

```

/**
 * @file init.c
 * @brief Perform initialization and start handler tasks
 * Copyright (C) 2017 Ethan Wells
 *
 * This program is free software: you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the Free
 * Software Foundation, either version 3 of the License, or(at your option) any
 * later version.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
 * details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program. If not, see <https://www.gnu.org/licenses/>
 */

#include "../include/robot.h"

static inline float lMogoRecalc(int p) {
    return p * 1.1;
} /* lMogoRecalc */

static inline float lineRecalc(int v) {
    return (float)(v > 16);
}

float clawPotRecalc(int v) {
    static int l = 0;
    float a = (l + v) / 2;
    l = v;
    return a;
}

void initializeIO() {
    watchdogInit();
}

/**
 * Notify both through the terminal and an lcd
 *
 * @param buffer the text to display
 */
void notice(const char *buffer) {

```

```

        #ifdef DEBUG_MODE
            print(buffer);
        #endif
        lcdSetText(uart1, 2, buffer);
        delay(5);
    } /* notice */

void init() {
    // LCD initialization
    lcdInit(uart1);
    lcdSetBacklight(uart1, true);

    #ifdef DEBUG_MODE
        print("\nInitializing... ");
    #endif
    lcdSetText(uart1, 1, "Initializing...");

    // Set up the analog sensors
    gyro = newGyro(1, true, 200);
    gyro.child = new(Sensor);
    *gyro.child = newGyro(2, true, 195);
    notice("gyroscopes, ");
    Sensor *mogoAngle = new(Sensor);
    *mogoAngle = newAnalog(3, true);
    notice("mobile goal angle, ");
    Sensor *clawAngle = new(Sensor);
    *clawAngle = newAnalog(5, false);
    clawAngle->zero = 0;
    clawAngle->recalc = &clawPotRecalc;
    notice("claw angle, ");
    for (int i = 0; i < 3; i++) {
        line[i] = newAnalog(i + 6, false);
        line[i].inverted = true;
        line[i].recalc = &lineRecalc;
    }
    notice("line sensors");

    // Set up the digital sensors
    Sensor *armCoder = new(Sensor);
    *armCoder = newQuad(1, 2, false);
    notice("arm quad, ");
    Sensor *driveCoder[2] = { new(Sensor), new(Sensor) };
    *driveCoder[0] = newQuad(4, 5, true);
    notice("left drive quad, ");
    *driveCoder[1] = newQuad(8, 9, true);
    notice("right drive quad, ");
}

```

```

armLimit[0] = newDigital(11, true);
armLimit[1] = newDigital(12, true);
notice("arm limit switches, ");
sonic = new(Sensor);
*sonic = newSonic(6,7);
notice("ultrasonic, ");

// Initialize and set up all of the motors, servos, etc
claw      = motorCreate(3, false);
claw.sensor = clawAngle;
notice("claw motor, ");

arm      = motorCreate(5, false);
arm.child = new(Motor);
*arm.child = motorCreate(6, true);
arm.child->child = new(Motor);
*arm.child->child = motorCreate(8, false);
arm.sensor = armCoder;
notice("arm motors, ");

mogo      = motorCreate(1, false);
mogo.recalc = &lMogoRecalc;
mogo.deadband = 6;
mogo.child = new(Motor);
mogo.child->deadband = 6;
*mogo.child = motorCreate(10, true);
mogo.sensor = mogoAngle;
notice("mobile goal motors, ");

drive[0] = motorCreate(2, true);
drive[0].child = new(Motor);
*drive[0].child = motorCreate(4, true);
drive[0].sensor = driveCoder[0];

drive[1] = motorCreate(9, false);
drive[1].child = new(Motor);
*drive[1].child = motorCreate(7, false);
drive[1].sensor = driveCoder[1];
notice("drive motors, ");

lcdSetText(uart1, 1, "Ready!");
#ifdef DEBUG_MODE
    print("\n\n");
#endif
setTeamName("709S");
notice("done!");

```

```
        // Start the LCD task
        LCDHandle = GO(lcdTask, NULL);
    } /* init */
```