ACTIVITY 9B: Liquid Crystal Display (Digital Watch)

Jason Harvey Lorenzo, April 16, 2024
ENGG 122.02, Department of Electronics, Computer, and Communications Engineering
School of Science and Engineering, Ateneo de Manila University
Quezon City, 1108, Philippines
jason.lorenzo@obf.ateneo.edu

***Abstract*—This project introduces a simple digital watch created using an Arduino microcontroller and a 1602 Liquid Crystal Display (LCD). The LCD's first row serves to display the author's name, while the second row showcases the time in the 12-hour format (HH:MM:SS AM/PM). Two push buttons, connected to digital pins 2 and 3 of the Arduino, enable adjustment of the hour and minute values, respectively. The code initializes the LCD and sets up interrupt functions for the push buttons to facilitate time adjustment. Each button press triggers an interrupt, allowing for incrementing the hour or minute values, with consideration given to rollovers. The loop() function ensures accurate timekeeping by updating the seconds, minutes, and hour values, while also handling rollovers and formatting the time display. Additionally, the Arduino setup involves connecting the LCD and push buttons to the appropriate pins, with resistors and capacitors incorporated to address debounce issues. Overall, the project combines hardware and software components to create a functional digital watch with user-friendly time adjustment capabilities.**

***Index Terms*—Arduino, LCD, digital watch**

I. Introduction

This project involves creating a simple digital watch using a 1602 Liquid Crystal Display (LCD) and an Arduino microcontroller. The LCD's first row will present the author's first and last name, while the second row will display the time in the 12-hr format: HH:MM:SS AM/PM. To facilitate time adjustment, two push buttons will be employed. Their respective functionalities are detailed in Table I.

TABLE I
Button Functions

Button	Function
PB1 (pin 2)	Increment the Hour by 1 (should automatically cycle AM/PM if it reaches the 12th hour)
PB2 (pin 3)	Increment the Minute by 1 (should automatically adjust the Hour if it reaches 60 minutes)

II. Program Listing

A. Source Code

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

int time_before, time_after;
int seconds = 0;
int minutes = 0;
int hour = 12;
int lcd_refresh_rate = 100;
String period = "AM";

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.print("Jason■Lorenzo");
  time_before = millis();
  attachInterrupt(
    digitalPinToInterrupt(2),
    increment_hour, RISING);
  attachInterrupt(
    digitalPinToInterrupt(3),
    increment_minutes, RISING);
}

void increment_hour(){
  noInterrupts();
  check_period();
  hour++;
  check_hour();
}

void increment_minutes(){
  noInterrupts();
  minutes++;
  check_minutes();
  check_hour();
}

void loop() {
  time_after = millis();
  if (time_after-time_before >= 1000){
    seconds++;
    if (seconds >= 60){
      seconds = 0;
      minutes++;
    }
    check_minutes();
```

```
      check_hour();
      time_before = time_after;
    }
    lcd.setCursor(0,1);
    if (hour < 10) lcd.print("0");
    lcd.print(hour);
    lcd.print(":");
    if (minutes < 10) lcd.print("0");
    lcd.print(minutes);
    lcd.print(":");
    if (seconds < 10) lcd.print("0");
    lcd.print(seconds);
    lcd.print("■");
    lcd.print(period);
    delay(lcd_refresh_rate);
}

void check_minutes(){
  if (minutes >= 60){
    minutes = 0;
    check_period();
    hour++;
  }
}

void check_hour(){
  if (hour >= 13){
    hour = 1;
  }
}

void check_period(){
  if (hour == 11){
    if (period == "AM")
      period = "PM";
    else if (period == "PM")
      period = "AM";
  }
}
```

B. How It Works

This code initializes a digital watch using a 1602 LCD and incorporates two push buttons for time adjustment. To facilitate LCD operations, the LiquidCrystal_I2C library is imported and utilized. The time components, including seconds, minutes, and hours, are stored as integer variables, while the period (AM/PM) is stored as a string for manipulation. Two additional variables, time_before and time_after, are used to track each passing second accurately via the millis() function.

The LCD's first row is initialized to display the author's name (since this value won't change later), while the second row is reserved for displaying the time in the format "HH:MM:SS AM/PM". Two push buttons connected to digital pins 2 and 3 of the Arduino are utilized for adjusting the hour and minute values, respectively, with each button triggering an interrupt to increment the corresponding values when pressed (taking rollovers into consideration as well).

In the loop() function, the time_after variable is updated with the current millis() value, allowing for accurate timing. If one second has elapsed since the last update, the seconds variable is incremented, and if it reaches 60, it resets to 0, and minutes are incremented accordingly. The check_minutes() function is called to handle minute rollovers, and the check_hour() function is called to handle hour rollovers. The time_before variable is updated to maintain the timing loop.

The LCD is then updated with the current time and period. Leading zeros are added to ensure consistent formatting of the time display. The hour, minute, and second values are printed to the LCD, followed by a space and the current period (AM or PM).

III. Arduino Setup

The Arduino setup, as illustrated in Fig. 1, showcases an Arduino UNO connected to a laptop via a USB cord, serving dual purposes of power supply and program uploading. The LCD's VCC and GND connections are linked to the 5V and GND pins, respectively. Additionally, the SCA and SCL pins of the LCD are connected to the A4 and A5 pins, respectively, as per default configuration from the library. Interrupt pins 2 and 3 are connected to their respective push buttons, with 1-kΩ resistors ensuring pull-up configuration. Furthermore, to mitigate debounce issues, a 33 μF electrolytic capacitor is integrated into each input circuit, allowing current flow only upon a change in current, induced momentarily by the button switch action.

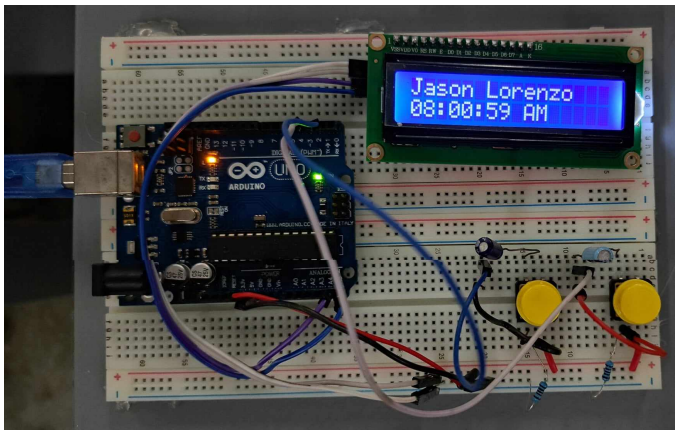

Fig. 1. Arduino setup with 1602 LCD and Push Button Inputs