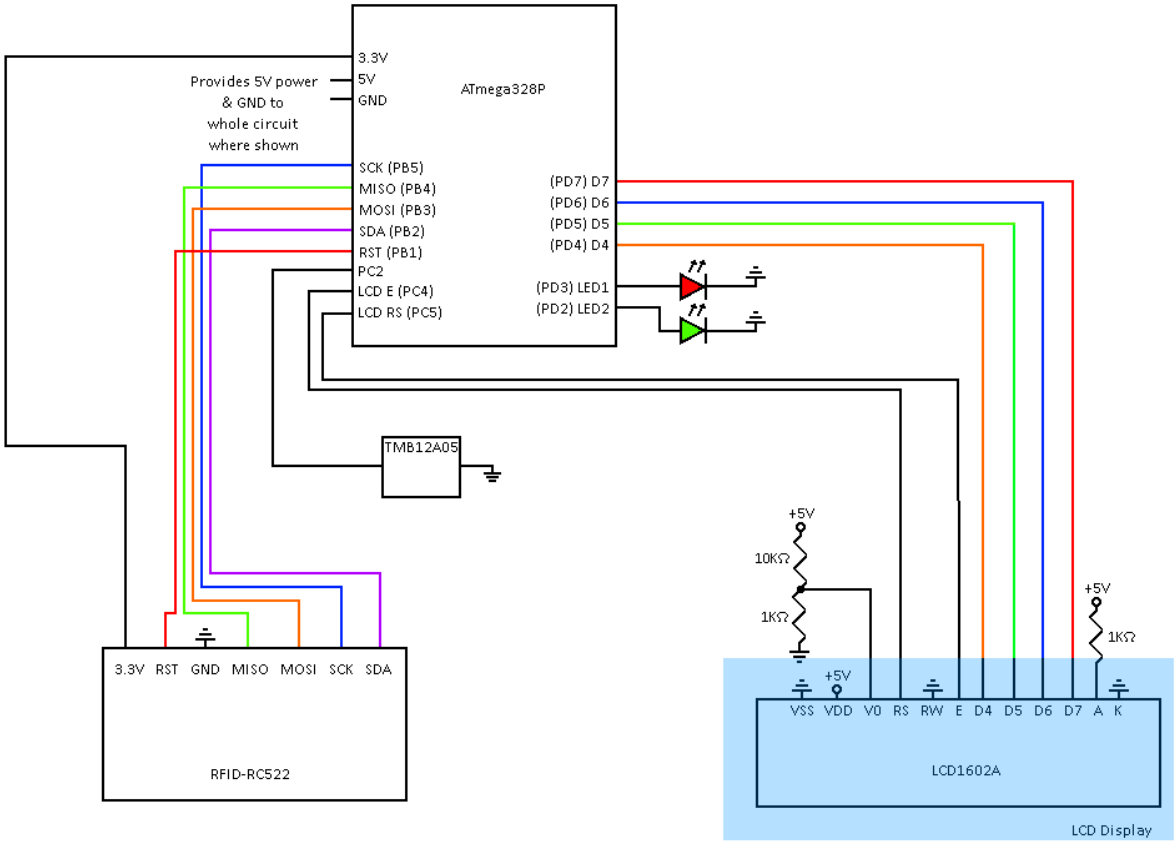
Team Members: Joseph Bartoszczyk & Brendan Suter
Professor Beichel
ECE:3360:00001
Final Project Report

1. Introduction

For our final project, we decided to implement a testing center mechanism. Our implementation relied on students scanning their student ID cards using an rc-522 RFID scanner to sign in and an LCD showing if the scan was successful and how much time they have left to take their exam. LED's were added to show the state of the system, mainly whether that 'room' in the testing center is currently being used indicated by a red and green LED. Finally, a TMB12A05 buzzer was included to sound when the RFID recognizes a scan has been attempted and repeated on/off buzzes at the end of the testing period, indicating the allowed time for testing has expired and the 'room' has once again become available. Initially we planned to include an RTC component that would display time of sign in on the serial monitor, however due to time constraints and how little that component affected the project overall, it was dropped from the final design.

How is it supposed to be used? A student scans their student ID with the RFID scanner, and the buzzer will sound. Looking at the LCD, the student will be prompted with either a "Access Granted" or "Access Denied". Our system must have the student's ID reserved to prevent unauthorized access to the testing center. If the LCD reads "Access Denied", then that student's card was not reserved by the system prior to the day of the test and will promptly be ignored. If the LCD reads "Access Granted" then the test will begin in a few seconds, giving the student time to return to their seat and begin the test. Next, the LCD switches to a screen that displays "Time Remaining:" on the top line and a countdown timer below. The countdown timer must be pre-set prior to signing in to reflect the exact time allowed for that student's test. This allows for tests of up to 3 hours in length. Once the timer expires, the buzzer will beep several times while the LED switches back and forth, indicating to the proctor and the student that the test is over. The system will continue to do so until the student signs out by scanning their student ID, at which point it plays a little jingle (just for creativity's sake) and reverts to the initial waiting state. Also, the student may sign out prior to the end of the testing period by scanning their card, but if an invalid card is scanned before the end of the testing period, the buzzer will beep at you.

2. Schematic



3. Discussion

To begin this discussion, we will first look at the hardware. There are six components worth noting, those are: our ATmega328P, our LCD from our lab kit, a TMB12A05 buzzer, two LEDs, and an RC522 RFID scanner. We decided to emulate lab 4 in our project by using the same circuit wiring between the LCD and ATmega328P. As our software was coded in C instead of assembly, this introduced some challenges which are discussed later in his section. Next, we wired the RFID-RC522 based on the pins (shown in the diagram above) to the pins sharing their description from the ARDUINO_V2.pdf provided on ICON. For example, the MISO, MOSI, SCK, and SDA pins all have dedicated pins on that pdf, so we mapped those to each. We discovered the RST pin was versatile in terms of placement and as such we placed it where it wouldn't conflict with the LCD, which required use of PD4-PD7. The other components were simple to wire, as they each required one port to feed into the component, which then feeds into a GND. One important aspect to note is the RC522 would be damaged if used with the 5V power source, so we had to use 3.3V power source on the Arduino.

The software implications for the LEDs and the buzzer were uncomplicated, as we simply had to enable their inputs when desired. As described above, this project was coded in C. Most libraries and examples online that enable the use of our RFID and LCD components were either built for the Arduino IDE or C++, however we were able to locate example code for the LCD, the link to which is included at the end of the report. This was useful in giving us an idea of how to use the LCD in C, however this code

wasn't sufficient as it was built for 8-bit mode. Therefore, we had to modify the example to limit the number of ports required to run the LCD. This was accomplished in a similar manner to lab 4, by reading in the upper and lower nibble at different times for both instructions and the strings we wanted to display. Once this was configured properly, it was just a matter of modifying the instruction and character writing instructions. This allowed us to write any character to any location on the LCD and was primarily how we were able to format our LCD to display the strings we wanted. For example, we found that using the 'set cursor' command with the address 0x00 would allow the subsequent write commands to write characters starting on the top line at the leftmost place. The address 0x40 begins the second line, note that the LCD is in a 16x2 configuration. However, a different approach was required to countdown properly.

An issue that fundamentally changed the way we implemented the countdown timer was encountered by attempting to read integers as the inputs to the countdown timer. Initially we used a set of nested for loops to countdown, using the 'set cursor' command from above to display each number in the correct location. The formatting of the countdown would be broken when the minute and second portions of the timer hit single digits, as there would be no leading zeroes on those numbers. This would violate conventional formatting, so to combat this we created two arrays that each of the for loops would iterate through, displaying the character representation of each number to the correct location. The two arrays corresponded to the tens place for minute ands seconds (which only include 0-5) and the ones place (0-9). Finally, these functions were given input parameters that allow the 'test proctor' to enter the code and preset the time on the clock to the allowed time for the 'student's test'. Initially we considered including a hardware component that allows the student to set the time allowed, but that poses some academic integrity concerns, so the 'proctor' must be the one to do it within the code. The format for our countdown timer practically supports times up to 9:59 minutes, but we decided to limit that to 3:00 at most.

One of the main challenges of the RFID component was finding a library for the component that could be implemented using C in Atmel Studio, but through extensive research we were able to find a library that fulfilled all of the requirements of this lab. The link to this library is included at the end of this report. The RC522 component is used to verify a user's ID which is set in our main file. This is done by using the included library functions to poll the RC522. The RC522 is initialized before the polling process begins, this is done by calling "mfrc522_init()". This function writes to the timer mode register to ensure that the timer resets at the reload value, which is set immediately after this. Then the ASK is set to 100% and the mode register is configured. This method then ensures that both Tx channels on the RC522 are enabled. To poll the RC522 we constantly call the "mfrc522_request" method, this method takes two arguments: a request mode, and an array to store received data. Once mfrc522_request is called it configures the bit framing of the received data and then calls the "mfrc522_to_card" method. This method in summary checks if a card has been correctly detected within a specified range of the rc522 this is done by, placing a transieve command into the FIFO data register, and then waiting to receive from the RIFD card and validating this interaction. The "mfrc522_to_card" method then returns a status, this being: CARD_FOUND, CARD_NOT_FOUND, or ERROR. If CARD_FOUND is not returned or the backbits are not 16 the status is, then set to error. The backbits being 16 ensure that two 8 bit communicated have taken place between the rc522 and a RIFD card. Otherwise, the interaction is not valid. Once the "mfrc522_request" method returns CARD_FOUND we can then attempt to get the serial of the card that has interacted with the rc522. This is done with the "mfrc_get_card_serial" method. This method is

similar to “mfrc522_rquest” but the request type is configured differently and when “mfrc522_to_card” is called it specifies two interactions taking place. Which allows the card to send it’s stored serial to the rc522. This method then parses it and stores it in the array passed into the method. Once we have the serial of the card, we check it against the serial of the authorized user. If they match, we begin the timer, display “ACCESS GRANTED” to the lcd, light up an LED and, sound the buzzer. Otherwise, we display “ACCESS DENIED” to the LCD and sound the buzzer. We repeat this process once the timer has begun to check if the user has scanned the card to sign out before the time is up. If so, we return to our main and continue polling the rc522. If not we display that the time limit has been exceeded and wait for the authorized user to scan their card to return back to our main function.

LCD have to “refresh” to continue functioning and change outputs, but the RFID must remain in a waiting state to ensure when a card get scanned the RFID doesn’t have to wait for other processes to complete before confirming that the scan was successful. However, we learned that our initialization process will allow us to input a string into the LCD and continue to display even as the RFID is waiting to scan. The beginSignin() method is a good example of this where it runs all the necessary initializations for an LCD and calls the lcdString() method to print whatever pre-defined string segment is used as an input. This method is called right at the beginning of the main while loop for the program and once executes puts the LCD into a steady-state of displaying,” Please Sign In” without running any processes on the LCD, instead it waits for the RFID to detect a scan. If a card is scanned, it check the card’s ID which gets printed as 4 hex numbers. We used a list of 4 hex numbers (which we determined were the values of our student ID cards) as a comparison to validate the input as correct. An invalid input enables the buzzer and prints to the LCD “Access Denied”, otherwise it prints “Access Granted” and calls the finalCountdown() function. This function displays “Time Remaining:” on the top line and formats the countdown clock on the bottom, as described above. Embedded within this function is a check on the RFID scanner that executes every second to determine if the student has completed the test and attempted to sign out early. Using the same code for the validation of the initial sign in, it either exits the countdown from a signout using the correct card or sounds the buzzer for an invalid sign out (I.E. the room is reserved to one and only one student).

At the end of our countdown function, once the time has expired, strings are printed to the LCD (using the same method described above) that prompt the student to sign out and the buzzer begins to pulse on and off at an increasing pace (like that of a bomb in a movie), which can only be exited with the correct card being scanned by the RFID scanner. These methods were converted to bools in the development process so returns can be used to exit the countdown where specified, which exit at function and allow the program to complete a run through of the while loop in main and loop back to the start, awaiting a new sign on. Simple pulses to the buzzer create a ‘jingle’ upon sign out before reverting to the starting condition.

4. Conclusion

In conclusion, this project was the most practical application of the skills we have acquired through taking this course. Not just from the knowledge we gained of devices used in previous labs, but the investigation of documentation and the search for useful libraries that help interface our components with our microcontroller. It also gave us more experience in Embedded C, which we found to be more practical than assembly. Finally, and most importantly, this project gave us the opportunity to go on our own to develop projects of our own design. The labs were useful in building the skills, but with the

independence this project gave us, we were finally able to test our knowledge and skill from Embedded Systems to make something of our own. Everything from this project was invaluable to our growth as Embedded Engineers, and we look forward to applying these skills in the future.

5. Source Code

```
#define F_CPU 16000000L
#define UART_BAUD 9600

#include <mfrc522.h>
#include <spi.h>
#include <util/delay.h>
#include <avr/io.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <ctype.h>
#include <stdint.h>
#include <stdbool.h>

// LCD instructions
#define lcdClear                    0b00000001
#define lcdEntryMode                0b00000110
#define lcdOff                      0b00001000
#define lcdOn                       0b00001100
#define lcdReset                    0b00110000
#define lcd4bitMode                 0b00101000
#define lcdCursor                   0b10000000

// LCD Strings
uint8_t signin[] = "Please Sign In";
uint8_t timeRemaining[]   = "Time Remaining:";
uint8_t timeList[][10] = {"0","1","2","3","4","5","6","7","8","9"};
uint8_t timeList2[][7] = {"0","1","2","3","4","5","0"};

// Prototype for Functions
bool countdownMinutes(int);
bool seconds(int);
void lcdWrite(uint8_t);
void instrWrite(uint8_t);
void lcdChar(uint8_t);
void lcdString(uint8_t *);
void lcdInit(void);
int lcdCountdown(void);
int uart_putchar(char, FILE*);
int uart_getchar(FILE*);

static FILE uart_io = FDEV_SETUP_STREAM(uart_putchar,uart_getchar,_FDEV_SETUP_RW);

int uart_putchar(char c, FILE *f){
       if(c == '\n'){
              uart_putchar('r',f);
       }
       while(!(UCSR0A & (1<<UDRE0)));
       UDR0 = c;
```

```
	return 0;
}

int uart_getchar(FILE *F){
	while(!(UCSR0A & (1<<RXC0)));
	uint8_t c = UDR0;
	return c;
}

uint8_t byte;
uint8_t str[MAX_LEN];
uint8_t granted[4] = {0x88, 0x04, 0x06, 0x52};

int main(void)
{
	stdout = stdin = &uart_io;

	UCSR0A = 1 << U2X0;
	UBRR0L = (F_CPU / (8UL * UART_BAUD)) - 1;
	UCSR0B = 1 << TXEN0 | 1 << RXEN0;
	bool check;
	DDRD |= (1 << PORTD2);
	DDRD |= (1 << PORTD3);
	DDRC |= (1 << PORTC2);
	printf("Please Sign in");
	spi_init();
	mfrc522_init();
	PORTD &= ~(1 << PORTD2);
	PORTD |= (1 << PORTD3);
	while (1)
	{
		beginSignin();
		byte = mfrc522_request(PICC_REQALL,str);
		if(byte == CARD_FOUND){
			byte = mfrc522_get_card_serial(str);
			if(byte == CARD_FOUND){
				for(int i = 0; i < 8; i++){
					printf("%02x ",str[i]);
				}
				for(int i = 0; i < 4; i++){
					check = granted[i] == str[i];
				}
				if(check){
					PORTD |= (1 << PORTD2);
					PORTD &= ~(1 << PORTD3);
					PORTC |= (1<< PORTC2);
					DDRD |= (1<<PORTD7);
					DDRD |= (1<<PORTD6);
					DDRD |= (1<<PORTD5);
					DDRD |= (1<<PORTD4);

					DDRC |= (1<<PORTC4);
					DDRC |= (1<<PORTC5);

					lcdInit();
					lcdString("ACCESS GRANTED");
					printf("Access Granted");
					lcdCountdown();
```

```
                        }
                        else{
                                printf("Access Denied");
                                DDRD |= (1<<PORTD7);
                                DDRD |= (1<<PORTD6);
                                DDRD |= (1<<PORTD5);
                                DDRD |= (1<<PORTD4);

                                DDRC |= (1<<PORTC4);
                                DDRC |= (1<<PORTC5);

                                lcdInit();
                                lcdString("ACCESS DENIED");
                                PORTC |= (1<< PORTC2);
                                _delay_ms(300);
                                PORTC &= ~(1<< PORTC2);
                                _delay_ms(300);
                        }
                        printf("\n");
                }
        }
        else{
                PORTD |= (1 << PORTD3);
                PORTD &= ~(1 << PORTD2);
        }
    }
}

int lcdCountdown(void)
{
    _delay_ms(300);
    PORTC &= ~(1<< PORTC2);
    _delay_ms(700);
    PORTD |= (1 << PORTD3);
    PORTD &= ~(1 << PORTD2);

    DDRD |= (1<<PORTD7);
    DDRD |= (1<<PORTD6);
    DDRD |= (1<<PORTD5);
    DDRD |= (1<<PORTD4);

    DDRC |= (1<<PORTC4);
    DDRC |= (1<<PORTC5);

    lcdInit();

    lcdString(timeRemaining);

    instrWrite(lcdCursor | 0x40 );
    _delay_us(80);

    int hourTime = 0;
    int minuteTime = 1;
    finalCountdown(hourTime,minuteTime);

    return 0;
}
```

```
int beginSignin(void){
	DDRD |= (1<<PORTD7);
	DDRD |= (1<<PORTD6);
	DDRD |= (1<<PORTD5);
	DDRD |= (1<<PORTD4);

	DDRC |= (1<<PORTC4);
	DDRC |= (1<<PORTC5);

	lcdInit();

	lcdString(signin);
}

void lcdInit(void)
{
	_delay_ms(100);

	PORTC &= ~(1<<PORTC5);
	PORTC &= ~(1<<PORTC4);

	lcdWrite(lcdReset);
	_delay_ms(10);

	lcdWrite(lcdReset);
	_delay_us(200);

	lcdWrite(lcdReset);
	_delay_us(200);

	lcdWrite(lcd4bitMode);
	_delay_us(80);

	instrWrite(lcd4bitMode);
	_delay_us(80);

	instrWrite(lcdOff);
	_delay_us(80);

	instrWrite(lcdClear);
	_delay_ms(4);

	instrWrite(lcdEntryMode);
	_delay_us(80);

	instrWrite(lcdOn);
	_delay_us(80);
}

void lcdString(uint8_t theString[])
{
	volatile int i = 0;
	while (theString[i] != 0){
		lcdChar(theString[i]);
		i++;
		_delay_us(80);
	}
}
```

```
void lcdChar(uint8_t theData){
	PORTC |= (1<<PORTC5);
	PORTC &= ~(1<<PORTC4);
	lcdWrite(theData);
	lcdWrite(theData << 4);
}

void instrWrite(uint8_t theInstruction){
	PORTC &= ~(1<<PORTC5);
	PORTC &= ~(1<<PORTC4);
	lcdWrite(theInstruction);
	lcdWrite(theInstruction << 4);
}

void lcdWrite(uint8_t theByte){
	PORTD &= ~(1<<PORTD7);
	if (theByte & 1<<7) PORTD |= (1<<PORTD7);

	PORTD &= ~(1<<PORTD6);
	if (theByte & 1<<6) PORTD |= (1<<PORTD6);

	PORTD &= ~(1<<PORTD5);
	if (theByte & 1<<5) PORTD |= (1<<PORTD5);

	PORTD &= ~(1<<PORTD4);
	if (theByte & 1<<4) PORTD |= (1<<PORTD4);

	PORTC |= (1<<PORTC4);
	_delay_us(1);
	PORTC &= ~(1<<PORTC4);
	_delay_us(1);
}

void finalCountdown(int hours, int minutes){
	if (hours > 3){
		hours = 3;
	}
	if (minutes > 59){
		minutes = 59;
	}

	int tenPlace = minutes/10;
	int onePlace = minutes - (tenPlace * 10);
	instrWrite(lcdCursor | 0x40 );
	_delay_us(80);
	lcdString(timeList[hours]);
	_delay_ms(1000);
	instrWrite(lcdCursor | 0x41 );
	_delay_us(80);
	lcdString(":");
	instrWrite(lcdCursor | 0x42 );
	_delay_us(80);
	lcdString(timeList2[tenPlace]);
	instrWrite(lcdCursor | 0x43 );
	_delay_us(80);
	lcdString(timeList[onePlace]);
	instrWrite(lcdCursor | 0x44 );
```

```
_delay_us(80);
lcdString(":00");
_delay_ms(1000);

if(countdownMinutes(minutes)){
	PORTD |= (1 << PORTD2);
	PORTD &= ~(1 << PORTD3);
	PORTC |= (1<< PORTC2);
	_delay_ms(300);
	PORTC &= ~(1<< PORTC2);

	_delay_ms(700);

	PORTD |= (1 << PORTD3);
	PORTD &= ~(1 << PORTD2);
	return;
}
for(int i = (hours-1); i >= 0; i--){
	instrWrite(lcdCursor | 0x40 );
	_delay_us(80);
	lcdString(timeList[i]);
	if(countdownMinutes(60)){
	return;
	}
}
DDRD |= (1<<PORTD7);
DDRD |= (1<<PORTD6);
DDRD |= (1<<PORTD5);
DDRD |= (1<<PORTD4);

DDRC |= (1<<PORTC4);
DDRC |= (1<<PORTC5);

lcdInit();

lcdString("TIME EXCEEDED!");
instrWrite(lcdCursor | 0x40);
lcdString("SSCAN ID!");
bool check;
int x = 500;
while(1){
	PORTC |= (1<< PORTC2);
	PORTD |= (1 << PORTD3);
	PORTD &= ~(1 << PORTD2);
for(int i = x; i > 0; i--){
	_delay_ms(1);
}
PORTD |= (1 << PORTD2);
PORTD &= ~(1 << PORTD3);
PORTC &= ~(1<< PORTC2);
for(int i = x; i > 0; i--){
	_delay_ms(1);
}
x = x-20;
if(x < 100){
	x = 100;
}
byte = mfrc522_request(PICC_REQALL,str);
```

```
if(byte == CARD_FOUND){
	byte = mfrc522_get_card_serial(str);
	if(byte == CARD_FOUND){
		for(int i = 0; i < 8; i++){
			printf("%02x ",str[i]);
		}
		for(int i = 0; i < 4; i++){
			check = granted[i] == str[i];
		}
	if(check){
		printf("Access Granted");
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(550);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(400);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(200);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(500);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(480);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(380);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(150);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(400);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		_delay_ms(160);
		PORTC |= (1<< PORTC2);
		_delay_ms(100);
		PORTC &= ~(1<< PORTC2);
		return;
	}
	else{
		printf("Access Denied");
	}
	printf("\n");
}
}
```

```
        }
}

bool countdownMinutes(int minutes){
        instrWrite(lcdCursor | 0x44 );
        _delay_us(80);
        lcdString(":");

        for(int j = (minutes - 1); j >= 0; j--){
        int tens = j/10;
        int ones = j - (tens * 10);
        instrWrite(lcdCursor | 0x42 );
        _delay_us(80);
        lcdString(timeList[tens]);
        instrWrite(lcdCursor | 0x43);
        _delay_us(80);
        lcdString(timeList[ones]);
        if(seconds(0x45)){
                return false;
        }
        }
        return true;
}

bool seconds(int addr){
        bool check;
        int addr2 = addr + 1;
        for(int j = 59; j >= 0; j--){
        byte = mfrc522_request(PICC_REQALL,str);
        if(byte == CARD_FOUND){
        byte = mfrc522_get_card_serial(str);
        if(byte == CARD_FOUND){
        for(int i = 0; i < 8; i++){
        printf("%02x ",str[i]);
        }
        for(int i = 0; i < 4; i++){
        check = granted[i] == str[i];
        }
        if(check){
        printf("Access Granted");
        return false;
        }
        else{
        PORTC |= (1<< PORTC2);
        _delay_ms(300);
        PORTC &= ~(1<< PORTC2);
        printf("Access Denied");
        }
        printf("\n");
        }
        }
        int tens = j/10;
        int ones = j - (tens * 10);
        instrWrite(lcdCursor | addr );
        _delay_us(80);
        lcdString(timeList2[tens]);
        instrWrite(lcdCursor | addr2);
        _delay_us(80);
```

```
        lcdString(timeList[ones]);
        _delay_ms(1000);
        }
        return true;
}
```

Libraries/example code:
https://web.alfredstate.edu/faculty/weimandn/programming/lcd/ATmega328/LCD_code_gcc_8d.html,

https://github.com/asif-mahmud/MIFARE-RFID-with-AVR,