Keypad with lcd and rtc menu bar

/\* @file EventSerialKeypad.pde

|| @version 1.0

|| @author Alexander Brevig

|| @contact alexanderbrevig@gmail.com

||

|| @description

|| | Demonstrates using the KeypadEvent.

|| #

\*/

#include <TimeLib.h>

#include <Time.h>

#include <Wire.h>

#include <DS1307RTC.h> // a basic DS1307 library that returns time as a time\_t

#include <LiquidCrystal.h>

#include <Keypad.h>

//String menuItems[] = {"PROG 1", "PROG 2", "PROG 3", "PROG 4", "QTY 5", "PROG 6"};

String menuItems[] = {"PROG 1", "PROG 2", "PROG 3", "QTY 5", "PROG 4"};

// Navigation button variables

int readKey;

int key;

// Menu control variables

int menuPage = 0;

int maxMenuPages = round(((sizeof(menuItems) / sizeof(String)) / 2) + .5);

int cursorPosition = 0;

int result = 0;

// Creates 3 custom characters for the menu display

byte downArrow[8] = {

0b00100, // \*

0b00100, // \*

0b00100, // \*

0b00100, // \*

0b00100, // \*

0b10101, // \* \* \*

0b01110, // \*\*\*

0b00100 // \*

};

byte upArrow[8] = {

0b00100, // \*

0b01110, // \*\*\*

0b10101, // \* \* \*

0b00100, // \*

0b00100, // \*

0b00100, // \*

0b00100, // \*

0b00100 // \*

};

byte menuCursor[8] = {

B01000, // \*

B00100, // \*

B00010, // \*

B00001, // \*

B00010, // \*

B00100, // \*

B01000, // \*

B00000 //

};

const byte ROWS = 4; //four rows

const byte COLS = 4; //four columns

char keys[ROWS][COLS] = {

{'1','2','3','A'}

,

{ '4','5','6','B'}

,

{'7','8','9','C'}

,

{'\*','0','#','D'}

};

//byte rowPins[ROWS] = {4, 5, 6, 7}; //connect to the row pinouts of the keypad

//byte colPins[COLS] = {8, 9, 10, 11}; //connect to the column pinouts of the keypad

byte rowPins[ROWS] = { 5, 4, 3, 2 };

// Connect keypad COL0, COL1, COL2 and COL3 to Arduino pins.

byte colPins[COLS] = { 6, 7, 8, 9 };

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

LiquidCrystal lcd(A0,A1,A2,A3,11,10); //Parameters: (rs, enable, d4, d5, d6, d7) pin(A5,A4,A3,A2,A1,A0)

//byte ledPin = 13;

void start\_motor(void);

void program1(void);

void program2(void);

void program3(void);

// for feeder

boolean blink = false;

boolean ledPin\_state;

int selctionQty=0;

int selctionProg=0;

int state;

int relay1 = 10;

int x,y,z;

int feedSetTime = 10;

char feedFlag = 0,startFlagMotor=0;

int feednexttime ;

int feednexttimeH ;

void setup(){

lcd.begin(16, 2);

Serial.begin(9600);

setSyncProvider(RTC.get); // the function to get the time from the RTC

if(timeStatus()!= timeSet)

Serial.println("Unable to sync with the RTC");

else

Serial.println("RTC has set the system time ");

digitalClockDisplay();

delay(1000);

lcd.clear();

keypad.addEventListener(keypadEvent); // Add an event listener for this keypad

lcd.createChar(0, menuCursor);

lcd.createChar(1, upArrow);

lcd.createChar(2, downArrow);

}

void loop(){

x =minute();

y = hour();

z = day();

mainMenuDraw();

drawCursor();

operateMainMenu();

}

// Taking care of some special events.

void keypadEvent(KeypadEvent key){

switch (keypad.getState()){

int readKey;

case PRESSED:

break;

}

}

// This function will generate the 2 menu items that can fit on the screen. They will change as you scroll through your menu. Up and down arrows will indicate your current menu position.

void mainMenuDraw() {

// Serial.print(menuPage);

lcd.clear();

lcd.setCursor(1, 0);

lcd.print(menuItems[menuPage]);

lcd.setCursor(1, 1);

lcd.print(menuItems[menuPage + 1]);

if (menuPage == 0) {

lcd.setCursor(15, 1);

lcd.write(byte(2));

} else if (menuPage > 0 and menuPage < maxMenuPages) {

lcd.setCursor(15, 1);

lcd.write(byte(2));

lcd.setCursor(15, 0);

lcd.write(byte(1));

} else if (menuPage == maxMenuPages) {

lcd.setCursor(15, 0);

lcd.write(byte(1));

}

}

// When called, this function will erase the current cursor and redraw it based on the cursorPosition and menuPage variables.

void drawCursor() {

for (int x = 0; x < 2; x++) { // Erases current cursor

lcd.setCursor(0, x);

lcd.print(" ");

}

// The menu is set up to be progressive (menuPage 0 = Item 1 & Item 2, menuPage 1 = Item 2 & Item 3, menuPage 2 = Item 3 & Item 4), so

// in order to determine where the cursor should be you need to see if you are at an odd or even menu page and an odd or even cursor position.

if (menuPage % 2 == 0) {

if (cursorPosition % 2 == 0) { // If the menu page is even and the cursor position is even that means the cursor should be on line 1

lcd.setCursor(0, 0);

lcd.write(byte(0));

}

if (cursorPosition % 2 != 0) { // If the menu page is even and the cursor position is odd that means the cursor should be on line 2

lcd.setCursor(0, 1);

lcd.write(byte(0));

}

}

if (menuPage % 2 != 0) {

if (cursorPosition % 2 == 0) { // If the menu page is odd and the cursor position is even that means the cursor should be on line 2

lcd.setCursor(0, 1);

lcd.write(byte(0));

}

if (cursorPosition % 2 != 0) { // If the menu page is odd and the cursor position is odd that means the cursor should be on line 1

lcd.setCursor(0, 0);

lcd.write(byte(0));

}

}

}

void operateMainMenu() {

int activeButton = 0;

while (activeButton == 0) {

int button;

button = evaluateButton(readKey);

switch (button) {

case 1: // When button returns as 0 there is no action taken

break;

Serial.println("operate main menu: button= ");

Serial.println(button);

case 3:

button = 0;

if (menuPage == 0) {

cursorPosition = cursorPosition - 1;

cursorPosition = constrain(cursorPosition, 0, ((sizeof(menuItems) / sizeof(String)) - 1));

}

if (menuPage % 2 == 0 and cursorPosition % 2 == 0) {

menuPage = menuPage - 1;

menuPage = constrain(menuPage, 0, maxMenuPages);

}

if (menuPage % 2 != 0 and cursorPosition % 2 != 0) {

menuPage = menuPage - 1;

menuPage = constrain(menuPage, 0, maxMenuPages);

}

cursorPosition = cursorPosition - 1;

cursorPosition = constrain(cursorPosition, 0, ((sizeof(menuItems) / sizeof(String)) - 1));

mainMenuDraw();

drawCursor();

activeButton = 1;

break;

case 4: // This case will execute if the "forward" button is pressed

button = 0;

switch (cursorPosition) { // The case that is selected here is dependent on which menu page you are on and where the cursor is.

case 0:

menuItem1();

break;

case 1:

menuItem2();

break;

case 2:

menuItem3();

break;

case 3:

menuItem4();

break;

case 4:

menuItem5();

break;

case 5:

menuItem6();

break;

}

activeButton = 1;

mainMenuDraw();

drawCursor();

break;

case 5:

button = 0;

if (menuPage % 2 == 0 and cursorPosition % 2 != 0) {

menuPage = menuPage + 1;

menuPage = constrain(menuPage, 0, maxMenuPages);

}

if (menuPage % 2 != 0 and cursorPosition % 2 == 0) {

menuPage = menuPage + 1;

menuPage = constrain(menuPage, 0, maxMenuPages);

}

cursorPosition = cursorPosition + 1;

cursorPosition = constrain(cursorPosition, 0, ((sizeof(menuItems) / sizeof(String)) - 1));

mainMenuDraw();

drawCursor();

activeButton = 1;

break;

}

}

}

// This function is called whenever a button press is evaluated. The LCD shield works by observing a voltage drop across the buttons all hooked up to A0.

int evaluateButton(int x) {

int result = 0;

int key = keypad.getKey();

if (key < 57) {

delay(100);

}

x=key;

if (x < 51) {

result = 1; // right

} else if (x < 53) {

result = 2; // up

} else if (x < 54) {

result = 3; // down

} else if (x < 55) {

result = 4; // left

} else if (x < 57) {

result = 5; // left

}

return result;

}

// If there are common usage instructions on more than 1 of your menu items you can call this function from the sub

// menus to make things a little more simplified. If you don't have common instructions or verbage on multiple menus

// I would just delete this void. You must also delete the drawInstructions()function calls from your sub menu functions.

void drawInstructions() {

lcd.setCursor(0, 1); // Set cursor to the bottom line

lcd.print("Use ");

lcd.print(byte(1)); // Up arrow

lcd.print("/");

lcd.print(byte(2)); // Down arrow

lcd.print(" buttons");

}

// prog1 selection

void menuItem1() { // Function executes when you select the 1st item from main menu

int activeButton = 0;

lcd.clear();

lcd.setCursor(3, 0);

lcd.print("PROG 1");

program1();

while (activeButton == 0) {

int button;

button = evaluateButton(readKey);

switch (button) {

case 2: // This case will execute if the "back" button is pressed

button = 0;

activeButton = 1;

break;

}

}

}

void menuItem2() { // Function executes when you select the 2nd item from main menu

int activeButton = 0;

lcd.clear();

lcd.setCursor(3, 0);

lcd.print("PROG 2");

while (activeButton == 0) {

int button;

button = evaluateButton(readKey);

switch (button) {

case 2: // This case will execute if the "back" button is pressed

button = 0;

activeButton = 1;

break;

}

}

}

void menuItem3() { // Function executes when you select the 3rd item from main menu

int activeButton = 0;

lcd.clear();

lcd.setCursor(3, 0);

lcd.print("PROG 3");

while (activeButton == 0) {

int button;

button = evaluateButton(readKey);

switch (button) {

case 2: // This case will execute if the "back" button is pressed

button = 0;

activeButton = 1;

break;

}

}

}

void menuItem4() { // Function executes when you select the 4th item from main menu

int activeButton = 0;

int qty = 0;

lcd.clear();

lcd.setCursor(3, 0);

lcd.print("Enter qty");

while (activeButton == 0) {

int button;

button = evaluateButton(readKey);

switch (button) {

case 2: // This case will execute if the "back" button is pressed

button = 0;

activeButton = 1;

break;

}

qty=evaluateButton\_qty(readKey);

switch (qty) {

case 1: // This case will execute if the "back" button is pressed

button = 0;

activeButton = 1;

break;

}

Serial.println("select of qty in menu item4");

Serial.println(qty);

}

}

void menuItem5() { // Function executes when you select the 5th item from main menu

int activeButton = 0;

while (activeButton == 0) {

int button;

button = evaluateButton(readKey);

switch (button) {

case 2: // This case will execute if the "back" button is pressed

button = 0;

activeButton = 1;

break;

}

}

}

void menuItem6() { // Function executes when you select the 6th item from main menu

int activeButton = 0;

lcd.clear();

lcd.setCursor(3, 0);

lcd.print("Sub Menu 6");

while (activeButton == 0) {

int button;

button = evaluateButton(readKey);

switch (button) {

case 2: // This case will execute if the "back" button is pressed

button = 0;

activeButton = 1;

break;

}

}

}

void digitalClockDisplay(){

// digital clock display of the time

Serial.print(hour());

printDigits(minute());

printDigits(second());

Serial.print(" ");

Serial.print(day());

Serial.print(" ");

Serial.print(month());

Serial.print(" ");

Serial.print(year());

Serial.println();

lcd.setCursor(0, 0); // Set LCD cursor position (column, row)

lcd.print(hour());

lcd.print(":");

lcd.print (minute());

lcd.print(":");

lcd.print(second());

lcd.print(" ");

delay(1000);

//lcd.setCursor(0, 0);

//

//lcd.print(" "); // to clear lcd

//delay(10);

//lcd.setCursor(0, 1); // Set LCD cursor position (column,row)

//lcd.print(day());

//lcd.print(".");

//lcd.print(month());

//lcd.print(".");

//lcd.print(year());

//delay(1000);

//lcd.setCursor(0, 1);

//lcd.print(" "); // to clear lcd

delay(10);

// Print text to LCD

// Delay to read text

// Clear the display

}

void printDigits(int digits){

// utility function for digital clock display: prints preceding colon and leading 0

Serial.print(":");

if(digits < 10)

Serial.print('0');

Serial.print(digits);

}

void program1(void)

{

digitalClockDisplay();

lcd.setCursor(9,0);

lcd.print(" P1 ");

delay(100);

Serial.println("enter in program1 ");

if(startFlagMotor ==0)

{

start\_motor();

startFlagMotor = 1;

}

if( feedFlag == 0)

{

Serial.println("current time in hours =");

Serial.println(y);

Serial.println("current time in minutes =");

Serial.println(x);

feednexttime = x+feedSetTime;

feednexttimeH = y;

if (feednexttime == 59 )

{

feednexttime = 0;

feednexttimeH =feednexttimeH + 1;

if(feednexttimeH > 23){

feednexttimeH = feednexttimeH - 24;}

}

else if (feednexttime > 59)

{

feednexttime = feednexttime - 60;

feednexttimeH = feednexttimeH+1;

if(feednexttimeH > 23){

feednexttimeH = feednexttimeH - 24;}

}

digitalClockDisplay();

Serial.println("feed set time in hours");

Serial.println(feednexttimeH );

Serial.println("feed set time in minutes");

Serial.println(feednexttime );

//Serial.println("enter in program1 ");

lcd.setCursor(0, 1);

lcd.print("NEXT TIME ");

lcd.setCursor(11, 1);

lcd.print(feednexttimeH);

lcd.print(":");

lcd.print(feednexttime);

delay(2000);

feedFlag = 1;

}

if( x == feednexttime && y== feednexttimeH)

{

start\_motor();

feedFlag =0;

}

}

void start\_motor(void)

{

lcd.setCursor(0, 1);

if (selctionQty == 1)

{

// on for 1 msec

// lcd.setCursor(1,1);

// lcd.print("count 1");

lcd.print("MOTOR ON 1kg ");

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

}

else if (selctionQty == 2)

{

lcd.print("MOTOR ON 2kg ");

// on for 2 msec

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

digitalWrite(10,HIGH);

delay(10000);

digitalWrite(10,LOW);

delay(3000);

}

}

int evaluateButton\_qty(int x) {

int result = 0;

int key = keypad.getKey();

selctionQty=key;

Serial.println("select of qty");

Serial.println(selctionQty);

return selctionQty;

}

Key pad with lcd;

#include <LiquidCrystal.h>

#include <Keypad.h>

//initialize the library with the numbers of the interface pins

LiquidCrystal lcd(A0,A1,A2,A3,A4,A5);

//4x4 Matrix key pad

const byte ROWS = 4; // Four rows

const byte COLS = 4; // Four columns

// Define the Keymap

char keys[ROWS][COLS] =

{

{'7','8','9','/'},

{'4','5','6','X'},

{'1','2','3','-'},

{'#','0','=','+'}

};

// Connect keypad ROW0, ROW1, ROW2 and ROW3 to Arduino pins.

byte rowPins[ROWS] = { 5, 4, 3, 2 };

// Connect keypad COL0, COL1, COL2 and COL3 to Arduino pins.

byte colPins[COLS] = { 6, 7, 8, 9 };

// Create the Keypad

Keypad kpd = Keypad( makeKeymap(keys), rowPins,

colPins, ROWS, COLS );

void setup()

{

// set up // the LCD's number of columns and rows:

lcd.begin(16, 2);

// Print a message to the LCD.

lcd.print("KEYPAD LCD TEST");

}

void loop()

{

char key = kpd.getKey();

// set the cursor to column 0, line 1

// (note: line 1 is the second row, since counting begins with 0):

lcd.setCursor(0, 1);

// print the number of seconds since reset:

lcd.print("SEC= ");

lcd.print(millis() / 1000);

// Check for a valid key

if(key)

{

// set the cursor to column 9, line 1

// (note: line 1 is the second row, since counting begins with 0):

lcd.setCursor(9, 1);

//Print the detected key

lcd.print("KEY= ");

lcd.print(key);

}

}