```
/**
 * @file auto.h
 * @brief Structures and information pertianing to autonomous that is needed in
 * places other than auto.c
 * 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/>
 */

#ifndef CARL_AUTO_H_
#define CARL_AUTO_H_

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

#define MAX_AUTON 7

enum MOGO_POS {
        MOGO_UP   = 75,
        MOGO_PART = 550,
        MOGO_MID  = 1350,
        MOGO_DOWN = 2200,
};

enum ARM_POS {
        ARM_DOWN      = 10,
        ARM_QUARTER   = 275,
        ARM_HALF      = 430,
        ARM_3_QUARTER = 500,
};


typedef struct Auton {
        const char *name;
        const char *sensorName;
        Sensor    **sensor;
```

```
        void (*execute)();
} Auton;

typedef enum Direction {
        dUp,
        dDown,
        dLeft,
        dRight,
        dIn,
        dOut,
} Direction;

typedef struct Triple {
        int a;
        int b;
        int c;
} Triple;

/**
 * A list of the autonomouses/LCD menus
 */
extern Auton autons[MAX_AUTON + 1];
/**
 * The autonomous, as selected by the LCD menu, to run
 */
extern int   selectedAuton;

/*
 * @breif Bring the arm to the specified position
 *
 * @param pos the position to bring the arm to
 * @param until the maximum amount of time this can take in ms
 */
void armToPosition(float         pos,
                   unsigned long until);

/**
 * @brief Bring the drive to a specific position
 *
 * @param l the left position
 * @param r the right position
 * @param until the maximum amount of time this can take
 */
void driveToPosition(int           l,
                     int           r,
                     unsigned long until);
```

```
/**
 * @brief Bring the drive to a specific position while attempting to maintain an angle
 *
 * @param l the left position
 * @param r the right position
 * @param a the angle to maintain
 * @param until the maximum amount of time this can take
 */
void driveToPositionAngle(int           l,
                          int           r,
                          int           a,
                          unsigned long until);

/**
 * @brief Bring the mobile goal intake to a position
 *
 * @param p the position to go to
 */
void mogoP(int p);

/**
 * Use PID to turn to a specific angle
 *
 * @param angle the angle to turn to
 * @param until the max amount of time this can take
 */
void turnTo(int           angle,
            unsigned long until);

/**
 * @breif Go forward and get the mobile goal! (the beginning of nearly any
 * autonomous here)
 */
void getMogo();

/**
 * @brief Place the cone on dat goal!
 */
void placeCone();

/**
 * Drop mobile goal into the 20 point zone
 *
 * @return a TaskHandle of the task bringing the intake back into the robot
 */
```

```
TaskHandle dropMogo20(TaskHandle mogoHandle);

/**
 * @brief Back up at a certain time for about half a second
 */
Task backUp(void *time);

/**
 * @brief bring the mobile goal intake to a position in a task
 */
Task mogoPT(void *p);

/**
 * @brief task for placing a cone
 */
Task placeConeT(void *none);

/**
 * @brief Task for armToPosition
 */
Task armPID(void *none);

/**
 * @brief don't use, it doesn't work
 */
void moveTo(int leftV,
            int rightV,
            int armV,
            int mogoV,
            int clawV,
            int gyroV);

#endif // AUTO_ROBOT_H_
```