```c
typedef struct {
    uint16_t h;
    uint8_t s;
    uint8_t v;
} Color;

/**
 * Set motor speed
 * @param mot_lr -  MOT_L for left, MOT_R for right
 * @param speed  -  between -100 and +100
 * @return       -  0 on success
 */
int setMotorSpeed(uint8_t mot_lr, float speed);

/**
 * Read the current value of the encoder's counter
 * @param mot_lr -  MOT_L for left, MOT_R for right motor
 * @return       -  encoder counter value
 */
int getEncoderPosition(uint8_t mot_lr);

/**
 * Prints text to the LCD. Works just like printf after the position specifiers
 * @param row -  row of starting position
 * @param col -  column of starting position
 * @param fmt -  printf-like format string followed by a variable number of arguments
 */
int lcdPrintf(uint8_t row, uint8_t col, const char *fmt, ...);

/**
 * Print to the USB serial port. Works just like regular printf.
 * @param fmt -  printf-like format string followed by a variable number of arguments
 */
int uartPrintf(const char *fmt, ...);

/**
 * Print to the telemetry webpage. Works just like regular printf.
 * @param fmt -  printf-like format string followed by a variable number of arguments
 */
int espPrintf(const char *fmt, ...);

/**
 * Read input from the telemetry webpage console input.
 * @param data -  pointer to the character buffer where we want to get the output
 * @return     -  1 if there is data available, else returns 0
 */
int espRead(char* data);

/**
 * Read color in HSV format
 * @param color -  the Color struct pointer in which we want to get the result
 */
void getColorHsv(Color* color);

/**
 * Set the servo's position in degrees
 * @param position -  must be between -90 and 90. Otherwise it will be clipped to those values.
 */
void setServoPosition(int8_t position);

/**
 * Get the distance measured by the ultrasonic ranging sensor
 * @return distance -  measured distance in cm
 */
uint16_t getUsDistance();

/**
 * Do nothing for the specified time
 * @param delay -  time to wait in ms
 */
void delayMs(uint32_t delay);
```