

```

/**
 * @file autoLeft.c
 * @brief Left side autonomous routines
 * 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/auto.h"

void autonLeft12() {
    getMogo(); // Get the mobile goal

    turnTo(-6, 500); // Align to a left tilt
    driveSettings[1].max -= 40; // Limit right side speed
    GO(placeConeT, NULL);
    driveToPosition(220, 650, 2400); // Back up
    driveSettings[1].max += 40; // Correct speed
    turnTo(-165, 2000); // Turn around
    delay(400);

    // Reset drive encoders & gyro
    sensorReset(drive[0].sensor);
    sensorReset(drive[1].sensor);
    sensorReset(&gyro);

    driveToPositionAngle(1000, 900, 13, 1850); // Drive arc 13 degrees clockwise
    mogoP(MOGO_DOWN);

    driveSet(-127, -127); // Back up the drive
    delay(130);
    mogoP(MOGO_DOWN - 300); // Bring the mobile goal up a bit
    delay(250);
    driveSet(0, 0); // Stop the drive
}

```

```

} /* autonLeft12 */

void autonLeft22() {
    getMogo(); // Get the mobile goal

    turnTo(-1, 550);
    driveToPosition(1650, 1650, 3500);
    turnTo(-17, 1850); // Align to a left tilt of 12 degrees

    // driveSettings[1].max -= 40; // Limit right side speed
    GO(placeConeT, NULL); // Place cone
    // driveToPosition(-485, -210, 5500); // Back up
    driveToPosition(-850, -850, 5500); // Back up
    // driveSettings[1].max += 40; // Correct speed
    GO(liftPID, NULL);
    turnTo(-144, 2500); // Turn around

    // Reset drive encoders & gyro
    sensorReset(drive[0].sensor);
    sensorReset(drive[1].sensor);
    sensorReset(&gyro);

    /*
    mogo.power = 127;
    mogo.child->power = 127;
    motorSet(mogo.port, 127 * mogo.isInverted);
    motorSet(mogo.child->port, 127 * mogo.child->isInverted);

    driveSet(70, 70);
    delay(250);
    mutexGive(mogo._mutex);
    */

    mogo.power = 70;
    motorUpdate(&mogo);
    // TaskHandle mogoHandle = GO(mogoPT, MOGO_MID + 125);
    // driveToPositionAngle(1525, 1425, 13, 1675); // Drive arc 13 degrees clockwise
    driveToPositionAngle(1525, 1425, 13, 1675); // Drive arc 13 degrees clockwise

    // if (taskGetState(mogoHandle))
    //     taskDelete(mogoHandle);

    driveSet(-10, -10);
    // mogo.power = 127;
    // motorUpdate(&mogo);

```

```

delay(350);
sensorReset(&gyro);

mogo.power = 127;
motorUpdate(&mogo);
driveSet(-127, -127);

delay(150);

// if (taskGetState(mogoHandle))
//     taskDelete(mogoHandle);
TaskHandle mogoHandle = GO(mogoPT, MOGO_MID);

/*
mutexGive(mogo._mutex);
while (!mutexTake(mogo._mutex, 1))
    mutexGive(mogo._mutex);

mogo.power = 127;
mogo.child->power = 127;
motorSet(mogo.port, 127 * mogo.isInverted);
motorSet(mogo.child->port, 127 * mogo.child->isInverted);

driveSet(64, 64);
delay(200);
driveSet(-80, -80);

mogo.power = 0;
mogo.child->power = 0;
motorSet(mogo.port, 0);
motorSet(mogo.child->port, 0);

delay(150);
mutexGive(mogo._mutex);

mogoHandle = GO(mogoPT, MOGO_MID - 100);
// */

driveToPosition(200, 200, 2495);

while (taskGetState(mogoHandle)) {
    delay(10);
}

liftSettings.target = lift.sensor->averageVal; // Reset the lift position to it's
                                              // current position

```

```
} /* autonLeft22 */
```