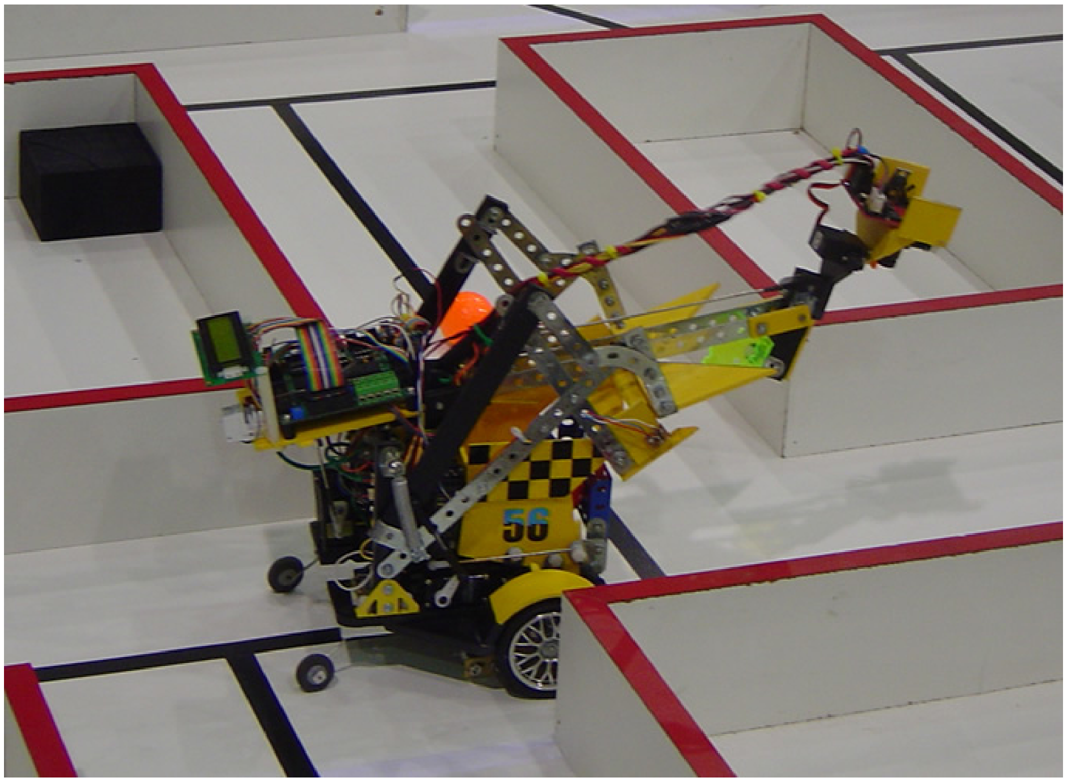
Caddy

A 2005 Roborodentia entry with vision and path planning abilities



By: Taylor Braun-Jones

Advisor: Dr. John Seng

Computer Engineering

California Polytechnic State University - San Luis Obispo

March 2013

Contents

1 Introduction

1.1 Problem Summary

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

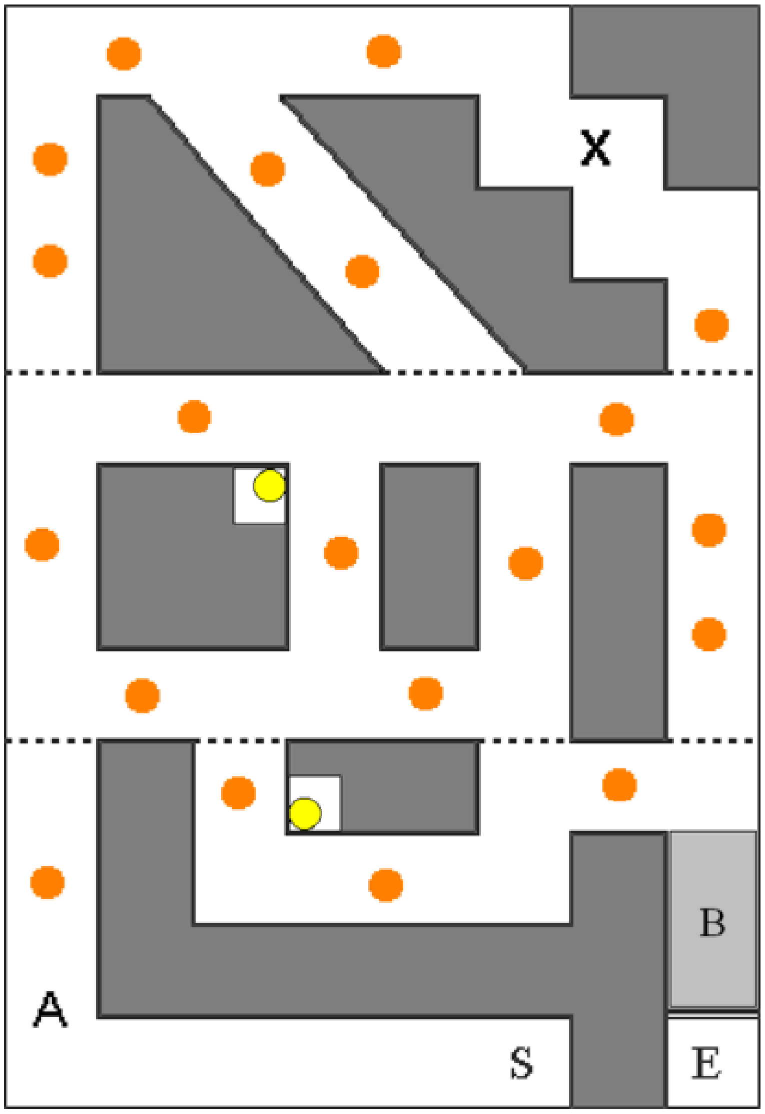


Figure 1: Arena map showing the two fixed bonus ball locations and the potential locations of the randomly placed ground balls

The competition scoring breakdown is as follows:

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

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

2 Goals and Requirements

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

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

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

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

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

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

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

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

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

3 Design

3.1 Collaborative Team Process

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

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

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

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

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

3.2 System Architecture

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

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

3.3 Electrical Design

3.3.1 Power Regulation and Motor Controller

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.

3.3.2 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:

3.3.3 Servo Reverser

The mechanical design of Caddy required 6 servos:

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

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

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

- **Mechanical:** Modify the mechanical design so that the ball pickup mechanism could be controlled by just one high-torque servo. 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.

3.3.4 Wheel Encoders

The maneuvers needed at junctions and for the bonus ball pick up sequences needed to be accurate and repeatable. To achieve this we used a black and white encoder disk that we printed out and glued to the inside edge of each drive wheel. [1]

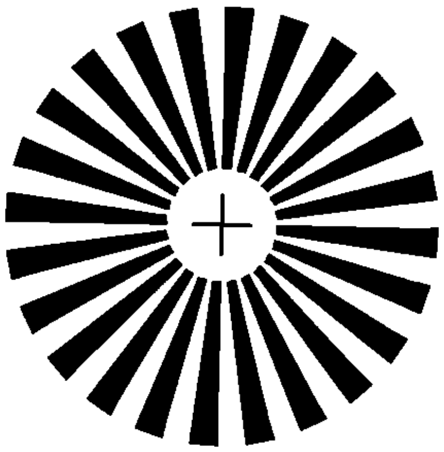


Figure 2: Reflective IR wheel encoder pattern

3.4 Software Architecture

3.4.1 Computing Platform

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

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

3.4.2 PID Line Tracking

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

$$c(t) = P_E e(t) + P_I \int e(t)dt + \P_0 \frac{de}{dt}$$

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.

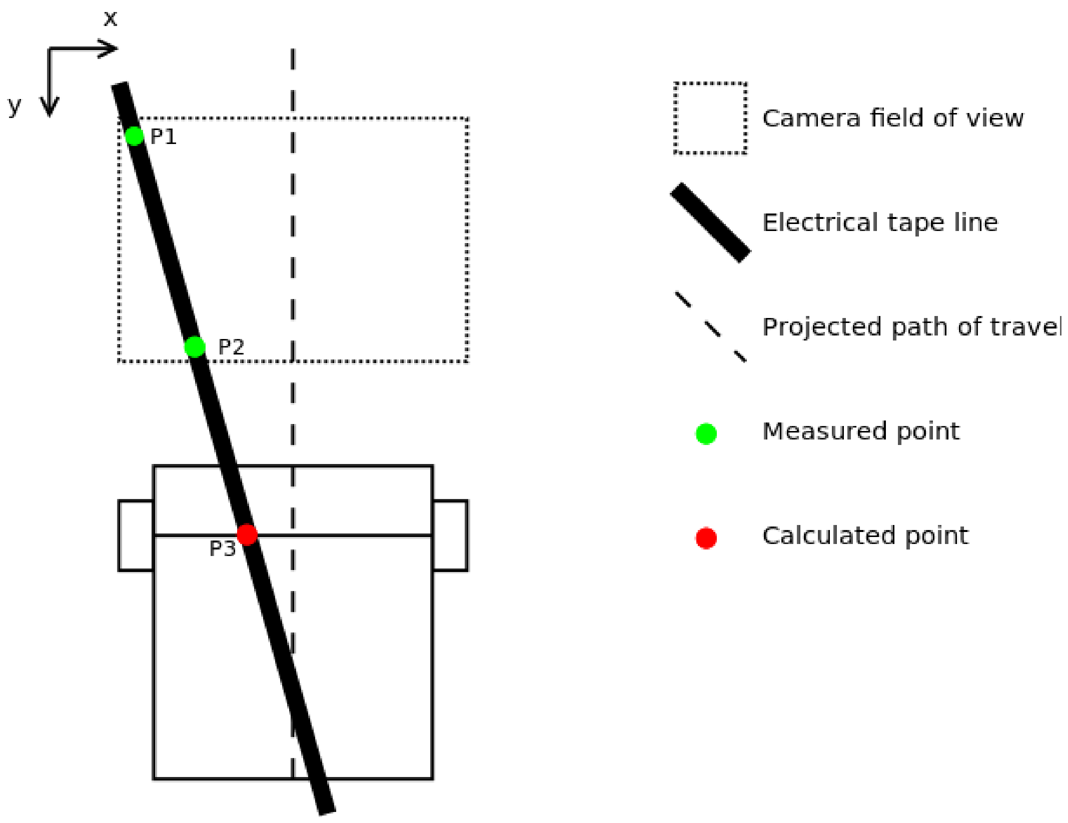


Figure 4: Diagram of line tracking geometry

3.4.3 Maneuvering

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

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

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

3.4.4 Ball Detection and Localization

3.4.5 Path Planning

4 Conclusion

4.1 Future Work

4.1.1 Regulated Motor Voltage

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

4.1.2

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

4.1.3 Quadrature Wheel Encoders

Quadrature wheel encoders would have required more mechanical work (to mount the reflective IR sensors 90 degrees out-of-phase) and electrical work (wiring for twice as many sensors) but it would have helped 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

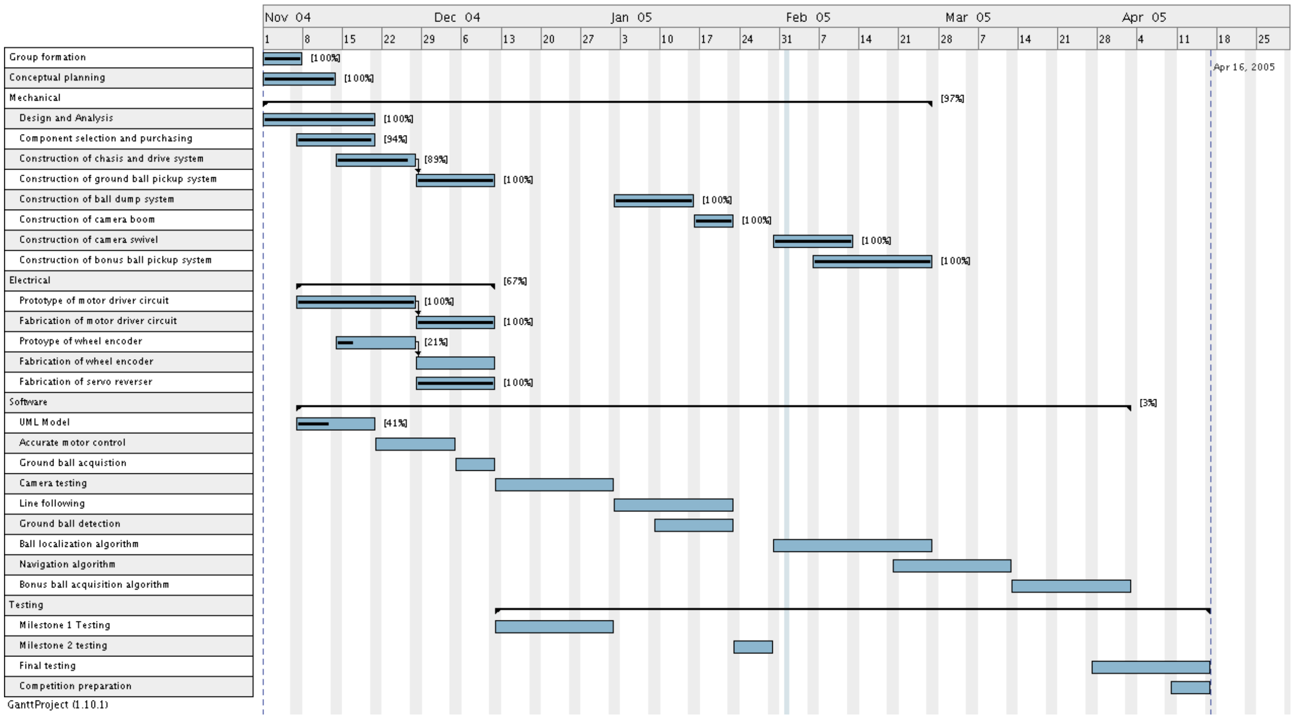


Figure 5: Caddy project gantt chart

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.

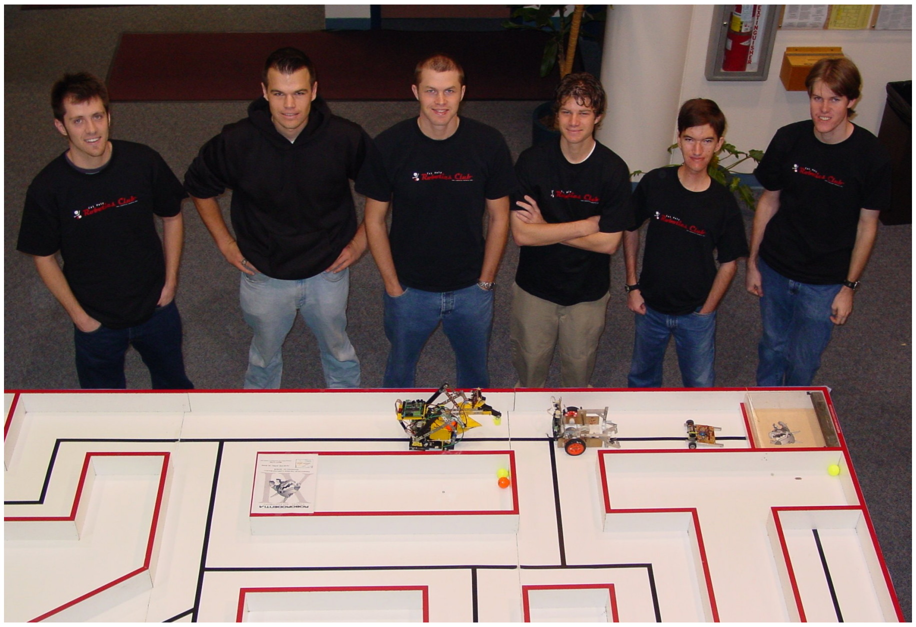


Figure 6: Team Photo. Left to right: Logan Kinde, Tyson Messori, Taylor Braun-Jones, Scott Barlow, Michael Shelley, Patrick McCarty

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:

7 File Index

7.1 File List

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

8 Data Structure Documentation

8.1 nodeStruct Struct Reference

Data Fields

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

8.1.1 Detailed Description

Definition at line 69 of file nodeList.h.

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

- nodeList.h

8.2 PathList Struct Reference

Collaboration diagram for PathList:

Data Fields

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

8.2.1 Detailed Description

Definition at line 23 of file linkedList.h.

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

- linkedList.h

8.3 searchNode Struct Reference

Data Fields

- uint8_t **parent**

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

8.3.1 Detailed Description

Definition at line 38 of file updatePath.h.

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

- updatePath.h

8.4 struct_EncoderState Struct Reference

Encoder state structure.

```
#include <encoder.h>
```

Data Fields

- uint16_t position
  position

8.4.1 Detailed Description

Encoder state structure.

Definition at line 115 of file encoder.h.

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

- encoder.h

9 File Documentation

9.1 botCntrl.c File Reference

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

Include dependency graph for botCntrl.c:

Macros

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

Functions

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

Variables

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

9.1.1 Detailed Description

Definition in file botCntrl.c.

9.2 botCntrl.c

```
/*
 * This file is part of Caddy.
 *
 *  Caddy is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  Caddy is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
 */
#include "botCntrl.h"
#include "trackLine.h"
#include "trackColor.h"
#include "junctionCode.h"
#include "updatePath.h"
#include "motorCntrl.h"
#include "camera.h"
#include "servos.h"
#include "buttons.h"
#include "nodeList.h"
#include "tetherUI.h"
#include "eeProm.h"
#include "ourLCD.h"
#include "helperFunctions.h"

// avr-libc
#include <string.h>

#define BEAM_IGNORE_COUNT      6
```

```
#define CORRAL_COUNT           3
#define LIFT_DONE_COUNT        8

// Global variables
uint8_t botNode = START_NODE;
int8_t botHeading = START_HEADING;
uint8_t numUnreachedGoals = NUM_GOALS;

static bool liftDown;
static uint8_t upComingBallNum;

static inline int8_t getNextHeading(uint8_t nextBotNode);

inline void runRoborodentiaCourse(void)
{
    bool justTurned = false;
    bool firstRun = true;

    updatePath();

    // run through first leg, skipping positionBot
    junctionCode();
    moveToJunction(1, justTurned);

#if DEBUGGING
    if (lcdMode == NAV_LCD_MODE)
    {
        lcdWriteStr("                ", 0, 0);
        lcdWriteStr("                ", 1, 0);
        lcdPrintDecU08(botNode, 1, 0);
        lcdPrintDecS08(botHeading, 1, 3);
    }
#endif

    // run through arena
    while (pathList[pathListIndex + 1] != STOP_NODE)
    {
        junctionCode();       // ball search, bonous ball pickup, best path
 code

        justTurned = positionBot();      // turning, preparing for linetracking

        if (firstRun)
        {
            firstRun = false;
            setServo(LIFT, LIFT_OPEN); // Lower lift, on first run, b/c
 skipping seek at node 21
            msDelay(30);
            upComingBallNum = 1;
            liftDown = true;
        }

#if DEBUGGING
        if (lcdMode == NAV_LCD_MODE)
        {
            lcdPrintDecS08(botHeading, 1, 3);
        }
#endif

        moveToJunction(1, justTurned);   // linetracking, ground ball pickup

#if DEBUGGING
        if (lcdMode == NAV_LCD_MODE)
        {
            lcdPrintDecU08(botNode, 1, 0);
        }
#endif
    }

    if (pathList[pathListIndex + 1] == STOP_NODE)
    {
        positionBot();
        nestSequence();
    }
}

/*
 *  Initializes some of bot's global variables
 */
inline void initBotGlobals(void)
{
    // init bot's path to INITIAL_PATH_LIST
    pathListIndex = 0;
    pathListSize = INITIAL_PATH_LIST_SIZE;
    // (pathList initialized in updatePath.c)

    initGoalList();
    numKnownGoals = NUM_FIXED_GOALS;

    liftDown = false;
    upComingBallNum = 0;
}

/*
 * @brief Turn bot at junctions and, if necessary, ball nodes
 *
 * Maintains botHeading. Performs bonus ball pickup liftDown actions
 *
 * @return True when bot just turned. (Used to tell moveToJunction to
 * begin looking for next junction immediately.)
 *
 * PRE: camera is not streaming
 */
inline bool positionBot(void)
{
    bool justTurned = true;

    int8_t nextHeading = getNextHeading(pathList[pathListIndex + 1]);
    int8_t bradsToTurn = nextHeading - botHeading;

    // BB PICKUP CHECK
    if (botNode == BONUS_BALL_1 && isInGoalList(BONUS_BALL_1))
    {
        bbPositioning(BB1_HEADING, nextHeading);
        removeFromGoalList(BONUS_BALL_1);
    }
    else if (botNode == BONUS_BALL_2 && isInGoalList(BONUS_BALL_2))
    {
        bbPositioning(BB2_HEADING, nextHeading);
        removeFromGoalList(BONUS_BALL_2);
    }

    // TURN/STRAIGHT CHECK
    else if (bradsToTurn != 0)
    {
        int8_t ticksToTurn;
        switch ((int8_t) bradsToTurn)
        {
        case -128: // U-turn
            if (botNode == 37)
            {
                moveToJunction(1, false);
                tickWheels(20, 20, 255);
                msDelay(0x50);
                moveStraight(-20, 255);
            }
            //tankTurn(245, -58);
            tickWheels(-29, 29, 250);
            tankTurn(245, -58);
            break;
        case -105: // Hard Diagonal
            tickWheels(28, 28, 250);      //28
            tractorTurn(255, -tempTweak4);
            tankTurn(250, -70);          //-80
            break;
        case 23: // Soft Diagonal
            tractorTurn(255, 23);        //23
            break;
        case -23:
            tractorTurn(255, -28);
            break;
        case 105:
            tickWheels(17, 17, 250);
            tankTurn(250, 80);           //80
            break;
        default:
            // fixed ticks forward here?

            // convert brads to turn to ticks and turn
            if (bradsToTurn < 0)
            {
                ticksToTurn = bradsToTurn + turnSubtract;
            } else
            {
                ticksToTurn = bradsToTurn - turnSubtract;
            }
            tractorTurn(255, ticksToTurn);
            break;
        }
    }
    else
    {
        justTurned = false;
    }

    if (botNode == SENSOR_NODE)
    {
        removeFromGoalList(SENSOR_NODE);
    }

    // update botHeading
    botHeading = nextHeading;

    // GB PICKUP CHECK: lower lift, if bot knows it will travel over ball
    upComingBallNum = getUpcomingBallNum();
    if (upComingBallNum != 0)
    {
        setServo(LIFT, LIFT_OPEN);
        msDelay(30);
        liftDown = true;
    }

    return justTurned;
}

/*
 * Returns absolute heading of next node given botNode and the next botNode.
 */
static inline int8_t getNextHeading(uint8_t nextBotNode)
{
    NODE nextNode;            // info about nodes adjacent to botNode
    int8_t nextNodeIndex;        // nextNode offset to nextBotNode
    int8_t nextHeading;          // absolute direction to nextBotNode

    // get absolute direction of nextBotNode from node list
    getNode(botNode, &nextNode);
    nextNodeIndex = findValue(nextNode.adjNodes,
                              nextNode.numAdjNodes,
                              nextBotNode);

    // get next heading or report error
    if (nextNodeIndex == -1)
    {
#if DEBUGGING
        lcdWriteStr("pathList error  ", 0, 0);
#endif
        brake(BOTH);
        while (1) ;
    }
    nextHeading = nextNode.adjHeadings[nextNodeIndex];

    return nextHeading;
}

/* Rotates bot before and after Bonus Ball grab
 *    bbHeading   - heading bot must have for bb pickup.
 *    nextHeading - heading bot must have after bb pickup
 */
inline void bbPositioning(int8_t bbHeading, int8_t nextHeading)
{
    // move forward (camera will be over junction at this point)
    // May or may not need to move foward (requires testing)
    //    Some are fine without foward, some seem to need it
    //    Right now, only -32 case moves forward

    // rotate by (bbHeading - botHeading)
    switch ((int8_t) (bbHeading - botHeading))
    {
    case 96:
        // example of 96 brad rotation
        tickWheels(28, 0, 255); // allows fluid motion (no overshoot
 correction)
        tankTurn(255, 58);      //58
        break;
    case -96:
        tickWheels(0, 28, 255); // allows fluid motion (no overshoot
 correction)
        tankTurn(255, -64);     //-64
        break;
    case -32:
        tickWheels(10, 10, 255); //10 Move bot forward a few ticks to make it
 correctly aligned
        tickWheels(0, 32, 255);                  //28
        break;
    default:
#if DEBUGGING
        lcdWriteStr("ERROR:          ", 0, 0);
        lcdWriteStr("Turn Amt =      ", 1, 0);
        lcdPrintDecS08(bbHeading - botHeading, 1, 11);
        brake(BOTH);
        while (1) ;
#endif
        break;
    }

    grabBonusBall(); // Grab the BB

    // Rotate by (nextHeading - bbHeading)
    // (This should only be 32, -32, or -96)
    switch ((int8_t) (nextHeading - bbHeading))
    {
    case 32:
        tankTurn(250, 32);
        break;
    case -32:
        tankTurn(250, -32);
        break;
    case -96:
        tankTurn(250, -90);
        break;
    default:        // Error, this should only be 32, -32, or -96
#if DEBUGGING
        lcdWriteStr("ERROR:          ", 0, 0);
        lcdWriteStr("nH - bbH =      ", 1, 0);
        lcdPrintDecS08(bbHeading - botHeading, 1, 11);
        brake(BOTH);
        while (1) ;
#endif
        break;
    }
}

/*
 * Moves to next junction in pathList.
 */
inline void moveToJunction(uint8_t numJunctions, bool justTurned)
{
    bool onLine = true;
    bool juncApproaching = false;
    uint8_t juncCount = 0;

    uint8_t ignoreJuncCount;
    if (!justTurned)
    {
        ignoreJuncCount = 3;
    } else
    {
        ignoreJuncCount = 0;
    }

    uint8_t pickingUp = false;
    uint8_t pickingUpCount = 0;

    uint8_t ignoreBreakBeamCount = BEAM_IGNORE_COUNT;

    trackLineInit();

    // Linetrack, until bot is at junction or nest.
    // If see ground ball, pickup it up and continue linetracking.
    while (onLine)
    {
        while (lineStatsProcessed) ;

        analyzeLineStats();
```

```c
        adjustPWM();

        // CURRENT JUNCTION IGNORE
        if (ignoreJuncCount > 0 && junctionY == 0)
        {
            ignoreJuncCount--;
        }

        // JUNCTION CHECK
        if (ignoreJuncCount == 0 && junctionY != 0)
        {
            if (junctionY < turnPoint)
            {
                juncApproaching = true;
            } else if (juncApproaching)
            {
                juncApproaching = false;
                juncCount++;

                // set botNode to next junction in pathList
                do
                {
                    pathListIndex++;
                    botNode = pathList[pathListIndex];
                } while (!isJunction(botNode));

                // Break out of line tracking
                if (juncCount >= numJunctions)
                {
                    onLine = false;
                }
            }
        }

        // STOP IGNORING BEAM CHECK
        if (liftDown && ignoreBreakBeamCount != 0)
        {
            ignoreBreakBeamCount--;
        }

        // BEGIN PICKUP CHECK
        if (liftDown && ignoreBreakBeamCount == 0 && BREAK_BEAM_TRIGGERED)
        {
            streamModeOff();
            setServo(LIFT, LIFT_CORRAL); // Perhaps raise it slowly if there are pick-up problems
            msDelay(30);
            trackLineInit();

            liftDown = false;
            pickingUp = true;
            pickingUpCount = 0;
        }

        // COMPLETE/STOP LIFTING CHECK
        if (pickingUp)
        {
            pickingUpCount++;

            if (pickingUpCount == CORRAL_COUNT)
            {
                streamModeOff();
                setServo(LIFT, LIFT_UP);
                trackLineInit();
            }

            if (pickingUpCount == LIFT_DONE_COUNT)
            {
                pickingUp = false;

                // Set current botNode to node where this ball is
                botNode = upComingBallNum;
                removeFromGoalList(upComingBallNum);

                if (upComingBallNum == 1) // account for ball not found by camera prior to pickup
                {
                    numKnownGoals++;
                }

                // Find correct pathListIndex
                while (botNode != pathList[pathListIndex])
                {
                    pathListIndex++;
                }

                streamModeOff();      // Turn off line tracking
                disableServo(LIFT);
                positionBot(); // In case we want to make a -128 brad turn after picking up ball
                ignoreBreakBeamCount = BEAM_IGNORE_COUNT;
                trackLineInit();       // Turn line tracking back on
            }

        }
    }

    streamModeOff();

    // Make sure lift is up (in case we missed a ball or incorrectly thought one was there)
    if (liftDown)
    {
        brake(BOTH);
#if DEBUGGING
        lcdWriteStr("No ball         ", 0, 0);
#endif
        setServo(LIFT, LIFT_UP);    // Raise the lift
        msDelay(700);
        disableServo(LIFT);
        liftDown = false;

        // correct goal state
        removeFromGoalList(upComingBallNum);
        numUnreachedGoals--;
        numKnownGoals--;
    }
}

void nestSequence(void)
{
    // line track, until NEST_BUTTON is pressed
    trackLineInit();

    while (!justPressed(NEST_BUTTON))
    {
        if (!lineStatsProcessed)
        {
            analyzeLineStats();
            adjustPWM();
        }

        debounceButtons();
    }

    brake(BOTH);
    streamModeOff();
    setServo(LIFT, LIFT_UP);       // Turn lift on
    msDelay(300);

    // Open door, back up, close door
    setServo(DOOR, DOOR_OPEN);
    moveStraight(-1, 255);              // Back up to take pressure off button
    brake(BOTH);
    //myDelay(25);                     // Let balls roll out
    msDelay(3000);
    setServo(DOOR, DOOR_CLOSED); // Leaves door closed, so lift and door don't colide on power up.
    //myDelay(10);                     // Wait for door to close
    msDelay(1000);

    // Disable all servos
    disableServo(PAN);
    disableServo(TILT);
    disableServo(BOOM);
    disableServo(LIFT);
    disableServo(DOOR);
}
```

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

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

// avr-libc
#include <stdint.h>
#include <stdbool.h>

```

```
// Global variables
extern uint8_t botNode;
extern int8_t botHeading;
extern uint8_t numUnreachedGoals;

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

#endif // #ifndef BOTCNTRL_H_
```

9.5 buttons.c File Reference

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

Include dependency graph for buttons.c:

Macros

- #define **DEBOUNCE_COUNT** 3

Functions

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

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

9.5.1 Detailed Description

Definition in file buttons.c.

9.5.2 Function Documentation

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

Returns

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

Definition at line 114 of file buttons.c.

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

Returns

true when confirmed rising edge at last debouncing.

Definition at line 50 of file buttons.c.

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

Returns

true when confirmed falling edge at last debouncing.

Definition at line 58 of file buttons.c.

9.6 buttons.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "buttons.h"
00019
00020 // AVRLIB
00021 #include "avrlibdefs.h"
00022
00023 // avr-libc
00024 #include <stdint.h>
00025 #include <stdbool.h>
00026
00027 #define DEBOUNCE_COUNT   3   // should be equal to 2 or greater
00028
00029 static bool isDown[NUM_BUTTONS]   = { false, false, false,
00030                                       false, false, false };
00031 static bool wasEvent[NUM_BUTTONS] = { false, false, false,
00032                                       false, false, false };
00033 static uint8_t upCount[NUM_BUTTONS] = { DEBOUNCE_COUNT, DEBOUNCE_COUNT,
00034                                         DEBOUNCE_COUNT, DEBOUNCE_COUNT,
00035                                         DEBOUNCE_COUNT, DEBOUNCE_COUNT };
00036 static uint8_t downCount[NUM_BUTTONS] = { 0, 0, 0, 0, 0, 0 };
00037
00038 void waitFor(uint8_t button)
00039 {
00040     debounceButtons();
00041     while (!justReleased(button))
00042     {
00043         debounceButtons();
00044     }
00045 }
00046
00050 inline bool justPressed(uint8_t button)
00051 {
00052     return wasEvent[button] && isDown[button];
00053 }
00054
00058 inline bool justReleased(uint8_t button)
00059 {
00060     return wasEvent[button] && !isDown[button];
00061 }
00062
```

```c
void debounceButtons(void)
{
    uint8_t button;
    for (button = 0; button < NUM_BUTTONS; button++)
    {
        // count times buttons have been consecutively up/down (upto DEBOUNCE_COUNT).
        if (isPressed(button))
        {
            downCount[button] = MIN(downCount[button]+1, DEBOUNCE_COUNT);
            upCount[button] = 0;
        }
        else
        {
            upCount[button] = MIN(upCount[button]+1, DEBOUNCE_COUNT);
            downCount[button] = 0;
        }

        // check for confirmed up/down event
        if (isDown[button])
        {
            if (upCount[button] >= DEBOUNCE_COUNT)
            {
                isDown[button] = false;
                wasEvent[button] = true;
            }
            else
            {
                wasEvent[button] = false;
            }
        }
        else
        {
            if (downCount[button] >= DEBOUNCE_COUNT)
            {
                isDown[button] = true;
                wasEvent[button] = true;
            }
            else
            {
                wasEvent[button] = false;
            }
        }
    }
}

inline bool isPressed(uint8_t button)
{
    switch (button)
    {
    case RED_BUTTON:      return RED_BUTTON_DOWN;
    case L_UP_BUTTON:     return L_UP_BUTTON_DOWN;
    case L_DOWN_BUTTON:   return L_DOWN_BUTTON_DOWN;
    case R_UP_BUTTON:     return R_UP_BUTTON_DOWN;
    case R_DOWN_BUTTON:   return R_DOWN_BUTTON_DOWN;
    case NEST_BUTTON:     return NEST_BUTTON_DOWN;
    default:              break;
    }

    return false;
}

inline bool bothRightButtonsPressed(void)
{
    return (justPressed(R_UP_BUTTON) && justPressed(R_DOWN_BUTTON)) ||
           (justPressed(R_UP_BUTTON) && isDown[R_DOWN_BUTTON]) ||
           (justPressed(R_DOWN_BUTTON) && isDown[R_UP_BUTTON]);

}

inline bool bothLeftButtonsPressed(void)
{
    return (justPressed(L_UP_BUTTON) && justPressed(L_DOWN_BUTTON)) ||
           (justPressed(L_UP_BUTTON) && isDown[L_DOWN_BUTTON]) ||
           (justPressed(L_DOWN_BUTTON) && isDown[L_UP_BUTTON]);

}
```

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

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

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

#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    // change isPressed(uint8_t button) when adding a button

#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)

void initButtons(void);
void waitFor(uint8_t button);
inline bool justPressed(uint8_t button);
inline bool justReleased(uint8_t button);
void debounceButtons(void);
inline bool isPressed(uint8_t button);
inline bool bothRightButtonsPressed(void);
inline bool bothLeftButtonsPressed(void);

#endif // #ifndef BUTTONS_H_
```

9.9 caddy.c File Reference

Caddy's main loop and Atmel initialization.

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

Include dependency graph for caddy.c:

Macros

- #define START_DELAY 5

Functions

- int main (void)

  Caddy's power-on entry function.

9.9.1 Detailed Description

Caddy's main loop and Atmel initialization.

Definition in file caddy.c.

9.9.2 Macro Definition Documentation

9.9.2.1 #define START_DELAY 5

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

Definition at line 38 of file caddy.c.

9.10 caddy.c

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

```c
#include "botCntrl.h"
#include "motorCntrl.h"
#include "camera.h"
#include "servos.h"
#include "encoder.h"
#include "buttons.h"
#include "eeProm.h"
#include "helperFunctions.h"
#include "ourLCD.h"

// avr-libc
#include <avr/io.h>

#define START_DELAY 5

static inline void initAtmel(void)
{
    /*
     * Initialize Timer
     */
    timerInit();

#if DEBUGGING
    /*
     * Initialize LCD
     */
    lcdInit();
    lcdWriteStr("Init:                ", 0, 0);
    lcdWriteStr("                     ", 1, 0);
#endif

    /*
     * Initialize UART for CMUcam communication
     */
    cmuCamInit();

    /*
     * Initialize PWM motor control
     */
    outb(DDRD, 0xff);
    timer1PWMInit(8);
    neutral();
    enableMotors();

    /*
     * Set data direction registers
     */
    outb(DDRA, 0xF0); // Motor control and up/down buttons
    cbi(DDRD, 6);     // red button
    cbi(DDRB, 0);     // nest button
    cbi(DDRB, 1);     // break beam

    /*
     * Apply internal pull-up resister to certain digital inputs
     */
    sbi(PORTB, 0); // internal pull-up for PINB0
    sbi(PORTA, 3); // internal pull-up for PINA3
    sbi(PORTA, 2); // internal pull-up for PINA2
    sbi(PORTA, 1); // internal pull-up for PINA1
    sbi(PORTA, 0); // internal pull-up for PINA0

    /*
     * Initialize quadrature wheel encoders
     */
    cbi(DDRD, 2);
    cbi(DDRD, 3);
    encoderInit();
}

int main(void)
{
    initAtmel();
    loadTweakValues();
    initBotGlobals();
    resetCamera();
    moveServosToStart();
    cameraWhiteBalance();

#if DEBUGGING
    runTetherUI();
    myDelay(START_DELAY);
    runTest();
#else
    waitFor(RED_BUTTON);
    myDelay(START_DELAY);
    runRoborodentiaCourse();
#endif

    brake(BOTH);
#if DEBUGGING
    lcdWriteStr("Done            ", 0, 0);
    lcdWriteStr("                ", 1, 0);
#endif

    return 0;
}
```

9.11 camera.c File Reference

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

Include dependency graph for camera.c:

Macros

- #define **CMU_BAUD** 38400

Functions

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

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

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

  Constrain field of view used for subsequent image processing commands.

9.11.1 Detailed Description

Definition in file camera.c.

9.11.2 Function Documentation

9.11.2.1 void cameraWhiteBalance (void) [inline]

Optimize the white balance for current conditions.

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

See Also

CMUcam2 manual p.31

Definition at line 48 of file camera.c.

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

Constrain field of view used for subsequent image processing commands.

See Also

CMUcam2 manual p.55

Definition at line 142 of file camera.c.

9.12 camera.c

```c
/*
 * This file is part of Caddy.
 *
 *  Caddy is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  Caddy is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
 */
#include "camera.h"
#include "trackColor.h"
#include "trackLine.h"
#include "helperFunctions.h"
#include "ourLCD.h"

// AVRLIB
#include "rprintf.h"
#include "uart.h"

// avr-libc
#include <stdbool.h>

#define CMU_BAUD 38400

static uint8_t mode;
static uint16_t byteNum;

void packetRcv( uint8_t c );
inline void lineMode2Rcv( uint8_t c );
inline void trackColorRcv( uint8_t c );

inline void cmuCamInit(void)
{
    uartInit();
    uartSetBaudRate(CMU_BAUD);
    uartSetRxHandler(packetRcv);
    rprintfInit(uartSendByte);
}

```

```
inline void cameraWhiteBalance()
{
    // turn auto white balance on
#if DEBUGGING
    lcdWriteStr("white Bal ", 0, 6);
#endif
    rprintf("CR 18 44\r");
    myDelay(200);
    // turn auto white balance off
    rprintf("CR 18 40\r");
}

inline void resetCamera( void )
{
    mode = NEW_PACKET;
    byteNum = 0;

    rprintf("RM 3\r");
}

void packetRcv(uint8_t c)
{
    if (c == 0xff)
    {
        mode = NEW_PACKET;
        byteNum = 0;
    }
    else
    {
        switch (mode)
        {
        case NEW_PACKET:
            switch (c)
            {
            case 0xfe:
                mode = FE_RCV;
                break;
            case 'T':
                mode = T_RCV;
                break;
            default:
                break;
            }
            break;
        case FE_RCV:
            lineMode2Rcv(c);
            if (c == 0xfd)
            {
                mode = NEW_PACKET;
            }
            break;
        case T_RCV:
            trackColorRcv(c);
            break;
        }
    }
}


inline void lineMode2Rcv(uint8_t c)
{
    if (c == 0xfd)
    {
        lineStatsProcessed = false;
        byteNum = 0;
    }
    else
    {
        lineStats[(byteNum - 1) / LINE_STATS_COLS]
                 [(byteNum - 1) % LINE_STATS_COLS] = c;
        byteNum++;
    }
}


inline void trackColorRcv(uint8_t c)
{
    lineStats[0][byteNum] = c;
    byteNum++;

    if (byteNum >= NUM_COLOR_STATS)
    {
        colorStatsProcessed = false;
    }
}


inline void streamModeOff( void )
{
    rprintf("\r\r"); // add an extra return as recommended by CMUcam manual
    msDelay(32);     // wait for streaming to stop ( 16ms delay ok )
}


inline void setVirtualWindow(uint8_t x1, uint8_t y1, uint8_t x2
, uint8_t y2)
{
    rprintf("VW %d %d %d %d\r", x1, y1, x2, y2);
}
```

9.13 camera.h File Reference

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

Macros

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

Functions

- void cmuCamInit (void)

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

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

  Constrain field of view used for subsequent image processing commands.

9.13.1 Detailed Description

Definition in file camera.h.

9.13.2 Function Documentation

9.13.2.1 void cameraWhiteBalance (void) [inline]

Optimize the white balance for current conditions.

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

See Also

CMUcam2 manual p.31

Definition at line 48 of file camera.c.

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

Constrain field of view used for subsequent image processing commands.

See Also

CMUcam2 manual p.55

Definition at line 142 of file camera.c.

9.14 camera.h

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

#include <stdint.h>

// Packet types
#define NEW_PACKET   0
#define FE_RCV       1
#define T_RCV        2

#define hiResMode()  rprintf("HR 1\r")
#define lowResMode() rprintf("HR 0\r")

inline void cmuCamInit(void);

inline void cameraWhiteBalance( void );

inline void resetCamera( void );
inline void streamModeOff( void );

inline void setVirtualWindow(uint8_t x1, uint8_t y1, uint8_t x2
, uint8_t y2);

#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

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

// avr-libc
#include <avr/io.h>
#include <avr/interrupt.h>

// Global variables - Runtime configurable parameters
uint8_t l_base;
uint8_t r_base;
uint16_t slopeCoef;
```

```c
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;  // <- need debugger menu for this, remove old init/toggling, and save in eeProm
uint8_t testMode; // <- need to save this in eeProm

// Initializes constants that can be tweaked by debugger
inline void loadTweakValues(void)
{
    cli(); // disable all interrupts

    // EEPROM Reads
    l_base                = EEPROM_read(EE_ADDR_LEFT_BASE);
    r_base                = EEPROM_read(EE_ADDR_RIGHT_BASE);
    slopeCoef             = (EEPROM_read(EE_ADDR_SLOPE_COEF) << 8) +
                             EEPROM_read(EE_ADDR_SLOPE_COEF + 1);
    offCoef               = (EEPROM_read(EE_ADDR_OFF_COEF) << 8) +
                             EEPROM_read(EE_ADDR_OFF_COEF + 1);
    dampCoef              = EEPROM_read(EE_ADDR_DAMP_COEF);
    lineCenter            = EEPROM_read(EE_ADDR_LINE_X_CENTER);
    turnPoint             = EEPROM_read(EE_ADDR_TURN_POINT);
    turnSubtract          = EEPROM_read(EE_ADDR_TURN_SUBTRACT);
    panOffset             = EEPROM_read(EE_ADDR_PAN_OFFSET);
    tiltOffset            = EEPROM_read(EE_ADDR_TILT_OFFSET);
    tractorOvershootDelay = (EEPROM_read(EE_ADDR_TRACTOR_OVERSHOOT_DELAY) << 8) +
                             EEPROM_read(EE_ADDR_TRACTOR_OVERSHOOT_DELAY + 1);
    testMode              = EEPROM_read(EE_ADDR_TEST_MODE);
    tempTweak1            = EEPROM_read(EE_ADDR_TEMP_TWEAK1);
    tempTweak2            = EEPROM_read(EE_ADDR_TEMP_TWEAK2);
    tempTweak3            = (EEPROM_read(EE_ADDR_TEMP_TWEAK3) << 8) +
                             EEPROM_read(EE_ADDR_TEMP_TWEAK3 + 1);
    tempTweak4            = (EEPROM_read(EE_ADDR_TEMP_TWEAK4) << 8) +
                             EEPROM_read(EE_ADDR_TEMP_TWEAK4 + 1);

    sei(); // enable all interrupts
}

// Saves constants after they have been changed by the debugger
inline void storeTweakValues(void)
{
    cli(); // disable all interrupts

    // EEPROM writes
    EEPROM_write(EE_ADDR_LEFT_BASE, l_base);
    EEPROM_write(EE_ADDR_RIGHT_BASE, r_base);
    EEPROM_write(EE_ADDR_SLOPE_COEF, slopeCoef >> 8);
    EEPROM_write(EE_ADDR_SLOPE_COEF + 1, slopeCoef);
    EEPROM_write(EE_ADDR_OFF_COEF, offCoef >> 8);
    EEPROM_write(EE_ADDR_OFF_COEF + 1, offCoef);
    EEPROM_write(EE_ADDR_DAMP_COEF, dampCoef);
    EEPROM_write(EE_ADDR_LINE_X_CENTER, lineCenter);
    EEPROM_write(EE_ADDR_TURN_POINT, turnPoint);
    EEPROM_write(EE_ADDR_TURN_SUBTRACT, turnSubtract);
    EEPROM_write(EE_ADDR_PAN_OFFSET, panOffset);
    EEPROM_write(EE_ADDR_TILT_OFFSET, tiltOffset);
    EEPROM_write(EE_ADDR_TRACTOR_OVERSHOOT_DELAY, tractorOvershootDelay >> 8);
    EEPROM_write(EE_ADDR_TRACTOR_OVERSHOOT_DELAY + 1, tractorOvershootDelay);
    EEPROM_write(EE_ADDR_TEST_MODE, testMode);
    EEPROM_write(EE_ADDR_TEMP_TWEAK1, tempTweak1);
    EEPROM_write(EE_ADDR_TEMP_TWEAK2, tempTweak2);
    EEPROM_write(EE_ADDR_TEMP_TWEAK3, tempTweak3 >> 8);
    EEPROM_write(EE_ADDR_TEMP_TWEAK3 + 1, tempTweak3);
    EEPROM_write(EE_ADDR_TEMP_TWEAK4, tempTweak4 >> 8);
    EEPROM_write(EE_ADDR_TEMP_TWEAK4 + 1, tempTweak4);

    sei(); // enable all interrupts
}

uint8_t EEPROM_read(unsigned int uiAddress)
```

```
{
    // Wait for completion of previous write
    while (EECR & (1 << EEWE)) ;
    // Set up address register
    EEAR = uiAddress;
    // Start eeprom read by writing EERE
    EECR |= (1 << EERE);
    // Return data from data register
    return EEDR;
}

void EEPROM_write(unsigned int uiAddress, uint8_t ucData)
{
    // Wait for completion of previous write
    while (EECR & (1 << EEWE)) ;
    // Set up address and data registers
    EEAR = uiAddress;
    EEDR = ucData;
    // Write logical one to EEMWE
    EECR |= (1 << EEMWE);
    // Start eeprom write by setting EEWE
    EECR |= (1 << EEWE);
    // EEMWE and EEWE are automatically cleared back to 0 by hardware
}
```

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

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

#include <stdint.h>

//Locations in EEPROM
#define EE_ADDR_LEFT_BASE           0x50
```

```
#define EE_ADDR_RIGHT_BASE          0x51
#define EE_ADDR_SLOPE_COEF          0x52 //uint16_t
#define EE_ADDR_OFF_COEF            0x54 //uint16_t
#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 //uint16_t
#define EE_ADDR_TEST_MODE           0x5E
#define EE_ADDR_TEMP_TWEAK1         0x5F
#define EE_ADDR_TEMP_TWEAK2         0x60
#define EE_ADDR_TEMP_TWEAK3         0x61 //uint16_t
#define EE_ADDR_TEMP_TWEAK4         0x63 //uint16_t
//next address 0x65

/*
// Current values - Bigger motors at 6 Volts
#define INIT_LEFT_BASE_SPEED        0xF7
#define INIT_RIGHT_BASE_SPEED       0xF0
#define INIT_SLOPE_COEF             0x0110   // <--- 0x110 could be tweaked
#define INIT_OFF_COEF               0x0001
#define INIT_DAMP_COEF              0x01     // <--- 0x01 could be tweaked
#define INIT_LINE_X_CENTER          0x25
#define INIT_TURN_POINT             0x15     // Turn values
#define INIT_TURN_SUBTRACT          0x0A
pan offset 0x05
tilt offset 0xE7

#define TRACTOR_OVERSHOOT_DELAY     5000
*/

#define BASE_MIN                    0x60     // <--- also worked at 0xB0
#define BASE_MAX                    0xFF

// Pickup values
#define BALL_CHECK_RATIO            16
#define PICK_UP_POINT               0x16


/*
// Old values - 6V Solarbotics before damping fix
#define LEFT_BASE_SPEED           0x8C
#define RIGHT_BASE_SPEED          0xC2
#define BASE_MIN                  100
#define BASE_MAX                  255
#define OFF_COEF                  0x2
#define SLOPE_COEF                0x100
#define DAMP_COEF                 0x14
*/

// Global variables - Runtime configurable parameters
extern uint8_t l_base;
extern uint8_t r_base;
extern uint16_t slopeCoef;
extern uint16_t offCoef;
extern uint8_t dampCoef;
extern uint8_t lineCenter;
extern uint8_t turnPoint;
extern uint8_t turnSubtract;
extern int8_t panOffset;
extern int8_t tiltOffset;
extern uint16_t tractorOvershootDelay;
extern uint8_t tempTweak1;
extern int8_t tempTweak2;
extern uint16_t tempTweak3;
extern uint16_t tempTweak4;

extern uint8_t lcdMode;
extern uint8_t testMode;

inline void loadTweakValues( void );
inline void storeTweakValues( void );
uint8_t EEPROM_read(unsigned int uiAddress);
void EEPROM_write(unsigned int uiAddress, uint8_t ucData);

#endif // #ifndef EEPROM_H_
```

9.19 encoder.c File Reference

Quadrature Encoder reader/driver.

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

Include dependency graph for encoder.c:

Functions

- void encoderInit (void)

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

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

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

  Encoder 0 interrupt handler.
- SIGNAL (ENC1_SIGNAL)

  Encoder 1 interrupt handler.

Variables

- volatile EncoderStateType **EncoderState** [NUM_ENCODERS]

9.19.1 Detailed Description

Quadrature Encoder reader/driver.

Definition in file encoder.c.

9.20 encoder.c

```
/*
 * This file is part of Caddy.
 *
 *  Caddy is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  Caddy is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
 */
//*****************************************************************************
//
// File Name    : 'encoder.c'
// Title        : Quadrature Encoder reader/driver
// Author       : Pascal Stang - Copyright (C) 2003-2004
// Created      : 2003.01.26
// Revised      : 2004.06.25
// Version      : 0.3
// Target MCU : Atmel AVR Series
// Editor Tabs  : 4
//
// NOTE: This code is currently below version 1.0, and therefore is considered
// to be lacking in some functionality or documentation, or may not be fully
// tested.  Nonetheless, you can expect most functions to work.
//
// This code is distributed under the GNU Public License
//    which can be found at http://www.gnu.org/licenses/gpl.txt
//
//*****************************************************************************
#ifndef WIN32
#include <avr/io.h>
#include <avr/interrupt.h>
#endif

#include "global.h"
#include "encoder.h"

// Program ROM constants

// Global variables
volatile EncoderStateType EncoderState[NUM_ENCODERS];

// Functions

// encoderInit() initializes hardware and encoder position readings
//    Run this init routine once before using any other encoder functions.
inline void encoderInit(void)
{
    uint8_t i;

    // initialize/clear encoder data
    for (i = 0; i < NUM_ENCODERS; i++)
    {
        EncoderState[i].position = 0;
        //EncoderState[i].velocity = 0;   // NOT CURRENTLY USED
    }

    // configure direction and interrupt I/O pins:
    // - for input
    // - apply pullup resistors
    // - any-edge interrupt triggering
    // - enable interrupt

#ifdef ENC0_SIGNAL
    // set interrupt pins to input and apply pullup resistor
    cbi(ENC0_PHASEA_DDR, ENC0_PHASEA_PIN);
    sbi(ENC0_PHASEA_PORT, ENC0_PHASEA_PIN);
    // configure interrupts for any-edge triggering
    sbi(ENC0_ICR, ENC0_ISCX0);
    cbi(ENC0_ICR, ENC0_ISCX1);
    // enable interrupts
    sbi(IMSK, ENC0_INT);
    // ISMK is auto-defined in encoder.h
#endif
#ifdef ENC1_SIGNAL
    // set interrupt pins to input and apply pullup resistor
    cbi(ENC1_PHASEA_DDR, ENC1_PHASEA_PIN);
    sbi(ENC1_PHASEA_PORT, ENC1_PHASEA_PIN);
    // configure interrupts for any-edge triggering
    sbi(ENC1_ICR, ENC1_ISCX0);
    cbi(ENC1_ICR, ENC1_ISCX1);
    // enable interrupts
    sbi(IMSK, ENC1_INT);
    // ISMK is auto-defined in encoder.h
#endif
#ifdef ENC2_SIGNAL
    // set interrupt pins to input and apply pullup resistor
    cbi(ENC2_PHASEA_DDR, ENC2_PHASEA_PIN);
    sbi(ENC2_PHASEA_PORT, ENC2_PHASEA_PIN);
    // configure interrupts for any-edge triggering
    sbi(ENC2_ICR, ENC2_ISCX0);
    cbi(ENC2_ICR, ENC2_ISCX1);
    // enable interrupts
    sbi(IMSK, ENC2_INT);// ISMK is auto-defined in encoder.h
#endif
#ifdef ENC3_SIGNAL
    // set interrupt pins to input and apply pullup resistor
    cbi(ENC3_PHASEA_DDR, ENC3_PHASEA_PIN);
    sbi(ENC3_PHASEA_PORT, ENC3_PHASEA_PIN);
    // set encoder direction pin for input and apply pullup resistor
    cbi(ENC3_PHASEB_DDR, ENC3_PHASEB_PIN);
    sbi(ENC3_PHASEB_PORT, ENC3_PHASEB_PIN);
    // configure interrupts for any-edge triggering
    sbi(ENC3_ICR, ENC3_ISCX0);
    cbi(ENC3_ICR, ENC3_ISCX1);
    // enable interrupts
    sbi(IMSK, ENC3_INT);// ISMK is auto-defined in encoder.h
#endif

    // enable global interrupts
    sei();
}

// encoderGetPosition() reads the current position of the encoder
uint16_t encoderGetPosition(uint8_t encoderNum)
{
    // sanity check
    if (encoderNum < NUM_ENCODERS)
        return EncoderState[encoderNum].position;
    else
        return 0;
}

// encoderSetPosition() sets the current position of the encoder
void encoderSetPosition(uint8_t encoderNum, uint16_t position
)
{
    // sanity check
    if (encoderNum < NUM_ENCODERS)
        EncoderState[encoderNum].position = position;
    // else do nothing
}

#ifdef ENC0_SIGNAL

SIGNAL(ENC0_SIGNAL)
{
    /*********************************************/
    /* Modified by Taylor                        */
    /*********************************************/
    EncoderState[0].position++;
}
#endif

#ifdef ENC1_SIGNAL

SIGNAL(ENC1_SIGNAL)
{
    /*********************************************/
    /* Modified by Taylor                        */
    /*********************************************/
    EncoderState[1].position++;
}
#endif

#ifdef ENC2_SIGNAL

SIGNAL(ENC2_SIGNAL)
{
    // encoder has generated a pulse
    // check the relative phase of the input channels
    // and update position accordingly
    if( ((inb(ENC2_PHASEA_PORTIN) & (1<<ENC2_PHASEA_PIN)) == 0) ^
        ((inb(ENC2_PHASEB_PORTIN) & (1<<ENC2_PHASEB_PIN)) == 0) )
    {
        EncoderState[2].position++;
    }
    else
    {
        EncoderState[2].position--;
    }
}
#endif

#ifdef ENC3_SIGNAL

SIGNAL(ENC3_SIGNAL)
{
    // encoder has generated a pulse
    // check the relative phase of the input channels
    // and update position accordingly
    if( ((inb(ENC3_PHASEA_PORTIN) & (1<<ENC3_PHASEA_PIN)) == 0) ^
        ((inb(ENC3_PHASEB_PORTIN) & (1<<ENC3_PHASEB_PIN)) == 0) )
    {
        EncoderState[3].position++;
    }
    else
    {
        EncoderState[3].position--;
    }
}
#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

```
/*
 * This file is part of Caddy.
 *
 *  Caddy is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  Caddy is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
 */
//*****************************************************************************
//
// File Name  : 'encoder.h'
// Title    : Quadrature Encoder reader/driver
// Author   : Pascal Stang - Copyright (C) 2003-2004
// Created    : 2003.01.26
// Revised    : 2004.06.25
// Version    : 0.3
// Target MCU : Atmel AVR Series
// Editor Tabs  : 4
//
// Description  : This library allows easy interfacing of quadrature encoders
//    to the Atmel AVR-series processors.
//
//    Quadrature encoders have two digital outputs usually called PhaseA and
//  PhaseB.  When the encoder rotates, PhaseA and PhaseB produce square wave
//  pulses where each pulse represents a fraction of a turn of the encoder
//  shaft.  Encoders are rated for a certain number of pulses (or counts) per
//  complete revolution of the shaft.  Common counts/revolution specs are 50,
//  100,128,200,250,256,500,etc.  By counting the number of pulses output on
//  one of the phases starting from time0, you can calculate the total
//  rotational distance the encoder has traveled.
//
//  Often, however, we want current position not just total distance traveled.
//  For this it is necessary to know not only how far the encoder has traveled,
//  but also which direction it was going at each step of the way.  To do this
//  we need to use both outputs (or phases) of the quadrature encoder.
//
//  The pulses from PhaseA and PhaseB on quadrature encoders are always aligned
//  90 degrees out-of-phase (otherwise said: 1/4 wavelength apart).  This
//  special phase relationship lets us extract both the distance and direction
//  the encoder has rotated from the outputs.
//
//  To do this, consider Phase A to be the distance counter.  On each rising
//  edge of PhaseA we will count 1 "tic" of distance, but we need to know the
//  direction.  Look at the quadrature waveform plot below.  Notice that when
//  we travel forward in time (left->right), PhaseB is always low (logic 0) at
//  the rising edge of PhaseA.  When we travel backwards in time (right->left),
//  PhaseB is always high (logic 1) at the rising edge of PhaseA.  Note that
//  traveling forward or backwards in time is the same thing as rotating
//  forwards or bardwards. Thus, if PhaseA is our counter, PhaseB indicates
//  direction.
//
//  Here is an example waveform from a quadrature encoder:
/*
//                /---\   /---\   /---\   /---\   /---\   /---\
//  Phase A:      |   |   |   |   |   |   |   |   |   |   |   |
//             ---/   \---/   \---/   \---/   \---/   \---/   \-
//             -\   /---\   /---\   /---\   /---\   /---\   /---
//  Phase B:    |   |   |   |   |   |   |   |   |   |   |   |
//              \---/   \---/   \---/   \---/   \---/   \---/
//  Time:    <---------------------------------------------------->
//  Rotate FWD: >----------------------------------------------->
//  Rotate REV: <-----------------------------------------------<
*/
//  To keep track of the encoder position in software, we connect PhaseA to an
//  external processor interrupt line, and PhaseB to any I/O pin.  We set up
//  the external interrupt to trigger whenever PhaseA produces a rising edge.
//  When a rising edge is detected, our interrupt handler function is executed.
//  Inside the handler function, we quickly check the PhaseB line to see if it
//  is high or low.  If it is high, we increment the encoder's position
//  counter, otherwise we decrement it.  The encoder position counter can be
//  read at any time to find out the current position.
//
//
// NOTE: This code is currently below version 1.0, and therefore is considered
// to be lacking in some functionality or documentation, or may not be fully
// tested.  Nonetheless, you can expect most functions to work.
//
// This code is distributed under the GNU Public License
//    which can be found at http://www.gnu.org/licenses/gpl.txt
//
//*****************************************************************************

#ifndef ENCODER_H
#define ENCODER_H

#include "global.h"

// include encoder configuration file
#include "encoderconf.h"

#include <stdint.h>

// constants/macros/typdefs

// defines for processor compatibility
// chose proper Interrupt Mask (IMSK)
#ifdef EIMSK
  #define IMSK  EIMSK // for processors mega128, mega64
#else
  #define IMSK  GIMSK // for other processors 90s8515, mega163, etc
#endif


//   stores the position and other information from each encoder
typedef struct struct_EncoderState
{
  uint16_t position;
//  s32 velocity;       ///< velocity
} EncoderStateType;


// functions

//    Run this init routine once before using any other encoder function.
inline void encoderInit(void);

uint16_t encoderGetPosition(uint8_t encoderNum);

void encoderSetPosition(uint8_t encoderNum, uint16_t position
);

#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

```
/*
 * This file is part of Caddy.
 *
 *  Caddy is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  Caddy is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
 */
//*****************************************************************************
//
// File Name  : 'encoderconf.h'
// Title    : Quadrature Encoder driver configuration
// Author   : Pascal Stang - Copyright (C) 2003-2004
// Created    : 2003.01.26
// Revised    : 2004.06.25
// Version    : 0.2
// Target MCU : Atmel AVR Series
// Editor Tabs  : 4
//
// The default number of encoders supported is 2 because most AVR processors
// have two external interrupts.  To use more or fewer encoders, you must do
// four things:
//
//  1. Use a processor with at least as many external interrutps as number of
//    encoders you want to have.
//  2. Set NUM_ENCODERS to the number of encoders you will use.
//  3. Comment/Uncomment the proper ENCx_SIGNAL defines for your encoders
//    (the encoders must be used sequentially, 0 then 1 then 2 then 3)
//  4. Configure the various defines so that they match your processor and
//    specific hardware.  The notes below may help.
//
//
// -------------------- NOTES --------------------
// The external interrupt pins are mapped as follows on most AVR processors:
// (90s8515, mega161, mega163, mega323, mega16, mega32, etc)
//
// INT0 -> PD2 (PORTD, pin 2)
// INT1 -> PD3 (PORTD, pin 3)
//
// The external interrupt pins on the processors mega128 and mega64 are:
//
// INT0 -> PD0 (PORTD, pin 0)
// INT1 -> PD1 (PORTD, pin 1)
// INT2 -> PD2 (PORTD, pin 2)
// INT3 -> PD3 (PORTD, pin 3)
// INT4 -> PE4 (PORTE, pin 4)
// INT5 -> PE5 (PORTE, pin 5)
// INT6 -> PE6 (PORTE, pin 6)
// INT7 -> PE7 (PORTE, pin 7)
//
// This code is distributed under the GNU Public License
//    which can be found at http://www.gnu.org/licenses/gpl.txt
//
//*****************************************************************************

#ifndef ENCODERCONF_H
#define ENCODERCONF_H

// constants/macros/typdefs

// defines for processor compatibility
// quick compatiblity for mega128, mega64
//#ifndef MCUCR
//  #define MCUCR EICRA
//#endif

// Set the total number of encoders you wish to support
#define NUM_ENCODERS        2


// -------------------- Encoder 0 connections --------------------
// Phase A quadrature encoder output should connect to this interrupt line:
// *** NOTE: the choice of interrupt PORT, DDR, and PIN must match the external
// interrupt you are using on your processor.  Consult the External Interrupts
// section of your processor's datasheet for more information.

// Interrupt Configuration
#define ENC0_SIGNAL         SIG_INTERRUPT0  // Interrupt signal name
#define ENC0_INT          INT0  // matching INTx bit in GIMSK/EIMSK
#define ENC0_ICR          MCUCR // matching Int. Config Register (MCUCR,EICRA/
B)
#define ENC0_ISCX0          ISC00 // matching Interrupt Sense Config bit0
#define ENC0_ISCX1          ISC01 // matching Interrupt Sense Config bit1
// PhaseA Port/Pin Configuration
// *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
```

```
#define ENC0_PHASEA_PORT      PORTD // PhaseA port register
#define ENC0_PHASEA_DDR       DDRD  // PhaseA port direction register
#define ENC0_PHASEA_PORTIN      PIND  // PhaseA port input register
#define ENC0_PHASEA_PIN       PD2   // PhaseA port pin
// Phase B quadrature encoder output should connect to this direction line:
// *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
//#define ENC0_PHASEB_PORT      PORTC // PhaseB port register
//#define ENC0_PHASEB_DDR       DDRC  // PhaseB port direction register
//#define ENC0_PHASEB_PORTIN      PINC  // PhaseB port input register
//#define ENC0_PHASEB_PIN       PC0   // PhaseB port pin


// -------------------- Encoder 1 connections --------------------
// Phase A quadrature encoder output should connect to this interrupt line:
// *** NOTE: the choice of interrupt pin and port must match the external
// interrupt you are using on your processor.  Consult the External Interrupts
// section of your processor's datasheet for more information.

// Interrupt Configuration
#define ENC1_SIGNAL         SIG_INTERRUPT1  // Interrupt signal name
#define ENC1_INT          INT1  // matching INTx bit in GIMSK/EIMSK
#define ENC1_ICR          MCUCR // matching Int. Config Register (MCUCR,EICRA/B)
#define ENC1_ISCX0          ISC10 // matching Interrupt Sense Config bit0
#define ENC1_ISCX1          ISC11 // matching Interrupt Sense Config bit1
// PhaseA Port/Pin Configuration
// *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
#define ENC1_PHASEA_PORT      PORTD // PhaseA port register
#define ENC1_PHASEA_PORTIN      PIND  // PhaseA port input register
#define ENC1_PHASEA_DDR       DDRD  // PhaseA port direction register
#define ENC1_PHASEA_PIN       PD3   // PhaseA port pin
// Phase B quadrature encoder output should connect to this direction line:
// *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
//#define ENC1_PHASEB_PORT      PORTC // PhaseB port register
//#define ENC1_PHASEB_DDR       DDRC  // PhaseB port direction register
//#define ENC1_PHASEB_PORTIN      PINC  // PhaseB port input register
//#define ENC1_PHASEB_PIN       PC1   // PhaseB port pin


// -------------------- Encoder 2 connections --------------------
// Phase A quadrature encoder output should connect to this interrupt line:
// *** NOTE: the choice of interrupt pin and port must match the external
// interrupt you are using on your processor.  Consult the External Interrupts
// section of your processor's datasheet for more information.

// Interrupt Configuration
//#define ENC2_SIGNAL         SIG_INTERRUPT6  // Interrupt signal name
#define ENC2_INT          INT6  // matching INTx bit in GIMSK/EIMSK
#define ENC2_ICR          EICRB // matching Int. Config Register (MCUCR,EICRA/B)
#define ENC2_ISCX0          ISC60 // matching Interrupt Sense Config bit0
#define ENC2_ISCX1          ISC61 // matching Interrupt Sense Config bit1
// PhaseA Port/Pin Configuration
// *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
#define ENC2_PHASEA_PORT      PORTE // PhaseA port register
#define ENC2_PHASEA_PORTIN      PINE  // PhaseA port input register
#define ENC2_PHASEA_DDR       DDRE  // PhaseA port direction register
#define ENC2_PHASEA_PIN       PE6   // PhaseA port pin
// Phase B quadrature encoder output should connect to this direction line:
// *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
#define ENC2_PHASEB_PORT      PORTC // PhaseB port register
#define ENC2_PHASEB_DDR       DDRC  // PhaseB port direction register
#define ENC2_PHASEB_PORTIN      PINC  // PhaseB port input register
#define ENC2_PHASEB_PIN       PC2   // PhaseB port pin


// -------------------- Encoder 3 connections --------------------
// Phase A quadrature encoder output should connect to this interrupt line:
// *** NOTE: the choice of interrupt pin and port must match the external
// interrupt you are using on your processor.  Consult the External Interrupts
// section of your processor's datasheet for more information.

// Interrupt Configuration
//#define ENC3_SIGNAL         SIG_INTERRUPT7  // Interrupt signal name
#define ENC3_INT          INT7  // matching INTx bit in GIMSK/EIMSK
#define ENC3_ICR          EICRB // matching Int. Config Register (MCUCR,EICRA/B)
#define ENC3_ISCX0          ISC70 // matching Interrupt Sense Config bit0
#define ENC3_ISCX1          ISC71 // matching Interrupt Sense Config bit1
// PhaseA Port/Pin Configuration
// *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
```

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

#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

```
/*
 * This file is part of Caddy.
 *
 *  Caddy is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  Caddy is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
 */
#include "exercises.h"
#include "botCntrl.h"
#include "motorCntrl.h"
#include "buttons.h"
#include "helperFunctions.h"
#include "ourLCD.h"

// avr-libc
```

```
#include <stdbool.h>

void bbPickupTest(void)
{
    bool justTurned = true;
    bool firstRun = true;

    while (1)
    {
#if DEBUGGING
        lcdWriteStr("-96 brads, then ", 0, 0);
        lcdWriteStr("-32 brads       ", 1, 0);
#endif
        waitFor(RED_BUTTON);

        //hard left approach
        botNode = 23;
        botHeading = -128;
        pathListIndex = 0;
        pathList[0] = 23;
        pathList[1] = 3;
        pathList[2] = 24;
        pathList[3] = 3;
        pathList[4] = 23;
        justTurned = true;
        moveToJunction(1, justTurned);
        justTurned = positionBot();
        moveToJunction(1, justTurned);
        brake(BOTH);

#if DEBUGGING
        lcdWriteStr("96 brads, then  ", 0, 0);
        lcdWriteStr("32 brads        ", 1, 0);
#endif
        waitFor(RED_BUTTON);

        //Hard right approach
        botNode = 29;
        botHeading = 64;
        pathListIndex = 0;
        pathList[0] = 29;
        pathList[1] = 10;
        pathList[2] = 30;
        pathList[3] = 10;
        pathList[4] = 29;
        justTurned = true;
        moveToJunction(1, justTurned);
        justTurned = positionBot();
        moveToJunction(1, justTurned);
        brake(BOTH);

#if DEBUGGING
        lcdWriteStr("-32 brads, then ", 0, 0);
        lcdWriteStr("-96 brads       ", 1, 0);
#endif
        waitFor(RED_BUTTON);

        if (firstRun)
        {
            addToGoalList(BONUS_BALL_2);
        }

        //Soft Left approach
        botNode = 31;
        botHeading = -64;
        pathListIndex = 0;
        pathList[0] = 31;
        pathList[1] = 30;
        pathList[2] = 31;
        justTurned = true;
        moveToJunction(1, justTurned);
        justTurned = positionBot();
        moveToJunction(1, justTurned);
        brake(BOTH);

        printGoalList();
        waitFor(RED_BUTTON);
    }
}

void zigZagTest(void)
```

```
{
    bool justTurned = true;

#if DEBUGGING
    lcdWriteStr("botNode = 18    ", 0, 0);
    lcdWriteStr("botHeading = 0  ", 1, 0);
#endif
    waitFor(RED_BUTTON);

    botNode = 40;
    botHeading = 0;
    pathListIndex = 0;
    pathList[0] = 40;
    pathList[1] = 18;
    pathList[2] = 39;
    pathList[3] = 38;
    pathList[4] = 37;
    pathList[5] = 38;
    pathList[6] = 39;
    pathList[7] = 18;
    pathList[8] = 40;

    justTurned = true;
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);

#if DEBUGGING
    lcdWriteStr("Assume we do not", 0, 0);
    lcdWriteStr("seek ground ball", 1, 0);
#endif
    msDelay(3000);

    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);

#if DEBUGGING
    lcdWriteStr("botNode = 18    ", 0, 0);    //-------------------------------
    lcdWriteStr("botHeading = 0  ", 1, 0);
#endif
    waitFor(RED_BUTTON);

    botNode = 40;
    botHeading = 0;
    pathListIndex = 0;
    pathList[0] = 40;
    pathList[1] = 18;
    pathList[2] = 39;
    pathList[3] = 38;
    pathList[4] = 37;
    pathList[5] = 36;
    pathList[6] = 35;
    pathList[7] = 17;
    pathList[8] = 34;

    justTurned = true;
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);

#if DEBUGGING
    lcdWriteStr("Assume we seek  ", 0, 0);
    lcdWriteStr("ground ball.    ", 1, 0);
#endif
    msDelay(3000);

    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);

#if DEBUGGING
    lcdWriteStr("botNode = 17    ", 0, 0);   //------------------------------
    lcdWriteStr("botHeading = 64 ", 1, 0);
#endif
    waitFor(RED_BUTTON);

    botNode = 34;
    botHeading = 64;
    pathListIndex = 0;
    pathList[0] = 34;
    pathList[1] = 17;
    pathList[2] = 35;
    pathList[3] = 36;
    pathList[4] = 37;
    pathList[5] = 38;
    pathList[6] = 39;
    pathList[7] = 18;
    pathList[8] = 40;

    justTurned = true;
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);
}

void gbPickupTest(void)
{
    bool justTurned = true;

    while (1)
    {
#if DEBUGGING
        lcdWriteStr("One ground ball ", 0, 0);  //
------------------------------
        lcdWriteStr("(1 junc b4 ball)", 1, 0);
#endif
        waitFor(RED_BUTTON);

        botNode = 22;                     // set path
        botHeading = 0;
        pathListIndex = 0;
        pathList[0] = 22;
        pathList[1] = 9;
        pathList[2] = 29;
        pathList[3] = 11;
        pathList[4] = 12;
        pathList[5] = 33;
        pathList[6] = 13;
        pathList[7] = 34;

        initGoalList();                   // tell bot where balls are
        removeFromGoalList(BONUS_BALL_1);
        removeFromGoalList(BONUS_BALL_2);
        //removeFromGoalList(SENSOR_NODE);
        addToGoalList(11);

        justTurned = true;                // run test
        moveToJunction(1, justTurned);
        justTurned = positionBot();
        moveToJunction(1, false);
        justTurned = positionBot();
        moveToJunction(1, false);
        brake(BOTH);

#if DEBUGGING
        lcdWriteStr("Two ground balls", 0, 0);  //
------------------------------
        lcdWriteStr("(1 junc b4 ball)", 1, 0); //  try placing just one ball to
```

```
 test
#endif                                         //  safeguard before turn
        waitFor(RED_BUTTON);
        botNode = 22;                     // set path
        botHeading = 0;
        pathListIndex = 0;
        pathList[0] = 22;
        pathList[1] = 9;
        pathList[2] = 29;
        pathList[3] = 11;
        pathList[4] = 12;
        pathList[5] = 33;
        pathList[6] = 13;
        pathList[7] = 34;

        initGoalList();                   // tell bot where balls are
        removeFromGoalList(BONUS_BALL_1);
        removeFromGoalList(BONUS_BALL_2);
        removeFromGoalList(SENSOR_NODE);
        addToGoalList(11);
        addToGoalList(12);

        justTurned = true;                // run test
        moveToJunction(1, justTurned);
        justTurned = positionBot();
        moveToJunction(1, justTurned);
        justTurned = positionBot();
        moveToJunction(1, justTurned);
        brake(BOTH);

#if DEBUGGING
        lcdWriteStr("botNode = 20    ", 0, 0);  //
------------------------------
        lcdWriteStr("botHeading =-128", 1, 0); // ( -128 brad turn after pickup
 )
#endif                                         // make sure nestSequence is
 called
        waitFor(RED_BUTTON);
        botNode = 20;                     // set path
        botHeading = -128;
        pathListIndex = 0;
        pathList[0] = 20;
        pathList[1] = 41;
        pathList[2] = 5;
        pathList[3] = 41;
        pathList[4] = 42;

        initGoalList();                   // tell bot where balls are
        removeFromGoalList(BONUS_BALL_1);
        removeFromGoalList(BONUS_BALL_2);
        removeFromGoalList(SENSOR_NODE);
        addToGoalList(5);

        justTurned = true;                // run test
        moveToJunction(1, justTurned);
        justTurned = positionBot();
        moveToJunction(1, justTurned);
        justTurned = positionBot();
        nestSequence();
        brake(BOTH);
    }

}

void diagTest(void)
{
    bool justTurned = true;

#if DEBUGGING
    lcdWriteStr("botNode = 10    ", 0, 0);
    lcdWriteStr("botHeading = 64 ", 1, 0);
#endif
    waitFor(RED_BUTTON);

    botNode = 10;                     // set path
    botHeading = 64;
    pathListIndex = 0;
    pathList[0] = 10;
    pathList[1] = 30;
    pathList[2] = 31;
    pathList[3] = 15;
    pathList[4] = 14;
    pathList[5] = 34;
    pathList[6] = 13;
    pathList[7] = 33;
    pathList[8] = 13;
    pathList[9] = 34;
    pathList[10] = 14;
    pathList[11] = 15;
    pathList[12] = 31;
    pathList[13] = 30;
    pathList[14] = 10;
    pathList[15] = 29;

    initGoalList();                   // tell bot where balls are
    //addToGoalList(14);
    //addToGoalList(13);

    justTurned = true;                // run test
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();       // Should be -128 brad turn
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);

#if DEBUGGING
    lcdWriteStr("botNode = 16    ", 0, 0);   //------------------------------
    lcdWriteStr("botHeading = -64", 1, 0);
#endif
    waitFor(RED_BUTTON);

    botNode = 16;                     // set path
    botHeading = -64;
    pathListIndex = 0;
    pathList[0] = 16;
    pathList[1] = 32;
    pathList[2] = 31;
    pathList[3] = 15;
    pathList[4] = 14;
    pathList[5] = 34;
    pathList[6] = 17;
    pathList[7] = 35;
    pathList[8] = 17;
    pathList[9] = 34;
    pathList[10] = 14;
    pathList[11] = 15;
    pathList[12] = 31;
    pathList[13] = 32;
    pathList[14] = 16;
    pathList[15] = 40;

    initGoalList();                   // tell bot where balls are
    //addToGoalList(15);
    //addToGoalList(17);

    justTurned = true;                // run test
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();       // Should be -128 brad turn
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);
}

void node31Test(void)
{
    bool justTurned = true;

#if DEBUGGING
    lcdWriteStr("botNode = 7      ", 0, 0);   //-----------------------------
    lcdWriteStr("botHeading = 0   ", 1, 0);
#endif
    waitFor(RED_BUTTON);

    /* initial testing for ball on diagonal
     botNode = 7;                       // set path
     botHeading = 0;
     pathListIndex = 0;
     pathList[0] = 7;
     pathList[1] = 32;
     pathList[2] = 31;
     pathList[3] = 15;
     pathList[4] = 31;
     pathList[5] = 32;
     pathList[6] = 7;
     pathList[7] = 27;
     */
    //testing for diagonal, 2 junctions, and 180 at end of diag
    botNode = 7;                       // set path
    botHeading = 0;
    pathListIndex = 0;
    pathList[0] = 7;
    pathList[1] = 32;
    pathList[2] = 31;
    pathList[3] = 15;
    pathList[4] = 14;
    pathList[5] = 34;
    pathList[6] = 14;
    pathList[7] = 15;
    pathList[8] = 31;
    pathList[9] = 32;
    pathList[10] = 7;
    pathList[11] = 27;

    initGoalList();                     // tell bot where balls are
    //addToGoalList(15);

    /* Initial diag ball test
     justTurned = true;                 // run test
     moveToJunction(1, justTurned);
     justTurned = positionBot();
     moveToJunction(1, justTurned);
     justTurned = positionBot();
     moveToJunction(1, justTurned);     // -128 brads, try no ball
     justTurned = positionBot();
     moveToJunction(1, justTurned);
     justTurned = positionBot();
     moveToJunction(1, justTurned);
     brake(BOTH);
     */

    justTurned = true;                  // run test
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);     // -128 brads, try no ball
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);

#if DEBUGGING
    lcdWriteStr("botNode = 16     ", 0, 0);   //-----------------------------
    lcdWriteStr("botHeading = -64", 1, 0);
#endif
    waitFor(RED_BUTTON);

    botNode = 16;                       // set path
    botHeading = -64;
    pathListIndex = 0;
    pathList[0] = 16;
    pathList[1] = 32;
    pathList[2] = 31;
    pathList[3] = 30;
    pathList[4] = 31;
    pathList[5] = 32;
    pathList[6] = 16;
    pathList[7] = 40;

    initGoalList();                    // tell bot where balls are

    justTurned = true;                 // run test
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);

#if DEBUGGING
    lcdWriteStr("botNode = 16    ", 0, 0);   //------------------------------
    lcdWriteStr("botHeading = -64", 1, 0);
#endif
    waitFor(RED_BUTTON);

    botNode = 16;                      // set path
    botHeading = -64;
    pathListIndex = 0;
    pathList[0] = 16;
    pathList[1] = 32;
    pathList[2] = 7;
    pathList[3] = 27;
    pathList[4] = 7;
    pathList[5] = 32;
    pathList[6] = 31;
    pathList[7] = 30;
    pathList[8] = 31;
    pathList[9] = 32;
    pathList[10] = 7;
    pathList[11] = 27;
    pathList[12] = 7;
    pathList[13] = 32;
    pathList[14] = 16;
    pathList[15] = 40;

    initGoalList();                    // tell bot where balls are

    justTurned = true;                 // run test
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    justTurned = positionBot();
    moveToJunction(1, justTurned);
    brake(BOTH);

}
```

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

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

void bbPickupTest( void );
void gbPickupTest( void );
void zigZagTest( void );
void diagTest( void );
void node31Test( void );

#endif // #ifndef EXERCISES_H_
```

9.29 junctionCode.c File Reference

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

Include dependency graph for junctionCode.c:

Functions

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

Variables

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

9.29.1 Detailed Description

Definition in file junctionCode.c.

9.29.2 Variable Documentation

9.29.2.1 bool checkedList[ ]

Initial value:

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

Definition at line 31 of file junctionCode.c.

9.30 junctionCode.c

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

```
#include "nodeList.h"
#include "eeProm.h"
#include "motorCntrl.h"
#include "updatePath.h"
#include "helperFunctions.h"

// Global variables
// initialized in initBotGlobals
bool checkedList[] = { false, false, false, false, false, false,
                       false, false, false, false, false, false,
                       false, false, false, false, false, false,
                       false, false, false };
// initialized in initBotGlobals
uint8_t goalList[NUM_GOALS];

uint8_t goalListSize;
uint8_t numKnownGoals;

/*
 * Searches for ground balls, picks-up bonous balls, and computes best path.
 */
void junctionCode(void)
{
    bool foundBall = false;

    switch (botNode)
    {
    case (0):                      // old virtual windowing look for ball 7 and
 8
        foundBall = nodeCode0();
        break;
    case (21):                           // Suppress standard seek, should
 already
        break;                           // skip junction code at this node
    case (22):
        foundBall = standardBallSearch();  // standard seek and
        foundBall |= nodeCode22();         // tilt look for ball 9, 11, and 12
        break;
    case (31):                            // rotate bot for diagonal ball search
    case (34):                            // from any heading at node 31 or 34
        foundBall = diagNodeCode();
        break;
    case (37):                             // seek for top balls from nest
 sensor
        if (numKnownGoals < NUM_GOALS && (!checkedList[13] || !checkedList[17])
)
        {
            botNode = 35;
            moveStraight(10, 255);
            foundBall = diagNodeCode();
            moveStraight(-10, 255);
            botNode = 37;

            //pathList[pathListIndex--] = 36;
            //pathList[pathListIndex--] = 35;
        }
        break;
    default:
        foundBall = standardBallSearch();
        break;
    }

    if (foundBall)
    {
        // clear checked list, if last ball found
        if (numKnownGoals == NUM_GOALS)
        {
            uint8_t i;
            for (i = 0; i < NUM_BALL_NODES + 1; i++)
            {
                checkedList[i] = true;
            }
        }
        updatePath();
        printGoalList();
    }
}


/*
 * Returns true if a ball is found and the goal list is updated
```

```c
 */
bool standardBallSearch( void )
{
    NODE curNode;
    NODE nextNode;
    uint8_t nextNodeNum;
    int8_t lookDir = -1;                    // look left first
    int8_t hallHeading = 0;
    uint8_t ballDist = 0;
    uint8_t uncheckedBalls[3][2];
    uint8_t numUncheckedBalls = 0;
    bool foundBall = false;
    uint8_t i;

    bool stopped = false;
    inSeekPosition = false;

    // Check for balls in two directions (left/right)
    for (i = 0; i < 2; i++)
    {
        ballDist = 0;
        hallHeading = botHeading + lookDir * 64;
        nextNodeNum = botNode;
        numUncheckedBalls = 0;
        // Continue traversing nodes left of right until you hit the end
        while (nextNodeNum > 0)
        {
            /*
             #if DEBUGGING
             lcdWriteStr("N:   H:   ", 0, 0);
             lcdPrintHex(nextNodeNum, 0, 2);
             lcdPrintHex(hallHeading, 0, 7);
             waitFor(RED_BUTTON);
             #endif
             */
            getNode(nextNodeNum, &curNode);
            nextNodeNum = getNodeAtHeading(&curNode, hallHeading);
            if (nextNodeNum > 0)
            {
                getNode(nextNodeNum, &nextNode);
                // Keep track of how far away we are from the bot’s current node
                ballDist += getCostToNode(&curNode, nextNodeNum);
                if (isBallNode(nextNodeNum) && !checkedList[nextNodeNum])
                {
                    uncheckedBalls[numUncheckedBalls][BALL_DIST] = ballDist;
                    uncheckedBalls[numUncheckedBalls][BALL_NODE_NUM] =
                                    nextNodeNum;
                    checkedList[nextNodeNum] = true;
                    numUncheckedBalls++;
                }
            }
        }
        // Set pan, tilt, hi-res, etc...

        if (numUncheckedBalls > 0)
        {
            stopped = true;
            trackColorInit(lookDir);

            if (lookDir == -1)
            {
                foundBall |= cameraSeekLeft(uncheckedBalls, numUncheckedBalls);
            } else if (lookDir == 1)
            {
                foundBall |= cameraSeekRight(uncheckedBalls, numUncheckedBalls);
            }
        }

        /*
         #if DEBUGGING
         lcdWriteStr("Seek (  ) for:  ", 0, 0);
         lcdPrintHex(lookDir, 0, 6);
         lcdWriteStr("                ", 1, 0);
         for(j = 0; j < numUncheckedBalls; j++)
         {
         lcdPrintHex(uncheckedBalls[j][BALL_NODE_NUM], 1, 0);
         lcdPrintHex(uncheckedBalls[j][BALL_DIST], 1, 3);
         waitFor(RED_BUTTON);
         }
         waitFor(RED_BUTTON);
         #endif
         */

        lookDir *= -1;  // Look the other way the next time through
    }

    if (stopped)
    {
        moveStraight(0xb, 255);
        setServo(PAN, PAN_CENTER + panOffset);
        setServo(TILT, TILT_FORWARD);
        msDelay(600);
    }

    // Returns true if one or more balls are found
    return foundBall;
}


inline bool nodeCode0(void)
{
   bool foundBall = false;

   // two virtual windows

   return foundBall;
}


inline bool nodeCode22()
{
   bool foundBall = false;    // Return value
   uint8_t scanHeight = 4;
   uint8_t y = 254;
   uint8_t scanLimit = 1;
   uint8_t foundBallNum = 0;

   if (botHeading != 0)
      return false;

   trackColorInit(LOOK_UP);

   // scan from small ground distance to large ground distance
   while ( y - scanHeight > scanLimit )
   {
      y -= scanHeight;
      setVirtualWindow(1, y-scanHeight, 174, y);
      if ( seeBall() )
      {
         foundBall = true;

         // find ball number of ball at this x
         if( y > 148 )
            foundBallNum = 9;
         else if ( y > 50 )
            foundBallNum = 11;
         else
            foundBallNum = 12;

         addToGoalList( foundBallNum );

#if DEBUGGING
         labelColorStats();
         refreshColorStats();
         //msDelay(1000);
         //clearColorStats();
#endif

/*
#if DEBUGGING
         lcdWriteStr("Added:          ",0,0);
         lcdWriteStr("                ",1,0);
         lcdPrintHex(foundBallNum,1,0);
         waitFor(RED_BUTTON);
#endif
*/
         while ( seeBall() )
         {
            y -= scanHeight;
            setVirtualWindow(1, y-scanHeight, 174, y);
         }
      }
   }

   setServo(PAN, PAN_CENTER+panOffset);
   setServo(TILT, TILT_FORWARD);
   msDelay(300);

   return foundBall;
}


inline bool diagNodeCode(void)
{
   bool foundBall = false;

   if( botHeading == N_WEST && (!checkedList[13] || !checkedList[17]) )
   {
      tankTurn(255,tempTweak3);    // tank right
      botHeading += 41;
      foundBall = standardBallSearch();
      botHeading -= 41;
      tankTurn(255, -1*tempTweak3);      // tank left
   }
   else if( botHeading != S_EAST && (!checkedList[14] || !checkedList[15]) )
   {
      tankTurn(255, -1*tempTweak3);      // tank left
      botHeading -= 41;
      foundBall = standardBallSearch();
      botHeading += 41;
      tankTurn(255,tempTweak3);    // tank right
   }

   return foundBall;
}

inline bool nodeCode37( void )
{
   bool foundBall = false;

   // pass special values into cameraSeekLeft

   return foundBall;
}
```

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

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

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

#define BALL_DIST       0
#define BALL_NODE_NUM   1

// Global variables
extern bool checkedList[];
extern uint8_t goalList[];
extern uint8_t goalListSize;
extern uint8_t numKnownGoals;

void junctionCode(void);
bool standardBallSearch( void );
inline bool nodeCode0( void );
inline bool nodeCode22( void );
inline bool diagNodeCode(void);
inline bool nodeCode37( void );

#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

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

// avr-libc
#include <string.h>

inline bool isJunction(uint8_t nodeNum)
{
    return (nodeNum >= JUNCTION_MIN && nodeNum <= JUNCTION_MAX);
}

inline bool isBallNode(uint8_t nodeNum)
{
    return (nodeNum >= BALL_NODE_MIN && nodeNum <= BALL_NODE_MAX);
}

uint8_t getCostToNode(NODE *node, uint8_t nodeNum)
{
    uint8_t i;
    for (i = 0; i < node->numAdjNodes; i++)
    {
        if (node->adjNodes[i] == nodeNum)
        {
            return node->adjCosts[i];
        }
    }
    return 0;
}

uint8_t getNodeAtHeading(NODE *node, int8_t heading)
{
    uint8_t i;
    for (i = 0; i < node->numAdjNodes; i++)
    {
        if (node->adjHeadings[i] == heading)
        {
            return node->adjNodes[i];
        }
    }
    return 0;
}

void getNode(uint8_t nodeNum, NODE *node)
{
    if (nodeNum >= NUM_NODES)
    {
        node = NULL;
    }

    switch (nodeNum)
    {
    case 0:                             // START_NODE
        node->numAdjNodes = 1;
        node->   adjNodes[0] = 21;
        node->   adjCosts[0] = 9;
        node->adjHeadings[0] = -64;
        break;
    case 1:                             // First ball node
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 21;
        node->   adjNodes[1] = 22;
        node->   adjCosts[0] = 4;
        node->   adjCosts[1] = 4;
        node->adjHeadings[0] = -128;
        node->adjHeadings[1] = 0;
        break;
    case 2:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 22;
        node->   adjNodes[1] = 23;
        node->   adjCosts[0] = 2;
        node->   adjCosts[1] = 2;
        node->adjHeadings[0] = -64;
        node->adjHeadings[1] = 64;
        break;
    case 3:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 23;
        node->   adjNodes[1] = 24;
        node->   adjCosts[0] = 2;
        node->   adjCosts[1] = 2;
        node->adjHeadings[0] = 0;
        node->adjHeadings[1] = -128;
        break;
    case 4:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 24;
        node->   adjNodes[1] = 25;
        node->   adjCosts[0] = 3;
        node->   adjCosts[1] = 3;
        node->adjHeadings[0] = -64;
        node->adjHeadings[1] = 64;
        break;
    case 5:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 26;
        node->   adjNodes[1] = 41;
        node->   adjCosts[0] = 2;
        node->   adjCosts[1] = 2;
        node->adjHeadings[0] = -64;
        node->adjHeadings[1] = 64;
        break;
    case 6:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 27;
        node->   adjNodes[1] = 28;
        node->   adjCosts[0] = 2;
        node->   adjCosts[1] = 2;
        node->adjHeadings[0] = 64;
        node->adjHeadings[1] = -64;
        break;
    case 7:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 27;
        node->   adjNodes[1] = 32;
        node->   adjCosts[0] = 3;
        node->   adjCosts[1] = 3;
        node->adjHeadings[0] = -128;
        node->adjHeadings[1] = 0;
        break;
    case 8:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 28;
        node->   adjNodes[1] = 30;
        node->   adjCosts[0] = 3;
        node->   adjCosts[1] = 3;
        node->adjHeadings[0] = -128;
        node->adjHeadings[1] = 0;
        break;
    case 9:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 22;
        node->   adjNodes[1] = 29;
        node->   adjCosts[0] = 3;
        node->   adjCosts[1] = 3;
        node->adjHeadings[0] = -128;
        node->adjHeadings[1] = 0;
        break;
    case 10:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 29;
        node->   adjNodes[1] = 30;
        node->   adjCosts[0] = 3;
        node->   adjCosts[1] = 3;
        node->adjHeadings[0] = -64;
        node->adjHeadings[1] = 64;
        break;
    case 11:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 12;
        node->   adjNodes[1] = 29;
        node->   adjCosts[0] = 4;
        node->   adjCosts[1] = 2;
        node->adjHeadings[0] = 0;
        node->adjHeadings[1] = -128;
        break;
    case 12:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 11;
        node->   adjNodes[1] = 33;
        node->   adjCosts[0] = 4;
        node->   adjCosts[1] = 2;
        node->adjHeadings[0] = -128;
        node->adjHeadings[1] = 0;
        break;
    case 13:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 33;
        node->   adjNodes[1] = 34;
        node->   adjCosts[0] = 2;
        node->   adjCosts[1] = 1;
        node->adjHeadings[0] = -64;
        node->adjHeadings[1] = 64;
        break;
    case 14:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 15;
        node->   adjNodes[1] = 34;
        node->   adjCosts[0] = 3;
        node->   adjCosts[1] = 4;
        node->adjHeadings[0] = S_EAST;
        node->adjHeadings[1] = N_WEST;
        break;
    case 15:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 14;
        node->   adjNodes[1] = 31;
        node->   adjCosts[0] = 3;
        node->   adjCosts[1] = 4;
        node->adjHeadings[0] = N_WEST;
        node->adjHeadings[1] = S_EAST;
        break;
    case 16:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 32;
        node->   adjNodes[1] = 40;
        node->   adjCosts[0] = 2;
        node->   adjCosts[1] = 2;
        node->adjHeadings[0] = -64;
        node->adjHeadings[1] = 64;
        break;
    case 17:
        node->numAdjNodes = 2;
        node->   adjNodes[0] = 34;
        node->   adjNodes[1] = 35;
        node->   adjCosts[0] = 3;
        node->   adjCosts[1] = 3;
        node->adjHeadings[0] = -64;
        node->adjHeadings[1] = 64;
```

```
            break;
        case 18:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 39;
            node->   adjNodes[1] = 40;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = 0;
            node->adjHeadings[1] = -128;
            break;
        case 19:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 20;
            node->   adjNodes[1] = 40;
            node->   adjCosts[0] = 4;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = -128;
            node->adjHeadings[1] = 0;
            break;
        case 20:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 19;
            node->   adjNodes[1] = 41;
            node->   adjCosts[0] = 4;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = 0;
            node->adjHeadings[1] = -128;
            break;
        case 21:                             // First Junction Node
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 0;
            node->   adjNodes[1] = 1;
            node->   adjCosts[0] = 9;
            node->   adjCosts[1] = 4;
            node->adjHeadings[0] = 64;
            node->adjHeadings[1] = 0;
            break;
        case 22:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 1;
            node->   adjNodes[1] = 2;
            node->   adjNodes[2] = 9;
            node->   adjCosts[0] = 4;
            node->   adjCosts[1] = 2;
            node->   adjCosts[2] = 3;
            node->adjHeadings[0] = -128;
            node->adjHeadings[1] = 64;
            node->adjHeadings[2] = 0;
            break;
        case 23:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 2;
            node->   adjNodes[1] = 3;
            node->   adjNodes[2] = 28;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->   adjCosts[2] = 2;
            node->adjHeadings[0] = -64;
            node->adjHeadings[1] = -128;
            node->adjHeadings[2] = 64;
            break;
        case 24:                             // BONUS_BALL_1
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 3;
            node->   adjNodes[1] = 4;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 3;
            node->adjHeadings[0] = 0;
            node->adjHeadings[1] = 64;
            break;
        case 25:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 4;
            node->   adjNodes[1] = 26;
            node->   adjCosts[0] = 3;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = -64;
            node->adjHeadings[1] = 0;
            break;
        case 26:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 5;
            node->   adjNodes[1] = 25;
            node->   adjNodes[2] = 27;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->   adjCosts[2] = 2;
            node->adjHeadings[0] = 64;
            node->adjHeadings[1] = -128;
            node->adjHeadings[2] = 0;
            break;
        case 27:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 6;
            node->   adjNodes[1] = 7;
            node->   adjNodes[2] = 26;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 3;
            node->   adjCosts[2] = 2;
            node->adjHeadings[0] = -64;
            node->adjHeadings[1] = 0;
            node->adjHeadings[2] = -128;
            break;
        case 28:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 6;
            node->   adjNodes[1] = 8;
            node->   adjNodes[2] = 23;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 3;
            node->   adjCosts[2] = 2;
            node->adjHeadings[0] = 64;
            node->adjHeadings[1] = 0;
            node->adjHeadings[2] = -64;
            break;
        case 29:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 9;
            node->   adjNodes[1] = 10;
            node->   adjNodes[2] = 11;
            node->   adjCosts[0] = 3;
            node->   adjCosts[1] = 3;
            node->   adjCosts[2] = 2;
            node->adjHeadings[0] = -128;
            node->adjHeadings[1] = 64;
            node->adjHeadings[2] = 0;
            break;
        case 30:                                  // BONUS_BALL_2
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 8;
            node->   adjNodes[1] = 10;
            node->   adjNodes[2] = 31;
            node->   adjCosts[0] = 3;
            node->   adjCosts[1] = 3;
            node->   adjCosts[2] = 3;
            node->adjHeadings[0] = -128;
            node->adjHeadings[1] = -64;
            node->adjHeadings[2] = 64;
            break;
        case 31:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 15;
            node->   adjNodes[1] = 30;
            node->   adjNodes[2] = 32;
            node->   adjCosts[0] = 4;
            node->   adjCosts[1] = 3;
            node->   adjCosts[2] = 1;
            node->adjHeadings[0] = N_WEST;
            node->adjHeadings[1] = -64;
            node->adjHeadings[2] = 64;
            break;
        case 32:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 7;
            node->   adjNodes[1] = 16;
            node->   adjNodes[2] = 31;
            node->   adjCosts[0] = 3;
            node->   adjCosts[1] = 2;
            node->   adjCosts[2] = 1;
            node->adjHeadings[0] = -128;
            node->adjHeadings[1] = 64;
            node->adjHeadings[2] = -64;
            break;
        case 33:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 12;
            node->   adjNodes[1] = 13;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = -128;
            node->adjHeadings[1] = 64;
            break;
        case 34:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 13;
            node->   adjNodes[1] = 14;
            node->   adjNodes[2] = 17;
            node->   adjCosts[0] = 1;
            node->   adjCosts[1] = 4;
            node->   adjCosts[2] = 3;
            node->adjHeadings[0] = -64;
            node->adjHeadings[1] = S_EAST;
            node->adjHeadings[2] = 64;
            break;
        case 35:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 17;
            node->   adjNodes[1] = 36;
            node->   adjCosts[0] = 3;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = -64;
            node->adjHeadings[1] = -128;
            break;
        case 36:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 35;
            node->   adjNodes[1] = 37;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = 0;
            node->adjHeadings[1] = 64;
            break;
        case 37:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 36;
            node->   adjNodes[1] = 38;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = -64;
            node->adjHeadings[1] = -128;
            break;
        case 38:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 37;
            node->   adjNodes[1] = 39;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = 0;
            node->adjHeadings[1] = 64;
            break;
        case 39:
            node->numAdjNodes = 2;
            node->   adjNodes[0] = 18;
            node->   adjNodes[1] = 38;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->adjHeadings[0] = -128;
            node->adjHeadings[1] = -64;
            break;
        case 40:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 16;
            node->   adjNodes[1] = 18;
            node->   adjNodes[2] = 19;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->   adjCosts[2] = 2;
            node->adjHeadings[0] = -64;
            node->adjHeadings[1] = 0;
            node->adjHeadings[2] = -128;
            break;
        case 41:
            node->numAdjNodes = 3;
            node->   adjNodes[0] = 5;
            node->   adjNodes[1] = 20;
            node->   adjNodes[2] = 42;
            node->   adjCosts[0] = 2;
            node->   adjCosts[1] = 2;
            node->   adjCosts[2] = 5;
            node->adjHeadings[0] = -64;
            node->adjHeadings[1] = 0;
            node->adjHeadings[2] = -128;
            break;
        case 42:                         // STOP_NODE
            node->numAdjNodes = 1;
            node->   adjNodes[0] = 41;
            node->   adjCosts[0] = 5;
            node->adjHeadings[0] = 0;
            break;
        }
    }
```

9.35 nodeList.h File Reference

Course defined by a connected grid of nodes.

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

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

Data Structures

- struct nodeStruct

Macros

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

Typedefs

- typedef struct nodeStruct **NODE**

Functions

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

9.35.1 Detailed Description

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

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

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

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

Definition in file nodeList.h.

9.36 nodeList.h

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

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

#define NUM_NODES       43        // number of nodes in arena

#define BALL_NODE_MIN   1         // ball node number range
#define BALL_NODE_MAX   20
#define JUNCTION_MIN    21        // junction node number range
#define JUNCTION_MAX    41

#define NUM_BALL_NODES  (BALL_NODE_MAX - BALL_NODE_MAX + 1)


```

```
#define MAX_ADJ_NODES   3         // max. nodes that can be adjacent to one
 node
#define N_WEST          -41       // direction of north west in binary radians
 (brads)
#define S_EAST          87        // direction of south east in binary radians
 (brads)

#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

typedef struct nodeStruct
{
    uint8_t numAdjNodes;                  // number of nodes adjacent to this
 node
    uint8_t adjNodes[MAX_ADJ_NODES];      // node numbers of adjacent nodes
    uint8_t adjCosts[MAX_ADJ_NODES];      // distances to adjacent nodes (6
 inches increments)
    int8_t adjHeadings[MAX_ADJ_NODES];   // directions towards adjacent nodes
 (brads)
} NODE;


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


#endif // #ifndef NODELIST_H_
```

9.37 perms.h File Reference

Iterative (non-recursive!) permutation generator.

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

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

Functions

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

  Reorder an array of values to the next higher permutation.

9.37.1 Detailed Description

Iterative (non-recursive!) permutation generator.

Definition in file perms.h.

9.37.2 Function Documentation

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

Reorder an array of values to the next higher permutation.

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

Typical usage example:

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

Remarks

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

Returns

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

Definition at line 22 of file perms.c.

9.38 perms.h

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

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

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

#endif // #ifndef PERMS_H_
```

9.39 tetherUI.h File Reference

Simple user interface to change parameters without reprogramming.

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

Macros

- #define NAV_LCD_MODE 0
- #define LINE_LCD_MODE 1

Functions

- void runTetherUI (void)

  Allow tweaking via tether remote until red button pressed.

9.39.1 Detailed Description

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

Definition in file tetherUI.h.

9.39.2 Macro Definition Documentation

9.39.2.1 #define LINE_LCD_MODE 1

Special LCD display mode for debugging line tracking

Definition at line 36 of file tetherUI.h.

9.39.2.2 #define NAV_LCD_MODE 0

Default LCD display mode

Definition at line 31 of file tetherUI.h.

9.40 tetherUI.h

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

9.41 trackColor.c File Reference

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

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

Include dependency graph for trackColor.c:

Macros

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

Functions

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

Variables

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

9.41.1 Detailed Description

Definition in file trackColor.c.

9.42 trackColor.c

```
/*
 * This file is part of Caddy.
 *
 *  Caddy is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  Caddy is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
 */
#include "trackColor.h"
#include "trackLine.h"
#include "camera.h"
#include "servos.h"
#include "junctionCode.h"
#include "motorCntrl.h"
#include "eeProm.h"
#include "helperFunctions.h"

// AVRLIB
#include "rprintf.h"

// avr-libc
#include <stdint.h>
#include <stdbool.h>

// Track the RED ball on black/white background
#define BALL_RMIN    150
#define BALL_RMAX    240
#define BALL_GMIN    16
#define BALL_GMAX    60
#define BALL_BMIN    16
#define BALL_BMAX    50

// Global variables
volatile bool colorStatsProcessed;
bool inSeekPosition;

static uint8_t distToPix( uint8_t distance );

void trackColorInit(int8_t dir)
{
    if (!inSeekPosition)
    {
        brake(BOTH);
        msDelay(200);
        moveStraight(-1 * 0xb, 255);
        inSeekPosition = true;
    }

    // Set pan (center) and tilt
    switch (dir)
    {
    case LOOK_LEFT:
        setServo(PAN, PAN_CENTER + panOffset + PAN_SEEK_OFFSET);
        setServo(TILT, TILT_VERT + tiltOffset);
        break;
    case LOOK_RIGHT:
        setServo(PAN, PAN_CENTER + panOffset - PAN_SEEK_OFFSET);
        setServo(TILT, TILT_VERT + tiltOffset);
        break;
    case LOOK_UP:
        setServo(PAN, PAN_CENTER + panOffset);
        setServo(TILT, TILT_LOOKUP);
        break;
    default:
        break;
    }
    msDelay(500);

    hiResMode();
    rprintf("DS 1 1\r");
    rprintf("LM 0 0\r");

    // Change to poll mode so only one packet is sent
    rprintf("PM 1\r");
}

/*
 * Returns Y1 (top of ball) if camera sees a ball, zero otherwise
 */
uint8_t getBallY( void )
{
    rprintf("lm 0 0\r");

    // Mask everything but the 'My' value
    //rprintf("OM 0 2\r");                             //<- NO MASKING?

    // Change to poll mode so only one packet is sent
    rprintf("PM 1\r");

    // Track red
    rprintf("TC %d %d %d %d %d %d\r",
                    BALL_RMIN, BALL_RMAX, BALL_GMIN, BALL_GMAX, BALL_BMIN,
BALL_BMAX);

    colorStatsProcessed = true;
    while (colorStatsProcessed) ;

    return (lineStats[0][Y1_NDX]);
}


bool seeBall( void )
{
   // Track red
   rprintf("TC %d %d %d %d %d %d\r",
           BALL_RMIN, BALL_RMAX, BALL_GMIN, BALL_GMAX, BALL_BMIN, BALL_BMAX);
   colorStatsProcessed = true;
   while (colorStatsProcessed) ;

   return lineStats[0][Y1_NDX] > 0;
}


/*
 * Just does left seeks
 *    PRE - the longest check is the last element of the uncheckedBalls array
 *
 *    uncheckedBalls - ball node numbers and ground distances away from bot
 */
bool cameraSeekLeft( uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls )
{
    bool foundBall = false;    // Return value
    uint8_t scanHeight = 4;
    uint8_t x = 174;
    //uint8_t ballDist[3];
    //uint8_t ballCount = 0;
    uint8_t scanLimit = distToPix(
                    uncheckedBalls[numUncheckedBalls - 1][BALL_DIST] + 1);

    // get pixel ranges for unchecked balls passed in
    uint8_t i = 0;
    uint8_t maxBallX[3];
    while (i + 1 < numUncheckedBalls)
    {
        maxBallX[i] = (distToPix(uncheckedBalls[i][BALL_DIST]) +
                       distToPix(uncheckedBalls[i + 1][BALL_DIST])) / 2;
        i++;
    }
    maxBallX[i] = scanLimit;

    /*
     #if DEBUGGING
     lcdWriteStr("maxBallX =      ",0,0);
     lcdWriteStr("                ",1,0);
     for( i = 0; i < numUncheckedBalls; i++ )
     {
     lcdPrintHex(maxBallX[i],1,3*i);
     }
     waitFor(RED_BUTTON);
     #endif
     */

    // scan from small ground distance to large ground distance
    while (x - scanHeight > scanLimit)
    {
        x -= scanHeight;
        setVirtualWindow(x - scanHeight, 1, x, 254);
        if (seeBall())
        {
            foundBall = true;
            //ballDist[ballCount++] = xToDist(x);

            // find ball number of ball at this x
            i = 0;
            while (maxBallX[i] > x)
            {
                i++;
            }
            addToGoalList(uncheckedBalls[i][BALL_NODE_NUM]);
```

```

#if DEBUGGING
            labelColorStats();
            refreshColorStats();
#endif

            /*
             #if DEBUGGING
             lcdWriteStr("Added:          ",0,0);
             lcdWriteStr("                ",1,0);
             lcdPrintHex(uncheckedBalls[i][BALL_NODE_NUM],1,0);
             waitFor(RED_BUTTON);
             #endif
             */

            while (seeBall())
            {
                x -= scanHeight;
                setVirtualWindow(x - scanHeight, 1, x, 254);
            }
        }
    }

    return foundBall;
}

// returns pixel equivalent of 'distance'
static uint8_t distToPix( uint8_t distance )
{
    switch (distance)
    {
    case 0:
    case 1:
        return 174;
    case 2:
        return 0x8d;
    case 3:
        return 0x61;
    case 4:
        return 0x48;
    case 5:
        return 0x36;
    case 6:
        return 0x2b;
    case 7:
        return 0x22;
    case 8:
        return 0x1d;
    case 9:
        return 0x18;
    case 10:
        return 0x14;
    case 11:
        return 0x11;
    case 12:
        return 0x0e;
    default:
        return 0x0;
    }
}

/*
 * Just does right seeks
 *    PRE - the longest check is the last element of the uncheckedBalls array
 *
 *    uncheckedBalls - ball node numbers and ground distances away from bot
 */
bool cameraSeekRight(uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls)
{
    bool foundBall = false;    // Return value
    uint8_t scanHeight = 4;
    uint8_t x = 0;
    uint8_t scanLimit = 174 - distToPix(
                        uncheckedBalls[numUncheckedBalls - 1][BALL_DIST] + 1);

    // get pixel ranges for unchecked balls passed in
    uint8_t i = 0;
    uint8_t maxBallX[3];
    while (i + 1 < numUncheckedBalls)
    {
        maxBallX[i] = ((174 - distToPix(uncheckedBalls[i][BALL_DIST])) +
                        (174 - distToPix(uncheckedBalls[i + 1][BALL_DIST])))
                        / 2;
        i++;
    }
    maxBallX[i] = scanLimit;

/*
#if DEBUGGING
    lcdWriteStr("maxBallX =      ", 0, 0);
    lcdWriteStr("                ", 1, 0);
    for (i = 0; i < numUncheckedBalls; i++)
    {
        lcdPrintHex(maxBallX[i], 1, 3 * i);
    }
    waitFor(RED_BUTTON);
#endif
*/

    // scan from small ground distance to large ground distance
    while (x + scanHeight < scanLimit)
    {
        x += scanHeight;
        setVirtualWindow(x, 1, x + scanHeight, 254);
        if (seeBall())
        {
            foundBall = true;
            //ballDist[ballCount++] = xToDist(x);

            // find ball number of ball at this x
            i = 0;
            while (maxBallX[i] < x)
            {
                i++;
            }
            addToGoalList(uncheckedBalls[i][BALL_NODE_NUM]);

#if DEBUGGING
            labelColorStats();
            refreshColorStats();
#endif

/*
#if DEBUGGING
            msDelay(1000);
            clearColorStats();
            lcdWriteStr("Added:          ", 0, 0);
            lcdWriteStr("                ", 1, 0);
            lcdPrintHex(uncheckedBalls[i][BALL_NODE_NUM], 1, 0);
            waitFor(RED_BUTTON);
#endif
*/

            while (seeBall())
            {
                x += scanHeight;
                setVirtualWindow(x, 1, x + scanHeight, 254);
            }
        }
    }

   return foundBall;
}

```

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

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

// avr-libc
#include <stdint.h>
#include <stdbool.h>

#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

// Global variables
extern volatile bool colorStatsProcessed;
extern bool inSeekPosition;

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 );

#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

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

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

#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    // must correspond to bits in LINE_STAT_MASK

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);

extern int8_t junctionY;

```

```
// Global variables
extern volatile uint8_t lineStats[LINE_STATS_ROWS][LINE_STATS_COLS];
extern volatile bool lineStatsProcessed;

#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