc
00032 #include <avr/io.h>
00033
00038 #define START_DELAY 5
00039
00043 static inline void initAtmel(void)
00044 {
00045     /*
00046      * Initialize Timer
00047      */
00048     timerInit();
00049
00050 #if DEBUGGING
00051     /*
00052      * Initialize LCD
00053      */
00054     lcdInit();
00055     lcdWriteStr("Init:           ", 0, 0);
00056     lcdWriteStr("                ", 1, 0);
00057 #endif
00058
00059     /*
00060      * Initialize UART for CMUcam communication
00061      */
00062     cmuCamInit();
00063
00064     /*
00065      * Initialize PWM motor control
00066      */
00067     outb(DDRD, 0xff);
00068     timer1PWMInit(8);
00069     neutral();
00070     enableMotors();
00071
00072     /*
00073      * Set data direction registers
00074      */
00075     outb(DDRA, 0xF0); // Motor control and up/down buttons
00076     cbi(DDRD, 6);     // red button
00077     cbi(DDRB, 0);     // nest button
00078     cbi(DDRB, 1);     // break beam
00079
00080     /*
00081      * Apply internal pull-up resister to certain digital inputs
00082      */
00083     sbi(PORTB, 0); // internal pull-up for PINB0
00084     sbi(PORTA, 3); // internal pull-up for PINA3
00085     sbi(PORTA, 2); // internal pull-up for PINA2
00086     sbi(PORTA, 1); // internal pull-up for PINA1
00087     sbi(PORTA, 0); // internal pull-up for PINA0
00088
00089     /*
```

```
00090      * Initialize quadrature wheel encoders
00091      */
00092     cbi(DDRD, 2);
00093     cbi(DDRD, 3);
00094     encoderInit();
00095 }
00096
00100 int main(void)
00101 {
00102     initAtmel();
00103     loadTweakValues();
00104     initBotGlobals();
00105     resetCamera();
00106     moveServosToStart();
00107     cameraWhiteBalance();
00108
00109 #if DEBUGGING
00110     runTetherUI();
00111     myDelay(START_DELAY);
00112     runTest();
00113 #else
00114     waitFor(RED_BUTTON);
00115     myDelay(START_DELAY);
00116     runRoborodentiaCourse();
00117 #endif
00118
00119     brake(BOTH);
00120 #if DEBUGGING
00121     lcdWriteStr("Done            ", 0, 0);
00122     lcdWriteStr("                ", 1, 0);
00123 #endif
00124
00125     return 0;
00126 }
```

9.11 camera.c File Reference

```
#include "camera.h"
#include "trackColor.h"
#include "trackLine.h"
#include "helperFunctions.h"
#include "ourLCD.h"
#include "rprintf.h"
#include "uart.h"
#include <stdbool.h>
```

Include dependency graph for camera.c:

Macros

- #define **CMU_BAUD** 38400

Functions

- void **packetRcv** (uint8_t c)
- void **lineMode2Rcv** (uint8_t c)
- void **trackColorRcv** (uint8_t c)
- void cmuCamInit (void)

  Initialize the UART for communicating with the CMUcam.
- void cameraWhiteBalance ()

  Optimize the white balance for current conditions.
- void **resetCamera** (void)
- void **streamModeOff** (void)

- void setVirtualWindow (uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2)

  Constrain field of view used for subsequent image processing commands.

9.11.1 Detailed Description

Definition in file camera.c.

9.11.2 Function Documentation

9.11.2.1 void cameraWhiteBalance (void) [inline]

Optimize the white balance for current conditions.

Turn auto-white balance on, give it time to settle, then turn auto-white balance off.

See Also

CMUcam2 manual p.31

Definition at line 48 of file camera.c.

9.11.2.2 void setVirtualWindow (uint8_t *x1*, uint8_t *y1*, uint8_t *x2*, uint8_t *y2*) [inline]

Constrain field of view used for subsequent image processing commands.

See Also

CMUcam2 manual p.55

Definition at line 142 of file camera.c.

9.12 camera.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "camera.h"
00019 #include "trackColor.h"
00020 #include "trackLine.h"
00021 #include "helperFunctions.h"
00022 #include "ourLCD.h"
00023
00024 // AVRLIB
00025 #include "rprintf.h"
00026 #include "uart.h"
00027
00028 // avr-libc
00029 #include <stdbool.h>
00030
00031 #define CMU_BAUD 38400
```

```
00032
00033 static uint8_t mode;
00034 static uint16_t byteNum;
00035
00036 void packetRcv( uint8_t c );
00037 inline void lineMode2Rcv( uint8_t c );
00038 inline void trackColorRcv( uint8_t c );
00039
00040 inline void cmuCamInit(void)
00041 {
00042     uartInit();
00043     uartSetBaudRate(CMU_BAUD);
00044     uartSetRxHandler(packetRcv);
00045     rprintfInit(uartSendByte);
00046 }
00047
00048 inline void cameraWhiteBalance()
00049 {
00050     // turn auto white balance on
00051 #if DEBUGGING
00052     lcdWriteStr("white Bal ", 0, 6);
00053 #endif
00054     rprintf("CR 18 44\r");
00055     myDelay(200);
00056     // turn auto white balance off
00057     rprintf("CR 18 40\r");
00058 }
00059
00060 inline void resetCamera( void )
00061 {
00062     mode = NEW_PACKET;
00063     byteNum = 0;
00064
00065     rprintf("RM 3\r");
00066 }
00067
00068 void packetRcv(uint8_t c)
00069 {
00070     if (c == 0xff)
00071     {
00072         mode = NEW_PACKET;
00073         byteNum = 0;
00074     }
00075     else
00076     {
00077         switch (mode)
00078         {
00079         case NEW_PACKET:
00080             switch (c)
00081             {
00082             case 0xfe:
00083                 mode = FE_RCV;
00084                 break;
00085             case 'T':
00086                 mode = T_RCV;
00087                 break;
00088             default:
00089                 break;
00090             }
00091             break;
00092         case FE_RCV:
00093             lineMode2Rcv(c);
00094             if (c == 0xfd)
00095             {
00096                 mode = NEW_PACKET;
00097             }
00098             break;
00099         case T_RCV:
00100             trackColorRcv(c);
00101             break;
00102         }
00103     }
00104 }
00105
00106
00107 inline void lineMode2Rcv(uint8_t c)
00108 {
00109     if (c == 0xfd)
00110     {
00111         lineStatsProcessed = false;
00112         byteNum = 0;
```

```
00113     }
00114     else
00115     {
00116         lineStats[(byteNum - 1) / LINE_STATS_COLS]
00117                 [(byteNum - 1) % LINE_STATS_COLS] = c;
00118         byteNum++;
00119     }
00120 }
00121
00122
00123 inline void trackColorRcv(uint8_t c)
00124 {
00125     lineStats[0][byteNum] = c;
00126     byteNum++;
00127
00128     if (byteNum >= NUM_COLOR_STATS)
00129     {
00130         colorStatsProcessed = false;
00131     }
00132 }
00133
00134
00135 inline void streamModeOff( void )
00136 {
00137     rprintf("\r\r"); // add an extra return as recommended by CMUcam manual
00138     msDelay(32);     // wait for streaming to stop ( 16ms delay ok )
00139 }
00140
00141
00142 inline void setVirtualWindow(uint8_t x1, uint8_t y1, uint8_t x2
      , uint8_t y2)
00143 {
00144     rprintf("VW %d %d %d %d\r", x1, y1, x2, y2);
00145 }
```

9.13 camera.h File Reference

`#include <stdint.h>`
Include dependency graph for camera.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **NEW_PACKET** 0
- #define **FE_RCV** 1
- #define **T_RCV** 2
- #define **hiResMode**() rprintf("HR 1\r")
- #define **lowResMode**() rprintf("HR 0\r")

Functions

- void cmuCamInit (void)

  Initialize the UART for communicating with the CMUcam.
- void cameraWhiteBalance (void)

  Optimize the white balance for current conditions.
- void **resetCamera** (void)
- void **streamModeOff** (void)
- void setVirtualWindow (uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2)

  Constrain field of view used for subsequent image processing commands.

9.13.1 Detailed Description

Definition in file camera.h.

9.13.2 Function Documentation

9.13.2.1 void cameraWhiteBalance (void) [inline]

Optimize the white balance for current conditions.

Turn auto-white balance on, give it time to settle, then turn auto-white balance off.

See Also

CMUcam2 manual p.31

Definition at line 48 of file camera.c.

9.13.2.2 void setVirtualWindow (uint8_t *x1*, uint8_t *y1*, uint8_t *x2*, uint8_t *y2*) [inline]

Constrain field of view used for subsequent image processing commands.

See Also

CMUcam2 manual p.55

Definition at line 142 of file camera.c.

9.14 camera.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #ifndef CAMERA_H_
00019 #define CAMERA_H_
00020
00021 #include <stdint.h>
00022
00023 // Packet types
00024 #define NEW_PACKET   0
00025 #define FE_RCV       1
00026 #define T_RCV        2
00027
00028 #define hiResMode()  rprintf("HR 1\r")
00029 #define lowResMode() rprintf("HR 0\r")
00030
00034 inline void cmuCamInit(void);
00035
00044 inline void cameraWhiteBalance( void );
00045
00046 inline void resetCamera( void );
00047 inline void streamModeOff( void );
00048
00054 inline void setVirtualWindow(uint8_t x1, uint8_t y1, uint8_t x2
      , uint8_t y2);
00055
00056 #endif // #ifndef CAMERA_H_
```

9.15 eeProm.c File Reference

```
#include "eeProm.h"
#include <avr/io.h>
#include <avr/interrupt.h>
```

Include dependency graph for eeProm.c:

Functions

- void **loadTweakValues** (void)
- void **storeTweakValues** (void)
- uint8_t **EEPROM_read** (unsigned int uiAddress)
- void **EEPROM_write** (unsigned int uiAddress, uint8_t ucData)

Variables

- uint8_t **l_base**
- uint8_t **r_base**
- uint16_t **slopeCoef**
- uint16_t **offCoef**
- uint8_t **dampCoef**
- uint8_t **lineCenter**
- uint8_t **turnPoint**
- uint8_t **turnSubtract**
- int8_t **panOffset**
- int8_t **tiltOffset**
- uint16_t **tractorOvershootDelay**
- uint8_t **tempTweak1**
- int8_t **tempTweak2**
- uint16_t **tempTweak3**
- uint16_t **tempTweak4**
- uint8_t **lcdMode**
- uint8_t **testMode**

9.15.1 Detailed Description

Definition in file eeProm.c.

9.16 eeProm.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
```

```
00016  */
00018 #include "eeProm.h"
00019
00020 // avr-libc
00021 #include <avr/io.h>
00022 #include <avr/interrupt.h>
00023
00024 // Global variables - Runtime configurable parameters
00025 uint8_t l_base;
00026 uint8_t r_base;
00027 uint16_t slopeCoef;
00028 uint16_t offCoef;
00029 uint8_t dampCoef;
00030 uint8_t lineCenter;
00031 uint8_t turnPoint;
00032 uint8_t turnSubtract;
00033 int8_t panOffset;
00034 int8_t tiltOffset;
00035 uint16_t tractorOvershootDelay;
00036 uint8_t tempTweak1;
00037 int8_t tempTweak2;
00038 uint16_t tempTweak3;
00039 uint16_t tempTweak4;
00040
00041 uint8_t lcdMode;  // <- need debugger menu for this, remove old init/toggling,
       and save in eeProm
00042 uint8_t testMode; // <- need to save this in eeProm
00043
00044 // Initializes constants that can be tweaked by debugger
00045 inline void loadTweakValues(void)
00046 {
00047     cli(); // disable all interrupts
00048
00049     // EEPROM Reads
00050     l_base                = EEPROM_read(EE_ADDR_LEFT_BASE);
00051     r_base                = EEPROM_read(EE_ADDR_RIGHT_BASE);
00052     slopeCoef             = (EEPROM_read(EE_ADDR_SLOPE_COEF) << 8) +
00053                              EEPROM_read(EE_ADDR_SLOPE_COEF + 1);
00054     offCoef               = (EEPROM_read(EE_ADDR_OFF_COEF) << 8) +
00055                              EEPROM_read(EE_ADDR_OFF_COEF + 1);
00056     dampCoef              = EEPROM_read(EE_ADDR_DAMP_COEF);
00057     lineCenter            = EEPROM_read(EE_ADDR_LINE_X_CENTER);
00058     turnPoint             = EEPROM_read(EE_ADDR_TURN_POINT);
00059     turnSubtract          = EEPROM_read(EE_ADDR_TURN_SUBTRACT);
00060     panOffset             = EEPROM_read(EE_ADDR_PAN_OFFSET);
00061     tiltOffset            = EEPROM_read(EE_ADDR_TILT_OFFSET);
00062     tractorOvershootDelay = (EEPROM_read(EE_ADDR_TRACTOR_OVERSHOOT_DELAY) << 8)
       +
00063                              EEPROM_read(EE_ADDR_TRACTOR_OVERSHOOT_DELAY + 1);
00064     testMode              = EEPROM_read(EE_ADDR_TEST_MODE);
00065     tempTweak1            = EEPROM_read(EE_ADDR_TEMP_TWEAK1);
00066     tempTweak2            = EEPROM_read(EE_ADDR_TEMP_TWEAK2);
00067     tempTweak3            = (EEPROM_read(EE_ADDR_TEMP_TWEAK3) << 8) +
00068                              EEPROM_read(EE_ADDR_TEMP_TWEAK3 + 1);
00069     tempTweak4            = (EEPROM_read(EE_ADDR_TEMP_TWEAK4) << 8) +
00070                              EEPROM_read(EE_ADDR_TEMP_TWEAK4 + 1);
00071
00072     sei(); // enable all interrupts
00073 }
00074
00075 // Saves constants after they have been changed by the debugger
00076 inline void storeTweakValues(void)
00077 {
00078     cli(); // disable all interrupts
00079
00080     // EEPROM writes
00081     EEPROM_write(EE_ADDR_LEFT_BASE, l_base);
00082     EEPROM_write(EE_ADDR_RIGHT_BASE, r_base);
00083     EEPROM_write(EE_ADDR_SLOPE_COEF, slopeCoef >> 8);
00084     EEPROM_write(EE_ADDR_SLOPE_COEF + 1, slopeCoef);
00085     EEPROM_write(EE_ADDR_OFF_COEF, offCoef >> 8);
00086     EEPROM_write(EE_ADDR_OFF_COEF + 1, offCoef);
00087     EEPROM_write(EE_ADDR_DAMP_COEF, dampCoef);
00088     EEPROM_write(EE_ADDR_LINE_X_CENTER, lineCenter);
00089     EEPROM_write(EE_ADDR_TURN_POINT, turnPoint);
00090     EEPROM_write(EE_ADDR_TURN_SUBTRACT, turnSubtract);
00091     EEPROM_write(EE_ADDR_PAN_OFFSET, panOffset);
00092     EEPROM_write(EE_ADDR_TILT_OFFSET, tiltOffset);
00093     EEPROM_write(EE_ADDR_TRACTOR_OVERSHOOT_DELAY, tractorOvershootDelay >> 8);
00094     EEPROM_write(EE_ADDR_TRACTOR_OVERSHOOT_DELAY + 1, tractorOvershootDelay);
00095     EEPROM_write(EE_ADDR_TEST_MODE, testMode);
```

```
00096     EEPROM_write(EE_ADDR_TEMP_TWEAK1, tempTweak1);
00097     EEPROM_write(EE_ADDR_TEMP_TWEAK2, tempTweak2);
00098     EEPROM_write(EE_ADDR_TEMP_TWEAK3, tempTweak3 >> 8);
00099     EEPROM_write(EE_ADDR_TEMP_TWEAK3 + 1, tempTweak3);
00100     EEPROM_write(EE_ADDR_TEMP_TWEAK4, tempTweak4 >> 8);
00101     EEPROM_write(EE_ADDR_TEMP_TWEAK4 + 1, tempTweak4);
00102
00103     sei(); // enable all interrupts
00104 }
00105
00106 uint8_t EEPROM_read(unsigned int uiAddress)
00107 {
00108     // Wait for completion of previous write
00109     while (EECR & (1 << EEWE)) ;
00110     // Set up address register
00111     EEAR = uiAddress;
00112     // Start eeprom read by writing EERE
00113     EECR |= (1 << EERE);
00114     // Return data from data register
00115     return EEDR;
00116 }
00117
00118 void EEPROM_write(unsigned int uiAddress, uint8_t ucData)
00119 {
00120     // Wait for completion of previous write
00121     while (EECR & (1 << EEWE)) ;
00122     // Set up address and data registers
00123     EEAR = uiAddress;
00124     EEDR = ucData;
00125     // Write logical one to EEMWE
00126     EECR |= (1 << EEMWE);
00127     // Start eeprom write by setting EEWE
00128     EECR |= (1 << EEWE);
00129     // EEMWE and EEWE are automatically cleared back to 0 by hardware
00130 }
```

9.17 eeProm.h File Reference

Loading and store "tweak values" into eeProm.

`#include <stdint.h>`
Include dependency graph for eeProm.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **EE_ADDR_LEFT_BASE** 0x50
- #define **EE_ADDR_RIGHT_BASE** 0x51
- #define **EE_ADDR_SLOPE_COEF** 0x52
- #define **EE_ADDR_OFF_COEF** 0x54
- #define **EE_ADDR_DAMP_COEF** 0x56
- #define **EE_ADDR_LINE_X_CENTER** 0x57
- #define **EE_ADDR_TURN_POINT** 0x58
- #define **EE_ADDR_TURN_SUBTRACT** 0x59
- #define **EE_ADDR_PAN_OFFSET** 0x5A
- #define **EE_ADDR_TILT_OFFSET** 0x5B
- #define **EE_ADDR_TRACTOR_OVERSHOOT_DELAY** 0x5C
- #define **EE_ADDR_TEST_MODE** 0x5E
- #define **EE_ADDR_TEMP_TWEAK1** 0x5F
- #define **EE_ADDR_TEMP_TWEAK2** 0x60
- #define **EE_ADDR_TEMP_TWEAK3** 0x61
- #define **EE_ADDR_TEMP_TWEAK4** 0x63
- #define **BASE_MIN** 0x60
- #define **BASE_MAX** 0xFF

- #define **BALL_CHECK_RATIO** 16
- #define **PICK_UP_POINT** 0x16

Functions

- void **loadTweakValues** (void)
- void **storeTweakValues** (void)
- uint8_t **EEPROM_read** (unsigned int uiAddress)
- void **EEPROM_write** (unsigned int uiAddress, uint8_t ucData)

Variables

- uint8_t **l_base**
- uint8_t **r_base**
- uint16_t **slopeCoef**
- uint16_t **offCoef**
- uint8_t **dampCoef**
- uint8_t **lineCenter**
- uint8_t **turnPoint**
- uint8_t **turnSubtract**
- int8_t **panOffset**
- int8_t **tiltOffset**
- uint16_t **tractorOvershootDelay**
- uint8_t **tempTweak1**
- int8_t **tempTweak2**
- uint16_t **tempTweak3**
- uint16_t **tempTweak4**
- uint8_t **lcdMode**
- uint8_t **testMode**

9.17.1 Detailed Description

Loading and store "tweak values" into eeProm. Tweak values are runtime configurable parameters that can be adjusted e.g. with the tether UI and saved persistently in EEPROM.

Definition in file eeProm.h.

9.18 eeProm.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00024 #ifndef EEPROM_H_
```

```
00025 #define EEPROM_H_
00026
00027 #include <stdint.h>
00028
00029 //Locations in EEPROM
00030 #define EE_ADDR_LEFT_BASE           0x50
00031 #define EE_ADDR_RIGHT_BASE          0x51
00032 #define EE_ADDR_SLOPE_COEF          0x52 //uint16_t
00033 #define EE_ADDR_OFF_COEF            0x54 //uint16_t
00034 #define EE_ADDR_DAMP_COEF           0x56
00035 #define EE_ADDR_LINE_X_CENTER       0x57
00036 #define EE_ADDR_TURN_POINT          0x58
00037 #define EE_ADDR_TURN_SUBTRACT       0x59
00038 #define EE_ADDR_PAN_OFFSET          0x5A
00039 #define EE_ADDR_TILT_OFFSET         0x5B
00040 #define EE_ADDR_TRACTOR_OVERSHOOT_DELAY 0x5C //uint16_t
00041 #define EE_ADDR_TEST_MODE           0x5E
00042 #define EE_ADDR_TEMP_TWEAK1         0x5F
00043 #define EE_ADDR_TEMP_TWEAK2         0x60
00044 #define EE_ADDR_TEMP_TWEAK3         0x61 //uint16_t
00045 #define EE_ADDR_TEMP_TWEAK4         0x63 //uint16_t
00046 //next address 0x65
00047
00048 /*
00049 // Current values - Bigger motors at 6 Volts
00050 #define INIT_LEFT_BASE_SPEED        0xF7
00051 #define INIT_RIGHT_BASE_SPEED       0xF0
00052 #define INIT_SLOPE_COEF             0x0110   // <--- 0x110 could be tweaked
00053 #define INIT_OFF_COEF               0x0001
00054 #define INIT_DAMP_COEF              0x01     // <--- 0x01 could be tweaked
00055 #define INIT_LINE_X_CENTER          0x25
00056 #define INIT_TURN_POINT             0x15     // Turn values
00057 #define INIT_TURN_SUBTRACT          0x0A
00058 pan offset 0x05
00059 tilt offset 0xE7
00060
00061 #define TRACTOR_OVERSHOOT_DELAY     5000
00062 */
00063
00064 #define BASE_MIN                    0x60     // <--- also worked at 0xB0
00065 #define BASE_MAX                    0xFF
00066
00067 // Pickup values
00068 #define BALL_CHECK_RATIO            16
00069 #define PICK_UP_POINT               0x16
00070
00071
00072 /*
00073 // Old values - 6V Solarbotics before damping fix
00074 #define LEFT_BASE_SPEED           0x8C
00075 #define RIGHT_BASE_SPEED          0xC2
00076 #define BASE_MIN                  100
00077 #define BASE_MAX                  255
00078 #define OFF_COEF                  0x2
00079 #define SLOPE_COEF                0x100
00080 #define DAMP_COEF                 0x14
00081 */
00082
00083 // Global variables - Runtime configurable parameters
00084 extern uint8_t l_base;
00085 extern uint8_t r_base;
00086 extern uint16_t slopeCoef;
00087 extern uint16_t offCoef;
00088 extern uint8_t dampCoef;
00089 extern uint8_t lineCenter;
00090 extern uint8_t turnPoint;
00091 extern uint8_t turnSubtract;
00092 extern int8_t panOffset;
00093 extern int8_t tiltOffset;
00094 extern uint16_t tractorOvershootDelay;
00095 extern uint8_t tempTweak1;
00096 extern int8_t tempTweak2;
00097 extern uint16_t tempTweak3;
00098 extern uint16_t tempTweak4;
00099
00100 extern uint8_t lcdMode;
00101 extern uint8_t testMode;
00102
00103 inline void loadTweakValues( void );
00104 inline void storeTweakValues( void );
00105 uint8_t EEPROM_read(unsigned int uiAddress);
```

```
00106 void EEPROM_write(unsigned int uiAddress, uint8_t ucData);
00107
00108 #endif // #ifndef EEPROM_H_
```

9.19 encoder.c File Reference

Quadrature Encoder reader/driver.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "global.h"
#include "encoder.h"
```

Include dependency graph for encoder.c:

Functions

- void encoderInit (void)

  encoderInit() initializes hardware and encoder position readings
- uint16_t encoderGetPosition (uint8_t encoderNum)

  encoderGetPosition() reads the current position of the encoder
- void encoderSetPosition (uint8_t encoderNum, uint16_t position)

  encoderSetPosition() sets the current position of the encoder
- SIGNAL (ENC0_SIGNAL)

  Encoder 0 interrupt handler.
- SIGNAL (ENC1_SIGNAL)

  Encoder 1 interrupt handler.

Variables

- volatile EncoderStateType **EncoderState** [NUM_ENCODERS]

9.19.1 Detailed Description

Quadrature Encoder reader/driver.

Definition in file encoder.c.

9.20 encoder.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 //*****************************************************************************
```

```
00019 //
00020 // File Name  : 'encoder.c'
00021 // Title    : Quadrature Encoder reader/driver
00022 // Author   : Pascal Stang - Copyright (C) 2003-2004
00023 // Created    : 2003.01.26
00024 // Revised    : 2004.06.25
00025 // Version    : 0.3
00026 // Target MCU : Atmel AVR Series
00027 // Editor Tabs  : 4
00028 //
00029 // NOTE: This code is currently below version 1.0, and therefore is considered
00030 // to be lacking in some functionality or documentation, or may not be fully
00031 // tested.  Nonetheless, you can expect most functions to work.
00032 //
00033 // This code is distributed under the GNU Public License
00034 //    which can be found at http://www.gnu.org/licenses/gpl.txt
00035 //
00036 //******************************************************************************
00037 #ifndef WIN32
00038 #include <avr/io.h>
00039 #include <avr/interrupt.h>
00040 #endif
00041
00042 #include "global.h"
00043 #include "encoder.h"
00044
00045 // Program ROM constants
00046
00047 // Global variables
00048 volatile EncoderStateType EncoderState[NUM_ENCODERS];
00049
00050 // Functions
00051
00052 // encoderInit() initializes hardware and encoder position readings
00053 //    Run this init routine once before using any other encoder functions.
00054 inline void encoderInit(void)
00055 {
00056     uint8_t i;
00057
00058     // initialize/clear encoder data
00059     for (i = 0; i < NUM_ENCODERS; i++)
00060     {
00061         EncoderState[i].position = 0;
00062         //EncoderState[i].velocity = 0;   // NOT CURRENTLY USED
00063     }
00064
00065     // configure direction and interrupt I/O pins:
00066     // - for input
00067     // - apply pullup resistors
00068     // - any-edge interrupt triggering
00069     // - enable interrupt
00070
00071 #ifdef ENC0_SIGNAL
00072     // set interrupt pins to input and apply pullup resistor
00073     cbi(ENC0_PHASEA_DDR, ENC0_PHASEA_PIN);
00074     sbi(ENC0_PHASEA_PORT, ENC0_PHASEA_PIN);
00075     // configure interrupts for any-edge triggering
00076     sbi(ENC0_ICR, ENC0_ISCX0);
00077     cbi(ENC0_ICR, ENC0_ISCX1);
00078     // enable interrupts
00079     sbi(IMSK, ENC0_INT);
00080     // ISMK is auto-defined in encoder.h
00081 #endif
00082 #ifdef ENC1_SIGNAL
00083     // set interrupt pins to input and apply pullup resistor
00084     cbi(ENC1_PHASEA_DDR, ENC1_PHASEA_PIN);
00085     sbi(ENC1_PHASEA_PORT, ENC1_PHASEA_PIN);
00086     // configure interrupts for any-edge triggering
00087     sbi(ENC1_ICR, ENC1_ISCX0);
00088     cbi(ENC1_ICR, ENC1_ISCX1);
00089     // enable interrupts
00090     sbi(IMSK, ENC1_INT);
00091     // ISMK is auto-defined in encoder.h
00092 #endif
00093 #ifdef ENC2_SIGNAL
00094     // set interrupt pins to input and apply pullup resistor
00095     cbi(ENC2_PHASEA_DDR, ENC2_PHASEA_PIN);
00096     sbi(ENC2_PHASEA_PORT, ENC2_PHASEA_PIN);
00097     // configure interrupts for any-edge triggering
00098     sbi(ENC2_ICR, ENC2_ISCX0);
00099     cbi(ENC2_ICR, ENC2_ISCX1);
```

```
00100     // enable interrupts
00101     sbi(IMSK, ENC2_INT);// ISMK is auto-defined in encoder.h
00102 #endif
00103 #ifdef ENC3_SIGNAL
00104     // set interrupt pins to input and apply pullup resistor
00105     cbi(ENC3_PHASEA_DDR, ENC3_PHASEA_PIN);
00106     sbi(ENC3_PHASEA_PORT, ENC3_PHASEA_PIN);
00107     // set encoder direction pin for input and apply pullup resistor
00108     cbi(ENC3_PHASEB_DDR, ENC3_PHASEB_PIN);
00109     sbi(ENC3_PHASEB_PORT, ENC3_PHASEB_PIN);
00110     // configure interrupts for any-edge triggering
00111     sbi(ENC3_ICR, ENC3_ISCX0);
00112     cbi(ENC3_ICR, ENC3_ISCX1);
00113     // enable interrupts
00114     sbi(IMSK, ENC3_INT);// ISMK is auto-defined in encoder.h
00115 #endif
00116 
00117     // enable global interrupts
00118     sei();
00119 }
00120 
00121 // encoderGetPosition() reads the current position of the encoder
00122 uint16_t encoderGetPosition(uint8_t encoderNum)
00123 {
00124     // sanity check
00125     if (encoderNum < NUM_ENCODERS)
00126         return EncoderState[encoderNum].position;
00127     else
00128         return 0;
00129 }
00130 
00131 // encoderSetPosition() sets the current position of the encoder
00132 void encoderSetPosition(uint8_t encoderNum, uint16_t position
    )
00133 {
00134     // sanity check
00135     if (encoderNum < NUM_ENCODERS)
00136         EncoderState[encoderNum].position = position;
00137     // else do nothing
00138 }
00139 
00140 #ifdef ENC0_SIGNAL
00141 
00142 SIGNAL(ENC0_SIGNAL)
00143 {
00144     /*****************************************/
00145     /* Modified by Taylor                    */
00146     /*****************************************/
00147     EncoderState[0].position++;
00148 }
00149 #endif
00150 
00151 #ifdef ENC1_SIGNAL
00152 
00153 SIGNAL(ENC1_SIGNAL)
00154 {
00155     /*****************************************/
00156     /* Modified by Taylor                    */
00157     /*****************************************/
00158     EncoderState[1].position++;
00159 }
00160 #endif
00161 
00162 #ifdef ENC2_SIGNAL
00163 
00164 SIGNAL(ENC2_SIGNAL)
00165 {
00166     // encoder has generated a pulse
00167     // check the relative phase of the input channels
00168     // and update position accordingly
00169     if( ((inb(ENC2_PHASEA_PORTIN) & (1<<ENC2_PHASEA_PIN)) == 0) ^
00170         ((inb(ENC2_PHASEB_PORTIN) & (1<<ENC2_PHASEB_PIN)) == 0) )
00171     {
00172         EncoderState[2].position++;
00173     }
00174     else
00175     {
00176         EncoderState[2].position--;
00177     }
00178 }
00179 #endif
```

```
00180
00181 #ifdef ENC3_SIGNAL
00182
00183 SIGNAL(ENC3_SIGNAL)
00184 {
00185     // encoder has generated a pulse
00186     // check the relative phase of the input channels
00187     // and update position accordingly
00188     if( ((inb(ENC3_PHASEA_PORTIN) & (1<<ENC3_PHASEA_PIN)) == 0) ^
00189         ((inb(ENC3_PHASEB_PORTIN) & (1<<ENC3_PHASEB_PIN)) == 0) )
00190     {
00191         EncoderState[3].position++;
00192     }
00193     else
00194     {
00195         EncoderState[3].position--;
00196     }
00197 }
00198 #endif
```

9.21 encoder.h File Reference

Quadrature Encoder reader/driver.

```
#include "global.h"
#include "encoderconf.h"
#include <stdint.h>
```

Include dependency graph for encoder.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct struct_EncoderState
  Encoder state structure.

Macros

- #define **IMSK** GIMSK

Typedefs

- typedef struct struct_EncoderState EncoderStateType
  Encoder state structure.

Functions

- void encoderInit (void)
  encoderInit() initializes hardware and encoder position readings
- uint16_t encoderGetPosition (uint8_t encoderNum)
  encoderGetPosition() reads the current position of the encoder
- void encoderSetPosition (uint8_t encoderNum, uint16_t position)
  encoderSetPosition() sets the current position of the encoder

9.21.1 Detailed Description

Quadrature Encoder reader/driver.

Definition in file encoder.h.

9.22 encoder.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 //*****************************************************************************
00019 //
00020 // File Name  : 'encoder.h'
00021 // Title    : Quadrature Encoder reader/driver
00022 // Author   : Pascal Stang - Copyright (C) 2003-2004
00023 // Created    : 2003.01.26
00024 // Revised    : 2004.06.25
00025 // Version    : 0.3
00026 // Target MCU : Atmel AVR Series
00027 // Editor Tabs  : 4
00028 //
00029 // Description  : This library allows easy interfacing of quadrature encoders
00030 //    to the Atmel AVR-series processors.
00031 //
00032 //    Quadrature encoders have two digital outputs usually called PhaseA and
00033 //  PhaseB.  When the encoder rotates, PhaseA and PhaseB produce square wave
00034 //  pulses where each pulse represents a fraction of a turn of the encoder
00035 //  shaft.  Encoders are rated for a certain number of pulses (or counts) per
00036 //  complete revolution of the shaft.  Common counts/revolution specs are 50,
00037 //  100,128,200,250,256,500,etc.  By counting the number of pulses output on
00038 //  one of the phases starting from time0, you can calculate the total
00039 //  rotational distance the encoder has traveled.
00040 //
00041 //  Often, however, we want current position not just total distance traveled.
00042 //  For this it is necessary to know not only how far the encoder has traveled,
00043 //  but also which direction it was going at each step of the way.  To do this
00044 //  we need to use both outputs (or phases) of the quadrature encoder.
00045 //
00046 //  The pulses from PhaseA and PhaseB on quadrature encoders are always aligned
00047 //  90 degrees out-of-phase (otherwise said: 1/4 wavelength apart).  This
00048 //  special phase relationship lets us extract both the distance and direction
00049 //  the encoder has rotated from the outputs.
00050 //
00051 //  To do this, consider Phase A to be the distance counter.  On each rising
00052 //  edge of PhaseA we will count 1 "tic" of distance, but we need to know the
00053 //  direction.  Look at the quadrature waveform plot below.  Notice that when
00054 //  we travel forward in time (left->right), PhaseB is always low (logic 0) at
00055 //  the rising edge of PhaseA.  When we travel backwards in time (right->left),
00056 //  PhaseB is always high (logic 1) at the rising edge of PhaseA.  Note that
00057 //  traveling forward or backwards in time is the same thing as rotating
00058 //  forwards or bardwards. Thus, if PhaseA is our counter, PhaseB indicates
00059 //  direction.
00060 //
00061 //  Here is an example waveform from a quadrature encoder:
00062 /*
00063 //               /---\   /---\   /---\   /---\   /---\   /---\
00064 //  Phase A:     |   |   |   |   |   |   |   |   |   |   |   |
00065 //            ---/   \---/   \---/   \---/   \---/   \---/   \-
00066 //            -\   /---\   /---\   /---\   /---\   /---\   /---
00067 //  Phase B:   |   |   |   |   |   |   |   |   |   |   |   |
00068 //             \---/   \---/   \---/   \---/   \---/   \---/
00069 //  Time:    <-------------------------------------------------->
00070 //  Rotate FWD: >-------------------------------------------->
00071 //  Rotate REV: <--------------------------------------------<
00072 */
00073 //  To keep track of the encoder position in software, we connect PhaseA to an
00074 //  external processor interrupt line, and PhaseB to any I/O pin.  We set up
00075 //  the external interrupt to trigger whenever PhaseA produces a rising edge.
00076 //  When a rising edge is detected, our interrupt handler function is executed.
00077 //  Inside the handler function, we quickly check the PhaseB line to see if it
00078 //  is high or low.  If it is high, we increment the encoder's position
00079 //  counter, otherwise we decrement it.  The encoder position counter can be
```

```
00080 //  read at any time to find out the current position.
00081 //
00082 //
00083 // NOTE: This code is currently below version 1.0, and therefore is considered
00084 // to be lacking in some functionality or documentation, or may not be fully
00085 // tested.  Nonetheless, you can expect most functions to work.
00086 //
00087 // This code is distributed under the GNU Public License
00088 //    which can be found at http://www.gnu.org/licenses/gpl.txt
00089 //
00090 //*****************************************************************************
00091
00092 #ifndef ENCODER_H
00093 #define ENCODER_H
00094
00095 #include "global.h"
00096
00097 // include encoder configuration file
00098 #include "encoderconf.h"
00099
00100 #include <stdint.h>
00101
00102 // constants/macros/typdefs
00103
00104 // defines for processor compatibility
00105 // chose proper Interrupt Mask (IMSK)
00106 #ifdef EIMSK
00107   #define IMSK  EIMSK // for processors mega128, mega64
00108 #else
00109   #define IMSK  GIMSK // for other processors 90s8515, mega163, etc
00110 #endif
00111
00112
00114 //   stores the position and other information from each encoder
00115 typedef struct struct_EncoderState
00116 {
00117   uint16_t position;
00118 //  s32 velocity;       ///< velocity
00119 } EncoderStateType;
00120
00121
00122 // functions
00123
00125 //    Run this init routine once before using any other encoder function.
00126 inline void encoderInit(void);
00127
00129 uint16_t encoderGetPosition(uint8_t encoderNum);
00130
00132 void encoderSetPosition(uint8_t encoderNum, uint16_t position
      );
00133
00134 #endif
```

9.23 encoderconf.h File Reference

Quadrature Encoder driver configuration.

This graph shows which files directly or indirectly include this file:

Macros

- #define **NUM_ENCODERS** 2
- #define **ENC0_SIGNAL** SIG_INTERRUPT0
- #define **ENC0_INT** INT0
- #define **ENC0_ICR** MCUCR
- #define **ENC0_ISCX0** ISC00
- #define **ENC0_ISCX1** ISC01
- #define **ENC0_PHASEA_PORT** PORTD
- #define **ENC0_PHASEA_DDR** DDRD
- #define **ENC0_PHASEA_PORTIN** PIND

- #define **ENC0_PHASEA_PIN** PD2
- #define **ENC1_SIGNAL** SIG_INTERRUPT1
- #define **ENC1_INT** INT1
- #define **ENC1_ICR** MCUCR
- #define **ENC1_ISCX0** ISC10
- #define **ENC1_ISCX1** ISC11
- #define **ENC1_PHASEA_PORT** PORTD
- #define **ENC1_PHASEA_PORTIN** PIND
- #define **ENC1_PHASEA_DDR** DDRD
- #define **ENC1_PHASEA_PIN** PD3
- #define **ENC2_INT** INT6
- #define **ENC2_ICR** EICRB
- #define **ENC2_ISCX0** ISC60
- #define **ENC2_ISCX1** ISC61
- #define **ENC2_PHASEA_PORT** PORTE
- #define **ENC2_PHASEA_PORTIN** PINE
- #define **ENC2_PHASEA_DDR** DDRE
- #define **ENC2_PHASEA_PIN** PE6
- #define **ENC2_PHASEB_PORT** PORTC
- #define **ENC2_PHASEB_DDR** DDRC
- #define **ENC2_PHASEB_PORTIN** PINC
- #define **ENC2_PHASEB_PIN** PC2
- #define **ENC3_INT** INT7
- #define **ENC3_ICR** EICRB
- #define **ENC3_ISCX0** ISC70
- #define **ENC3_ISCX1** ISC71
- #define **ENC3_PHASEA_PORT** PORTE
- #define **ENC3_PHASEA_PORTIN** PINE
- #define **ENC3_PHASEA_DDR** DDRE
- #define **ENC3_PHASEA_PIN** PE7
- #define **ENC3_PHASEB_PORT** PORTC
- #define **ENC3_PHASEB_DDR** DDRC
- #define **ENC3_PHASEB_PORTIN** PINC
- #define **ENC3_PHASEB_PIN** PC3

9.23.1 Detailed Description

Quadrature Encoder driver configuration.

Definition in file encoderconf.h.

9.24 encoderconf.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
```

```
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 //*****************************************************************************
00019 //
00020 // File Name  : 'encoderconf.h'
00021 // Title    : Quadrature Encoder driver configuration
00022 // Author   : Pascal Stang - Copyright (C) 2003-2004
00023 // Created    : 2003.01.26
00024 // Revised    : 2004.06.25
00025 // Version    : 0.2
00026 // Target MCU : Atmel AVR Series
00027 // Editor Tabs  : 4
00028 //
00029 // The default number of encoders supported is 2 because most AVR processors
00030 // have two external interrupts.  To use more or fewer encoders, you must do
00031 // four things:
00032 //
00033 //  1. Use a processor with at least as many external interrutps as number of
00034 //    encoders you want to have.
00035 //  2. Set NUM_ENCODERS to the number of encoders you will use.
00036 //  3. Comment/Uncomment the proper ENCx_SIGNAL defines for your encoders
00037 //    (the encoders must be used sequentially, 0 then 1 then 2 then 3)
00038 //  4. Configure the various defines so that they match your processor and
00039 //    specific hardware.  The notes below may help.
00040 //
00041 //
00042 // -------------------- NOTES --------------------
00043 // The external interrupt pins are mapped as follows on most AVR processors:
00044 // (90s8515, mega161, mega163, mega323, mega16, mega32, etc)
00045 //
00046 // INT0 -> PD2 (PORTD, pin 2)
00047 // INT1 -> PD3 (PORTD, pin 3)
00048 //
00049 // The external interrupt pins on the processors mega128 and mega64 are:
00050 //
00051 // INT0 -> PD0 (PORTD, pin 0)
00052 // INT1 -> PD1 (PORTD, pin 1)
00053 // INT2 -> PD2 (PORTD, pin 2)
00054 // INT3 -> PD3 (PORTD, pin 3)
00055 // INT4 -> PE4 (PORTE, pin 4)
00056 // INT5 -> PE5 (PORTE, pin 5)
00057 // INT6 -> PE6 (PORTE, pin 6)
00058 // INT7 -> PE7 (PORTE, pin 7)
00059 //
00060 // This code is distributed under the GNU Public License
00061 //    which can be found at http://www.gnu.org/licenses/gpl.txt
00062 //
00063 //*****************************************************************************
00064
00065 #ifndef ENCODERCONF_H
00066 #define ENCODERCONF_H
00067
00068 // constants/macros/typdefs
00069
00070 // defines for processor compatibility
00071 // quick compatiblity for mega128, mega64
00072 //#ifndef MCUCR
00073 //  #define MCUCR EICRA
00074 //#endif
00075
00076 // Set the total number of encoders you wish to support
00077 #define NUM_ENCODERS        2
00078
00079
00080 // -------------------- Encoder 0 connections --------------------
00081 // Phase A quadrature encoder output should connect to this interrupt line:
00082 // *** NOTE: the choice of interrupt PORT, DDR, and PIN must match the external
00083 // interrupt you are using on your processor.  Consult the External Interrupts
00084 // section of your processor's datasheet for more information.
00085
00086 // Interrupt Configuration
00087 #define ENC0_SIGNAL         SIG_INTERRUPT0  // Interrupt signal name
00088 #define ENC0_INT          INT0  // matching INTx bit in GIMSK/EIMSK
00089 #define ENC0_ICR          MCUCR // matching Int. Config Register (MCUCR,EICRA/
      B)
00090 #define ENC0_ISCX0          ISC00 // matching Interrupt Sense Config bit0
00091 #define ENC0_ISCX1          ISC01 // matching Interrupt Sense Config bit1
00092 // PhaseA Port/Pin Configuration
00093 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
```

```
00094 #define ENC0_PHASEA_PORT      PORTD // PhaseA port register
00095 #define ENC0_PHASEA_DDR       DDRD  // PhaseA port direction register
00096 #define ENC0_PHASEA_PORTIN      PIND  // PhaseA port input register
00097 #define ENC0_PHASEA_PIN       PD2   // PhaseA port pin
00098 // Phase B quadrature encoder output should connect to this direction line:
00099 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00100 //#define ENC0_PHASEB_PORT      PORTC // PhaseB port register
00101 //#define ENC0_PHASEB_DDR       DDRC  // PhaseB port direction register
00102 //#define ENC0_PHASEB_PORTIN      PINC  // PhaseB port input register
00103 //#define ENC0_PHASEB_PIN       PC0   // PhaseB port pin
00104
00105
00106 // -------------------- Encoder 1 connections --------------------
00107 // Phase A quadrature encoder output should connect to this interrupt line:
00108 // *** NOTE: the choice of interrupt pin and port must match the external
00109 // interrupt you are using on your processor.  Consult the External Interrupts
00110 // section of your processor's datasheet for more information.
00111
00112 // Interrupt Configuration
00113 #define ENC1_SIGNAL         SIG_INTERRUPT1  // Interrupt signal name
00114 #define ENC1_INT          INT1  // matching INTx bit in GIMSK/EIMSK
00115 #define ENC1_ICR          MCUCR // matching Int. Config Register (MCUCR,EICRA/
      B)
00116 #define ENC1_ISCX0          ISC10 // matching Interrupt Sense Config bit0
00117 #define ENC1_ISCX1          ISC11 // matching Interrupt Sense Config bit1
00118 // PhaseA Port/Pin Configuration
00119 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00120 #define ENC1_PHASEA_PORT      PORTD // PhaseA port register
00121 #define ENC1_PHASEA_PORTIN      PIND  // PhaseA port input register
00122 #define ENC1_PHASEA_DDR       DDRD  // PhaseA port direction register
00123 #define ENC1_PHASEA_PIN       PD3   // PhaseA port pin
00124 // Phase B quadrature encoder output should connect to this direction line:
00125 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00126 //#define ENC1_PHASEB_PORT      PORTC // PhaseB port register
00127 //#define ENC1_PHASEB_DDR       DDRC  // PhaseB port direction register
00128 //#define ENC1_PHASEB_PORTIN      PINC  // PhaseB port input register
00129 //#define ENC1_PHASEB_PIN       PC1   // PhaseB port pin
00130
00131
00132 // -------------------- Encoder 2 connections --------------------
00133 // Phase A quadrature encoder output should connect to this interrupt line:
00134 // *** NOTE: the choice of interrupt pin and port must match the external
00135 // interrupt you are using on your processor.  Consult the External Interrupts
00136 // section of your processor's datasheet for more information.
00137
00138 // Interrupt Configuration
00139 //#define ENC2_SIGNAL         SIG_INTERRUPT6  // Interrupt signal name
00140 #define ENC2_INT          INT6  // matching INTx bit in GIMSK/EIMSK
00141 #define ENC2_ICR          EICRB // matching Int. Config Register (MCUCR,EICRA/
      B)
00142 #define ENC2_ISCX0          ISC60 // matching Interrupt Sense Config bit0
00143 #define ENC2_ISCX1          ISC61 // matching Interrupt Sense Config bit1
00144 // PhaseA Port/Pin Configuration
00145 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00146 #define ENC2_PHASEA_PORT      PORTE // PhaseA port register
00147 #define ENC2_PHASEA_PORTIN      PINE  // PhaseA port input register
00148 #define ENC2_PHASEA_DDR       DDRE  // PhaseA port direction register
00149 #define ENC2_PHASEA_PIN       PE6   // PhaseA port pin
00150 // Phase B quadrature encoder output should connect to this direction line:
00151 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00152 #define ENC2_PHASEB_PORT      PORTC // PhaseB port register
00153 #define ENC2_PHASEB_DDR       DDRC  // PhaseB port direction register
00154 #define ENC2_PHASEB_PORTIN      PINC  // PhaseB port input register
00155 #define ENC2_PHASEB_PIN       PC2   // PhaseB port pin
00156
00157
00158 // -------------------- Encoder 3 connections --------------------
00159 // Phase A quadrature encoder output should connect to this interrupt line:
00160 // *** NOTE: the choice of interrupt pin and port must match the external
00161 // interrupt you are using on your processor.  Consult the External Interrupts
00162 // section of your processor's datasheet for more information.
00163
00164 // Interrupt Configuration
00165 //#define ENC3_SIGNAL         SIG_INTERRUPT7  // Interrupt signal name
00166 #define ENC3_INT          INT7  // matching INTx bit in GIMSK/EIMSK
00167 #define ENC3_ICR          EICRB // matching Int. Config Register (MCUCR,EICRA/
      B)
00168 #define ENC3_ISCX0          ISC70 // matching Interrupt Sense Config bit0
00169 #define ENC3_ISCX1          ISC71 // matching Interrupt Sense Config bit1
00170 // PhaseA Port/Pin Configuration
00171 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
```

```
00172 #define ENC3_PHASEA_PORT      PORTE // PhaseA port register
00173 #define ENC3_PHASEA_PORTIN      PINE  // PhaseA port input register
00174 #define ENC3_PHASEA_DDR       DDRE  // PhaseA port direction register
00175 #define ENC3_PHASEA_PIN       PE7   // PhaseA port pin
00176 // Phase B quadrature encoder output should connect to this direction line:
00177 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00178 #define ENC3_PHASEB_PORT      PORTC // PhaseB port register
00179 #define ENC3_PHASEB_DDR       DDRC  // PhaseB port direction register
00180 #define ENC3_PHASEB_PORTIN      PINC  // PhaseB port input register
00181 #define ENC3_PHASEB_PIN       PC3   // PhaseB port pin
00182
00183 #endif
```

9.25 exercises.c File Reference

```
#include "exercises.h"
#include "botCntrl.h"
#include "motorCntrl.h"
#include "buttons.h"
#include "helperFunctions.h"
#include "ourLCD.h"
#include <stdbool.h>
```

Include dependency graph for exercises.c:

Functions

- void **bbPickupTest** (void)
- void **zigZagTest** (void)
- void **gbPickupTest** (void)
- void **diagTest** (void)
- void **node31Test** (void)

9.25.1 Detailed Description

Definition in file exercises.c.

9.26 exercises.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "exercises.h"
00019 #include "botCntrl.h"
00020 #include "motorCntrl.h"
00021 #include "buttons.h"
00022 #include "helperFunctions.h"
00023 #include "ourLCD.h"
00024
00025 // avr-libc
```

```
00026 #include <stdbool.h>
00027
00028 void bbPickupTest(void)
00029 {
00030     bool justTurned = true;
00031     bool firstRun = true;
00032
00033     while (1)
00034     {
00035 #if DEBUGGING
00036         lcdWriteStr("-96 brads, then ", 0, 0);
00037         lcdWriteStr("-32 brads       ", 1, 0);
00038 #endif
00039         waitFor(RED_BUTTON);
00040
00041         //hard left approach
00042         botNode = 23;
00043         botHeading = -128;
00044         pathListIndex = 0;
00045         pathList[0] = 23;
00046         pathList[1] = 3;
00047         pathList[2] = 24;
00048         pathList[3] = 3;
00049         pathList[4] = 23;
00050         justTurned = true;
00051         moveToJunction(1, justTurned);
00052         justTurned = positionBot();
00053         moveToJunction(1, justTurned);
00054         brake(BOTH);
00055
00056 #if DEBUGGING
00057         lcdWriteStr("96 brads, then  ", 0, 0);
00058         lcdWriteStr("32 brads        ", 1, 0);
00059 #endif
00060         waitFor(RED_BUTTON);
00061
00062         //Hard right approach
00063         botNode = 29;
00064         botHeading = 64;
00065         pathListIndex = 0;
00066         pathList[0] = 29;
00067         pathList[1] = 10;
00068         pathList[2] = 30;
00069         pathList[3] = 10;
00070         pathList[4] = 29;
00071         justTurned = true;
00072         moveToJunction(1, justTurned);
00073         justTurned = positionBot();
00074         moveToJunction(1, justTurned);
00075         brake(BOTH);
00076
00077 #if DEBUGGING
00078         lcdWriteStr("-32 brads, then ", 0, 0);
00079         lcdWriteStr("-96 brads       ", 1, 0);
00080 #endif
00081         waitFor(RED_BUTTON);
00082
00083         if (firstRun)
00084         {
00085             addToGoalList(BONUS_BALL_2);
00086         }
00087
00088         //Soft Left approach
00089         botNode = 31;
00090         botHeading = -64;
00091         pathListIndex = 0;
00092         pathList[0] = 31;
00093         pathList[1] = 30;
00094         pathList[2] = 31;
00095         justTurned = true;
00096         moveToJunction(1, justTurned);
00097         justTurned = positionBot();
00098         moveToJunction(1, justTurned);
00099         brake(BOTH);
00100
00101         printGoalList();
00102         waitFor(RED_BUTTON);
00103     }
00104 }
00105
00106 void zigZagTest(void)
```

```
00107 {
00108     bool justTurned = true;
00109
00110 #if DEBUGGING
00111     lcdWriteStr("botNode = 18    ", 0, 0);
00112     lcdWriteStr("botHeading = 0  ", 1, 0);
00113 #endif
00114     waitFor(RED_BUTTON);
00115
00116     botNode = 40;
00117     botHeading = 0;
00118     pathListIndex = 0;
00119     pathList[0] = 40;
00120     pathList[1] = 18;
00121     pathList[2] = 39;
00122     pathList[3] = 38;
00123     pathList[4] = 37;
00124     pathList[5] = 38;
00125     pathList[6] = 39;
00126     pathList[7] = 18;
00127     pathList[8] = 40;
00128
00129     justTurned = true;
00130     moveToJunction(1, justTurned);
00131     justTurned = positionBot();
00132     moveToJunction(1, justTurned);
00133     justTurned = positionBot();
00134     moveToJunction(1, justTurned);
00135     brake(BOTH);
00136
00137 #if DEBUGGING
00138     lcdWriteStr("Assume we do not", 0, 0);
00139     lcdWriteStr("seek ground ball", 1, 0);
00140 #endif
00141     msDelay(3000);
00142
00143     justTurned = positionBot();
00144     moveToJunction(1, justTurned);
00145     justTurned = positionBot();
00146     moveToJunction(1, justTurned);
00147     justTurned = positionBot();
00148     moveToJunction(1, justTurned);
00149     brake(BOTH);
00150
00151 #if DEBUGGING
00152     lcdWriteStr("botNode = 18    ", 0, 0);   //------------------------------
00153     lcdWriteStr("botHeading = 0  ", 1, 0);
00154 #endif
00155     waitFor(RED_BUTTON);
00156
00157     botNode = 40;
00158     botHeading = 0;
00159     pathListIndex = 0;
00160     pathList[0] = 40;
00161     pathList[1] = 18;
00162     pathList[2] = 39;
00163     pathList[3] = 38;
00164     pathList[4] = 37;
00165     pathList[5] = 36;
00166     pathList[6] = 35;
00167     pathList[7] = 17;
00168     pathList[8] = 34;
00169
00170     justTurned = true;
00171     moveToJunction(1, justTurned);
00172     justTurned = positionBot();
00173     moveToJunction(1, justTurned);
00174     justTurned = positionBot();
00175     moveToJunction(1, justTurned);
00176     brake(BOTH);
00177
00178 #if DEBUGGING
00179     lcdWriteStr("Assume we seek  ", 0, 0);
00180     lcdWriteStr("ground ball.    ", 1, 0);
00181 #endif
00182     msDelay(3000);
00183
00184     justTurned = positionBot();
00185     moveToJunction(1, justTurned);
00186     justTurned = positionBot();
00187     moveToJunction(1, justTurned);
```

```
00188     justTurned = positionBot();
00189     moveToJunction(1, justTurned);
00190     brake(BOTH);
00191
00192 #if DEBUGGING
00193     lcdWriteStr("botNode = 17    ", 0, 0);   //------------------------------
00194     lcdWriteStr("botHeading = 64 ", 1, 0);
00195 #endif
00196     waitFor(RED_BUTTON);
00197
00198     botNode = 34;
00199     botHeading = 64;
00200     pathListIndex = 0;
00201     pathList[0] = 34;
00202     pathList[1] = 17;
00203     pathList[2] = 35;
00204     pathList[3] = 36;
00205     pathList[4] = 37;
00206     pathList[5] = 38;
00207     pathList[6] = 39;
00208     pathList[7] = 18;
00209     pathList[8] = 40;
00210
00211     justTurned = true;
00212     moveToJunction(1, justTurned);
00213     justTurned = positionBot();
00214     moveToJunction(1, justTurned);
00215     justTurned = positionBot();
00216     moveToJunction(1, justTurned);
00217     justTurned = positionBot();
00218     moveToJunction(1, justTurned);
00219     justTurned = positionBot();
00220     moveToJunction(1, justTurned);
00221     justTurned = positionBot();
00222     moveToJunction(1, justTurned);
00223     brake(BOTH);
00224 }
00225
00226 void gbPickupTest(void)
00227 {
00228     bool justTurned = true;
00229
00230     while (1)
00231     {
00232 #if DEBUGGING
00233         lcdWriteStr("One ground ball ", 0, 0);  //
    ------------------------------
00234         lcdWriteStr("(1 junc b4 ball)", 1, 0);
00235 #endif
00236         waitFor(RED_BUTTON);
00237
00238         botNode = 22;                     // set path
00239         botHeading = 0;
00240         pathListIndex = 0;
00241         pathList[0] = 22;
00242         pathList[1] = 9;
00243         pathList[2] = 29;
00244         pathList[3] = 11;
00245         pathList[4] = 12;
00246         pathList[5] = 33;
00247         pathList[6] = 13;
00248         pathList[7] = 34;
00249
00250         initGoalList();                   // tell bot where balls are
00251         removeFromGoalList(BONUS_BALL_1);
00252         removeFromGoalList(BONUS_BALL_2);
00253         //removeFromGoalList(SENSOR_NODE);
00254         addToGoalList(11);
00255
00256         justTurned = true;                // run test
00257         moveToJunction(1, justTurned);
00258         justTurned = positionBot();
00259         moveToJunction(1, false);
00260         justTurned = positionBot();
00261         moveToJunction(1, false);
00262         brake(BOTH);
00263
00264 #if DEBUGGING
00265         lcdWriteStr("Two ground balls", 0, 0);  //
    ------------------------------
00266         lcdWriteStr("(1 junc b4 ball)", 1, 0); //  try placing just one ball to
```

```
      test
00267 #endif                                          //  safeguard before turn
00268         waitFor(RED_BUTTON);
00269         botNode = 22;                     // set path
00270         botHeading = 0;
00271         pathListIndex = 0;
00272         pathList[0] = 22;
00273         pathList[1] = 9;
00274         pathList[2] = 29;
00275         pathList[3] = 11;
00276         pathList[4] = 12;
00277         pathList[5] = 33;
00278         pathList[6] = 13;
00279         pathList[7] = 34;
00280
00281         initGoalList();                   // tell bot where balls are
00282         removeFromGoalList(BONUS_BALL_1);
00283         removeFromGoalList(BONUS_BALL_2);
00284         removeFromGoalList(SENSOR_NODE);
00285         addToGoalList(11);
00286         addToGoalList(12);
00287
00288         justTurned = true;                // run test
00289         moveToJunction(1, justTurned);
00290         justTurned = positionBot();
00291         moveToJunction(1, justTurned);
00292         justTurned = positionBot();
00293         moveToJunction(1, justTurned);
00294         brake(BOTH);
00295
00296 #if DEBUGGING
00297         lcdWriteStr("botNode = 20    ", 0, 0);  //
      ------------------------------
00298         lcdWriteStr("botHeading =-128", 1, 0); // ( -128 brad turn after pickup
      )
00299 #endif                                          // make sure nestSequence is
       called
00300         waitFor(RED_BUTTON);
00301         botNode = 20;                     // set path
00302         botHeading = -128;
00303         pathListIndex = 0;
00304         pathList[0] = 20;
00305         pathList[1] = 41;
00306         pathList[2] = 5;
00307         pathList[3] = 41;
00308         pathList[4] = 42;
00309
00310         initGoalList();                   // tell bot where balls are
00311         removeFromGoalList(BONUS_BALL_1);
00312         removeFromGoalList(BONUS_BALL_2);
00313         removeFromGoalList(SENSOR_NODE);
00314         addToGoalList(5);
00315
00316         justTurned = true;                // run test
00317         moveToJunction(1, justTurned);
00318         justTurned = positionBot();
00319         moveToJunction(1, justTurned);
00320         justTurned = positionBot();
00321         nestSequence();
00322         brake(BOTH);
00323     }
00324
00325 }
00326
00327 void diagTest(void)
00328 {
00329     bool justTurned = true;
00330
00331 #if DEBUGGING
00332     lcdWriteStr("botNode = 10    ", 0, 0);
00333     lcdWriteStr("botHeading = 64 ", 1, 0);
00334 #endif
00335     waitFor(RED_BUTTON);
00336
00337     botNode = 10;                       // set path
00338     botHeading = 64;
00339     pathListIndex = 0;
00340     pathList[0] = 10;
00341     pathList[1] = 30;
00342     pathList[2] = 31;
00343     pathList[3] = 15;
```

```
00344     pathList[4] = 14;
00345     pathList[5] = 34;
00346     pathList[6] = 13;
00347     pathList[7] = 33;
00348     pathList[8] = 13;
00349     pathList[9] = 34;
00350     pathList[10] = 14;
00351     pathList[11] = 15;
00352     pathList[12] = 31;
00353     pathList[13] = 30;
00354     pathList[14] = 10;
00355     pathList[15] = 29;
00356
00357     initGoalList();                   // tell bot where balls are
00358     //addToGoalList(14);
00359     //addToGoalList(13);
00360
00361     justTurned = true;                // run test
00362     moveToJunction(1, justTurned);
00363     justTurned = positionBot();
00364     moveToJunction(1, justTurned);
00365     justTurned = positionBot();
00366     moveToJunction(1, justTurned);
00367     justTurned = positionBot();
00368     moveToJunction(1, justTurned);
00369     justTurned = positionBot();       // Should be -128 brad turn
00370     moveToJunction(1, justTurned);
00371     justTurned = positionBot();
00372     moveToJunction(1, justTurned);
00373     justTurned = positionBot();
00374     moveToJunction(1, justTurned);
00375     justTurned = positionBot();
00376     moveToJunction(1, justTurned);
00377     brake(BOTH);
00378
00379 #if DEBUGGING
00380     lcdWriteStr("botNode = 16    ", 0, 0);   //------------------------------
00381     lcdWriteStr("botHeading = -64", 1, 0);
00382 #endif
00383     waitFor(RED_BUTTON);
00384
00385     botNode = 16;                     // set path
00386     botHeading = -64;
00387     pathListIndex = 0;
00388     pathList[0] = 16;
00389     pathList[1] = 32;
00390     pathList[2] = 31;
00391     pathList[3] = 15;
00392     pathList[4] = 14;
00393     pathList[5] = 34;
00394     pathList[6] = 17;
00395     pathList[7] = 35;
00396     pathList[8] = 17;
00397     pathList[9] = 34;
00398     pathList[10] = 14;
00399     pathList[11] = 15;
00400     pathList[12] = 31;
00401     pathList[13] = 32;
00402     pathList[14] = 16;
00403     pathList[15] = 40;
00404
00405     initGoalList();                   // tell bot where balls are
00406     //addToGoalList(15);
00407     //addToGoalList(17);
00408
00409     justTurned = true;                // run test
00410     moveToJunction(1, justTurned);
00411     justTurned = positionBot();
00412     moveToJunction(1, justTurned);
00413     justTurned = positionBot();
00414     moveToJunction(1, justTurned);
00415     justTurned = positionBot();
00416     moveToJunction(1, justTurned);
00417     justTurned = positionBot();       // Should be -128 brad turn
00418     moveToJunction(1, justTurned);
00419     justTurned = positionBot();
00420     moveToJunction(1, justTurned);
00421     justTurned = positionBot();
00422     moveToJunction(1, justTurned);
00423     justTurned = positionBot();
00424     moveToJunction(1, justTurned);
```

```
00425     brake(BOTH);
00426 }
00427
00428 void node31Test(void)
00429 {
00430     bool justTurned = true;
00431
00432 #if DEBUGGING
00433     lcdWriteStr("botNode = 7     ", 0, 0);   //------------------------------
00434     lcdWriteStr("botHeading = 0  ", 1, 0);
00435 #endif
00436     waitFor(RED_BUTTON);
00437
00438     /* initial testing for ball on diagonal
00439      botNode = 7;                     // set path
00440      botHeading = 0;
00441      pathListIndex = 0;
00442      pathList[0] = 7;
00443      pathList[1] = 32;
00444      pathList[2] = 31;
00445      pathList[3] = 15;
00446      pathList[4] = 31;
00447      pathList[5] = 32;
00448      pathList[6] = 7;
00449      pathList[7] = 27;
00450      */
00451     //testing for diagonal, 2 junctions, and 180 at end of diag
00452     botNode = 7;                     // set path
00453     botHeading = 0;
00454     pathListIndex = 0;
00455     pathList[0] = 7;
00456     pathList[1] = 32;
00457     pathList[2] = 31;
00458     pathList[3] = 15;
00459     pathList[4] = 14;
00460     pathList[5] = 34;
00461     pathList[6] = 14;
00462     pathList[7] = 15;
00463     pathList[8] = 31;
00464     pathList[9] = 32;
00465     pathList[10] = 7;
00466     pathList[11] = 27;
00467
00468     initGoalList();                  // tell bot where balls are
00469     //addToGoalList(15);
00470
00471     /* Initial diag ball test
00472      justTurned = true;               // run test
00473      moveToJunction(1, justTurned);
00474      justTurned = positionBot();
00475      moveToJunction(1, justTurned);
00476      justTurned = positionBot();
00477      moveToJunction(1, justTurned);    // -128 brads, try no ball
00478      justTurned = positionBot();
00479      moveToJunction(1, justTurned);
00480      justTurned = positionBot();
00481      moveToJunction(1, justTurned);
00482      brake(BOTH);
00483      */
00484
00485     justTurned = true;               // run test
00486     moveToJunction(1, justTurned);
00487     justTurned = positionBot();
00488     moveToJunction(1, justTurned);
00489     justTurned = positionBot();
00490     moveToJunction(1, justTurned);    // -128 brads, try no ball
00491     justTurned = positionBot();
00492     moveToJunction(1, justTurned);
00493     justTurned = positionBot();
00494     moveToJunction(1, justTurned);
00495     justTurned = positionBot();
00496     moveToJunction(1, justTurned);
00497     brake(BOTH);
00498
00499 #if DEBUGGING
00500     lcdWriteStr("botNode = 16    ", 0, 0);   //------------------------------
00501     lcdWriteStr("botHeading = -64", 1, 0);
00502 #endif
00503     waitFor(RED_BUTTON);
00504
00505     botNode = 16;                    // set path
```

```
00506     botHeading = -64;
00507     pathListIndex = 0;
00508     pathList[0] = 16;
00509     pathList[1] = 32;
00510     pathList[2] = 31;
00511     pathList[3] = 30;
00512     pathList[4] = 31;
00513     pathList[5] = 32;
00514     pathList[6] = 16;
00515     pathList[7] = 40;
00516
00517     initGoalList();                   // tell bot where balls are
00518
00519     justTurned = true;                // run test
00520     moveToJunction(1, justTurned);
00521     justTurned = positionBot();
00522     moveToJunction(1, justTurned);
00523     justTurned = positionBot();
00524     moveToJunction(1, justTurned);
00525     justTurned = positionBot();
00526     moveToJunction(1, justTurned);
00527     justTurned = positionBot();
00528     moveToJunction(1, justTurned);
00529     justTurned = positionBot();
00530     moveToJunction(1, justTurned);
00531     brake(BOTH);
00532
00533 #if DEBUGGING
00534     lcdWriteStr("botNode = 16    ", 0, 0);   //------------------------------
00535     lcdWriteStr("botHeading = -64", 1, 0);
00536 #endif
00537     waitFor(RED_BUTTON);
00538
00539     botNode = 16;                     // set path
00540     botHeading = -64;
00541     pathListIndex = 0;
00542     pathList[0] = 16;
00543     pathList[1] = 32;
00544     pathList[2] = 7;
00545     pathList[3] = 27;
00546     pathList[4] = 7;
00547     pathList[5] = 32;
00548     pathList[6] = 31;
00549     pathList[7] = 30;
00550     pathList[8] = 31;
00551     pathList[9] = 32;
00552     pathList[10] = 7;
00553     pathList[11] = 27;
00554     pathList[12] = 7;
00555     pathList[13] = 32;
00556     pathList[14] = 16;
00557     pathList[15] = 40;
00558
00559     initGoalList();                   // tell bot where balls are
00560
00561     justTurned = true;                // run test
00562     moveToJunction(1, justTurned);
00563     justTurned = positionBot();
00564     moveToJunction(1, justTurned);
00565     justTurned = positionBot();
00566     moveToJunction(1, justTurned);
00567     justTurned = positionBot();
00568     moveToJunction(1, justTurned);
00569     justTurned = positionBot();
00570     moveToJunction(1, justTurned);
00571     justTurned = positionBot();
00572     moveToJunction(1, justTurned);
00573     justTurned = positionBot();
00574     moveToJunction(1, justTurned);
00575     justTurned = positionBot();
00576     moveToJunction(1, justTurned);
00577     justTurned = positionBot();
00578     moveToJunction(1, justTurned);
00579     justTurned = positionBot();
00580     moveToJunction(1, justTurned);
00581     brake(BOTH);
00582
00583 }
```

9.27 exercises.h File Reference

Exercise various high-level capabilities.

This graph shows which files directly or indirectly include this file:

Functions

- void **bbPickupTest** (void)
- void **gbPickupTest** (void)
- void **zigZagTest** (void)
- void **diagTest** (void)
- void **node31Test** (void)

9.27.1 Detailed Description

Exercise various high-level capabilities.

Definition in file exercises.h.

9.28 exercises.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00021 #ifndef EXERCISES_H_
00022 #define EXERCISES_H_
00023
00024 void bbPickupTest( void );
00025 void gbPickupTest( void );
00026 void zigZagTest( void );
00027 void diagTest( void );
00028 void node31Test( void );
00029
00030 #endif // #ifndef EXERCISES_H_
```

9.29 junctionCode.c File Reference

```
#include "junctionCode.h"
#include "botCntrl.h"
#include "trackColor.h"
#include "servos.h"
#include "camera.h"
#include "nodeList.h"
#include "eeProm.h"
#include "motorCntrl.h"
#include "updatePath.h"
#include "helperFunctions.h"
```

Include dependency graph for junctionCode.c:

Functions

- void **junctionCode** (void)
- bool **standardBallSearch** (void)
- bool **nodeCode0** (void)
- bool **nodeCode22** ()
- bool **diagNodeCode** (void)
- bool **nodeCode37** (void)

Variables

- bool **checkedList** []
- uint8_t **goalList** [NUM_GOALS]
- uint8_t **goalListSize**
- uint8_t **numKnownGoals**

9.29.1 Detailed Description

Definition in file junctionCode.c.

9.29.2 Variable Documentation

9.29.2.1 bool checkedList[]

Initial value:

```
{ false, false, false, false, false, false,
                      false, false, false, false, false, false,
                      false, false, false, false, false, false,
                      false, false, false }
```

Definition at line 31 of file junctionCode.c.

9.30 junctionCode.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "junctionCode.h"
00019 #include "botCntrl.h"
00020 #include "trackColor.h"
00021 #include "servos.h"
00022 #include "camera.h"
```

```
00023 #include "nodeList.h"
00024 #include "eeProm.h"
00025 #include "motorCntrl.h"
00026 #include "updatePath.h"
00027 #include "helperFunctions.h"
00028
00029 // Global variables
00030 // initialized in initBotGlobals
00031 bool checkedList[] = { false, false, false, false, false, false,
00032                        false, false, false, false, false, false,
00033                        false, false, false, false, false, false,
00034                        false, false, false };
00035 // initialized in initBotGlobals
00036 uint8_t goalList[NUM_GOALS];
00037
00038 uint8_t goalListSize;
00039 uint8_t numKnownGoals;
00040
00041 /*
00042  * Searches for ground balls, picks-up bonous balls, and computes best path.
00043  */
00044 void junctionCode(void)
00045 {
00046     bool foundBall = false;
00047
00048     switch (botNode)
00049     {
00050     case (0):                     // old virtual windowing look for ball 7 and
       8
00051         foundBall = nodeCode0();
00052         break;
00053     case (21):                          // Suppress standard seek, should
       already
00054         break;                          // skip junction code at this node
00055     case (22):
00056         foundBall = standardBallSearch();  // standard seek and
00057         foundBall |= nodeCode22();         // tilt look for ball 9, 11, and 12
00058         break;
00059     case (31):                          // rotate bot for diagonal ball search
00060     case (34):                          // from any heading at node 31 or 34
00061         foundBall = diagNodeCode();
00062         break;
00063     case (37):                           // seek for top balls from nest
       sensor
00064         if (numKnownGoals < NUM_GOALS && (!checkedList[13] || !checkedList[17])
      )
00065         {
00066             botNode = 35;
00067             moveStraight(10, 255);
00068             foundBall = diagNodeCode();
00069             moveStraight(-10, 255);
00070             botNode = 37;
00071
00072             //pathList[pathListIndex--] = 36;
00073             //pathList[pathListIndex--] = 35;
00074         }
00075         break;
00076     default:
00077         foundBall = standardBallSearch();
00078         break;
00079     }
00080
00081     if (foundBall)
00082     {
00083         // clear checked list, if last ball found
00084         if (numKnownGoals == NUM_GOALS)
00085         {
00086             uint8_t i;
00087             for (i = 0; i < NUM_BALL_NODES + 1; i++)
00088             {
00089                 checkedList[i] = true;
00090             }
00091         }
00092         updatePath();
00093         printGoalList();
00094     }
00095 }
00096
00097
00098 /*
00099  * Returns true if a ball is found and the goal list is updated
```

```
00100  */
00101 bool standardBallSearch( void )
00102 {
00103     NODE curNode;
00104     NODE nextNode;
00105     uint8_t nextNodeNum;
00106     int8_t lookDir = -1;                    // look left first
00107     int8_t hallHeading = 0;
00108     uint8_t ballDist = 0;
00109     uint8_t uncheckedBalls[3][2];
00110     uint8_t numUncheckedBalls = 0;
00111     bool foundBall = false;
00112     uint8_t i;
00113
00114     bool stopped = false;
00115     inSeekPosition = false;
00116
00117     // Check for balls in two directions (left/right)
00118     for (i = 0; i < 2; i++)
00119     {
00120         ballDist = 0;
00121         hallHeading = botHeading + lookDir * 64;
00122         nextNodeNum = botNode;
00123         numUncheckedBalls = 0;
00124         // Continue traversing nodes left of right until you hit the end
00125         while (nextNodeNum > 0)
00126         {
00127             /*
00128              #if DEBUGGING
00129              lcdWriteStr("N:   H:   ", 0, 0);
00130              lcdPrintHex(nextNodeNum, 0, 2);
00131              lcdPrintHex(hallHeading, 0, 7);
00132              waitFor(RED_BUTTON);
00133              #endif
00134              */
00135             getNode(nextNodeNum, &curNode);
00136             nextNodeNum = getNodeAtHeading(&curNode, hallHeading);
00137             if (nextNodeNum > 0)
00138             {
00139                 getNode(nextNodeNum, &nextNode);
00140                 // Keep track of how far away we are from the bot's current
      node
00141                 ballDist += getCostToNode(&curNode, nextNodeNum);
00142                 if (isBallNode(nextNodeNum) && !checkedList[nextNodeNum])
00143                 {
00144                     uncheckedBalls[numUncheckedBalls][BALL_DIST] = ballDist;
00145                     uncheckedBalls[numUncheckedBalls][BALL_NODE_NUM] =
00146                                     nextNodeNum;
00147                     checkedList[nextNodeNum] = true;
00148                     numUncheckedBalls++;
00149                 }
00150             }
00151         }
00152         // Set pan, tilt, hi-res, etc...
00153
00154         if (numUncheckedBalls > 0)
00155         {
00156             stopped = true;
00157             trackColorInit(lookDir);
00158
00159             if (lookDir == -1)
00160             {
00161                 foundBall |= cameraSeekLeft(uncheckedBalls, numUncheckedBalls);
00162             } else if (lookDir == 1)
00163             {
00164                 foundBall |= cameraSeekRight(uncheckedBalls, numUncheckedBalls)
      ;
00165             }
00166         }
00167
00168         /*
00169          #if DEBUGGING
00170          lcdWriteStr("Seek (  ) for:  ", 0, 0);
00171          lcdPrintHex(lookDir, 0, 6);
00172          lcdWriteStr("                ", 1, 0);
00173          for(j = 0; j < numUncheckedBalls; j++)
00174          {
00175          lcdPrintHex(uncheckedBalls[j][BALL_NODE_NUM], 1, 0);
00176          lcdPrintHex(uncheckedBalls[j][BALL_DIST], 1, 3);
00177          waitFor(RED_BUTTON);
00178          }
```

```
00179          waitFor(RED_BUTTON);
00180          #endif
00181          */
00182
00183          lookDir *= -1;  // Look the other way the next time through
00184      }
00185
00186      if (stopped)
00187      {
00188          moveStraight(0xb, 255);
00189          setServo(PAN, PAN_CENTER + panOffset);
00190          setServo(TILT, TILT_FORWARD);
00191          msDelay(600);
00192      }
00193
00194      // Returns true if one or more balls are found
00195      return foundBall;
00196  }
00197
00198
00199  inline bool nodeCode0(void)
00200  {
00201     bool foundBall = false;
00202
00203     // two virtual windows
00204
00205     return foundBall;
00206  }
00207
00208
00209  inline bool nodeCode22()
00210  {
00211     bool foundBall = false;    // Return value
00212     uint8_t scanHeight = 4;
00213     uint8_t y = 254;
00214     uint8_t scanLimit = 1;
00215     uint8_t foundBallNum = 0;
00216
00217     if (botHeading != 0)
00218        return false;
00219
00220     trackColorInit(LOOK_UP);
00221
00222     // scan from small ground distance to large ground distance
00223     while ( y - scanHeight > scanLimit )
00224     {
00225        y -= scanHeight;
00226        setVirtualWindow(1, y-scanHeight, 174, y);
00227        if ( seeBall() )
00228        {
00229           foundBall = true;
00230
00231           // find ball number of ball at this x
00232           if( y > 148 )
00233              foundBallNum = 9;
00234           else if ( y > 50 )
00235              foundBallNum = 11;
00236           else
00237              foundBallNum = 12;
00238
00239           addToGoalList( foundBallNum );
00240
00241  #if DEBUGGING
00242           labelColorStats();
00243           refreshColorStats();
00244           //msDelay(1000);
00245           //clearColorStats();
00246  #endif
00247
00248  /*
00249  #if DEBUGGING
00250           lcdWriteStr("Added:          ",0,0);
00251           lcdWriteStr("                ",1,0);
00252           lcdPrintHex(foundBallNum,1,0);
00253           waitFor(RED_BUTTON);
00254  #endif
00255  */
00256           while ( seeBall() )
00257           {
00258              y -= scanHeight;
00259              setVirtualWindow(1, y-scanHeight, 174, y);
```

```
00260          }
00261       }
00262    }
00263
00264    setServo(PAN, PAN_CENTER+panOffset);
00265    setServo(TILT, TILT_FORWARD);
00266    msDelay(300);
00267
00268    return foundBall;
00269 }
00270
00271
00272 inline bool diagNodeCode(void)
00273 {
00274    bool foundBall = false;
00275
00276    if( botHeading == N_WEST && (!checkedList[13] || !checkedList[17]) )
00277    {
00278       tankTurn(255,tempTweak3);   // tank right
00279       botHeading += 41;
00280       foundBall = standardBallSearch();
00281       botHeading -= 41;
00282       tankTurn(255, -1*tempTweak3);     // tank left
00283    }
00284    else if( botHeading != S_EAST && (!checkedList[14] || !checkedList[15]) )
00285    {
00286       tankTurn(255, -1*tempTweak3);     // tank left
00287       botHeading -= 41;
00288       foundBall = standardBallSearch();
00289       botHeading += 41;
00290       tankTurn(255,tempTweak3);   // tank right
00291    }
00292
00293    return foundBall;
00294 }
00295
00296 inline bool nodeCode37( void )
00297 {
00298    bool foundBall = false;
00299
00300    // pass special values into cameraSeekLeft
00301
00302    return foundBall;
00303 }
```

9.31 junctionCode.h File Reference

Actions that occur at junctions.

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for junctionCode.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **BALL_DIST** 0
- #define **BALL_NODE_NUM** 1

Functions

- void **junctionCode** (void)
- bool **standardBallSearch** (void)
- bool **nodeCode0** (void)
- bool **nodeCode22** (void)
- bool **diagNodeCode** (void)
- bool **nodeCode37** (void)

Variables

- bool **checkedList** []
- uint8_t **goalList** []
- uint8_t **goalListSize**
- uint8_t **numKnownGoals**

9.31.1 Detailed Description

Actions that occur at junctions.

Definition in file junctionCode.h.

9.32 junctionCode.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00021 #ifndef JUNCTIONCODE_H_
00022 #define JUNCTIONCODE_H_
00023
00024 #include <stdint.h>
00025 #include <stdbool.h>
00026
00027 #define BALL_DIST       0
00028 #define BALL_NODE_NUM   1
00029
00030 // Global variables
00031 extern bool checkedList[];
00032 extern uint8_t goalList[];
00033 extern uint8_t goalListSize;
00034 extern uint8_t numKnownGoals;
00035
00036 void junctionCode(void);
00037 bool standardBallSearch( void );
00038 inline bool nodeCode0( void );
00039 inline bool nodeCode22( void );
00040 inline bool diagNodeCode(void);
00041 inline bool nodeCode37( void );
00042
00043 #endif // #ifndef JUNCTIONCODE_H_
```

9.33 nodeList.c File Reference

```
#include "nodeList.h"
#include <string.h>
```

Include dependency graph for nodeList.c:

Functions

- bool **isJunction** (uint8_t nodeNum)

- bool **isBallNode** (uint8_t nodeNum)
- uint8_t **getCostToNode** (NODE *node, uint8_t nodeNum)
- uint8_t **getNodeAtHeading** (NODE *node, int8_t heading)
- void **getNode** (uint8_t nodeNum, NODE *node)

9.33.1 Detailed Description

Definition in file nodeList.c.

9.34 nodeList.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "nodeList.h"
00019
00020 // avr-libc
00021 #include <string.h>
00022
00023 inline bool isJunction(uint8_t nodeNum)
00024 {
00025     return (nodeNum >= JUNCTION_MIN && nodeNum <= JUNCTION_MAX);
00026 }
00027
00028 inline bool isBallNode(uint8_t nodeNum)
00029 {
00030     return (nodeNum >= BALL_NODE_MIN && nodeNum <= BALL_NODE_MAX);
00031 }
00032
00033 uint8_t getCostToNode(NODE *node, uint8_t nodeNum)
00034 {
00035     uint8_t i;
00036     for (i = 0; i < node->numAdjNodes; i++)
00037     {
00038         if (node->adjNodes[i] == nodeNum)
00039         {
00040             return node->adjCosts[i];
00041         }
00042     }
00043     return 0;
00044 }
00045
00046 uint8_t getNodeAtHeading(NODE *node, int8_t heading)
00047 {
00048     uint8_t i;
00049     for (i = 0; i < node->numAdjNodes; i++)
00050     {
00051         if (node->adjHeadings[i] == heading)
00052         {
00053             return node->adjNodes[i];
00054         }
00055     }
00056     return 0;
00057 }
00058
00059 void getNode(uint8_t nodeNum, NODE *node)
00060 {
00061     if (nodeNum >= NUM_NODES)
00062     {
00063         node = NULL;
```

```
00064     }
00065
00066     switch (nodeNum)
00067     {
00068     case 0:                           // START_NODE
00069         node->numAdjNodes = 1;
00070         node->   adjNodes[0] = 21;
00071         node->   adjCosts[0] = 9;
00072         node->adjHeadings[0] = -64;
00073         break;
00074     case 1:                           // First ball node
00075         node->numAdjNodes = 2;
00076         node->   adjNodes[0] = 21;
00077         node->   adjNodes[1] = 22;
00078         node->   adjCosts[0] = 4;
00079         node->   adjCosts[1] = 4;
00080         node->adjHeadings[0] = -128;
00081         node->adjHeadings[1] = 0;
00082         break;
00083     case 2:
00084         node->numAdjNodes = 2;
00085         node->   adjNodes[0] = 22;
00086         node->   adjNodes[1] = 23;
00087         node->   adjCosts[0] = 2;
00088         node->   adjCosts[1] = 2;
00089         node->adjHeadings[0] = -64;
00090         node->adjHeadings[1] = 64;
00091         break;
00092     case 3:
00093         node->numAdjNodes = 2;
00094         node->   adjNodes[0] = 23;
00095         node->   adjNodes[1] = 24;
00096         node->   adjCosts[0] = 2;
00097         node->   adjCosts[1] = 2;
00098         node->adjHeadings[0] = 0;
00099         node->adjHeadings[1] = -128;
00100         break;
00101     case 4:
00102         node->numAdjNodes = 2;
00103         node->   adjNodes[0] = 24;
00104         node->   adjNodes[1] = 25;
00105         node->   adjCosts[0] = 3;
00106         node->   adjCosts[1] = 3;
00107         node->adjHeadings[0] = -64;
00108         node->adjHeadings[1] = 64;
00109         break;
00110     case 5:
00111         node->numAdjNodes = 2;
00112         node->   adjNodes[0] = 26;
00113         node->   adjNodes[1] = 41;
00114         node->   adjCosts[0] = 2;
00115         node->   adjCosts[1] = 2;
00116         node->adjHeadings[0] = -64;
00117         node->adjHeadings[1] = 64;
00118         break;
00119     case 6:
00120         node->numAdjNodes = 2;
00121         node->   adjNodes[0] = 27;
00122         node->   adjNodes[1] = 28;
00123         node->   adjCosts[0] = 2;
00124         node->   adjCosts[1] = 2;
00125         node->adjHeadings[0] = 64;
00126         node->adjHeadings[1] = -64;
00127         break;
00128     case 7:
00129         node->numAdjNodes = 2;
00130         node->   adjNodes[0] = 27;
00131         node->   adjNodes[1] = 32;
00132         node->   adjCosts[0] = 3;
00133         node->   adjCosts[1] = 3;
00134         node->adjHeadings[0] = -128;
00135         node->adjHeadings[1] = 0;
00136         break;
00137     case 8:
00138         node->numAdjNodes = 2;
00139         node->   adjNodes[0] = 28;
00140         node->   adjNodes[1] = 30;
00141         node->   adjCosts[0] = 3;
00142         node->   adjCosts[1] = 3;
00143         node->adjHeadings[0] = -128;
00144         node->adjHeadings[1] = 0;
```

```
00145         break;
00146     case 9:
00147         node->numAdjNodes = 2;
00148         node->   adjNodes[0] = 22;
00149         node->   adjNodes[1] = 29;
00150         node->   adjCosts[0] = 3;
00151         node->   adjCosts[1] = 3;
00152         node->adjHeadings[0] = -128;
00153         node->adjHeadings[1] = 0;
00154         break;
00155     case 10:
00156         node->numAdjNodes = 2;
00157         node->   adjNodes[0] = 29;
00158         node->   adjNodes[1] = 30;
00159         node->   adjCosts[0] = 3;
00160         node->   adjCosts[1] = 3;
00161         node->adjHeadings[0] = -64;
00162         node->adjHeadings[1] = 64;
00163         break;
00164     case 11:
00165         node->numAdjNodes = 2;
00166         node->   adjNodes[0] = 12;
00167         node->   adjNodes[1] = 29;
00168         node->   adjCosts[0] = 4;
00169         node->   adjCosts[1] = 2;
00170         node->adjHeadings[0] = 0;
00171         node->adjHeadings[1] = -128;
00172         break;
00173     case 12:
00174         node->numAdjNodes = 2;
00175         node->   adjNodes[0] = 11;
00176         node->   adjNodes[1] = 33;
00177         node->   adjCosts[0] = 4;
00178         node->   adjCosts[1] = 2;
00179         node->adjHeadings[0] = -128;
00180         node->adjHeadings[1] = 0;
00181         break;
00182     case 13:
00183         node->numAdjNodes = 2;
00184         node->   adjNodes[0] = 33;
00185         node->   adjNodes[1] = 34;
00186         node->   adjCosts[0] = 2;
00187         node->   adjCosts[1] = 1;
00188         node->adjHeadings[0] = -64;
00189         node->adjHeadings[1] = 64;
00190         break;
00191     case 14:
00192         node->numAdjNodes = 2;
00193         node->   adjNodes[0] = 15;
00194         node->   adjNodes[1] = 34;
00195         node->   adjCosts[0] = 3;
00196         node->   adjCosts[1] = 4;
00197         node->adjHeadings[0] = S_EAST;
00198         node->adjHeadings[1] = N_WEST;
00199         break;
00200     case 15:
00201         node->numAdjNodes = 2;
00202         node->   adjNodes[0] = 14;
00203         node->   adjNodes[1] = 31;
00204         node->   adjCosts[0] = 3;
00205         node->   adjCosts[1] = 4;
00206         node->adjHeadings[0] = N_WEST;
00207         node->adjHeadings[1] = S_EAST;
00208         break;
00209     case 16:
00210         node->numAdjNodes = 2;
00211         node->   adjNodes[0] = 32;
00212         node->   adjNodes[1] = 40;
00213         node->   adjCosts[0] = 2;
00214         node->   adjCosts[1] = 2;
00215         node->adjHeadings[0] = -64;
00216         node->adjHeadings[1] = 64;
00217         break;
00218     case 17:
00219         node->numAdjNodes = 2;
00220         node->   adjNodes[0] = 34;
00221         node->   adjNodes[1] = 35;
00222         node->   adjCosts[0] = 3;
00223         node->   adjCosts[1] = 3;
00224         node->adjHeadings[0] = -64;
00225         node->adjHeadings[1] = 64;
```

```
00226         break;
00227     case 18:
00228         node->numAdjNodes = 2;
00229         node->   adjNodes[0] = 39;
00230         node->   adjNodes[1] = 40;
00231         node->   adjCosts[0] = 2;
00232         node->   adjCosts[1] = 2;
00233         node->adjHeadings[0] = 0;
00234         node->adjHeadings[1] = -128;
00235         break;
00236     case 19:
00237         node->numAdjNodes = 2;
00238         node->   adjNodes[0] = 20;
00239         node->   adjNodes[1] = 40;
00240         node->   adjCosts[0] = 4;
00241         node->   adjCosts[1] = 2;
00242         node->adjHeadings[0] = -128;
00243         node->adjHeadings[1] = 0;
00244         break;
00245     case 20:
00246         node->numAdjNodes = 2;
00247         node->   adjNodes[0] = 19;
00248         node->   adjNodes[1] = 41;
00249         node->   adjCosts[0] = 4;
00250         node->   adjCosts[1] = 2;
00251         node->adjHeadings[0] = 0;
00252         node->adjHeadings[1] = -128;
00253         break;
00254     case 21:                          // First Junction Node
00255         node->numAdjNodes = 2;
00256         node->   adjNodes[0] = 0;
00257         node->   adjNodes[1] = 1;
00258         node->   adjCosts[0] = 9;
00259         node->   adjCosts[1] = 4;
00260         node->adjHeadings[0] = 64;
00261         node->adjHeadings[1] = 0;
00262         break;
00263     case 22:
00264         node->numAdjNodes = 3;
00265         node->   adjNodes[0] = 1;
00266         node->   adjNodes[1] = 2;
00267         node->   adjNodes[2] = 9;
00268         node->   adjCosts[0] = 4;
00269         node->   adjCosts[1] = 2;
00270         node->   adjCosts[2] = 3;
00271         node->adjHeadings[0] = -128;
00272         node->adjHeadings[1] = 64;
00273         node->adjHeadings[2] = 0;
00274         break;
00275     case 23:
00276         node->numAdjNodes = 3;
00277         node->   adjNodes[0] = 2;
00278         node->   adjNodes[1] = 3;
00279         node->   adjNodes[2] = 28;
00280         node->   adjCosts[0] = 2;
00281         node->   adjCosts[1] = 2;
00282         node->   adjCosts[2] = 2;
00283         node->adjHeadings[0] = -64;
00284         node->adjHeadings[1] = -128;
00285         node->adjHeadings[2] = 64;
00286         break;
00287     case 24:                          // BONUS_BALL_1
00288         node->numAdjNodes = 2;
00289         node->   adjNodes[0] = 3;
00290         node->   adjNodes[1] = 4;
00291         node->   adjCosts[0] = 2;
00292         node->   adjCosts[1] = 3;
00293         node->adjHeadings[0] = 0;
00294         node->adjHeadings[1] = 64;
00295         break;
00296     case 25:
00297         node->numAdjNodes = 2;
00298         node->   adjNodes[0] = 4;
00299         node->   adjNodes[1] = 26;
00300         node->   adjCosts[0] = 3;
00301         node->   adjCosts[1] = 2;
00302         node->adjHeadings[0] = -64;
00303         node->adjHeadings[1] = 0;
00304         break;
00305     case 26:
00306         node->numAdjNodes = 3;
```

```
00307         node->   adjNodes[0] = 5;
00308         node->   adjNodes[1] = 25;
00309         node->   adjNodes[2] = 27;
00310         node->   adjCosts[0] = 2;
00311         node->   adjCosts[1] = 2;
00312         node->   adjCosts[2] = 2;
00313         node->adjHeadings[0] = 64;
00314         node->adjHeadings[1] = -128;
00315         node->adjHeadings[2] = 0;
00316         break;
00317     case 27:
00318         node->numAdjNodes = 3;
00319         node->   adjNodes[0] = 6;
00320         node->   adjNodes[1] = 7;
00321         node->   adjNodes[2] = 26;
00322         node->   adjCosts[0] = 2;
00323         node->   adjCosts[1] = 3;
00324         node->   adjCosts[2] = 2;
00325         node->adjHeadings[0] = -64;
00326         node->adjHeadings[1] = 0;
00327         node->adjHeadings[2] = -128;
00328         break;
00329     case 28:
00330         node->numAdjNodes = 3;
00331         node->   adjNodes[0] = 6;
00332         node->   adjNodes[1] = 8;
00333         node->   adjNodes[2] = 23;
00334         node->   adjCosts[0] = 2;
00335         node->   adjCosts[1] = 3;
00336         node->   adjCosts[2] = 2;
00337         node->adjHeadings[0] = 64;
00338         node->adjHeadings[1] = 0;
00339         node->adjHeadings[2] = -64;
00340         break;
00341     case 29:
00342         node->numAdjNodes = 3;
00343         node->   adjNodes[0] = 9;
00344         node->   adjNodes[1] = 10;
00345         node->   adjNodes[2] = 11;
00346         node->   adjCosts[0] = 3;
00347         node->   adjCosts[1] = 3;
00348         node->   adjCosts[2] = 2;
00349         node->adjHeadings[0] = -128;
00350         node->adjHeadings[1] = 64;
00351         node->adjHeadings[2] = 0;
00352         break;
00353     case 30:                            // BONUS_BALL_2
00354         node->numAdjNodes = 3;
00355         node->   adjNodes[0] = 8;
00356         node->   adjNodes[1] = 10;
00357         node->   adjNodes[2] = 31;
00358         node->   adjCosts[0] = 3;
00359         node->   adjCosts[1] = 3;
00360         node->   adjCosts[2] = 3;
00361         node->adjHeadings[0] = -128;
00362         node->adjHeadings[1] = -64;
00363         node->adjHeadings[2] = 64;
00364         break;
00365     case 31:
00366         node->numAdjNodes = 3;
00367         node->   adjNodes[0] = 15;
00368         node->   adjNodes[1] = 30;
00369         node->   adjNodes[2] = 32;
00370         node->   adjCosts[0] = 4;
00371         node->   adjCosts[1] = 3;
00372         node->   adjCosts[2] = 1;
00373         node->adjHeadings[0] = N_WEST;
00374         node->adjHeadings[1] = -64;
00375         node->adjHeadings[2] = 64;
00376         break;
00377     case 32:
00378         node->numAdjNodes = 3;
00379         node->   adjNodes[0] = 7;
00380         node->   adjNodes[1] = 16;
00381         node->   adjNodes[2] = 31;
00382         node->   adjCosts[0] = 3;
00383         node->   adjCosts[1] = 2;
00384         node->   adjCosts[2] = 1;
00385         node->adjHeadings[0] = -128;
00386         node->adjHeadings[1] = 64;
00387         node->adjHeadings[2] = -64;
```

```
00388         break;
00389     case 33:
00390         node->numAdjNodes = 2;
00391         node->   adjNodes[0] = 12;
00392         node->   adjNodes[1] = 13;
00393         node->   adjCosts[0] = 2;
00394         node->   adjCosts[1] = 2;
00395         node->adjHeadings[0] = -128;
00396         node->adjHeadings[1] = 64;
00397         break;
00398     case 34:
00399         node->numAdjNodes = 3;
00400         node->   adjNodes[0] = 13;
00401         node->   adjNodes[1] = 14;
00402         node->   adjNodes[2] = 17;
00403         node->   adjCosts[0] = 1;
00404         node->   adjCosts[1] = 4;
00405         node->   adjCosts[2] = 3;
00406         node->adjHeadings[0] = -64;
00407         node->adjHeadings[1] = S_EAST;
00408         node->adjHeadings[2] = 64;
00409         break;
00410     case 35:
00411         node->numAdjNodes = 2;
00412         node->   adjNodes[0] = 17;
00413         node->   adjNodes[1] = 36;
00414         node->   adjCosts[0] = 3;
00415         node->   adjCosts[1] = 2;
00416         node->adjHeadings[0] = -64;
00417         node->adjHeadings[1] = -128;
00418         break;
00419     case 36:
00420         node->numAdjNodes = 2;
00421         node->   adjNodes[0] = 35;
00422         node->   adjNodes[1] = 37;
00423         node->   adjCosts[0] = 2;
00424         node->   adjCosts[1] = 2;
00425         node->adjHeadings[0] = 0;
00426         node->adjHeadings[1] = 64;
00427         break;
00428     case 37:
00429         node->numAdjNodes = 2;
00430         node->   adjNodes[0] = 36;
00431         node->   adjNodes[1] = 38;
00432         node->   adjCosts[0] = 2;
00433         node->   adjCosts[1] = 2;
00434         node->adjHeadings[0] = -64;
00435         node->adjHeadings[1] = -128;
00436         break;
00437     case 38:
00438         node->numAdjNodes = 2;
00439         node->   adjNodes[0] = 37;
00440         node->   adjNodes[1] = 39;
00441         node->   adjCosts[0] = 2;
00442         node->   adjCosts[1] = 2;
00443         node->adjHeadings[0] = 0;
00444         node->adjHeadings[1] = 64;
00445         break;
00446     case 39:
00447         node->numAdjNodes = 2;
00448         node->   adjNodes[0] = 18;
00449         node->   adjNodes[1] = 38;
00450         node->   adjCosts[0] = 2;
00451         node->   adjCosts[1] = 2;
00452         node->adjHeadings[0] = -128;
00453         node->adjHeadings[1] = -64;
00454         break;
00455     case 40:
00456         node->numAdjNodes = 3;
00457         node->   adjNodes[0] = 16;
00458         node->   adjNodes[1] = 18;
00459         node->   adjNodes[2] = 19;
00460         node->   adjCosts[0] = 2;
00461         node->   adjCosts[1] = 2;
00462         node->   adjCosts[2] = 2;
00463         node->adjHeadings[0] = -64;
00464         node->adjHeadings[1] = 0;
00465         node->adjHeadings[2] = -128;
00466         break;
00467     case 41:
00468         node->numAdjNodes = 3;
```

```
00469         node->   adjNodes[0] = 5;
00470         node->   adjNodes[1] = 20;
00471         node->   adjNodes[2] = 42;
00472         node->   adjCosts[0] = 2;
00473         node->   adjCosts[1] = 2;
00474         node->   adjCosts[2] = 5;
00475         node->adjHeadings[0] = -64;
00476         node->adjHeadings[1] = 0;
00477         node->adjHeadings[2] = -128;
00478         break;
00479     case 42:                          // STOP_NODE
00480         node->numAdjNodes = 1;
00481         node->   adjNodes[0] = 41;
00482         node->   adjCosts[0] = 5;
00483         node->adjHeadings[0] = 0;
00484         break;
00485     }
00486 }
```

9.35 nodeList.h File Reference

Course defined by a connected grid of nodes.

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for nodeList.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct nodeStruct

Macros

- #define **NUM_NODES** 43
- #define **BALL_NODE_MIN** 1
- #define **BALL_NODE_MAX** 20
- #define **JUNCTION_MIN** 21
- #define **JUNCTION_MAX** 41
- #define **NUM_BALL_NODES** (BALL_NODE_MAX - BALL_NODE_MAX + 1)
- #define **MAX_ADJ_NODES** 3
- #define **N_WEST** -41
- #define **S_EAST** 87
- #define **BONUS_BALL_1** 24
- #define **BONUS_BALL_2** 30
- #define **SENSOR_NODE** 37
- #define **BB1_HEADING** 32
- #define **BB2_HEADING** -96
- #define **NUM_FIXED_GOALS** 3
- #define **NUM_RANDOM_GOALS** 3
- #define **NUM_GOALS** NUM_FIXED_GOALS + NUM_RANDOM_GOALS

Typedefs

- typedef struct nodeStruct **NODE**

Functions

- bool **isJunction** (uint8_t nodeNum)
- uint8_t **getCostToNode** (NODE *node, uint8_t nodeNum)
- uint8_t **getNodeAtHeading** (NODE *node, int8_t heading)
- bool **isBallNode** (uint8_t nodeNum)
- void **getNode** (uint8_t nodeNum, NODE *node)

9.35.1 Detailed Description

Course defined by a connected grid of nodes. Conserves SRAM by storing graph of arena in FLASH memory. See doc directory for image of arena with node numbers.

- Nodes are represented by numbers:
  - Nodes 0 and 42 are terminal nodes
  - Nodes 1-20 are ball nodes
  - Nodes 21-41 are junctions
- Distance resolution is 6 inches. -Direction is measured in binary radians or brads. (see www.urcp.com)

Version History: 2/17/05 - Created by Logan 2/21/05 - Checked by Logan, Scott, and Patrick

- Changed syntax for Atmel - Logan
- Added more defines - Logan 4/11/05 - Re-structured for FLASH - Logan

Definition in file nodeList.h.

9.36 nodeList.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00038 #ifndef NODELIST_H_
00039 #define NODELIST_H_
00040
00041 #include <stdint.h>
00042 #include <stdbool.h>
00043
00044 #define NUM_NODES       43        // number of nodes in arena
00045
00046 #define BALL_NODE_MIN   1         // ball node number range
00047 #define BALL_NODE_MAX   20
00048 #define JUNCTION_MIN    21        // junction node number range
00049 #define JUNCTION_MAX    41
00050
00051 #define NUM_BALL_NODES  (BALL_NODE_MAX - BALL_NODE_MAX + 1)
00052
00053
```

```
00054 #define MAX_ADJ_NODES   3         // max. nodes that can be adjacent to one
       node
00055 #define N_WEST          -41       // direction of north west in binary radians
       (brads)
00056 #define S_EAST          87        // direction of south east in binary radians
       (brads)
00057
00058 #define BONUS_BALL_1    24
00059 #define BONUS_BALL_2    30
00060 #define SENSOR_NODE     37
00061
00062 #define BB1_HEADING     32
00063 #define BB2_HEADING     -96
00064
00065 #define NUM_FIXED_GOALS    3
00066 #define NUM_RANDOM_GOALS   3
00067 #define NUM_GOALS          NUM_FIXED_GOALS + NUM_RANDOM_GOALS
00068
00069 typedef struct nodeStruct
00070 {
00071     uint8_t numAdjNodes;                  // number of nodes adjacent to this
       node
00072     uint8_t adjNodes[MAX_ADJ_NODES];      // node numbers of adjacent nodes
00073     uint8_t adjCosts[MAX_ADJ_NODES];      // distances to adjacent nodes (6
       inches increments)
00074     int8_t adjHeadings[MAX_ADJ_NODES];   // directions towards adjacent nodes
       (brads)
00075 } NODE;
00076
00077
00078 inline bool isJunction( uint8_t nodeNum );
00079 uint8_t getCostToNode(NODE *node, uint8_t nodeNum);
00080 uint8_t getNodeAtHeading(NODE *node, int8_t heading);
00081 inline bool isBallNode( uint8_t nodeNum );
00082 void getNode( uint8_t nodeNum, NODE *node );
00083
00084
00085 #endif // #ifndef NODELIST_H_
```

9.37 perms.h File Reference

Iterative (non-recursive!) permutation generator.

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for perms.h: This graph shows which files directly or indirectly include this file:

Functions

- bool generateNextPermutation (uint8_t *first, uint8_t *last)

  Reorder an array of values to the next higher permutation.

9.37.1 Detailed Description

Iterative (non-recursive!) permutation generator.

Definition in file perms.h.

9.37.2 Function Documentation

9.37.2.1 bool generateNextPermutation (uint8_t * *first*, uint8_t * *last*)

Reorder an array of values to the next higher permutation.

The "next higher" permuation is the one that is lexicographically one step higher than the input order. The order that would compare smaller to all other permutations is the one in which all elements are sorted in ascending order. This is the initial order that should be used in order to cycle through all possible permutations.

Typical usage example:

```
uint8_t myArray[] = { 1, 2, 3 };
do {
    // ... do something with current permuation of myArray
} while (generateNextPermutation(myArray, myArray + 3);
```

Remarks

This iterative permutation generation algorithm was taken, with slight modifications, from the GNU implementation of the C++ STL (libstdc++). It was chosen for for its lower memory usage over simpler and more common recursive implementations.

Returns

true if the next higher permutation could be generated, false otherwise

Definition at line 22 of file perms.c.

9.38 perms.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00021 #ifndef PERMS_H_
00022 #define PERMS_H_
00023
00024 #include <stdint.h>
00025 #include <stdbool.h>
00026
00053 bool generateNextPermutation(uint8_t *first, uint8_t *last);
00054
00055 #endif // #ifndef PERMS_H_
```

9.39 tetherUI.h File Reference

Simple user interface to change parameters without reprogramming.

This graph shows which files directly or indirectly include this file:

Macros

- #define NAV_LCD_MODE 0
- #define LINE_LCD_MODE 1

Functions

- void runTetherUI (void)

  Allow tweaking via tether remote until red button pressed.

9.39.1 Detailed Description

Simple user interface to change parameters without reprogramming. The user interface is implemented as push buttons and LEDs on a small solder-less breadboard connected to Caddy using CAT5 cable and RJ-45 connector for quick and easy attach/detach.

Definition in file tetherUI.h.

9.39.2 Macro Definition Documentation

9.39.2.1 #define LINE_LCD_MODE 1

Special LCD display mode for debugging line tracking

Definition at line 36 of file tetherUI.h.

9.39.2.2 #define NAV_LCD_MODE 0

Default LCD display mode

Definition at line 31 of file tetherUI.h.

9.40 tetherUI.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00025 #ifndef TETHERUI_H_
00026 #define TETHERUI_H_
00027
00031 #define NAV_LCD_MODE        0
00032
00036 #define LINE_LCD_MODE       1
00037
00041 inline void runTetherUI(void);
00042
00043 #endif // #ifndef TETHERUI_H_
```

9.41 trackColor.c File Reference

```
#include "trackColor.h"
```

```
#include "trackLine.h"
#include "camera.h"
#include "servos.h"
#include "junctionCode.h"
#include "motorCntrl.h"
#include "eeProm.h"
#include "helperFunctions.h"
#include "rprintf.h"
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for trackColor.c:

Macros

- #define **BALL_RMIN** 150
- #define **BALL_RMAX** 240
- #define **BALL_GMIN** 16
- #define **BALL_GMAX** 60
- #define **BALL_BMIN** 16
- #define **BALL_BMAX** 50

Functions

- void **trackColorInit** (int8_t dir)
- uint8_t **getBallY** (void)
- bool **seeBall** (void)
- bool **cameraSeekLeft** (uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls)
- bool **cameraSeekRight** (uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls)

Variables

- volatile bool **colorStatsProcessed**
- bool **inSeekPosition**

9.41.1 Detailed Description

Definition in file trackColor.c.

9.42 trackColor.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
```

```
00016  */
00018 #include "trackColor.h"
00019 #include "trackLine.h"
00020 #include "camera.h"
00021 #include "servos.h"
00022 #include "junctionCode.h"
00023 #include "motorCntrl.h"
00024 #include "eeProm.h"
00025 #include "helperFunctions.h"
00026
00027 // AVRLIB
00028 #include "rprintf.h"
00029
00030 // avr-libc
00031 #include <stdint.h>
00032 #include <stdbool.h>
00033
00034 // Track the RED ball on black/white background
00035 #define BALL_RMIN    150
00036 #define BALL_RMAX    240
00037 #define BALL_GMIN    16
00038 #define BALL_GMAX    60
00039 #define BALL_BMIN    16
00040 #define BALL_BMAX    50
00041
00042 // Global variables
00043 volatile bool colorStatsProcessed;
00044 bool inSeekPosition;
00045
00046 static uint8_t distToPix( uint8_t distance );
00047
00048 void trackColorInit(int8_t dir)
00049 {
00050     if (!inSeekPosition)
00051     {
00052         brake(BOTH);
00053         msDelay(200);
00054         moveStraight(-1 * 0xb, 255);
00055         inSeekPosition = true;
00056     }
00057
00058     // Set pan (center) and tilt
00059     switch (dir)
00060     {
00061     case LOOK_LEFT:
00062         setServo(PAN, PAN_CENTER + panOffset + PAN_SEEK_OFFSET);
00063         setServo(TILT, TILT_VERT + tiltOffset);
00064         break;
00065     case LOOK_RIGHT:
00066         setServo(PAN, PAN_CENTER + panOffset - PAN_SEEK_OFFSET);
00067         setServo(TILT, TILT_VERT + tiltOffset);
00068         break;
00069     case LOOK_UP:
00070         setServo(PAN, PAN_CENTER + panOffset);
00071         setServo(TILT, TILT_LOOKUP);
00072         break;
00073     default:
00074         break;
00075     }
00076     msDelay(500);
00077
00078     hiResMode();
00079     rprintf("DS 1 1\r");
00080     rprintf("LM 0 0\r");
00081
00082     // Change to poll mode so only one packet is sent
00083     rprintf("PM 1\r");
00084 }
00085
00086 /*
00087  * Returns Y1 (top of ball) if camera sees a ball, zero otherwise
00088  */
00089 uint8_t getBallY( void )
00090 {
00091     rprintf("lm 0 0\r");
00092
00093     // Mask everything but the 'My' value
00094     //rprintf("OM 0 2\r");                              //<- NO MASKING?
00095
00096     // Change to poll mode so only one packet is sent
00097     rprintf("PM 1\r");
```

```
00098
00099     // Track red
00100     rprintf("TC %d %d %d %d %d %d\r",
00101                     BALL_RMIN, BALL_RMAX, BALL_GMIN, BALL_GMAX, BALL_BMIN,
      BALL_BMAX);
00102
00103     colorStatsProcessed = true;
00104     while (colorStatsProcessed) ;
00105
00106     return (lineStats[0][Y1_NDX]);
00107 }
00108
00109
00110 bool seeBall( void )
00111 {
00112    // Track red
00113    rprintf("TC %d %d %d %d %d %d\r",
00114            BALL_RMIN, BALL_RMAX, BALL_GMIN, BALL_GMAX, BALL_BMIN, BALL_BMAX);
00115    colorStatsProcessed = true;
00116    while (colorStatsProcessed) ;
00117
00118    return lineStats[0][Y1_NDX] > 0;
00119 }
00120
00121
00122 /*
00123  * Just does left seeks
00124  *    PRE - the longest check is the last element of the uncheckedBalls array
00125  *
00126  *    uncheckedBalls - ball node numbers and ground distances away from bot
00127  */
00128 bool cameraSeekLeft( uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls )
00129 {
00130     bool foundBall = false;    // Return value
00131     uint8_t scanHeight = 4;
00132     uint8_t x = 174;
00133     //uint8_t ballDist[3];
00134     //uint8_t ballCount = 0;
00135     uint8_t scanLimit = distToPix(
00136                     uncheckedBalls[numUncheckedBalls - 1][BALL_DIST] + 1);
00137
00138     // get pixel ranges for unchecked balls passed in
00139     uint8_t i = 0;
00140     uint8_t maxBallX[3];
00141     while (i + 1 < numUncheckedBalls)
00142     {
00143         maxBallX[i] = (distToPix(uncheckedBalls[i][BALL_DIST]) +
00144                        distToPix(uncheckedBalls[i + 1][BALL_DIST])) / 2;
00145         i++;
00146     }
00147     maxBallX[i] = scanLimit;
00148
00149     /*
00150      #if DEBUGGING
00151      lcdWriteStr("maxBallX =      ",0,0);
00152      lcdWriteStr("                ",1,0);
00153      for( i = 0; i < numUncheckedBalls; i++ )
00154      {
00155      lcdPrintHex(maxBallX[i],1,3*i);
00156      }
00157      waitFor(RED_BUTTON);
00158      #endif
00159      */
00160
00161     // scan from small ground distance to large ground distance
00162     while (x - scanHeight > scanLimit)
00163     {
00164         x -= scanHeight;
00165         setVirtualWindow(x - scanHeight, 1, x, 254);
00166         if (seeBall())
00167         {
00168             foundBall = true;
00169             //ballDist[ballCount++] = xToDist(x);
00170
00171             // find ball number of ball at this x
00172             i = 0;
00173             while (maxBallX[i] > x)
00174             {
00175                 i++;
00176             }
00177             addToGoalList(uncheckedBalls[i][BALL_NODE_NUM]);
```

```
00178
00179 #if DEBUGGING
00180             labelColorStats();
00181             refreshColorStats();
00182 #endif
00183
00184             /*
00185              #if DEBUGGING
00186              lcdWriteStr("Added:          ",0,0);
00187              lcdWriteStr("                ",1,0);
00188              lcdPrintHex(uncheckedBalls[i][BALL_NODE_NUM],1,0);
00189              waitFor(RED_BUTTON);
00190              #endif
00191              */
00192
00193             while (seeBall())
00194             {
00195                 x -= scanHeight;
00196                 setVirtualWindow(x - scanHeight, 1, x, 254);
00197             }
00198         }
00199     }
00200
00201     return foundBall;
00202 }
00203
00204 // returns pixel equivalent of 'distance'
00205 static uint8_t distToPix( uint8_t distance )
00206 {
00207     switch (distance)
00208     {
00209     case 0:
00210     case 1:
00211         return 174;
00212     case 2:
00213         return 0x8d;
00214     case 3:
00215         return 0x61;
00216     case 4:
00217         return 0x48;
00218     case 5:
00219         return 0x36;
00220     case 6:
00221         return 0x2b;
00222     case 7:
00223         return 0x22;
00224     case 8:
00225         return 0x1d;
00226     case 9:
00227         return 0x18;
00228     case 10:
00229         return 0x14;
00230     case 11:
00231         return 0x11;
00232     case 12:
00233         return 0x0e;
00234     default:
00235         return 0x0;
00236     }
00237 }
00238
00239 /*
00240  * Just does right seeks
00241  *    PRE - the longest check is the last element of the uncheckedBalls array
00242  *
00243  *    uncheckedBalls - ball node numbers and ground distances away from bot
00244  */
00245 bool cameraSeekRight(uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls)
00246 {
00247     bool foundBall = false;    // Return value
00248     uint8_t scanHeight = 4;
00249     uint8_t x = 0;
00250     uint8_t scanLimit = 174 - distToPix(
00251                     uncheckedBalls[numUncheckedBalls - 1][BALL_DIST] + 1);
00252
00253     // get pixel ranges for unchecked balls passed in
00254     uint8_t i = 0;
00255     uint8_t maxBallX[3];
00256     while (i + 1 < numUncheckedBalls)
00257     {
00258         maxBallX[i] = ((174 - distToPix(uncheckedBalls[i][BALL_DIST])) +
```

```
00259                       (174 - distToPix(uncheckedBalls[i + 1][BALL_DIST]))
00260                       / 2;
00261         i++;
00262     }
00263     maxBallX[i] = scanLimit;
00264
00265 /*
00266 #if DEBUGGING
00267     lcdWriteStr("maxBallX =      ", 0, 0);
00268     lcdWriteStr("                ", 1, 0);
00269     for (i = 0; i < numUncheckedBalls; i++)
00270     {
00271         lcdPrintHex(maxBallX[i], 1, 3 * i);
00272     }
00273     waitFor(RED_BUTTON);
00274 #endif
00275 */
00276
00277     // scan from small ground distance to large ground distance
00278     while (x + scanHeight < scanLimit)
00279     {
00280         x += scanHeight;
00281         setVirtualWindow(x, 1, x + scanHeight, 254);
00282         if (seeBall())
00283         {
00284             foundBall = true;
00285             //ballDist[ballCount++] = xToDist(x);
00286
00287             // find ball number of ball at this x
00288             i = 0;
00289             while (maxBallX[i] < x)
00290             {
00291                 i++;
00292             }
00293             addToGoalList(uncheckedBalls[i][BALL_NODE_NUM]);
00294
00295 #if DEBUGGING
00296             labelColorStats();
00297             refreshColorStats();
00298 #endif
00299
00300 /*
00301 #if DEBUGGING
00302             msDelay(1000);
00303             clearColorStats();
00304             lcdWriteStr("Added:          ", 0, 0);
00305             lcdWriteStr("                ", 1, 0);
00306             lcdPrintHex(uncheckedBalls[i][BALL_NODE_NUM], 1, 0);
00307             waitFor(RED_BUTTON);
00308 #endif
00309 */
00310
00311             while (seeBall())
00312             {
00313                 x += scanHeight;
00314                 setVirtualWindow(x, 1, x + scanHeight, 254);
00315             }
00316         }
00317     }
00318
00319    return foundBall;
00320 }
00321
```

9.43 trackColor.h File Reference

Simple tracking Roborodentia objects of interest by color.

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for trackColor.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **LOOK_RIGHT** 1
- #define **LOOK_LEFT** -1
- #define **LOOK_UP** 0
- #define **MX_NDX** 0
- #define **MY_NDX** 1
- #define **X1_NDX** 2
- #define **Y1_NDX** 3
- #define **X2_NDX** 4
- #define **Y2_NDX** 5
- #define **PIXEL_CNT_NDX** 6
- #define **CONFIDENCE_NDX** 7
- #define **NUM_COLOR_STATS** 8
- #define **PAN_SEEK_OFFSET** 66

Functions

- void **trackColorInit** (int8_t dir)
- uint8_t **getBallY** (void)
- bool **seeBall** (void)
- bool **cameraSeekLeft** (uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls)
- bool **cameraSeekRight** (uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls)

Variables

- volatile bool **colorStatsProcessed**
- bool **inSeekPosition**

9.43.1 Detailed Description

Simple tracking Roborodentia objects of interest by color. Uses the CMUcam2 color blob tracking to:

- Identify ball and estimate distance from robot
- Identify nest

Definition in file trackColor.h.

9.44 trackColor.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
```

```
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00025 #ifndef TRACKCOLOR_H_
00026 #define TRACKCOLOR_H_
00027
00028 // avr-libc
00029 #include <stdint.h>
00030 #include <stdbool.h>
00031
00032 #define LOOK_RIGHT   1
00033 #define LOOK_LEFT    -1
00034 #define LOOK_UP      0
00035
00036 #define MX_NDX            0
00037 #define MY_NDX            1
00038 #define X1_NDX            2
00039 #define Y1_NDX            3
00040 #define X2_NDX            4
00041 #define Y2_NDX            5
00042 #define PIXEL_CNT_NDX     6
00043 #define CONFIDENCE_NDX    7
00044
00045 #define NUM_COLOR_STATS    8
00046
00047 #define PAN_SEEK_OFFSET    66
00048
00049 // Global variables
00050 extern volatile bool colorStatsProcessed;
00051 extern bool inSeekPosition;
00052
00053 void trackColorInit(int8_t dir);
00054 uint8_t getBallY( void );
00055 bool seeBall( void );
00056 bool cameraSeekLeft( uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls );
00057 bool cameraSeekRight( uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls );
00058
00059 #endif // #ifndef TRACKCOLOR_H_
```

9.45 trackLine.h File Reference

Line detection and PID tracking using CMUcam2.

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for trackLine.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **DS_X_LINE** 1
- #define **DS_Y_LINE** 4
- #define **VW_X1_LINE** 10
- #define **VW_Y1_LINE** 1
- #define **VW_X2_LINE** 77
- #define **VW_Y2_LINE** 35
- #define **VW_X_SIZE_LINE** (VW_X2_LINE - VW_X1_LINE + 1)
- #define **VW_Y_SIZE_LINE** (VW_Y2_LINE - VW_Y1_LINE + 1)
- #define **LINE_STATS_ROWS** VW_Y_SIZE_LINE
- #define **LINE_STATS_COLS** 4

Functions

- void **adjustPWM** (void)
- void **trackLineInit** (void)
- void **restartLineMode** (void)

- void **analyzeLineStats** (void)
- bool **isGoodScan** (uint8_t y)
- bool **isJunctionScan** (uint8_t y)
- bool **mayBeBallScan** (uint8_t y)
- void **printPacket** (void)

Variables

- int8_t **junctionY**
- volatile uint8_t **lineStats** [LINE_STATS_ROWS][LINE_STATS_COLS]
- volatile bool **lineStatsProcessed**

9.45.1 Detailed Description

Line detection and PID tracking using CMUcam2.

Definition in file trackLine.h.

9.46 trackLine.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00021 #ifndef TRACKLINE_H_
00022 #define TRACKLINE_H_
00023
00024 #include <stdint.h>
00025 #include <stdbool.h>
00026
00027 #define DS_X_LINE             1
00028 #define DS_Y_LINE             4
00029 #define VW_X1_LINE            10
00030 #define VW_Y1_LINE            1
00031 #define VW_X2_LINE            77
00032 #define VW_Y2_LINE            35
00033 #define VW_X_SIZE_LINE        (VW_X2_LINE - VW_X1_LINE + 1)
00034 #define VW_Y_SIZE_LINE        (VW_Y2_LINE - VW_Y1_LINE + 1)
00035
00036 #define LINE_STATS_ROWS       VW_Y_SIZE_LINE
00037 #define LINE_STATS_COLS       4    // must correspond to bits in LINE_STAT_MASK
00038
00039 void adjustPWM( void );
00040
00041 void trackLineInit(void);
00042 void restartLineMode(void);
00043
00044 void analyzeLineStats(void);
00045 bool isGoodScan(uint8_t y);
00046 bool isJunctionScan(uint8_t y);
00047 bool mayBeBallScan(uint8_t y);
00048
00049 void printPacket(void);
00050
00051 extern int8_t junctionY;
00052
```

```
00053 // Global variables
00054 extern volatile uint8_t lineStats[LINE_STATS_ROWS][LINE_STATS_COLS];
00055 extern volatile bool lineStatsProcessed;
00056
00057 #endif // #ifndef TRACKLINE_H_
```

References

[1] C. Brent Dane. The reverser: Servo reversing for y-cord operation. 1997.

[2] Dafydd Walters. Implementing dead reckoning by odometry on a robot with r/c servo differential drive. September 2000.

Index