Manual de Usuario para Control de Motor DC con Arduino

Introducción

Este manual describe detalladamente el funcionamiento del código para controlar un motor DC utilizando Arduino, un codificador rotatorio y una pantalla LCD I2C. Está diseñado para personas con conocimientos básicos de programación y proporciona una guía completa sobre la configuración inicial y el bucle principal del programa.

Librerías Incluidas

`#include <Arduino.h>` Proporciona las funciones básicas de Arduino para control de hardware y funciones básicas.

`#include <LiquidCrystal_I2C_Koba.h>` Controla la pantalla LCD I2C, facilitando la comunicación y control de la pantalla.
`#include <RotaryEncoder.h>` Maneja el codificador rotatorio, permitiendo detectar giros y pulsaciones del mismo.

Definiciones y Configuraciones

Definiciones de Barras para LCD

Estas matrices definen las barras que se mostrarán en el LCD para indicar la velocidad del motor.

```
uint8_t barra0[] = {0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10};
uint8_t barra1[] = {0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18};
uint8_t barra2[] = {0x1c, 0x1c, 0x1c, 0x1c, 0x1c, 0x1c, 0x1c, 0x1c};
uint8_t barra3[] = {0x1e, 0x1e, 0x1e, 0x1e, 0x1e, 0x1e, 0x1e, 0x1e};
uint8_t barra4[] = {0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f};
```

Definiciones de Estados del Sistema

Estas constantes representan los diferentes estados en los que puede estar el sistema.

```
#define INITIALIZING 0
#define READY 1
#define WORKING 2
#define PAUSED 3
#define FINISHING 4
#define FINISH 5
#define CANCELLED 6
```

Constantes de Tiempo y Configuración

Estas constantes definen los tiempos y las frecuencias del buzzer.

TIEMPOS EN MS, MULTIPLICADOS POR MIN_LOOP_MS_TIME
TIEMPOS EN SEC = (VALOR * MIN_LOOP_MS_TIME) /1000

```
#define MIN_LOOP_MS_TIME 10 //Tiempo minimo del loop
#define TIME_TO_RESET 5 //Tiempo de espera para cancelar reseteo de tiempo, está en segundos

#define MIN_MS_BUTTON_TIME 5 //Tiempo minimo de boton apretado
#define MIN_MS_BUTTON_LONGTIME 150 //Tiempo minimo para longpress apretado
#define BUZZER_INITIALIZING_TIME 2 //Tiempo de pitido inicial
#define BUZZER_FINISHING_TIME 10 //Tiempo de pitido < 5secs
#define BUZZER_FINISH_TIME 100 //Tiempo de pitido finalizado
#define BUZZER_CANCELLED_TIME 50 //Tiempo de pitido cancelado
#define BUZZER_INITIALIZING_FC 1000 //Frecuencia de pitido inicializando
#define BUZZER_FINISHING_FC 1500
#define BUZZER_FINISH_FC 1500
#define BUZZER_CANCELLED_FC 250
#define LCD_ON_TIME 500 //Tiempo de LCD ON
#define LCD_UPDATE_TIME 500 //Tiempo que tarda en actualizar banner informativos
```

Definición de Pines

Define los pines utilizados para el codificador rotatorio, el motor, el buzzer y el LED.

```
#define PIN_ENCODER_DT 2
#define PIN_ENCODER_CLK 3
#define PIN_ENCODER_SW 4
#define PIN_MOTOR 5
#define PIN_BUZZER 6
#define PIN_LED_TESTIGO 7
```

Banderas del Sistema

Estas banderas son bits individuales que representan varios estados y eventos del sistema.

```
#define MOTOR_ON bandera.unionBit.bit0
#define BUZZER_ON bandera.unionBit.bit1
#define CHANGING_PARAMS bandera.unionBit.bit2
#define DISPLAY_DATA bandera.unionBit.bit3
#define TIME_SET bandera.unionBit.bit4
#define SEC_PASSED bandera.unionBit.bit5
#define ENCODER_TURN_CCW bandera.unionBit.bit6
#define ENCODER_TURN_CW bandera.unionBit.bit7
#define BUTTON_SW_PRESS bandera2.unionBit.bit0
#define BUTTON_SW_LONG_PRESS bandera2.unionBit.bit1
#define BUTTON_SW_RELEASED bandera2.unionBit.bit2
#define LCD_ON bandera2.unionBit.bit3
#define UPDATE_LCD bandera2.unionBit.bit4
#define CHANGING_VEL bandera2.unionBit.bit5
#define CHANGING_MIN bandera2.unionBit.bit6
#define RESETTING bandera2.unionBit.bit7
#define BUZZERTIMEPASSED bandera3.unionBit.bit0
```

Estructuras de Unión

Estructuras de unión para manejar banderas como bytes individuales.

```
typedef union {
  struct {
    uint8_t bit0 : 1;
    uint8_t bit1 : 1;
    uint8_t bit2 : 1;
    uint8_t bit3 : 1;
    uint8_t bit4 : 1;
    uint8_t bit5 : 1;
    uint8_t bit6 : 1;
    uint8_t bit7 : 1;
  } unionBit;
  uint8_t unionByte;
} myByte;

myByte bandera;
myByte bandera2;
myByte bandera3;
```

Estructuras de Tiempo

Estas estructuras se utilizan para manejar el tiempo en segundos y minutos.

```
typedef struct {
  uint16_t sec;
  uint8_t min;
} timeStruct;

timeStruct timeToRun;
timeStruct lastTimeRun;
```

Variables Globales

Variables globales para manejar estados, contadores y configuraciones.

```
uint8_t mode = 0;
uint8_t secCounter = 0;
uint8_t timerEncoderSW = 0;
uint8_t timerUpdateLCD = 0;
uint16_t secsToGo;
uint16_t timerLCD;
uint16_t frecuencyBuzzer;
uint32_t lastTime;
char carac;
```

```
char linea[16]; //MAXIMO 16 CARACTERES *
uint8_t velocidad;
uint8_t timeToReset;
uint8_t buzzerTime;
const char clientName[] PROGMEM = "NOMBRE CLIENTE"; //MODIFICAR SEGÚN NECESIDAD – 16 CARACTERES MAXIMOS
```

*** Mas de 16 caracteres llevan a error fatal que crashea el programa**

Funciones del Sistema

Función `mostrarBarras`

```
void mostrarBarras(uint8_t valor);
```

Muestra barras en el LCD para indicar la velocidad del motor.

Función `imprimirLCD`

```
void imprimirLCD(char *aImprimir, char altura);
```

Imprime una cadena en el LCD en la posición especificada por `altura`.

Función `my_ceil`

```
int my_ceil(float num);
```

Redondea un número hacia arriba.

Función `my_strlen`

```
char my_strlen(const char *p);
```

Calcula la longitud de una cadena.

Función `updateTimeOnLCD`

```
void updateTimeOnLCD();
```

Actualiza el tiempo mostrado en el LCD.

Función `updateTimeStruct`

```
void updateTimeStruct();
```

Actualiza la estructura de tiempo.

Función `RotaryChanged`

```
void RotaryChanged();
```

Maneja los cambios en el codificador rotatorio.

Configuración Inicial - Setup

La configuración inicial es crucial para preparar el sistema y asegurarse de que todos los componentes estén configurados correctamente.

Inicialización Serial

La comunicación serial se configura a 9600 baudios, lo que permite la salida de mensajes de depuración a través del monitor serial. Esto es útil para monitorear el estado del sistema y diagnosticar problemas.

Configuración de Pines

Los pines utilizados para el motor, buzzer, codificador y LED se configuran como entradas o salidas según sea necesario. Esto asegura que cada componente pueda ser controlado correctamente por el Arduino.

Pines configurados:

- - PIN_ENCODER_DT: Configurado como entrada para recibir señales del codificador rotatorio.
- - PIN_ENCODER_CLK: Configurado como entrada para recibir señales del codificador rotatorio.
- - PIN_ENCODER_SW: Configurado como entrada para recibir señales del botón del codificador.
- - PIN_MOTOR: Configurado como salida para controlar el motor DC.
- - PIN_BUZZER: Configurado como salida para controlar el buzzer.
- - PIN_LED_TESTIGO: Configurado como salida para controlar el LED indicador.

Inicialización del Buzzer

El buzzer se activa inicialmente con una frecuencia predefinida para indicar que el sistema está iniciando. Esto proporciona una señal audible de que el sistema está en funcionamiento.

Inicialización del LCD

El LCD se inicializa y se muestran mensajes iniciales para indicar que el sistema está en proceso de inicio. Esto proporciona retroalimentación visual al usuario sobre el estado del sistema.

Acciones realizadas:

- - Se inicializa la pantalla LCD.
- - Se limpia la pantalla LCD.
- - Se configuran caracteres personalizados para mostrar barras en el LCD.
- - Se muestra el mensaje 'INICIANDO' en la pantalla LCD.
- - Se enciende la luz de fondo del LCD.

Configuración de Estados

Se inicializan varias variables y banderas de estado para asegurar que el sistema comience en el estado correcto y con valores iniciales adecuados. Esto incluye el modo del sistema, contadores de tiempo y estados de los componentes.

Variables y banderas inicializadas:

- - Modo del sistema: Establecido en INITIALIZING.
- - Banderas de control: Inicializadas para reflejar el estado inicial del sistema.
- - Contadores de tiempo: Establecidos a sus valores iniciales.

Bucle principal - Loop

El bucle principal maneja la lógica del sistema en tiempo real, verificando y actualizando el estado del sistema en cada iteración. El loop se ejecuta constantemente, pero el sistema entra a controlar estados dependiendo de la variable MIN_LOOP_MS_TIME, cuando se sobrepasa ese diferencial de tiempo, se actualiza el puntero de tiempo y se verifican estados nuevamente

Manejo del Tiempo

Al encenderse, el sistema entra en modo INITIALIZING. En este modo, se establecen las banderas y variables a sus estados iniciales, se muestra "RADAL" y el nombre del cliente en la pantalla, y luego el sistema pasa a modo READY.

Al girar el encoder, se pueden modificar los valores de minutos de acción, con un máximo de 59 minutos. Después, se puede ajustar la velocidad del motor. Al presionar el botón del encoder, el sistema cambia a modo WORKING. En este modo, el tiempo decrece en el display, y se accionan el motor y el LED testigo.

Si se presiona el botón del encoder nuevamente, el sistema pasa a modo PAUSE. Otra presión del botón del encoder lo devuelve a modo WORKING, pero en este modo, una pulsación larga inicia un conteo para reiniciar el tiempo restante. Presionar el botón del encoder durante este conteo cancela el reinicio y deja el tiempo pausado.

En modo WORKING, si queda tiempo restante, se verifica si ha pasado un segundo y se actualizan los contadores de tiempo. En los últimos segundos, el sistema pasa a modo FINISHING y emite pitidos cada vez que descuenta un segundo. Al llegar a 0, emite un pitido de finalización y vuelve a modo READY, con el tiempo definido previamente al pulsar el encoder.

Manejo del LCD

El LCD se auto apaga después del tiempo definido en su variable, cualquier interacción con el display apagado no tiene otro efecto más que encenderlo. Con el display encendido todas las funciones están activadas

Manejo del Buzzer

Se controla el tiempo y la frecuencia del buzzer para proporcionar señales audibles en diferentes estados del sistema. Esto incluye el inicio, finalización y cancelación de tareas.

Codificador Rotatorio

Se detectan giros y pulsaciones del codificador para ajustar parámetros como el tiempo y la velocidad del motor.

Botón del Codificador

Se detectan presiones y liberaciones del botón del codificador y se actualizan los estados del sistema en consecuencia. Esto permite que el usuario inicie, pause, reinicie o reanude el funcionamiento del motor.

Actualizar Tiempo en el LCD

Se llama a la función de actualización del LCD para mostrar el tiempo restante o mensajes de estado en la pantalla. Esto proporciona una indicación visual del progreso del tiempo y los estados del sistema.

Control del Motor

Se enciende y apaga el motor según el estado del sistema, ajustando su velocidad según los parámetros establecidos.