Blackjack using Arduino

Cosmin-Nicolae Tianu

Contents

1 Introduction

The goal of this project is to use the given libraries that perform normal Arduino tasks by manipulating the board's microcontroller registers. The aim is to design a card game application based on the game of Blackjack.

2 Project Assumptions

The following assumptions were made during the development of the project:

- The hardware components, such as the Arduino board and shield, are working as intended.
- ADC input will be used for player interaction.
- The game logic will adhere to standard Blackjack rules.
- A good understanding and correct usage of the *HD44780*, *libADC* and *uart_buffer* libraries.

3 Schematic of the System

The program was developed to work on the Arduino Uno board which has a D1 Robot LCD Keypad Shield soldered to it. The system contains the following:

- An Arduino Uno board.
- An HD44780 16x2 LCD display for visual output.
- The buttons that are connected through a resistor ladder, thus outputting an analog signal that will go through Analog input 0 (A0) of the Arduino board.
- The integrated ADC will turn the said signal into a digital one that is ready to use within the program.

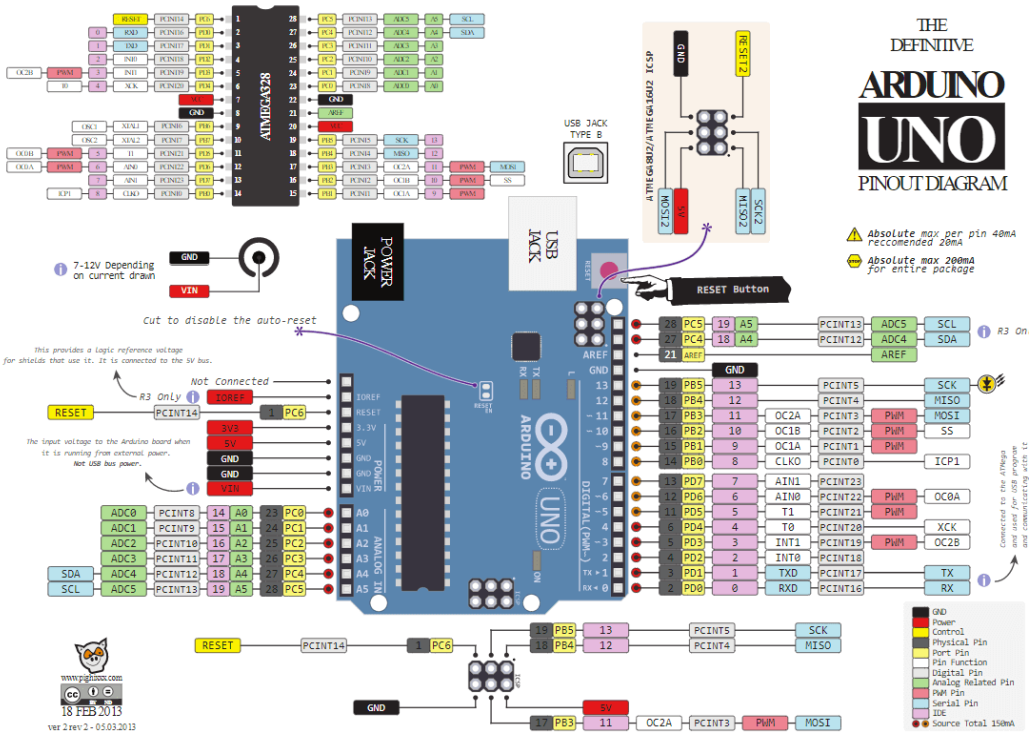


Figure 1: Arduino pin diagram

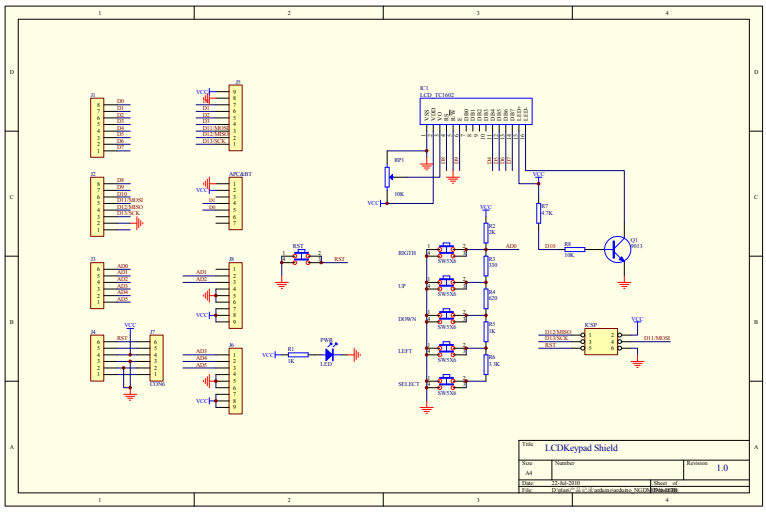


Figure 2: D1 Robot LCD Keypad Shield schematic

4 Implementation Details

4.1 System Architecture

The application implements a simple card game (Blackjack) using the following structure:

- Deck creation.
- Dealing cards randomly to the player and the dealer.
- Handling game logic: the player has to press the button **LEFT** to hit or **RIGHT** to stand on the shield of Arduino Uno. After each game the player should press the **RESET** button to be able to play another game, and finally determining the winner.
- Displaying game result on the LCD and through the UART protocol.
- Using ADC for getting the input of the player.
- The structures *Card* and *Hand* have been created for easier manipulation of the objects needed for the game. The *Card* one contains the suit, rank, card value and out, a bool that indicates if the card has been already picked out of the deck. The *Hand* structure represents mostly a group of cards and their sum as the hand value.

4.2 Code Breakdown

4.2.1 Deck Creation

The deck is initialized with 52 cards, including suits and ranks, representing a real-life deck of card.

```
void createDeck(Card* deck) {
    char suits[4][9] = {"Hearts", "Diamonds", "Clubs", "Spades"};
    char ranks[13][3] = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q",
    ...
}
```

4.2.2 Random Card Dealing

Random cards are dealt using a seed generated from ADC noise of analog pin 6 (A6) which is not connected to everything to ensure pseudo randomness.

```
int ADC_read_noise(void){
    // Setting the A6 as the source pin
    ADMUX |= (1<<MUX2) | (1<<MUX1);
    // ADSC: bit 6 - ADC - Start Conversion
    ADCSRA |= (1<<ADSC);
    while(ADCSRA & (1<<ADSC)); // when the conversion ends ADSC will turn 0

    return ADC;}
```

The reasoning for creating a random value generator is that the Arduino Uno does not have an internal timer to create a different seed, thus resulting in the same cards being dealt every game. The least significant bit of read value, which is considered the most noisy of the bits, of the ADC noise is repeatedly being added as the last bit of the seed, thus creating a random one.

```
uint16_t generateRandom() {
    uint16_t seed = 0;
    for (uint8_t i = 0; i < 16; i++) {
        seed = (seed << 1) | (ADC_read_noise() & 0x01);
    }
    return seed;
}
```

Finally, the *generateRandom* function is also used to pick a random card out of the pack, that has already not been picked by checking the card's *out* value, which is true if it has been picked in the past or false if it has not been picked.

```
Card dealCard(Card deck[]) {
    while (1) {
        int pos = generateRandom() % 52;
        if (!deck[pos].out) {
            deck[pos].out = true;
            return deck[pos];
        }
    }
}
```

4.2.3 Game Logic

The game begins by dealing two card to the dealer and to the player. The dealer's first card is unknown until the player decides to stand, at that moment, all the dealer's cards are revealed and takes the rest of the cards.

Following the start of the game, the player's turn starts, where he can *hit* or *stand*.

```
if (action == HIT) {
    Card card = dealCard(deck);
    player_hand.cards[player_hand.size++] = card;
    updateScoreAndAces(&player_hand, &card);
} else if (action == STAND) {
    while (dealer_hand.value < 17) {
        Card card = dealCard(deck);
        dealer_hand.cards[dealer_hand.size++] = card;
        updateScoreAndAces(&dealer_hand, &card);
        gameOver = true;
    }
}
```

The player can either continue to hit until he gets a good score under or equal to 21 or get a score over 21 and get busted.

```
if(player_hand.value >21){
   gameOver = true;
   strcpy(buf, "You got busted.\0");
   uart_send_array((uint8_t*) buf, strlen(buf));
   uart_send_byte('\n');
  }
```

When, and if, the player decides to stand the outcome of the game is calculated.

```
if(gameOver == true){
   if(player_hand.value > dealer_hand.value || dealer_hand.value > 21){
    strcpy(buf, "You won.     \0");
    uart_send_array((uint8_t*) buf, strlen(buf));
   }else if(player_hand.value == dealer_hand.value){
    strcpy(buf, "It's a tie.    \0");
    uart_send_array((uint8_t*) buf, strlen(buf));
   }else{
    strcpy(buf, "You lost.     \0");
    uart_send_array((uint8_t*) buf, strlen(buf));
   }
   uart_send_byte('\n');
   updateScreen = 1;
  }
```

Now the result of the game is sent through the UART protocol and to the LCD of the shield. The LCD screen is updated everytime the player hits or when the game ends.

```
if (updateScreen == 1){
   LCD_GoTo(0,0);
   if(!gameOver){
    sprintf(buf, ">\0");

    int len = strlen(buf);

    for (int i = 0; i < player_hand.size ; ++i) {
     if(player_hand.cards[i].rank[0] == '1')
      sprintf(buf + len , " 10\0");
     else
      sprintf(buf + len, " %c\0", player_hand.cards[i].rank[0]);

     len = strlen(buf);
    }
    buf[16]= '\0';
    LCD_WriteText(buf);
    _delay_ms(100);

    LCD_GoTo(0,1);
    sprintf(buf, "Score: %d\0", player_hand.value);
    _delay_ms(100);
    LCD_WriteText(buf);

   }else{

    LCD_WriteText(buf);
    _delay_ms(100);

    gameOver = false;
   }
   updateScreen = 0;
  }
```

4.2.4 Hand value calculation

The program has been developed to support the duality of any card with the rank of *Aces*, initially it has value 11, but if the value of the hand goes over

21, this card's value becomes 1 instead. This property might save the player from a bust.

```
void updateScoreAndAces(Hand* hand, Card* card) {
    hand->value += card->value;
    if (card->rank[0] == 'A') {
        hand->aces++;
    }
    while (hand->value > 21 && hand->aces > 0) {
        hand->value -= 10;
        hand->aces--;
    }
}
```

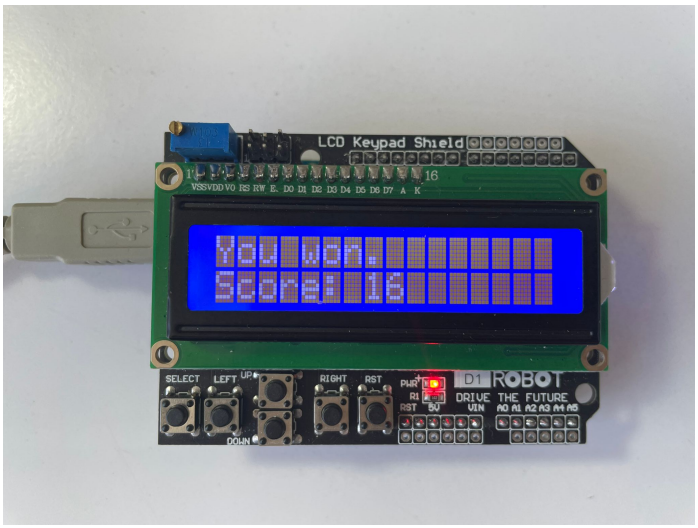
4.2.5 Player input

The input of the player is read using an ADC. The buttons on the shield used for this project are connected through a resistor ladder, thus sending an analog signal to the board. Knowing this, we use an integrated ADC to convert this value into a digital one so that we can distinguish between the buttons pressed.

```
raw = ADC_conversion();
  // Check if ADC measurement have changed
  if((raw - rawOld) < 50){
   rawOld = raw;
  }
  else {
   if (raw < 100){
    // STAND on Right
    action = STAND;
    updateScreen = 1;
   }else if (raw < 500) {
    // HIT on Left
    action = HIT;
    updateScreen = 1;
   }
  }
```

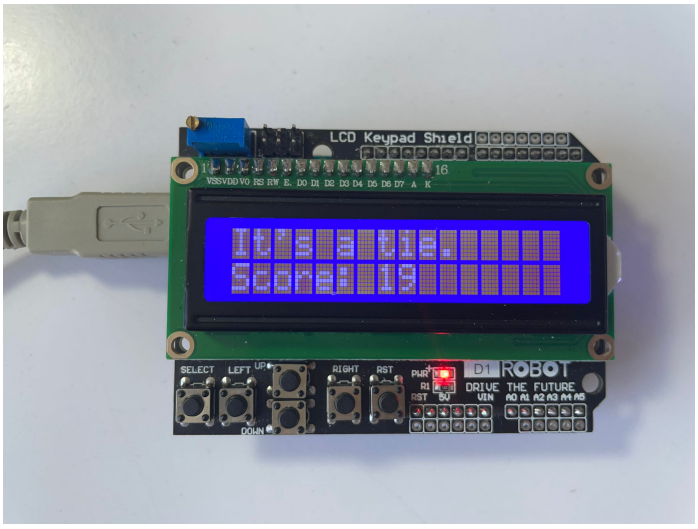
5 Pictures of Application in Operation

The player can win if he has a score of under 21 and higher than that of the dealer, or if the dealer gets busted.



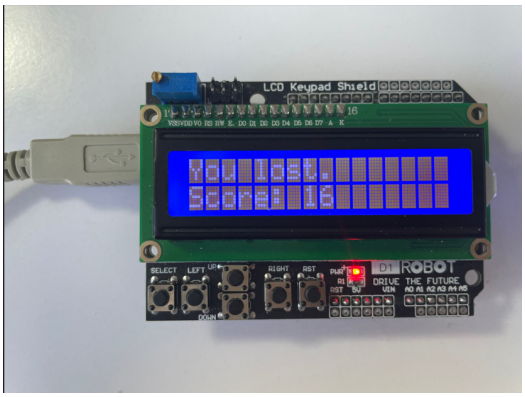

```
Dealers score: 23 and size: 4
5 of Diamonds has value 5 and out: 1
K of Spades has value 10 and out: 1
A of Spades has value 11 and out: 1
7 of Spades has value 7 and out: 1
You won.
```

The player and the dealer can also tie at the same score under 21.



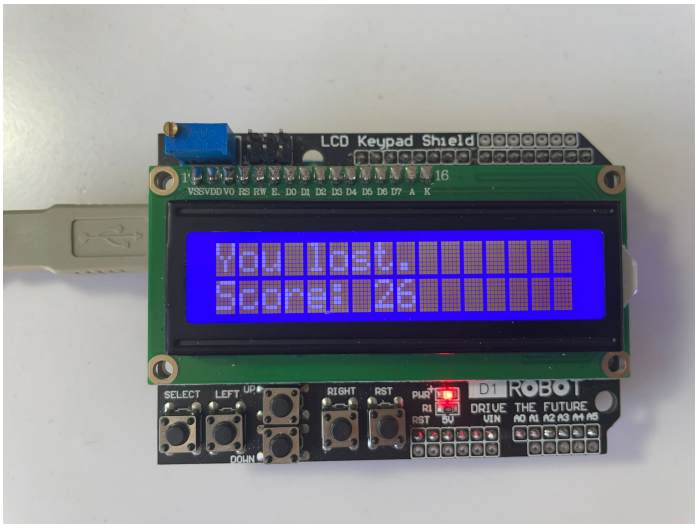
```
Dealers score: 19 and size: 3
4 of Hearts has value 4 and out: 1
5 of Hearts has value 5 and out: 1
J of Clubs has value 10 and out: 1
It's a tie.
```

The player can also lose if the his hand is of a lower value than the dealer's, both under the score of 21.




```
8 of Diamonds has value 8 and out: 1
Dealers score: 17 and size: 2
9 of Diamonds has value 9 and out: 1
8 of Diamonds has value 8 and out: 1
You lost.
```

Finally, the player gets busted, he loses, if he gets a score above 21. In this case the cards belonging to the dealer are not revealed.



```
Dealer has :
? of ? has value ? and out: ?
J of Clubs has value 10 and out: 1
You got busted.
You lost.
```

6 Source Code

The full source code for the project is available on GitHub. Visit the following link: `https://github.com/cosmintianu/BlackJack-in-Arduino`

7 Summary

The project demonstrates the effective use of Arduino and PlatformIO, together with the libraries used: `HD44780`, `libADC`, `uart_buffer`, for creating an interactive application. The implementation successfully integrates hardware and software components to create a functional card game.

Future improvements include:

- Adding more interactive features, such as a graphical interface.
- Supporting multiple players, multiple boards.
- Implementing advanced strategies for the dealer.