

آزمایش ۷ - ریزپردازند

سپهر توکلی - ۹۸۳۱۱۱۱

محمد چوپان - ۹۸۳۱۱۲۵

۱) در چه کاربردهایی EEPROM به کار برد میشود ؟ چرا در اینجا حافظه فلش یا RAM به کار نمیبریم ؟ تفاوت حافظه RAM با EEPROM چیست ؟ از EEPROM برای ذخیره دائمی دیتا استفاده میشود چون این نوع حافظه، بدون جریان الکتریکی (وقتی دستگاه خاموش است) نیز اطلاعات را در خود نگه میدارد. دیتاهای دائمی یک سیستم کامپیوتری مثل آیدی ها و یا پسورد ها در EEPROM ذخیره میشود. تفاوت EEPROM و Flash در نحوه Overwrite کردن آن است، این اتفاق در RAM به صورت بلوکی انحصار میشود یعنی کل اطلاعات پاک میشوند و دوباره نوشته میشوند. اما در EEPROM اطلاعات به صورت بایت به بایت نوشته میشوند. پس در این آزمایش چون نیاز داریم برخی متغیر ها را به صورت بایت به بایت بازنویسی کنیم، پس از EEPROM استفاده میکنیم نه از Flash. و همچنین چون میخواهیم حافظه دائمی داشته باشیم و با خاموش شدن دستگاه اطلاعات کاربر پاک نشود پس از RAM استفاده نمیکنیم چون حافظه غیر دائم است و با قطع شدن برق اطلاعات داخل آن پاک میشوند. تفاوت اصلی RAM و EEPROM در این است که حافظه غیر دائمی است ولی EEPROM حافظه دائمی است .

۲) اگر بخواهیم برای نگهداری مدهای کاری حافظه Flash را به کار ببریم، فرآیند نوشتن باید چگونه انجام شود که داده های دیگری که بر روی همان بلاک هستند از دست نروند؟ احتمالا باید ایتدا داده های آن بلاک را بخوانیم و قسمت هایی که میخواهیم را تغییر دهیم و کل آن بلاک را دوباره بنویسیم. مثلا میتوان یکی از دو راه زیر را انجام داد:

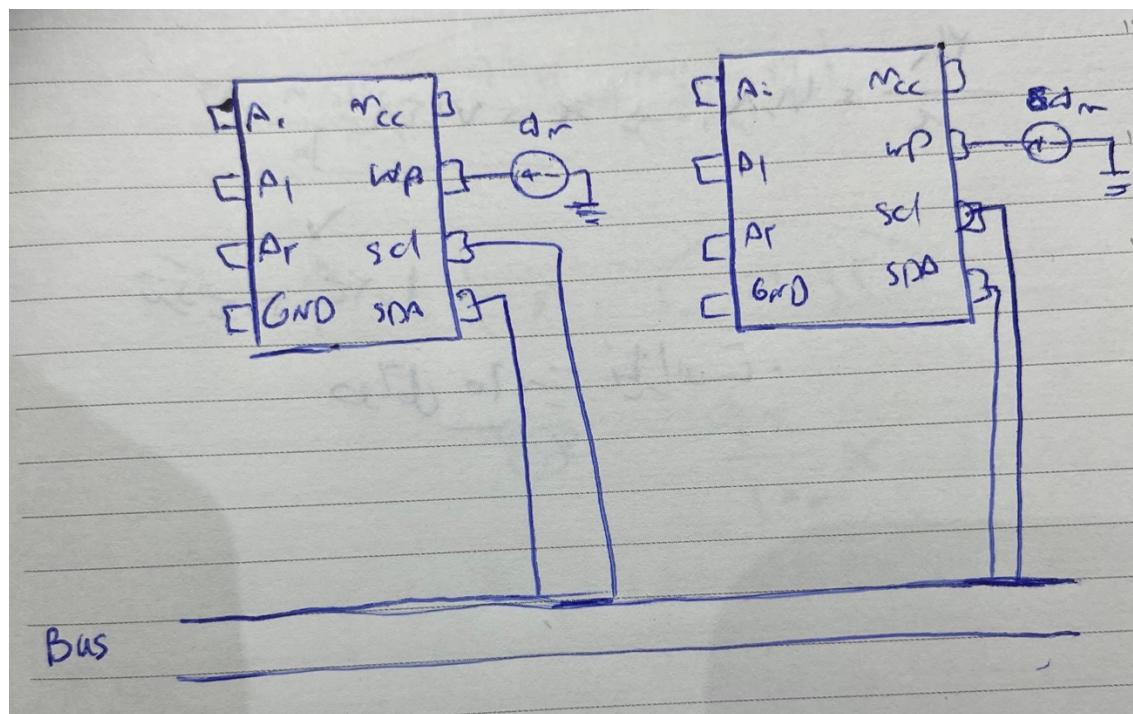
۱- محتويات صفحه Flash را در RAM داخلی کپی کرده و به روزرسانی ها را در RAM اعمال کنید. سپس صفحه Flash را پاک کرده و مطالب را از RAM دوباره بنویسیم.

۲- محتوای صفحه Flash را که با به روزرسانی ها ادغام شده است، در یک صفحه فلاش که قبلاً پاک شده است کپی کنیم. بعداً، هنگامی که دستگاه بیکار است، صفحاتی را که منسوخ شده اند پاک کنیم.

۳) اگر حافظه EEPROM بیرونی دارای 4KB حافظه و 2 پایه آدرس باشد، در این صورت میتوان حداکثر چند KB حافظه EEPROM بیرونی بر روی یک باس مشترک داشت؟

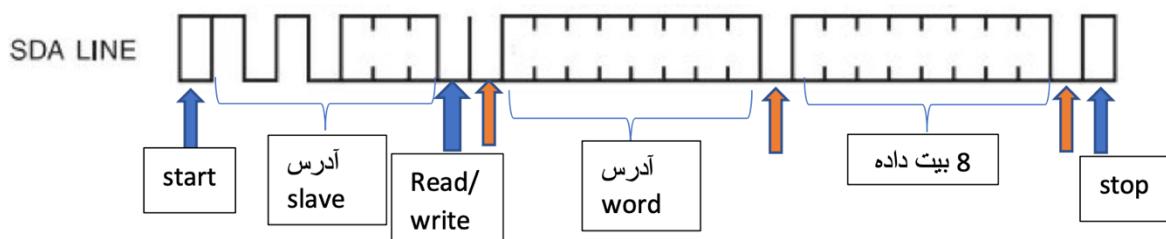
با دو پایه آدرس یعنی میتوانیم 4 slave داشته باشیم که روی یک باس مشترک اند. هر یک از 16KB حافظه ها هم 4KB حافظه میتوانند در اختیار ما قرار دهند پس در مجموع میتوان حداکثر 4 slave حافظه EEPROM بیرونی بر روی یک باس مشترک داشت.

۴) نمودار شماتیک برای اینکه دو AT24C02 را به یک باس مشترک وصل کنیم و حفاظت نوشتن غیر فعال باشد را رسم کنید. (آدرسدهی سخت افزاری دلخواه - باس را هم به پایه های میکروکنترلر متصل کنید.)

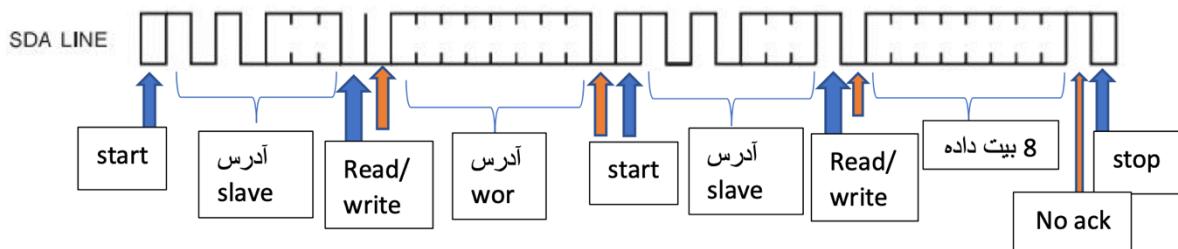


۵) همخوانی این دنباله فریم ها را با پروتکل TWI بررسی کنید. (فریمهای آدرس و داده را مشخص کنید. دستور خواندن و نوشتن چگونه مشخص میشود؟) بیتی که بعد از آدرس دستگاه فرستاده میشود مشخص کنندهٔ خواندن و یا نوشتمن است. اگر ۰ باشد یعنی میخواهد بنویسد و اگر یک باشد یعنی میخواهد بخواند.

نوشتمن بایتی: فلش های نارنجی Ack هستند:



خواندن بایت تصادفی: فلش های نارنجی Ack هستند:



۶) فرکانس کلاک در کدام دستگاه پیکربندی میشود؟ کلاک را کدام دستگاه فراهم میکند؟ با توجه به زمان نیاز برای انجام عملیات نوشتن، با فرض اینکه کلاک را 10 KHz تنظیم کرده باشیم، در این صورت حداقل با چه نرخی میتوان عملیات نوشتن را انجام داد؟

فرکانس کلاک در دستگاه master در این آزمایش میکروی ما) پیکربندی و فراهم میشود. با فرض فرکانس کلاک 10000 هرتز، با توجه به اینکه برای نوشتن یک بایت 29 کلاک نیاز است(طبق شکل نوشتن بایتی در سوال قبل)، یک عملیات نوشتن را در $29 \times 10^{-4} \text{ s}$ میتوان انجام داد.

۷) هر یک از تابع های نوشته شده را از راه لینک کتابخانه wire، در مستندات آردوبینو بررسی کنید و کد لازم را برای تولید دنباله‌ی فریم ها برای عملیات نوشتن و خواندن گفته شده (با این تابع ها) بنویسید.

: کتابخانه Wire را آغاز کرده و به عنوان slave یا master به گذرگاه I2C میپیوندد. به طور معمول فقط یک بار باید صدا زده شود. به میکرو میگوید که میخواهیم از I2C استفاده و آن را فعال کند. اگر در ورودی تابع چیزی ندهیم یعنی این دستگاه master است. اما اگر در ورودی عدد دهیم یعنی slave است و عدد ورودی آدرس آن است.

setClock : این تابع فرکانس کلاک را برای ارتباطات I2C تعیین میکند. دستگاه های I2C slave حداقل فرکانس کاری کلاک ندارند ، با این وجود 100 KHz معمولاً خط پایه است . beginTransmission : بیت start و آدرس Slave و بیت write را ارسال میکند. مقدار ورودی اش آدرس دستگاهیست که میخواهیم با آن ارتباط برقرار کنیم.

Write : یک بایت روی باس ارسال میکند. مقدار تعداد بایتی که ارسال کرده را خروجی میدهد. ورودی آن میتواند یک مقدار یک بایتی یا یک string که به صورت مجموعه ای از بایت ها فرستاده شود و یا یک دیتا که بایت به بایت ارسال شود باشد.

true : ارتباط را تمام میکند. به عنوان ورودی یک Boolean میگیرد که اگر endTransmission باشد بیت stop را میفرستد. و اگر false باشد پس از ارسال، پیام راه اندازی مجدد را ارسال میکند bus. آزاد نمی شود. و باعث میشود تا master دیگری نتواند bus را اشغال کند.

requestFrom: وقتی master بخواهد دیتا بخواند به جای write ، از این تابع استفاده میکند. دو آرگومان ورودی میگیرد. اولی آدرس slave ای که میخواهد از آن بخواند را مشخص میکند. دومی مشخص میکند که چند بایت میخواهد بخواند.(بیت start و stop را هم ارسال میکند)

available : تعداد بایت های موجود برای بازیابی با read را مشخص میکند.

read: برای خواندن یک بایت داده ای که از سمت slave می آید استفاده میشود.

```
void eeprom_write(uint8_t memory_address, uint8_t* data, int _size){
    Serial.println("--- Writing ---" + String(memory_address));
    Wire.beginTransmission(DEVICE_ADDRESS);
    Wire.write(memory_address);

    for(int i=0; i<_size; i++){
        Wire.write(data[i]);
        // Serial.print(data[i]+ " ");
    }

    Wire.endTransmission();
    // Serial.println("\n--- Start Finished ---");
}

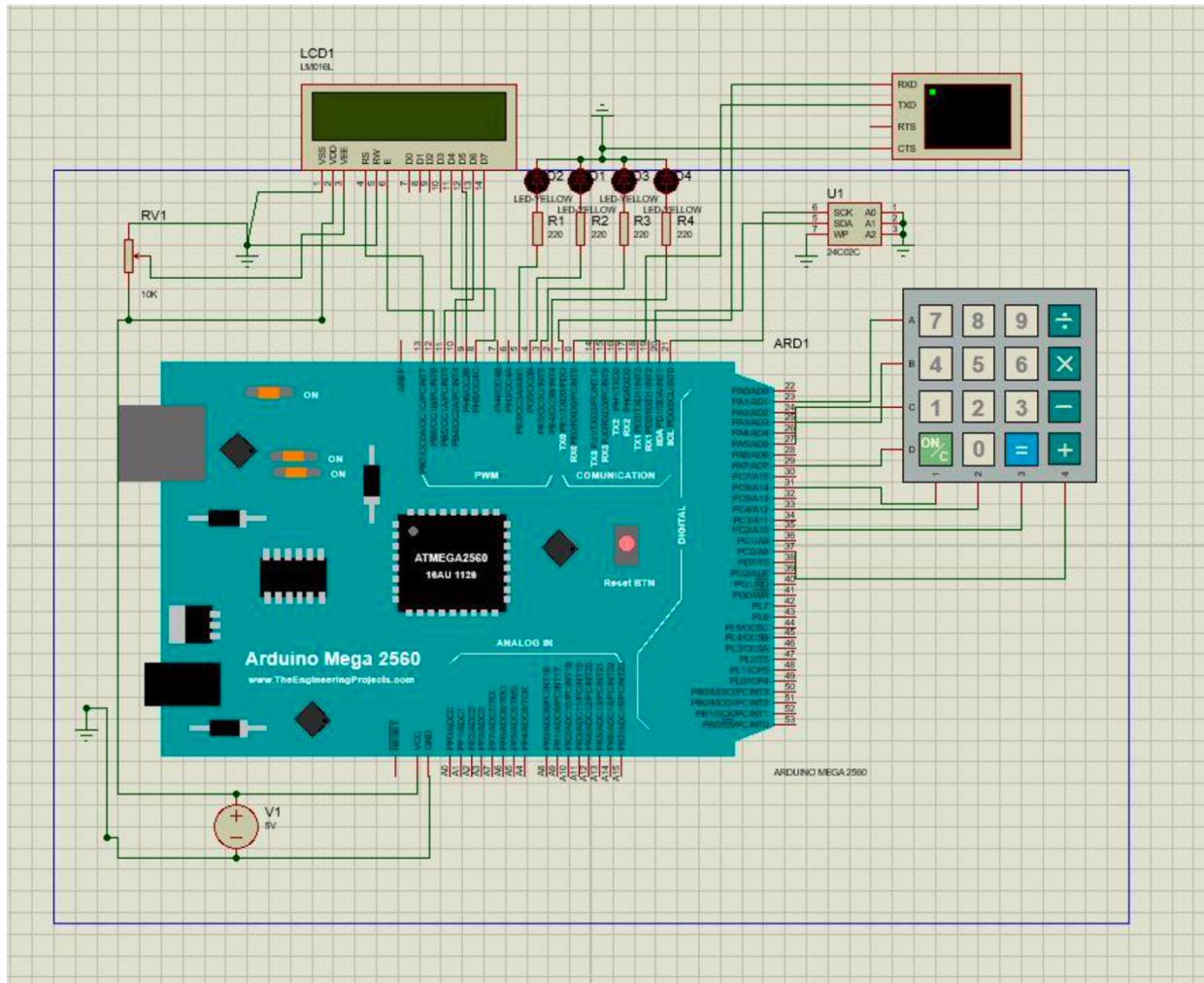
void eeprom_read(uint8_t memory_address, uint8_t *data, uint8_t _size){
    // Serial.println("Accessing address for reading...");

    Wire.beginTransmission(DEVICE_ADDRESS);
    Wire.write(memory_address);
    Wire.endTransmission();

    // Serial.println("--- Start reading ---");
    Wire.requestFrom(DEVICE_ADDRESS, _size);
    for(int i=0; i<_size; i++){
        data[i] = Wire.read();
        // Serial.print(data[i]+ " ");
    }
}
```

شرح آزمایش :

مدار به شکل زیر است



کد آزمایش :

```
#include <Wire.h>
#include <Keypad.h>
#include <LiquidCrystal.h>

#define DEVICE_ADDRESS 0b1010000
#define MODE_MEMORY_ADDR 100

#define PRE_WASH 0
#define DETERGENT_WASH 1
#define WATER_WASH 2
#define DRYING 3
#define FINISH 4
int ledPins[4] = {5, 4, 3, 2};
uint8_t modeTimes[4] = {4, 4, 4, 4}; //in seconds
String modeMessages[5]={"PRE", "DETERGENT", "WATER", "DRYING", "FINISH"};

//Setting LCD configs
const int rs = 13, en = 12, d4 = 8, d5 = 9, d6 = 10, d7 = 11;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

//Setting Keypad configs
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] = {
    {'7','8','9','/'},
    {'4','5','6','*'},
    {'1','2','3','-'},
    {'C','0','=','+'}
};
byte rowPins[ROWS] = {23, 25, 27, 29}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {31, 33, 35, 37}; //connect to the column pinouts of the keypad

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

```
int currMode = PRE_WASH;
bool isWashing = false;
int remainingTime;
unsigned long startTime = millis();
void setup() {
    Serial.begin(9600);

    for(int i=0; i<FINISH; i++) {
        pinMode(ledPins[i], OUTPUT);
    }
    Wire.begin();

    lcd.begin(16, 2);
    lcd.clear();

    loadConfigs();
}

void loadConfigs(){
    Serial.println("##### Loading Saved Data #####");
    uint8_t savedData[5];
    eeprom_read(MODE_MEMORY_ADDR, savedData, 5);
    if((char)savedData[0] >= '0' && savedData[0] <= '4')
        currMode = uint8ToInt(savedData[0]);

    printLCD(currMode);

    for(int i=1; i<=4; i++){
        if((char)savedData[i] > '0'){
            modeTimes[i-1] = uint8ToInt(savedData[i]);
        }
    }
    turnOnLED(currMode);

    Serial.println("Current Mode is (" + modeMessages[currMode] + ")");
    Serial.println("Mode Times:");
    for(int i=0; i<FINISH; i++){
        Serial.println(modeMessages[i]+ "(" + modeTimes[i] + ")");
    }
}
```

```
void printLCD(int currMode){
    if(currMode > -1) {
        lcd.setCursor(0, 0);
        lcd.print("                ");
        lcd.setCursor(0, 0);
        lcd.print(modeMessages[currMode]);
    }
}

bool isKeyNumber(char key){
    int keyAsciiCode = (int)key;
    return keyAsciiCode >= (int)'0' && keyAsciiCode <= (int)'9';
}

void turnOnLED(int currMode){
    if(currMode == FINISH) {
        for(int i=PRE_WASH; i<FINISH; i++){
            digitalWrite(ledPins[i], HIGH);
        }
    }else{
        for(int i=PRE_WASH; i<FINISH; i++){
            digitalWrite(ledPins[i], LOW);
        }
        digitalWrite(ledPins[currMode], HIGH);
    }
}

int charToInt(char key){
    return (int)key - 48;
}

int uint8ToInt(uint8_t in){
    return in - 48;
}
```

```
'\n\nvoid readKeypad(){\n    char key = keypad.getKey();\n    static char lastKey;\n\n    if(key){\n        Serial.println(key);\n        if(isKeyNumber(key)){\n            lastKey = key;\n        }\n        else if(key == '='){//Save in EEPROM\n            if(charToInt(lastKey)>4)\n                return;\n            currMode = charToInt(lastKey);\n            turnOnLED(currMode);\n            uint8_t tempData[1] = {(uint8_t)lastKey}//ascii code\n\n            eeprom_write(MODE_MEMORY_ADDR, tempData, 1);\n            printLCD(currMode);\n            lcd.setCursor(0, 1);\n            lcd.print("                 ");\n        }else if(key == 'C'){//Change Time\n            modeTimes[currMode] = charToInt(lastKey);\n            uint8_t tempData[1] = {(uint8_t)lastKey};\n            eeprom_write(MODE_MEMORY_ADDR + 1 + currMode, tempData, 1);\n        }else if(key == '/'){\n            remainingTime -= millis() - startTime;\n            if(!isWashing)\n                isWashing = true;\n            else\n                isWashing = false;\n        }else if(key == '*'){\n            loadConfigs();\n            isWashing = true;\n        }\n    }\n}\n\n'
```

```
}

void eeprom_write(uint8_t memory_address, uint8_t* data, int _size) {
    Serial.println("--- Writing ---" + String(memory_address));
    Wire.beginTransmission(DEVICE_ADDRESS);
    Wire.write(memory_address);

    for(int i=0; i<_size; i++){
        Wire.write(data[i]);
    }

    Wire.endTransmission();
}

void eeprom_read(uint8_t memory_address, uint8_t *data, uint8_t _size){

    Wire.beginTransmission(DEVICE_ADDRESS);
    Wire.write(memory_address);
    Wire.endTransmission();

    Wire.requestFrom(DEVICE_ADDRESS, _size);
    for(int i=0; i<_size; i++){
        data[i] = Wire.read();
    }

}

void checkTime(){
    static int lastMode = -1;
    int passedTime = 0;

    if(!isWashing){
        startTime = millis();
        return;
    }

    if(lastMode != currMode){
        lastMode = currMode;
        startTime = millis();
        remainingTime = modeTimes[currMode] * 1000;
    }else if(currMode == FINISH){
        lastMode = -1;
        isWashing = false;
        loadConfigs();
    }
}
```

```
    remainingTime = modeTimes[currMode] * 1000;
}else if(currMode == FINISH){
    lastMode = -1;
    isWashing = false;
    loadConfigs();
    return;
}else{
    int tempTime= (millis() - startTime) % 1000;
    if(tempTime <= 10 || tempTime >= 990){
        passedTime = ((millis() - startTime)/1000);

        //SHOW TIME
        lcd.setCursor(0, 1);
        lcd.print("          ");
        lcd.setCursor(0, 1);
        lcd.print(String(remainingTime/1000 - passedTime)+"s");

    }
    if(remainingTime/1000 - ((millis() - startTime)/1000) <= 0){
        currMode = (currMode + 1) % (FINISH+1);  
    }

    uint8_t tempData[1] = {currMode + 48}; //ascii code
    eeprom_write(MODE_MEMORY_ADDR, tempData, 1);

    printLCD(currMode);
    turnOnLED(currMode);
    lcd.setCursor(0, 1);
    lcd.print("          ");

    if(currMode == 4){
        isWashing = false;
    }
}

void loop() {
    readKeypad();
    checkTime();
}
```