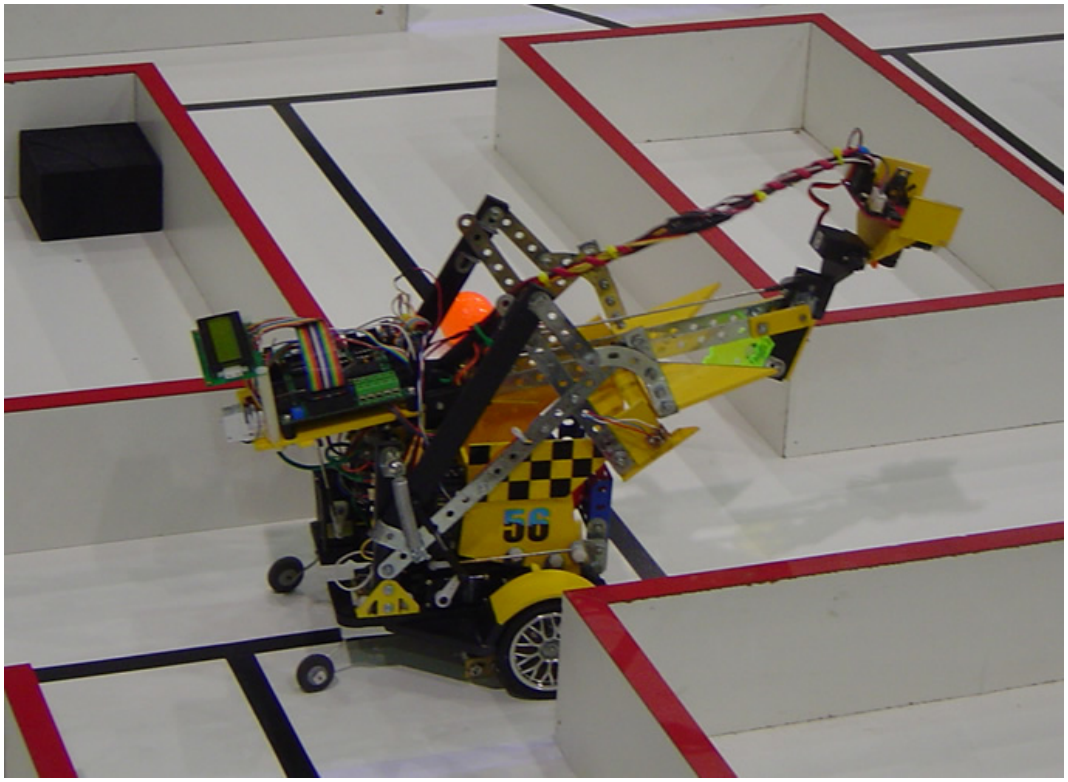
Caddy

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 that was entered into the 2005 Roborodentia competition. Roborodentia is an annual autonomous robotics competition held during Cal Poly's Open House by the Cal Poly Chapter of 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. A newly added aspect for the 2005 competition included two bonus balls that were placed on a platform behind the wall in two predetermined corners of the maze. These platforms raised the bonus balls such that the top of the golf ball was flush with the top of the wall.

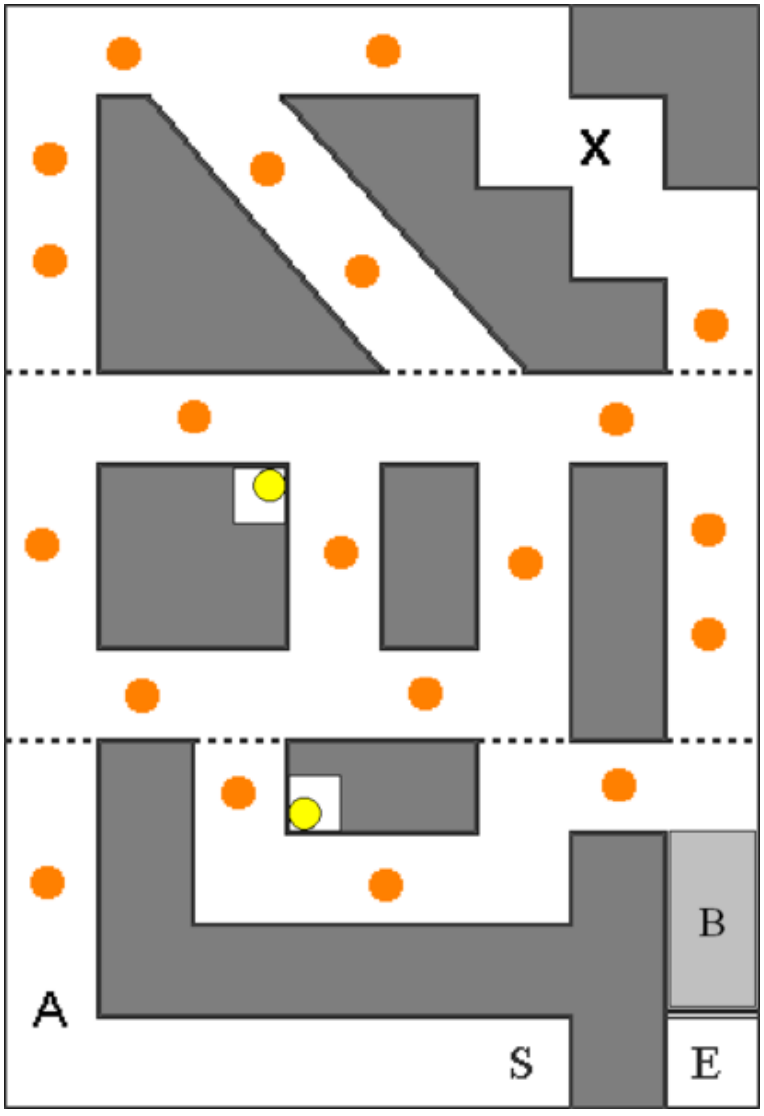


Figure 1: Arena map showing the two fixed bonus ball locations and the potential locations of the randomly placed ground balls

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 the “nest” - Area B (1 point per ball)
9	Placing each ground ball in the “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 requirements that needed to be met in order to be successful:

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

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

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

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

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

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

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

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

3 Design

3.1 Collaborative Team Process

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

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

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

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

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

3.2 System Architecture

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

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

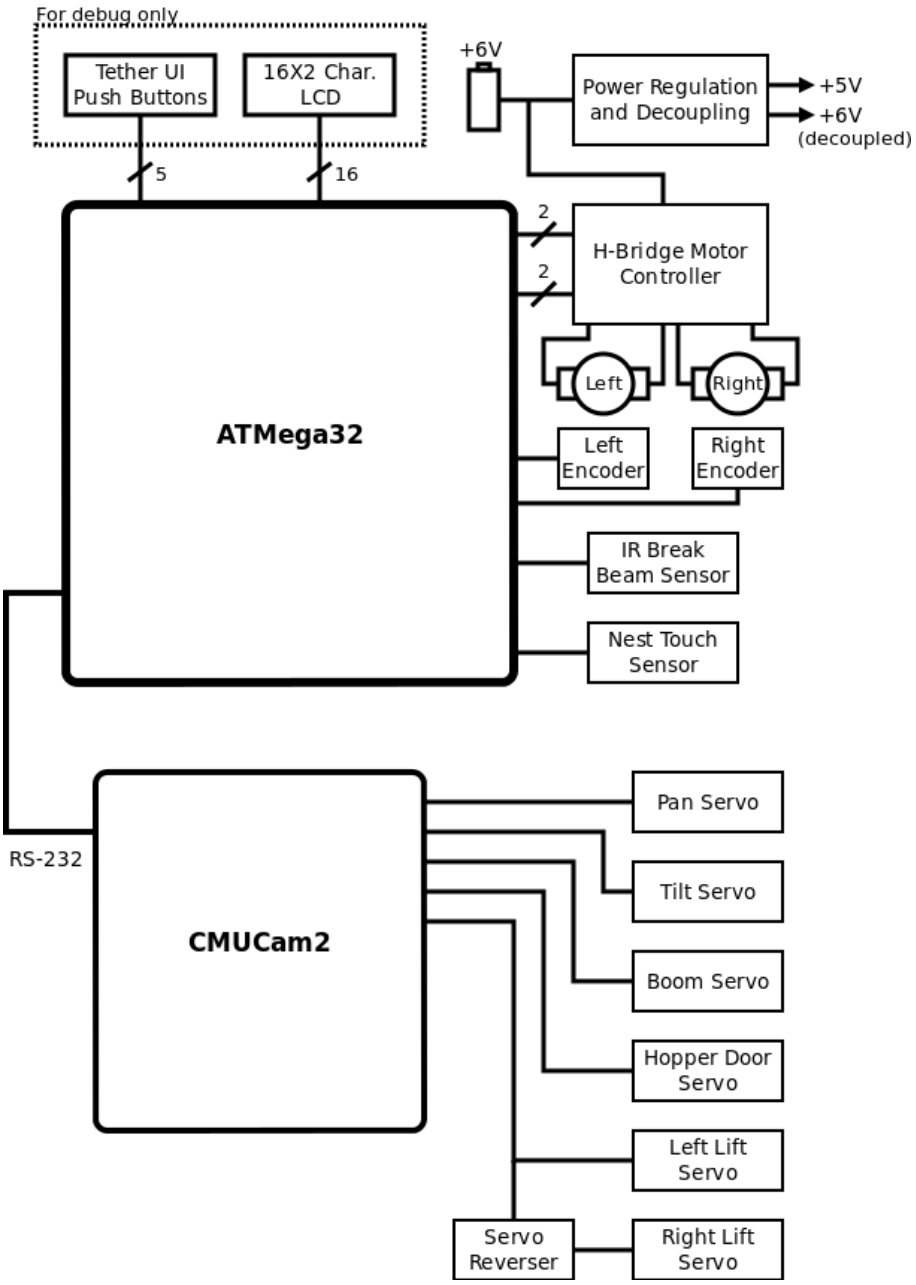


Figure 2: Electronics block diagram

3.3 Electrical Design

3.3.1 Power Regulation and Motor Controller

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

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

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

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

- Lower output noise - We wanted to take every precaution we could to ensure the EMF voltage generated by the motors did not affect the electronics.
- Simpler to implement - Switching regulators typically require more passive components to filter the inherently higher noise they generate.
- Cheaper

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

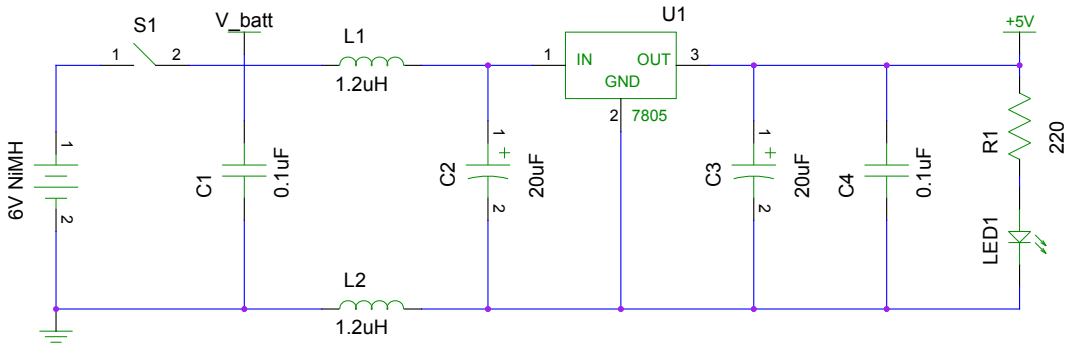


Figure 3: Power regulation circuit

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

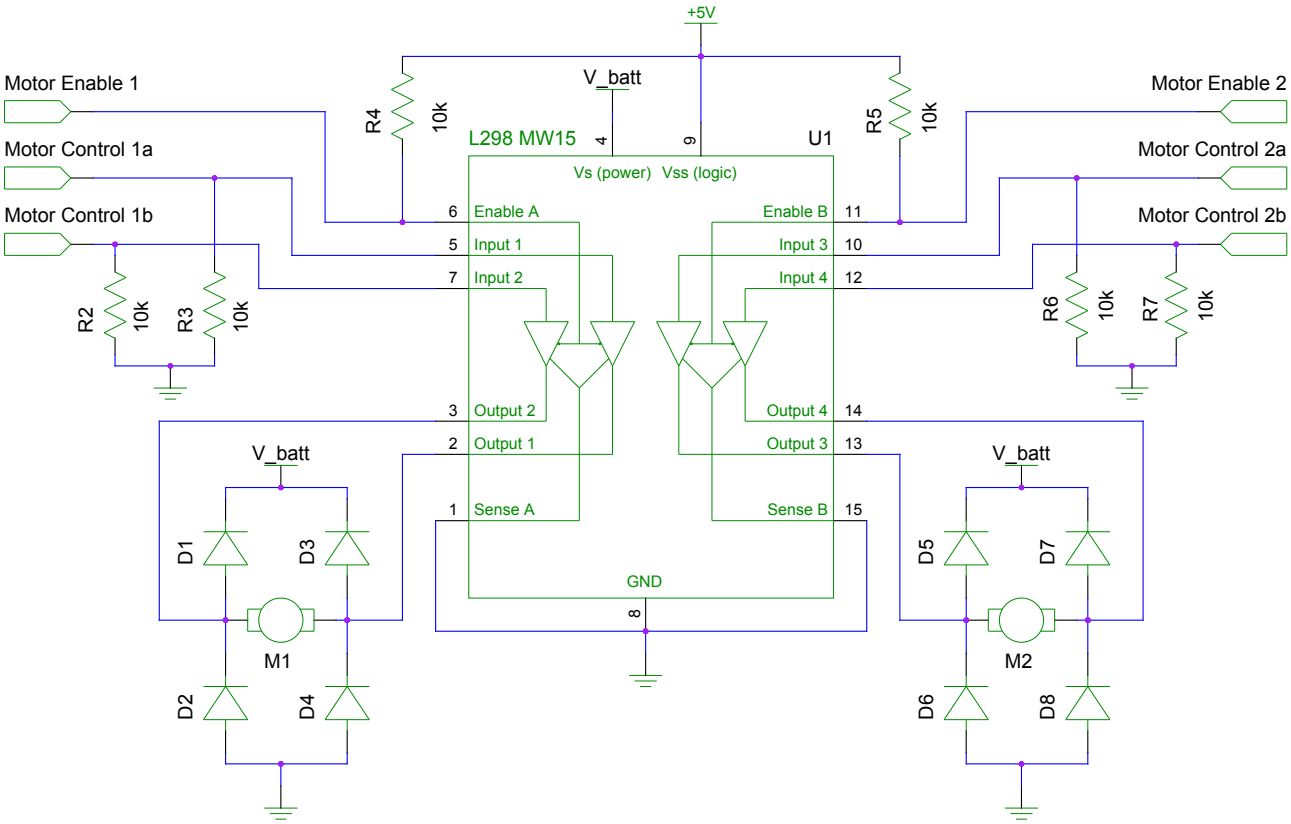


Figure 4: Motor control circuit

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

3.3.2 Wheel Encoders

Figure 5: Reflective IR sensors mounted on a bracket inside the left drive wheel

The maneuvers needed at junctions and for the bonus ball pick up sequences needed to be accurate and repeatable. To achieve this we use a simple differential drive system with encoders on each drive wheel. The encoders were made up of reflective IR sensors pointed at black and white patterned disks pasted to the inside edge of each drive wheel. [2]

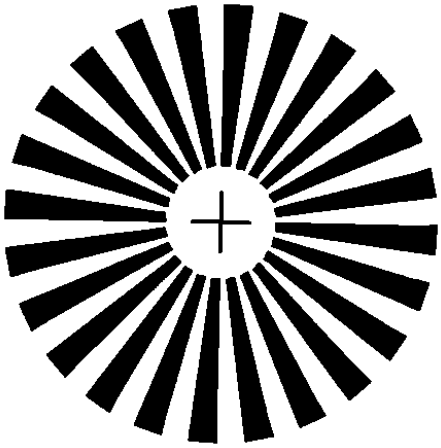


Figure 6: 48 segment pattern for reflective IR wheel encoders

The reflective IR sensors emit infrared light and measure the amount of infrared light that is reflected back with a photodiode. Since the white segments of the pattern reflect more light than the black segments, rotations of the wheel can be detected by thresholding the measured IR signal level. To guard against "chattering" when signals close to the threshold are applied a dual-threshold Schmitt trigger is needed. The precision of this wheel encoder system is equal to about one segment arc angle — 7.5° for a 48 segment pattern.

The P5587 reflective IR sensor package chosen for Caddy was designed for exactly this type of high contrast edge detection and so came with built in Schmitt triggers and a logic-level output that could be easily read by GPIO input port on the microcontroller.

3.3.3 IR Break Beam

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

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

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

3.3.4 Servo Reverser

The mechanical design of Caddy required 6 servos:

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

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

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

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

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

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

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

3.4 Software Architecture and Algorithms

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 7: EMBMega32 development board from ERE CO.,LTD

3.4.2 PID Line Tracking

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

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

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

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

By substitution, we get:

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

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

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

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

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

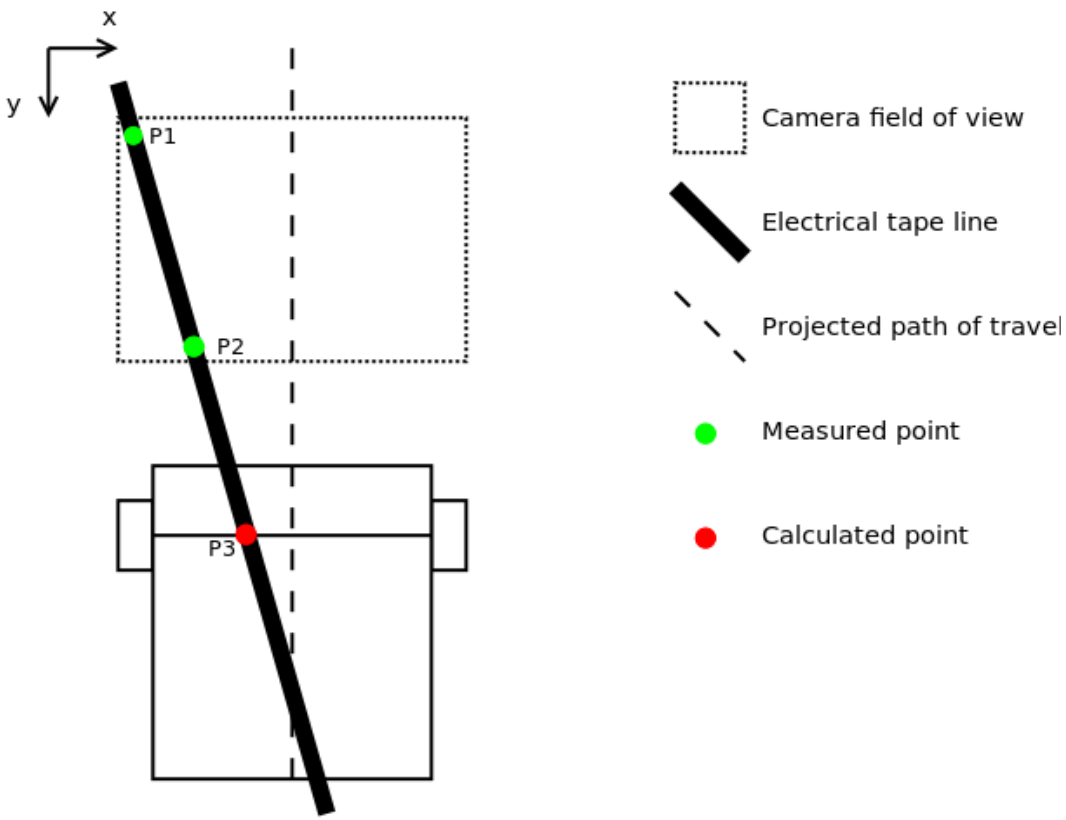


Figure 8: Diagram of line tracking geometry (not to scale)

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

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

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

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

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

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

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

3.4.3 Maneuvering

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

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

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

3.4.4 Localization

While the locations of the ground balls were not known a priori, the map of the course was. We defined the course as a connected undirected graph with 42 nodes (vertices) as shown below. A node was placed at:

- The start of the course
- The end of the course
- Every junction
- Every potential ground ball location

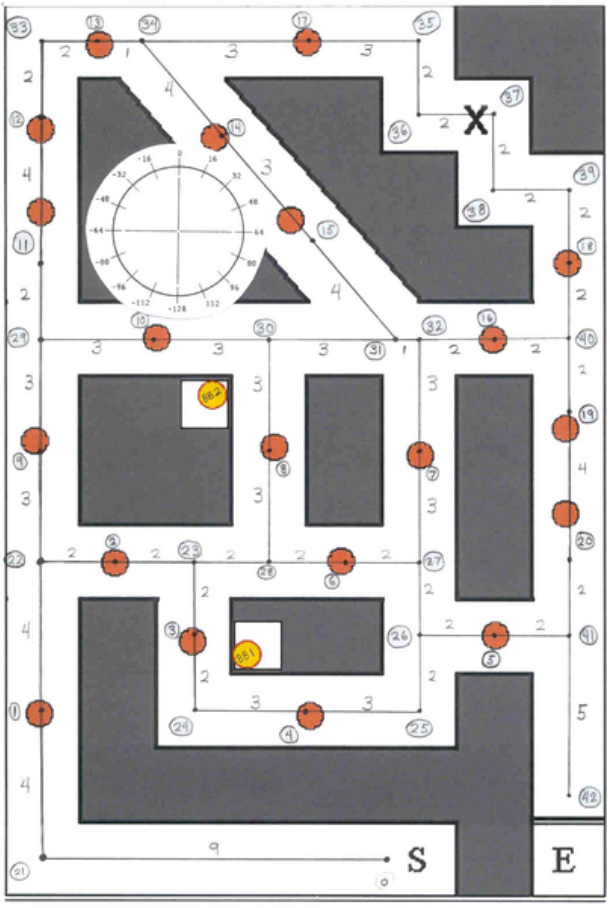


Figure 9: Caddy's connected graph representation of the arena

The software implementation of this graph was done using a C struct:

```
typedef struct nodeStruct
{
    uint8_t numAdjNodes;
    uint8_t adjNodes[MAX_ADJ_NODES];
    uint8_t adjCosts[MAX_ADJ_NODES];
    int8_t adjHeadings[MAX_ADJ_NODES];
} NODE;
```

Each node defined the adjacent node numbers, directions, and distances to each adjacent node. Headings were defined in units of 8-bit binary radians or "brads". An 8-bit brad is 1/256 of a complete rotation. Brads are particularly useful in that they leverage the inherent rollover digital adders to automatically constrain the range of angular arithmetic operations to [0°360°). Distances were stored using inches scaled by a factor of 1/6 since most node-to-node distances were multiples of 6 inches. This allowed all distances to be encoded in an 8-bit integer data type.

As an additional measure to conserve memory resources, information about the type of node was encoded in the node number:

- Node 0: Start
- Node 1-20: Ball nodes
- Node 21-41: Junction nodes
- Node 42: End

SRAM memory was particularly limited on our ATMega32 platform, so to conserve it for other uses, the node list was stored in FLASH using a switch statement lookup table. This technique forces the constant values to be encoded in Load Program Memory (LPM) instruction words which are stored and accessed directly from FLASH.

```
void getNode(uint8_t nodeNum, NODE *node)
{
    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 42:                         // STOP_NODE
        node->numAdjNodes = 1;
        node->   adjNodes[0] = 41;
        node->   adjCosts[0] = 5;
        node->adjHeadings[0] = 0;
        break;
    }
```

3.4.5 Ball Detection

Except for a couple special cases, all ball detection was done at junctions with the tilt servo oriented vertically and the panning servo oriented at a fixed angle to the left or to the right. This covered all junctions in which a ground ball could be located down a perpendicular corridor to the left or right.

The ball detection algorithm was implemented using the built-in color tracking and virtual window features of the C-MUcam. As with the line tracking, it was important that all the data processing intensive operations be done on the CMUcam processor in order to have low execution time.

At every junction, Caddy traversed all connected nodes in the graph extending to the left, for example. For any ground ball node encountered during this graph traversal that had not yet been checked, the node number was recorded along with distance away from the Caddy's current node location. If there was at least one unchecked ball node to the left, Caddy would orient the camera pan/tilt servos to look in that direction and use a sliding window algorithm to search for the ball.

As shown in the diagram below, the ball detection algorithm searched for the bottom of the ball by progressively moving a narrow image window through the image and using an upper and lower color intensity thresholds to determine if any orange pixels were within the image window.

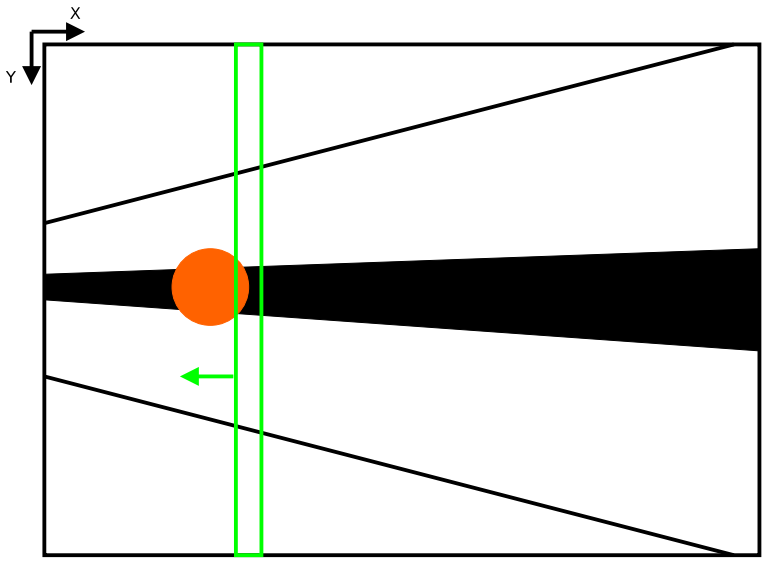


Figure 10: The camera's view while searching for a ball down a corridor to the left

If an orange pixel was found, a look-up table was used to convert the the horizontal X coordinate of the image window to a physical distance away from the bot. This distance was then compared with the values in the unchecked ball list to find the most likely node number corresponding to the blob of orange pixels.

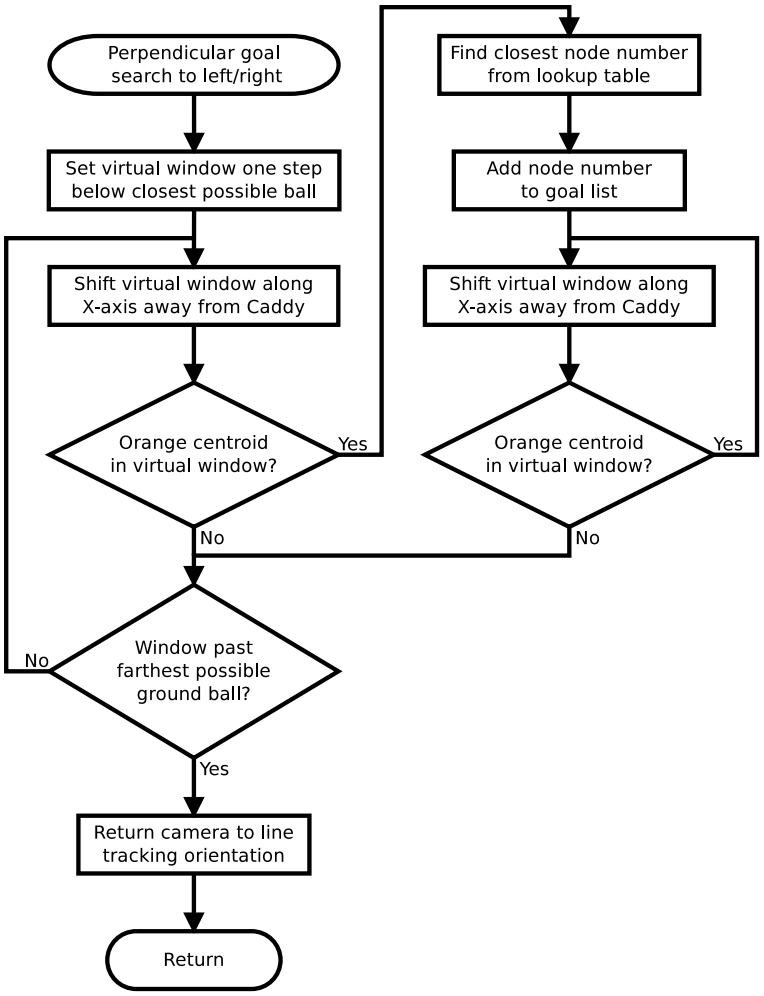


Figure 11: Perpendicular corridor ball searching flow chart

Since stopping to perform these ball searches added time to the run we made sure to mark any node that was "seen" during a ball search or physically crossed by Caddy as having been checked. Also, once the final ground ball was discovered all remaining unchecked nodes were marked as having been checked (by process of elimination). These optimizations ensured that Caddy only performed ball searches when necessary.

To optimize the computation time of the sliding window search algorithm, the window width was increased from 1 pixel to 4 pixels. We found that this gave sufficient resolution while providing a speedup of nearly 4.0. We also limited the far end of the sliding window range to cover just six inches past the farthest unchecked ball node since there was no point in search through parts of the image which we knew would not offer any new information. With these optimizations a typical ball search to the left or right took about 1 second.

3.4.6 Path Planning

The connected graph map of the arena and ball detection features described in the previous sections provided pieces of information needed for a path planning algorithm:

- A map
- A location

- One or more destinations (goals)

Now, every time a goal was added to the goal list (i.e. a ground ball was found), the path planning algorithm would compute the cost through all permutations of known goals starting from the current node and ending at nest node. This required two sub-algorithms:

- A method to compute the shortest path between any two nodes
- A method to compute all permutations of known goals

The permutation algorithm proved to be more challenging than originally expected because the limited stack space of the ATmega32 precluded the use of a recursive permuation algorithm. With some research we found a simple itertive implementation in the GNU C++ Standard Template Library (STL). Adapting this templated code to C99 C code we had a solution for computing all permuations of any array of 8-bit values.

4 Conclusion

The overall design of the robot proved to be a success. Caddy was able to perform all the tasks reliably and quickly. Unfortunately, the time constraints of the Robocompetition meant that full implemenation and debugging of all the features was not completed until just *after* the competition, when meant that Caddy did not rank as well as it have.

The following section describes some of the technical aspects that, in hindsight, we would have chosen to do differently.

4.1 Future Work

4.1.1 Regulated Motor Voltage

If we end up working with higher voltage motors again, it may be worth the added cost and decrease in power efficiency of regulating the motor voltage. This allows for more consistent operation across fully charged and partially charged batteries, especially with low precision wheel encoders that do not sense direction.

4.1.2

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

4.1.3 Quadrature Wheel Encoders

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

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

5 Appendix

5.1 Gantt Chart

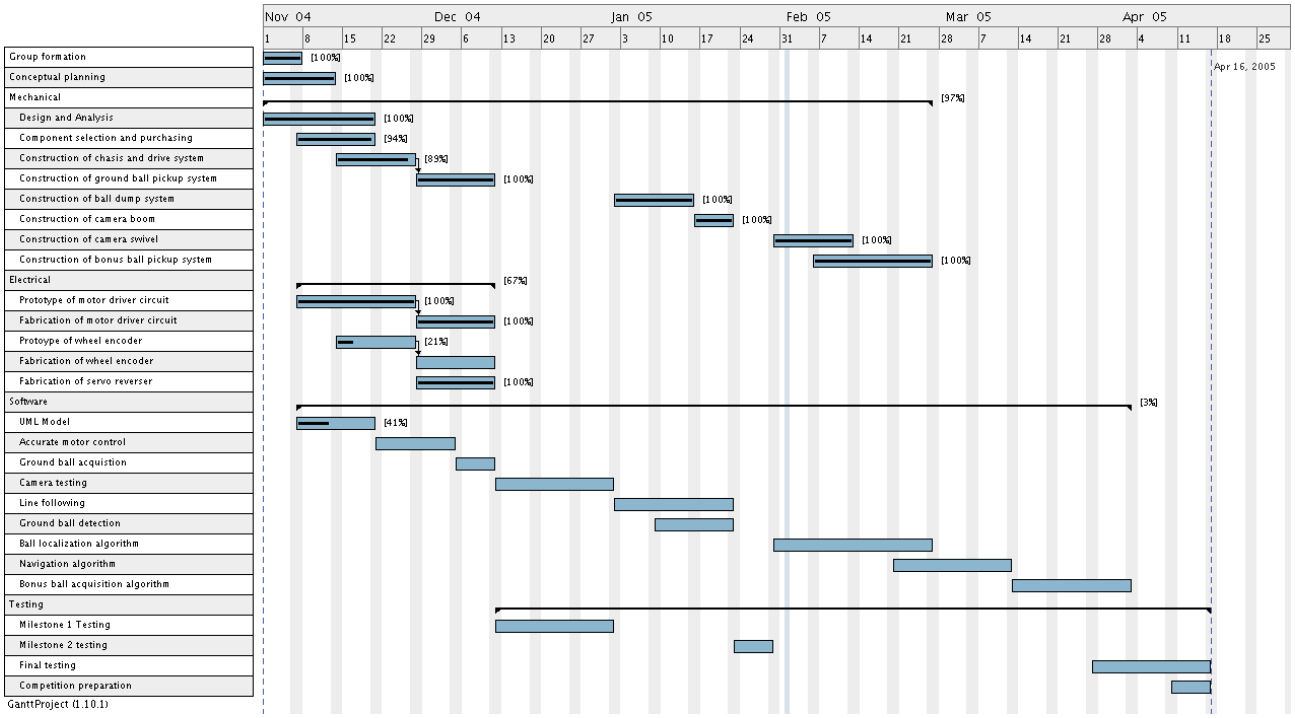


Figure 12: Caddy project gantt chart

5.2 Individual Contributions

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

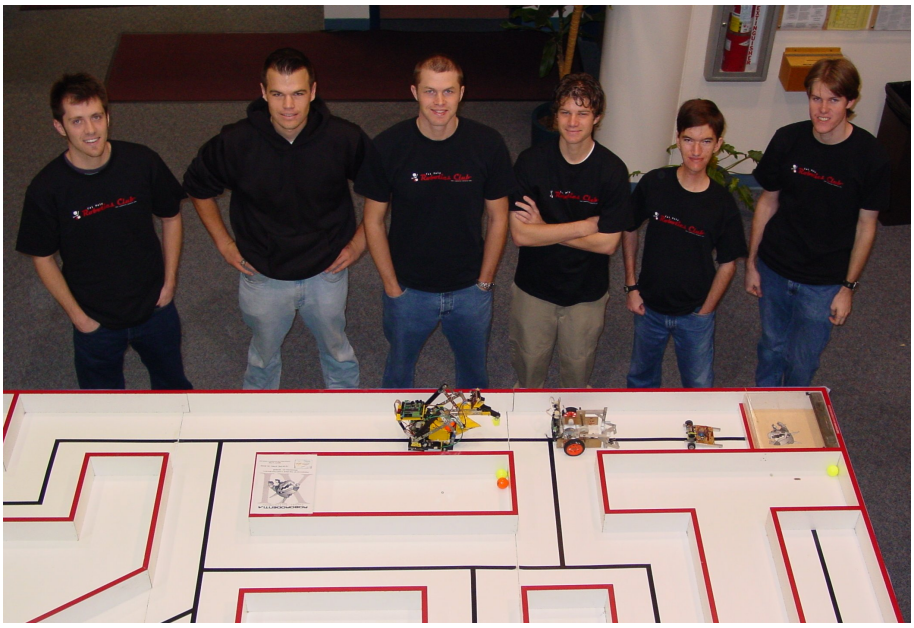


Figure 13: Team Photo. Left to right: Logan Kinde, Tyson Messori, Taylor Braun-Jones, Scott Barlow, Michael Shelley, Patrick McCarty

5.2.1 Contributions of Taylor Braun-Jones

Taylor was responsible for overall project coordination and administration including:

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

Taylor's electrical and mechanical contributions include:

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

The software contributions are attributed as follows:

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

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

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

5.3 Acknowledgments

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

6 Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

7 File Index

7.1 File List

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

8 Data Structure Documentation

8.1 GraphNode Struct Reference

Definition of each node (vertex) in the course map node.

`#include <node_list.h>`

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 of each node (vertex) in the course map node.

Defines the directions and distances to adjacent nodes.

Definition at line 91 of file node_list.h.

8.1.2 Field Documentation

8.1.2.1 uint8_t GraphNode::adjCosts[MAX_ADJ_NODES]

distances to adjacent nodes (6 inches increments)

Definition at line 104 of file node_list.h.

8.1.2.2 int8_t GraphNode::adjHeadings[MAX_ADJ_NODES]

directions towards adjacent nodes in 8-bit brads

Definition at line 108 of file node_list.h.

8.1.2.3 uint8_t GraphNode::adjNodes[MAX_ADJ_NODES]

node numbers of adjacent nodes

Definition at line 100 of file node_list.h.

8.1.2.4 uint8_t GraphNode::numAdjNodes

number of nodes adjacent to this node

Definition at line 96 of file node_list.h.

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

- node_list.h

8.2 PathListNode Struct Reference

Collaboration diagram for PathListNode:

Data Fields

- uint8_t **nodeNum**
- struct PathListNode * **nextNode**

8.2.1 Detailed Description

Definition at line 39 of file path_planning.c.

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

- path_planning.c

8.3 SearchNode Struct Reference

Data Fields

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

8.3.1 Detailed Description

Definition at line 32 of file path_planning.c.

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

- path_planning.c

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 ball_tracking.c File Reference

```
#include "ball_tracking.h"
#include "line_tracking.h"
#include "camera.h"
#include "servos.h"
#include "robot_control.h"
#include "motor_control.h"
#include "tweak_data.h"
#include "utility.h"
#include "rprintf.h"
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for ball_tracking.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**

9.1.1 Detailed Description

Definition in file ball_tracking.c.

9.2 ball_tracking.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 "ball_tracking.h"
00019 #include "line_tracking.h"
00020 #include "camera.h"
00021 #include "servos.h"
00022 #include "robot_control.h"
00023 #include "motor_control.h"
00024 #include "tweak_data.h"
00025 #include "utility.h"
00026
00027 // AVRLIB
00028 #include "rprintf.h"
00029
00030 // avr-libc
00031 #include <stdint.h>
00032 #include <stdbool.h>
00033
00034 // Track the RED ball on black/white background
00035 #define BALL_RMIN    150
00036 #define BALL_RMAX    240
00037 #define BALL_GMIN    16
00038 #define BALL_GMAX    60
00039 #define BALL_BMIN    16
00040 #define BALL_BMAX    50
00041
00042 // Global variables
00043 volatile bool colorStatsProcessed;
00044 bool inSeekPosition;
00045
00046 static uint8_t distToPix( uint8_t distance );
00047
00048 void trackColorInit(int8_t dir)
00049 {
00050     if (!inSeekPosition)
00051     {
00052         brake(BOTH_MOTORS);
00053         msDelay(200);
00054         moveStraight(-1 * 0xb, 255);
00055         inSeekPosition = true;
00056     }
00057
00058     // Set pan (center) and tilt
00059     switch (dir)
00060     {
00061     case LOOK_LEFT:
00062         setServo(PAN, PAN_CENTER + panOffset + PAN_SEEK_OFFSET);
00063         setServo(TILT, TILT_VERT + tiltOffset);
```

```
00064         break;
00065     case LOOK_RIGHT:
00066         setServo(PAN, PAN_CENTER + panOffset - PAN_SEEK_OFFSET);
00067         setServo(TILT, TILT_VERT + tiltOffset);
00068         break;
00069     case LOOK_UP:
00070         setServo(PAN, PAN_CENTER + panOffset);
00071         setServo(TILT, TILT_LOOKUP);
00072         break;
00073     default:
00074         break;
00075     }
00076     msDelay(500);
00077 
00078     cameraHighResMode();
00079     rprintf("DS 1 1\r");
00080     rprintf("LM 0 0\r");
00081 
00082     // Change to poll mode so only one packet is sent
00083     rprintf("PM 1\r");
00084 }
00085 
00086 /*
00087  * Returns Y1 (top of ball) if camera sees a ball, zero otherwise
00088  */
00089 uint8_t getBallY( void )
00090 {
00091     rprintf("lm 0 0\r");
00092 
00093     // Mask everything but the 'My' value
00094     //rprintf("OM 0 2\r");                              //<- NO MASKING?
00095 
00096     // Change to poll mode so only one packet is sent
00097     rprintf("PM 1\r");
00098 
00099     // Track red
00100     rprintf("TC %d %d %d %d %d %d\r",
00101                     BALL_RMIN, BALL_RMAX, BALL_GMIN, BALL_GMAX, BALL_BMIN,
      BALL_BMAX);
00102 
00103     colorStatsProcessed = true;
00104     while (colorStatsProcessed) ;
00105 
00106     return (lineStats[0][Y1_NDX]);
00107 }
00108 
00109 
00110 bool seeBall( void )
00111 {
00112    // Track red
00113    rprintf("TC %d %d %d %d %d %d\r",
00114            BALL_RMIN, BALL_RMAX, BALL_GMIN, BALL_GMAX, BALL_BMIN, BALL_BMAX);
00115    colorStatsProcessed = true;
00116    while (colorStatsProcessed) ;
00117 
00118    return lineStats[0][Y1_NDX] > 0;
00119 }
00120 
00121 
00122 /*
00123  * Just does left seeks
00124  *    PRE - the longest check is the last element of the uncheckedBalls array
00125  *
00126  *    uncheckedBalls - ball node numbers and ground distances away from bot
00127  */
00128 bool cameraSeekLeft( uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls )
00129 {
00130     bool foundBall = false;    // Return value
00131     uint8_t scanHeight = 4;
00132     uint8_t x = 174;
00133     //uint8_t ballDist[3];
00134     //uint8_t ballCount = 0;
00135     uint8_t scanLimit = distToPix(
00136                     uncheckedBalls[numUncheckedBalls - 1][BALL_DIST] + 1);
00137 
00138     // get pixel ranges for unchecked balls passed in
00139     uint8_t i = 0;
00140     uint8_t maxBallX[3];
00141     while (i + 1 < numUncheckedBalls)
00142     {
00143         maxBallX[i] = (distToPix(uncheckedBalls[i][BALL_DIST]) +
```

```
00144                         distToPix(uncheckedBalls[i + 1][BALL_DIST])) / 2;
00145         i++;
00146     }
00147     maxBallX[i] = scanLimit;
00148 
00149     // scan from small ground distance to large ground distance
00150     while (x - scanHeight > scanLimit)
00151     {
00152         x -= scanHeight;
00153         setVirtualWindow(x - scanHeight, 1, x, 254);
00154         if (seeBall())
00155         {
00156             foundBall = true;
00157             //ballDist[ballCount++] = xToDist(x);
00158 
00159             // find ball number of ball at this x
00160             i = 0;
00161             while (maxBallX[i] > x)
00162             {
00163                 i++;
00164             }
00165             addToGoalList(uncheckedBalls[i][BALL_NODE_NUM]);
00166 
00167             while (seeBall())
00168             {
00169                 x -= scanHeight;
00170                 setVirtualWindow(x - scanHeight, 1, x, 254);
00171             }
00172         }
00173     }
00174 
00175     return foundBall;
00176 }
00177 
00178 // returns pixel equivalent of 'distance'
00179 static uint8_t distToPix( uint8_t distance )
00180 {
00181     switch (distance)
00182     {
00183     case 0:
00184     case 1:
00185         return 174;
00186     case 2:
00187         return 0x8d;
00188     case 3:
00189         return 0x61;
00190     case 4:
00191         return 0x48;
00192     case 5:
00193         return 0x36;
00194     case 6:
00195         return 0x2b;
00196     case 7:
00197         return 0x22;
00198     case 8:
00199         return 0x1d;
00200     case 9:
00201         return 0x18;
00202     case 10:
00203         return 0x14;
00204     case 11:
00205         return 0x11;
00206     case 12:
00207         return 0x0e;
00208     default:
00209         return 0x0;
00210     }
00211 }
00212 
00213 /*
00214  * Just does right seeks
00215  *    PRE - the longest check is the last element of the uncheckedBalls array
00216  *
00217  *    uncheckedBalls - ball node numbers and ground distances away from bot
00218  */
00219 bool cameraSeekRight(uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls)
00220 {
00221     bool foundBall = false;    // Return value
00222     uint8_t scanHeight = 4;
00223     uint8_t x = 0;
00224     uint8_t scanLimit = 174 - distToPix(
```

```
00225                     uncheckedBalls[numUncheckedBalls - 1][BALL_DIST] + 1);
00226
00227     // get pixel ranges for unchecked balls passed in
00228     uint8_t i = 0;
00229     uint8_t maxBallX[3];
00230     while (i + 1 < numUncheckedBalls)
00231     {
00232         maxBallX[i] = ((174 - distToPix(uncheckedBalls[i][BALL_DIST])) +
00233                        (174 - distToPix(uncheckedBalls[i + 1][BALL_DIST])))
00234                        / 2;
00235         i++;
00236     }
00237     maxBallX[i] = scanLimit;
00238
00239     // scan from small ground distance to large ground distance
00240     while (x + scanHeight < scanLimit)
00241     {
00242         x += scanHeight;
00243         setVirtualWindow(x, 1, x + scanHeight, 254);
00244         if (seeBall())
00245         {
00246             foundBall = true;
00247             //ballDist[ballCount++] = xToDist(x);
00248
00249             // find ball number of ball at this x
00250             i = 0;
00251             while (maxBallX[i] < x)
00252             {
00253                 i++;
00254             }
00255             addToGoalList(uncheckedBalls[i][BALL_NODE_NUM]);
00256
00257             while (seeBall())
00258             {
00259                 x += scanHeight;
00260                 setVirtualWindow(x, 1, x + scanHeight, 254);
00261             }
00262         }
00263     }
00264
00265     return foundBall;
00266 }
00267
```

9.3 ball_tracking.h File Reference

Simple tracking Roborodentia objects of interest by color.

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

Include dependency graph for ball_tracking.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**

9.3.1 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 ball_tracking.h.

9.4 ball_tracking.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00025 #ifndef TRACKCOLOR_H_
00026 #define TRACKCOLOR_H_
00027
00028 // avr-libc
00029 #include <stdint.h>
00030 #include <stdbool.h>
00031
00032 #define LOOK_RIGHT   1
00033 #define LOOK_LEFT    -1
00034 #define LOOK_UP      0
00035
00036 #define MX_NDX            0
00037 #define MY_NDX            1
00038 #define X1_NDX            2
00039 #define Y1_NDX            3
00040 #define X2_NDX            4
00041 #define Y2_NDX            5
00042 #define PIXEL_CNT_NDX     6
00043 #define CONFIDENCE_NDX    7
00044
00045 #define NUM_COLOR_STATS    8
00046
00047 #define PAN_SEEK_OFFSET    66
00048
```

```
00049 // Global variables
00050 extern volatile bool colorStatsProcessed;
00051 extern bool inSeekPosition;
00052
00053 void trackColorInit(int8_t dir);
00054 uint8_t getBallY( void );
00055 bool seeBall( void );
00056 bool cameraSeekLeft( uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls );
00057 bool cameraSeekRight( uint8_t uncheckedBalls[][2], uint8_t numUncheckedBalls );
00058
00059 #endif // #ifndef TRACKCOLOR_H_
```

9.5 buttons.c File Reference

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

Include dependency graph for buttons.c:

Macros

- #define **DEBOUNCE_COUNT** 3

Functions

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

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

9.5.1 Detailed Description

Definition in file buttons.c.

9.5.2 Function Documentation

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

Returns

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

Definition at line 114 of file buttons.c.

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

Returns

true when confirmed rising edge at last debouncing.

Definition at line 50 of file buttons.c.

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

Returns

true when confirmed falling edge at last debouncing.

Definition at line 58 of file buttons.c.

9.6 buttons.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "buttons.h"
00019
00020 // AVRLIB
00021 #include "avrlibdefs.h"
00022
00023 // avr-libc
00024 #include <stdint.h>
00025 #include <stdbool.h>
00026
00027 #define DEBOUNCE_COUNT   3   // should be equal to 2 or greater
00028
00029 static bool isDown[NUM_BUTTONS]   = { false, false, false,
00030                                       false, false, false };
00031 static bool wasEvent[NUM_BUTTONS] = { false, false, false,
00032                                       false, false, false };
00033 static uint8_t upCount[NUM_BUTTONS] = { DEBOUNCE_COUNT, DEBOUNCE_COUNT,
00034                                         DEBOUNCE_COUNT, DEBOUNCE_COUNT,
00035                                         DEBOUNCE_COUNT, DEBOUNCE_COUNT };
00036 static uint8_t downCount[NUM_BUTTONS] = { 0, 0, 0, 0, 0, 0 };
00037
00038 void waitFor(uint8_t button)
00039 {
00040     debounceButtons();
00041     while (!justReleased(button))
00042     {
00043         debounceButtons();
00044     }
00045 }
00046
00050 inline bool justPressed(uint8_t button)
00051 {
00052     return wasEvent[button] && isDown[button];
00053 }
00054
00058 inline bool justReleased(uint8_t button)
00059 {
00060     return wasEvent[button] && !isDown[button];
00061 }
00062
00066 void debounceButtons(void)
00067 {
00068     uint8_t button;
00069     for (button = 0; button < NUM_BUTTONS; button++)
00070     {
00071         // count times buttons have been consecutively up/down (upto
      DEBOUNCE_COUNT).
00072         if (isPressed(button))
00073         {
00074             downCount[button] = MIN(downCount[button]+1, DEBOUNCE_COUNT);
00075             upCount[button] = 0;
```

```
00076         }
00077         else
00078         {
00079             upCount[button] = MIN(upCount[button]+1, DEBOUNCE_COUNT);
00080             downCount[button] = 0;
00081         }
00082 
00083         // check for confirmed up/down event
00084         if (isDown[button])
00085         {
00086             if (upCount[button] >= DEBOUNCE_COUNT)
00087             {
00088                 isDown[button] = false;
00089                 wasEvent[button] = true;
00090             }
00091             else
00092             {
00093                 wasEvent[button] = false;
00094             }
00095         }
00096         else
00097         {
00098             if (downCount[button] >= DEBOUNCE_COUNT)
00099             {
00100                 isDown[button] = true;
00101                 wasEvent[button] = true;
00102             }
00103             else
00104             {
00105                 wasEvent[button] = false;
00106             }
00107         }
00108     }
00109 }
00110 
00114 inline bool isPressed(uint8_t button)
00115 {
00116     switch (button)
00117     {
00118     case RED_BUTTON:      return RED_BUTTON_DOWN;
00119     case L_UP_BUTTON:     return L_UP_BUTTON_DOWN;
00120     case L_DOWN_BUTTON:   return L_DOWN_BUTTON_DOWN;
00121     case R_UP_BUTTON:     return R_UP_BUTTON_DOWN;
00122     case R_DOWN_BUTTON:   return R_DOWN_BUTTON_DOWN;
00123     case NEST_BUTTON:     return NEST_BUTTON_DOWN;
00124     default:              break;
00125     }
00126 
00127     return false;
00128 }
00129 
00130 inline bool bothRightButtonsPressed(void)
00131 {
00132     return (justPressed(R_UP_BUTTON) && justPressed(
      R_DOWN_BUTTON)) ||
00133             (justPressed(R_UP_BUTTON) && isDown[R_DOWN_BUTTON])
      ||
00134             (justPressed(R_DOWN_BUTTON) && isDown[R_UP_BUTTON]);
00135 
00136 }
00137 
00138 inline bool bothLeftButtonsPressed(void)
00139 {
00140     return (justPressed(L_UP_BUTTON) && justPressed(
      L_DOWN_BUTTON)) ||
00141             (justPressed(L_UP_BUTTON) && isDown[L_DOWN_BUTTON])
      ||
00142             (justPressed(L_DOWN_BUTTON) && isDown[L_UP_BUTTON]);
00143 
00144 }
```

9.7 buttons.h File Reference

Button debouncing, start bot logic.

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

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

Macros

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

Functions

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

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

9.7.1 Detailed Description

Button debouncing, start bot logic.

Definition in file buttons.h.

9.7.2 Function Documentation

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

Returns

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

Definition at line 114 of file buttons.c.

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

Returns

true when confirmed rising edge at last debouncing.

Definition at line 50 of file buttons.c.

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

Returns

true when confirmed falling edge at last debouncing.

Definition at line 58 of file buttons.c.

9.8 buttons.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00021 #ifndef BUTTONS_H_
00022 #define BUTTONS_H_
00023
00024 #include <avr/io.h>
00025 #include <stdint.h>
00026 #include <stdbool.h>
00027
00028 #define RED_BUTTON       0
00029 #define L_UP_BUTTON      1
00030 #define L_DOWN_BUTTON    2
00031 #define R_UP_BUTTON      3
00032 #define R_DOWN_BUTTON    4
00033 #define NEST_BUTTON      5
00034 #define NUM_BUTTONS      6    // change isPressed(uint8_t button) when adding a
       button
00035
00036 #define RED_BUTTON_DOWN       bit_is_clear(PIND,6)
00037 #define L_UP_BUTTON_DOWN      bit_is_clear(PINA,0)
00038 #define L_DOWN_BUTTON_DOWN    bit_is_clear(PINA,1)
00039 #define R_UP_BUTTON_DOWN      bit_is_clear(PINA,2)
00040 #define R_DOWN_BUTTON_DOWN    bit_is_clear(PINA,3)
00041 #define NEST_BUTTON_DOWN      bit_is_clear(PINB,0)
00042
00043 #define BREAK_BEAM_TRIGGERED  bit_is_set(PINB,1)
00044
00045 void initButtons(void);
00046 void waitFor(uint8_t button);
00047 inline bool justPressed(uint8_t button);
00048 inline bool justReleased(uint8_t button);
00049 void debounceButtons(void);
00050 inline bool isPressed(uint8_t button);
00051 inline bool bothRightButtonsPressed(void);
00052 inline bool bothLeftButtonsPressed(void);
00053
00054 #endif // #ifndef BUTTONS_H_
```

9.9 caddy.c File Reference

Caddy's main loop and Atmel initialization.

```
#include "robot_control.h"
#include "motor_control.h"
#include "camera.h"
#include "encoder.h"
#include "buttons.h"
#include "tweak_data.h"
#include "utility.h"
#include "lcd_16x2.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 37 of file caddy.c.

9.10 caddy.c

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

```
00023 #include "camera.h"
00024 #include "encoder.h"
00025 #include "buttons.h"
00026 #include "tweak_data.h"
00027 #include "utility.h"
00028 #include "lcd_16x2.h"
00029
00030 // avr-libc
00031 #include <avr/io.h>
00032
00037 #define START_DELAY 5
00038
00042 static inline void initAtmel(void)
00043 {
00044     /*
00045      * Initialize Timer
00046      */
00047     timerInit();
00048
00049 #if DEBUGGING
00050     /*
00051      * Initialize LCD
00052      */
00053     lcdInit();
00054     lcdWriteStr("Init:           ", 0, 0);
00055     lcdWriteStr("                ", 1, 0);
00056 #endif
00057
00058     /*
00059      * Initialize UART for CMUcam communication
00060      */
00061     cmuCamInit();
00062
00063     /*
00064      * Initialize PWM motor control
00065      */
00066     outb(DDRD, 0xff);
00067     timer1PWMInit(8);
00068     neutral();
00069     enableMotors();
00070
00071     /*
00072      * Set data direction registers
00073      */
00074     outb(DDRA, 0xF0); // Motor control and up/down buttons
00075     cbi(DDRD, 6);     // red button
00076     cbi(DDRB, 0);     // nest button
00077     cbi(DDRB, 1);     // break beam
00078
00079     /*
00080      * Apply internal pull-up resister to certain digital inputs
00081      */
00082     sbi(PORTB, 0); // internal pull-up for PINB0
00083     sbi(PORTA, 3); // internal pull-up for PINA3
00084     sbi(PORTA, 2); // internal pull-up for PINA2
00085     sbi(PORTA, 1); // internal pull-up for PINA1
00086     sbi(PORTA, 0); // internal pull-up for PINA0
00087
00088     /*
00089      * Initialize quadrature wheel encoders
00090      */
00091     cbi(DDRD, 2);
00092     cbi(DDRD, 3);
00093     encoderInit();
00094 }
00095
00099 int main(void)
00100 {
00101     initAtmel();
00102     loadTweakValues();
00103     initCamera();
00104     cameraWhiteBalance();
00105
00106 #if DEBUGGING
00107     runTetherUI();
00108     myDelay(START_DELAY);
00109     runTest();
00110 #else
00111     waitFor(RED_BUTTON);
00112     myDelay(START_DELAY);
00113     runRoborodentiaCourse();
```

```
00114 #endif
00115
00116     brake(BOTH_MOTORS);
00117 #if DEBUGGING
00118     lcdWriteStr("Done            ", 0, 0);
00119     lcdWriteStr("                ", 1, 0);
00120 #endif
00121
00122     return 0;
00123 }
```

9.11 camera.c File Reference

```
#include "camera.h"
#include "ball_tracking.h"
#include "line_tracking.h"
#include "utility.h"
#include "lcd_16x2.h"
#include "rprintf.h"
#include "uart.h"
#include <stdbool.h>
```

Include dependency graph for camera.c:

Macros

- #define **NEW_PACKET** 0
- #define **FE_RCV** 1
- #define **T_RCV** 2
- #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 initCamera (void)

  Reset the packet receiving state and put the camera into raw send mode.
- void cameraStreamingOff (void)

  Stop the CMUcam from streaming data.
- void setVirtualWindow (uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2)

  Constrain field of view used for subsequent image processing commands.
- void cameraHighResMode (void)

  Set the camera to 176x255 resolution.
- void cameraLowResMode (void)

  Set the camera to 88x143 resolution.

9.11.1 Detailed Description

Definition in file camera.c.

9.11.2 Function Documentation

9.11.2.1 void cameraHighResMode (void) [inline]

Set the camera to 176x255 resolution.

Note

This resolution is *truncated* from 176x287

Definition at line 159 of file camera.c.

9.11.2.2 void cameraStreamingOff (void) [inline]

Stop the CMUcam from streaming data.

See Also

CMUcam2 manual p.30

Definition at line 147 of file camera.c.

9.11.2.3 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 53 of file camera.c.

9.11.2.4 void initCamera (void) [inline]

Reset the packet receiving state and put the camera into raw send mode.

See Also

CMUcam2 manual p.48

Definition at line 65 of file camera.c.

9.11.2.5 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 154 of file camera.c.

9.12 camera.c

```
00001 /*
```

```
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 #include "camera.h"
00019 #include "ball_tracking.h"
00020 #include "line_tracking.h"
00021 #include "utility.h"
00022 #include "lcd_16x2.h"
00023
00024 // AVRLIB
00025 #include "rprintf.h"
00026 #include "uart.h"
00027
00028 // avr-libc
00029 #include <stdbool.h>
00030
00031 // Packet types
00032 #define NEW_PACKET   0
00033 #define FE_RCV       1
00034 #define T_RCV        2
00035
00036 #define CMU_BAUD 38400
00037
00038 static uint8_t mode;
00039 static uint16_t byteNum;
00040
00041 void packetRcv( uint8_t c );
00042 inline void lineMode2Rcv( uint8_t c );
00043 inline void trackColorRcv( uint8_t c );
00044
00045 inline void cmuCamInit(void)
00046 {
00047     uartInit();
00048     uartSetBaudRate(CMU_BAUD);
00049     uartSetRxHandler(packetRcv);
00050     rprintfInit(uartSendByte);
00051 }
00052
00053 inline void cameraWhiteBalance()
00054 {
00055     // turn auto white balance on
00056 #if DEBUGGING
00057     lcdWriteStr("white Bal ", 0, 6);
00058 #endif
00059     rprintf("CR 18 44\r");
00060     myDelay(200);
00061     // turn auto white balance off
00062     rprintf("CR 18 40\r");
00063 }
00064
00065 inline void initCamera( void )
00066 {
00067     mode = NEW_PACKET;
00068     byteNum = 0;
00069
00070     /*
00071      * Applies the following settings:
00072      *  - Output from the camera is in raw bytes
00073      *  - "ACK\r" and "NCK\r" confirmations are suppressed
00074      *  - Input to the camera is in ASCII
00075      */
00076     rprintf("RM 3\r");
00077 }
00078
00079
00080 void packetRcv(uint8_t c)
00081 {
00082     if (c == 0xff)
00083     {
```

```
00084         mode = NEW_PACKET;
00085         byteNum = 0;
00086     }
00087     else
00088     {
00089         switch (mode)
00090         {
00091         case NEW_PACKET:
00092             switch (c)
00093             {
00094             case 0xfe:
00095                 mode = FE_RCV;
00096                 break;
00097             case 'T':
00098                 mode = T_RCV;
00099                 break;
00100             default:
00101                 break;
00102             }
00103             break;
00104         case FE_RCV:
00105             lineMode2Rcv(c);
00106             if (c == 0xfd)
00107             {
00108                 mode = NEW_PACKET;
00109             }
00110             break;
00111         case T_RCV:
00112             trackColorRcv(c);
00113             break;
00114         }
00115     }
00116 }
00117
00118
00119 inline void lineMode2Rcv(uint8_t c)
00120 {
00121     if (c == 0xfd)
00122     {
00123         lineStatsProcessed = false;
00124         byteNum = 0;
00125     }
00126     else
00127     {
00128         lineStats[(byteNum - 1) / LINE_STATS_COLS]
00129                  [(byteNum - 1) % LINE_STATS_COLS] = c;
00130         byteNum++;
00131     }
00132 }
00133
00134
00135 inline void trackColorRcv(uint8_t c)
00136 {
00137     lineStats[0][byteNum] = c;
00138     byteNum++;
00139
00140     if (byteNum >= NUM_COLOR_STATS)
00141     {
00142         colorStatsProcessed = false;
00143     }
00144 }
00145
00146
00147 inline void cameraStreamingOff( void )
00148 {
00149     rprintf("\r\r"); // add an extra return as recommended by CMUcam manual
00150     msDelay(32);     // wait for streaming to stop ( 16ms delay ok )
00151 }
00152
00153
00154 inline void setVirtualWindow(uint8_t x1, uint8_t y1, uint8_t x2
      , uint8_t y2)
00155 {
00156     rprintf("VW %d %d %d %d\r", x1, y1, x2, y2);
00157 }
00158
00159 inline void cameraHighResMode(void)
00160 {
00161     rprintf("HR 1\r");
00162 }
00163
```

```
00164 inline void cameraLowResMode(void)
00165 {
00166     rprintf("HR 0\r");
00167 }
```

9.13 camera.h File Reference

Interface to the CMUcam vision system.

`#include <stdint.h>`

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

Functions

- void cmuCamInit (void)

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

  Optimize the white balance for current conditions.
- void initCamera (void)

  Reset the packet receiving state and put the camera into raw send mode.
- void cameraStreamingOff (void)

  Stop the CMUcam from streaming data.
- void setVirtualWindow (uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2)

  Constrain field of view used for subsequent image processing commands.
- void cameraHighResMode (void)

  Set the camera to 176x255 resolution.
- void cameraLowResMode (void)

  Set the camera to 88x143 resolution.

9.13.1 Detailed Description

Interface to the CMUcam vision system.

Definition in file camera.h.

9.13.2 Function Documentation

9.13.2.1 void cameraHighResMode (void) [inline]

Set the camera to 176x255 resolution.

Note

This resolution is *truncated* from 176x287

Definition at line 159 of file camera.c.

9.13.2.2 void cameraStreamingOff (void) [inline]

Stop the CMUcam from streaming data.

See Also

CMUcam2 manual p.30

Definition at line 147 of file camera.c.

9.13.2.3 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 53 of file camera.c.

9.13.2.4 void initCamera (void) [inline]

Reset the packet receiving state and put the camera into raw send mode.

See Also

CMUcam2 manual p.48

Definition at line 65 of file camera.c.

9.13.2.5 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 154 of file camera.c.

9.14 camera.h

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

```
00021 #define CAMERA_H_
00022
00023 #include <stdint.h>
00024
00028 inline void cmuCamInit(void);
00029
00038 inline void cameraWhiteBalance( void );
00039
00045 inline void initCamera( void );
00046
00052 inline void cameraStreamingOff( void );
00053
00059 inline void setVirtualWindow(uint8_t x1, uint8_t y1, uint8_t x2
      , uint8_t y2);
00060
00066 inline void cameraHighResMode(void);
00067
00071 inline void cameraLowResMode(void);
00072
00073 #endif // #ifndef CAMERA_H_
```

9.15 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]

9.15.1 Detailed Description

Quadrature Encoder reader/driver.

Definition in file encoder.c.

9.16 encoder.c

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 //*****************************************************************************
00019 //
00020 // File Name  : 'encoder.c'
00021 // Title    : Quadrature Encoder reader/driver
00022 // Author   : Pascal Stang - Copyright (C) 2003-2004
00023 // Created    : 2003.01.26
00024 // Revised    : 2004.06.25
00025 // Version    : 0.3
00026 // Target MCU : Atmel AVR Series
00027 // Editor Tabs  : 4
00028 //
00029 // NOTE: This code is currently below version 1.0, and therefore is considered
00030 // to be lacking in some functionality or documentation, or may not be fully
00031 // tested.  Nonetheless, you can expect most functions to work.
00032 //
00033 // This code is distributed under the GNU Public License
00034 //    which can be found at http://www.gnu.org/licenses/gpl.txt
00035 //
00036 //*****************************************************************************
00037 #ifndef WIN32
00038 #include <avr/io.h>
00039 #include <avr/interrupt.h>
00040 #endif
00041
00042 #include "global.h"
00043 #include "encoder.h"
00044
00045 // Program ROM constants
00046
00047 // Global variables
00048 volatile EncoderStateType EncoderState[NUM_ENCODERS];
00049
00050 // Functions
00051
00052 // encoderInit() initializes hardware and encoder position readings
00053 //    Run this init routine once before using any other encoder functions.
00054 inline void encoderInit(void)
00055 {
00056     uint8_t i;
00057
00058     // initialize/clear encoder data
00059     for (i = 0; i < NUM_ENCODERS; i++)
00060     {
00061         EncoderState[i].position = 0;
00062         //EncoderState[i].velocity = 0;   // NOT CURRENTLY USED
00063     }
00064
00065     // configure direction and interrupt I/O pins:
00066     // - for input
00067     // - apply pullup resistors
00068     // - any-edge interrupt triggering
00069     // - enable interrupt
00070
00071 #ifdef ENC0_SIGNAL
00072     // set interrupt pins to input and apply pullup resistor
00073     cbi(ENC0_PHASEA_DDR, ENC0_PHASEA_PIN);
00074     sbi(ENC0_PHASEA_PORT, ENC0_PHASEA_PIN);
00075     // configure interrupts for any-edge triggering
00076     sbi(ENC0_ICR, ENC0_ISCX0);
00077     cbi(ENC0_ICR, ENC0_ISCX1);
00078     // enable interrupts
00079     sbi(IMSK, ENC0_INT);
```

```
00080     // ISMK is auto-defined in encoder.h
00081 #endif
00082 #ifdef ENC1_SIGNAL
00083     // set interrupt pins to input and apply pullup resistor
00084     cbi(ENC1_PHASEA_DDR, ENC1_PHASEA_PIN);
00085     sbi(ENC1_PHASEA_PORT, ENC1_PHASEA_PIN);
00086     // configure interrupts for any-edge triggering
00087     sbi(ENC1_ICR, ENC1_ISCX0);
00088     cbi(ENC1_ICR, ENC1_ISCX1);
00089     // enable interrupts
00090     sbi(IMSK, ENC1_INT);
00091     // ISMK is auto-defined in encoder.h
00092 #endif
00093 #ifdef ENC2_SIGNAL
00094     // set interrupt pins to input and apply pullup resistor
00095     cbi(ENC2_PHASEA_DDR, ENC2_PHASEA_PIN);
00096     sbi(ENC2_PHASEA_PORT, ENC2_PHASEA_PIN);
00097     // configure interrupts for any-edge triggering
00098     sbi(ENC2_ICR, ENC2_ISCX0);
00099     cbi(ENC2_ICR, ENC2_ISCX1);
00100     // enable interrupts
00101     sbi(IMSK, ENC2_INT);// ISMK is auto-defined in encoder.h
00102 #endif
00103 #ifdef ENC3_SIGNAL
00104     // set interrupt pins to input and apply pullup resistor
00105     cbi(ENC3_PHASEA_DDR, ENC3_PHASEA_PIN);
00106     sbi(ENC3_PHASEA_PORT, ENC3_PHASEA_PIN);
00107     // set encoder direction pin for input and apply pullup resistor
00108     cbi(ENC3_PHASEB_DDR, ENC3_PHASEB_PIN);
00109     sbi(ENC3_PHASEB_PORT, ENC3_PHASEB_PIN);
00110     // configure interrupts for any-edge triggering
00111     sbi(ENC3_ICR, ENC3_ISCX0);
00112     cbi(ENC3_ICR, ENC3_ISCX1);
00113     // enable interrupts
00114     sbi(IMSK, ENC3_INT);// ISMK is auto-defined in encoder.h
00115 #endif
00116
00117     // enable global interrupts
00118     sei();
00119 }
00120
00121 // encoderGetPosition() reads the current position of the encoder
00122 uint16_t encoderGetPosition(uint8_t encoderNum)
00123 {
00124     // sanity check
00125     if (encoderNum < NUM_ENCODERS)
00126         return EncoderState[encoderNum].position;
00127     else
00128         return 0;
00129 }
00130
00131 // encoderSetPosition() sets the current position of the encoder
00132 void encoderSetPosition(uint8_t encoderNum, uint16_t position
      )
00133 {
00134     // sanity check
00135     if (encoderNum < NUM_ENCODERS)
00136         EncoderState[encoderNum].position = position;
00137     // else do nothing
00138 }
00139
00140 #ifdef ENC0_SIGNAL
00141
00142 SIGNAL(ENC0_SIGNAL)
00143 {
00144     /*****************************************/
00145     /* Modified by Taylor                    */
00146     /*****************************************/
00147     EncoderState[0].position++;
00148 }
00149 #endif
00150
00151 #ifdef ENC1_SIGNAL
00152
00153 SIGNAL(ENC1_SIGNAL)
00154 {
00155     /*****************************************/
00156     /* Modified by Taylor                    */
00157     /*****************************************/
00158     EncoderState[1].position++;
00159 }
```

```
00160 #endif
00161
00162 #ifdef ENC2_SIGNAL
00163
00164 SIGNAL(ENC2_SIGNAL)
00165 {
00166     // encoder has generated a pulse
00167     // check the relative phase of the input channels
00168     // and update position accordingly
00169     if( ((inb(ENC2_PHASEA_PORTIN) & (1<<ENC2_PHASEA_PIN)) == 0) ^
00170         ((inb(ENC2_PHASEB_PORTIN) & (1<<ENC2_PHASEB_PIN)) == 0) )
00171     {
00172         EncoderState[2].position++;
00173     }
00174     else
00175     {
00176         EncoderState[2].position--;
00177     }
00178 }
00179 #endif
00180
00181 #ifdef ENC3_SIGNAL
00182
00183 SIGNAL(ENC3_SIGNAL)
00184 {
00185     // encoder has generated a pulse
00186     // check the relative phase of the input channels
00187     // and update position accordingly
00188     if( ((inb(ENC3_PHASEA_PORTIN) & (1<<ENC3_PHASEA_PIN)) == 0) ^
00189         ((inb(ENC3_PHASEB_PORTIN) & (1<<ENC3_PHASEB_PIN)) == 0) )
00190     {
00191         EncoderState[3].position++;
00192     }
00193     else
00194     {
00195         EncoderState[3].position--;
00196     }
00197 }
00198 #endif
```

9.17 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

9.17.1 Detailed Description

Quadrature Encoder reader/driver.

Definition in file encoder.h.

9.18 encoder.h

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

```
00055 //  the rising edge of PhaseA.  When we travel backwards in time (right->left),
00056 //  PhaseB is always high (logic 1) at the rising edge of PhaseA.  Note that
00057 //  traveling forward or backwards in time is the same thing as rotating
00058 //  forwards or bardwards. Thus, if PhaseA is our counter, PhaseB indicates
00059 //  direction.
00060 //
00061 //  Here is an example waveform from a quadrature encoder:
00062 /*
00063 //                /---\   /---\   /---\   /---\   /---\   /---\
00064 //  Phase A:      |   |   |   |   |   |   |   |   |   |   |   |
00065 //             ---/   \---/   \---/   \---/   \---/   \---/   \-
00066 //             -\   /---\   /---\   /---\   /---\   /---\   /---
00067 //  Phase B:    |   |   |   |   |   |   |   |   |   |   |   |
00068 //              \---/   \---/   \---/   \---/   \---/   \---/
00069 //  Time:    <--------------------------------------------------->
00070 //  Rotate FWD: >--------------------------------------------->
00071 //  Rotate REV: <---------------------------------------------<
00072 */
00073 //  To keep track of the encoder position in software, we connect PhaseA to an
00074 //  external processor interrupt line, and PhaseB to any I/O pin.  We set up
00075 //  the external interrupt to trigger whenever PhaseA produces a rising edge.
00076 //  When a rising edge is detected, our interrupt handler function is executed.
00077 //  Inside the handler function, we quickly check the PhaseB line to see if it
00078 //  is high or low.  If it is high, we increment the encoder's position
00079 //  counter, otherwise we decrement it.  The encoder position counter can be
00080 //  read at any time to find out the current position.
00081 //
00082 //
00083 // NOTE: This code is currently below version 1.0, and therefore is considered
00084 // to be lacking in some functionality or documentation, or may not be fully
00085 // tested.  Nonetheless, you can expect most functions to work.
00086 //
00087 // This code is distributed under the GNU Public License
00088 //    which can be found at http://www.gnu.org/licenses/gpl.txt
00089 //
00090 //*****************************************************************************
00091
00092 #ifndef ENCODER_H
00093 #define ENCODER_H
00094
00095 #include "global.h"
00096
00097 // include encoder configuration file
00098 #include "encoderconf.h"
00099
00100 #include <stdint.h>
00101
00102 // constants/macros/typdefs
00103
00104 // defines for processor compatibility
00105 // chose proper Interrupt Mask (IMSK)
00106 #ifdef EIMSK
00107   #define IMSK  EIMSK // for processors mega128, mega64
00108 #else
00109   #define IMSK  GIMSK // for other processors 90s8515, mega163, etc
00110 #endif
00111
00112
00114 //   stores the position and other information from each encoder
00115 typedef struct struct_EncoderState
00116 {
00117   uint16_t position;
00118 //  s32 velocity;       ///< velocity
00119 } EncoderStateType;
00120
00121
00122 // functions
00123
00125 //    Run this init routine once before using any other encoder function.
00126 inline void encoderInit(void);
00127
00129 uint16_t encoderGetPosition(uint8_t encoderNum);
00130
00132 void encoderSetPosition(uint8_t encoderNum, uint16_t position
      );
00133
00134 #endif
```

9.19 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

9.19.1 Detailed Description

Quadrature Encoder driver configuration.

Definition in file encoderconf.h.

9.20 encoderconf.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00018 //*****************************************************************************
00019 //
00020 // File Name  : 'encoderconf.h'
00021 // Title    : Quadrature Encoder driver configuration
00022 // Author   : Pascal Stang - Copyright (C) 2003-2004
00023 // Created    : 2003.01.26
00024 // Revised    : 2004.06.25
00025 // Version    : 0.2
00026 // Target MCU : Atmel AVR Series
00027 // Editor Tabs  : 4
00028 //
00029 // The default number of encoders supported is 2 because most AVR processors
00030 // have two external interrupts.  To use more or fewer encoders, you must do
00031 // four things:
00032 //
00033 //  1. Use a processor with at least as many external interrutps as number of
00034 //    encoders you want to have.
00035 //  2. Set NUM_ENCODERS to the number of encoders you will use.
00036 //  3. Comment/Uncomment the proper ENCx_SIGNAL defines for your encoders
00037 //    (the encoders must be used sequentially, 0 then 1 then 2 then 3)
00038 //  4. Configure the various defines so that they match your processor and
00039 //    specific hardware.  The notes below may help.
00040 //
00041 //
00042 // -------------------- NOTES --------------------
00043 // The external interrupt pins are mapped as follows on most AVR processors:
00044 // (90s8515, mega161, mega163, mega323, mega16, mega32, etc)
00045 //
00046 // INT0 -> PD2 (PORTD, pin 2)
00047 // INT1 -> PD3 (PORTD, pin 3)
00048 //
00049 // The external interrupt pins on the processors mega128 and mega64 are:
00050 //
00051 // INT0 -> PD0 (PORTD, pin 0)
00052 // INT1 -> PD1 (PORTD, pin 1)
00053 // INT2 -> PD2 (PORTD, pin 2)
00054 // INT3 -> PD3 (PORTD, pin 3)
00055 // INT4 -> PE4 (PORTE, pin 4)
00056 // INT5 -> PE5 (PORTE, pin 5)
00057 // INT6 -> PE6 (PORTE, pin 6)
00058 // INT7 -> PE7 (PORTE, pin 7)
00059 //
00060 // This code is distributed under the GNU Public License
00061 //    which can be found at http://www.gnu.org/licenses/gpl.txt
00062 //
00063 //*****************************************************************************
00064
00065 #ifndef ENCODERCONF_H
00066 #define ENCODERCONF_H
00067
00068 // constants/macros/typdefs
00069
00070 // defines for processor compatibility
```

```
00071 // quick compatiblity for mega128, mega64
00072 //#ifndef MCUCR
00073 //  #define MCUCR EICRA
00074 //#endif
00075
00076 // Set the total number of encoders you wish to support
00077 #define NUM_ENCODERS        2
00078
00079
00080 // -------------------- Encoder 0 connections --------------------
00081 // Phase A quadrature encoder output should connect to this interrupt line:
00082 // *** NOTE: the choice of interrupt PORT, DDR, and PIN must match the external
00083 // interrupt you are using on your processor.  Consult the External Interrupts
00084 // section of your processor's datasheet for more information.
00085
00086 // Interrupt Configuration
00087 #define ENC0_SIGNAL         SIG_INTERRUPT0  // Interrupt signal name
00088 #define ENC0_INT          INT0  // matching INTx bit in GIMSK/EIMSK
00089 #define ENC0_ICR          MCUCR // matching Int. Config Register (MCUCR,EICRA/
      B)
00090 #define ENC0_ISCX0          ISC00 // matching Interrupt Sense Config bit0
00091 #define ENC0_ISCX1          ISC01 // matching Interrupt Sense Config bit1
00092 // PhaseA Port/Pin Configuration
00093 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00094 #define ENC0_PHASEA_PORT      PORTD // PhaseA port register
00095 #define ENC0_PHASEA_DDR       DDRD  // PhaseA port direction register
00096 #define ENC0_PHASEA_PORTIN      PIND  // PhaseA port input register
00097 #define ENC0_PHASEA_PIN       PD2   // PhaseA port pin
00098 // Phase B quadrature encoder output should connect to this direction line:
00099 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00100 //#define ENC0_PHASEB_PORT      PORTC // PhaseB port register
00101 //#define ENC0_PHASEB_DDR       DDRC  // PhaseB port direction register
00102 //#define ENC0_PHASEB_PORTIN      PINC  // PhaseB port input register
00103 //#define ENC0_PHASEB_PIN       PC0   // PhaseB port pin
00104
00105
00106 // -------------------- Encoder 1 connections --------------------
00107 // Phase A quadrature encoder output should connect to this interrupt line:
00108 // *** NOTE: the choice of interrupt pin and port must match the external
00109 // interrupt you are using on your processor.  Consult the External Interrupts
00110 // section of your processor's datasheet for more information.
00111
00112 // Interrupt Configuration
00113 #define ENC1_SIGNAL         SIG_INTERRUPT1  // Interrupt signal name
00114 #define ENC1_INT          INT1  // matching INTx bit in GIMSK/EIMSK
00115 #define ENC1_ICR          MCUCR // matching Int. Config Register (MCUCR,EICRA/
      B)
00116 #define ENC1_ISCX0          ISC10 // matching Interrupt Sense Config bit0
00117 #define ENC1_ISCX1          ISC11 // matching Interrupt Sense Config bit1
00118 // PhaseA Port/Pin Configuration
00119 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00120 #define ENC1_PHASEA_PORT      PORTD // PhaseA port register
00121 #define ENC1_PHASEA_PORTIN      PIND  // PhaseA port input register
00122 #define ENC1_PHASEA_DDR       DDRD  // PhaseA port direction register
00123 #define ENC1_PHASEA_PIN       PD3   // PhaseA port pin
00124 // Phase B quadrature encoder output should connect to this direction line:
00125 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00126 //#define ENC1_PHASEB_PORT      PORTC // PhaseB port register
00127 //#define ENC1_PHASEB_DDR       DDRC  // PhaseB port direction register
00128 //#define ENC1_PHASEB_PORTIN      PINC  // PhaseB port input register
00129 //#define ENC1_PHASEB_PIN       PC1   // PhaseB port pin
00130
00131
00132 // -------------------- Encoder 2 connections --------------------
00133 // Phase A quadrature encoder output should connect to this interrupt line:
00134 // *** NOTE: the choice of interrupt pin and port must match the external
00135 // interrupt you are using on your processor.  Consult the External Interrupts
00136 // section of your processor's datasheet for more information.
00137
00138 // Interrupt Configuration
00139 //#define ENC2_SIGNAL         SIG_INTERRUPT6  // Interrupt signal name
00140 #define ENC2_INT          INT6  // matching INTx bit in GIMSK/EIMSK
00141 #define ENC2_ICR          EICRB // matching Int. Config Register (MCUCR,EICRA/
      B)
00142 #define ENC2_ISCX0          ISC60 // matching Interrupt Sense Config bit0
00143 #define ENC2_ISCX1          ISC61 // matching Interrupt Sense Config bit1
00144 // PhaseA Port/Pin Configuration
00145 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00146 #define ENC2_PHASEA_PORT      PORTE // PhaseA port register
00147 #define ENC2_PHASEA_PORTIN      PINE  // PhaseA port input register
00148 #define ENC2_PHASEA_DDR       DDRE  // PhaseA port direction register
```

```
00149 #define ENC2_PHASEA_PIN       PE6   // PhaseA port pin
00150 // Phase B quadrature encoder output should connect to this direction line:
00151 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00152 #define ENC2_PHASEB_PORT      PORTC // PhaseB port register
00153 #define ENC2_PHASEB_DDR       DDRC  // PhaseB port direction register
00154 #define ENC2_PHASEB_PORTIN      PINC  // PhaseB port input register
00155 #define ENC2_PHASEB_PIN       PC2   // PhaseB port pin
00156
00157
00158 // -------------------- Encoder 3 connections --------------------
00159 // Phase A quadrature encoder output should connect to this interrupt line:
00160 // *** NOTE: the choice of interrupt pin and port must match the external
00161 // interrupt you are using on your processor.  Consult the External Interrupts
00162 // section of your processor's datasheet for more information.
00163
00164 // Interrupt Configuration
00165 //#define ENC3_SIGNAL         SIG_INTERRUPT7  // Interrupt signal name
00166 #define ENC3_INT          INT7  // matching INTx bit in GIMSK/EIMSK
00167 #define ENC3_ICR          EICRB // matching Int. Config Register (MCUCR,EICRA/
      B)
00168 #define ENC3_ISCX0          ISC70 // matching Interrupt Sense Config bit0
00169 #define ENC3_ISCX1          ISC71 // matching Interrupt Sense Config bit1
00170 // PhaseA Port/Pin Configuration
00171 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00172 #define ENC3_PHASEA_PORT      PORTE // PhaseA port register
00173 #define ENC3_PHASEA_PORTIN      PINE  // PhaseA port input register
00174 #define ENC3_PHASEA_DDR       DDRE  // PhaseA port direction register
00175 #define ENC3_PHASEA_PIN       PE7   // PhaseA port pin
00176 // Phase B quadrature encoder output should connect to this direction line:
00177 // *** PORTx, DDRx, PINx, and Pxn should all have the same letter for "x" ***
00178 #define ENC3_PHASEB_PORT      PORTC // PhaseB port register
00179 #define ENC3_PHASEB_DDR       DDRC  // PhaseB port direction register
00180 #define ENC3_PHASEB_PORTIN      PINC  // PhaseB port input register
00181 #define ENC3_PHASEB_PIN       PC3   // PhaseB port pin
00182
00183 #endif
```

9.21 global.h File Reference

Global project include file (required by AVRLIB)

```
#include "avrlibdefs.h"
#include "avrlibtypes.h"
```

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

Macros

- #define F_CPU 16000000
- #define CYCLES_PER_US ((F_CPU+500000)/1000000)

9.21.1 Detailed Description

Global project include file (required by AVRLIB) A file of the name global.h is required by the AVRLIB and must contain, at a minimum, a defining for F_CPU.

Definition in file global.h.

9.21.2 Macro Definition Documentation

9.21.2.1 #define CYCLES_PER_US ((F_CPU+500000)/1000000)

CPU cycles per microsecond. (macro required by AVRLIB)

Definition at line 35 of file global.h.

9.21.2.2 #define F_CPU 16000000

CPU clock speed [Hz]. (macro required by AVRLIB)

Definition at line 33 of file global.h.

9.22 global.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00024 #ifndef GLOBAL_H_
00025 #define GLOBAL_H_
00026
00027 // global AVRLIB defines
00028 #include "avrlibdefs.h"
00029 // global AVRLIB types definitions
00030 #include "avrlibtypes.h"
00031
00033 #define F_CPU        16000000
00034
00035 #define CYCLES_PER_US ((F_CPU+500000)/1000000)
00036
00037 #endif /* end ifndef GLOBAL_H_ */
```

9.23 line_tracking.h File Reference

Line detection and PID tracking using CMUcam2.

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

Include dependency graph for line_tracking.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**

9.23.1 Detailed Description

Line detection and PID tracking using CMUcam2.

Definition in file line_tracking.h.

9.24 line_tracking.h

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

```
00045 bool isGoodScan(uint8_t y);
00046 bool isJunctionScan(uint8_t y);
00047 bool mayBeBallScan(uint8_t y);
00048
00049 void printPacket(void);
00050
00051 extern int8_t junctionY;
00052
00053 // Global variables
00054 extern volatile uint8_t lineStats[LINE_STATS_ROWS][LINE_STATS_COLS];
00055 extern volatile bool lineStatsProcessed;
00056
00057 #endif // #ifndef TRACKLINE_H_
```

9.25 motor_control.h File Reference

Interface to PWM motor controller.

```
#include "timer.h"
#include <avr/io.h>
#include <stdint.h>
```

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

Macros

- #define **LEFT_MOTOR** 0
- #define **RIGHT_MOTOR** 1
- #define **BOTH_MOTORS** 2
- #define **LEFT_ENC** 1
- #define **RIGHT_ENC** 0
- #define **MAX_BRAKE** 255
- #define **MC_PORT** PORTA
- #define **FORWARD_LEFT** 7
- #define **FORWARD_RIGHT** 5
- #define **REVERSE_LEFT** 6
- #define **REVERSE_RIGHT** 4
- #define **PWM_LEFT** timer1PWMASet
- #define **PWM_RIGHT** timer1PWMBSet

Functions

- void **moveStraight** (int16_t ticks, uint8_t speed)
- void **tractorTurn** (uint8_t speed, int8_t brads)
- void **tankTurn** (uint8_t speed, int8_t brads)
- void **enableMotors** (void)
- void **disableMotors** (void)
- void **forward** (uint8_t motorSelect, uint8_t speed)
- void **reverse** (uint8_t motorSelect, uint8_t speed)
- void **neutral** (void)
- void **brake** (uint8_t motorSelect)
- void **tickWheels** (int16_t leftTicks, int16_t rightTicks, uint8_t speed)

9.25.1 Detailed Description

Interface to PWM motor controller.

Author

Mike Shelley

Pinout: 2,3 Left motor control 6,7 Right motor control 5,4 motor enable, PWM outputs 5 - A, Right 4 - B, Left

Note

: Assumes motor enable is connected to PWM A,B defined in timer.h

Definition in file motor_control.h.

9.26 motor_control.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  */
00031 #ifndef MOTORCNTRL_H_
00032 #define MOTORCNTRL_H_
00033
00034 // AVRLIB
00035 #include "timer.h"
00036
00037 // avr-libc
00038 #include <avr/io.h>
00039 #include <stdint.h>
00040
00041 //Motor selection constants
00042 #define LEFT_MOTOR  0
00043 #define RIGHT_MOTOR 1
00044 #define BOTH_MOTORS 2
00045 #define LEFT_ENC    1
00046 #define RIGHT_ENC   0
00047
00048 //Speed definitions
00049 #define MAX_BRAKE 255
00050
00051 //Pin defines
00052 //defined by the active high pin
00053 #define MC_PORT PORTA
00054 #define FORWARD_LEFT  7
00055 #define FORWARD_RIGHT 5
00056 #define REVERSE_LEFT  6
00057 #define REVERSE_RIGHT 4
00058 #define PWM_LEFT  timer1PWMASet
00059 #define PWM_RIGHT timer1PWMBSet
00060
00061 void moveStraight(int16_t ticks, uint8_t speed);
00062 void tractorTurn(uint8_t speed, int8_t brads);
00063 void tankTurn(uint8_t speed, int8_t brads);
00064 void enableMotors( void );
00065 void disableMotors( void );
00066
00067 //select left, right, or both motors
00068 //sets selected motor(s) to forward at speed
```

```
00069 void forward(uint8_t motorSelect, uint8_t speed);
00070 //select left, right, or both motors
00071 //sets selected motor(s) to reverse at speed
00072 void reverse(uint8_t motorSelect, uint8_t speed);
00073 //disables motor controller
00074 void neutral(void);
00075 //brake selected motor
00076 //full brake assumed
00077 void brake(uint8_t motorSelect);
00078 void tickWheels(int16_t leftTicks, int16_t rightTicks, uint8_t speed);
00079
00080 #endif // #ifndef MOTORCNTRL_H_
```

9.27 node_list.c File Reference

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

Include dependency graph for node_list.c:

Functions

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

9.27.1 Detailed Description

Definition in file node_list.c.

9.28 node_list.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 "node_list.h"
00019
00020 // avr-libc
00021 #include <string.h>
00022
00023 inline bool isJunction(uint8_t nodeNum)
00024 {
00025     return (nodeNum >= JUNCTION_MIN && nodeNum <= JUNCTION_MAX
      );
00026 }
00027
00028 inline bool isBallNode(uint8_t nodeNum)
00029 {
00030     return (nodeNum >= BALL_NODE_MIN && nodeNum <= BALL_NODE_MAX
      );
00031 }
```

```
00032
00033 uint8_t getCostToNode(GraphNodeType *node, uint8_t nodeNum)
00034 {
00035     uint8_t i;
00036     for (i = 0; i < node->numAdjNodes; i++)
00037     {
00038         if (node->adjNodes[i] == nodeNum)
00039         {
00040             return node->adjCosts[i];
00041         }
00042     }
00043     return 0;
00044 }
00045
00046 uint8_t getNodeAtHeading(GraphNodeType *node, int8_t heading)
00047 {
00048     uint8_t i;
00049     for (i = 0; i < node->numAdjNodes; i++)
00050     {
00051         if (node->adjHeadings[i] == heading)
00052         {
00053             return node->adjNodes[i];
00054         }
00055     }
00056     return 0;
00057 }
00058
00059 void getNode(uint8_t nodeNum, GraphNodeType *node)
00060 {
00061     // Bounds check on input node number
00062     if (nodeNum >= NUM_NODES)
00063     {
00064         node->numAdjNodes = 0;
00065         return;
00066      }
00067
00068     switch (nodeNum)
00069     {
00070     case 0:                             // START_NODE
00071         node->numAdjNodes = 1;
00072         node->   adjNodes[0] = 21;
00073         node->   adjCosts[0] = 9;
00074         node->adjHeadings[0] = -64;
00075         break;
00076     case 1:                             // First ball node
00077         node->numAdjNodes = 2;
00078         node->   adjNodes[0] = 21;
00079         node->   adjNodes[1] = 22;
00080         node->   adjCosts[0] = 4;
00081         node->   adjCosts[1] = 4;
00082         node->adjHeadings[0] = -128;
00083         node->adjHeadings[1] = 0;
00084         break;
00085     case 2:
00086         node->numAdjNodes = 2;
00087         node->   adjNodes[0] = 22;
00088         node->   adjNodes[1] = 23;
00089         node->   adjCosts[0] = 2;
00090         node->   adjCosts[1] = 2;
00091         node->adjHeadings[0] = -64;
00092         node->adjHeadings[1] = 64;
00093         break;
00094     case 3:
00095         node->numAdjNodes = 2;
00096         node->   adjNodes[0] = 23;
00097         node->   adjNodes[1] = 24;
00098         node->   adjCosts[0] = 2;
00099         node->   adjCosts[1] = 2;
00100         node->adjHeadings[0] = 0;
00101         node->adjHeadings[1] = -128;
00102         break;
00103     case 4:
00104         node->numAdjNodes = 2;
00105         node->   adjNodes[0] = 24;
00106         node->   adjNodes[1] = 25;
00107         node->   adjCosts[0] = 3;
00108         node->   adjCosts[1] = 3;
00109         node->adjHeadings[0] = -64;
00110         node->adjHeadings[1] = 64;
00111         break;
00112     case 5:
```

```
00113         node->numAdjNodes = 2;
00114         node->   adjNodes[0] = 26;
00115         node->   adjNodes[1] = 41;
00116         node->   adjCosts[0] = 2;
00117         node->   adjCosts[1] = 2;
00118         node->adjHeadings[0] = -64;
00119         node->adjHeadings[1] = 64;
00120         break;
00121     case 6:
00122         node->numAdjNodes = 2;
00123         node->   adjNodes[0] = 27;
00124         node->   adjNodes[1] = 28;
00125         node->   adjCosts[0] = 2;
00126         node->   adjCosts[1] = 2;
00127         node->adjHeadings[0] = 64;
00128         node->adjHeadings[1] = -64;
00129         break;
00130     case 7:
00131         node->numAdjNodes = 2;
00132         node->   adjNodes[0] = 27;
00133         node->   adjNodes[1] = 32;
00134         node->   adjCosts[0] = 3;
00135         node->   adjCosts[1] = 3;
00136         node->adjHeadings[0] = -128;
00137         node->adjHeadings[1] = 0;
00138         break;
00139     case 8:
00140         node->numAdjNodes = 2;
00141         node->   adjNodes[0] = 28;
00142         node->   adjNodes[1] = 30;
00143         node->   adjCosts[0] = 3;
00144         node->   adjCosts[1] = 3;
00145         node->adjHeadings[0] = -128;
00146         node->adjHeadings[1] = 0;
00147         break;
00148     case 9:
00149         node->numAdjNodes = 2;
00150         node->   adjNodes[0] = 22;
00151         node->   adjNodes[1] = 29;
00152         node->   adjCosts[0] = 3;
00153         node->   adjCosts[1] = 3;
00154         node->adjHeadings[0] = -128;
00155         node->adjHeadings[1] = 0;
00156         break;
00157     case 10:
00158         node->numAdjNodes = 2;
00159         node->   adjNodes[0] = 29;
00160         node->   adjNodes[1] = 30;
00161         node->   adjCosts[0] = 3;
00162         node->   adjCosts[1] = 3;
00163         node->adjHeadings[0] = -64;
00164         node->adjHeadings[1] = 64;
00165         break;
00166     case 11:
00167         node->numAdjNodes = 2;
00168         node->   adjNodes[0] = 12;
00169         node->   adjNodes[1] = 29;
00170         node->   adjCosts[0] = 4;
00171         node->   adjCosts[1] = 2;
00172         node->adjHeadings[0] = 0;
00173         node->adjHeadings[1] = -128;
00174         break;
00175     case 12:
00176         node->numAdjNodes = 2;
00177         node->   adjNodes[0] = 11;
00178         node->   adjNodes[1] = 33;
00179         node->   adjCosts[0] = 4;
00180         node->   adjCosts[1] = 2;
00181         node->adjHeadings[0] = -128;
00182         node->adjHeadings[1] = 0;
00183         break;
00184     case 13:
00185         node->numAdjNodes = 2;
00186         node->   adjNodes[0] = 33;
00187         node->   adjNodes[1] = 34;
00188         node->   adjCosts[0] = 2;
00189         node->   adjCosts[1] = 1;
00190         node->adjHeadings[0] = -64;
00191         node->adjHeadings[1] = 64;
00192         break;
00193     case 14:
```

```
00194         node->numAdjNodes = 2;
00195         node->   adjNodes[0] = 15;
00196         node->   adjNodes[1] = 34;
00197         node->   adjCosts[0] = 3;
00198         node->   adjCosts[1] = 4;
00199         node->adjHeadings[0] = S_EAST;
00200         node->adjHeadings[1] = N_WEST;
00201         break;
00202     case 15:
00203         node->numAdjNodes = 2;
00204         node->   adjNodes[0] = 14;
00205         node->   adjNodes[1] = 31;
00206         node->   adjCosts[0] = 3;
00207         node->   adjCosts[1] = 4;
00208         node->adjHeadings[0] = N_WEST;
00209         node->adjHeadings[1] = S_EAST;
00210         break;
00211     case 16:
00212         node->numAdjNodes = 2;
00213         node->   adjNodes[0] = 32;
00214         node->   adjNodes[1] = 40;
00215         node->   adjCosts[0] = 2;
00216         node->   adjCosts[1] = 2;
00217         node->adjHeadings[0] = -64;
00218         node->adjHeadings[1] = 64;
00219         break;
00220     case 17:
00221         node->numAdjNodes = 2;
00222         node->   adjNodes[0] = 34;
00223         node->   adjNodes[1] = 35;
00224         node->   adjCosts[0] = 3;
00225         node->   adjCosts[1] = 3;
00226         node->adjHeadings[0] = -64;
00227         node->adjHeadings[1] = 64;
00228         break;
00229     case 18:
00230         node->numAdjNodes = 2;
00231         node->   adjNodes[0] = 39;
00232         node->   adjNodes[1] = 40;
00233         node->   adjCosts[0] = 2;
00234         node->   adjCosts[1] = 2;
00235         node->adjHeadings[0] = 0;
00236         node->adjHeadings[1] = -128;
00237         break;
00238     case 19:
00239         node->numAdjNodes = 2;
00240         node->   adjNodes[0] = 20;
00241         node->   adjNodes[1] = 40;
00242         node->   adjCosts[0] = 4;
00243         node->   adjCosts[1] = 2;
00244         node->adjHeadings[0] = -128;
00245         node->adjHeadings[1] = 0;
00246         break;
00247     case 20:
00248         node->numAdjNodes = 2;
00249         node->   adjNodes[0] = 19;
00250         node->   adjNodes[1] = 41;
00251         node->   adjCosts[0] = 4;
00252         node->   adjCosts[1] = 2;
00253         node->adjHeadings[0] = 0;
00254         node->adjHeadings[1] = -128;
00255         break;
00256     case 21:                          // First Junction Node
00257         node->numAdjNodes = 2;
00258         node->   adjNodes[0] = 0;
00259         node->   adjNodes[1] = 1;
00260         node->   adjCosts[0] = 9;
00261         node->   adjCosts[1] = 4;
00262         node->adjHeadings[0] = 64;
00263         node->adjHeadings[1] = 0;
00264         break;
00265     case 22:
00266         node->numAdjNodes = 3;
00267         node->   adjNodes[0] = 1;
00268         node->   adjNodes[1] = 2;
00269         node->   adjNodes[2] = 9;
00270         node->   adjCosts[0] = 4;
00271         node->   adjCosts[1] = 2;
00272         node->   adjCosts[2] = 3;
00273         node->adjHeadings[0] = -128;
00274         node->adjHeadings[1] = 64;
```

```
00275         node->adjHeadings[2] = 0;
00276         break;
00277     case 23:
00278         node->numAdjNodes = 3;
00279         node->   adjNodes[0] = 2;
00280         node->   adjNodes[1] = 3;
00281         node->   adjNodes[2] = 28;
00282         node->   adjCosts[0] = 2;
00283         node->   adjCosts[1] = 2;
00284         node->   adjCosts[2] = 2;
00285         node->adjHeadings[0] = -64;
00286         node->adjHeadings[1] = -128;
00287         node->adjHeadings[2] = 64;
00288         break;
00289     case 24:                            // BONUS_BALL_1
00290         node->numAdjNodes = 2;
00291         node->   adjNodes[0] = 3;
00292         node->   adjNodes[1] = 4;
00293         node->   adjCosts[0] = 2;
00294         node->   adjCosts[1] = 3;
00295         node->adjHeadings[0] = 0;
00296         node->adjHeadings[1] = 64;
00297         break;
00298     case 25:
00299         node->numAdjNodes = 2;
00300         node->   adjNodes[0] = 4;
00301         node->   adjNodes[1] = 26;
00302         node->   adjCosts[0] = 3;
00303         node->   adjCosts[1] = 2;
00304         node->adjHeadings[0] = -64;
00305         node->adjHeadings[1] = 0;
00306         break;
00307     case 26:
00308         node->numAdjNodes = 3;
00309         node->   adjNodes[0] = 5;
00310         node->   adjNodes[1] = 25;
00311         node->   adjNodes[2] = 27;
00312         node->   adjCosts[0] = 2;
00313         node->   adjCosts[1] = 2;
00314         node->   adjCosts[2] = 2;
00315         node->adjHeadings[0] = 64;
00316         node->adjHeadings[1] = -128;
00317         node->adjHeadings[2] = 0;
00318         break;
00319     case 27:
00320         node->numAdjNodes = 3;
00321         node->   adjNodes[0] = 6;
00322         node->   adjNodes[1] = 7;
00323         node->   adjNodes[2] = 26;
00324         node->   adjCosts[0] = 2;
00325         node->   adjCosts[1] = 3;
00326         node->   adjCosts[2] = 2;
00327         node->adjHeadings[0] = -64;
00328         node->adjHeadings[1] = 0;
00329         node->adjHeadings[2] = -128;
00330         break;
00331     case 28:
00332         node->numAdjNodes = 3;
00333         node->   adjNodes[0] = 6;
00334         node->   adjNodes[1] = 8;
00335         node->   adjNodes[2] = 23;
00336         node->   adjCosts[0] = 2;
00337         node->   adjCosts[1] = 3;
00338         node->   adjCosts[2] = 2;
00339         node->adjHeadings[0] = 64;
00340         node->adjHeadings[1] = 0;
00341         node->adjHeadings[2] = -64;
00342         break;
00343     case 29:
00344         node->numAdjNodes = 3;
00345         node->   adjNodes[0] = 9;
00346         node->   adjNodes[1] = 10;
00347         node->   adjNodes[2] = 11;
00348         node->   adjCosts[0] = 3;
00349         node->   adjCosts[1] = 3;
00350         node->   adjCosts[2] = 2;
00351         node->adjHeadings[0] = -128;
00352         node->adjHeadings[1] = 64;
00353         node->adjHeadings[2] = 0;
00354         break;
00355     case 30:                            // BONUS_BALL_2
```

```
00356         node->numAdjNodes = 3;
00357         node->   adjNodes[0] = 8;
00358         node->   adjNodes[1] = 10;
00359         node->   adjNodes[2] = 31;
00360         node->   adjCosts[0] = 3;
00361         node->   adjCosts[1] = 3;
00362         node->   adjCosts[2] = 3;
00363         node->adjHeadings[0] = -128;
00364         node->adjHeadings[1] = -64;
00365         node->adjHeadings[2] = 64;
00366         break;
00367     case 31:
00368         node->numAdjNodes = 3;
00369         node->   adjNodes[0] = 15;
00370         node->   adjNodes[1] = 30;
00371         node->   adjNodes[2] = 32;
00372         node->   adjCosts[0] = 4;
00373         node->   adjCosts[1] = 3;
00374         node->   adjCosts[2] = 1;
00375         node->adjHeadings[0] = N_WEST;
00376         node->adjHeadings[1] = -64;
00377         node->adjHeadings[2] = 64;
00378         break;
00379     case 32:
00380         node->numAdjNodes = 3;
00381         node->   adjNodes[0] = 7;
00382         node->   adjNodes[1] = 16;
00383         node->   adjNodes[2] = 31;
00384         node->   adjCosts[0] = 3;
00385         node->   adjCosts[1] = 2;
00386         node->   adjCosts[2] = 1;
00387         node->adjHeadings[0] = -128;
00388         node->adjHeadings[1] = 64;
00389         node->adjHeadings[2] = -64;
00390         break;
00391     case 33:
00392         node->numAdjNodes = 2;
00393         node->   adjNodes[0] = 12;
00394         node->   adjNodes[1] = 13;
00395         node->   adjCosts[0] = 2;
00396         node->   adjCosts[1] = 2;
00397         node->adjHeadings[0] = -128;
00398         node->adjHeadings[1] = 64;
00399         break;
00400     case 34:
00401         node->numAdjNodes = 3;
00402         node->   adjNodes[0] = 13;
00403         node->   adjNodes[1] = 14;
00404         node->   adjNodes[2] = 17;
00405         node->   adjCosts[0] = 1;
00406         node->   adjCosts[1] = 4;
00407         node->   adjCosts[2] = 3;
00408         node->adjHeadings[0] = -64;
00409         node->adjHeadings[1] = S_EAST;
00410         node->adjHeadings[2] = 64;
00411         break;
00412     case 35:
00413         node->numAdjNodes = 2;
00414         node->   adjNodes[0] = 17;
00415         node->   adjNodes[1] = 36;
00416         node->   adjCosts[0] = 3;
00417         node->   adjCosts[1] = 2;
00418         node->adjHeadings[0] = -64;
00419         node->adjHeadings[1] = -128;
00420         break;
00421     case 36:
00422         node->numAdjNodes = 2;
00423         node->   adjNodes[0] = 35;
00424         node->   adjNodes[1] = 37;
00425         node->   adjCosts[0] = 2;
00426         node->   adjCosts[1] = 2;
00427         node->adjHeadings[0] = 0;
00428         node->adjHeadings[1] = 64;
00429         break;
00430     case 37:
00431         node->numAdjNodes = 2;
00432         node->   adjNodes[0] = 36;
00433         node->   adjNodes[1] = 38;
00434         node->   adjCosts[0] = 2;
00435         node->   adjCosts[1] = 2;
00436         node->adjHeadings[0] = -64;
```

```
00437         node->adjHeadings[1] = -128;
00438         break;
00439     case 38:
00440         node->numAdjNodes = 2;
00441         node->   adjNodes[0] = 37;
00442         node->   adjNodes[1] = 39;
00443         node->   adjCosts[0] = 2;
00444         node->   adjCosts[1] = 2;
00445         node->adjHeadings[0] = 0;
00446         node->adjHeadings[1] = 64;
00447         break;
00448     case 39:
00449         node->numAdjNodes = 2;
00450         node->   adjNodes[0] = 18;
00451         node->   adjNodes[1] = 38;
00452         node->   adjCosts[0] = 2;
00453         node->   adjCosts[1] = 2;
00454         node->adjHeadings[0] = -128;
00455         node->adjHeadings[1] = -64;
00456         break;
00457     case 40:
00458         node->numAdjNodes = 3;
00459         node->   adjNodes[0] = 16;
00460         node->   adjNodes[1] = 18;
00461         node->   adjNodes[2] = 19;
00462         node->   adjCosts[0] = 2;
00463         node->   adjCosts[1] = 2;
00464         node->   adjCosts[2] = 2;
00465         node->adjHeadings[0] = -64;
00466         node->adjHeadings[1] = 0;
00467         node->adjHeadings[2] = -128;
00468         break;
00469     case 41:
00470         node->numAdjNodes = 3;
00471         node->   adjNodes[0] = 5;
00472         node->   adjNodes[1] = 20;
00473         node->   adjNodes[2] = 42;
00474         node->   adjCosts[0] = 2;
00475         node->   adjCosts[1] = 2;
00476         node->   adjCosts[2] = 5;
00477         node->adjHeadings[0] = -64;
00478         node->adjHeadings[1] = 0;
00479         node->adjHeadings[2] = -128;
00480         break;
00481     case 42:                          // STOP_NODE
00482         node->numAdjNodes = 1;
00483         node->   adjNodes[0] = 41;
00484         node->   adjCosts[0] = 5;
00485         node->adjHeadings[0] = 0;
00486         break;
00487     }
00488 }
```

9.29 node_list.h File Reference

Course defined by a connected grid of nodes.

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

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

Data Structures

- struct GraphNode

  Definition of each node (vertex) in the course map node.

Macros

- #define NUM_NODES 43

- #define START_NODE 0
- #define START_HEADING -64
- #define STOP_NODE 42
- #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 GraphNode GraphNodeType

  Definition of each node (vertex) in the course map node.

Functions

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

9.29.1 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)

Definition in file node_list.h.

9.29.2 Macro Definition Documentation

9.29.2.1 #define BALL_NODE_MAX 20

End of node number range used for ball nodes

Definition at line 51 of file node_list.h.

9.29.2.2 #define BALL_NODE_MIN 1

Beginning of node number range used for ball nodes

Definition at line 49 of file node_list.h.

9.29.2.3 #define BB1_HEADING 32

The heading required to pickup bonus ball 1

Definition at line 75 of file node_list.h.

9.29.2.4 #define BB2_HEADING -96

The heading required to pickup bonus ball 2

Definition at line 77 of file node_list.h.

9.29.2.5 #define BONUS_BALL_1 24

Node number of bonus ball 1

Definition at line 68 of file node_list.h.

9.29.2.6 #define BONUS_BALL_2 30

Node number of bonus ball 2

Definition at line 70 of file node_list.h.

9.29.2.7 #define JUNCTION_MAX 41

End of node number range used for junction nodes

Definition at line 55 of file node_list.h.

9.29.2.8 #define JUNCTION_MIN 21

Beginning of node number range used for junction nodes

Definition at line 53 of file node_list.h.

9.29.2.9 #define MAX_ADJ_NODES 3

Maximum number nodes that can be adjacent to one node

Definition at line 61 of file node_list.h.

9.29.2.10 #define N_WEST -41

Direction of north west in binary radians (brads)

Definition at line 63 of file node_list.h.

9.29.2.11 #define NUM_FIXED_GOALS 3

The number of goals known a priori (nest sensor and two bonus balls)

Definition at line 80 of file node_list.h.

9.29.2.12 #define NUM_GOALS NUM_FIXED_GOALS + NUM_RANDOM_GOALS

Total number of goals

Definition at line 84 of file node_list.h.

9.29.2.13 #define NUM_NODES 43

Total number of nodes in the arena map

Definition at line 39 of file node_list.h.

9.29.2.14 #define NUM_RANDOM_GOALS 3

The number of goals with unknown location at the start of a run (ground balls)

Definition at line 82 of file node_list.h.

9.29.2.15 #define S_EAST 87

Convenience macor for direction of south east in (brads)

Definition at line 65 of file node_list.h.

9.29.2.16 #define SENSOR_NODE 37

Node number of the nest release sensor

Definition at line 72 of file node_list.h.

9.29.2.17 #define START_HEADING -64

Heading of the robot at the beginning of the course

Definition at line 44 of file node_list.h.

9.29.2.18 #define START_NODE 0

Beginning of the course

Definition at line 42 of file node_list.h.

9.29.2.19 #define STOP_NODE 42

End of the course

Definition at line 46 of file node_list.h.

9.29.3 Typedef Documentation

9.29.3.1 typedef struct GraphNode GraphNodeType

Definition of each node (vertex) in the course map node.

Defines the directions and distances to adjacent nodes.

9.30 node_list.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  */
00032 #ifndef NODELIST_H_
00033 #define NODELIST_H_
00034
00035 #include <stdint.h>
00036 #include <stdbool.h>
00037
00039 #define NUM_NODES       43
00040
00042 #define START_NODE      0
00043
00044 #define START_HEADING   -64
00045
00046 #define STOP_NODE       42
00047
00049 #define BALL_NODE_MIN   1
00050
00051 #define BALL_NODE_MAX   20
00052
00053 #define JUNCTION_MIN    21
00054
00055 #define JUNCTION_MAX    41
00056
00057 #define NUM_BALL_NODES  (BALL_NODE_MAX - BALL_NODE_MAX + 1)
00058
00059
00061 #define MAX_ADJ_NODES   3
00062
00063 #define N_WEST          -41
00064
00065 #define S_EAST          87
00066
00068 #define BONUS_BALL_1    24
00069
00070 #define BONUS_BALL_2    30
00071
00072 #define SENSOR_NODE     37
00073
00075 #define BB1_HEADING     32
00076
00077 #define BB2_HEADING     -96
00078
00080 #define NUM_FIXED_GOALS    3
00081
00082 #define NUM_RANDOM_GOALS   3
00083
00084 #define NUM_GOALS          NUM_FIXED_GOALS + NUM_RANDOM_GOALS
00085
00091 typedef struct GraphNode
00092 {
00096     uint8_t numAdjNodes;
00100     uint8_t adjNodes[MAX_ADJ_NODES];
00104     uint8_t adjCosts[MAX_ADJ_NODES];
00108     int8_t adjHeadings[MAX_ADJ_NODES];
00109 } GraphNodeType;
00110
00111
00112 inline bool isJunction( uint8_t nodeNum );
00113 uint8_t getCostToNode(GraphNodeType *node, uint8_t nodeNum);
00114 uint8_t getNodeAtHeading(GraphNodeType *node, int8_t heading);
00115 inline bool isBallNode( uint8_t nodeNum );
00116 void getNode( uint8_t nodeNum, GraphNodeType *node );
00117
```

```
00118
00119 #endif // #ifndef NODELIST_H_
```

9.31 permutation.h File Reference

Iterative (non-recursive!) permutation generator.

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

Include dependency graph for permutation.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.

9.31.1 Detailed Description

Iterative (non-recursive!) permutation generator.

Definition in file permutation.h.

9.31.2 Function Documentation

9.31.2.1 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 permutation.c.

9.32 permutation.h

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

9.33 robot_control.c File Reference

```
#include "robot_control.h"
#include "line_tracking.h"
#include "ball_tracking.h"
#include "path_planning.h"
#include "motor_control.h"
#include "camera.h"
#include "servos.h"
#include "buttons.h"
#include "node_list.h"
#include "tether_ui.h"
#include "tweak_data.h"
#include "lcd_16x2.h"
#include "utility.h"
#include <string.h>
```

Include dependency graph for robot_control.c:

Macros

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

Functions

- void runRoborodentiaCourse (void)

  Run the Roborodentia course from start to finish.
- void bonusBallPickUpManeuver (int8_t bbHeading, int8_t nextHeading)

  Orients Caddy to grab a bonus ball, grabs the ball, and reorients for the next node.
- void moveToJunction (uint8_t numJunctions, bool justTurned)

Move to next junction in pathList.

- void nestSequence (void)

  Sequence of actions to peform once the nest is reached.

- void **junctionCode** (void)

Variables

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

9.33.1 Detailed Description

Definition in file robot_control.c.

9.33.2 Function Documentation

9.33.2.1 void bonusBallPickUpManeuver (int8_t *bbHeading*, int8_t *nextHeading*) [inline]

Orients Caddy to grab a bonus ball, grabs the ball, and reorients for the next node.

Parameters

in	*bbHeading*	Heading Caddy must have for bonus ball pickup
in	*nextHeading*	Heading Caddy must have after bonus ball pickup

Definition at line 266 of file robot_control.c.

9.33.2.2 void nestSequence (void)

Sequence of actions to peform once the nest is reached.

Main purpose is to release the balls from the hopper.

Definition at line 496 of file robot_control.c.

9.33.2.3 void runRoborodentiaCourse (void) [inline]

Run the Roborodentia course from start to finish.

Returns once all balls have been collected and placed in the nest.

Definition at line 79 of file robot_control.c.

9.34 robot_control.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 "robot_control.h"
00019 #include "line_tracking.h"
00020 #include "ball_tracking.h"
00021 #include "path_planning.h"
00022 #include "motor_control.h"
00023 #include "camera.h"
00024 #include "servos.h"
00025 #include "buttons.h"
00026 #include "node_list.h"
00027 #include "tether_ui.h"
00028 #include "tweak_data.h"
00029 #include "lcd_16x2.h"
00030 #include "utility.h"
00031
00032 // avr-libc
00033 #include <string.h>
00034
00035 #define BEAM_IGNORE_COUNT      6
00036 #define CORRAL_COUNT           3
00037 #define LIFT_DONE_COUNT        8
00038
00039 //
00040 // Static global variables
00041 //
00042 static bool checkedList[BALL_NODE_MAX];
00043
00044 //
00045 // Global variables
00046 //
00047 uint8_t botNode = START_NODE;
00048 int8_t botHeading = START_HEADING;
00049 uint8_t numUnreachedGoals = NUM_GOALS;
00050
00051 static bool liftDown = false;
00052 static uint8_t nextBallNodeNum = 0;
00053
00057 static void moveServosToStart(void);
00058
00070 static inline bool positionBot(void);
00071
00072 static bool standardBallSearch( void );
00073 static inline bool nodeCode0( void );
00074 static inline bool nodeCode22( void );
00075 static inline bool diagNodeCode(void);
00076 static inline bool nodeCode37( void );
00077 static inline void initBallSeeking(void);
00078
00079 inline void runRoborodentiaCourse(void)
00080 {
00081     bool justTurned = false;
00082     bool firstRun = true;
00083
00084     // Known, fixed goals
00085     addToGoalList(BONUS_BALL_1);
00086     addToGoalList(BONUS_BALL_2);
00087     addToGoalList(SENSOR_NODE);
00088
00089     moveServosToStart();
00090     initBallSeeking();
00091
00092     updatePath();
00093
00094     // run through first leg, skipping positionBot
00095     junctionCode();
00096     moveToJunction(1, justTurned);
00097
00098 #if DEBUGGING
00099     if (lcdMode == NAV_LCD_MODE)
00100     {
00101         lcdWriteStr("                ", 0, 0);
00102         lcdWriteStr("                ", 1, 0);
00103         lcdPrintDecU08(botNode, 1, 0);
00104         lcdPrintDecS08(botHeading, 1, 3);
00105     }
00106 #endif
00107
```

```
00108     // run through arena
00109     while (pathList[pathListIndex + 1] != STOP_NODE)
00110     {
00111         junctionCode();       // ball search, bonous ball pickup, best path
      code
00112
00113         justTurned = positionBot();      // turning, preparing for linetracking
00114
00115         if (firstRun)
00116         {
00117             firstRun = false;
00118             setServo(LIFT, LIFT_OPEN); // Lower lift, on first run, b/c
      skipping seek at node 21
00119             msDelay(30);
00120             nextBallNodeNum = 1;
00121             liftDown = true;
00122         }
00123
00124 #if DEBUGGING
00125         if (lcdMode == NAV_LCD_MODE)
00126         {
00127             lcdPrintDecS08(botHeading, 1, 3);
00128         }
00129 #endif
00130
00131         moveToJunction(1, justTurned);   // linetracking, ground
      ball pickup
00132
00133 #if DEBUGGING
00134         if (lcdMode == NAV_LCD_MODE)
00135         {
00136             lcdPrintDecU08(botNode, 1, 0);
00137         }
00138 #endif
00139     }
00140
00141     if (pathList[pathListIndex + 1] == STOP_NODE)
00142     {
00143         positionBot();
00144         nestSequence();
00145     }
00146 }
00147
00148 inline void initBallSeeking(void)
00149 {
00150     uint8_t i;
00151     for (i = 0; i < ELEMENTSOF(checkedList); ++i)
00152     {
00153         checkedList[i] = false;
00154     }
00155 }
00156
00157 void moveServosToStart(void)
00158 {
00159 #if DEBUGGING
00160     lcdWriteStr("Servos    ", 0, 6);
00161 #endif
00162     setServo(PAN, PAN_CENTER + panOffset);
00163     setServo(TILT, TILT_FORWARD);
00164     setServo(BOOM, BOOM_UP);
00165     setServo(DOOR, DOOR_CLOSED);
00166     myDelay(30);
00167     setServo(LIFT, LIFT_UP);
00168     myDelay(50);
00169     disableServo(BOOM);
00170     disableServo(DOOR);
00171     disableServo(LIFT);
00172 }
00173
00174 inline bool positionBot(void)
00175 {
00176     bool justTurned = true;
00177
00178     int8_t nextHeading = getNextHeading();
00179     int8_t bradsToTurn = nextHeading - botHeading;
00180
00181     // BB PICKUP CHECK
00182     if (botNode == BONUS_BALL_1 && isInGoalList(BONUS_BALL_1
      ))
00183     {
00184         bonusBallPickUpManeuver(BB1_HEADING,
```

```
      nextHeading);
00185         removeFromGoalList(BONUS_BALL_1);
00186     }
00187     else if (botNode == BONUS_BALL_2 && isInGoalList(BONUS_BALL_2
      ))
00188     {
00189         bonusBallPickUpManeuver(BB2_HEADING,
      nextHeading);
00190         removeFromGoalList(BONUS_BALL_2);
00191     }
00192
00193     // TURN/STRAIGHT CHECK
00194     else if (bradsToTurn != 0)
00195     {
00196         int8_t ticksToTurn;
00197         switch ((int8_t) bradsToTurn)
00198         {
00199         case -128: // U-turn
00200             if (botNode == 37)
00201             {
00202                 moveToJunction(1, false);
00203                 tickWheels(20, 20, 255);
00204                 msDelay(0x50);
00205                 moveStraight(-20, 255);
00206             }
00207             //tankTurn(245, -58);
00208             tickWheels(-29, 29, 250);
00209             tankTurn(245, -58);
00210             break;
00211         case -105: // Hard Diagonal
00212             tickWheels(28, 28, 250);     //28
00213             tractorTurn(255, -tempTweak4);
00214             tankTurn(250, -70);          //-80
00215             break;
00216         case 23: // Soft Diagonal
00217             tractorTurn(255, 23);        //23
00218             break;
00219         case -23:
00220             tractorTurn(255, -28);
00221             break;
00222         case 105:
00223             tickWheels(17, 17, 250);
00224             tankTurn(250, 80);           //80
00225             break;
00226         default:
00227             // fixed ticks forward here?
00228
00229             // convert brads to turn to ticks and turn
00230             if (bradsToTurn < 0)
00231             {
00232                 ticksToTurn = bradsToTurn + turnSubtract;
00233             } else
00234             {
00235                 ticksToTurn = bradsToTurn - turnSubtract;
00236             }
00237             tractorTurn(255, ticksToTurn);
00238             break;
00239         }
00240     }
00241     else
00242     {
00243         justTurned = false;
00244     }
00245
00246     if (botNode == SENSOR_NODE)
00247     {
00248         removeFromGoalList(SENSOR_NODE);
00249     }
00250
00251     // update botHeading
00252     botHeading = nextHeading;
00253
00254     // GB PICKUP CHECK: lower lift, if bot knows it will travel over ball
00255     nextBallNodeNum = getNextBallNodeNum();
00256     if (nextBallNodeNum != 0)
00257     {
00258         setServo(LIFT, LIFT_OPEN);
00259         msDelay(30);
00260         liftDown = true;
00261     }
00262
```

```
00263     return justTurned;
00264 }
00265
00266 inline void bonusBallPickUpManeuver(int8_t bbHeading,
      int8_t nextHeading)
00267 {
00268     // move forward (camera will be over junction at this point)
00269     // May or may not need to move foward (requires testing)
00270     //    Some are fine without foward, some seem to need it
00271     //    Right now, only -32 case moves forward
00272
00273     // rotate by (bbHeading - botHeading)
00274     switch ((int8_t) (bbHeading - botHeading))
00275     {
00276     case 96:
00277         // example of 96 brad rotation
00278         tickWheels(28, 0, 255); // allows fluid motion (no overshoot
       correction)
00279         tankTurn(255, 58);      //58
00280         break;
00281     case -96:
00282         tickWheels(0, 28, 255); // allows fluid motion (no overshoot
       correction)
00283         tankTurn(255, -64);     //-64
00284         break;
00285     case -32:
00286         tickWheels(10, 10, 255); //10 Move bot forward a few ticks to make it
       correctly aligned
00287         tickWheels(0, 32, 255);                   //28
00288         break;
00289     default:
00290         fatalError("ERR: bonusBallPi", "ckUpManeuver: 1");
00291         break;
00292     }
00293
00294     // Get in position
00295     // setPose(BB_READY);
00296     setServo(TILT, TILT_BACK);
00297     setServo(LIFT, LIFT_BB_READY);
00298     myDelay(40);
00299
00300     // Drive up to the corner
00301     forward(BOTH_MOTORS, 255);
00302     msDelay(1000);
00303     brake(BOTH_MOTORS);
00304
00305     // Back and and move back into the corner (help align better)
00306     moveStraight(-10, 255);
00307     forward(BOTH_MOTORS, 255);
00308     msDelay(400);
00309     brake(BOTH_MOTORS);
00310
00311     // Execute the pickup sequence
00312     setServo(BOOM, BOOM_BB_GRAB);
00313     myDelay(10);
00314     setServo(TILT, TILT_BB_GRAB);
00315     myDelay(10);
00316     setServo(BOOM, BOOM_UP);
00317     myDelay(10);
00318     setServo(TILT, TILT_VERT + tiltOffset);
00319     myDelay(10);
00320     setServo(LIFT, LIFT_UP);
00321
00322     // Back away from the corner
00323     moveStraight(-18, 255);
00324     setServo(TILT, TILT_FORWARD);
00325
00326     disableServo(BOOM);
00327     disableServo(LIFT);
00328
00329     // Rotate by (nextHeading - bbHeading)
00330     // (This should only be 32, -32, or -96)
00331     switch ((int8_t) (nextHeading - bbHeading))
00332     {
00333     case 32:
00334         tankTurn(250, 32);
00335         break;
00336     case -32:
00337         tankTurn(250, -32);
00338         break;
00339     case -96:
```

```
00340         tankTurn(250, -90);
00341         break;
00342     default:
00343         // Error, this should only be 32, -32, or -96
00344         fatalError("ERR: bonusBallPi", "ckUpManeuver: 2");
00345         break;
00346     }
00347 }
00348
00352 inline void moveToJunction(uint8_t numJunctions, bool justTurned)
00353 {
00354     bool onLine = true;
00355     bool juncApproaching = false;
00356     uint8_t juncCount = 0;
00357
00358     uint8_t ignoreJuncCount;
00359     if (!justTurned)
00360     {
00361         ignoreJuncCount = 3;
00362     } else
00363     {
00364         ignoreJuncCount = 0;
00365     }
00366
00367     uint8_t pickingUp = false;
00368     uint8_t pickingUpCount = 0;
00369
00370     uint8_t ignoreBreakBeamCount = BEAM_IGNORE_COUNT;
00371
00372     trackLineInit();
00373
00374     // Linetrack, until bot is at junction or nest.
00375     // If see ground ball, pickup it up and continue linetracking.
00376     while (onLine)
00377     {
00378         while (lineStatsProcessed) ;
00379
00380         analyzeLineStats();
00381         adjustPWM();
00382
00383         // CURRENT JUNCTION IGNORE
00384         if (ignoreJuncCount > 0 && junctionY == 0)
00385         {
00386             ignoreJuncCount--;
00387         }
00388
00389         // JUNCTION CHECK
00390         if (ignoreJuncCount == 0 && junctionY != 0)
00391         {
00392             if (junctionY < turnPoint)
00393             {
00394                 juncApproaching = true;
00395             } else if (juncApproaching)
00396             {
00397                 juncApproaching = false;
00398                 juncCount++;
00399
00400                 // set botNode to next junction in pathList
00401                 do
00402                 {
00403                     pathListIndex++;
00404                     botNode = pathList[pathListIndex];
00405                 } while (!isJunction(botNode));
00406
00407                 // Break out of line tracking
00408                 if (juncCount >= numJunctions)
00409                 {
00410                     onLine = false;
00411                 }
00412             }
00413         }
00414
00415         // STOP IGNORING BEAM CHECK
00416         if (liftDown && ignoreBreakBeamCount != 0)
00417         {
00418             ignoreBreakBeamCount--;
00419         }
00420
00421         // BEGIN PICKUP CHECK
00422         if (liftDown && ignoreBreakBeamCount == 0 && BREAK_BEAM_TRIGGERED)
00423         {
```

```
00424             cameraStreamingOff();
00425             setServo(LIFT, LIFT_CORRAL); // Perhaps raise it
    slowly if there are pick-up problems
00426             msDelay(30);
00427             trackLineInit();
00428 
00429             liftDown = false;
00430             pickingUp = true;
00431             pickingUpCount = 0;
00432         }
00433 
00434         // COMPLETE/STOP LIFTING CHECK
00435         if (pickingUp)
00436         {
00437             pickingUpCount++;
00438 
00439             if (pickingUpCount == CORRAL_COUNT)
00440             {
00441                 cameraStreamingOff();
00442                 setServo(LIFT, LIFT_UP);
00443                 trackLineInit();
00444             }
00445 
00446             if (pickingUpCount == LIFT_DONE_COUNT)
00447             {
00448                 pickingUp = false;
00449 
00450                 // Set current botNode to node where this ball is
00451                 botNode = nextBallNodeNum;
00452                 removeFromGoalList(nextBallNodeNum);
00453 
00454                 if (nextBallNodeNum == 1) // account for ball not found by
    camera prior to pickup
00455                 {
00456                     adjustNumKnownGoals(1);
00457                 }
00458 
00459                 // Find correct pathListIndex
00460                 while (botNode != pathList[pathListIndex])
00461                 {
00462                     pathListIndex++;
00463                 }
00464 
00465                 cameraStreamingOff();       // Turn off line
    tracking
00466                 disableServo(LIFT);
00467                 positionBot(); // In case we want to make a -128 brad turn
    after picking up ball
00468                 ignoreBreakBeamCount = BEAM_IGNORE_COUNT;
00469                 trackLineInit();       // Turn line tracking back on
00470             }
00471 
00472         }
00473     }
00474 
00475     cameraStreamingOff();
00476 
00477     // Make sure lift is up (in case we missed a ball or incorrectly thought
    one was there)
00478     if (liftDown)
00479     {
00480         brake(BOTH_MOTORS);
00481 #if DEBUGGING
00482         lcdWriteStr("No ball         ", 0, 0);
00483 #endif
00484         setServo(LIFT, LIFT_UP);    // Raise the lift
00485         msDelay(700);
00486         disableServo(LIFT);
00487         liftDown = false;
00488 
00489         // correct goal state
00490         removeFromGoalList(nextBallNodeNum);
00491         numUnreachedGoals--;
00492         adjustNumKnownGoals(-1);
00493     }
00494 }
00495 
00496 void nestSequence(void)
00497 {
00498     // line track, until NEST_BUTTON is pressed
00499     trackLineInit();
```

```
00500
00501     while (!justPressed(NEST_BUTTON))
00502     {
00503         if (!lineStatsProcessed)
00504         {
00505             analyzeLineStats();
00506             adjustPWM();
00507         }
00508
00509         debounceButtons();
00510     }
00511
00512     brake(BOTH_MOTORS);
00513     cameraStreamingOff();
00514     setServo(LIFT, LIFT_UP);       // Turn lift on
00515     msDelay(300);
00516
00517     // Open door, back up, close door
00518     setServo(DOOR, DOOR_OPEN);
00519     moveStraight(-1, 255);              // Back up to take pressure off button
00520     brake(BOTH_MOTORS);
00521     //myDelay(25);                      // Let balls roll out
00522     msDelay(3000);
00523     setServo(DOOR, DOOR_CLOSED); // Leaves door closed, so lift and door
      don't colide on power up.
00524     //myDelay(10);                      // Wait for door to close
00525     msDelay(1000);
00526
00527     // Disable all servos
00528     disableServo(PAN);
00529     disableServo(TILT);
00530     disableServo(BOOM);
00531     disableServo(LIFT);
00532     disableServo(DOOR);
00533 }
00534
00535 /*
00536  * Searches for ground balls, picks-up bonous balls, and computes best path.
00537  */
00538 void junctionCode(void)
00539 {
00540     bool foundBall = false;
00541
00542     switch (botNode)
00543     {
00544     case (0):                     // old virtual windowing look for ball 7 and
      8
00545         foundBall = nodeCode0();
00546         break;
00547     case (21):                          // Suppress standard seek, should
      already
00548         break;                          // skip junction code at this node
00549     case (22):
00550         foundBall = standardBallSearch();  // standard seek and
00551         foundBall |= nodeCode22();         // tilt look for ball 9, 11, and 12
00552         break;
00553     case (31):                           // rotate bot for diagonal ball search
00554     case (34):                           // from any heading at node 31 or 34
00555         foundBall = diagNodeCode();
00556         break;
00557     case (37):                           // seek for top balls from nest
      sensor
00558         if (!allGoalsFound() && (!checkedList[13] || !checkedList[17]))
00559         {
00560             botNode = 35;
00561             moveStraight(10, 255);
00562             foundBall = diagNodeCode();
00563             moveStraight(-10, 255);
00564             botNode = 37;
00565
00566             //pathList[pathListIndex--] = 36;
00567             //pathList[pathListIndex--] = 35;
00568         }
00569         break;
00570     default:
00571         foundBall = standardBallSearch();
00572         break;
00573     }
00574
00575     if (foundBall)
00576     {
```

```
00577         // clear checked list, if last ball found
00578         if (allGoalsFound())
00579         {
00580             uint8_t i;
00581             for (i = 0; i < NUM_BALL_NODES + 1; i++)
00582             {
00583                 checkedList[i] = true;
00584             }
00585         }
00586         updatePath();
00587         printGoalList();
00588     }
00589 }
00590
00591
00592 /*
00593  * Returns true if a ball is found and the goal list is updated
00594  */
00595 bool standardBallSearch( void )
00596 {
00597     GraphNodeType curNode;
00598     GraphNodeType nextNode;
00599     uint8_t nextNodeNum;
00600     int8_t lookDir = -1;
00601     int8_t hallHeading = 0;
00602     uint8_t ballDist = 0;
00603     uint8_t uncheckedBalls[3][2];
00604     uint8_t numUncheckedBalls = 0;
00605     bool foundBall = false;
00606     uint8_t i;
00607
00608     bool stopped = false;
00609     inSeekPosition = false;
00610
00611     // Check for balls to the left, then to the right
00612     for (i = 0; i < 2; i++)
00613     {
00614         ballDist = 0;
00615         hallHeading = botHeading + lookDir * 64;
00616         nextNodeNum = botNode;
00617         numUncheckedBalls = 0;
00618         // Continue traversing nodes to the left (or right) until you hit the
       end
00619         while (nextNodeNum > 0)
00620         {
00621             getNode(nextNodeNum, &curNode);
00622             nextNodeNum = getNodeAtHeading(&curNode, hallHeading);
00623             if (nextNodeNum > 0)
00624             {
00625                 getNode(nextNodeNum, &nextNode);
00626                 // Keep track of how far away we are from the bot's current
       node
00627                 ballDist += getCostToNode(&curNode, nextNodeNum);
00628                 if (isBallNode(nextNodeNum) && !checkedList[nextNodeNum])
00629                 {
00630                     uncheckedBalls[numUncheckedBalls][BALL_DIST] = ballDist;
00631                     uncheckedBalls[numUncheckedBalls][BALL_NODE_NUM] =
      nextNodeNum;
00632                     checkedList[nextNodeNum] = true;
00633                     numUncheckedBalls++;
00634                 }
00635             }
00636         }
00637         // Set pan, tilt, hi-res, etc...
00638
00639         if (numUncheckedBalls > 0)
00640         {
00641             stopped = true;
00642             trackColorInit(lookDir);
00643
00644             if (lookDir == -1)
00645             {
00646                 foundBall |= cameraSeekLeft(uncheckedBalls, numUncheckedBalls);
00647             } else if (lookDir == 1)
00648             {
00649                 foundBall |= cameraSeekRight(uncheckedBalls, numUncheckedBalls)
      ;
00650             }
00651         }
00652
00653         lookDir *= -1;  // Look the other way the next time through
```

```
00654     }
00655
00656     if (stopped)
00657     {
00658         moveStraight(0xb, 255);
00659         setServo(PAN, PAN_CENTER + panOffset);
00660         setServo(TILT, TILT_FORWARD);
00661         msDelay(600);
00662     }
00663
00664     // Returns true if one or more balls are found
00665     return foundBall;
00666 }
00667
00668
00669 inline bool nodeCode0(void)
00670 {
00671    bool foundBall = false;
00672
00673    // two virtual windows
00674
00675    return foundBall;
00676 }
00677
00678
00679 inline bool nodeCode22()
00680 {
00681    bool foundBall = false;    // Return value
00682    uint8_t scanHeight = 4;
00683    uint8_t y = 254;
00684    uint8_t scanLimit = 1;
00685    uint8_t foundBallNum = 0;
00686
00687    if (botHeading != 0)
00688       return false;
00689
00690    trackColorInit(LOOK_UP);
00691
00692    // scan from small ground distance to large ground distance
00693    while ( y - scanHeight > scanLimit )
00694    {
00695       y -= scanHeight;
00696       setVirtualWindow(1, y-scanHeight, 174, y);
00697       if ( seeBall() )
00698       {
00699          foundBall = true;
00700
00701          // find ball number of ball at this x
00702          if( y > 148 )
00703             foundBallNum = 9;
00704          else if ( y > 50 )
00705             foundBallNum = 11;
00706          else
00707             foundBallNum = 12;
00708
00709          addToGoalList( foundBallNum );
00710
00711          while ( seeBall() )
00712          {
00713             y -= scanHeight;
00714             setVirtualWindow(1, y-scanHeight, 174, y);
00715          }
00716       }
00717    }
00718
00719    setServo(PAN, PAN_CENTER+panOffset);
00720    setServo(TILT, TILT_FORWARD);
00721    msDelay(300);
00722
00723    return foundBall;
00724 }
00725
00726
00727 inline bool diagNodeCode(void)
00728 {
00729    bool foundBall = false;
00730
00731    if( botHeading == N_WEST && (!checkedList[13] || !checkedList[17]) )
00732    {
00733       tankTurn(255,tempTweak3);   // tank right
00734       botHeading += 41;
```

```
00735       foundBall = standardBallSearch();
00736       botHeading -= 41;
00737       tankTurn(255, -1*tempTweak3);     // tank left
00738    }
00739    else if( botHeading != S_EAST && (!checkedList[14] || !checkedList[15]
      ) )
00740    {
00741       tankTurn(255, -1*tempTweak3);     // tank left
00742       botHeading -= 41;
00743       foundBall = standardBallSearch();
00744       botHeading += 41;
00745       tankTurn(255,tempTweak3);   // tank right
00746    }
00747
00748    return foundBall;
00749 }
00750
00751 inline bool nodeCode37( void )
00752 {
00753    bool foundBall = false;
00754
00755    // pass special values into cameraSeekLeft
00756
00757    return foundBall;
00758 }
```

9.35 robot_control.h File Reference

High-level logic controlling Caddy's actions.

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

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

Macros

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

Functions

- void runRoborodentiaCourse (void)

  Run the Roborodentia course from start to finish.
- void moveToJunction (uint8_t numJunctions, bool justTurned)

  Move to next junction in pathList.
- void nestSequence (void)

  Sequence of actions to peform once the nest is reached.
- void **junctionCode** (void)
- void bonusBallPickUpManeuver (int8_t bbHeading, int8_t nextHeading)

  Orients Caddy to grab a bonus ball, grabs the ball, and reorients for the next node.

Variables

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

9.35.1 Detailed Description

High-level logic controlling Caddy's actions.

See Also

Problem Summary

Definition in file robot_control.h.

9.35.2 Function Documentation

9.35.2.1 void bonusBallPickUpManeuver (int8_t *bbHeading*, int8_t *nextHeading*) [inline]

Orients Caddy to grab a bonus ball, grabs the ball, and reorients for the next node.

Parameters

in	*bbHeading*	Heading Caddy must have for bonus ball pickup
in	*nextHeading*	Heading Caddy must have after bonus ball pickup

Definition at line 266 of file robot_control.c.

9.35.2.2 void nestSequence (void)

Sequence of actions to peform once the nest is reached.

Main purpose is to release the balls from the hopper.

Definition at line 496 of file robot_control.c.

9.35.2.3 void runRoborodentiaCourse (void) [inline]

Run the Roborodentia course from start to finish.

Returns once all balls have been collected and placed in the nest.

Definition at line 79 of file robot_control.c.

9.36 robot_control.h

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

```
00029
00030 #define BALL_DIST       0
00031 #define BALL_NODE_NUM   1
00032
00033 // Global variables
00034 extern uint8_t botNode;
00035 extern int8_t botHeading;
00036 extern uint8_t numUnreachedGoals;
00037
00043 inline void runRoborodentiaCourse(void);
00044
00045 inline void moveToJunction(uint8_t numJunctions, bool justTurned)
      ;
00046
00052 void nestSequence(void);
00053
00054 void junctionCode(void);
00055
00063 inline void bonusBallPickUpManeuver(int8_t bbHeading,
      int8_t nextHeading);
00064
00065 #endif // #ifndef BOTCNTRL_H_
```

9.37 servos.h File Reference

Servo control interface for Caddy.

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

Macros

- #define PAN 0

  Servo to control the left/right swivel motion of the camera.
- #define TILT 1

  Servo to control the forward/back rocking motion of the camera.
- #define BOOM 2

  Servo to control the boom arm.
- #define LIFT 3

  Servo to control the lift and scooping blade.
- #define DOOR 4

  Servo to control the hopper door.
- #define **PAN_CENTER** 128
- #define **MAX_PAN_LEFT** 50
- #define **MAX_PAN_RIGHT** 195
- #define **TILT_BB_GRAB** 133
- #define TILT_VERT 128

  requires boom to be MAX_BOOM_UP
- #define **TILT_LOOKUP** 186
- #define **TILT_BACK** MAX_TILT_BACK
- #define **TILT_FORWARD** MAX_TILT_FORWARD
- #define **MAX_TILT_BACK** 195
- #define **MAX_TILT_FORWARD** 70
- #define **BOOM_UP** MAX_BOOM_UP
- #define BOOM_BB_GRAB 183

  cup 2.32 inches off ground
- #define **MAX_BOOM_UP** 45

- #define **MAX_BOOM_DOWN** 148
- #define **LIFT_OPEN** MAX_LIFT_OPEN
- #define **LIFT_UP** MAX_LIFT_UP
- #define LIFT_CORRAL 150

  Lift position needed to release balls into corral.
- #define LIFT_BB_READY 114

  Places the "V" of the lift 3 inches off ground, ready for bonus ball grabbing maneuver.
- #define MAX_LIFT_OPEN 183

  Mechanical limit in which the ground ball blade is fully open.
- #define MAX_LIFT_UP 74
- #define **DOOR_CLOSED** MAX_DOOR_CLOSED
- #define **DOOR_OPEN** MAX_DOOR_OPEN
- #define MAX_DOOR_CLOSED 71

  Close door just to the point of touching.
- #define MAX_DOOR_OPEN 200

  Mechanical limit of the door lever arm.

Functions

- void setServo (uint8_t servoNum, uint8_t servoPos)

  Move servo to position and leave it on.
- void disableServo (uint8_t servoNum)

  Cut power to a specified servo.

9.37.1 Detailed Description

Servo control interface for Caddy.

Definition in file servos.h.

9.37.2 Macro Definition Documentation

9.37.2.1 #define BOOM 2

Servo to control the boom arm.

The boom moves the long arm which holds the entire camera/scoop/pan/tilt assembly.

This is used for the bonus ball pickup maneuver as well as positioning of the camera.

Position is held mechanically (servo can be passive after moving to position)

Definition at line 61 of file servos.h.

9.37.2.2 #define DOOR 4

Servo to control the hopper door.

Opens and closes hopper door (e.g. to release the balls into the corral at the end of the Roborodentia course)

Position is held mechanically (servo can be passive after moving to position)

Definition at line 88 of file servos.h.

9.37.2.3 #define LIFT 3

Servo to control the lift and scooping blade.

The lift servo, with just simple rotational actuation, drives the complex mechanical motion of lift assembly which:

- Opens/closes the ground ball scooping blade
- Lifts captured ground balls into the hopper
- Aligns the robot with the corner of a wall using the 90 degree "V" shape of the front of the lift (used as the first step of the bonus ball pickup maneuver)

Position is held mechanically (servo can be passive after moving to position)

Definition at line 77 of file servos.h.

9.37.2.4 #define MAX_LIFT_UP 74

Definition at line 113 of file servos.h.

9.37.2.5 #define PAN 0

Servo to control the left/right swivel motion of the camera.

Allows the camera to look down corridors to the side

Position is held by actively driving the servo.

Definition at line 34 of file servos.h.

9.37.2.6 #define TILT 1

Servo to control the forward/back rocking motion of the camera.

Allows the camera to look down at the ground (e.g. for line tracking) and to look forward to see if balls lie in the path ahead.

The tilt also controls the bonus ball scooping cup since the cup and the camera are on the same rigid assembly.

Position is held by actively driving the servo.

Definition at line 47 of file servos.h.

9.37.3 Function Documentation

9.37.3.1 void disableServo (uint8_t *servoNum*)

Cut power to a specified servo.

param[in] servoNum Number of the servo output on the CMUcam2

Definition at line 30 of file servos.c.

9.37.3.2 void setServo (uint8_t *servoNum*, uint8_t *servoPos*)

Move servo to position and leave it on.

param[in] servoNum Number of the servo output on the CMUcam2 param[in] servoPos 0-255 value which maps to servo angle of 0-180 degrees

Definition at line 25 of file servos.c.

9.38 servos.h

```
00001 /*
00002  * This file is part of Caddy.
00003  *
00004  *  Caddy is free software: you can redistribute it and/or modify
00005  *  it under the terms of the GNU General Public License as published by
00006  *  the Free Software Foundation, either version 3 of the License, or
00007  *  (at your option) any later version.
00008  *
00009  *  Caddy is distributed in the hope that it will be useful,
00010  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  *  GNU General Public License for more details.
00013  *
00014  *  You should have received a copy of the GNU General Public License
00015  *  along with Caddy.  If not, see <http://www.gnu.org/licenses/>.
00016  */
00021 #ifndef SERVOS_H
00022 #define SERVOS_H
00023
00024 // avr-libc
00025 #include <stdint.h>
00026
00034 #define PAN    0
00035
00047 #define TILT   1
00048
00061 #define BOOM   2
00062
00077 #define LIFT   3
00078
00088 #define DOOR   4
00089
00090 // Servo positions
00091 #define PAN_CENTER          128 //133
00092 #define MAX_PAN_LEFT        50
00093 #define MAX_PAN_RIGHT       195
00094
00095 #define TILT_BB_GRAB        133
00096 #define TILT_VERT           128
00097 #define TILT_LOOKUP         186
00098 #define TILT_BACK           MAX_TILT_BACK
00099 #define TILT_FORWARD        MAX_TILT_FORWARD
00100 #define MAX_TILT_BACK       195
00101 #define MAX_TILT_FORWARD    70
00102
00103 #define BOOM_UP             MAX_BOOM_UP
00104 #define BOOM_BB_GRAB        183
00105 #define MAX_BOOM_UP         45
00106 #define MAX_BOOM_DOWN       148
00107
00108 #define LIFT_OPEN           MAX_LIFT_OPEN
00109 #define LIFT_UP             MAX_LIFT_UP
00110 #define LIFT_CORRAL         150
00111 #define LIFT_BB_READY       114
00112 #define MAX_LIFT_OPEN       183
00113 #define MAX_LIFT_UP         74
00114
00115 #define DOOR_CLOSED         MAX_DOOR_CLOSED
00116 #define DOOR_OPEN           MAX_DOOR_OPEN
00117 #define MAX_DOOR_CLOSED     71
00118 #define MAX_DOOR_OPEN       200
00119
00120
00126 void setServo(uint8_t servoNum, uint8_t servoPos);
00127
00133 void disableServo(uint8_t servoNum);
00134
00135 #endif  // #ifndef
```

9.39 tether_ui.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 tether_ui.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 tether_ui.h.

9.39.2.2 #define NAV_LCD_MODE 0

Default LCD display mode

Definition at line 31 of file tether_ui.h.

9.40 tether_ui.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 tweak_data.c File Reference

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

Include dependency graph for tweak_data.c:

Functions

- void **loadTweakValues** (void)
- void **storeTweakValues** (void)

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**

9.41.1 Detailed Description

Definition in file tweak_data.c.

9.42 tweak_data.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 "tweak_data.h"
00019
```

```
00020 // avr-libc
00021 #include <avr/io.h>
00022 #include <avr/interrupt.h>
00023
00024 // Global variables - Runtime configurable parameters
00025 uint8_t l_base;
00026 uint8_t r_base;
00027 uint16_t slopeCoef;
00028 uint16_t offCoef;
00029 uint8_t dampCoef;
00030 uint8_t lineCenter;
00031 uint8_t turnPoint;
00032 uint8_t turnSubtract;
00033 int8_t panOffset;
00034 int8_t tiltOffset;
00035 uint16_t tractorOvershootDelay;
00036 uint8_t tempTweak1;
00037 int8_t tempTweak2;
00038 uint16_t tempTweak3;
00039 uint16_t tempTweak4;
00040
00041 uint8_t lcdMode;  // <- need debugger menu for this, remove old init/toggling,
       and save in EEPROM
00042 uint8_t testMode; // <- need to save this in EEPROM
00043
00044 static uint8_t EEPROM_read(unsigned int uiAddress);
00045 static void EEPROM_write(unsigned int uiAddress, uint8_t ucData);
00046
00047 // Initializes constants that can be tweaked by debugger
00048 inline void loadTweakValues(void)
00049 {
00050     cli(); // disable all interrupts
00051
00052     // EEPROM Reads
00053     l_base                = EEPROM_read(EE_ADDR_LEFT_BASE);
00054     r_base                = EEPROM_read(EE_ADDR_RIGHT_BASE);
00055     slopeCoef             = (EEPROM_read(EE_ADDR_SLOPE_COEF) << 8) +
00056                              EEPROM_read(EE_ADDR_SLOPE_COEF + 1);
00057     offCoef               = (EEPROM_read(EE_ADDR_OFF_COEF) << 8) +
00058                              EEPROM_read(EE_ADDR_OFF_COEF + 1);
00059     dampCoef              = EEPROM_read(EE_ADDR_DAMP_COEF);
00060     lineCenter            = EEPROM_read(EE_ADDR_LINE_X_CENTER);
00061     turnPoint             = EEPROM_read(EE_ADDR_TURN_POINT);
00062     turnSubtract          = EEPROM_read(EE_ADDR_TURN_SUBTRACT);
00063     panOffset             = EEPROM_read(EE_ADDR_PAN_OFFSET);
00064     tiltOffset            = EEPROM_read(EE_ADDR_TILT_OFFSET);
00065     tractorOvershootDelay = (EEPROM_read(EE_ADDR_TRACTOR_OVERSHOOT_DELAY) << 8)
       +
00066                              EEPROM_read(EE_ADDR_TRACTOR_OVERSHOOT_DELAY + 1);
00067     testMode              = EEPROM_read(EE_ADDR_TEST_MODE);
00068     tempTweak1            = EEPROM_read(EE_ADDR_TEMP_TWEAK1);
00069     tempTweak2            = EEPROM_read(EE_ADDR_TEMP_TWEAK2);
00070     tempTweak3            = (EEPROM_read(EE_ADDR_TEMP_TWEAK3) << 8) +
00071                              EEPROM_read(EE_ADDR_TEMP_TWEAK3 + 1);
00072     tempTweak4            = (EEPROM_read(EE_ADDR_TEMP_TWEAK4) << 8) +
00073                              EEPROM_read(EE_ADDR_TEMP_TWEAK4 + 1);
00074
00075     sei(); // enable all interrupts
00076 }
00077
00078 // Saves constants after they have been changed by the debugger
00079 inline void storeTweakValues(void)
00080 {
00081     cli(); // disable all interrupts
00082
00083     // EEPROM writes
00084     EEPROM_write(EE_ADDR_LEFT_BASE, l_base);
00085     EEPROM_write(EE_ADDR_RIGHT_BASE, r_base);
00086     EEPROM_write(EE_ADDR_SLOPE_COEF, slopeCoef >> 8);
00087     EEPROM_write(EE_ADDR_SLOPE_COEF + 1, slopeCoef);
00088     EEPROM_write(EE_ADDR_OFF_COEF, offCoef >> 8);
00089     EEPROM_write(EE_ADDR_OFF_COEF + 1, offCoef);
00090     EEPROM_write(EE_ADDR_DAMP_COEF, dampCoef);
00091     EEPROM_write(EE_ADDR_LINE_X_CENTER, lineCenter);
00092     EEPROM_write(EE_ADDR_TURN_POINT, turnPoint);
00093     EEPROM_write(EE_ADDR_TURN_SUBTRACT, turnSubtract);
00094     EEPROM_write(EE_ADDR_PAN_OFFSET, panOffset);
00095     EEPROM_write(EE_ADDR_TILT_OFFSET, tiltOffset);
00096     EEPROM_write(EE_ADDR_TRACTOR_OVERSHOOT_DELAY, tractorOvershootDelay >> 8);
00097     EEPROM_write(EE_ADDR_TRACTOR_OVERSHOOT_DELAY + 1, tractorOvershootDelay);
00098     EEPROM_write(EE_ADDR_TEST_MODE, testMode);
```

```
00099     EEPROM_write(EE_ADDR_TEMP_TWEAK1, tempTweak1);
00100     EEPROM_write(EE_ADDR_TEMP_TWEAK2, tempTweak2);
00101     EEPROM_write(EE_ADDR_TEMP_TWEAK3, tempTweak3 >> 8);
00102     EEPROM_write(EE_ADDR_TEMP_TWEAK3 + 1, tempTweak3);
00103     EEPROM_write(EE_ADDR_TEMP_TWEAK4, tempTweak4 >> 8);
00104     EEPROM_write(EE_ADDR_TEMP_TWEAK4 + 1, tempTweak4);
00105
00106     sei(); // enable all interrupts
00107 }
00108
00109 uint8_t EEPROM_read(unsigned int uiAddress)
00110 {
00111     // Wait for completion of previous write
00112     while (EECR & (1 << EEWE)) ;
00113     // Set up address register
00114     EEAR = uiAddress;
00115     // Start eeprom read by writing EERE
00116     EECR |= (1 << EERE);
00117     // Return data from data register
00118     return EEDR;
00119 }
00120
00121 void EEPROM_write(unsigned int uiAddress, uint8_t ucData)
00122 {
00123     // Wait for completion of previous write
00124     while (EECR & (1 << EEWE)) ;
00125     // Set up address and data registers
00126     EEAR = uiAddress;
00127     EEDR = ucData;
00128     // Write logical one to EEMWE
00129     EECR |= (1 << EEMWE);
00130     // Start eeprom write by setting EEWE
00131     EECR |= (1 << EEWE);
00132     // EEMWE and EEWE are automatically cleared back to 0 by hardware
00133 }
```

9.43 tweak_data.h File Reference

Interface to the "tweak values" stored in EEPROM.

`#include <stdint.h>`
Include dependency graph for tweak_data.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)

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**

9.43.1 Detailed Description

Interface to the "tweak values" stored in 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 tweak_data.h.

9.44 tweak_data.h

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

```
00028
00029 //Locations in EEPROM
00030 #define EE_ADDR_LEFT_BASE           0x50
00031 #define EE_ADDR_RIGHT_BASE          0x51
00032 #define EE_ADDR_SLOPE_COEF          0x52 //uint16_t
00033 #define EE_ADDR_OFF_COEF            0x54 //uint16_t
00034 #define EE_ADDR_DAMP_COEF           0x56
00035 #define EE_ADDR_LINE_X_CENTER       0x57
00036 #define EE_ADDR_TURN_POINT          0x58
00037 #define EE_ADDR_TURN_SUBTRACT       0x59
00038 #define EE_ADDR_PAN_OFFSET          0x5A
00039 #define EE_ADDR_TILT_OFFSET         0x5B
00040 #define EE_ADDR_TRACTOR_OVERSHOOT_DELAY 0x5C //uint16_t
00041 #define EE_ADDR_TEST_MODE           0x5E
00042 #define EE_ADDR_TEMP_TWEAK1         0x5F
00043 #define EE_ADDR_TEMP_TWEAK2         0x60
00044 #define EE_ADDR_TEMP_TWEAK3         0x61 //uint16_t
00045 #define EE_ADDR_TEMP_TWEAK4         0x63 //uint16_t
00046 //next address 0x65
00047
00048 /*
00049 // Current values - Bigger motors at 6 Volts
00050 #define INIT_LEFT_BASE_SPEED        0xF7
00051 #define INIT_RIGHT_BASE_SPEED       0xF0
00052 #define INIT_SLOPE_COEF             0x0110   // <--- 0x110 could be tweaked
00053 #define INIT_OFF_COEF               0x0001
00054 #define INIT_DAMP_COEF              0x01     // <--- 0x01 could be tweaked
00055 #define INIT_LINE_X_CENTER          0x25
00056 #define INIT_TURN_POINT             0x15     // Turn values
00057 #define INIT_TURN_SUBTRACT          0x0A
00058 pan offset 0x05
00059 tilt offset 0xE7
00060
00061 #define TRACTOR_OVERSHOOT_DELAY     5000
00062 */
00063
00064 #define BASE_MIN                    0x60     // <--- also worked at 0xB0
00065 #define BASE_MAX                    0xFF
00066
00067 // Pickup values
00068 #define BALL_CHECK_RATIO            16
00069 #define PICK_UP_POINT               0x16
00070
00071
00072 /*
00073 // Old values - 6V Solarbotics before damping fix
00074 #define LEFT_BASE_SPEED           0x8C
00075 #define RIGHT_BASE_SPEED          0xC2
00076 #define BASE_MIN                  100
00077 #define BASE_MAX                  255
00078 #define OFF_COEF                  0x2
00079 #define SLOPE_COEF                0x100
00080 #define DAMP_COEF                 0x14
00081 */
00082
00083 // Global variables - Runtime configurable parameters
00084 extern uint8_t l_base;
00085 extern uint8_t r_base;
00086 extern uint16_t slopeCoef;
00087 extern uint16_t offCoef;
00088 extern uint8_t dampCoef;
00089 extern uint8_t lineCenter;
00090 extern uint8_t turnPoint;
00091 extern uint8_t turnSubtract;
00092 extern int8_t panOffset;
00093 extern int8_t tiltOffset;
00094 extern uint16_t tractorOvershootDelay;
00095 extern uint8_t tempTweak1;
00096 extern int8_t tempTweak2;
00097 extern uint16_t tempTweak3;
00098 extern uint16_t tempTweak4;
00099
00100 extern uint8_t lcdMode;
00101 extern uint8_t testMode;
00102
00103 inline void loadTweakValues( void );
00104 inline void storeTweakValues( void );
00105
00106 #endif // #ifndef EEPROM_H_
```

References

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

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

Index