SMARTLOCK_RFID_AVR

Simulation & Analysis

Nariman Ziaie
August 2024

فهرست مطالب

معرفی پروژه

۱-۱ هدف از انجام پروژه

هدف کلی پروژه: ارتقای دانش برنامه‌نویسی AVR و آشنایی با محیط توسعه CodeVisionAVR

هدف اصلی از انجام این پروژه، ارتقای دانش و مهارت برنامه‌نویسی میکروکنترلرهای AVR با تمرکز بر استفاده از محیط توسعه قدرتمند CodeVisionAVR بود که این هدف با طراحی و پیاده‌سازی یک پروژه کاربردی با استفاده از میکروکنترلر ATmega32 محقق شد.

در این راستا، بر روی موارد زیر تمرکز شد:

- ✓ **آشنایی با معماری AVR : درک عمیق‌تر از معماری AVR و میکروکنترلر ATmega32**
- ✓ **زبان برنامه‌نویسی C : ارتقای مهارت برنامه‌نویسی C با تمرکز بر برنامه‌نویسی میکروکنترلرها**
- ✓ **استفاده از CodeVisionAVR : آشنایی با محیط نرم‌افزار CodeVisionAVR و ابزارهای مختلف آن برای برنامه‌نویسی میکروکنترلرهای ساخت شرکت Atmel / Microchip**
- ✓ **برقراری ارتباط با LCD کاراکتری : یادگیری نحوه برقراری ارتباط با نمایشگر LCD (16x2)**
- ✓ **عملکرد USART: یادگیری و تسلط بر استفاده از USART برای برقراری ارتباط سریال با ماژول‌های خارجی، از جمله ماژول RFID.**
- ✓ **عملکرد ADC: آشنایی با نحوه استفاده از ADC برای خواندن سیگنال‌های آنالوگ و تبدیل آن‌ها به مقادیر دیجیتال، از جمله اندازه‌گیری دمای محیط.**
- ✓ **Interrupt: آشنایی با وقفه ها و ترتیب عملکرد آنها**

۲-۱ نحوه کارکرد مدار به طور خلاصه

مدار این پروژه از میکروکنترلر **ATmega32** به عنوان کنترل‌کننده اصلی استفاده می‌کند.

نمایشگر **LCD** برای نمایش زمان، دما و وضعیت قفل به کار رفته است که اطلاعات دما به کمک سنسور **LM35** و مبدل آنالوگ به دیجیتال **(adc)** داخلی میکرو در پورت **A0** دریافت میشود.
همینطور ماژول **RFID** برای خواندن تگ‌های **RFID** از **USART** استفاده می‌کند و زمانی که کاربر کارت (تگ) **RFID** را نزدیک به ماژول قرار می‌دهد، کد کارت خوانده شده و با کدهای از پیش تعریف شده در حافظه سیستم مقایسه می‌شود. در صورت تطبیق، میکروکنترلر سیگنالی را به **LED** یا رله ارسال می‌کند که منجر به باز شدن قفل می‌شود اما در غیر این صورت، قفل بسته باقی می‌ماند و پیامی مبنی بر نامعتبر بودن کارت بر روی نمایشگر **LCD** نمایش داده می‌شود.

در این پروژه به دلیل شبیه‌سازی در نرم‌افزار پروتئوس به‌جای استفاده از ماژول یا دستگاه **RFID reader** از یک ترمینال مجازی **(Virtual Terminal)** استفاده شده و **id** مشخصه یک کارت را به صورت دستی به میکرو ارسال میکنیم و میکرو کد دریافت شده را با کد معتبری که از پیش تعریف شده مقایسه میکند.

لازم به ذکر هست که برای جلوگیری از اسکن مداوم و یا ناخواسته تگ از وقفه داخلی **INT0** استفاده شده که حساس به لبه پایین رونده میباشد، در این سناریو هر زمان که کاربر نیاز داشته باشد کارتی را اسکن نماید با فشردن کلید ۵ ثانیه زمان دارد تا اقدام نماید و پس از گذشت این زمان برنامه به روند کاری معمول خود که همان نمایش ساعت و دما میباشد برمیگردد.

البته این مدار مشکلاتی دارد که در بخش سخت افزار به طور کامل به آنها پرداخته خواهد شد.

بخش سخت‌افزار

۲-۱ لیست قطعات

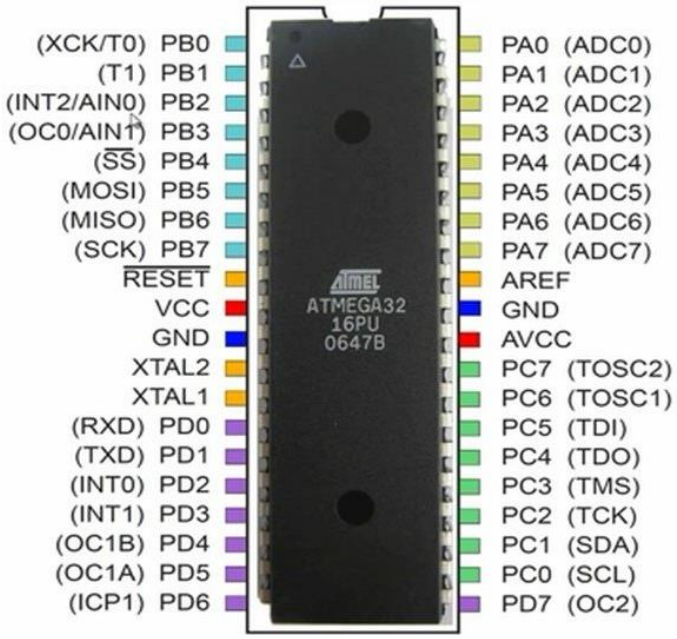
۱. برد بورد
۲. مقاومت
۳. سیم jumper
۴. میکروکنترلر ATmega32
۵. نمایشگر LCD (16x2)
۶. ماژول RFID Reader
۷. حسگر دما (NTC یا LM35)
۸. رله، قفل الکتریکی یا LED

آی سی ATmega32 میکروکنترلری با پردازنده ۸ بیتی سری AVR و با ۳۲ کیلو بایت حافظه FLASH و ۸ کانال مبدل آنالوگ به دیجیتال با دقت ۱۰ بیت و بسیاری قابلیت های دیگر که همگی در یکجا جمع شده اند میباشد.

نمایشگر متصل به میکروکنترلر دارای ۱۶ ستون و ۲ ردیف است که در هر یک از این ۳۲ بخش یک کاراکتر جای میگیرد و می‌تواند کاراکترهای لاتین و اعداد را به نمایش بگذارد.

۲-۲ بررسی DataSheet قطعات

بررسی میکروکنترلر ATmega32 :



معماری ATmega32 : بر اساس معماری Reduced Instruction Set Computing (RISC) ساخته شده است که باعث می‌شود دستورالعمل‌ها با سرعت بالا و بهینه اجرا شوند.

این معماری شامل ۳۲ رجیستر همه منظوره با دسترسی سریع است که به پردازنده اجازه می‌دهد عملیات پیچیده را با استفاده از تعداد کمتری از دستورالعمل‌ها انجام دهد.

سرعت کلاک : این میکروکنترلر می‌تواند با فرکانس‌های مختلف تا حداکثر ۱۶ مگاهرتز کار کند که باعث می‌شود بتواند وظایف پیچیده را به سرعت انجام دهد. در این پروژه از فرکانس ۸ مگاهرتز استفاده شده است.

واحد حافظه:

- **حافظه برنامه (Flash Memory) :** دارای ۳۲ کیلوبایت حافظه فلش داخلی است که برای ذخیره برنامه استفاده می‌شود. این حافظه قابل پاک کردن و برنامه‌ریزی مجدد است.

- **حافظه داده (SRAM) :** این میکروکنترلر دارای ۲ کیلوبایت حافظه SRAM است که برای ذخیره داده‌های موقت و متغیرهای برنامه در طول اجرای برنامه مورد استفاده قرار می‌گیرد.

- **حافظه (EEPROM) :** 1 کیلوبایت حافظه EEPROM داخلی که برای ذخیره داده‌هایی که باید پس از خاموش شدن سیستم نیز حفظ شوند (مانند کدهای RFID معتبر) استفاده می‌شود.

واحد ورودی/خروجی (I/O Ports) :

- **پورت‌های دیجیتال :** ATmega32 دارای ۳۲ پورت ورودی/خروجی (I/O) دیجیتال است که در چهار پورت ۸ بیتی (PORTA ، PORTB ، PORTC ، PORTD) دسته‌بندی شده‌اند. هر پورت می‌تواند به صورت مستقل به عنوان ورودی یا خروجی تنظیم شود. در این پروژه، از این پورت‌ها برای کنترل نمایشگرLCD ، خواندن دما و کنترل قفل استفاده می‌شود.

- **پورت‌های آنالوگ (ADC) :** ATmega32 دارای ۸ کانال ورودی آنالوگ (ADC) با رزولوشن ۱۰ بیتی است که برای خواندن سیگنال‌های آنالوگ و تبدیل آن‌ها به مقادیر دیجیتال استفاده می‌شود. در این پروژه، از یکی از این کانال‌ها (ADC0) برای اندازه‌گیری دمای محیط استفاده شده است.

واحد ارتباطی (Communication Units) :

- **پروتکل USART** (Universal Synchronous/Asynchronous Receiver/Transmitter) :

این واحد برای ارتباط سریال با دستگاه‌های خارجی مانند ماژول RFID به کار می‌رود. USART امکان ارسال و دریافت داده‌ها را با فرمت‌های مختلف و سرعت‌های متغیر فراهم می‌کند. در این پروژه، از USART برای خواندن کدهای RFID استفاده شده است.

- **پروتکل SPI (Serial Peripheral Interface) :** این پروتکل برای ارتباط سریع سریال با دیگر میکروکنترلرها یا دستگاه‌های خارجی مانند حافظه‌های فلش استفاده می‌شود.

- **پروتکل TWI (Two-Wire Interface/I2C)** : ارتباطی دو سیم که برای ارتباط با دستگاه‌های جانبی مانند سنسورها و حافظه‌های خارجی استفاده می‌شود.

وقفه‌ها (Interrupts) :

- **وقفه‌های خارجی ATmega32 :** دارای سه وقفه خارجی (INT0 ، INT1 ، INT2) است که امکان اجرای وظایف فوری در پاسخ به رویدادهای خاص را فراهم می‌کنند.

در این پروژه، از وقفه INT0 برای شروع فرآیند اسکن RFID استفاده شده است.

- **وقفه‌های تایمر**: این میکروکنترلر دارای سه تایمر/کانتر است که می‌توانند برای اندازه‌گیری زمان، ایجاد تأخیر و یا تولید سیگنال PWM استفاده شوند. این تایمرها می‌توانند به وقفه‌های داخلی مرتبط شده و باعث اجرای کد در زمان‌های خاص شوند.

Vector No.	Program Address	Source	Interrupt Description
1	$000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	INT2	External Interrupt Request 2
5	$008	TIMER2 COMP	Timer/Counter2 Compare Match
6	$00A	TIMER2 OVF	Timer/Counter2 Overflow
7	$00C	TIMER1 CAPT	Timer/Counter1 Capture Event
8	$00E	TIMER1 COMPA	Timer/Counter1 Compare Match A
9	$010	TIMER1 COMPB	Timer/Counter1 Compare Match B
10	$012	TIMER1 OVF	Timer/Counter1 Overflow
11	$014	TIMER0 COMP	Timer/Counter0 Compare Match
12	$016	TIMER0 OVF	Timer/Counter0 Overflow
13	$018	SPI, STC	Serial Transfer Complete
14	$01A	USART, RXC	USART, Rx Complete
15	$01C	USART, UDRE	USART Data Register Empty
16	$01E	USART, TXC	USART, Tx Complete
17	$020	ADC	ADC Conversion Complete
18	$022	EE_RDY	EEPROM Ready
19	$024	ANA_COMP	Analog Comparator
20	$026	TWI	Two-wire Serial Interface
21	$028	SPM_RDY	Store Program Memory Ready

جدول بردارهای وقفه برای ATmega32

ویژگی‌های دیگر

- **واچ‌داگ تایمر (Watchdog Timer) :** یک تایمر مستقل که در صورت بروز مشکل در برنامه، باعث ریست شدن میکروکنترلر می‌شود.

- **قابلیت Brown-out Detection :** میکروکنترلر را قادر می‌سازد تا در صورت کاهش ولتاژ تغذیه به سطحی کمتر از حد مجاز، به‌طور خودکار ریست شود.

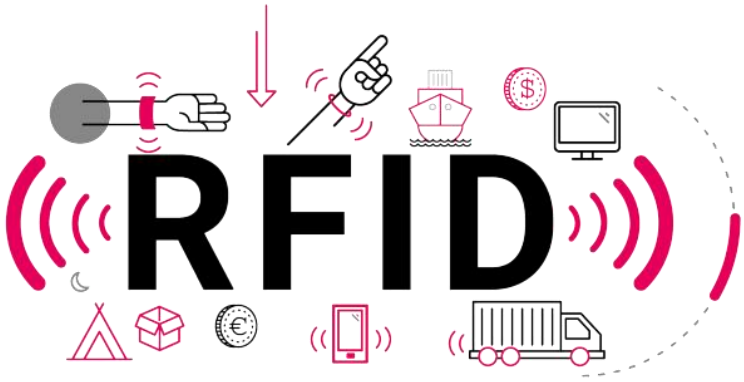
- **مدهای کم مصرف (Power-Saving Modes) :** شامل مدهای Idle، ADC Noise Reduction، Power-Down و Standby که به صرفه‌جویی در مصرف انرژی کمک می‌کنند.

بررسی نمایشگر LCD 16x2 :

LCD کاراکتری یکی اساسی ترین نمایشگرهای الکترونیکی می‌باشد که در بسیاری از مدارها کاربرد دارد. یکی از دلایل پر مصرف بودن این مدل LCD در مقابل نمایشگرهایی همچون سون سگمنت، عدم محدودیت در نمایش کاراکترها و توانایی ساختن کاراکترهای دلخواه می‌باشد.

پایه‌های شماره ۱ و ۲ مربوط به تغذیه (5v+)، پایه شماره ۳ مربوط به شدت روشنایی (کنتراست)، پایه شماره ٤ انتخاب رجیستر(رای ارسال دستور یا دیتا)، پایه شماره ۵ مربوط به وضعیت خواندن یا نوشتن، پایه شماره ٦ ورودی فعال ساز (باید به میکروکنترلر متصل شود و همیشه High باشد)، پایه شماره ۷ تا ١٤ نیز پایه های دیتا (هشت بیت) و در نهایت پایه های ۱۵ و ١٦ مربوط به نور پس‌زمینه (Backlight) میباشند.

بررسی ماژول RFID reader :



RFID (Radio Frequency Identification) یک تکنولوژی برای شناسایی اشیاء یا افراد با استفاده از امواج رادیویی است. در این سیستم‌ها، از یک کارت یا برچسب (tag) و یک دستگاه خواندن (reader) استفاده می‌شود:

- ✓ **برچسب (RFID Tag) :** شامل یک تراشه و یک آنتن است که اطلاعات منحصر به فردی (مانند یک کد عددی) را نگهداری می‌کند. این برچسب‌ها می‌توانند فعال (با منبع تغذیه داخلی) یا غیرفعال (بدون منبع تغذیه داخلی) باشند.

- ✓ **ماژول RFID Reader :** این دستگاه امواج رادیویی را ارسال کرده و پس از دریافت پاسخ از برچسب RFID ، اطلاعات تگ را خوانده و برای پردازش به میکروکنترلر ارسال می‌کند.

ویژگی‌های اصلی ماژول RFID Reader :

- **فرکانس کاری :** ماژول‌های RFID Reader معمولاً در یکی از دو باند فرکانسی 125kHz یا 13.56MHz کار می‌کنند.

- **برد خواندن :** برد عملیاتی این ماژول‌ها معمولاً بین ۲ تا ۱۰ سانتی‌متر است. (البته در ماژول‌های صنعتی این برد تا چندین متر هم میرسد)

- **فرمت داده‌ها** : خروجی به صورت رشته کاراکتری می‌باشد، ماژول RFID Reader اطلاعات کارت RFID را به صورت یک رشته کاراکتری ارسال می‌کند که معمولاً شامل یک کد منحصر به فرد ۱۲ یا ۱۰ رقمی است. این کد به عنوان شناسه کارت مورد استفاده قرار می‌گیرد و با کدهای ذخیره شده در میکروکنترلر مقایسه می‌شود. در این پروژه، میکروکنترلر با استفاده از واحد USART و با سرعت ارتباط (Baud Rate) برابر با 9600 این اطلاعات انتقال میابد.

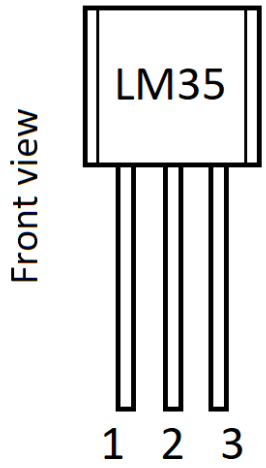
کاربرد ماژول RFID Reader در پروژه

هر کاربر با در دست داشتن یک کارت RFID که دارای یک کد منحصر به فرد است، می‌تواند با نزدیک کردن کارت به ماژول، اطلاعات ذخیره شده در کارت را به سیستم ارسال کند و میکروکنترلر پس از دریافت کد RFID از طریق ارتباط سریال، آن را با کدهای مجاز ذخیره شده در حافظه خود مقایسه می‌کند و یکی از دو حالت زیر رخ میدهد:

- ✓ کارت معتبر : اگر کد کارت با یکی از کدهای ذخیره شده در سیستم همخوانی داشته باشد، سیستم اجازه باز شدن قفل را صادر می‌کند و پس از گذشت ۵ ثانیه مجدد آن را قفل میکند.

- ✗ کارت نامعتبر : اگر کد کارت با هیچ یک از کدهای معتبر مطابقت نداشته باشد، سیستم قفل را بسته نگه می‌دارد.

بررسی حسگر دما (LM35) :



LM35 Pinout:

1. +Vs
2. Vout
3. GND

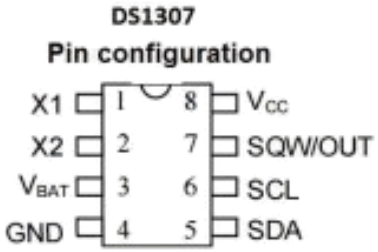
حسگر دما LM35 می‌تواند دما را در محدوده 55- تا 150+ درجه سلسیوس با دقت حدود 0.5+ درجه سلسیوس در دمای اتاق اندازه‌گیری کند، همچنین دارای نویز و تغییرات حرارتی کمی است که باعث می‌شود خروجی آن بسیار پایدار و قابل اعتماد باشد.

این حسگر توان بسیار کمی مصرف می‌کند، که آن را برای استفاده در سیستم‌های کم‌مصرف و یا باتری‌محور ایده‌آل می‌سازد.

خروجی این سنسور ولتاژی آنالوگ است که در دمای صفر درجه، ولتاژ پایه خروجی آن نیز صفر می‌شود و به ازای افزایش هر درجه سانتیگراد دمای محیط اطرافش، خروجی سنسور 10 میلی ولت افزایش پیدا می کند،. برای مثال، در دمای 25 درجه سلسیوس، خروجی حسگر 250 میلی‌ولت خواهد بود. این خروجی ولتاژ که در بیشتر کاربردها نیازی به تقویت‌کننده‌های خارجی ندارد و مستقیماً به ورودی ADC میکروکنترلر متصل می‌شود به راحتی توسط مبدل آنالوگ به دیجیتال (ADC) میکروکنترلر قابل خواندن است.

۳-۲ یک پیشنهاد برای بهبود عملکرد مدار

بررسی ماژول DS1307 :



V_{CC}	- Primary Power Supply
X1, X2	- 32.768kHz Crystal Connection
V_{BAT}	- +3V Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square Wave/Output Driver

کاربرد ماژول DS1307 محصول Dallas محاسبه و نگهداری زمان واقعی real time clock می‌باشد. ماژول ساعت دقیق DS1307 می‌تواند ساعت، دقیقه، ثانیه، روز، ماه، سال و ایام هفته را نگهداری و پردازش کند. این ماژول دارای یک باتری لیتیومی پشتیبان داخلی است که حتی در صورت قطع تغذیه مدار، امکان نگهداری اطلاعات تاریخ و زمان را به مدت چندین سال فراهم می‌کند که این ویژگی آن را به انتخابی ایده‌آل برای پروژه‌هایی تبدیل می‌کند که نیاز به ذخیره‌سازی دائمی تاریخ و زمان دارند.

همچنین بر روی این ماژول علاوه بر RTC، یک EEPROM با حافظه ۳۲ کیلوبیت تعبیه شده که از طریق رابط I2C مشترک می‌توان به آن دسترسی داشت.

همچنین می‌توان بر روی آن حسگر دما ds18b20 نیز به کار برد و با بدین ترتیب علاوه بر رفع مشکل دقیق نبودن زمان و یا از دست رفتن آن در زمان قطع برق به راحتی میتوان دما را نیز اندازه گرفت.

۲-۴ پیکربندی میکروکنترلر ATmega32

پیکربندی اولیه میکرو:

- ابتدا بایستی داخل نرم افزار CodeVisionAVR از منو File گزینه New Project را انتخاب کنید و در پنجره باز شده گزینه ATtiny, ATmega را انتخاب کنید.
- سپس در پنجره باز شده در قسمت Chip Settings نام میکروکنترلر و فرکانس کاری متناسب با پروژه را انتخاب کنید (برای مثال ATmega32 و 8MHz).
- پس از آن در قسمت Ports Settings به کانفیگ پورت‌های ورودی/خروجی بپردازید.
- سپس در قسمت External Interrupts Settings گزینه مربوط به وقفه صفر (INT0) را فعال کرده و آن را در حالت لبه پایین رونده (Falling Edge) قرار دهید.
- اکنون در قسمت USART Settings هر دو گزینه فرستنده و گیرنده را فعال کنید.
- پس از آن در قسمت ADC Settings مبدل آنالوگ به دیجیتال را فعال کنید.
- همچنین در قسمت Alphanumeric LCD Settings به کانفیگ پایه‌های نمایشگر با توجه به نیاز پروژه بپردازید (در اینجا ما یک LCD شانزده کاراکتری متصل به پورت C میکرو داریم).
- ✓ حال از منو بالای پنجره بر روی گزینه Program کلیک کرده، Generate, Save and Exit را انتخاب کنید و در آخر نامی دلخواه برای پروژه خود وارد کنید.

بخش نرم‌افزار

۳-۱ لیست کتابخانه‌ها و بررسی جزئی آنها

هر یک از این کتابخانه‌ها نقش مهمی در عملکرد کلی نرم‌افزار دارند و توابع ضروری برای مدیریت ورودی‌ها، خروجی‌ها و نمایش داده‌ها را فراهم می‌کنند:

- ☑ کتابخانه **mega32** : مربوط به تعاریف مربوط به رجیسترهای خود میکروکنترلر می‌باشد.
- ☑ کتابخانه **delay** : برای ایجاد تاخیر یا مکث در اجرای دستورات به کار می‌رود.
- ☑ کتابخانه **alcd** : توابع مربوط به راه‌اندازی و کنترل نمایشگر **LCD** را شامل می‌شود.
- ☑ کتابخانه **string** : برای کار با رشته‌ها و مقایسه آن‌ها نوشته می‌شود.
- ☑ کتابخانه **stdio** : توابع استاندارد ورودی/خروجی مانند **sprintf** از آن استفاده می‌کنند.

۲-۳ آشنایی با ساختار اصلی برنامه

زمانی که از نرم افزار CodeVisionAVR برای ساخت پروژه استفاده می‌کنید، در فایل هایی که توسط این نرم افزار ساخته می‌شود بخش‌هایی برای نوشتن کد توسط کاربر مشخص شده است.

شما میتوانید صرفا کدهای مشخص شده در هر قسمت را به این بخش‌ها اضافه کنید.

برنامه به سه بخش اصلی تقسیم می‌شود:

۱. **تنظیمات اولیه و پیکربندی :** در این بخش پورت‌ها، ADC، USART و نمایشگر LCD پیکربندی می‌شوند.

۲. **حلقه اصلی (Main Loop) :** در این حلقه، زمان و دما به صورت لحظه‌ای به‌روز شده و نمایش داده می‌شوند و همچنین، برنامه همواره آماده است تا ورودی‌های خارجی مانند اسکن کارت RFID را دریافت کند.

۳. **مدیریت وقفه‌ها (Interrupts) :** هنگامی که یک کارت RFID اسکن می‌شود، وقفه‌ای خارجی ایجاد می‌شود که برنامه را از حالت عادی خود وارد بخش ویژه‌ای برای پردازش سریع کارت می‌کند. در این بخش، کد RFID خوانده شده، با کدهای معتبر ذخیره شده در حافظه میکروکنترلر مقایسه می‌شود و سپس بر اساس نتیجه مقایسه، تصمیم‌گیری برای باز یا بسته نگه‌داشتن قفل انجام شده و این اطلاعات بر روی نمایشگر LCD نمایش داده می‌شود.

این ساختار سه‌بخشی به برنامه امکان می‌دهد که علاوه بر اجرای وظایف مداوم مانند نمایش زمان و دما، به صورت پویا به رویدادهای مهم مانند اسکن کارت RFID نیز پاسخ دهد، که این امر بهره‌وری و کارایی سیستم را به طور قابل توجهی افزایش می‌دهد.

۳-۳ کدهای پروژه

```
/*****************************************************
Project : RFID Lock with time and temp support
Date    : 10 August 2024
Author  : Nariman Ziaie
Company : Islamic Azad University Central Tehran Branch

Chip type               : ATmega32
Program type            : Application
AVR Core Clock frequency: 8.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 512
*****************************************************/

#include <mega32.h>
#include <delay.h>

// Alphanumeric LCD functions
#include <alcd.h>

// Standard Input/Output functions
#include <string.h>
#include <stdio.h>

// Declare your global variables here
unsigned int temperature;
char s[15];

#define RFID_LENGTH 12
char rfid[RFID_LENGTH + 1];  // Adjust size for RFID string +1 for null termination
volatile unsigned char rfid_index = 0;
volatile unsigned char rfid_complete = 0;

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE 0xC0

#define TIMEOUT_THRESHOLD 5000  // Timeout threshold in milliseconds

unsigned int read_adc(unsigned char adc_input) {
    ADMUX = adc_input | (ADC_VREF_TYPE & 0xff);
    delay_us(10);
    ADCSRA |= 0x40;
    while ((ADCSRA & 0x10) == 0);
    ADCSRA |= 0x10;
    return ADCW;
}

// Function to initialize USART
void usart_init(unsigned int baud) {
    unsigned int UBR_val = (500000 / baud) - 1;
    UCSRB = (1 << RXEN) | (1 << TXEN);  // Enable receiver and transmitter
    UCSRC = (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);  // 8 data bits, 1 stop bit
    UBRRL = UBR_val;  // Set baud rate
    UBRRH = (UBR_val >> 8);
}

void usart_send_char(unsigned char ch) {
        while(!(UCSRA & (1<<UDRE)));
        UDR = ch;
}

void usart_send_str(unsigned char *str) {
        while(*str!='\0')
        {
                while(!(UCSRA & (1<<UDRE)));
                UDR = *str;
                str++;
        }
}

unsigned char usart_receive() {
    while (!(UCSRA & (1 << RXC)));  // Wait for data to be received
```

```
    return UDR;  // Return received data
}

unsigned char usart_receive_string() {
        unsigned char i=0;
        char str[13];
        for(i=0;i<13;i++)
        {
                str[i] = usart_receive();     // this will take in 12 digit RFID number
        }
        return *str;
}

void process_rfid(unsigned char ch) {
    if (rfid_index < RFID_LENGTH) {
        rfid[rfid_index++] = ch;
        if (rfid_index == RFID_LENGTH) {
            rfid[RFID_LENGTH] = '\0';  // Null-terminate the string
            rfid_complete = 1;  // Mark the RFID as complete
        }
    } else {
        rfid_index = 0;  // Reset index if overflow
    }
}

void reset_rfid() {
    rfid_index = 0;
            rfid_complete = 0;
        }

        interrupt [EXT_INT0] void ext_int0_isr(void) {
            unsigned long timeout_counter = 0;
            unsigned char ch;

            lcd_clear();
            lcd_gotoxy(0, 0);
            lcd_puts("Scan RFID:");
                printf("Please Scan your RFID TAG: \n");

            reset_rfid();  // Reset RFID status

            while (timeout_counter < TIMEOUT_THRESHOLD) {
                if (UCSRA & (1 << RXC)) {  // Check if data received
                    ch = usart_receive();
                    process_rfid(ch);
                    timeout_counter = 0;  // Reset timeout if data is received
                }

                if (rfid_complete) {
                    lcd_clear();
                    // Compare RFID with predefined ID
                    if (strncmp(rfid, "160066A5EC39", RFID_LENGTH) == 0) {
                        lcd_gotoxy(0, 0);
                        lcd_puts("Valid Card");
                                    lcd_gotoxy(0,1);
                                    lcd_puts("Door unlocked");
                                    printf("The Card is VALID, Door will be unlocked. \n");
                        PORTB |= (1 << 0);  // Turn on LED or relay
                                    delay_ms(5000);
                                    PORTB &= ~(1 << 0);  // Turn off LED or relay
                    } else {
                        lcd_gotoxy(0, 0);
                        lcd_puts("Invalid Card");
                                    lcd_gotoxy(0,1);
                                    lcd_puts("Door locked");
                                    printf("The Card is INVALID, Door remains locked. \n");
                        PORTB &= ~(1 << 0);  // Turn off LED or relay
                    }
                    delay_ms(1000);  // Hold the result on the screen for 1 second
                    break;  // Exit the RFID scan mode
                }

                delay_ms(1);
                timeout_counter++;
            }

            lcd_clear();  // Clear the LCD after timeout or valid/invalid card check
        }

        void main(void) {
                // Declare your local variables here
            char seconds = 0, minutes = 0, hours = 0, A[16];

            // Initialize Ports
                // Port A initialization
                DDRA  = 0x00;
                PORTA = 0x00;
```

```
160.
161.            // Port B initialization
162.                DDRB  = 0x01;  // Set PORTB.0 as output (for LED/Relay control)
163.            PORTB = 0x00;  // Initialize PORTB to low
164.
165.                // Port C initialization
166.            DDRC = 0xF7;
167.
168.                // Initialize other ports (if needed)
169.
170.                // ADC initialization
171.                // ADC Clock frequency: 1000.000 kHz
172.                // ADC Voltage Reference: AREF pin
173.                // ADC Auto Trigger Source: ADC Stopped
174.            ADMUX = ADC_VREF_TYPE & 0xff;
175.            ADCSRA = 0x83;
176.                // SFIOR=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);
177.
178.            // Alphanumeric LCD initialization
179.                // Connections are specified in the
180.                // Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
181.                // RS - PORTC Bit 0
182.                // RD - PORTC Bit 1
183.                // EN - PORTC Bit 2
184.                // D4 - PORTC Bit 4
185.                // D5 - PORTC Bit 5
186.                // D6 - PORTC Bit 6
187.                // D7 - PORTC Bit 7
188.                // Characters/line: 16
189.            lcd_init(16);
190.            lcd_clear();
191.
192.            // Initialize USART for RFID communication (9600 baud rate)
193.                // Communication Parameters: 8 Data, 1 Stop, No Parity
194.                // USART Receiver: On
195.                // USART Transmitter: On
196.                // USART Mode: Asynchronous
197.                // USART Baud Rate: 9600
198.            usart_init(9600);
199.
200.            // Initialize external interrupt on INT0 (PD2)
201.                // INT0: On
202.                // INT0 Mode: Falling Edge
203.            GICR |= (1 << INT0);  // Enable INT0
204.            MCUCR |= (1 << ISC01);  // Trigger INT0 on falling edge
205.            GIFR |= (1 << INTF0);  // Clear INT0 interrupt flag
206.
207.            // Enable global interrupts
208.            #asm("sei");
209.
210.            lcd_gotoxy(0,0);
211.                lcd_puts("Developed by:");
212.                lcd_gotoxy(0,1);
213.                lcd_puts("Nariman Ziaie");
214.                delay_ms(1000);
215.
216.            while (1) {
217.                // Update time
218.                seconds++;
219.                if (seconds > 59) {
220.                    seconds = 0;
221.                    minutes++;
222.                }
223.                if (minutes > 59) {
224.                    minutes = 0;
225.                    hours++;
226.                }
227.                if (hours > 23) {
228.                    hours = minutes = seconds = 0;
229.                }
230.
231.                // Display time on the LCD
232.                sprintf(A, "Time: %02u:%02u:%02u ", hours, minutes, seconds);
233.                lcd_gotoxy(0, 0);
234.                lcd_puts(A);
235.
236.                // Read temperature and display it
237.                temperature = read_adc(0);
238.                sprintf(s, "Actual Temp=%u", temperature / 4);
239.                lcd_gotoxy(0, 1);
240.                lcd_puts(s);
241.
242.                delay_ms(1000);  // Update time and temp every second
243.            }
244.        }
```

جمع‌بندی

✓ توضیحات تکمیلی

این پروژه با هدف طراحی و پیاده‌سازی یک قفل دیجیتال با استفاده از RFID، نمایشگر LCD و حسگر دما انجام شد. میکروکنترلر ATmega32 با بهره‌گیری از قابلیت‌هایی چون USART، ADC و وقفه‌ها، مدیریت مؤثر سیستم را ممکن ساخت.

نتیجه، یک سیستم چندکاره و کارآمد است که به امنیت و راحتی کاربران کمک می‌کند.

جهت مشاهده کدهای برنامه و فایل شبیه‌سازی لینک بالا را اسکن نمایید