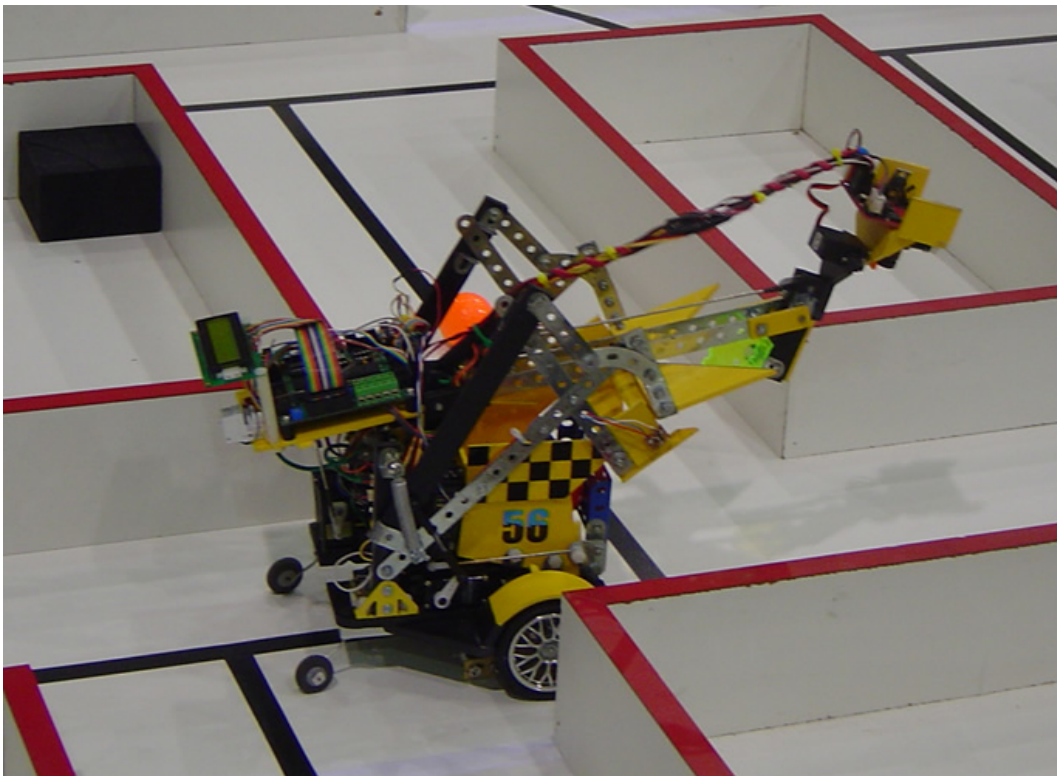


Caddy

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
1.1	Problem Summary	1
2	Goals	2
3	Design	2
3.1	Collaborative Team Process	2
3.2	Collaborative Tools	2
3.3	System Architecture	3
3.4	Electrical Design	3
3.4.1	Power Regulation and Motor Controller	3
3.4.2	IR Break Beam	3
3.4.3	Servo Reverser	3
3.4.4	Wheel Encoders	4
3.5	Software Architecture	4
3.5.1	Computing Platform	5
3.5.2	PID Line Tracking	5
3.5.3	Maneuvering	6
3.5.4	Ball Detection and Localization	6
3.5.5	Path Planning	6
4	Conclusion	6
4.1	Future Work	6
4.1.1	Regulated Motor Voltage	7
4.1.2	7
4.1.3	Quadrature Wheel Encoders	7
5	Appendix	7
5.1	Bill of Materials	7
5.2	Gantt Chart	8
5.3	Individual Contributions	8
5.3.1	Contributions of Taylor Braun-Jones	9
6	Data Structure Index	10
6.1	Data Structures	10
7	File Index	10
7.1	File List	10

8	Data Structure Documentation	12
8.1	nodeStruct Struct Reference	12
8.1.1	Detailed Description	12
8.2	PathList Struct Reference	12
8.2.1	Detailed Description	12
8.3	searchNode Struct Reference	12
8.3.1	Detailed Description	13
8.4	struct_EncoderState Struct Reference	13
8.4.1	Detailed Description	13
9	File Documentation	13
9.1	botCntrl.c File Reference	13
9.1.1	Detailed Description	14
9.2	botCntrl.c	14
9.3	botCntrl.h File Reference	21
9.3.1	Detailed Description	21
9.4	botCntrl.h	21
9.5	buttons.c File Reference	22
9.5.1	Detailed Description	22
9.5.2	Function Documentation	22
9.6	buttons.c	23
9.7	buttons.h File Reference	25
9.7.1	Detailed Description	25
9.7.2	Function Documentation	25
9.8	buttons.h	26
9.9	caddy.c File Reference	27
9.9.1	Detailed Description	27
9.9.2	Macro Definition Documentation	27
9.10	caddy.c	27
9.11	camera.c File Reference	29
9.11.1	Detailed Description	29
9.11.2	Function Documentation	29
9.12	camera.c	30
9.13	camera.h File Reference	32
9.13.1	Detailed Description	32
9.13.2	Function Documentation	32
9.14	camera.h	33

9.15 eeProm.c File Reference	33
9.15.1 Detailed Description	34
9.16 eeProm.c	34
9.17 eeProm.h File Reference	36
9.17.1 Detailed Description	37
9.18 eeProm.h	37
9.19 encoder.c File Reference	38
9.19.1 Detailed Description	39
9.20 encoder.c	39
9.21 encoder.h File Reference	42
9.21.1 Detailed Description	42
9.22 encoder.h	43
9.23 encoderconf.h File Reference	44
9.23.1 Detailed Description	45
9.24 encoderconf.h	45
9.25 exercises.c File Reference	48
9.25.1 Detailed Description	48
9.26 exercises.c	48
9.27 exercises.h File Reference	56
9.27.1 Detailed Description	56
9.28 exercises.h	56
9.29 junctionCode.c File Reference	56
9.29.1 Detailed Description	57
9.29.2 Variable Documentation	57
9.30 junctionCode.c	57
9.31 junctionCode.h File Reference	61
9.31.1 Detailed Description	62
9.32 junctionCode.h	62
9.33 nodeList.c File Reference	62
9.33.1 Detailed Description	63
9.34 nodeList.c	63
9.35 nodeList.h File Reference	69
9.35.1 Detailed Description	70
9.36 nodeList.h	70
9.37 perms.h File Reference	71
9.37.1 Detailed Description	71
9.37.2 Function Documentation	71

9.38	perms.h	72
9.39	tetherUI.h File Reference	72
9.39.1	Detailed Description	73
9.39.2	Macro Definition Documentation	73
9.40	tetherUI.h	73
9.41	trackColor.c File Reference	73
9.41.1	Detailed Description	74
9.42	trackColor.c	74
9.43	trackColor.h File Reference	78
9.43.1	Detailed Description	79
9.44	trackColor.h	79
9.45	trackLine.h File Reference	80
9.45.1	Detailed Description	81
9.46	trackLine.h	81

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.

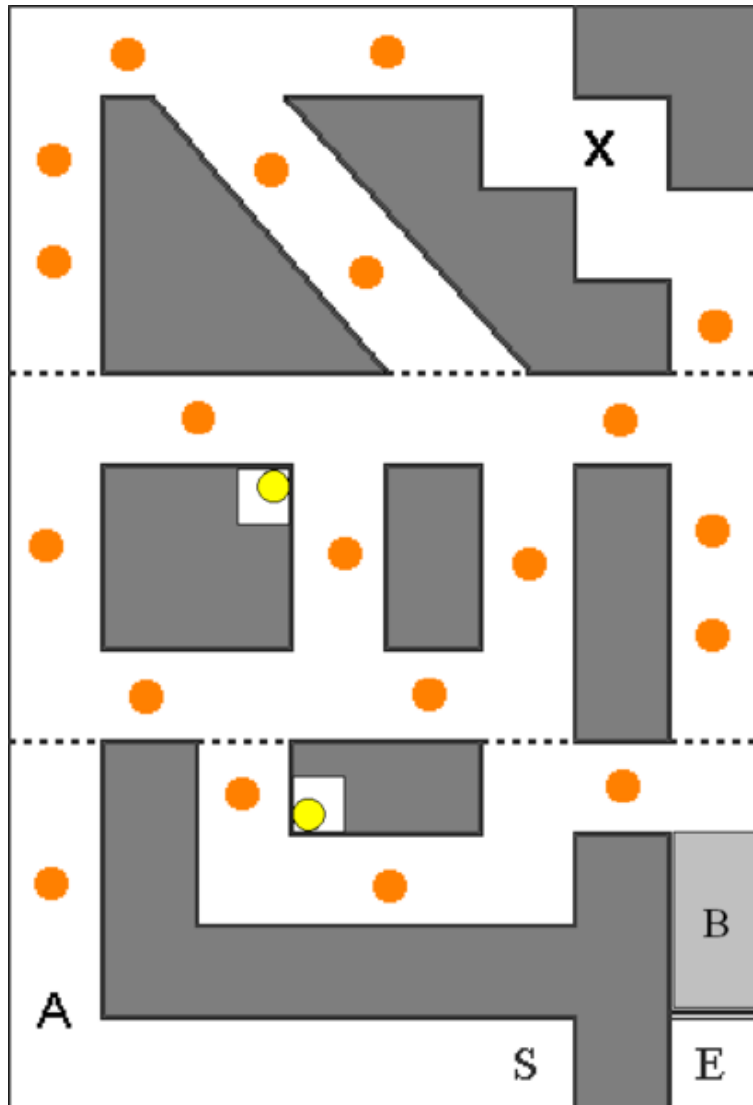


Figure 1: Arena map showing the two fixed bonus ball locations and the potential locations of the randomly placed ground balls

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

For our entry 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 two of Caddy's unique 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.

Since the scoring distribution weighted bonus balls so heavily, we also decided that Caddy would need bonus ball collection capability in order to be competitive.

3 Design

3.1 Collaborative Team Process

3.2 Collaborative Tools

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 over some of the free CVS hosting services found on the internet.

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

Inline code documentation as well as this project report were 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 is "close to the code".

3.3 System Architecture

Info about system architecture

3.4 Electrical Design

3.4.1 Power Regulation and Motor Controller

Resolving software bugs and electrical noise issues at the same time is almost impossible, so we wanted to build a reliable power supply board to provide clean regulated power to the electronics, free of all the back EMF generated by the DC motors. We came up with a simple design to provide

- Raw, unregulated battery voltage for the motors via an H-bridge driver
- Decoupled, unregulated power for the CMUcam
- Decoupled, regulated power for the rest of the electronics

Using unregulated batter power to drive the motors meant we could save on the cost of an expensive voltage capable of driving the relatively high power demands of the DC motors.

We made sure to use polarized headers for all the connections because at 5 in the morning after a long coding session you are liable to make all kinds of stupid blunders. Even with prevention measures like this, we made a few.

3.4.2 IR Break Beam

To detect when a ball is within the grasp of the lift we had two options. Originally we thought that we could simply use the centroid tracking feature of the camera since we would have the camera facing down watching the line anyway. This turned out to be difficult for a couple reasons.

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 sees – a gap in the line. This seemed like an easily surmountable problem at first. Just change modes whenever a gap is detected, determine if it is a ball or a glare, and act accordingly. As with any software program, introducing one seemingly small change has the potential to severely affect the rest of the system. This particular case was no exception. First, the CMUcam did not handle rapid mode/parameter changes well, taking longer than we expected to go from one mode to another. This lead to a failure in our finely 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 and still much to do, we decided that the quickest way to solve the problem was to simply setup a break beam sensor in just the right position to detect when the lift mechanism should be raised. This was fast to implement and worked reliably.

Here is a schematic of our break beam circuit:

3.4.3 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. Tyson had already done such an awesome job of designing the lift to be actuated by just one mechanical motion that this seemed like too much to ask.
- **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.

We decided to use the 555 timer approach. Using plans found online, we fabricated the board with a 4-pin header so that the circuit could easily be reused in the future.

3.4.4 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. [\[1\]](#)

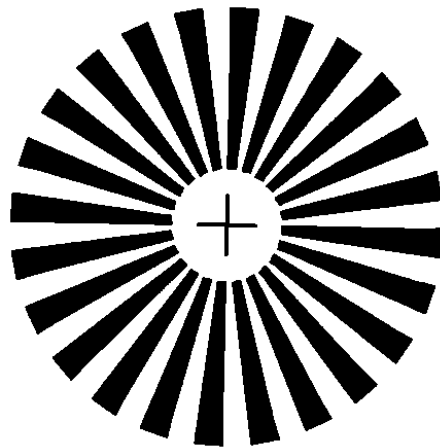


Figure 2: Reflective IR wheel encoder pattern

3.5 Software Architecture

3.5.1 Computing Platform

For 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 3: EMBMega32 development board from ERE CO.,LTD

3.5.2 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.

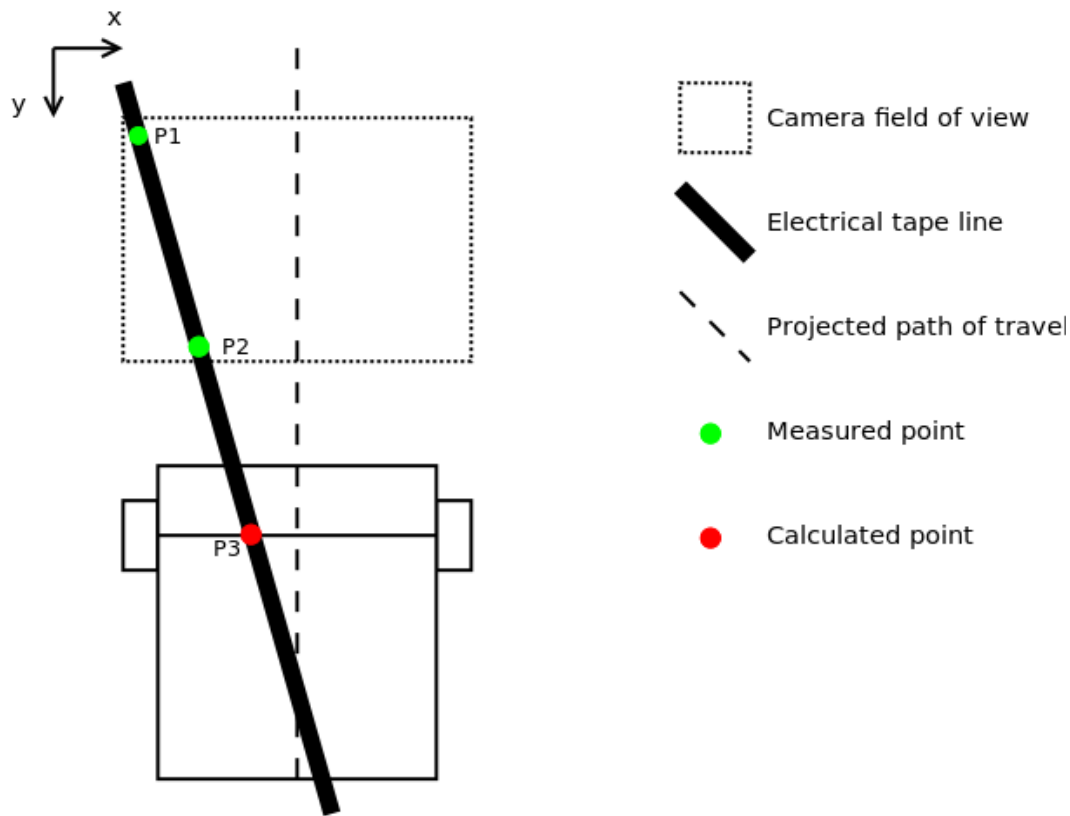


Figure 4: Diagram of line tracking geometry

3.5.3 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 a "reverse ticks" being counted before the wheel had actually started moving in the reverse direction.

3.5.4 Ball Detection and Localization

3.5.5 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 solved 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 Bill of Materials

The materials for this project were funded in part by a \$525 grant from the Mechanical Engineering Student Fee Allocation Committee (MESFAC)

5.2 Gantt Chart

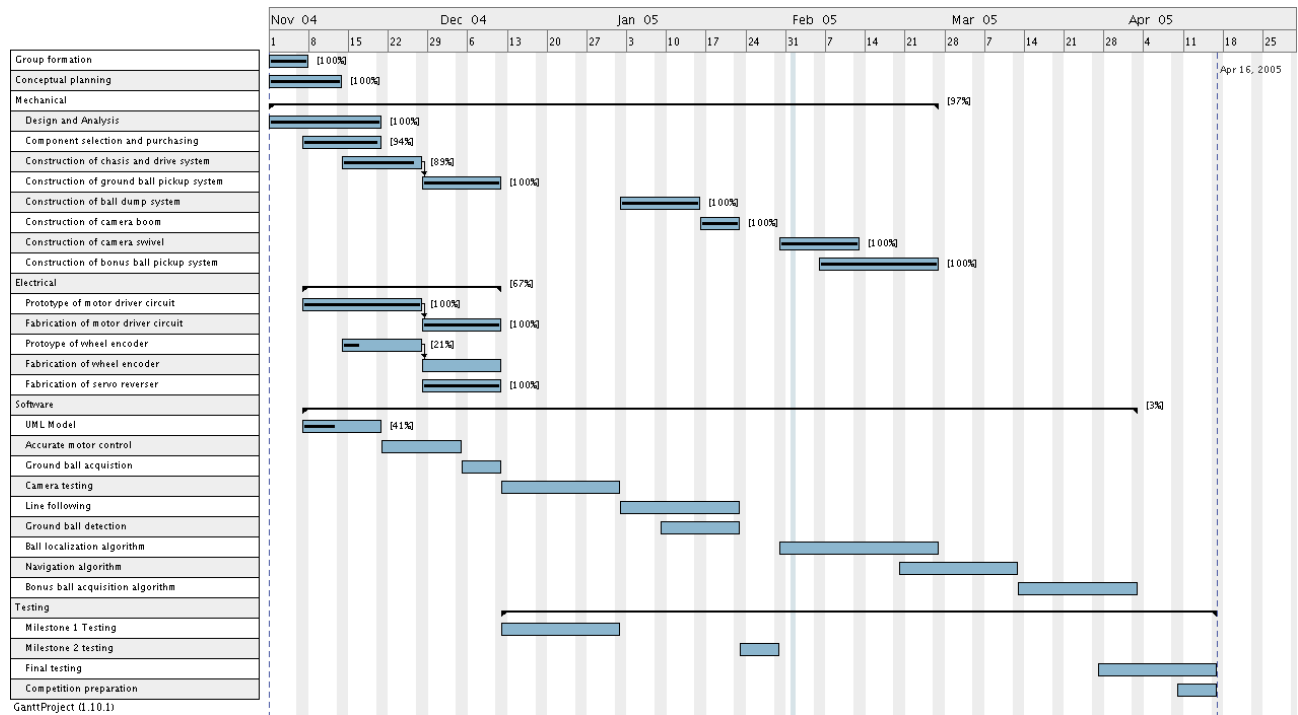


Figure 5: Caddy project gantt chart

5.3 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.

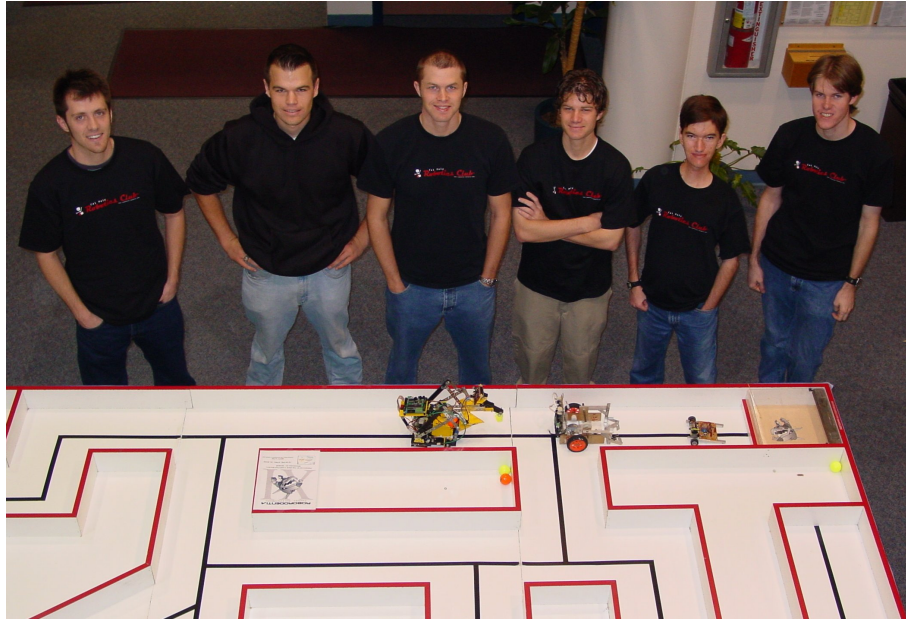


Figure 6: Team Photo. Left to right: Logan Kinde, Tyson Messori, Taylor Braun-Jones, Scott Barlow, Michael Shelley, Patrick McCarty

5.3.1 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 contributed the bracket used to mount the CMUcam2 to the panning servo. Nearly all the rest of the mechanical work was owned by Tyson Messori.

Taylor contributed 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

Nearly all the rest of the code (the majority of the code base) was developed between Taylor and Logan together using the **pair programming** technique. This includes:

- PID line tracking

- Ball detection and seeking
- Course traversal
- Ball collection maneuvers

6 Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

nodeStruct	12
PathList	12
searchNode	12
struct_EncoderState Encoder state structure	13

7 File Index

7.1 File List

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

botCntrl.c	14
botCntrl.h High-level logic controlling Caddy's actions	21
buttons.c	23
buttons.h Button debouncing, start bot logic	26
caddy.c Caddy's main loop and Atmel initialization	27
camera.c	30
camera.h	33
eeProm.c	34
eeProm.h Loading and store "tweak values" into eeProm	37
encoder.c Quadrature Encoder reader/driver	39
encoder.h Quadrature Encoder reader/driver	43

encoderconf.h	
Quadrature Encoder driver configuration	45
exercises.c	48
exercises.h	
Exercise various high-level capabilities	56
global.h	??
helperFunctions.c	??
helperFunctions.h	??
junctionCode.c	57
junctionCode.h	
Actions that occur at junctions	62
linkedList.c	??
linkedList.h	??
motorCntrl.c	??
motorCntrl.h	??
nodeList.c	63
nodeList.h	
Course defined by a connected grid of nodes	70
ourLCD.c	??
ourLCD.h	??
perms.c	??
perms.h	
Iterative (non-recursive!) permutation generator	72
servos.c	??
servos.h	??
testCode.c	??
testCode.h	??
tetherUI.c	??
tetherUI.h	
Simple user interface to change parameters without reprogramming	73
trackColor.c	74
trackColor.h	
Simple tracking Roborodentia objects of interest by color	79

trackLine.c	??
trackLine.h	
Line detection and PID tracking using CMUcam2	81
updatePath.c	??
updatePath.h	??

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 "servos.h"
#include "buttons.h"
#include "nodeList.h"
#include "tetherUI.h"
#include "eeProm.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  */
00017 #include "botCntrl.h"
00018 #include "trackLine.h"
00019 #include "trackColor.h"
00020 #include "junctionCode.h"
00021 #include "updatePath.h"
00022 #include "motorCntrl.h"
00023 #include "servos.h"
00024 #include "buttons.h"
00025 #include "nodeList.h"
00026 #include "tetherUI.h"
00027 #include "eeProm.h"
00028
00029 // avr-libc
00030 #include <string.h>
00031
00032 #define BEAM_IGNORE_COUNT 6
00033 #define CORRAL_COUNT 3
00034 #define LIFT_DONE_COUNT 8
00035
00036

```

```

00037 // Global variables
00038 uint8_t botNode = START_NODE;
00039 int8_t botHeading = START_HEADING;
00040 uint8_t numUnreachedGoals = NUM_GOALS;
00041
00042 static bool liftDown;
00043 static uint8_t upComingBallNum;
00044
00045 static inline int8_t getNextHeading(uint8_t nextBotNode);
00046
00047 inline void runRoborodentiaCourse(void)
00048 {
00049     bool justTurned = false;
00050     bool firstRun = true;
00051
00052     updatePath();
00053
00054     // run through first leg, skipping positionBot
00055     junctionCode();
00056     moveToJunction(1, justTurned);
00057
00058 #if DEBUGGING
00059     if (lcdMode == NAV_LCD_MODE)
00060     {
00061         lcdWriteStr("          ", 0, 0);
00062         lcdWriteStr("          ", 1, 0);
00063         lcdPrintDecU08(botNode, 1, 0);
00064         lcdPrintDecS08(botHeading, 1, 3);
00065     }
00066 #endif
00067
00068     // run through arena
00069     while (pathList[pathListIndex + 1] != STOP_NODE)
00070     {
00071         junctionCode(); // ball search, bonous ball pickup, best path
00072         code
00073         justTurned = positionBot(); // turning, preparing for linetracking
00074
00075         if (firstRun)
00076         {
00077             firstRun = false;
00078             setServo(LIFT, LIFT_OPEN); // Lower lift, on first run, b/c
00079             skipping seek at node 21
00080             msDelay(30);
00081             upComingBallNum = 1;
00082             liftDown = true;
00083         }
00084
00085 #if DEBUGGING
00086         if (lcdMode == NAV_LCD_MODE)
00087         {
00088             lcdPrintDecS08(botHeading, 1, 3);
00089         }
00090 #endif
00091         moveToJunction(1, justTurned); // linetracking, ground ball pickup
00092
00093 #if DEBUGGING
00094         if (lcdMode == NAV_LCD_MODE)
00095         {
00096             lcdPrintDecU08(botNode, 1, 0);
00097         }
00098 #endif
00099     }
00100
00101     if (pathList[pathListIndex + 1] == STOP_NODE)
00102     {
00103         positionBot();
00104         nestSequence();
00105     }
00106 }
00107
00108 /*
00109 *   Initializes some of bot's global variables
00110 */
00111 inline void initBotGlobals(void)
00112 {
00113     // init bot's path to INITIAL_PATH_LIST
00114     pathListIndex = 0;
00115     pathListSize = INITIAL_PATH_LIST_SIZE;

```

```

00116 // (pathList initialized in updatePath.c)
00117
00118 initGoalList();
00119 numKnownGoals = NUM_FIXED_GOALS;
00120
00121 liftDown = false;
00122 upComingBallNum = 0;
00123 }
00124
00125 /*
00126 * @brief Turn bot at junctions and, if necessary, ball nodes
00127 *
00128 * Maintains botHeading. Performs bonus ball pickup liftDown actions
00129 *
00130 * @return True when bot just turned. (Used to tell moveToJunction to
00131 * begin looking for next junction immediately.)
00132 *
00133 * PRE: camera is not streaming
00134 */
00135 inline bool positionBot(void)
00136 {
00137     bool justTurned = true;
00138
00139     int8_t nextHeading = getNextHeading(pathList[pathListIndex + 1]);
00140     int8_t bradsToTurn = nextHeading - botHeading;
00141
00142     // BB PICKUP CHECK
00143     if (botNode == BONUS_BALL_1 && isInGoalList(BONUS_BALL_1))
00144     {
00145         bbPositioning(BB1_HEADING, nextHeading);
00146         removeFromGoalList(BONUS_BALL_1);
00147     }
00148     else if (botNode == BONUS_BALL_2 && isInGoalList(BONUS_BALL_2))
00149     {
00150         bbPositioning(BB2_HEADING, nextHeading);
00151         removeFromGoalList(BONUS_BALL_2);
00152     }
00153
00154     // TURN/STRAIGHT CHECK
00155     else if (bradsToTurn != 0)
00156     {
00157         int8_t ticksToTurn;
00158         switch ((int8_t) bradsToTurn)
00159         {
00160             case -128: // U-turn
00161                 if (botNode == 37)
00162                 {
00163                     moveToJunction(1, false);
00164                     tickWheels(20, 20, 255);
00165                     msDelay(0x50);
00166                     moveStraight(-20, 255);
00167                 }
00168                 //tankTurn(245, -58);
00169                 tickWheels(-29, 29, 250);
00170                 tankTurn(245, -58);
00171                 break;
00172             case -105: // Hard Diagonal
00173                 tickWheels(28, 28, 250); //28
00174                 tractorTurn(255, -tempTweak4);
00175                 tankTurn(250, -70); // -80
00176                 break;
00177             case 23: // Soft Diagonal
00178                 tractorTurn(255, 23); //23
00179                 break;
00180             case -23:
00181                 tractorTurn(255, -28);
00182                 break;
00183             case 105:
00184                 tickWheels(17, 17, 250);
00185                 tankTurn(250, 80); //80
00186                 break;
00187             default:
00188                 // fixed ticks forward here?
00189
00190                 // convert brads to turn to ticks and turn
00191                 if (bradsToTurn < 0)
00192                 {
00193                     ticksToTurn = bradsToTurn + turnSubtract;
00194                 } else
00195                 {
00196                     ticksToTurn = bradsToTurn - turnSubtract;

```

```

00197         }
00198         tractorTurn(255, ticksToTurn);
00199         break;
00200     }
00201 }
00202 else
00203 {
00204     justTurned = false;
00205 }
00206
00207 if (botNode == SENSOR_NODE)
00208 {
00209     removeFromGoalList(SENSOR_NODE);
00210 }
00211
00212 // update botHeading
00213 botHeading = nextHeading;
00214
00215 // GB PICKUP CHECK: lower lift, if bot knows it will travel over ball
00216 upComingBallNum = getUpcomingBallNum();
00217 if (upComingBallNum != 0)
00218 {
00219     setServo(LIFT, LIFT_OPEN);
00220     msDelay(30);
00221     liftDown = true;
00222 }
00223
00224 return justTurned;
00225 }
00226
00227 /*
00228  * Returns absolute heading of next node given botNode and the next botNode.
00229  */
00230 static inline int8_t getNextHeading(uint8_t nextBotNode)
00231 {
00232     NODE nextNode;           // info about nodes adjacent to botNode
00233     int8_t nextNodeIndex;     // nextNode offset to nextBotNode
00234     int8_t nextHeading;       // absolute direction to nextBotNode
00235
00236     // get absolute direction of nextBotNode from node list
00237     getNode(botNode, &nextNode);
00238     if (&nextNode == NULL)
00239     {
00240         #if DEBUGGING
00241         lcdWriteStr("nodeList error ", 0, 0);
00242         #endif
00243         while (1) ;
00244     }
00245     nextNodeIndex = findValue(nextNode.adjNodes,
00246                             nextNode.numAdjNodes,
00247                             nextBotNode);
00248
00249     // get next heading or report error
00250     if (nextNodeIndex == -1)
00251     {
00252         #if DEBUGGING
00253         lcdWriteStr("pathList error ", 0, 0);
00254         #endif
00255         brake(BOTH);
00256         while (1) ;
00257     }
00258     nextHeading = nextNode.adjHeadings[nextNodeIndex];
00259
00260     return nextHeading;
00261 }
00262
00263 /* Rotates bot before and after Bonus Ball grab
00264  *   bbHeading - heading bot must have for bb pickup.
00265  *   nextHeading - heading bot must have after bb pickup
00266  */
00267 inline void bbPositioning(int8_t bbHeading, int8_t nextHeading)
00268 {
00269     // move forward (camera will be over junction at this point)
00270     // May or may not need to move forward (requires testing)
00271     // Some are fine without forward, some seem to need it
00272     // Right now, only -32 case moves forward
00273
00274     // rotate by (bbHeading - botHeading)
00275     switch ((int8_t) (bbHeading - botHeading))
00276     {
00277         case 96:

```

```

00278         // example of 96 brad rotation
00279         tickWheels(28, 0, 255); // allows fluid motion (no overshoot
correction)
00280         tankTurn(255, 58);          //58
00281         break;
00282     case -96:
00283         tickWheels(0, 28, 255); // allows fluid motion (no overshoot
correction)
00284         tankTurn(255, -64);        //-64
00285         break;
00286     case -32:
00287         tickWheels(10, 10, 255); //10 Move bot forward a few ticks to make it
correctly aligned
00288         tickWheels(0, 32, 255);          //28
00289         break;
00290     default:
00291 #if DEBUGGING
00292         lcdWriteStr("ERROR:          ", 0, 0);
00293         lcdWriteStr("Turn Amt =      ", 1, 0);
00294         lcdPrintDecS08(bbHeading - botHeading, 1, 11);
00295         brake(BOTH);
00296         while (1) ;
00297 #endif
00298         break;
00299     }
00300
00301     grabBonusBall(); // Grab the BB
00302
00303     // Rotate by (nextHeading - bbHeading)
00304     // (This should only be 32, -32, or -96)
00305     switch ((int8_t) (nextHeading - bbHeading))
00306     {
00307     case 32:
00308         tankTurn(250, 32);
00309         break;
00310     case -32:
00311         tankTurn(250, -32);
00312         break;
00313     case -96:
00314         tankTurn(250, -90);
00315         break;
00316     default:          // Error, this should only be 32, -32, or -96
00317 #if DEBUGGING
00318         lcdWriteStr("ERROR:          ", 0, 0);
00319         lcdWriteStr("nH - bbH =      ", 1, 0);
00320         lcdPrintDecS08(bbHeading - botHeading, 1, 11);
00321         brake(BOTH);
00322         while (1) ;
00323 #endif
00324         break;
00325     }
00326 }
00327
00328 /*
00329  * Moves to next junction in pathList.
00330  */
00331 inline void moveToJunction(uint8_t numJunctions, bool justTurned)
00332 {
00333     bool onLine = true;
00334     bool juncApproaching = false;
00335     uint8_t juncCount = 0;
00336
00337     uint8_t ignoreJuncCount;
00338     if (!justTurned)
00339     {
00340         ignoreJuncCount = 3;
00341     } else
00342     {
00343         ignoreJuncCount = 0;
00344     }
00345
00346     uint8_t pickingUp = false;
00347     uint8_t pickingUpCount = 0;
00348
00349     uint8_t ignoreBreakBeamCount = BEAM_IGNORE_COUNT;
00350
00351     trackLineInit();
00352
00353     // Linetrack, until bot is at junction or nest.
00354     // If see ground ball, pickup it up and continue linetracking.
00355     while (onLine)

```

```

00356     {
00357         while (lineStatsProcessed) ;
00358
00359         analyzeLineStats();
00360         adjustPWM();
00361
00362         // CURRENT JUNCTION IGNORE
00363         if (ignoreJuncCount > 0 && junctionY == 0)
00364         {
00365             ignoreJuncCount--;
00366         }
00367
00368         // JUNCTION CHECK
00369         if (ignoreJuncCount == 0 && junctionY != 0)
00370         {
00371             if (junctionY < turnPoint)
00372             {
00373                 juncApproaching = true;
00374             } else if (juncApproaching)
00375             {
00376                 juncApproaching = false;
00377                 juncCount++;
00378
00379                 // set botNode to next junction in pathList
00380                 do
00381                 {
00382                     pathListIndex++;
00383                     botNode = pathList[pathListIndex];
00384                 } while (!isJunction(botNode));
00385
00386                 // Break out of line tracking
00387                 if (juncCount >= numJunctions)
00388                 {
00389                     onLine = false;
00390                 }
00391             }
00392         }
00393
00394         // STOP IGNORING BEAM CHECK
00395         if (liftDown && ignoreBreakBeamCount != 0)
00396         {
00397             ignoreBreakBeamCount--;
00398         }
00399
00400         // BEGIN PICKUP CHECK
00401         if (liftDown && ignoreBreakBeamCount == 0 && BREAK_BEAM_TRIGGERED)
00402         {
00403             streamModeOff();
00404             setServo(LIFT, LIFT_CORRAL); // Perhaps raise it slowly if there
are pick-up problems
00405             msDelay(30);
00406             trackLineInit();
00407
00408             liftDown = false;
00409             pickingUp = true;
00410             pickingUpCount = 0;
00411         }
00412
00413         // COMPLETE/STOP LIFTING CHECK
00414         if (pickingUp)
00415         {
00416             pickingUpCount++;
00417
00418             if (pickingUpCount == CORRAL_COUNT)
00419             {
00420                 streamModeOff();
00421                 setServo(LIFT, LIFT_UP);
00422                 trackLineInit();
00423             }
00424
00425             if (pickingUpCount == LIFT_DONE_COUNT)
00426             {
00427                 pickingUp = false;
00428
00429                 // Set current botNode to node where this ball is
00430                 botNode = upComingBallNum;
00431                 removeFromGoalList(upComingBallNum);
00432
00433                 if (upComingBallNum == 1) // account for ball not found by
camera prior to pickup
00434                 {

```



```

00435         numKnownGoals++;
00436     }
00437
00438     // Find correct pathListIndex
00439     while (botNode != pathList[pathListIndex])
00440     {
00441         pathListIndex++;
00442     }
00443
00444     streamModeOff(); // Turn off line tracking
00445     disableServo(LIFT);
00446     positionBot(); // In case we want to make a -128 brad turn
    after picking up ball
00447     ignoreBreakBeamCount = BEAM_IGNORE_COUNT;
00448     trackLineInit(); // Turn line tracking back on
00449 }
00450
00451 }
00452 }
00453
00454     streamModeOff();
00455
00456     // Make sure lift is up (in case we missed a ball or incorrectly thought
    one was there)
00457     if (liftDown)
00458     {
00459         brake(BOTH);
00460 #if DEBUGGING
00461         lcdWriteStr("No ball", 0, 0);
00462 #endif
00463         setServo(LIFT, LIFT_UP); // Raise the lift
00464         msDelay(700);
00465         disableServo(LIFT);
00466         liftDown = false;
00467
00468         // correct goal state
00469         removeFromGoalList(upComingBallNum);
00470         numUnreachedGoals--;
00471         numKnownGoals--;
00472     }
00473 }
00474
00475 void nestSequence(void)
00476 {
00477     // line track, until NEST_BUTTON is pressed
00478     trackLineInit();
00479
00480     while (!justPressed(NEST_BUTTON))
00481     {
00482         if (!lineStatsProcessed)
00483         {
00484             analyzeLineStats();
00485             adjustPWM();
00486         }
00487
00488         debounceButtons();
00489     }
00490
00491     brake(BOTH);
00492     streamModeOff();
00493     setServo(LIFT, LIFT_UP); // Turn lift on
00494     msDelay(300);
00495
00496     // Open door, back up, close door
00497     setServo(DOOR, DOOR_OPEN);
00498     moveStraight(-1, 255); // Back up to take pressure off button
00499     brake(BOTH);
00500     //myDelay(25); // Let balls roll out
00501     msDelay(3000);
00502     setServo(DOOR, DOOR_CLOSED); // Leaves door closed, so lift and door don't
    colide on power up.
00503     //myDelay(10); // Wait for door to close
00504     msDelay(1000);
00505
00506     // Disable all servos
00507     disableServo(PAN);
00508     disableServo(TILT);
00509     disableServo(BOOM);
00510     disableServo(LIFT);
00511     disableServo(DOOR);
00512 }

```

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 BOTCTRL_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  */
00017 #include "buttons.h"
00018
00019 // AVRLIB
00020 #include "avr/avr-libc.h"
00021 // avr-libc
00022 #include <stdint.h>
00023 #include <stdbool.h>
00024
00025 #define DEBOUNCE_COUNT 3 // should be equal to 2 or greater
00026
00027 static bool isDown[NUM_BUTTONS] = { false, false, false,
00028                                     false, false, false };
00029 static bool wasEvent[NUM_BUTTONS] = { false, false, false,
00030                                     false, false, false };
00031 static uint8_t upCount[NUM_BUTTONS] = { DEBOUNCE_COUNT, DEBOUNCE_COUNT,
00032                                         DEBOUNCE_COUNT, DEBOUNCE_COUNT,
00033                                         DEBOUNCE_COUNT, DEBOUNCE_COUNT };
00034 static uint8_t downCount[NUM_BUTTONS] = { 0, 0, 0, 0, 0, 0 };
00035
00036 void waitFor(uint8_t button)
00037 {
00038     debounceButtons();
00039     while (!justReleased(button))
00040     {
00041         debounceButtons();
00042     }
00043 }
00044
00045 inline bool justPressed(uint8_t button)
00046 {
00047     return wasEvent[button] && isDown[button];
00048 }
00049
00050 inline bool justReleased(uint8_t button)
00051 {
00052     return wasEvent[button] && !isDown[button];
00053 }
00054
00055
00056
00057
00058
00059
00060
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 "buttons.h"
#include "eeProm.h"
#include "helperFunctions.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 [36](#) 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 "buttons.h"
00026 #include "eeprom.h"
00027 #include "helperFunctions.h"
00028
00029 // avr-libc
00030 #include <avr/io.h>
00031
00036 #define START_DELAY 5
00037
00041 static inline void initAtmel(void)
00042 {
00043     /*
00044      * Initialize Timer
00045      */
00046     timerInit();
00047
00048 #if DEBUGGING
00049     /*
00050      * Initialize LCD
00051      */
00052     lcdInit();
00053     lcdWriteStr("Init:           ", 0, 0);
00054     lcdWriteStr("           ", 1, 0);
00055 #endif
00056
00057     /*
00058      * Initialize UART for CMUcam communication
00059      */
00060     cmuCamInit();
00061
00062     /*
00063      * Initialize PWM motor control
00064      */
00065     outb(DDRD, 0xff);
00066     timer1PWMInit(8);
00067     neutral();
00068     enableMotors();
00069
00070     /*
00071      * Set data direction registers
00072      */
00073     outb(DDRA, 0xF0); // Motor control and up/down buttons
00074     cbi(DDRD, 6);     // red button
00075     cbi(DDRB, 0);     // nest button
00076     cbi(DDRB, 1);     // break beam
00077
00078     /*
00079      * Apply internal pull-up resistor to certain digital inputs
00080      */
00081     sbi(PORTB, 0); // internal pull-up for PINB0
00082     sbi(PORTA, 3); // internal pull-up for PINA3
00083     sbi(PORTA, 2); // internal pull-up for PINA2
00084     sbi(PORTA, 1); // internal pull-up for PINA1
00085     sbi(PORTA, 0); // internal pull-up for PINA0
00086
00087     /*
00088      * Initialize quadrature wheel encoders
00089      */
00090     cbi(DDRD, 2);
00091     cbi(DDRD, 3);
00092     encoderInit();
00093 }
00094
00098 int main(void)
00099 {
00100     initAtmel();
00101     loadTweakValues();
00102     initBotGlobals();
00103     resetCamera();
00104     moveServosToStart();
00105     cameraWhiteBalance();
00106
00107 #if DEBUGGING
00108     runTetherUI();
00109     myDelay(START_DELAY);
00110     runTest();
00111 #else
00112     waitFor(RED_BUTTON);
00113     myDelay(START_DELAY);
00114     runRoborodentiaCourse();

```

```

00115 #endif
00116
00117     brake(BOTH);
00118 #if DEBUGGING
00119     lcdWriteStr("Done", 0, 0);
00120     lcdWriteStr("    ", 1, 0);
00121 #endif
00122
00123     return 0;
00124 }

```

9.11 camera.c File Reference

```

#include "camera.h"
#include "trackColor.h"
#include "trackLine.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 46 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 140 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  */
00017 #include "camera.h"
00018 #include "trackColor.h"
00019 #include "trackLine.h"
00020
00021 // AVRLIB
00022 #include "rprintf.h"
00023 #include "uart.h"
00024
00025 // avr-libc
00026 #include <stdbool.h>
00027
00028 #define CMU_BAUD 38400
00029
00030 static uint8_t mode;
00031 static uint16_t byteNum;
00032
00033 void packetRcv( uint8_t c );
00034 inline void lineMode2Rcv( uint8_t c );
00035 inline void trackColorRcv( uint8_t c );
00036
00037 inline void cmuCamInit(void)
00038 {
00039     uartInit();
00040     uartSetBaudRate(CMU_BAUD);
00041     uartSetRxHandler(packetRcv);
00042     rprintfInit(uartSendByte);
00043 }
00044
00045 inline void cameraWhiteBalance()
00046 {
00047     // turn auto white balance on
00048     #if DEBUGGING
00049         lcdWriteStr("white Bal ", 0, 6);
00050     #endif
00051     rprintf("CR 18 44\r");
00052     myDelay(200);
00053     // turn auto white balance off
00054     rprintf("CR 18 40\r");
00055 }
00056

```

```
00057
00058 inline void resetCamera( void )
00059 {
00060     mode = NEW_PACKET;
00061     byteNum = 0;
00062
00063     rprintf("RM 3\r");
00064 }
00065
00066 void packetRcv(uint8_t c)
00067 {
00068     if (c == 0xff)
00069     {
00070         mode = NEW_PACKET;
00071         byteNum = 0;
00072     }
00073     else
00074     {
00075         switch (mode)
00076         {
00077             case NEW_PACKET:
00078                 switch (c)
00079                 {
00080                     case 0xfe:
00081                         mode = FE_RCV;
00082                         break;
00083                     case 'T':
00084                         mode = T_RCV;
00085                         break;
00086                     default:
00087                         break;
00088                 }
00089                 break;
00090             case FE_RCV:
00091                 lineMode2Rcv(c);
00092                 if (c == 0xfd)
00093                 {
00094                     mode = NEW_PACKET;
00095                 }
00096                 break;
00097             case T_RCV:
00098                 trackColorRcv(c);
00099                 break;
00100         }
00101     }
00102 }
00103
00104
00105 inline void lineMode2Rcv(uint8_t c)
00106 {
00107     if (c == 0xfd)
00108     {
00109         lineStatsProcessed = false;
00110         byteNum = 0;
00111     }
00112     else
00113     {
00114         lineStats[(byteNum - 1) / LINE_STATS_COLS]
00115             [(byteNum - 1) % LINE_STATS_COLS] = c;
00116         byteNum++;
00117     }
00118 }
00119
00120
00121 inline void trackColorRcv(uint8_t c)
00122 {
00123     lineStats[0][byteNum] = c;
00124     byteNum++;
00125
00126     if (byteNum >= NUM_COLOR_STATS)
00127     {
00128         colorStatsProcessed = false;
00129     }
00130 }
00131
00132
00133 inline void streamModeOff( void )
00134 {
00135     rprintf("\r\r"); // add an extra return as recommended by CMUcam manual
00136     msDelay(32);      // wait for streaming to stop ( 16ms delay ok )
00137 }
```

```

00138
00139
00140 inline void setVirtualWindow(uint8_t x1, uint8_t y1, uint8_t x2
, uint8_t y2)
00141 {
00142     rprintf("VW %d %d %d %d\r", x1, y1, x2, y2);
00143 }

```

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 46 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 140 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  */
00017 #ifndef CAMERA_H_
00018 #define CAMERA_H_
00019
00020 #include <stdint.h>
00021
00022 // Packet types
00023 #define NEW_PACKET 0
00024 #define FE_RCV 1
00025 #define T_RCV 2
00026
00027 #define hiResMode() rprintf("HR 1\r")
00028 #define lowResMode() rprintf("HR 0\r")
00029
00030 inline void cmuCamInit(void);
00031
00032 inline void cameraWhiteBalance( void );
00033
00034 inline void resetCamera( void );
00035 inline void streamModeOff( void );
00036
00037 inline void setVirtualWindow(uint8_t x1, uint8_t y1, uint8_t x2
00038 , uint8_t y2);
00039
00040 #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  */
00017 #include "eeProm.h"
00018
00019 // avr-libc
00020 #include <avr/io.h>
00021 #include <avr/interrupt.h>
00022
00023 // Global variables - Runtime configurable parameters
00024 uint8_t l_base;
00025 uint8_t r_base;
00026 uint16_t slopeCoef;
00027 uint16_t offCoef;
00028 uint8_t dampCoef;
00029 uint8_t lineCenter;
00030 uint8_t turnPoint;
00031 uint8_t turnSubtract;
00032 int8_t panOffset;
00033 int8_t tiltOffset;
00034 uint16_t tractorOvershootDelay;
00035 uint8_t tempTweak1;
00036 int8_t tempTweak2;
00037 uint16_t tempTweak3;
00038 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(EA_ADDR_LEFT_BASE);
00051     r_base = EEPROM_read(EA_ADDR_RIGHT_BASE);
00052     slopeCoef = (EEPROM_read(EA_ADDR_SLOPE_COEF) << 8) +
00053         EEPROM_read(EA_ADDR_SLOPE_COEF + 1);
00054     offCoef = (EEPROM_read(EA_ADDR_OFF_COEF) << 8) +
00055         EEPROM_read(EA_ADDR_OFF_COEF + 1);
00056     dampCoef = EEPROM_read(EA_ADDR_DAMP_COEF);
00057     lineCenter = EEPROM_read(EA_ADDR_LINE_X_CENTER);
00058     turnPoint = EEPROM_read(EA_ADDR_TURN_POINT);
00059     turnSubtract = EEPROM_read(EA_ADDR_TURN_SUBTRACT);
00060     panOffset = EEPROM_read(EA_ADDR_PAN_OFFSET);
00061     tiltOffset = EEPROM_read(EA_ADDR_TILT_OFFSET);
00062     tractorOvershootDelay = (EEPROM_read(EA_ADDR_TRACTOR_OVERSHOOT_DELAY) << 8)
+
00063         EEPROM_read(EA_ADDR_TRACTOR_OVERSHOOT_DELAY + 1);
00064     testMode = EEPROM_read(EA_ADDR_TEST_MODE);
00065     tempTweak1 = EEPROM_read(EA_ADDR_TEMP_TWEAK1);
00066     tempTweak2 = EEPROM_read(EA_ADDR_TEMP_TWEAK2);
00067     tempTweak3 = (EEPROM_read(EA_ADDR_TEMP_TWEAK3) << 8) +
00068         EEPROM_read(EA_ADDR_TEMP_TWEAK3 + 1);
00069     tempTweak4 = (EEPROM_read(EA_ADDR_TEMP_TWEAK4) << 8) +
00070         EEPROM_read(EA_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(EA_ADDR_LEFT_BASE, l_base);
00082     EEPROM_write(EA_ADDR_RIGHT_BASE, r_base);
00083     EEPROM_write(EA_ADDR_SLOPE_COEF, slopeCoef >> 8);
00084     EEPROM_write(EA_ADDR_SLOPE_COEF + 1, slopeCoef);
00085     EEPROM_write(EA_ADDR_OFF_COEF, offCoef >> 8);
00086     EEPROM_write(EA_ADDR_OFF_COEF + 1, offCoef);
00087     EEPROM_write(EA_ADDR_DAMP_COEF, dampCoef);
00088     EEPROM_write(EA_ADDR_LINE_X_CENTER, lineCenter);
00089     EEPROM_write(EA_ADDR_TURN_POINT, turnPoint);
00090     EEPROM_write(EA_ADDR_TURN_SUBTRACT, turnSubtract);
00091     EEPROM_write(EA_ADDR_PAN_OFFSET, panOffset);
00092     EEPROM_write(EA_ADDR_TILT_OFFSET, tiltOffset);
00093     EEPROM_write(EA_ADDR_TRACTOR_OVERSHOOT_DELAY, tractorOvershootDelay >> 8);
00094     EEPROM_write(EA_ADDR_TRACTOR_OVERSHOOT_DELAY + 1, tractorOvershootDelay);
00095     EEPROM_write(EA_ADDR_TEST_MODE, testMode);
00096     EEPROM_write(EA_ADDR_TEMP_TWEAK1, tempTweak1);
00097     EEPROM_write(EA_ADDR_TEMP_TWEAK2, tempTweak2);
00098     EEPROM_write(EA_ADDR_TEMP_TWEAK3, tempTweak3 >> 8);
00099     EEPROM_write(EA_ADDR_TEMP_TWEAK3 + 1, tempTweak3);
00100     EEPROM_write(EA_ADDR_TEMP_TWEAK4, tempTweak4 >> 8);
00101     EEPROM_write(EA_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 << EWE)) ;
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 << EEMWE)) ;
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 EEMWE
00128     EECR |= (1 << EEMWE);
00129     // EEMWE and EEMWE 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/signal.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  */
00017 //*****
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/signal.h>
00040 #include <avr/interrupt.h>
00041 #endif
00042
00043 #include "global.h"
00044 #include "encoder.h"
00045
00046 // Program ROM constants
00047
00048 // Global variables
00049 volatile EncoderStateType EncoderState[NUM_ENCODERS];
00050
00051 // Functions
00052
00053 // encoderInit() initializes hardware and encoder position readings
00054 // Run this init routine once before using any other encoder functions.
00055 inline void encoderInit(void)
00056 {
00057     uint8_t i;
00058
00059     // initialize/clear encoder data
00060     for (i = 0; i < NUM_ENCODERS; i++)
00061     {
00062         EncoderState[i].position = 0;
00063         //EncoderState[i].velocity = 0; // NOT CURRENTLY USED
00064     }
00065
00066     // configure direction and interrupt I/O pins:
00067     // - for input
00068     // - apply pullup resistors
00069     // - any-edge interrupt triggering
00070     // - enable interrupt
00071
00072 #ifdef ENC0_SIGNAL
00073     // set interrupt pins to input and apply pullup resistor
00074     cbi(ENC0_PHASEA_DDR, ENC0_PHASEA_PIN);
00075     sbi(ENC0_PHASEA_PORT, ENC0_PHASEA_PIN);
00076     // configure interrupts for any-edge triggering
00077     sbi(ENC0_ICR, ENC0_ISCX0);
00078     cbi(ENC0_ICR, ENC0_ISCX1);
00079     // enable interrupts
00080     sbi(IMSK, ENC0_INT);
00081     // ISMK is auto-defined in encoder.h
00082 #endif
00083 #ifdef ENC1_SIGNAL
00084     // set interrupt pins to input and apply pullup resistor
00085     cbi(ENC1_PHASEA_DDR, ENC1_PHASEA_PIN);
00086     sbi(ENC1_PHASEA_PORT, ENC1_PHASEA_PIN);
00087     // configure interrupts for any-edge triggering
00088     sbi(ENC1_ICR, ENC1_ISCX0);
00089     cbi(ENC1_ICR, ENC1_ISCX1);
00090     // enable interrupts
00091     sbi(IMSK, ENC1_INT);
00092     // ISMK is auto-defined in encoder.h
00093 #endif
00094 #ifdef ENC2_SIGNAL
00095     // set interrupt pins to input and apply pullup resistor
00096     cbi(ENC2_PHASEA_DDR, ENC2_PHASEA_PIN);
00097     sbi(ENC2_PHASEA_PORT, ENC2_PHASEA_PIN);
00098     // configure interrupts for any-edge triggering
00099     sbi(ENC2_ICR, ENC2_ISCX0);
00100     cbi(ENC2_ICR, ENC2_ISCX1);
00101     // enable interrupts
00102     sbi(IMSK, ENC2_INT); // ISMK is auto-defined in encoder.h
00103 #endif
00104 #ifdef ENC3_SIGNAL
00105     // set interrupt pins to input and apply pullup resistor
00106     cbi(ENC3_PHASEA_DDR, ENC3_PHASEA_PIN);
00107     sbi(ENC3_PHASEA_PORT, ENC3_PHASEA_PIN);
00108     // set encoder direction pin for input and apply pullup resistor
00109     cbi(ENC3_PHASEB_DDR, ENC3_PHASEB_PIN);

```

```

00110     sbi(ENC3_PHASEB_PORT, ENC3_PHASEB_PIN);
00111     // configure interrupts for any-edge triggering
00112     sbi(ENC3_ICR, ENC3_ISCX0);
00113     cbi(ENC3_ICR, ENC3_ISCX1);
00114     // enable interrupts
00115     sbi(IMSK, ENC3_INT); // ISMK is auto-defined in encoder.h
00116 #endif
00117
00118     // enable global interrupts
00119     sei();
00120 }
00121
00122 // encoderGetPosition() reads the current position of the encoder
00123 uint16_t encoderGetPosition(uint8_t encoderNum)
00124 {
00125     // sanity check
00126     if (encoderNum < NUM_ENCODERS)
00127         return EncoderState[encoderNum].position;
00128     else
00129         return 0;
00130 }
00131
00132 // encoderSetPosition() sets the current position of the encoder
00133 void encoderSetPosition(uint8_t encoderNum, uint16_t position)
00134 {
00135     // sanity check
00136     if (encoderNum < NUM_ENCODERS)
00137         EncoderState[encoderNum].position = position;
00138     // else do nothing
00139 }
00140
00141 #ifdef ENC0_SIGNAL
00142
00143 SIGNAL(ENC0_SIGNAL)
00144 {
00145     /******
00146     /* Modified by Taylor
00147     /******
00148     EncoderState[0].position++;
00149 }
00150 #endif
00151
00152 #ifdef ENC1_SIGNAL
00153
00154 SIGNAL(ENC1_SIGNAL)
00155 {
00156     /******
00157     /* Modified by Taylor
00158     /******
00159     EncoderState[1].position++;
00160 }
00161 #endif
00162
00163 #ifdef ENC2_SIGNAL
00164
00165 SIGNAL(ENC2_SIGNAL)
00166 {
00167     // encoder has generated a pulse
00168     // check the relative phase of the input channels
00169     // and update position accordingly
00170     if( ((inb(ENC2_PHASEA_PORTIN) & (1<<ENC2_PHASEA_PIN)) == 0) ^
00171         ((inb(ENC2_PHASEB_PORTIN) & (1<<ENC2_PHASEB_PIN)) == 0) )
00172     {
00173         EncoderState[2].position++;
00174     }
00175     else
00176     {
00177         EncoderState[2].position--;
00178     }
00179 }
00180 #endif
00181
00182 #ifdef ENC3_SIGNAL
00183
00184 SIGNAL(ENC3_SIGNAL)
00185 {
00186     // encoder has generated a pulse
00187     // check the relative phase of the input channels
00188     // and update position accordingly
00189     if( ((inb(ENC3_PHASEA_PORTIN) & (1<<ENC3_PHASEA_PIN)) == 0) ^

```

```

00190      ( (inb(ENC3_PHASEB_PORTIN) & (1<<ENC3_PHASEB_PIN)) == 0) )
00191      {
00192          EncoderState[3].position++;
00193      }
00194      else
00195      {
00196          EncoderState[3].position--;
00197      }
00198  }
00199  #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  */
00017 //*****
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 backwards. 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 //
00074 // To keep track of the encoder position in software, we connect PhaseA to an
00075 // external processor interrupt line, and PhaseB to any I/O pin. We set up
00076 // the external interrupt to trigger whenever PhaseA produces a rising edge.
00077 // When a rising edge is detected, our interrupt handler function is executed.
00078 // Inside the handler function, we quickly check the PhaseB line to see if it
00079 // is high or low. If it is high, we increment the encoder's position
00080 // 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/typedefs
00103
00104 // defines for processor compatibility
00105 // chose proper Interrupt Mask (IMSK)
00106 #ifdef EIMSK
00107     #define IMSK EIMSK // for processors megal28, mega64
00108 #else
00109     #define IMSK GIMSK // for other processors 90s8515, megal63, 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 interrupts 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, megal61, megal63, mega323, megal6, mega32, etc)
00045 //
00046 // INT0 -> PD2 (PORTD, pin 2)
00047 // INT1 -> PD3 (PORTD, pin 3)
00048 //
00049 // The external interrupt pins on the processors megal28 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/typedefs
00069
00070 // defines for processor compatibility
00071 // quick compatibility for megal28, 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 ENCO_SIGNAL SIG_INTERRUPT0 // Interrupt signal name
00088 #define ENCO_INT INT0 // matching INTx bit in GIMSK/EIMSK
00089 #define ENCO_ICR MCUCR // matching Int. Config Register (MCUCR,EICRA/
B)
00090 #define ENCO_ISCX0 ISC00 // matching Interrupt Sense Config bit0
00091 #define ENCO_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 ENCO_PHASEA_PORT      PORTD // PhaseA port register
00095 #define ENCO_PHASEA_DDR        DDRD  // PhaseA port direction register
00096 #define ENCO_PHASEA_PORTIN     PIND  // PhaseA port input register
00097 #define ENCO_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 ENCO_PHASEB_PORT      PORTC // PhaseB port register
00101 //#define ENCO_PHASEB_DDR        DDRC // PhaseB port direction register
00102 //#define ENCO_PHASEB_PORTIN     PINC // PhaseB port input register
00103 //#define ENCO_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/
00116                                     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/
00142                                     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/
00168                                     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  */
00017 #include "exercises.h"
00018 #include "botCntrl.h"
00019 #include "motorCntrl.h"
00020 #include "buttons.h"
00021 #include "helperFunctions.h"
00022 #include "ourLCD.h"
00023
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(REDBUTTON);
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(REDBUTTON);
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(REDBUTTON);
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(REDBUTTON);
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(REDBUTTON);
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 "nodeList.h"
#include "eeProm.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 27 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  */
00017 #include "junctionCode.h"
00018 #include "botCntrl.h"
00019 #include "trackColor.h"
00020 #include "servos.h"
00021 #include "nodeList.h"
00022 #include "eeProm.h"
00023
00024 // Global variables
00025 // initialized in initBotGlobals
```

```

00027 bool checkedList[] = { false, false, false, false, false, false,
00028                        false, false, false, false, false, false,
00029                        false, false, false, false, false, false,
00030                        false, false, false };
00031 // initialized in initBotGlobals
00032 uint8_t goalList[NUM_GOALS];
00033
00034 uint8_t goalListSize;
00035 uint8_t numKnownGoals;
00036
00037 /*
00038  * Searches for ground balls, picks-up bonous balls, and computes best path.
00039  */
00040 void junctionCode(void)
00041 {
00042     bool foundBall = false;
00043
00044     switch (botNode)
00045     {
00046     case (0): // old virtual windowing look for ball 7 and
00047         8
00048         foundBall = nodeCode0();
00049         break;
00050     case (21): // Suppress standard seek, should
00051         already break; // skip junction code at this node
00052     case (22):
00053         foundBall = standardBallSearch(); // standard seek and
00054         foundBall |= nodeCode22(); // tilt look for ball 9, 11, and 12
00055         break;
00056     case (31): // rotate bot for diagonal ball search
00057     case (34): // from any heading at node 31 or 34
00058         foundBall = diagNodeCode();
00059         break;
00060     case (37): // seek for top balls from nest
00061         sensor
00062         if (numKnownGoals < NUM_GOALS && (!checkedList[13] || !checkedList[17])
00063         )
00064         {
00065             botNode = 35;
00066             moveStraight(10, 255);
00067             foundBall = diagNodeCode();
00068             moveStraight(-10, 255);
00069             botNode = 37;
00070
00071             //pathList[pathListIndex--] = 36;
00072             //pathList[pathListIndex--] = 35;
00073         }
00074         break;
00075     default:
00076         foundBall = standardBallSearch();
00077         break;
00078     }
00079     if (foundBall)
00080     {
00081         // clear checked list, if last ball found
00082         if (numKnownGoals == NUM_GOALS)
00083         {
00084             uint8_t i;
00085             for (i = 0; i < NUM_BALL_NODES + 1; i++)
00086             {
00087                 checkedList[i] = true;
00088             }
00089             updatePath();
00090             printGoalList();
00091         }
00092     }
00093 }
00094 /*
00095  * Returns true if a ball is found and the goal list is updated
00096  */
00097 bool standardBallSearch( void )
00098 {
00099     NODE curNode;
00100     NODE nextNode;
00101     uint8_t nextNodeNum;
00102     int8_t lookDir = -1; // look left first
00103     int8_t hallHeading = 0;

```

```

00104     uint8_t ballDist = 0;
00105     uint8_t uncheckedBalls[3][2];
00106     uint8_t numUncheckedBalls = 0;
00107     bool foundBall = false;
00108     uint8_t i;
00109
00110     bool stopped = false;
00111     inSeekPosition = false;
00112
00113     // Check for balls in two directions (left/right)
00114     for (i = 0; i < 2; i++)
00115     {
00116         ballDist = 0;
00117         hallHeading = botHeading + lookDir * 64;
00118         nextNodeNum = botNode;
00119         numUncheckedBalls = 0;
00120         // Continue traversing nodes left of right until you hit the end
00121         while (nextNodeNum > 0)
00122         {
00123             /*
00124              *if DEBUGGING
00125              lcdWriteStr("N:  H:  ", 0, 0);
00126              lcdPrintHex(nextNodeNum, 0, 2);
00127              lcdPrintHex(hallHeading, 0, 7);
00128              waitFor(RED_BUTTON);
00129              #endif
00130              */
00131             getNode(nextNodeNum, &curNode);
00132             nextNodeNum = getNodeAtHeading(&curNode, hallHeading);
00133             if (nextNodeNum > 0)
00134             {
00135                 getNode(nextNodeNum, &nextNode);
00136                 // Keep track of how far away we are from the bot's current
00137                 node
00138                 ballDist += getCostToNode(&curNode, nextNodeNum);
00139                 if (isBallNode(nextNodeNum) && !checkedList[nextNodeNum])
00140                 {
00141                     uncheckedBalls[numUncheckedBalls][BALL_DIST] = ballDist;
00142                     uncheckedBalls[numUncheckedBalls][BALL_NODE_NUM] =
00143                         nextNodeNum;
00144                     checkedList[nextNodeNum] = true;
00145                     numUncheckedBalls++;
00146                 }
00147             }
00148             // Set pan, tilt, hi-res, etc...
00149
00150             if (numUncheckedBalls > 0)
00151             {
00152                 stopped = true;
00153                 trackColorInit(lookDir);
00154
00155                 if (lookDir == -1)
00156                 {
00157                     foundBall |= cameraSeekLeft(uncheckedBalls, numUncheckedBalls);
00158                 } else if (lookDir == 1)
00159                 {
00160                     foundBall |= cameraSeekRight(uncheckedBalls, numUncheckedBalls)
00161                 }
00162             }
00163
00164             /*
00165              *if DEBUGGING
00166              lcdWriteStr("Seek ( ) for: ", 0, 0);
00167              lcdPrintHex(lookDir, 0, 6);
00168              lcdWriteStr("          ", 1, 0);
00169              for(j = 0; j < numUncheckedBalls; j++)
00170              {
00171                  lcdPrintHex(uncheckedBalls[j][BALL_NODE_NUM], 1, 0);
00172                  lcdPrintHex(uncheckedBalls[j][BALL_DIST], 1, 3);
00173                  waitFor(RED_BUTTON);
00174              }
00175              waitFor(RED_BUTTON);
00176              #endif
00177              */
00178
00179             lookDir *= -1; // Look the other way the next time through
00180         }
00181
00182         if (stopped)

```



```

00183     {
00184         moveStraight(0xb, 255);
00185         setServo(PAN, PAN_CENTER + panOffset);
00186         setServo(TILT, TILT_FORWARD);
00187         msDelay(600);
00188     }
00189
00190     // Returns true if one or more balls are found
00191     return foundBall;
00192 }
00193
00194
00195 inline bool nodeCode0(void)
00196 {
00197     bool foundBall = false;
00198
00199     // two virtual windows
00200
00201     return foundBall;
00202 }
00203
00204
00205 inline bool nodeCode22()
00206 {
00207     bool foundBall = false;    // Return value
00208     uint8_t scanHeight = 4;
00209     uint8_t y = 254;
00210     uint8_t scanLimit = 1;
00211     uint8_t foundBallNum = 0;
00212
00213     if (botHeading != 0)
00214         return false;
00215
00216     trackColorInit(LOOK_UP);
00217
00218     // scan from small ground distance to large ground distance
00219     while ( y - scanHeight > scanLimit )
00220     {
00221         y -= scanHeight;
00222         setVirtualWindow(1, y-scanHeight, 174, y);
00223         if ( seeBall() )
00224         {
00225             foundBall = true;
00226
00227             // find ball number of ball at this x
00228             if( y > 148 )
00229                 foundBallNum = 9;
00230             else if ( y > 50 )
00231                 foundBallNum = 11;
00232             else
00233                 foundBallNum = 12;
00234
00235             addToGoalList( foundBallNum );
00236
00237 #if DEBUGGING
00238             labelColorStats();
00239             refreshColorStats();
00240             //msDelay(1000);
00241             //clearColorStats();
00242 #endif
00243
00244 /*
00245 #if DEBUGGING
00246         lcdWriteStr("Added:          ",0,0);
00247         lcdWriteStr("          ",1,0);
00248         lcdPrintHex(foundBallNum,1,0);
00249         waitFor(REDA_BUTTON);
00250 #endif
00251 */
00252         while ( seeBall() )
00253         {
00254             y -= scanHeight;
00255             setVirtualWindow(1, y-scanHeight, 174, y);
00256         }
00257     }
00258 }
00259
00260 setServo(PAN, PAN_CENTER+panOffset);
00261 setServo(TILT, TILT_FORWARD);
00262 msDelay(300);
00263

```

```

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

```

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  */
00017 #include "nodeList.h"
00018
00019 // avr-libc
00020 #include <string.h>
00021
00022 inline bool isJunction(uint8_t nodeNum)
00023 {
00024     return (nodeNum >= JUNCTION_MIN && nodeNum <= JUNCTION_MAX);
00025 }
00026
00027 inline bool isBallNode(uint8_t nodeNum)
00028 {
00029     return (nodeNum >= BALL_NODE_MIN && nodeNum <= BALL_NODE_MAX);
00030 }
00031
00032 uint8_t getCostToNode(NODE *node, uint8_t nodeNum)
00033 {
00034     uint8_t i;
00035     for (i = 0; i < node->numAdjNodes; i++)
00036     {
00037         if (node->adjNodes[i] == nodeNum)
00038         {
00039             return node->adjCosts[i];
00040         }
00041     }
00042     return 0;
00043 }
00044
00045 uint8_t getNodeAtHeading(NODE *node, int8_t heading)
00046 {
00047     uint8_t i;
00048     for (i = 0; i < node->numAdjNodes; i++)
00049     {
00050         if (node->adjHeadings[i] == heading)
00051         {
00052             return node->adjNodes[i];
00053         }
00054     }
00055     return 0;
00056 }
00057
00058 void getNode(uint8_t nodeNum, NODE *node)
00059 {
00060     if (nodeNum >= NUM_NODES)
00061     {
00062         node = NULL;
00063     }
00064
00065     switch (nodeNum)
00066     {
00067     case 0: // START_NODE
00068         node->numAdjNodes = 1;
00069         node->adjNodes[0] = 21;
00070         node->adjCosts[0] = 9;
00071         node->adjHeadings[0] = -64;
00072     }

```

```
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_MIN + 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_MIN + 1)
00052
00053
00054 #define MAX_ADJ_NODES  3          // max. nodes that can be adjacent to one
00055                                node
00056 #define N_WEST          -41       // direction of north west in binary radians
00057                                (brads)
00058 #define S_EAST          87        // direction of south east in binary radians
00059                                (brads)
00060
00061 #define BONUS_BALL_1    24
00062 #define BONUS_BALL_2    30
00063 #define SENSOR_NODE     37
00064
00065 #define BB1_HEADING     32
00066 #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
00072     node
00073     uint8_t adjNodes[MAX_ADJ_NODES]; // node numbers of adjacent nodes
00074     uint8_t adjCosts[MAX_ADJ_NODES]; // distances to adjacent nodes (6
00075     inches increments)
00076     int8_t adjHeadings[MAX_ADJ_NODES]; // directions towards adjacent nodes
00077     (brads)
00078 } NODE;
00079
00080 inline bool isJunction( uint8_t nodeNum );
00081 uint8_t getCostToNode( NODE *node, uint8_t nodeNum );
00082 uint8_t getNodeAtHeading( NODE *node, int8_t heading );
00083 inline bool isBallNode( uint8_t nodeNum );
00084 void getNode( uint8_t nodeNum, NODE *node );
00085
00086 #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" permutation 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 permutation 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 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 "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 "motorCntl.h"
00024 #include "eeProm.h"
00025
00026 // AVRLIB
00027 #include "rprintf.h"
00028
00029 // avr-libc
00030 #include <stdint.h>
00031 #include <stdbool.h>
00032
00033 // Track the RED ball on black/white background
00034 #define BALL_RMIN 150
00035 #define BALL_RMAX 240
00036 #define BALL_GMIN 16
00037 #define BALL_GMAX 60
00038 #define BALL_BMIN 16
00039 #define BALL_BMAX 50
00040
00041 // Global variables
00042 volatile bool colorStatsProcessed;
00043 bool inSeekPosition;
00044
00045 static uint8_t distToPix( uint8_t distance );
00046
00047 void trackColorInit(int8_t dir)
00048 {
00049     if (!inSeekPosition)
00050     {
00051         brake(BOTH);
00052         msDelay(200);
00053         moveStraight(-1 * 0xb, 255);
00054         inSeekPosition = true;
00055     }
00056
00057     // Set pan (center) and tilt
00058     switch (dir)
00059     {
00060     case LOOK_LEFT:
00061         setServo(PAN, PAN_CENTER + panOffset + PAN_SEEK_OFFSET);
00062         setServo(TILT, TILT_VERT + tiltOffset);
00063         break;
00064     case LOOK_RIGHT:
00065         setServo(PAN, PAN_CENTER + panOffset - PAN_SEEK_OFFSET);
00066         setServo(TILT, TILT_VERT + tiltOffset);
00067         break;
00068     case LOOK_UP:
00069         setServo(PAN, PAN_CENTER + panOffset);
00070         setServo(TILT, TILT_LOOKUP);
00071         break;
00072     default:
00073         break;
00074     }
00075     msDelay(500);
00076
00077     hiResMode();
00078     rprintf("DS 1 1\r");
00079     rprintf("LM 0 0\r");
00080
00081     // Change to poll mode so only one packet is sent
00082     rprintf("PM 1\r");
00083 }
00084
00085 /*
00086  * Returns Y1 (top of ball) if camera sees a ball, zero otherwise
00087  */
00088 uint8_t getBallY( void )
00089 {
00090     rprintf("lm 0 0\r");
00091
00092     // Mask everything but the 'My' value
00093     //rprintf("OM 0 2\r"); //<- NO MASKING?
00094
00095     // Change to poll mode so only one packet is sent
00096     rprintf("PM 1\r");
00097
00098     // Track red

```



```

00099     rprintf("TC %d %d %d %d %d %d\r",
00100             BALL_RMIN, BALL_RMAX, BALL_GMIN, BALL_GMAX, BALL_BMIN,
BALL_BMAX);
00101
00102     colorStatsProcessed = true;
00103     while (colorStatsProcessed) ;
00104
00105     return (lineStats[0][Y1_NDX]);
00106 }
00107
00108
00109 bool seeBall( void )
00110 {
00111     // Track red
00112     rprintf("TC %d %d %d %d %d %d\r",
00113             BALL_RMIN, BALL_RMAX, BALL_GMIN, BALL_GMAX, BALL_BMIN, BALL_BMAX);
00114     colorStatsProcessed = true;
00115     while (colorStatsProcessed) ;
00116
00117     return lineStats[0][Y1_NDX] > 0;
00118 }
00119
00120
00121 /*
00122  * Just does left seeks
00123  *   PRE - the longest check is the last element of the uncheckedBalls array
00124  *
00125  *   uncheckedBalls - ball node numbers and ground distances away from bot
00126  */
00127 bool cameraSeekLeft( uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls )
00128 {
00129     bool foundBall = false;    // Return value
00130     uint8_t scanHeight = 4;
00131     uint8_t x = 174;
00132     //uint8_t ballDist[3];
00133     //uint8_t ballCount = 0;
00134     uint8_t scanLimit = distToPix(
00135         uncheckedBalls[numUncheckedBalls - 1][BALL_DIST] + 1);
00136
00137     // get pixel ranges for unchecked balls passed in
00138     uint8_t i = 0;
00139     uint8_t maxBallX[3];
00140     while (i + 1 < numUncheckedBalls)
00141     {
00142         maxBallX[i] = (distToPix(uncheckedBalls[i][BALL_DIST]) +
00143             distToPix(uncheckedBalls[i + 1][BALL_DIST])) / 2;
00144         i++;
00145     }
00146     maxBallX[i] = scanLimit;
00147
00148     /*
00149     #if DEBUGGING
00150     lcdWriteStr("maxBallX =      ",0,0);
00151     lcdWriteStr("      ",1,0);
00152     for( i = 0; i < numUncheckedBalls; i++ )
00153     {
00154         lcdPrintHex(maxBallX[i],1,3*i);
00155     }
00156     waitFor(REDBUTTON);
00157     #endif
00158     */
00159
00160     // scan from small ground distance to large ground distance
00161     while (x - scanHeight > scanLimit)
00162     {
00163         x -= scanHeight;
00164         setVirtualWindow(x - scanHeight, 1, x, 254);
00165         if (seeBall())
00166         {
00167             foundBall = true;
00168             //ballDist[ballCount++] = xToDist(x);
00169
00170             // find ball number of ball at this x
00171             i = 0;
00172             while (maxBallX[i] > x)
00173             {
00174                 i++;
00175             }
00176             addToGoalList(uncheckedBalls[i][BALL_NODE_NUM]);
00177
00178             #if DEBUGGING

```

```

00179         labelColorStats();
00180         refreshColorStats();
00181     #endif
00182
00183     /*
00184     #if DEBUGGING
00185         lcdWriteStr("Added:      ",0,0);
00186         lcdWriteStr("      ",1,0);
00187         lcdPrintHex(uncheckedBalls[i][BALL_NODE_NUM],1,0);
00188         waitFor(REDA_BUTTON);
00189     #endif
00190     */
00191
00192     while (seeBall())
00193     {
00194         x -= scanHeight;
00195         setVirtualWindow(x - scanHeight, 1, x, 254);
00196     }
00197 }
00198 }
00199
00200     return foundBall;
00201 }
00202
00203 // returns pixel equivalent of 'distance'
00204 static uint8_t distToPix( uint8_t distance )
00205 {
00206     switch (distance)
00207     {
00208     case 0:
00209     case 1:
00210         return 174;
00211     case 2:
00212         return 0x8d;
00213     case 3:
00214         return 0x61;
00215     case 4:
00216         return 0x48;
00217     case 5:
00218         return 0x36;
00219     case 6:
00220         return 0x2b;
00221     case 7:
00222         return 0x22;
00223     case 8:
00224         return 0x1d;
00225     case 9:
00226         return 0x18;
00227     case 10:
00228         return 0x14;
00229     case 11:
00230         return 0x11;
00231     case 12:
00232         return 0x0e;
00233     default:
00234         return 0x0;
00235     }
00236 }
00237
00238 /*
00239  * Just does right seeks
00240  * PRE - the longest check is the last element of the uncheckedBalls array
00241  *
00242  * uncheckedBalls - ball node numbers and ground distances away from bot
00243  */
00244 bool cameraSeekRight(uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls)
00245 {
00246     bool foundBall = false;    // Return value
00247     uint8_t scanHeight = 4;
00248     uint8_t x = 0;
00249     uint8_t scanLimit = 174 - distToPix(
00250         uncheckedBalls[numUncheckedBalls - 1][BALL_DIST] + 1);
00251
00252     // get pixel ranges for unchecked balls passed in
00253     uint8_t i = 0;
00254     uint8_t maxBallX[3];
00255     while (i + 1 < numUncheckedBalls)
00256     {
00257         maxBallX[i] = ((174 - distToPix(uncheckedBalls[i][BALL_DIST])) +
00258             (174 - distToPix(uncheckedBalls[i + 1][BALL_DIST])))
00259             / 2;

```

```

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

```

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

Index

botCntrl.c, [13](#), [14](#)
botCntrl.h, [21](#)
buttons.c, [22](#), [23](#)
 isPressed, [22](#)
 justPressed, [22](#)
 justReleased, [23](#)
buttons.h, [25](#), [26](#)
 isPressed, [25](#)
 justPressed, [25](#)
 justReleased, [26](#)

caddy.c, [27](#)
 START_DELAY, [27](#)
camera.c, [29](#), [30](#)
 cameraWhiteBalance, [29](#)
 setVirtualWindow, [30](#)
camera.h, [32](#), [33](#)
 cameraWhiteBalance, [32](#)
 setVirtualWindow, [32](#)
cameraWhiteBalance
 camera.c, [29](#)
 camera.h, [32](#)
checkedList
 junctionCode.c, [57](#)

eeProm.c, [33](#), [34](#)
eeProm.h, [36](#), [37](#)
encoder.c, [38](#), [39](#)
encoder.h, [42](#), [43](#)
encoderconf.h, [44](#), [45](#)
exercises.c, [48](#)
exercises.h, [56](#)

generateNextPermutation
 perms.h, [71](#)

isPressed
 buttons.c, [22](#)
 buttons.h, [25](#)

junctionCode.c, [56](#), [57](#)
 checkedList, [57](#)
junctionCode.h, [61](#), [62](#)
justPressed
 buttons.c, [22](#)
 buttons.h, [25](#)
justReleased
 buttons.c, [23](#)
 buttons.h, [26](#)

LINE_LCD_MODE
 tetherUI.h, [73](#)

NAV_LCD_MODE
 tetherUI.h, [73](#)
nodeList.c, [62](#), [63](#)
nodeList.h, [69](#), [70](#)
nodeStruct, [12](#)

PathList, [12](#)
perms.h, [71](#), [72](#)
 generateNextPermutation, [71](#)

START_DELAY
 caddy.c, [27](#)
searchNode, [12](#)
setVirtualWindow
 camera.c, [30](#)
 camera.h, [32](#)
struct_EncoderState, [13](#)

tetherUI.h, [72](#), [73](#)
 LINE_LCD_MODE, [73](#)
 NAV_LCD_MODE, [73](#)
trackColor.c, [73](#), [74](#)
trackColor.h, [78](#), [79](#)
trackLine.h, [80](#), [81](#)